UNIVERSITY OF COPENHAGEN

A thesis presented to the Faculty of Humanities in partial fulfillment of the requirements for the degree

Master of Science in IT & Cognition

Evolutionary Search for Fashion Styles in the Latent Space of Generative Adversarial Networks

Imke Grabe

jbt569@alumni.ku.dk

Supervisor: Manex Agirrezabal

31. May 2021

Abstract

Fashion design is driven by the evolution of style trends. Research within fashion analysis can discover and predict fashion styles, while Generative Adversarial Networks have the ability to generate randomly conditioned clothing designs. This study proposes a framework to explore the latent space of a generative model with the purpose of generating style-coherent designs.

The penultimate embedding of an attribute model is leveraged to discover styles through clustering. Finding the latent vectors that correspond to a discovered style is approached as an evolutionary search problem. Over a number of generations, latent vectors are optimized with the objective of increasing the generated images' probability of belonging to a certain style cluster.

We find that while the system is able to generate images of maximum fitness, the designs do not necessarily correspond to the target styles in the way one would expect. Rather, the generations reveal a machine understanding of fashion style. Further investigation into the parameter setting of the genetic algorithm and the interplay between clustering model and latent space are required to make the system robust and reliable.

The findings contribute to the fields of intelligent fashion and generative deep learning by connection the discovery of fashion styles with a generative model.

Contents

List of Figures

List of Tables

1 Introduction

How we dress has for long been a fundamental part of human culture. Clothing conveys meaning beyond its material properties using its own language, that the human eye is trained to read (Hollander, 1978). It shapes our perception of beauty and the natural body, affecting concepts of gender and identity (Holmegaard, 2020). Fashion, describing clothing under constant style changes caused by trend mechanisms (Mackinney-Valentin, 2010), is deeply intertwined with the development of the technology facilitating its creation. With the recent technological advancements in generative deep learning, more artificially intelligent methods find their way into the fashion design process. However, their meaningful integration is yet to be explored.

When Acne Studio incorporated textures generated by a Generative Adversarial Network into their Fall 2020 Menswear Collection, the designs received the critique of being a ”math-crunched amalgam of all previous Acne Studio collections.”[1] That might be due to the model's nature, which learns to resemble the distribution of a training dataset. During the training, the model learns to map the random latent variables it is given to the output features, creating an entangled latent space. These output features capture the data distribution of the training set, hence the outputs might look similar to the input, with some stochasticity introduced to them. Prompted by vectors of random latent variables, the network generates new designs lying within the original data distribution. The inspection of the hidden mechanisms underlying a black box model like those of Generative Adversarial Networks, becomes impossible, which complicates the steering of the networks' output.

While the ”unpredictability of the algorithm”[2] might add interesting aspects to design processes, the characteristic also points to the main challenge in the usage, namely the control of deep generative models. Different approaches to facilitate differentiated control of the latent features of generative clothing models has been achieved by conditioning the generator with the encoding of a text description in the latent vector (Zhu et al., 2017), or by disentangling color, texture, and shape inputs through separate losses in the loss function during training (Yildirim et al., 2018).

Such approaches add to the research area of intelligent fashion, a young field concerned

[1] https://www.vogue.com/fashion-shows/fall-2020-menswear/acne-studios
[2] https://www.forbes.com/sites/brookerobertsislam/2020/09/21/why-fashion-needs-more-imagination-when-it-comes-to-using-artificial-intelligence/

with ”computer-vision-enabled fashion technology” (Cheng et al., 2021, p.1). Over the last two decades, machine learning, namely the use of algorithmic techniques that improve based on their ‘experience,’ has been broadly applied in the detection, recommendation and analysis of fashion topics (Cheng et al., 2021). Generative deep learning describes the family of machine learning models consisting of multiple layers that generate new creations (Deng and Yu, 2014). While generative intelligent fashion systems are still in their early stages, it is noticeable that most approaches in the field share an orientation towards algorithmic objectives rather than towards the subject matter of the creative field.

Returning to the concept of fashion as “a specific category of clothes (. . .) determine[d] by a process of style change” (Mackinney-Valentin, 2010, p.19), style remains the defining factor. Styles, ”used in the general sense of sort, kind, or type (of look, dress, design...)” (Mackinney-Valentin, 2010, p.19), having the ability to shape trends, are what distinguishes fashion from pure clothing.[3] Fashion styles have been the subject of several studies, mostly concerned with the classification of styles based on labelled data (Takagi et al., 2017; Hsiao and Grauman, 2017). A more recent study discovers fashion styles to model the influence of styles between cities and brands, ultimately following the goal of predicting trends (Al-Halah and Grauman, 2020). To this date, the nature of fashion styles, as defined above, has not been considered for the generation of new designs with generative deep learning. As styles seem to be the main concept driving fashion, responding to styles is relevant for generative fashion systems in order to guarantee the creation of trending outputs. This study aims to address this gap, by suggesting a system that affords the generation of designs based on fashion styles, that are discovered in an unsupervised manner.

To achieve a better control of the entangled latent space, finding the latent vectors that correspond to ’stylistic’ designs is approached as an evolutionary search problem. Coming from the field of evolutionary computation, the method draws on the concept of biological evolution. Over several generations, a population of possible solutions is improved by selecting the best ones, and further developing them by applying variation. One kind of evolutionary algorithms, genetic algorithms, have been shown to serve successful for guiding the design of fashion items

[3]With that definition in mind, ’intelligent clothing’ might be the more accurate term to describe the broad research field of intelligent fashion, as it is not only concerned with that specific category of clothing underlying style changes, but rather with clothing in general. But rather than pointing out naming issues here, this note might highlight the need to account for the core drivers of fashion, namely styles, in the research area.

in the past (Kim and Cho, 2000; Malhotra and Aggarwal, 2010; Mok et al., 2016). While genetic algorithms are best known for the optimization of bit-string representations, based on the idea of genes and chromosomes, they can also work on continuous representations, such as latent vectors. More recently, their application has been explored to guide the search of the latent space of generative models (Bontrager et al., 2018; Roziere et al., 2020; Fernandes et al., 2020; Tejeda-Ocampo et al., 2021).

Drawing on the above mentioned studies within the realm of fashion styles, generative deep learning, and the application of genetic optimization algorithms to search a design space, this work suggests a combination of the three concepts. In particular, it explores the application of a genetic algorithm to find latent vectors in the latent space of a Generative Adversarial Network, that represent certain styles, previously discovered by a clustering model, by asking:

Can a genetic algorithm guide a Generative Adversarial Network's generation of designs towards a specific fashion style?

In the following, relevant research within the creative field is presented in section 2, outlining the missing ability of generative models to respond to fashion styles. Along the way, how the concept of fashion style is approached from a computational angle is elaborated. After mapping the research field, the definition of the task underlying this work is defined in section 3, accompanied by the introduction of the dataset FashionGen used for the discovery styles and the generation of designs. To solve the task, section 4 presents the applied methods, and justifies their choice. We start with applying a feature model to extract a clothing-related feature embedding of the images, based on which a Gaussian Mixture Model is used to discover style clusters among the outfits. In parallel, a Generative Adversarial Network is trained on the same dataset. Then, both models are combined in an genetic algorithm, that optimizes a set of latent vectors by increasing their probability of belonging to a target style. Section 5 explains how the model is implemented and evaluated. The results of the final model are presented and analysed in section 6, followed by their discussion in section 7. Finally, section 8 summarizes the work and gives a short outlook into future work.

The main finding of the work shows that the proposed system can to some extend guide the exploration of the latent space towards style clusters. However, further research into the parameters underlying the search, and the interplay between latent space and clustering space

is required to obtain a robust and reliable model.

The research of this project bridges the gap between the phenomena of fashion styles and computational generative systems aiming to create desirable designs. Additionally, the findings contribute to the problem area of understanding and controlling the output of generative deep learning models. In light of the developing field of computational creativity, the approach opens up for questions regarding the role of generative models in the design process.

2 State of the Art

In the following literature review, relevant research is introduced and discussed to position this study within the field to which it aims to contribute. Starting by taking a quick glance on algorithmic intelligent applications for clothing design before (generative) deep learning, a few studies utilizing evolutionary algorithms to search pre-defined feature libraries are presented in section 2.1. Note that graphic design tools such as computer-aided design (CAD) software for sketch creation or pattern making and computer-aided manufacture (CAM) software for automated machine control are not considered as a part of the review, as they do not explicitly involve machine learning techniques into the creative process.

With the introduction of deep learning methods, new possibilities for the processing of visual features emerged, also affecting the analysis clothing features, as presented in section 2.2. Particularly, convolutional neural network structures to retrieve attributes are reviewed. Then, section 2.3 focuses on how attribute retrieval is leveraged to approach the analysis of fashion styles. Supervised and unsupervised techniques to identify styles and the corresponding understanding of the concept are taken into focus in section 2.3.1 and 2.3.2, respectively.

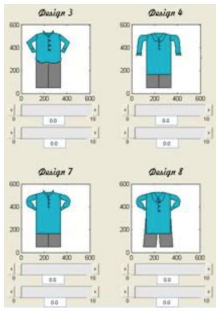
How deep learning can be applied to generate outputs with complex attributes is discussed in section 2.4 by introducing Generative Adversarial Networks, followed by applications of the concept to fashion design, introduced in section 2.4.1. Section 2.4.2 presents how the notion of style has been considered in generative approaches, illustrating the difficulty to control stylistiic features due to the entangled latent space that emerges during the training of generative networks. In connection to that, approaches of latent space disentanglement are reviewed in section 2.4.3, finalized by a review of the application of evolutionary search algorithms to navigate the latent space in section 2.4.4.

While the extensive review of the state of the art points to a collection of different approaches to control deep generative processes, it also points out a lack of considering the responsiveness to fashion styles in generative fashion technology.

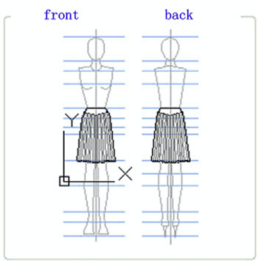
2.1 Intelligent clothing design before deep learning

Attempts to apply machine learning in computer-aided clothing design have been made long before the introduction of generative deep learning techniques. Previous attempts, that utilize

(a) Dress design by Kim and Cho (2000, p.642).



(b) Outfit design by Malhotra and Aggarwal (2010, p.212).



(c) Skirt design by Mok et al. (2013, p.1455).

Figure 1: **Clothing designs by interactive genetic algorithms.** The figures show exemplary designs of the reviewed studies. Figure 1b and 1c are extracts from the respective user interface.

evolutionary techniques to alter design suggestions interactively, are presented in the following.

Kim and Cho (2000) applied an interactive genetic algorithm to customize dresses in cooperation with a user. They split the clothing item's design broadly into the three parts of body, arm, and skirt components. The features and color of each part is represented in a joint encoding of 23 bits in total. Based on the user's selection, designs for the next generation are created through the reproduction, mutation, and crossover of the chosen designs' encoding.

In a similar manner, Mok et al. (2013) decompose skirt design into three levels, namely silhouette, key design elements, and design details. Parameters for each level can change within a pre-defined range, resulting in a "style feature library" (Mok et al., 2016, p.169) for skirts. As shown by Mok et al. (2016), the interactive fitness evaluation of designs can be supported by a design knowledge model judging the practicality of attribute combinations. More specifically, they apply fuzzy set theory to model the relationship between attributes with regard to the harmony and attractiveness of possible attribute combinations.

While Mok et al. (2013) implicitly structure the design components of skirts into high- and low-level features, Malhotra and Aggarwal's (2010) evolutionary design system makes the hierarchical structure of clothing attributes explicit in its encoding. By embedding outfit features into a hierarchy of basic and compositional parts, the authors define high-level changes of the former as explorative and more fine-grained low-level changes of the latter as exploitative. In doing so, they aim to address the problem of finding creative designs without violating design constraints, here related to 'wearability'.

Pre-defined attribute libraries, as used in the applications presented above, ensure the va-

lidity of the resulting designs. At the same time however, they limit the design space to the available set of pre-defined solutions. Hence, the design constraints underlying these above mentioned approaches are governed by the respective feature library, restricting the space of possible artifacts to a limited sample set. This becomes clearly visible when taking a look at the examples of resulting designs in figure 2. While they seem suitable to serve as a practical tool for sketching or pattern making (Mok et al., 2016), they do not come close to capturing the full imaginative possibilities that humans designers use when envisioning fashion designs.

2.2 Convolutional neural networks in computer vision

With the help of complex machine learning techniques, computers can gain a higher level of understanding of visual attributes, that might play a crucial roles in defining designs. Often, these techniques include multi-layered architectures, most prominently convolutional layers, which are presented in the following.

Even though the structure of the convolutional neural network (CNN) was already proposed in 1989 (LeCun et al.), its potential first became accessible with the emergence of fast graphics processing units (GPUs) over the last two decades. Like other computationally expensive machine learning techniques, neural networks could then be applied to problem areas involving huge amounts of data, as common for computer vision tasks. Since then, CNNs have become a popular tool for various supervised learning scenarios in image processing (Radford et al., 2016).

The model's architecture is inspired by the biological functioning of the part of the cerebral cortex in animals concerned with processing visual data, namely the visual cortex. The visual cortex distributes the processing visual data in low-, mid- and high-level visual features among its different visual areas, towards which the artificial network's structure is oriented. More specifically, functionality of the visual cortex is spread among its components ranging from the primary and secondary visual cortex (V1 and V2) processing lower-level features such as location, color, or size, over the middle visual cortex (V3 and V4) processing higher features like shape and form, to the Middle Temporal Visual Area (V5) and dorsomedial area (V6), where complex visual features such as the conception of motion or contours are processed.

This is reflected in the type and arrangement of the layers in a CNN. Like most neural networks, it consists of an input layer, hidden layers, and an output layer. The main characteristic of

a CNN however, that distinguishes it from a multi-layer perceptron, are the convolutional layers among its hidden layers. A convolution describes a neuron's activity being calculated through a convolving operation. The filters used in these operations allow the detection of patterns. More precisely, the scalar product of a filter matrix and a section of the input is calculated, resulting in a matrix of scalar products after 'moving' the filter kernel over the complete input. With the help of an activity function, the convolved input is then transformed to an output, modelling the (relative) firing of the respective neuron. The Rectified Linear Unit (ReLU) activation function is most typically used as an activity function for CNNs. Followed by a pooling layer, the output resulting from the convolutional layer is reduced to the most important information. This is commonly done by applying a max-pooling strategy, which selects the activity of the most active neuron. The weights of the kernels are updated by backpropagation (LeCun et al., 1989).

By stacking multiple of these combinations of convolutional and pooling layer together, a phenomena similar to the biological processing of visual information can be achieved. While the first layers detect rather simple features, deeper layers detect more complex visual attributes. The construct is followed by one or more fully connected layers, that compile the output of the last convolutional unit to a target output structure, for example serving the classification into labels.

The described architecture causes CNNs to be robust feature detectors with regard to arrangement and orientation of visual attributes, making it a suitable tool for detecting patterns when analyzing images. Mostly trained in a supervised manner, the network architecture has proven to serve as reliant attribute predictor.

However, with the increasing network depth comes a growing difficulty in training. To tackle that issue, He et al. introduced the Residual Neural Network (ResNet) in 2015. They suggest the creation of links between some non-neighboring layers by basically skipping other layers in the CNN. This, among other things, avoids the risk of vanishing gradient and achieves easier optimization. They demonstrate that the method outperforms other non-residual network structures in classification experiments on *ImageNet*.

ImageNet (Deng et al., 2009), consisting of 14 million images annotated with over 20,000 categories, is the most prominently used dataset for training of visual object recognition to date. Organized in a hierarchical manner like WordNet, the dataset holds categories, structured as synsets, also referring to clothing items. In 2016, Liu et al. introduced the large-scale dataset

DeepFashion, consisting of 800,000 clothing images, richly annotated with 50 categories and 1,000 attributes. Together with *DeepFashion2* (Ge et al., 2019), which extends it with richer annotations, the dataset serves as a benchmark for attribute detection tasks within intelligent fashion.

To sum up, large-scale annotated image datasets combined with the deep learning models introduced above, namely the CNN and the ResNet in particular, allow for a deeper machine 'understanding' of visual data based on the analysis of visual features. The statistical processing of huge amounts of data helps towards overcoming the technical challenge of making sense of unstructured data present in the form of images. Within the realm of intelligent fashion, deep learning tools and big data form the basis for detecting relevant attributes and patterns in clothing images. However, special to the field is the semantic gap between visual features and concepts surrounding design and style (Cheng et al., 2021). To bridge that gap, further analysis of the detected attributes and patterns is required.

2.3 Fashion styles in computer vision

Recalling the role that styles play in the evolution of fashion design, this section presents attempts that have tried to make sense of the phenomenon from different angles. Some studies approach styles in a supervised manner by utilizing pre-defined (weak and strong) style annotations of datasets, to be introduced in section 2.3.1. Others apply clustering techniques to identify styles not previously defined in an unsupervised manner. Section 2.3.2 elaborates on the latter kind.

2.3.1 Classifying pre-defined styles

In existing annotated fashion datasets, weak and strong notions of style can be noticed, to be illustrated in the following.

Weak style descriptions, like those in DeepFashion and DeepFashion2, refer to one characteristic or item. DeepFashion applies the term style in the form of an attribute group, carrying descriptors such as 'Mickey' (referring to Mickey Mouse prints) or 'Baseball' (Liu et al., 2016). DeepFashion2 refers to the style of clothing items as "color, printing, and logo" (Ge et al., 2019, p.4). Hence, the same shirt in two different colors could fall under different styles (see Ge et al., p.3, 2019).

Several datasets are labelled with strong style annotations, targeting rather cultural constructs of style that often require a combination of visual attributes in order to be defined. Table 1 provides an overview of datasets ranging from five to 67 strong styles and spanning a wide range of image per style ratio. Commonly used styles cover concepts referring to temporal events such as seasons and time periods, or social trends and constructs. Styles like 'bohemian' and 'casual' appear repeatedly among different datasets.

The reviewed strong style annotations have shown useful for the training of style classification models. To do so, Takagi et al. (2017) identify a ResNet50 (consisting of 50 layers) to be the best performing model when classifying the 14 styles available in *FashionStyle14*, which they judge to be representative for contemporary fashion trends in 2017 based on an expert opinion. Additionally, the style annotations might allow for the identification of style discriminative features, as was demonstrated by Kiapour et al. (2014) on the dataset *Hipster Wars*. Furthermore, Takahashi et al. (2020) use *FashionStyle14* (Takagi et al., 2017) for the training of a style classification model[4]. When applying the model to classify street style photos, the authors show that based on the derived style affiliations, the temporal development of trends appearing on a number of streets in Tokyo can be analysed (Takahashi et al., 2020).

Dataset	Size	Image type	Number of styles	Styles
ApparelStyle (Bossard et al., 2012)	800,000	items	25	20's, 50's, 60's, 70's, 80's, 90's, bohemian, business, casual, dandy, hip hop, hippie, mod, nerd, outdoor, preppy, punk, rock, romantic, sports, wedding, spring, summer, autumn, winter
Hipster Wars (Kiapour et al., 2014)	1,893	full body outfit	5	hipster, bohemian, pinup, preppy, goth
FashionStyle14 (Takagi et al., 2017)	13,126	full body outfit	14	Conservative, Dressy, Fairy, Feminine, Gal, Girlish, Casual, Lolita, Mode, Natural, Retro, Rock, Street
Chuanda (Liu et al., 2020a)	3,557	≤ 3 items	67	e.g. Bohemian, Loose, Casual, Business Casual

Table 1: Overview of fashion datasets with strong style annotations.

[4]Note that the authors refer to the model as a "style clustering model" (Takahashi et al., 2020, p.4). However, since the described model is concerned with classifying the pre-existing styles of *FashionStyle14* among the street style images and *not* with clustering the images to discover new styles, it is here referred to as classification.

2.3.2 Clustering models to discover styles

Taking a look at the runways around the world[5], one might notice that the pre-defined categories of the datasets presented above might not be sufficient to account for emerging trends. Instead of approaching styles as known categories, they can also be learned from images in an unsupervised manner. This section reviews studies offering different approaches to do so, focusing on the attribute and clustering models applied. Table 2 provides an overview of the model details.

Study	Dataset	Dataset size	Image representation	Number of styles	Clustering method
Hsiao and Grauman (2017)	*Hipster Wars*	1,893	PolyLDA	5	K-means
Hsiao and Grauman (2017)	*DeepFashion* (subset)	108,145	PolyLDA	> 200	K-means
Matzen et al. (2017)	*StreetStyle*	27,000	penultimate embedding	400	GMM
Al-Halah and Grauman (2020)	*GeoStyle*	7.7 million	attributes	50	GMM
Al-Halah and Grauman (2020)	*AmazonBrands* (dress subset)	41,000	attributes	20	NMF

Table 2: Overview of unsupervised approaches to cluster fashion styles.

Noticing a gap between too narrow item matching (e.g. street-to-shop retrieval (Cheng et al., 2021)) and too coarse classification of a small number of pre-defined styles (as in Hipster Wars), Hsiao and Grauman (2017) suggest to approach styles as 'latent looks' to identify style-coherent similarity between outfits. Inspired by language processing, they apply polylingual Latent Dirichlet Allocation (PolyLDA) as a topic model. The visual attributes of an outfit are understood as words in a document. Styles correspond to topics, describing the distribution of the attributes. Approaching regions (upper/lower body, hosiery, outer layer) as different languages allows for the identification of style coherency independent of item arrangement. A ResNet50 pre-trained on ImageNet is fine-tuned for localized attribute recognition, outputting 148 attribute predictions. The authors show that by using k-means clustering, they can discover styles consistent with the annotated labels of Hipster Wars and DeepFashion, with PolyLDA outperforming other attribute models as backbones.

Matzen et al. (2017) suggest a different approach to analyze the styles in dataset *StreetStyle*

[5]Market reports by fashion forecasting company Heuritech: https://www.heuritech.com/market-reports/

beyond predicted attributes. As the dataset comes with its own attribute labels (M=46), a CNN is trained on it as an attribute model. But instead of the network's final output, they choose to extract the image projection onto the 1024-dimensional feature space of the network's last convolutional layer. The deeper convolutional layers of a CNN can capture features learned by the network more specific than the output of the final fully-connected layer. Extracting the residual features captured by the penultimate layer of a ResNet-50 trained on ImageNet has proven useful for classification tasks in other domains Mahmood et al. (2020). More specifically, they prepare the embeddings through L_2 normalization followed by Principal component analysis (PCA), by projecting them onto 165 principal components that capture the 90% variance. Coming back to Matzen et al. (2017), they use a Gaussian mixture model (GMM) with 400 components to reveal the same number of style clusters based on images' retrieved penultimate embedding. Hence, styles are understood as clusters of visual themes, or mixture components of the GMM. According to the authors, the resulting clusters can be understood as a "global visual vocabulary for fashion" (Matzen et al., 2017, p.7), serving as the basis for the following temporal analysis revealing trends across cities.

In comparison to Matzen et al. (2017), Al-Halah and Grauman (2020) approach a style as a "mode in the data capturing a distribution of attributes" (Al-Halah and Grauman, 2020, p.7), equivalent to the predictions made by the output layer. The authors conduct two parallel studies to analyze the development of style trends, one on the dataset *GeoStyle*, the other on a subset of the dataset *AmazonBrands*. For GeoStyle[6], they implement the same attribute model for extracting features as Matzen et al. (2017), but extract the final output (vector of 46 attribute predictions) for clustering styles on the dataset. Similar to Matzen et al. (2017), they apply a GMM to identify styles in the dataset with the purpose of temporal analysis of trends across cities.

For the second part of their study, Al-Halah and Grauman (2020) use a subset if the AmazonBrands dataset, which consists of catalog images taken with consistent background and lighting conditions, to map trend behaviour among brands. They apply a ResNet18 (consisting of 18 layers) to predict 1000 features learned from DeepFashion. A non-negative tensor factorization (NMF) model is used to identify styles in the visual feature space, making styles

[6]GeoStyle extends the Instagram-based StreetStyle dataset with Flickr images, which do not come with attribute annotations (Mall et al., 2019).

correspond to the latent variable learned by the model (Al-Halah and Grauman, 2020). By mapping the trending of styles across the units of cities and brands, patterns of how trends influence each other can be drawn. Ultimately, this allows to make predictions and forecast trends (Al-Halah and Grauman, 2020).

In summary, deep learning applications within computer vision allow the processing of visual features going way beyond the attributes of pre-defined feature libraries discussed in section 2.1. The use of attribute models lies the basis for discovering high-level concepts of style in fashion datasets. While these styles can be pre-defined, they might also be discovered in an unattended manner, hence making style recognition adjustable to changes in styles over time. While the research presented above utilizes attribute models as a means of predicting trends, the following section 2.4 reviews how they might serve useful for fashion synthesis (Liu et al., 2020b).

2.4 Generative adversarial networks

The introduction of the *Generative Adversarial Nets*, or Networks (GANs) by Goodfellow et al. in 2014 revolutionized the creation of computer-generated content in many (creative) fields including the arts (Epstein et al., 2020), games (Volz et al., 2018), and fashion design (Cheng et al., 2021). As the name indicates, the framework consists of two neural networks competing against each other. A generator network G learns to produce 'fake' outputs to fool the discriminator network D into classifying them as real, creating the scenario of a minimax game (Goodfellow et al., 2014). In detail, the generator is given a vector of random variables, most commonly of a normal or Gaussian distribution, that it learns to map to output features that resemble the data distribution of the training set, hence are similar to input data. By mapping the random input variables from its low dimensional space to a high feature space like that of images, an entangled latent space is formed. Another way to imagine how the generator functions is to compare it to the decoder of a *Variational Auto Encoder* (VAE).

Over the last few years, extensive research has investigated how GANs can be trained more robustly. While we have seen the usefulness of CNNs for supervised tasks in section 2.2, Radford et al. (2016) suggested to integrate convolutional layers into GANs for unsupervised training. In a *Deep Convolutional GAN* (DCGAN), the convolutional units of both the generator and the discriminator model learn a hierarchical representation of visual features for images. To-

day, most GAN architectures, at least those concerned with the creation of visual output, carry convolutional units (Radford et al., 2016).

That is also the case for *progressively growing GANs* (P-GANs), a training methodology suggested by Karras et al. (2017). P-GANs adjust the size of the generator and discriminator networks' layers while increasing the resolution of images over training epochs. Optimizing training procedures is especially relevant for the generation of high-resolution images, where large datasets are required to train the generation of realistic images, such as in the domain of clothing design.

2.4.1 Application in fashion design

The generative adversarial technique has found its application in several, methodologically different fashion design scenarios, some of which are reviewed in the following.

While one challenge is the construction of model architectures for robust image generators, the second major challenge for generating fashion designs is the need for sufficiently sized datasets that are suitable for their training. To overcome that gap, Rostamzadeh et al. (2018) introduced the dataset *FashionGen*. Training a P-GAN with that dataset, the authors are able to generate new images of high-resolution (1024^2 pixels). Additionally, they show that the generated images can be conditioned by text descriptions with the help of a pre-trained text encoder.

Also Zhu et al. (2017) conditioned GANs with the text description of a desired look. With their model *FashionGAN*, the authors propose a method to adjust a person's outfit in an image. Together with human attributes retrieved from the image, the encoded text forms the design encoding. Additionally, a segmentation map of the object is created by parsing it into body and clothing components. To preserve the body shape, a first generator network generates a new segmentation map matching the arrangement of the desired outfit based on the given design encoding and the original segmentation map. The new map conditions the second generator network, which generates the texture of the new image in an individual channel for each spatial segment. This two-step generative process preserves the object's form and posture while adapting to the new design.

Another approach to impose control over design factors of GANs is to disentangle different features through separate losses in the loss function (Yildirim et al., 2018). In doing so, Yildirim

et al. (2018) teach GANs consistency with regard to color, texture, and shape of clothing items sourced from the webshop Zalando's inventory. More specifically, the factors are represented by RGB value, texture vector, and segmentation mask, respectively. Concatenated into one vector, the generator is trained to output images with these features, instead of mapping a randomly sampled vector to output features in an unsupervised manner. By means of the different losses per feature, the generator learns to distinguish between their effects when the three parts of the input vector are specifically varied during training.

While a tool like FashionGAN or the disentanglement of attributes to control certain features might find practical application in the customization of designs according to personal taste, the methods do not explicitly take the nature of styles underlying fashion evolution into account. The following section provides a quick overview of how GANs have considered the concept of style, also outside the fashion realm.

2.4.2 Style in generative models

In the context of deep learning, style is used to refer to the stylistic features of an image. The term has raised attention in connection with the idea of style transfer, describing the transformation of an image to taking on the visual style of another image (Gatys et al., 2015). Some of the most popular style transfer examples circulating the internet are the transformations of scenes adapting the painting style of a certain artist or artistic epoch. Prominent examples include a van Gogh-style applied to photos of landscapes.

Within the area of fashion, style transfer has particularly been relevant in the use case of digital try-on. With *SwapGAN*, Liu et al. (2019) also apply a segmentation map, as well as a body pose map, in an adversarial architecture with three generators and one discriminator. Their approach can be classified as style transfer, in that it aims to project the outfit of one person onto the image of another person, basically copying the items, as could be imaged in the scenario of trying on a piece of clothing. Instead of transferring an outfit from person to person, Jetchev and Bergmann (2017) approach the try-on scenario of items that are not previously worn in another image. With the *Conditional Analogy GAN* (CAGAN), they map the relation between items and how they look when being worn. Style transfer can also be applied in the creation of new designs, as shown by Jiang et al. (2021). Their fashion style generator *FashionG* learns to blend the style of an image not related to clothing (e.g. displaying a leopard, an impressionistic

painting, or a fabric texture) onto a clothing item, further conditioned by shape patterns, or the combination of two such styles (Jiang et al., 2021). In doing so, they produce item designs carrying the respective texture, hence approaching style only within this feature.

With *StyleGAN*, which is also building on P-GAN, Karras et al. (2019) aim to achieve control over the style, here approached as different features, of generated faces by interacting with the layers of GANs. With the help of a mapping network, the latent vectors are encoded as intermediate style vectors. Through Adaptive Instance Normalization (AdaIN) after each convolutional layer, different layers are targeted. That leads to style features becoming separated in an unsupervised manner, allowing more intuitive control of image generations. Yildirim et al. (2019) train a model of the StyleGAN architecture for clothing generation. Broadcasting the style vector to certain layers lets them control the color of items and the pose of the models wearing a target outfit. Additionally, they apply the model to "transfer the style and the pose of one generated outfit to another" (Yildirim et al., 2019, p.1).

In their approach underlying StyleGAN, Yildirim et al. (2019) use style to refer to an exact outfit to be transferred, similar to SwapGAN and CAGAN. Style can, however, also be understood as a wider concept. In the domain of visual arts, Elgammal et al. (2017) approach style in the form of art movements (such as Renaissance or Impressionism). Working with that definition, they suggest the *Creative Adversarial Network* (CAN). Besides being able to judge the realness of images, the network's discriminator is expanded to assess if an image belongs to a known art movement. Defining creativity as lying outside the pre-known art categories, CAN aims to create images that cannot be assigned to the known styles.

2.4.3 Latent space entanglement

While the general adversarial technique allows for the generation of complex outputs, it impedes the understanding of the relation between latent vectors and the output's feature properties. Together with approaching this issue with StyleGAN (see section 2.4.2), Karras et al. (2019) suggest different measures to approach this problem, leading towards a better interpretation of the latent space entanglement. Interpolations, describing the transition from one data point to another, can provide insight into the latent space. Analyzing how generated images respond to interpolations between latent vectors, aligned with human perception of change, can give insight into the entanglement of features, a measure referred to as *perceptual path length*.

Additionally, *linear separability* measures the ability to find hyperplanes that separate the latent vectors corresponding to output faces with and without certain attributes. Orthogonal to these hyperplanes, it is then possible to find the attribute vector corresponding to the attribute in question. Denton et al. (2019) show that by steering the interpolation in the latent space by subtracting or adding that attribute vector to latent vectors, it is possible to manipulate the respective characteristic of generated faces. That characteristic can in practice be used to produce counterfactual examples to analyze attribute classifiers. Similar to that, vector arithmetic properties arise in the latent space, as introduced by Radford et al. (2016) when extending GANs with convolutional units. Using the average of multiple faces corresponding to an attribute, and subtracting or adding that from another latent vector can e.g. add a smile to a face in a generated image.

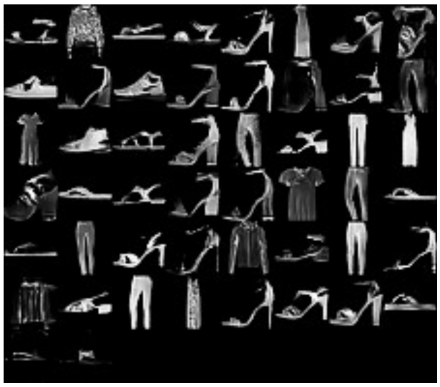
Wrapping up, while the complexity captured in a GAN's latent space makes the model a robust generator, that characteristic also makes the model hard to understand and control. Since GANs do not have an inference mechanism, training an encoder to achieve approximate inference is the only way to derive the latent vector that produced an image (Denton et al., 2019). While approaches like StyleGAN aim at disentangling features on different layers, applications like interpolation or vector arithmetic exploit the smooth characteristic of a GAN's latent space. Controlling minor features, as would be required for manipulating fine-grained details that make up a fashion style, presents an enormous effort when using the tools presented above.

2.4.4 Evolutionary techniques to explore the latent space

Another approach to steer the generation of a GAN's outputs is by applying a genetic algorithm to optimize the latent variables. The paradigm also takes advantage of the smooth property of the latent space, which it combines with evolutionary search. Drawing on the metaphor of genetic mechanisms, the technique is commonly applied as an optimization technique (Eiben et al., 2003). In particilar, the concept is built around an analogy to biological evolution. By seeing the generated images as phenotypes, a next generation of images can be created through manipulation of the latent vectors of the previous generation, hence understanding latent variables as genotypes. A population of latent vectors, also called individuals, is altered by applying variation techniques, such as recombination and mutation. Through selection based on a fitness objective, the individuals improve over a number of generations.

Roziere et al. (2020) employ *Evolutionary GANs* (EvolGANs) to improve the quality (in terms of realness) of images. By applying a quality and aesthetics estimator as a measure of fitness, the latent vector z is optimized to map to a better quality image $G(z^*)$. They demonstrate the procedure on different generative models, among others on the P-GAN trained on FashionGen. Compared to the original P-GAN approach, human raters prefer images of outfits generated with EvolGAN in 74% of the cases for FashionGen. While this technique starts from one latent vector that is chosen randomly and then to be improved, the choice does not follow a specific objective with regard to the design components of the resulting image.

A design objective to achieve could for example be the dissimilarity between a set of designs. Fernandes et al. (2020) explore the latent space of GANs, aiming to find sets of latent vectors that increase the diversity among the corresponding generated images. Their proposed genetic algorithm increases the fitness of the population by selecting the individuals with the lowest average similarity measure to each other. Uniform crossover is applied to the individuals chosen by tournament selection. The offspring is uniformly mutated, where genes are reset to random values of the latent space's distribution, which introduces new gene material to the population. The results are demonstrated on the datasets *MNIST*, *Fashion-MNIST*, and *Facity*, ranging from 18^2 to 128^2 pixels in resolution. They show that the generation of diverse objects is possible through evolutionary latent space exploration. A resulting set of dissimilar objects for Fashion-MNIST can be seen in figure 2a.



(a) Example of an evolution's final generation's set of dissimilar individuals generated by Fernandes et al.'s genetic algorithm with a GAN trained on the dataset Fashion-MNIST and the similarity NCC metric (Fernandes et al., 2020, p.13).

(b) Designs for four evolutions generated by DeepIE with a GAN trained on the dataset *UT Zappos50K shoes*. From left to right, each row displays the target image, the fittest image of the last generation, and the fittest image among all generations (Bontrager et al., 2018, p.11).

Figure 2: Designs by generative genetic algorithms.

Instead of a quality estimator or similarity measure as a function of fitness, searching the latent space to arrive at desirable output designs can also be supported by human interaction. Bontrager et al. (2018) present *Deep Interactive Evolution* (DeepIE), a paradigm for controlling the latent space of (deep) GANs with interactive evolutionary computation. While the fitness function normally displays the artificial counterpart to an organism's ability to survive in nature, here it is replaced by the users' judgement. Thus the selection process is similar to the applications presented in section 2.1. However, instead of a feature library offering different design combinations as in Mok et al. (2016) or Kim and Cho (2000), the design encoding now consists of the latent space embedding, in which a combination of latent variables represents a certain image. By choosing phenotypes with the goal of arriving at a desired design, users can guide the transformations of the generated images, thanks to the continuity of the latent space. Mutation and crossover support the exploration of the latent space. Similar to Fernandes et al. (2020), uniform crossover recombines individuals, while nonuniform mutation adds a vector of randomly drawn values to the latent variables. In the experiments conducted by (Bontrager et al., 2018), users were prompted to evolve images that resemble a specific target face or shoe. An excerpt of examples for the latter kind is shown in figure 2b. Table 3 provides a comparison of the parameters of (Bontrager et al., 2018)'s and (Fernandes et al., 2020)'s methods.

Building on (Bontrager et al., 2018)'s approach, Tejeda-Ocampo et al. (2021) improve the

model for the generation of images with more specific constraints with *StyleIE*. By using the disentanglement of secondary latent space for exploration instead, as demonstrated by Karras et al. (2019) with StyleGAN, better control over specific features can be achieved (Tejeda-Ocampo et al., 2021). Users do, however, not notice a difference between DeepIE and StyleIE when searching the latent space in a rather open and exploratory manner.

	Dissimilarity search (Fernandes et al., 2020)	DeepIE (Bontrager et al., 2018)
Fitness	Minimize average similarity	User judgement
Population Size	50	20
Number of generations	500	min. 10
Latent space dimension	100	20
Selection	Tournament selection (n=3	User selection)
Elite size	1	None
Recombination rate (per chromosome/gene)	0.7/n.a.	1/0.5
Recombination	uniform crossover	uniform crossover
Mutation rate (per chromosome/gene)	1/0.02	0.5/1
Mutation	uniform	nonuniform ($\mu = 0$, user chosen $\sigma = [0, 1]$)
New random individuals	None	2

Table 3: Parameter selection of genetic algorithms for the evolutionary exploration of the latent space.

Presenting the research field leads one to notice the following gap. As we have seen above, differentiated control of GANs applied to fashion design has been achieved through conditioning the models with text encoding, color, texture, and shape control. However, research within generative deep learning tends to come from algorithmic aspects, rather than factors underlying the world of fashion, such as the evolution of styles. While the stochastic nature utilized by generative approaches largely expands the space of possible designs for fashion synthesis in comparison to applications based on feature libraries, current generative approaches are not sensitive to fashion style trends either. Since fashion styles are the crucial characteristic that makes fashion a distinctive category of clothing (Hollander, 1978; Mackinney-Valentin, 2010), considering their role in generative design applications appears relevant to create meaningful results.

3 Definition of the Task

Reviewing the state of the art shows that emerging machine learning technology enables the analysis of fashion styles and the generation of clothing designs. But the notion of fashion styles, as defined in section 1, has not been considered in generative adversarial techniques. As fashion is defined by the evolution of style trends (Mackinney-Valentin, 2010), responding to styles is crucial for generative models that aim to output desirable designs. Aiming to make GANs respond to fashion styles leads to the definition of the following task.

3.1 Task and goal

The task of this thesis project is to apply a genetic algorithm that guides a GAN to output clothing designs corresponding to fashion styles. For that purpose, styles are approached as a subtle concept, that is neither limited to a pre-defined coarse definition such as 'Bohemian' or 'Hipster' (Kiapour et al., 2014; Takagi et al., 2017; Liu et al., 2020a), nor to one particular attribute such as color, form, or pattern only. These visual concepts can be discovered in fashion images as shown by Matzen et al. (2017) (see section 2.3.2). Due to the complexity of the problem statement, the task can be divided into three sub-tasks:

1. Apply feature extraction and clustering to discover fashion styles in a dataset.
2. Train a GAN to generate new images of the same data distribution as the dataset.
3. Combine part 1 and 2 in a genetic algorithm, that utilizes the discovered styles to guide the generation of new designs.

Hence, the desired output consists of designs adhering to the identified styles.

3.2 Dataset

The dataset chosen for this task consists of a subset of FashionGen released by Rostamzadeh et al. (2018). FashionGen consists of 293.008 high definition images of clothing, accessories, bags, and shoes and was collected from SSENSE[7], a webshop for luxury fashion and independent designers. The images are taken with the subject posing in front of a plain white background under consistent lighting conditions, and therefore ideal to serve as a (high-definition)

[7] https://www.ssense.com

training set for generative models. As this task focuses on the generation of clothing in particular, FashionGen's clothing partition, consisting of over 200,000 images with one or more clothing items worn by a model, is chosen as a dataset. The outfits in the clothing partition are depicted in different poses ranging from four to six images per outfit. Because all outfits are represented in poses 1-4 in the dataset, only images of those poses are chosen to ensure an equal representation of all instances. An overview documenting the number of images in the partition can be found in table 4.

Pose 1	49,108
Pose 2	49,069
Pose 3	49,061
Pose 4	49,010
Total	196,248

Table 4: Dataset statistics.

Figure 3 displays an example of images in pose 1-4 for some selected outfits of the dataset. While different clothing items are in focus, pose 1 most often displays a frontal view, pose 2 a side view, pose 3 captures the back of the outfit, and pose 4 depicts a full body view of the outfit. In a resolution of 256^2 pixels, the images should provide sufficient detail for the detection of style-relevant features suited for clustering, as well as the training of a generative model under limited time constraints and computing capabilities should be feasible in that quality.

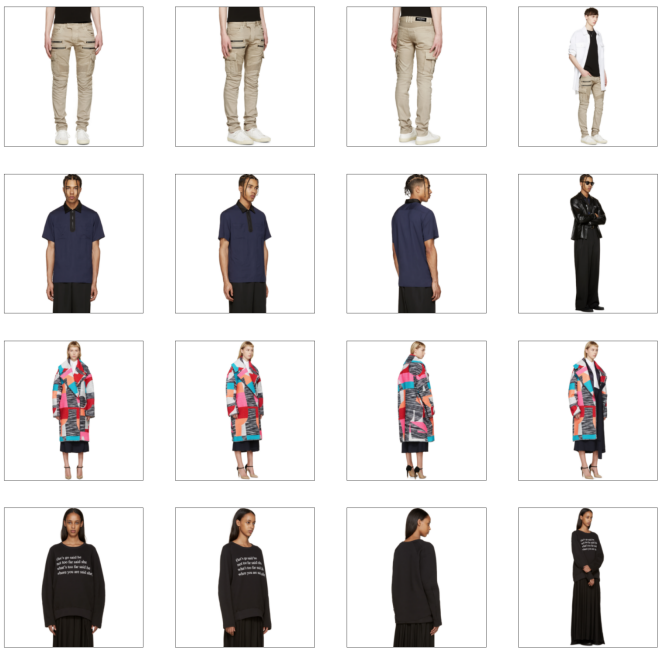


Figure 3: Instances of four outfits from FashionGen for poses 1-4 in resolution 256^2 pixels.

4 Method

This section presents the approach taken to solve the task defined in the previous section. The choice of methods is justified by explaining why they are expected to work. To create a model that can generate designs in response to fashion styles, a framework consisting of three parts is suggested. First, a style model is applied to discover styles in the dataset, presented in section 4.1. The second part consists of the training of a generative model, that learns to create new designs resembling the distribution of the dataset, as introduced in section 4.2. To guide the generative process towards output designs that correspond to the styles discovered in the first part, the final part combines components of the style model and the generative model. An evolutionary search algorithm is employed that aims to evolve the generations of the generative model with regard to a target style, as presented in section 4.3.

4.1 Style model

The goal of the first part of the framework is to discover styles in the dataset. To do that, the visual embedding learned by an attribute prediction model is leveraged to cluster outfits into styles, as explained in the following.

An attribute prediction model is applied to recognize relevant aspects in the images. Because the convolutional layers of a neural network can learn a "rich hierarchy of internal image features" (Matzen et al., 2017, p.4) (see section 2.2), a CNN serves well as an attribute extraction model for fashion images, as seen in the work by Al-Halah and Grauman (2020) and by Matzen et al. (2017), presented in section 2.3.2. Using a model trained on a clothing dataset like DeepFashion, in comparison to a non-clothing related dataset like e.g. ImageNet, adds a high-level clothing-specific sensitivity beneficial for clustering fashion styles (Matzen et al., 2017; Hsiao and Grauman, 2017; Al-Halah and Grauman, 2020).

The particular network chosen for this purpose is a ResNet (He et al., 2015), that is trained to map visual input images of DeepFashion to the prediction scores for labeled categories. DeepFashion provides the most diverse collection of clothing items currently available, making it a robust feature basis for the analysis of datasets containing a wide range of fashion items. Through transfer learning, the trained network can be applied to other datasets. The procedure has been demonstrated in Al-Halah and Grauman's (2020) implementation of a ResNet18 pretrained on DeepFashion to discover styles in a subset of AmazonBrands. As AmazonBrands also consists of catalog images with consistent lighting conditions and clean background, the setting is similar to this task. However, since Al-Halah and Grauman (2020) cluster only one category of clothes, a larger network architecture might be supportive for images of several clothing branches, hence a 50-layer ResNet[8] is chosen.

As the task is concerned with subtle styles, they are approached as a visual concept situated in between pre-defined coarse categories and low-level single-feature attribution. To be able to discover coherent style clusters representing such "visual themes" (Matzen et al., 2017, p.6), the penultimate layer of the ResNet50 can serve as a meaningful embedding space (Matzen et al., 2017). As the last layer before the linear layer outputting the prediction scores for attributes, it captures high-level features of fashion images, though not directly class-specific to Deep-

[8]The number includes the input, output, and five convolutional blocks of $1, 3 \times 3, 4 \times 3, 6 \times 3$, and 3×3 layers.

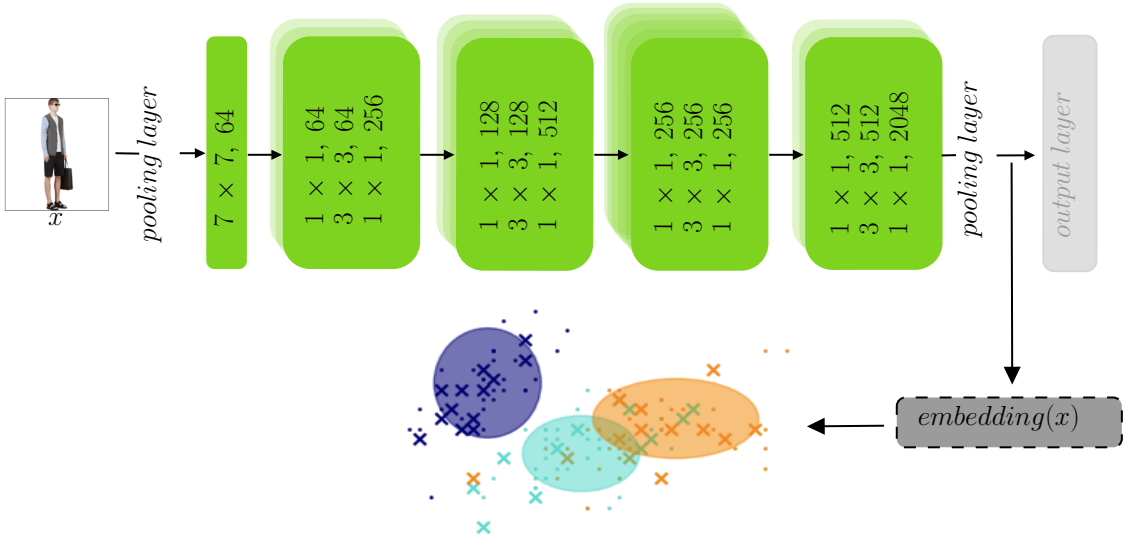


Figure 4: Style model consisting of a feature extraction and a clustering model. While the CNN's first convolutional unit only consists of one building block containing one layer, the other five units contain up to six building blocks of three layers each. As these layers are of the same size, some building blocks within a unit are 'skipped' by shortcuts. A layer contains several convolutions in the form of kernels, here notated as $r \times r, n$ with kernel size r and n channels. Since only the pre-trained layers up to the last convolutional layer are of relevance to extract the penultimate embedding, the last layer, that would normally output the predictions scores, is discarded, here depicted in light gray. The embedding for image x is extracted from the pooling layer. The embedding for all images are used to find k Gaussian mixture components[9].

Fashion. In practice, the output of the last convolutional unit, more precisely after it has been converted by to 2048-dimensional vector by the pooling layer, is extracted as the penultimate embedding $embedding(x)$ of image x, as can be seen in figure 4.

Based on the penultimate embedding, clustering is employed to find visual themes in the embedding space (Matzen et al., 2017). The projections onto the penultimate embedding space are retrieved for the complete (subset of the) dataset. According to Hsiao and Grauman (2017), the kind of clustering algorithm has a negligible effect when clustering fashion styles. Hence, they choose k-means clustering as a basic clustering algorithm sufficient for finding more than 200 styles in a dataset of over 100,000 images. However, k-means only provides the distances of data points to a cluster centroid. But the distance measure is not enough for judging the affiliation of an image to a cluster, which is required as a measure of fitness for the genetic algorithm, as will explained in section 4.3. That is because adjusting an image by reducing

[9]Image of the GMM from https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.htmlsphx-glr-auto-examples-mixture-plot-gmm-covariances-py

its distance to a cluster centroid does not guarantee that the image is not, at the same time, 'moving' closer to another centroid, too. Hence, a probabilistic clustering model is needed. By increasing an image's posterior probability of belonging to a cluster components, its probability of belonging to other clusters is reduced. In other words, as the posterior probability adds up do 1 among all clusters, maximizing the likelihood of belonging to one cluster reduces the likelihood of belonging to the others. A Gaussian Mixture Model (GMM) provides such a probability distribution, by finding a mixture of Gaussian probability distributions that represent a dataset. Based on Matzen et al. (2017), who applied a GMM with a diagonal covariance matrix to identify 400 coherent style clusters out of over 27.000 images projected onto the penultimate feature space (see section 2.3.2), the method should serve suitable for the discovery of a large number of styles in the dataset at hand. After identifying a number of coherent mixture components, also referred to as style clusters, in the dataset, an image's posterior probability of belonging to a to a component k, depicts how well it fits the into the style cluster as p_k.

4.2 Generative model

Parallel to discovering styles in the dataset, a GAN is trained to create images with the dataset. By competing against a discriminator network, a generator network learns how to map an input vector z, randomly sampled from a standard normal distribution $\mathcal{N}(\mu = 0, \sigma = 1)$, to output images resembling the real data distribution in the dataset. In the training, the generator is optimized with regard to the discriminator D's ability of assessing the generated samples being real. D itself is trained to predict the origin of real and generated images. For this work, the P-GAN architecture and training procedure is utilized, which applies the Wasserstein GAN loss with gradient penalty (WGAN-PG) . Hence, the goal of the training procedure is to minimize the following function with respect to G, and maximize it with respect to D:

$$\min_G \max_D E_{x\sim p_{data}}[D(x)] - E_{z\sim p_z}[D(G(z))] + \lambda E_{\hat{x}\sim p(\hat{x})}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{1}$$

The former two expectancies of the equation describe D's ability to judge image x to be from the real distribution and D's ability to estimate that a generated sample $G(z)$ is not real. The latter addend depicts the gradient penalty added for exceeding the gradient norm of 1.

As indicated by its name, a GAN is trained in an adversarial manner, where the generator network improves with the goal of fooling the discriminator network. The training procedure

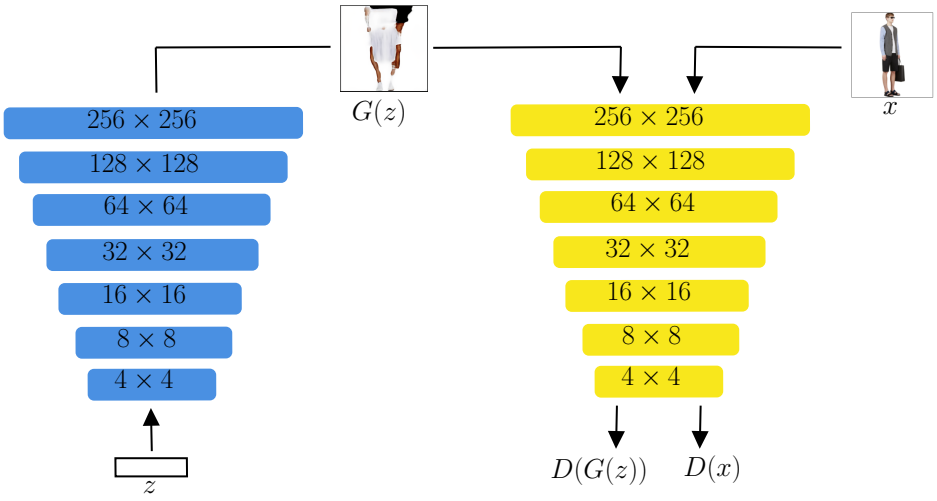


Figure 5: Generative model. The GAN consists of the generator network G, here depicted in blue, and the discriminator network D, depicted in yellow.

takes place as explained in the following. During the training, D is prompted to discriminate a sample x coming from the training set, as depicted in figure 5. Its ability to do so is evaluated by the loss $L(D(x), y_{real})$, namely the difference between the probability that D predicts x to be real and x's actual label $y_{real} = 1$. Secondly, a vector z is initialized randomly, based on which the generator network G creates a generated image $G(z)$. Now, D is tasked to judge the likeliness of $G(z)$ coming from the real dataset instead of being a generated sample. The estimated probability $D(G(z))$ is used to calculate the loss $L(D(G(z), y_{fake}))$ with regards to the actual label of fake samples, $y_{fake} = 0$. D is optimized with the total of both losses, $L(D(x), y_{real}) + L(D(G(z)), y_{fake})$. The discriminator loss will likely be low at the beginning, when the generator has not yet learned to output realistic images. Hence, the discriminator can easily learn to see the difference between real and generated samples. As the generator improves on its task, D will have a harder time identifying the images' origin, which is the ultimate goal underlying the training procedure.

After training the discriminator, a new instance of z is used for a second generation $G(z)$. Again, $D(G(z))$ describes the estimated probability of coming from the actual population. This time however, its prediction is measured against y_{real}, as the generator aims to improve to generate outputs indistinguishable from real samples. The loss $L(D(G(z)), y_{real})$ serves as a means to optimize the generator network. While this loss is high in the beginning, it should increase as the generator becomes better at fooling the discriminator.

During the training, the generator model learns to map the random latent variables it is given to the output features in the generated images, creating an entangled latent space. While this technique allows for the generation of complex outputs, it impedes the understanding of the relation between latent vectors and properties of the output.

4.3 Evolutionary search

Until here, a tool for the discovery of styles utilizing clustering was proposed, as well as a generative model that can create random designs resembling the distribution of the dataset. However, due to the entangled latent space of the model, retrieving images of a certain style remains challenging. To make the generative model output designs of a style that is discovered by the clustering model, parts of the two models are combined in an evolutionary search algorithm, as illustrated in figure 6. In line with evolutionary computation, the trained generator of the GAN is understood as a genotype-to-phenotype mapping, where the latent encoding, interpreted as the genotype, governs the appearance of the output designs, the phenotype. Changing a latent vector results in a change in the image. Recall that due to the smooth characteristic of the latent space, as explained in section 2.4.3, small changes in latent vectors lead to small adjustments in the corresponding visual features. Larger adjustments affect the image more.

Over several generations, the proposed genetic algorithm alters a population of N_{pop} individuals, or latent vectors, originally initialized from the distribution as the latent variables. Latent vectors are selected for the next generation by evaluating their probability of falling into the targeted style cluster. The procedure follows the goal of arriving at a set of latent vectors that represents designs of the desired style. While the following sections explain the details, algorithm 45 provides the pseudo code for the procedure. Figure 6 gives an overview of the midel model.

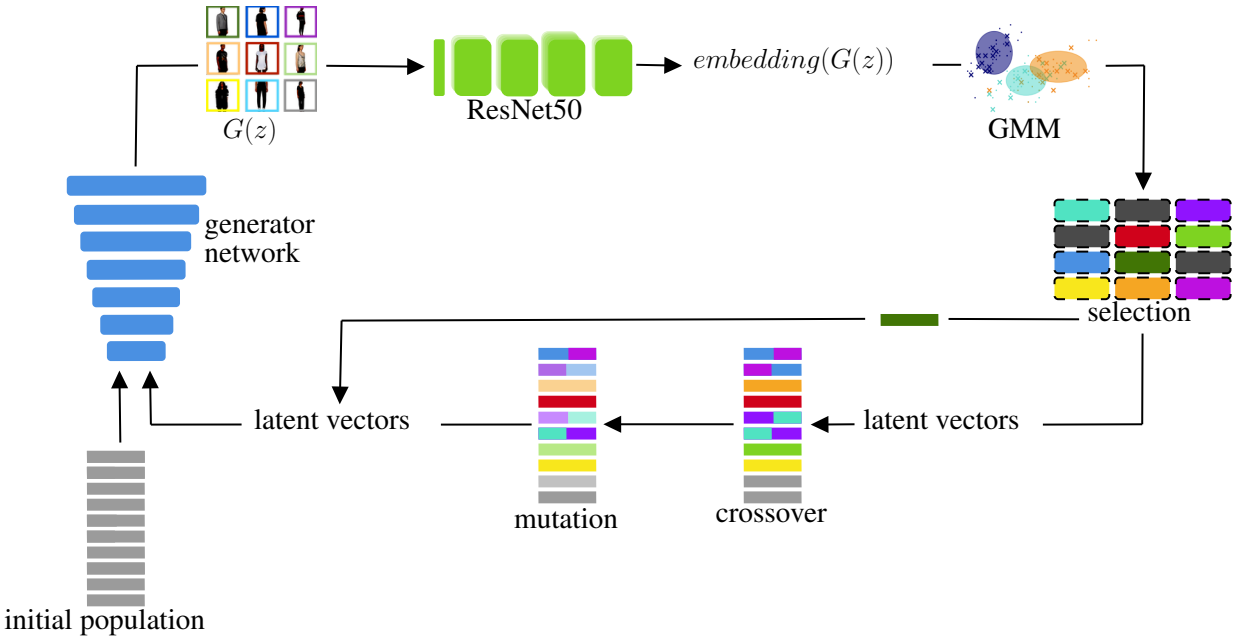


Figure 6: Model of the genetic algorithm for searching the generative model's latent space with the help of the clustering model. This figure illustrates the evolutionary search for style coherent designs. First, the trained generator network G creates images based on a randomly initilized population. Next, the generated images are represented as their penultimate embedding and projected onto the cluster space. Then, the fittest individuals are selected by based on their posterior probability of belonging to the target cluster. The fittest individual, depicted in dark green, is preserved for the next generation (elitism). The others are recombined and two new individuals, depicted in grey, are added. Last, the resulting individuals are mutated. The resulting latent vectors lay the basis for the next generation of images.

Representation. The problem is represented as follows. A latent variable v of a latent vector of length l corresponds to one out of l genes in a chromosome (Eiben et al., 2003). Originating from a continuous distribution of real values (or rather floating-points in a computer implementation) (Eiben et al., 2003), a generated design is represented by its latent vector

$$z = \langle v_1, ..., v_l \rangle \text{ with } x \in R \tag{2}$$

where l is the length of a latent vector z.

As has been shown in section 2.1, encoding fashion designs as genetic representations can capture relevant criteria for the design process. But the chromosome encoding based on a discrete distribution of parameters, as in the feature libraries by Mok et al. (2013); Kim and Cho (2000), provides a limited number of combinations to choose from. A continuous embedding

space like the one of a GAN can specify more fine-grained design quantities than can be captured in between discrete steps. This allows for the representation of candidate solutions with high sensitivity, which is required to meet the subtle style definitions that are to be generated.

Transformation. A variable of a latent vector, or gene, can take on alternative values, also called 'allele' values, that lie within a lower bound L_i and an upper bound U_i depending on the distribution underlying the latent space (Eiben et al., 2003). Hence, the transformation of a latent vector z can be described as the following:

$$\langle v_1, ..., v_l \rangle \rightarrow \langle v'_1, ..., x'_l \rangle \text{ where } v_i, v'_i \in [L_i, U_i] \tag{3}$$

While a latent vector z prompting the design creation by a generator network is normally sampled randomly from the underlying distribution, here $z \sim \mathcal{N}(\mu = 0, \sigma = 1)$ (see section 4.2), the genetic algorithm aims to transform z towards z^* with $G(z^*)$ representing a design of the targeted style. This transformation is guided by an objective, defined in the following.

Fitness and selection. To arrive at the desired goal, genotypes are evaluated against a fitness measure that judges their fitness with regard to an objective. Remember from section 4.1, that the an image's posterior probability of belonging to a style component k is p_k. Following the goal of resembling a certain target style t, the fitness criterion $\mathcal{F}_t$ is defined by an individual's posterior probability of belonging to the respective target style cluster, referred to as f_t. Applying the measure brought forward by the GMM, $f_t = p_t$.

To define the fitness of a latent vector, three steps are required. First, image $G(z)$ is generated. Then, the generated image is projected onto the penultimate embedding space, retrieving $embedding((G(z))$. Finally, the retrieved $embedding((G(z))$ is projected onto the cluster space, where the probability of belonging to target cluster t is extracted as p_t, representing the fitness criterion f_t. That defines the fitness f of an individual, or latent vector z of belonging to a target cluster t as

$$f_t(z) = p_t(embedding(G(z))). \tag{4}$$

For the generated images to fit better into a style cluster, the fitness $\mathcal{F}_t$ needs to be maximized to obtain a latent vector z^* of target style t:

$$z^* = \arg\max_z \mathcal{F}_t(z) \tag{5}$$

By maximizing the fitness function, the population of latent vectors should improve towards the defined requirement through selection and variation.

At the beginning of each generation, a copy of the best N_{elite} individuals is taken aside, to save the fittest individual(s) from crossover or mutation. Inheriting the copy to the next round guarantees that the fittest individual over generations is preserved. N_{pop} individuals are selected by conducting N_{pop} tournaments, where N_{ts} randomly chosen individuals compete against each other based on their fitness. To the population resulting from the tournament selection, two kinds of variation operations, recombination and mutation, are applied.

Recombination. The first way to introduce variation to the population is by recombination, commonly called crossover when only two individuals are recombined with each other. In a crossover of the proposed genetic algorithm, two latent vectors (understood as parents), here called a and b, are recombined to produce new individuals (understood as children). As proven successful in Bontrager et al. (2018) and Fernandes et al. (2020), *uniform crossover* is applied to generate new offspring. Hereby, each attribute i of child is randomly chosen from either parent a or parent b:

$$\hat{a}_i = \alpha a_i + (1-\alpha)b_i \text{ and } \hat{b}_i = \alpha b_i + (1-\alpha)a_i \text{ with } \alpha = Bernoulli(0.5) \tag{6}$$

If recombined, an individual is replaced by its child. The chance for an individual of the population to participate in recombination is defined by the crossover rate p_{cx}. Because high crossover rates are recommended, they are chosen as 0.7 (Fernandes et al., 2020) and 0.9 to ensure the continuing development of fit individuals (Hassanat et al., 2019). To introduce new gene material, N_{new} new random vectors are added to the population in every generation in addition to the crossover.

Mutation. The other applied variation technique is the mutation of individuals following the crossover of a population. Fernandes et al. (2020) randomly reset gene values with the goal of decreasing similarity in the population. Here, following the objective of arriving at a target cluster requires a mutation strategy that allows for vectors to move closer towards a target cluster. Therefore, instead of random replacing, noise can be added to the latent variables. As arriving at a target style is similar resembling a certain image, nonuniform mutation as in DeepIE is applied. While σ is controlled by the users in DeepIE, here, it is 1. Some values drawn from the same distribution as the latent variables are added to each variable of a latent

vector.

With a probability of 0.5, a variable of an individual, or latent vector, z is mutated by adding some $noise$:

$$z_i = z_i + noise \sim \mathcal{N}(\mu = 0, \sigma = 1) \tag{7}$$

The chance for an individual of the population to be mutated is defined by the mutation rate p_{mut}. While the mutation rate of a genetic algorithm is normally set to a few percent (Hassanat et al., 2019), both low (0.2) and high (0.5) mutation rates are considered, because in initial runs, low diversity of the population could be observed.

In accordance with the tuning method presented in the experimental design in section 5, different tournament sizes ($N_{ts} = \{3, 6\}$) and population sizes ($N_{pop} = \{100, 200\}$) adhering to the commonly used parameter choices are considered (Eiben et al., 2003; Hassanat et al., 2019). The number of generations is set to $N_{gen} = 500$ as a compromise of running time and complexity of the problem.

Algorithm 1: Deep Fashion Style Evolution

Result: $G(z^*)$ closest to style cluster centroids

Parameters:
z: latent vector of length n where $z_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$
N_{pop}: population size
N_{gen}: number of generations
N_{elite}: elite size
N_{new}: number of new individuals
N_{ts}: tournament size
p_{cx}: crossover rate per individual
p_{mut}: mutation rate per individual
t: target cluster

Function `fitness` (*individual, t*) **:**
  $embedding_z = penultimate(G(z))$
  $p_z = posteriorprobability[t]$
  return p_z

Function `selection` (*population, N*) **:**
  for $i \leftarrow 1$ *to* N **do**
    $best_i \leftarrow$ winner of tournament i with N_{ts} individuals with `fitness` (*individual*)
  end
  return $best_1, ..., best_N$

Function `crossover` (*a, b*) **:**
  for *gene in* z **do**
    $a_i = \alpha a_i + (1 - \alpha) b_i$ for $\alpha \sim Bernoulli(0.5)$
    $b_i = \alpha b_i + (1 - \alpha) a_i$ for $\alpha \sim Bernoulli(0.5)$
  end
  return $individual_a, individual_b$

Function `mutation` (*individual*) **:**
  $noise \leftarrow$ vector of length n where $noise_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$
  return $individual + noise$

$population \leftarrow N_{pop} \times individuals$ of z
for $g \leftarrow$ *to* $N_{generations}$ **do**
  $elite = N_{elite}$ best of population
  $population =$ `selection` (*population,* N_{pop})
  for $individual_a, individual_b$ *in population pairs)* **do**
    if *random* < p_{cx} **then**
      $individual_a, individual_b =$ `crossover` $(individual_a, individual_b)$
    end
  end
  $population = population + N_{new} \times individuals$ of z
  for $individual$ *in* $population$ **do**
    if $random < p_{ation}$ **then**
      $individual =$ individual
    end
  end
  $population = population + elite$
end

5 Experimental Design

The experimental setup is designed to analyze the ability of the proposed model to generate designs adhering to a discovered style, which is the goal underlying this task. This section describes the implementation of the three major parts of the model and the evaluation of the complete framework.

5.1 Clustering styles

As proposed above, to prepare images for the discovery of meaningful style clusters, they are converted into visual embeddings. The open-source toolbox MMFashion[10] provides models pre-trained on DeepFashion. Structured into *backbone* and *head* classes, the former is concerned with transforming images to feature maps, to be applied in specific tasks of the latter part (Liu et al., 2020b). For this task, the backbone of the attribute prediction model ResNet50 is chosen, pre-trained on the task of predicting 1000 attributes with 'global pooling', that extracts global features (compared to features localized around landmarks). As we focus on the penultimate layer of the network, the output layer is dropped, and the global pooling is replaced by an average pooling layer, that converts the last convolutional unit's output into a 2048-dimensional vector. The model is set to evaluation mode, meaning it does not update its weights when passing images through it. Before extracting the feature vector for images of the model, the images are normalized and scaled to mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] as expected by PyTorch models. While feeding samples through the network, the output of the average pooling is captured as the resulting visual embedding.

The subset of pose 4 of the clothing partition is chosen as a basis for clustering, as its size is more suitable for clustering, and every item of the training set is represented once in the subset, often as part of an outfit. The visual embedding is extracted for all images of the subset. The embedding is scaled to zero mean before principal component analysis (PCA) is conducted to project the embedding onto their 200 principal components capturing to 90% of the embedding's variance. We experiment with different numbers of components of the GMM to arrive at a set of k coherent style clusters, similar to Al-Halah and Grauman (2020) and Matzen et al. (2017).

[10] Open-source toolbox for visual fashion analysis based on PyTorch, developed by the Multimedia Lab, Chinese University of Hong Kong: https://github.com/open-mmlab/mmfashion

5.2 Training the GAN

The P-GAN available in the Pytorch GAN Zoo[11] is chosen for the training of the generative model. The training procedure is not modified. The layers of the GAN were grown progressively, from 4^2 pixels in the first epoch to 256^2 pixels in the seventh epoch. As common for deep learning models, the training is conducted in epochs, referring to the event of (forward and backward) passing the whole training set through the model completely once. Due to the large size of training sets, epochs are further divided into batches of smaller sizes, describing the number of instances passed at one time. A batch size of 16 was used, but had to be decreased to 8 for the last epoch, due to memory limitations. The images were randomly sampled from the input data. The GAN was trained on a *NVIDIA Tesla V100-SXM2-16GB* GPU[12] for seven days. Table 5 provides an overview of the training parameters. The resulting trained generator model creates images given a vector of 512 random variables.

Parameter	Setting
Optimizer	Adam
Activation function	leakyRELU
Learning rate	Equalized learning rate
Batch size	16 (8)
Loss function	WGAN-GP
Noise distribution	$\mathcal{N}(\mu = 0, \sigma = 1)$

Table 5: GAN training parameters.

5.3 Evolving stylistic designs

To combine the style model and the generative model in the genetic algorithm, the evolutionary computation framework DEAP[13] is used for the implementation of algorithm 45.

Finding the right parameter setting for a genetic algorithm depicts an acknowledged problem within the field of evolutionary computation (Eiben et al., 2003). This is due to the interdependence of parameters as well as problem dependency. Hence, decisions are often based on conventions or ad-hoc decisions (Eiben et al., 2003). To approach that difficulty, the testing methodology utilizes grid search to tune the parameters of the genetic algorithm.

[11] A GAN toolbox for PyTorch implementations made available by Facebook Research: https://github.com/facebookresearch/pytorch_GAN_zoo

[12] Accessed through Google Colab.

[13] https://deap.readthedocs.io/en/master/index.html

Quantitative evaluation. The first step of the experiment is designed to analyze the effect of different parameters on the algorithm's performance by comparing sets of different parameter settings for mutation rate, crossover rate, population size, and tournament size. The choice of parameter values presented in table 7 is informed by the parameters settings of the reviewed studies, as explained in section 4.3.

Evolutionary algorithms can either be evaluated by measuring the quality of the developed solutions (in terms of fitness), or by measuring the time it takes for an algorithm to converge to a certain quality. The proposed genetic algorithm is of high complexity, because it involves the use of two deep learning models utilizing the GPU to calculate the fitness for an individual, and searches within a large latent space. Therefore, the running time is likely to present a constraint in its execution. Hence, the former measure is chosen. By fixing the number of generations of an evolution to the feasible maximum, the algorithm is evaluated based on the best fitness reached before termination. Due to the algorithm's stochastic nature, several runs are performed for the same parameter setting. Moreover, the runs are distributed over five different clusters, to achieve robustness across styles. A collection of visually diverse clusters is chosen for this purpose. The same styles are used for all parameters sets to ensure compatibility. For each setting, the best minimum fitness is aggregated over all five runs, resulting in the 'mean best fitness' as a measure of comparison between parameter settings.

Parameter	Setting
Size of individual	512
$N_{generations}$	1000
N_{elite}	1
N_{new}	10
Recombination operator	uniform crossover ($\alpha = 0.5$)
Mutation operator	nonuniform mutation ($\mu = 0, \sigma = 1$)

Table 6: Constant GA parameters.

Qualitative evaluation. Secondly, the proposed model is evaluated by comparing its visual outputs against the creation of images under random sampling. The generation of designs for different individual clusters is inspected to observe similarities and differences in the algorithm's behaviour with regard to parameter choice and specific clusters. The resulting designs are compared against designs created based on random sampling. To do so, the fitness measure

Parameter	Settings
Crossover rate p_{cx}	0.7, 0.9
Mutation rate p_{mut}	0.2, 0.5
Population size p_{pop}	100, 200
Tournament size p_{ts}	3, 6

Table 7: GA parameters under variation.

f_t is applied as a measure of comparison. More precisely, the fittest design generated by an evolutionary search of N_{pop} individuals and N_{gen} generations for designs of style t is compared to the image of highest fitness out of $N_{pop} \times N_{gen}$ randomly initialized individuals.

6 Results

6.1 Style clusters

Based on the 2048-dimensional penultimate embedding, the GMM identified 150 visually coherent styles in the dataset, ranging from 20 to 385 images in size, as described in detail in table 8.

Characteristic	Value
Number of style clusters	150
Average cluster size	118
Average posterior probability to centroid (top 1)	0.9999
Average posterior probability to centroid (worst 1)	0.6411
Average posterior probability to centroid (top 20)	0.999

Table 8: Style clusters.

Figure 9 displays five of such styles picked from the set of style clusters. An overview of all clusters can be found in the appendix. Under closer inspection, one can notice that the identified visual themes are based on more than one feature. They are a combination of a several attributes, such as a long lower part (as part of a skirt or dress) in a light color with folds in the fabric for style 31 (see figure 7a). Taking a detailed look at the difference between style 58 and style 121 makes that explicit (see figure 7c and 7e respectively). While both clusters include the feature of vertical thinly striped tops, style 121 is characterized by the combination with a dark jacket worn on top of the striped top.

However, for two clusters it becomes apparent that the relevant features rely more on the human model than the clothing characteristics, as can be see in figure 8.

Additionally, even though duplicate embedding vectors were removed before applying the GMM, some clusters still reveal the existence of a few images appearing twice, or in unnoticeable slight variation. Instances of duplicates can be found in the cluster components 29, 54, 82, 85, 104, 111, 115, and 134 (see appendix for reference). However, since the clusters in question contain several other outfit images representing the corresponding style, their influence can be disregarded for the nature of this task.

The clustering revealed that the dataset contains some images of items not worn by models, such as underwear (component 97 and 123) or pants or (component 49) only, which as not discovered during the initial review of the data. Clusters specific to images taken in portrait

(a) Style 31

(b) Style 48

(c) Style 58

(d) Style 69

(e) Style 121

Figure 7: Examples of the clustering components. From left to right, each row shows the 10 outfits with the highest posterior probability of belonging to the style component. The images are sorted by decreasing probability from left to right. The depiction framed in green displays the average image of the 20 outfits with the highest posterior probability of belonging to the style component. It is recommended to zoom in to see the details of the images.

(a) Style 46

(b) Style 142

Figure 8: Examples of flawed clustering components. It is recommended to zoom in to see the details of the images.

pose (such as in component 9, 74, 98, 110, and 132) contrast with the majority of clusters. While some cluster components consist only of images in the same pose, others do not seem to distinguish between poses, like cluster component 1 or 21. For few components, like cluster 67, it is not obvious what the underlying similarity depicts. Overall, the analysis shows that the clustering model, including the underlying feature model, could be improved to avoid the described flaws. But for the purpose of this task, for which the clustering model serves as a basis and not as the main subject of investigation, the model is considered sufficient.

6.2 Image generation

After growing over seven epochs, the trained P-GAN is able to generate random samples in a resolution of 256^2 pixels, prompted by random latent vectors drawn from a Gaussian distribution. Figure 9 displays nine such random samples.

Figure 9: Random samples generated by the GAN.

6.3 Evolutionary search for styles

The following sections present the results of genetic algorithm. As described in the experimental design (see section 5), due to the stochastic characteristic of the evolutionary method, several runs are conducted under each parameter combination to receive reliable results. Section 6.3.1 presents these results, intending to offer insight into the effect of the different parameters onto the search. In section 6.3.2 the results for exemplary cluster components are analyzed in detail.

6.3.1 Quantitative analysis

For each parameter combination, five runs for different clusters (cluster 11, 48, 65, 96, and 138 were chosen semi-randomly, ensuring a visual diversity among styles) were conducted. Table 9 presents the comparison of the mean maximum fitness reached across all five runs. In other words, the values aggregate the maximum fitness reached by each of the five runs.

		$p_{mut} = 0.2$		$p_{mut} = 0.5$	
		$N_{te} = 3$	$N_{ts} = 6$	$N_{ts} = 3$	$N_{ts} = 6$
$p_{cx} = 0.7$	$N_{pop} = 100$	0.5746	0.2	0.6021	0.2119
	$N_{pop} = 200$	0.4031	0.5491	0.6341	0.4028
$p_{cx} = 0.9$	$N_{pop} = 100$	0.4538	0.3855	**0.735**	0.2982
	$N_{pop} = 200$	**0.8073**	0.2	0.7043	0.5248

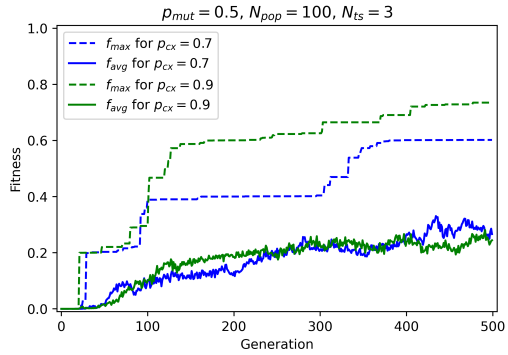
Table 9: Experimental results. The table shows the maximum fitness that was reached over all generations. The values are averages across all five runs per parameter combination. Best results are marked bold.

As the comparison of the maximum fitness revels, the algorithm finds individuals of highest fitness for a crossover rate of $p_{cx} = 0.9$ and a tournament size of $N_{tz} = 3$. In particular, the highest fitness is reached in combination with a population size of $N_{pop} = 200$ and a low mutation rate of $p_{mut} = 0.2$, followed by the second highest fitness with $N_{pop} = 100$ and $p_{mut} = 0.5$.

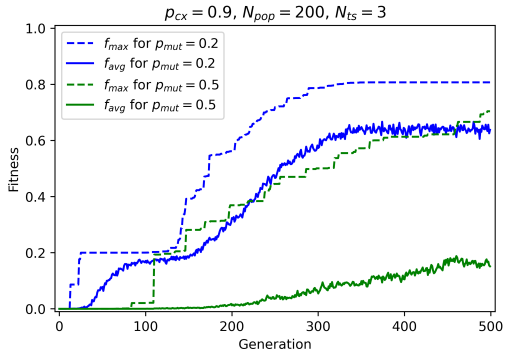
To better understand the effect of the crossover rate, mutation rate, population size, and tournament size on the search behaviour, they are examined one by one in the following. For one parameter to be examined, the other three parameters are chosen in the highest performing combination across both values of the parameter in question. To find that setting, for example when comparing the different parameter values used for mutation rate, the results of the runs in table 9 for $p_{mut} = 0.2$ and $p_{mut} = 0.5$ are compared in each combination with the other

parameter values. To illustrate, 0.5746 and 0.6021 are compared to 0.2 and 0.2119, 0.4031 and 0.6341, and so on for all eight parameter combinations. The setting reaching the highest fitness in total for both mutation rates is selected. In the case of mutation rate, the fitness is highest for the combination of $p_{cx} = 0.9$, $N_{pop} = 200$, $N_{ts} = 3$, than for all other combinations. Therefore, this setting is chosen for the comparison of the different parameter values of p_{mut}.

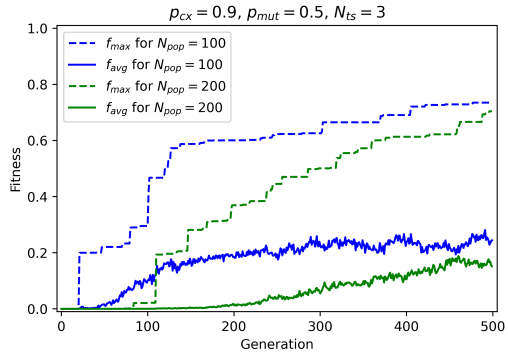
We start with the parameter of crossover rates, followed by mutation rate, population size, and finally tournament size.



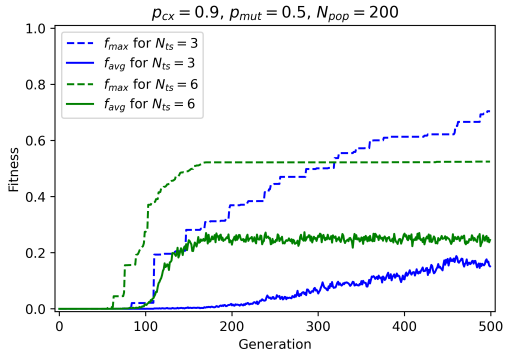
(a) Comparison of crossover rates.



(b) Comparison of mutation rates.



(c) Comparison of population sizes.



(d) Comparison of tournament sizes.

Figure 10: Comparison of parameter settings. The graph shows the maximum fitness and the average fitness per generation. The values are the average across all five runs conducted under these parameter values. The parameter values with the highest maximum fitness across both values of a parameter were chosen for the comparison. They are stated on top of the plot.

As displayed in figure 10a, choosing a higher crossover rate affects the development of the population after the first 100 generations by increasing the maximum fitness. The average population is not strongly affected by the higher crossover rate, as it behaves similarly for both values.

This is different for changing the parameter values of the mutation rate, as can be seen in figure 10b. Raising the rate decreases the fitness of the population, including the fittest individual(s). That is in line with the assumption that high mutation destroys the fit individuals in a population by introducing too much variation to the genotypes. However, as the maximum fitness is growing steadily, the final outcome of maximum fitness is not affected by the higher rate. For a larger number of generations, the maximum fitness might even continue its growth beyond the highest fitness reached with $p_{mut} = 0.2$, which stagnated over the last third of the evolution. Even with the low fitness of the average population, the evolution under a high mutation rate brings forth individuals of high fitness.

The next parameter under inspection is population size in figure 10c. Though a lower population size increases the fitness in the first few hundred generations, the difference in population size does not have a crucial effect on neither the development of the maximum nor the average fitness of the population towards the end of the evolution. This is opposite to what one would expect, as the probability of finding fit individuals in a population of twice the size is higher. Hence, one would assume the population to be fitter in the beginning of the genetic search. That the difference balances out towards the end though means that the higher population size shows its advantage after the variation over several generations.

Coming to the last parameter under comparison, the tournament size seems to affect the development of the fitness in the opposite direction, as displayed in figure 10d. The tournament size during the selection process has a major effect on the search behaviour when increased from $N_{ts} = 3$ to $N_{ts} = 6$. Recall that with a larger tournaments, the fittest individual is chosen out of a higher number of randomly drawn individuals. Naturally, that assigns fit individuals a higher chance of being selected, as the likelihood of participating in a tournament is higher. In other words, they a given more chances to compete and win against weaker individuals. By increasing the tournament size here, both the maximum and average fitness of the population grow drastically in the beginning, but stagnate after around 170 generations. While the maximum fitness for $N_{ts} = 3$ exceeds the maximum fitness for $N_{ts} = 6$ after the total of 500 generations, the average population has a higher fitness for $N_{ts} = 6$ at that point despite its stagnation.

Baring in mind that these values are averages across five different runs and clusters, one should interpret the phenomena described above with caution. For example, a stagnation below 1 might actually represent some runs having reached highest fitness, while other runs never

reached any. It is therefore informative to take a look at the behaviour of individual runs.

6.3.2 Qualitative analysis

To better understand the relationship between the fitness measure and the resulting designs, this section takes a closer look at the individual results of the runs and highlights some informative observations. Figure 151 to 166 in the appendix provide the results for all 80 runs. Due to the stochastic characteristic of evolutionary systems, some runs never achieve any fitness due to an 'unlucky' initialization of the population and/or a weak selection of parameter values. This is especially often the case for experiments with a tournament size of $N_{ts} = 6$, as is also reflected in the quantitative results. In this section, we are interested in the behaviour underlying runs that led to some informative results, hence the examples below are 'cherry-picked' from the collection of experiments.

Figure 11: Results for cluster 96 with $p_{cx} = 0.9$, $p_{mut} = 0.2$, $N_{pop} = 200$, $N_{ts} = 3$. The plot shows the fitness over generations, followed by the image with the highest posterior probability in the style cluster, the fittest generated image, and the fittest out of $N_{pop} \times N_{gen}$ randomly sampled images for comparison. The title of the images displays the fitness.

Under the parameter combination that was found to give the best average results in the previous section, the genetic algorithm produces an image of the highest fitness possible for cluster 96. Taking a look at the final generation in figure 11, the similarity to the target cluster is visible. The original and the generated style share the features of an open denim jacket with bleached plats and darker (denim) pants. Compared to random sampling, the fittest randomly sampled image also contains a denim item. But its fitness lies under 0.0, indicating that either the pants are not sufficient to define the cluster, or that the cluster is biased towards the full-body pose. The plot of the fitness over generations reveals that a high fitness was already reached within the first generations, followed by a high average fitness after around 150 generations. Since new individuals are introduced in every generation and recombined with existing individuals, it is unlikely that the average fitness of a population will ever reach a maximum of 1.

A similar behaviour can be observed for a run for cluster 11, as detailed in figure 12. The fittest individual reaches a fitness of 0.967, while the average fitness remains under 0.8. Inspecting some of the other runs of for the cluster (see appendix), a brown and green pattern is often visible in the upper body area, often as part of a dress instead of an outfit with pants and jacket. Taking e.g. a look at figure 159a and 166a, the search for images of style 11 converges in two visually similar images. In both cases, the maximum and average fitness stagnate below 0.6 for several generations.

Figure 12: Results for cluster 11 with $p_{cx} = 0.7$, $p_{mut} = 0.2$, $N_{pop} = 100$, $N_{ts} = 3$. The plot shows the fitness over generations, followed by the image with the highest posterior probability in the style cluster, the fittest generated image, and the fittest out of $N_{pop} \times N_{gen}$ randomly sampled image images for comparison. The title of the images displays the fitness.

For a run for cluster 65 displayed in figure 13, where the fitness drastically increased within the first generations, the algorithm finds a black dress with a fitness of $f = 1$. Here, however, the average fitness of the population stays under 0.5. For another run of that cluster, see figure 157c in the appendix, a white dress in similar shape results from a population of an average fitness below 0.1 as the fittest design with a fitness of 0.997.

Figure 13: Results for cluster 65 with $p_{cx} = 0.7$, $p_{mut} = 0.5$, $N_{pop} = 100$, $N_{ts} = 6$. The plot shows the fitness over generations, followed by the image with the highest posterior probability in the style cluster, the fittest generated image, and the fittest out of $N_{pop} \times N_{gen}$ randomly sampled image images for comparison. The title of the images displays the fitness.

The cluster with the most diverse results over all runs is cluster 48. Figure 14 shows the results for one of the runs. To assign the image a fitness of 1.0, the clustering model must base its judgement on other factors than the apparent clothing features of a black jacket with a

hoodie. In other cases of this style, the search seems to converge in non-optimal solutions with a fitness below $f = 0.5$, as it can be observed in figure 161b, 164b, or 166b in the appendix.

Figure 14: Results for style cluster 48 with $p_{cx} = 0.9$, $p_{mut} = 0.5$, $N_{pop} = 200$, $N_{ts} = 3$. The plot shows the fitness over generations, followed by the three images with the highest posterior probability in the style cluster, the fittest generated image, and the fittest image out of $N_{pop} \times N_{gen}$ randomly sampled images for comparison. The titles of the images display the fitness.

The phenomena of converging in non-optimal solutions can also be observed for cluster 138, where most of the runs get stuck in designs containing white gowns and bare legs. In that case, the fittest individual in the local optimum is developed further within its limitations, but other possible solutions outside of this frame are not explored. Here, in figure 15, that might be interpreted as the pattern being added to the shirt, while the rest of the design stays the same.

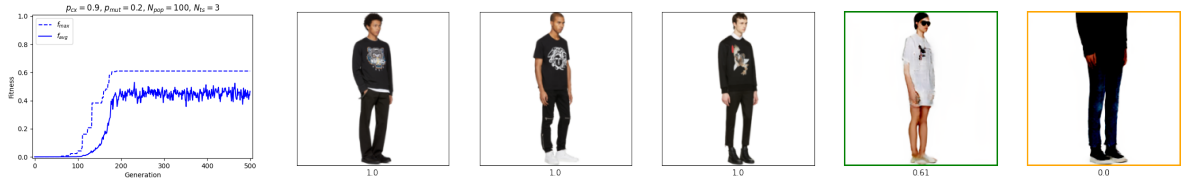


Figure 15: Results for cluster 138 with $p_{cx} = 0.9$, $p_{mut} = 0.2$, $N_{pop} = 100$, $N_{ts} = 3$. The plot shows the fitness over generations, followed by the image with the highest posterior probability in the style cluster, the fittest generated image, and the fittest out of $N_{pop} \times N_{gen}$ randomly sampled image images for comparison. The title of the images displays the fitness.

Overall, the generation of images is guided into the direction of features of the target style clusters. In comparison to random sampling, the derived individuals are fitter than randomly drawn ones. However, the similarities that are found between target style and generated designs do not necessarily correspond to the visual attributes we might assume relevant for a style. As the presented results only give a small insight into the experimental results, the reader is encouraged to take a look at the appendix. A discussion of the results and the aspects underlying the system's behaviour is presented in the next section.

7 Discussion

This section discusses the results presented above and considers possibilities to improve the search of the latent space.

7.1 Convergence in local maximum

Coming back to the last example of the previous section, a white outfit with a black print on the chest was generated in response to a cluster of black outfits with white prints. When taking a look at the algorithm's behaviour and the evolution of the population in detail, the following happens upon the convergence into a local maximum. If an individual is stronger than the others, however still only with a posterior probability of belonging to the target cluster of a few percent, it is still chosen over the others. As tournament selection allows individuals to be chosen several times in a generation, it is likely to be presented in the offspring more often than once, due to winning more that one of the selection tournaments. Over few generations in a population of a low average fitness, the fittest individual in question begins to dominate the population.

Though in general, convergence to a local maximum is a desirable end stadium for genetic algorithms, it poses a problem in this scenario. The chosen image of a white gown was only with very low probability part of the cluster. Based on that, the fittest individual evolved over the following generations. Within the boundaries of this design, the following populations are optimized. That could be interpreted as that the main outfit is not changing, but minor features, like the print on the chest, might be added. But due to one weak style match dominating the population, the resulting designs are limited to the prevailing look. Hence, there is no diversity within the population. With such a set of latent vectors, it is difficult to improve the design within these boundaries. The final population consists of images similar to each other, all sharing around the same fitness value. A possible solution to that, by adjusting the parameters of the genetic algorithm, is discussed in section 7.4.

7.2 What does the machine *see*?

While the previous example illustrates the convergence in a local maximum below the maximum posterior probability of 1, there are also many cases where the evolution reaches a max-

imum fitness comparable to the images of the cluster component with highest posterior probability, but the visual result does not seem to align with the target style at all. For examples, see figure 163a, 163e and 165d in the appendix. This points to the possible difference in the reception of styles between human and machine. We, as human beings, come with a background in clothing, style and design of the culture we are raised in. Behind what we identify as decisive properties for a style, the suggested system might find some completely different attributes, such as only the pattern as in figure 15 or the mix of colors (see figure 161a in the appendix). The difference in perception might also be due to the fact that computational networks are not able to capture subtle differences ins styles (yet) (Takagi et al., 2017). In addition to that, the found styles are not independent of the other images of the complete dataset, based on which the GMM's defines the components.

To quantify the relation between the clusters found by the evolutionary search and human's judgement, experiment could be conducted. To guide the system into the direction of human-compatible styles, the approach could eventually be combined with interactive evolution like in DeepIE or StyleIE, in order to integrate how humans see style. Oppositely, it might also be considered if this way of exploring the latent space can even tell us more about what the clustering model 'sees' in styles. That might give us insight into clustering components. Following that path might contribute to understand how the machine sees visual themes emerge, eventually leading to a better understanding of its functioning. Considering a machine sense of creating stylish design adds to the field of computational creativity.

7.3 What makes a style?

Coming back to the semantic concept of fashion style, its understanding plays a crucial role in the underlying approach. The applied interpretation can be placed in the middle of a spectrum of style ranging from low-level similarity on the one end of the spectrum to "manually crafted style categories" (Hsiao and Grauman, 2017, p.1) on the other end. In doing so, it aims to find a balance between seeing styles as a too specific concept, that makes the discovery of similar styles difficult, and as too abstract conceptions that make reference to visual features impossible. Ultimately, this should enable the detection of meaningful styles in an unsupervised manner.

Detecting style-coherent similarity of clothing images remains a challenge, in which also the attribute models used to retrieve the visual embedding, including the dataset on which it

is trained, play an important role (Hsiao and Grauman, 2017). Even though the penultimate embedding of images was used instead of the ResNet's final layer's attribute prediction scores, the categories underlying its training influence what the model 'sees' on every layer. A better understanding into what the features are that the network looks at could be achieved in a qualitative analysis of the ResNet's performance, e.g. by with the help of a Class Activation Map (CAM).

As Takagi et al. (2017) showed, such an analysis could shed insight on the critical features for a certain fashion style. With that knowledge, the latent space of the GAN might be controllable in a more targeted way, especially in combination with secondary latent space disentanglement, as illustrated ADDREF. Takagi's qualitative analysis of the ResNet50 used for the classification of pre-defined fashion styles shows that the detected features are independent of spatial position. Hence, the choice of images of a certain pose as a basis for the clustering model should not be of importance. However, with different poses comes a different proportion of image features in an image, which might play a role.

Proportion also matters when considering how the appearance of an attribute is weighted against the appearance of other attributes. This matters with regard to the question of applying normalization and/or feature scaling to the embedding vectors before the clustering. An example for illustration can be given from the domain of language. To compare the content of texts of varying length with each other, the appearance of words can be weighted against a text's length by applying L2 normalization, hence transforming a whole vector of word counts to unit length. By applying L2 normalization to the penultimate embedding of image features, (Matzen et al., 2017) weight an attribute's appearance by the total number of attributes in the outfit. Note that the penultimate embedding does not represent the features per se. But because it is translated to the attribute scores, some similarity is assumed. While L2 normalization is a common procedure in language processing, it was not applied here as the images of FashionGen consist of around the same size and pose, and there was no need to make up for size in that regard. Hence, fewer attributes due to fewer clothing items in an image are understood as part of a style, not as a smaller outfit. To be comparable, outfits are regarded as same size. To illustrate, imagine an outfit only consisting of a top and shorts with a zipper, hence only containing a few features. This zipper should not be assigned more weight than the zipper in an outfit containing more features, such as an additional blouse, boots, bag, and coat.

7.4 Parameters in the evolutionary search

One issue with applying evolutionary computational methods can be the setting of the many parameters that influence the behaviour of the optimization algorithm. The evolutionary parameters are not independent from each other, hence making their effect of the performance hard to observe during tuning procedures. Most crucially, as both recombination and mutation affect the diversity, in theory none of them can be updated individually.

This study only set out to explore two values each for four different parameters. In this grid search procedure, better settings might, however, lie in between the values under inspection. Even under close observation of the development from generation to generation, the initial setting of evolutionary parameters remains an open research question Eiben et al. (2003); Hassanat et al. (2019). On top of that, many more aspects in the evolutionary search could be considered, such as the independent probability of each variable in a latent vector to be mutated or recombined. As we work with a genetic representation of continuous numbers, the mutation is a more complex topic than the classical reset mutation for binary representations. In particular the Gaussian mutation applied here requires parameter settings. The values that are added to the genotypes can be controlled by setting the standard deviation of the distribution they are drawn from. Like the others, this parameter can also be set a dynamic manner, e.g. in response to the average fitness of the population, or by decreasing it with the growing number of generations. The latter would allow the exploratory search of the latent space in the beginning of an evolution. Towards the end, by which the population's fitness should have decreased, smaller mutation supports the fine-tuning of individuals. Hence, the distribution parameter would work as a step size taken by the mutation (Eiben et al., 2003). Additionally, due to the stochastic nature of evolutionary systems, running the algorithm more times, both for the same style cluster, but also for more style clusters, would lead to more robust results. However, due to the time restrictions and computational limitations underlying this work, the maximum effort was made.

Another issue when evaluating evolutionary systems is the choice of a means of measurement. Here, the (mean) best fitness was chosen to be measured against a pre-given number of generations. Reviewing the results revealed that in many cases, the conducted number of generations would have not been necessary, as the algorithm often converges earlier on. Hence, other performance measures could be reconsidered. Instead of measuring the solution qual-

ity, one could instead focus on the number of generations until convergence to a pre-defined fitness value. That could e.g. be the maximum fitness of 1, but also take more complex approaches, such as the average fitness of a number of images (of highest posterior probability) in the cluster component. Additionally, a more dynamic approach could consider the population's improvement over a certain range of generations. That could also be a potential way to avoid the convergence in a local maximum, that does not fulfill the fitness requirements. If for example the algorithm gets stuck at a low fitness level, the variation and/or the number of new individuals could be increased, as well as the fittest individuals could be aborted.

7.5 Limitations of the design space

While diversity in a genetic algorithm could normally be improved by increasing the mutation and crossover probability, or by introducing new random individuals to the population, a problem here might be the size of the design space. In such a large space of possible latent variables, it is impossible to oversee the link to visual features among it. However, the approach taken attempts to control the latent variables through evolutionary exploration of the latent space, without adding more transparency with regard to the variables effect on the generator's output.

A way to make the latent space better understandable and controllable could be by applying another network architecture for the generator. As shown by StyleIE, using a StyleGAN as GAN architecture can have its advantages in some use cases Tejeda-Ocampo et al. (2021). In comparison to the vanilla GAN in DeepIE (Bontrager et al., 2018), the search for certain design properties, as opposed to free exploration, is better supported when the secondary latent space is disentangled with regard to certain stylistic attributes. As the search for designs according to fashion styles is a targeted exploration of the design space, using disentangled representations as genotypes might support the control and variety of the visual features in a better manner. However, such a semi-supervised learning technique requires more complicated training procedure than a the P-GAN used in this task[14].

Another approach to generating fashion styles could be the identification of style vectors in the GAN's latent space. After generating a large population of images, their projection onto the clustering space could serve as a basis to train a linear classifier to separate latent vectors

[14]Zalando Research has trained a StyleGAN model with clothing data, which could potentially be used as a pre-trained model for the task. Unfortunately, the implementation is not released to the research community.

corresponding to images of a certain target style from latent vectors of images that do not respond to the style. In line with Denton et al. (2019), who identify facial attribute vectors orthogonal to the decision boundary of the classifier discriminating between smiling and not smiling faces, one could identify style vectors. By traversing along the directions of a style vector, one might in theory interpolate smoothly between designs not falling under the style in questions and 'stylish' designs. In practice however, we must remember that the identified styles consist of several features, hence making the interpolation an interplay of complex factors. In addition to that, the random sampling revealed that images of a certain style are difficult to discover by chance.

7.6 The role in the design process

As Al-Halah and Grauman (2020) and Takahashi et al. (2020) have shown, the discovery of styles over time enables the prediction of trend development in the future. Thinking the idea of generative models that can respond to styles further, future endeavors could include the integration of a time-dimension into the approach. By utilizing datasets with a temporal level, images could potentially be generated in response to emerging styles. With "fashion as a suitable field for trend research" (Mackinney-Valentin, 2010, p.24), the approach could also be taken to other fields, in which the responding to trends plays a role. Additionally, one could approach the goal of creating designs that do not fit into existing styles, in line with the approach of CANs, which aims to create new styles within the arts (Kravtsov and Kuznetsov, 2017), as presented in section **??**. To do so, the objective of the evolutionary search could be reversed, encouraging the generation of novel styles that have note been here before.

Formulating his vision of an Embodied Artificial Evolution technology, Eiben et al. (2012, p.263) takes the idea of the machine as a designer a step further by hypothesizing that

> *In the long term, the basic design-and-manufacture loop of the production industry may be transformed from the present off-line type with a critical role for the human designer to a more on-line process. In this process new designs arise though evolutionary variations (are 'born'), tested immediately in vivo, and reproduce to seed new designs, if successful. While this is clearly not an appropriate workflow for all production industries, there are several potential application areas ranging from fashion items to bio-medical nano-robots.*

When developing systems that mimic human tasks, such as creativity, one shall reflect on their purpose carefully, in particular on the technology's involvement of human operators and designers. Instead of replacing a human designer, we could also ask how the system supports human creativity. Through the co-creation of designer and machine, new ways of creating may arise.

As a final remark, it is important to reflect on the datasets used for training generative models. As Takagi et al. (2017) points out, fashion photographs do not represent what people actually wear, hence might not give an actual representation of current styles. Most datasets, used for training generative fashion design models consist of catalog or social media images, highly biased to the presented populations. A crucial task is therefore to discuss the data's effect on how stability of a system is reached with regards to design diversity, such as achieving desirable silhouettes while still considering diverse body forms.

8 Conclusion

This research aimed to extend the classical generative deep learning approach to facilitate the generation of designs that respond to fashion styles. The application of a genetic algorithm to guide the generation of images with regard to previously identified style clusters was investigated. A system was suggested that facilitates the search for images responding to a certain style cluster.

In particular, a ResNet50 pre-trained on DeepFashion was employed to translate images to an penultimate embedding capturing clothing-related features. Based on the embedding, a GMM was applied to identify style components, or clusters, among the images. In parallel, a GAN was trained on the dataset, creating a latent space mapping latent variables to visual features that represent the distribution of the training set. After the training, the images created by the generator can be translated to a penultimate embedding in the same manner. The embedding's projection onto the clustering space reveals its probability of belonging to a style cluster. Finally, a genetic algorithm was implemented by approaching the generated images as phenotypes, and the latent vectors, that prompt their generation, as the corresponding genotype. By picking a certain style as a target cluster, an image's posterior probability of belonging to that cluster is used as a measure of fitness. In a population of latent vectors, the ones corresponding to images with higher fitness are selected and manipulated in a reintegrating process of several generations. This evolutionary search followed the goal of picking latent vectors generating images of a certain style.

The experimental results indicate that the proposed procedure can guide the search for style-coherent generated designs, but that further research is required to establish a robust and reliable exploration process. We find that while the system is able to generate images of maximum fitness, the designs do not necessarily correspond to the target styles in the way one would expect. Rather, the generations reveal a machine understanding of fashion style. While some of the generated images exhibit similarity to the target cluster, that is visible to the human eye, other generated outputs raise the question of how the algorithm might understand styles differently. Better understanding the cluster space or disentangling the GAN's latent space could enable a more targeted navigation. Further investigation into the parameter setting of the genetic algorithm and the interplay between clustering model and latent space are required to

make the system reliable.

The findings contribute to the field of intelligent fashion by drawing a connection between fashion style detection and artificial fashion generation. Instead of a primary focus on algorithmic aspects, the suggested approach focuses on the relevance of generative models within the creative field, namely that fashion is guided by style trends. In comparison to generative approaches coming from an algorithmic angle, this study approaches the creative field of fashion based on what defines it at its core, namely styles. Guiding the generation based on the concept of style might extend the forecasting of trends with the generation of trending designs in the future.

Considering how a machine sense of creating stylish design adds to the field of computational creativity. In light of the developing field, the approach opens up for questions regarding the role of generative models in the design process.

9 Acknowledgement

I want to sincerely thank everyone who has helped and supported me throughout the entire process of writing this Thesis, to finalising and reading draft upon draft. I especially want to thank my close friends Mette, Kasper, Mads and Maren for reading and correcting grammar and layout as well as providing a clear a rationalised perspective. Additionally I want to thank my mother, father brother and sister who have listened, and given incredible motivational support when program upon program would encounter bugs. At last I want to thank my boyfriend Emil who has read, listened to my theories, ensured that shopping and cleaning and the general maintenance of our home. A heartfelt sincere thank you to Flemming who lend me his MacBook power supply when mine failed in the final hours. And at last thank you to my supervisor Manex for his spontaneous and always creative and motivating input, help and advice.

Bibliography

Al-Halah, Z. and Grauman, K. (2020). Modeling Fashion Influence from Photos. *IEEE Transactions on Multimedia*.

Bontrager, P., Lin, W., Togelius, J., and Risi, S. (2018). Deep interactive evolution. In *International Conference on Computational Intelligence in Music, Sound, Art and Design*, pages 267–282. Springer.

Bossard, L., Dantone, M., Leistner, C., Wengert, C., Quack, T., and Van Gool, L. (2012). Apparel classification with style. In *Asian conference on computer vision*, pages 321–335. Springer.

Cheng, W.-H., Song, S., Chen, C.-Y., Hidayati, S. C., and Liu, J. (2021). Fashion Meets Computer Vision: A Survey.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. Technical Report MSR-TR-2014-21, Microsoft.

Denton, E., Hutchinson, B., Mitchell, M., and Gebru, T. (2019). Detecting bias with generative counterfactual face attribute augmentation. *arXiv preprint arXiv:1906.06439*.

Eiben, A. E., Kernbach, S., and Haasdijk, E. (2012). Embodied artificial evolution. *Evolutionary Intelligence*, 5(4):261–272.

Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.

Elgammal, A., Liu, B., Elhoseiny, M., and Mazzone, M. (2017). CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms. *arXiv preprint arXiv:1706.07068*.

Epstein, Z., Levine, S., Rand, D. G., and Rahwan, I. (2020). Who gets credit for ai-generated art? *Iscience*, 23(9):101515.

Fernandes, P., Correia, J., and Machado, P. (2020). Evolutionary latent space exploration of generative adversarial networks. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 595–609. Springer.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

Ge, Y., Zhang, R., Wang, X., Tang, X., and Luo, P. (2019). Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5332–5340.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. *arXiv preprint arXiv:1406.2661*.

Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., and Prasath, V. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(12):390.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Hollander, A. (1978). *Seeing Through Clothes*. The Viking Press, New York, NY, USA.

Holmegaard, I. (2020). *Look*. Gyldendal.

Hsiao, W.-L. and Grauman, K. (2017). Learning the Latent "Look": Unsupervised Discovery of a Style-Coherent Embedding from Fashion Images.

Jetchev, N. and Bergmann, U. (2017). The conditional analogy gan: Swapping fashion articles on people images. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2287–2292.

Jiang, S., Li, J., and Fu, Y. (2021). Deep learning for fashion style generation. *IEEE Transactions on Neural Networks and Learning Systems*.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv preprint arXiv:1710.10196*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410.

Kiapour, M. H., Yamaguchi, K., Berg, A. C., and Berg, T. L. (2014). Hipster wars: Discovering elements of fashion styles. In *European conference on computer vision*, pages 472–488. Springer.

Kim, H.-S. and Cho, S.-B. (2000). Application of interactive genetic algorithm to fashion design. *Engineering Applications of Artificial Intelligence*, 13(6):635–644.

Kravtsov, P. and Kuznetsov, P. (2017). Creative Adversarial Networks. https://github.com/mlberkeley/Creative-Adversarial-Networks. commit xxxxxxx.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Liu, S., Yang, S., and Zhou, H. (2020a). Imitation Learning for Fashion Style Based on Hierarchical Multimodal Representation. *arXiv preprint arXiv:2004.06229*.

Liu, X., Li, J., Wang, J., and Liu, Z. (2020b). MMFashion: An Open-Source Toolbox for Visual Fashion Analysis. *arXiv preprint arXiv:2005.08847*.

Liu, Y., Chen, W., Liu, L., and Lew, M. S. (2019). Swapgan: A multistage generative approach for person-to-person fashion style transfer. *IEEE Transactions on Multimedia*, 21(9):2209–2222.

Liu, Z., Luo, P., Qiu, S., Wang, X., and Tang, X. (2016). Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104.

Mackinney-Valentin, M. (2010). *On the Nature of Trends: A Study of Trend Mechanisms in Contemporary Fashion*. PhD thesis.

Mahmood, A., Ospina, A. G., Bennamoun, M., An, S., Sohel, F., Boussaid, F., Hovey, R., Fisher, R. B., and Kendrick, G. A. (2020). Automatic hierarchical classification of kelps using deep residual features. *Sensors*, 20(2):447.

Malhotra, A. and Aggarwal, V. (2010). HIER-HEIR: An Evolutionary System with Hierarchical Representation and Operators Applied to Fashion Design. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 205–214. Springer.

Mall, U., Matzen, K., Hariharan, B., Snavely, N., and Bala, K. (2019). Geostyle: Discovering fashion trends and events. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Matzen, K., Bala, K., and Snavely, N. (2017). StreetStyle: Exploring world-wide clothing styles from millions of photos.

Mok, P., Xu, J., Wang, X., Fan, J., Kwok, Y., and Xin, J. H. (2013). An IGA-based design support system for realistic and practical fashion designs. *Computer-Aided Design*, 45(11):1442–1458.

Mok, P., Xu, J., and Wu, Y. (2016). Fashion Design Using Evolutionary Algorithms and Fuzzy Set Theory - A Case to Realize Skirt Design Customizations. In *Information systems for the fashion and apparel industry*, pages 163–197. Elsevier.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Rostamzadeh, N., Hosseini, S., Boquet, T., Stokowiec, W., Zhang, Y., Jauvin, C., and Pal, C. (2018). Fashion-Gen: The Generative Fashion Dataset and Challenge. *arXiv preprint arXiv:1806.08317*.

Roziere, B., Teytaud, F., Hosu, V., Lin, H., Rapin, J., Zameshina, M., and Teytaud, O. (2020). EvolGAN: Evolutionary Generative Adversarial Networks. In *Proceedings of the Asian Conference on Computer Vision*.

Takagi, M., Simo-Serra, E., Iizuka, S., and Ishikawa, H. (2017). What makes a Style: Experimental Analysis of Fashion Prediction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2247–2253.

Takahashi, S., Yamaguchi, K., and Watanabe, A. (2020). CAT STREET: Chronicle Archive of Tokyo Street-fashion. *arXiv preprint arXiv:2009.13395*.

Tejeda-Ocampo, C., López-Cuevas, A., and Terashima-Marin, H. (2021). Improving deep interactive evolution with a style-based generator for artistic expression and creative exploration. *Entropy*, 23(1).

Volz, V., Schrum, J., Liu, J., Lucas, S. M., Smith, A., and Risi, S. (2018). Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 221–228.

Yildirim, G., Jetchev, N., Vollgraf, R., and Bergmann, U. (2019). Generating high-resolution fashion model images wearing custom outfits. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0.

Yildirim, G., Seward, C., and Bergmann, U. (2018). Disentangling Multiple Conditional Inputs in GANs. *arXiv preprint arXiv:1806.07819*.

Zhu, S., Urtasun, R., Fidler, S., Lin, D., and Change Loy, C. (2017). Be your own Prada: Fashion Synthesis with Structural Coherence. In *Proceedings of the IEEE international conference on computer vision*, pages 1680–1688.