POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Forcing latent space Disentanglement for enhanced model Explainability

Tesi di Laurea Magistrale in
Computer Science Engineering

Author: **Federico Romeo**

Student ID: 969644
Advisor: Prof. Giacomo Boracchi
Co-advisors: Loris Giulivi
Academic Year: 2022-23

Abstract

Despite the widespread use of deep neural networks, their intricate nature and extensive parameterization makes them essentially black boxes, posing challenges to interpretability. Recent deep learning trends aim at developing more interpretable models by working on the implicit intermediate representations formed during training. To this end, enabling a direct link between the data semantic factors of variation (FoVs) and the latent representation could be beneficial to interpretability. This is the case of disentanglement, a property of latent representations in which a change in one latent dimension corresponds to a change in one single FoV, while being relatively invariant to changes in others. In a disentangled representation, individual latent dimensions correspond to distinct and interpretable FoVs, allowing for a clearer control over the latent space, and accordingly over the output. Yet, achieving substantial disentanglement is a complex task, especially in unsupervised fashions where careful design choices and training strategies are required. With the increase of data complexity, models struggle to grasp the factors variability and to isolate them in dedicated latent dimensions. Past attempts at unsupervised learning of disentangled representations are mostly confined to biased toy datasets, missing real-world potential. For these reasons, when focusing on explainability of real-world datasets, unsupervised frameworks results being too weak, claiming the need of a driven supervision. In this work, we propose a supervised framework able to enhance the latent space attribute-level disentanglement of a model in real-world datasets. It requires a supervision made of paired images that share all but one FoV; in this way the image pairs differ in a single aspect, or FoV, whose difference is forced to be encoded in a dedicated dimension through a custom loss function. To evaluate the effectiveness of our method also on real-world datasets, our contribution includes also the generation of the required paired supervision, with a focus on facial images. To this end we introduce a novel Semantic Facial Attribute Editing method able to perform fine-grained facial edits to create our proposed dataset of image pairs differing in a single facial attribute. Experiments on both datasets reached convincing results in term of attributes disentanglement.

Keywords: Deep Learning, Disentanglement, Beta Variational Autoencoders, Semantic Facial Attribute Editing.

Abstract in lingua italiana

Nonostante la diffusione delle reti neurali, la loro natura intricata e l'ampia parametrizzazione le rende essenzialmente delle black boxes, ponendo problemi di interpretabilità. Le recenti tendenze del deep learning mirano a sviluppare modelli più interpretabili lavorando sulle rappresentazioni implicite formate durante l'addestramento. A tal fine, la possibilità di creare un collegamento diretto tra i fattori semantici di variazione dei dati (FdV) e le rappresentazioni latenti potrebbe essere vantaggiosa per l'interpretabilità. Questo è il caso del disentanglement, una proprietà delle rappresentazioni latenti in cui un cambiamento in una dimensione latente corrisponde ad un cambiamento in un singolo FdV, rimanendo relativamente invariante agli altri. In una rappresentazione disentangled, le singole dimensioni latenti corrispondono a FdV distinti e interpretabili, consentendo un controllo più chiaro dello spazio latente e quindi dell'output. Tuttavia, raggiungere un disentanglement significativo è un compito complesso, soprattutto nei modelli non supervisionati, dove sono necessarie strategie di addestramento accurate per districare efficacemente i FdV. Con l'aumento della complessità dei dati, i modelli faticano a cogliere e districare la variabilità dei FdV. I precedenti tentativi di apprendimento non supervisionato di rappresentazioni disentangled sono per lo più confinati a dataset giocattolo. Per questi motivi, quando ci si concentra sull'explainability di dataset reali è richiesta una supervisione guidata. In questa tesi, proponiamo un framework supervisionato in grado di migliorare il disentanglement a livello di attributi dello spazio latente. Esso richiede una supervisione costituita da coppie di immagini che condividono tutti i FdV tranne uno; in questo modo le coppie di immagini differiscono in un singolo aspetto, o FdV, la cui differenza è forzata ad essere codificata in una dimensione dedicata attraverso una loss personalizzata. Per valutare l'efficacia del nostro metodo anche su dataset reali, proponiamo anche un nuovo metodo di Modifica Semantica di Attributi Facciali in grado di eseguire modifiche facciali specifiche, per creare un dataset di coppie di immagini che differiscono per un singolo attributo facciale, trattati come FdV dei dati. Gli esperimenti condotti su entrambi i dataset hanno raggiunto risultati convincenti in termini di disentanglement degli attributi.

Parole chiave: Deep Learning, Disentanglement, Autoencoders Variazionali, Modifica Semantica di Attributi Facciali

Contents

1 | Introduction

Deep learning models have established their dominance in the computer vision field, being employed for a variety of tasks, including representation learning. Despite their widespread use, their intricate nature and extensive parameterization are making deep neural networks essentially black boxes, posing challenges when it comes to interpreting their outputs. Recent deep learning trends aim at developing more interpretable models by working on the implicit intermediate latent representations formed during training.

To this end, enabling a direct link between the data semantic factors of variation (FoV) and the latent representation could be beneficial to interpretability. This is the case of **disentanglement**, a property of latent representations in which a change in one latent dimension corresponds to a change in one single FoV, while being relatively invariant to changes in others [2]. In a disentangled representation, individual latent dimensions correspond to distinct and interpretable FoVs, allowing for a clearer control over the latent space, and accordingly over the output. So, disentangled representation learning refers to the process of isolating (or untangling) different FoVs into semantically meaningful independent variables, crafting an efficient non-redundant latent representation. As an example, suppose to have a simple dataset of white dots: to fully explain it we just require the dot's x and y coordinates, since no other variabilities exist. We denote them as the **factors of variation (FoV)** of the data; thus we only need two independent and interpretable latent dimensions.

Yet, achieving substantial disentanglement is a complex task, especially in unsupervised fashions where careful design choices and training strategies are required to effectively disentangle the data FoVs. Moreover, with the increase of data complexity, models struggle to capture the totality of variability factors and to untangle them in independent dimensions. Past attempts at unsupervised learning of disentangled representations have shown promising results, but are mostly confined to biased toy datasets, missing out on fully unleashing its potential capabilities. Indeed, the exploitation of those inductive biases is exactly the reason behind that successful disentanglement. No factor disentanglement can

happen on a real-world dataset in absence of inductive biases or explicit supervision, due to the overwhelming volume of information it contains which cannot feasibly be managed through unsupervised methods. For these reasons, when focusing on explainability of real-world datasets, unsupervised frameworks results being too weak, claiming the need of a driven supervision [22].

In literature, many other works [24] [23] [18] [11] [10] claimed the convenience of disentangled representations; nevertheless we assert that restricting their application to toy datasets clearly limits its practical usefulness. To promote disentanglement, some methods have been proposed in the literature: incorporating labeled data during training [6], or introducing an adversarial network to discriminate between different dimensions of the latent space [25], or finally providing image pairs during training to encourage the model to disentangle factors that are consistent across the group, while allowing variations within the group [13] [3] [31].

We propose **SVAE**, a supervised framework able to enhance the attribute-level disentanglement of a Beta Variational Autoencoder (βVAE) [10], to develop interpretable models in which different semantic FoVs are encoded in separate latent dimensions. Its applicability extends also on real-world datasets, i.e. where the data generative process isn't trivially aligned to the data FoVs as in *DSprites* [26]. It requires a supervision made of paired images that share all but one FoV; in this way the image pairs differ in a single aspect, or FoV, whose difference is forced to be encoded in a dedicated dimension through a custom loss function. To evaluate the effectiveness our our method also on real-world datasets, our contribution includes also the generation of the required supervision with a focus on facial images. To this end we also introduce E2Editor, a novel Semantic Facial Attribute Editing (SFAE) method able to perform fine-grained edits to modify single facial attributes, to generate our proposed dataset *DFaces* of image pairs with a single non-shared FoV (for example changing only the hair color).

Experiments on both datasets reached convincing results in term of attributes disentanglement.

Outline of the Thesis

Briefly, the thesis is structured as follows:

- In Chapter 2, we discuss the literature on disentangled representation learning, both in the unsupervised and supervised case, we present an overview of VAE techniques, in particular β-VAE [10] and its modifications, to lay the necessary technical foundation for understanding the solution proposed in this thesis.
- In Chapter 3, we present our approach, which we refer to as **SVAE**, describing the rationale behind the adopted framework to favor the disentanglement of the factors of variation. We also describe **E2Editor**, our novel Semantic Facial Attribute Method used to craft the "facially disentangled" image pairs dataset *DFaces*, that reachs state-of-the-art results.
- In Chapter 4, we analyse the performance of SVAE in our experimental setting, evaluating its soundness on a tailored variation of DSprites dataset, *SDSprites*, and demonstrating its effectiveness and applicability on the generated *DFaces*.
- In Chapter 5 we summarise the work done, draw our conclusions and discuss future works.

2 | Background & Related Works

This chapter provides an overview of the key concepts in the field of representation learning; in particular *disentanglement* will be of crucial importance in this work. Then we will see how in the literature it has been promoted by some neural networks in the field of generative artificial intelligence, and how could it be a important dowel in model explainability. So we will first start by describing Variational Autoencoders, and various developments that have taken place over the years, describing their limitations and strengths towards disentangled representation learning. Finally, we will discuss a taxonomy of methods to perform Semantic Facial Attribute Editing, and we will explain how could it be useful to provide an explicit supervision to enhance disentanglement. To this end, we will first digress about Generative Adversarial Networks, and then focus on an useful variant used in this work, StyleGAN.

2.1. Explainable AI

Nowadays, a popular tendency in the Deep Learning field is known as "overparameterization": this concept is well described in the *scaling hypothesis*, that suggests that increasing the number of parameters of a neural network, the dataset size and the training epochs can potentially lead to better generalization performances. This whole process comes at the cost of explainability, agreed to be the biggest weakness of Deep Learning.

Explainability refers to the process of retaining an intellectual oversight over an algorithm and to the ability of understanding its decision-making process. Being deep neural networks successions of many layers, the output's explainability gets inevitably lost through these intricate connections. One could just take the output of the model as it is, and trust the mechanism behind. However the trust has to be built up from explainable patterns to be fully accepted. If we want AIs to gain more trust and to become mainstream, we must definitely work on their explainability. Also, if the model has to be corrected due to some issues or bugs, the troubleshoot would result way worse in a black box model. Thus, explainability techniques could definitely support the troubleshooting.

So, Explainable AI (XAI) aims to make the decision-making process of AI models more understandable and transparent to humans.

2.2. Disentanglement

In machine learning, representation learning is a key area of research in which algorithms learn to meaningfully encode and interpret data. The ultimate aim of representation learning is to develop generalizable models that start from meaningful embeddings and can be applied to any kind of task. Disentanglement is a one of the most desirable property for representations.

It has seen many definitions over the years. One of the most used and agreed definition is from [2], and claims that in a disentangled representation a change in one latent dimension corresponds to a change in a single generative factor while being relatively invariant to changes in the others. According to [10], disentanglement refers to the property of a learned representation where different latent dimensions capture independent factors of variation in the data. In [11] the authors tried to give a more formal and mathematical definition, connecting the concepts from groups theory.

Borrowing from basic *set theory* mathematics, we propose a formal yet intuitive intuition of disentanglemed representation. If we denote as $Fovs = \{FoV_1, FoV_2, ..., FoV_i, ..., FoV_n\}$ the real factors of variation of an input image x, and with $\mathcal{Z} = \{z_1, z_2, ..., z_i, ..., z_n\}$ the latent dimensions of a latent space $\mathcal{Z}$, a perfectly disentangled representation can be though of a **bijective** function $f : \mathcal{Z} \to FoV$, being it both:

- **injective**: $\forall z_i \neq z_j \implies f(z_i) \neq f(z_j)$
- **surjective**: $\forall \text{FoV} \in \text{FoVs} \exists z_i \in \mathcal{Z} : f(z_i) = FoV_i$

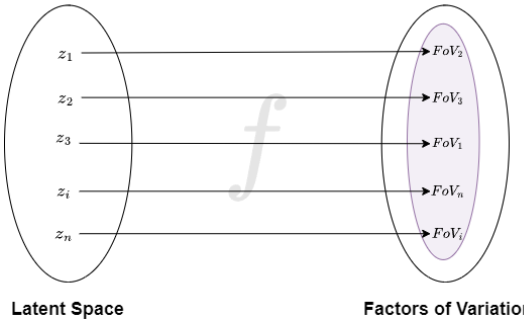


Figure 2.1: Formal connection between a *disentangled representation* and a *bijective function*. The source set $\mathcal{Z}$ is the set of the latent dimensions of the latent space, whereas the destination set represents the data FoVs.

Thus, a not perfectly disentangled representation is not injective, meaning that many latent dimensions are encoding the same FoV, and is not surjective, meaning that there are some FoV not captured by any latent dimensions. Instead, in an ideal disentangled representation there is a one-to-one mapping between each z_i with each FoV_i; this exactly defines a bijective function.

This parallelism with the bijectiveness of a function is essentially captured by the rationale behind the DCI disentanglement metric [7], described in detail in section 4.4, explained by figure 4.12. Briefly, we anticipate that DCI measures three terms separately; among them:

- *disentanglement*: measures the degree to which a representation factorises or disentangles the underlying factors of variation, with each variable (or dimension) capturing at most one generative factor.
- *completeness*: measures the degree to which each underlying factor is captured by a single code variable

Summing up, we can conclude that in a disentangled representation the underlying factors of variation are encoded in separate yet independent latent variables. This indicates that each aspect, such as object identity, shape, or color of the input data is depicted in a dedicated semantically-meaninfgul dimension. This property could allow for a clearer control over the latent space, and more importantly to a transparent understanding and partitioning of independent latent variables.

Disentanglement is a crucial property for latent spaces because it permits an easier comprehension of the underlying structure of data and gives one a foundation for analyzing its behavior. Disentanglement facilitates data manipulation and interpretation, improving the understanding of the underlying mechanisms and facilitating the discovery of causal relationships among variables. Disentangled representations also make it easier to transfer knowledge between tasks and can improve generalization abilities. Disentanglement is therefore a crucial and favorable characteristic of representations, which can enhance overall performance and encourage the creation of more sophisticated machine learning algorithms.

Although, as we will see, the learning of disentangled representation is a complicated task. The reference models that foster such learning are built up from Variational Autoencoders, and will be discussed in the next section.

2.3. Unsupervised Disentangled Representation Learning

In this chapter we will discuss the main deep neural networks adopted for disentangled representation learning. In this section we will focus only on the basic unsupervised methods. In the next section 2.4 we will showcase the main limitations of these methods, and discuss the main solutions adopted.

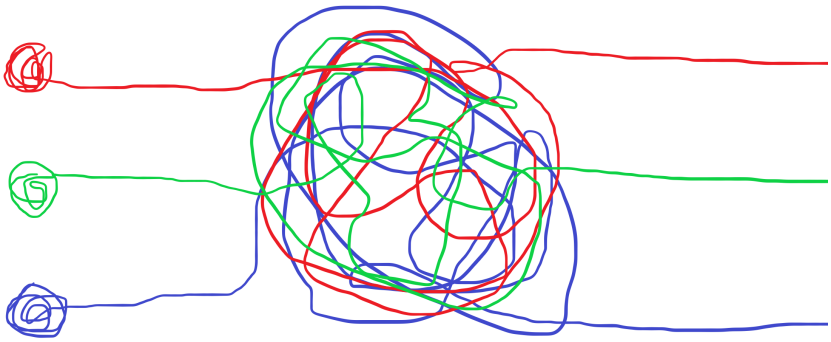


Figure 2.2: Intuitive visualization of the *disentangled representation learning* process. In this trivial example, each color represents a FoV of the input data, that after an encoding process should be maintained in separate latent dimensions.

2.3.1. Autoencoders

Autoencoders (AE) are neural networks mainly used used for data compression, feature extraction, and dimensionality reduction. The basic idea behind autoencoders is to learn a compressed representation of the input data by training the network to reconstruct its original input. They are composed of an Encoder network E and a Decoder network D. The encoder's goal is to embed the input x in a lower dimensional representation $z = E(x)$ which should extract its most relevant features. The decoder's goal is the opposite, reconstructing the original higher dimensional input x starting from the latent representation z: $y = D(z) = D(E(x))$. The latent space of a linear Autoencoder is quite similar to the eigenspace obtained from the principal component analysis (PCA) of the data. By utilizing non-linear activation functions, Autoencoders may learn relatively potent representations of the input data in lower dimensions with far less information loss. Therefore, the overall goal is to tradeoff the reconstruction quality with the embedding bottleneck to reach the optimal reconstruction. To do so, the loss function to be minimized

is merely an L2 pixel wise reconstruction loss:

$$loss_{ae} = \|x - \hat{x}\|_2 = \|x - D(z)\|_2 = \|x - D(E(x))\|_2, \tag{2.1}$$

Being trained in this way, by just trying to reconstruct the input at output, no assumptions can be made on the latent space distribution of Autoencoders. Not every sampled point of the latent space will result in a meaningful output, because there could exist regions of the latent space that don't correspond to any input data point. We refer to this as the latent space not being regularized, as can bee seen in Figure 2.4a. Therefore, since the generating capability does not hold over the full latent space, vanilla Autoencoders don't achieve consistent results as generative models.

2.3.2. Variational Autoencoders

Variational Autoencoders (VAE) [19] have been proposed to cope with the issue of non regularized latent space. Its encoder E, parametrized by a neural network $q_\phi(z|x)$, instead of outputting a latent vector, outputs the parameters of a Gaussian distribution, namely the mean μ and the standard deviation σ, for every latent variable z_i. After sampling the latent vector $z \sim \mathcal{N}(\mu, \sigma^2)$, the decoder D, parametrized by another neural network $q_\phi(x|z)$ is fed with it in order to reconstruct the input. The architecture can be seen in Figure 2.3.

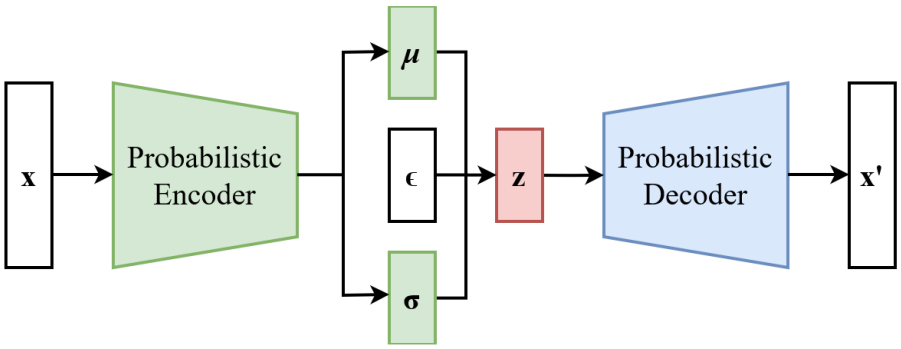


Figure 2.3: Variaitonal Autoencoder architecture, from Wikimedia Commons

The objective function is modified to force this latent distribution to be close to a normal one. This goal is achieved by adding a second term in the loss function, which is the KL Divergence term between the latent distribution and a normal Gaussian, with zero mean and unit variance. This term contributes to the loss by keeping the space regularized exactly by forcing each latent variable to be close to a zero-centered Gaussian. In this

way, we make sure that the latent space is evenly spread out and no significant gaps between clusters exist, as one can notice in Figure 2.4a.

Since the parameters of the latent distribution are sampled in the latent vector, back-propagation couldn't be performed through the full network in an end-to-end manner since we can't trace back errors due to this random sampling. To solve this issue a simple reparametrization trick is adopted:

$$z = \mu_x + \epsilon \cdot \sigma_x, \qquad \epsilon \sim \mathcal{N}(0, \mathbb{I}), \tag{2.2}$$

With this simple idea, the randomness of sampling is confined to the ϵ multiplicative term, and the propagation of the error can happen through the whole network. Here is the final loss function to be minimized:

$$\mathcal{L}_{vae} = \underbrace{E_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{reconstruction loss}} - \underbrace{D_{KL}(q_\phi(z|x)||p(z))}_{\text{KL Divergence}} \tag{2.3}$$

where $p(z)$ is chosen to be $\mathcal{N}(0, \mathbb{I})$

A nice property to note is that when data are sampled from a region with overlapping clusters, we get morphed data between the clusters of interest, and the resulting transition is smooth when passing from one another.

2.3.3. Beta Variational Autoencoders

Beta Variational Autoencoders (β-VAE) are a modification of Variational Autoencoders that put more emphasis on the regularization term. The β hyperparameter controls the balance between the reconstruction error and the capacity of the latent representation to capture independent and informative features. β-VAEs simply add a constant multiplicative term, β, to the Kullback-Leibler divergence loss to achieve this. The aim of β-VAEs is to maximize the probability of generating real data while also ensuring that the latent distribution term is close to a uniform one.

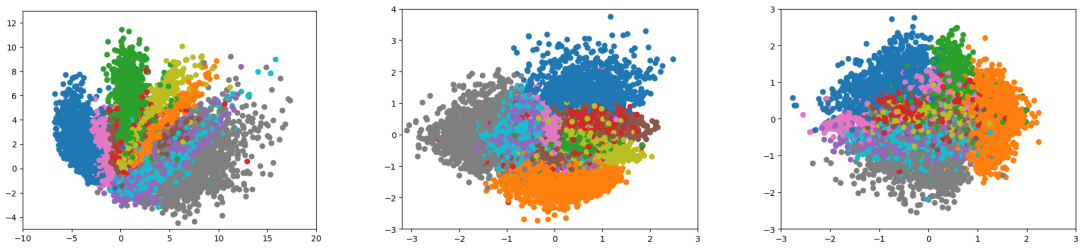
By doing this, we are actually enhancing the disentanglement property of our latent space. With β>1 we are strengthening the KL term, thus we aim to make our latent data distribution closer to an Isotropic Gaussian whose covariance matrix is unit. An unit covariance matrix is indeed forcing the latent distribution to be factorized in independent components, forcing covariance between dimension to be zero, thus enhancing disentanglement. So an higher β encourages the representation to be more efficient and supports disentanglement even further. By penalizing the model for encoding multiple factors of variation

in the same latent variable, β-VAEs promote disentanglement and encourage the model to assign different dimensions to different features. However, setting a high value of β might require a trade-off between the degree of disentanglement and the quality of the reconstruction. The resulting loss function to be minimized is then trivial:

$$\mathcal{L}_{\beta vae} = \underbrace{E_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{reconstruction loss}} - \beta \cdot \underbrace{D_{KL}(q_\phi(z|x)||p(z))}_{\text{KL Divergence}} \tag{2.4}$$

Note that when β=1, we reduce to the VAE case.

As we can notice in Figure 2.4c, the constraint on the regularization term of a β-VAE pushes the latent space to be even more uniformly distributed. In the AE latent space the clusters are defined but there are significant gaps among them, while in the VAE is clearly more uniform and the unit sphere is basically completely covered. The β-VAE does the same thing but going stronger in the regularization, as we can deduct from the more rounded shape. It's important to notice that the VAE and beta-VAE latent space is zero centered with an almost unit variance



(a) AE's latent space. (b) VAE's latent space. (c) β-VAE's latent space.

Figure 2.4: Visualizations of the latent spaces of different autoencoders, respectively of an AE (fig.A), of a VAE with β=1 (fig.B) and a β-VAE with β=4 (fig.C) when trained with the digit MNIST dataset. We can notice that the KL-loss increasingly regularizes the space by keeping it closer to an uniform distribution.

2.3.4. ELBO Surgery

Studies on disentanglement have continued to develop and flourish over the years. In particular, the VAE objective function received a lot of interests. In the ELBO Surgery [12] the authors argued that the KL divergence term could be further factorized in two

independent objectives:

$$\mathbb{E}_{p_{data}(x)}[D_{KL}(q(z|x)\|p(z))] = \underbrace{I(x;z)}_{\text{Mutual Information}} + \underbrace{D_{KL}(q(z)\|p(z))}_{\text{Actual KL divergence}} \tag{2.5}$$

where $I(x;z)$ is the mutual information between x and z under the joint distribution $p_{data}(x)q(z\|x)$. while the other term is the actual KL divergence term with respect to the prior $p(z)$.

In this work the authors claimed that in βVAE [10], by making $\beta>1$, the original KL term along with favoring disentanglement was also deteriorating the reconstruction capabilities of the network, by penalising $I(x;z)$, thus reducing the amount of information about x stored in z. In this new formulation, penalizing $D_{KL}(q(z)\|p(z))$ pushes $p(z)$ towards $q(z)$, forcing independence among z dimensions. In this way we are separating the Mutual Information between x and z that we aim to maximize.

2.3.5. Factor Variational Autoencoder

One development that followed the ELBO Surgery [12] decomposition was the Factor Variational Autoencoder (FactorVAE) [18]. In this work, the authors modified to standard VAE loss function (Section 2.3) dividing the KL Divergence in two separated terms, finally adjusting them with dedicated parameters:

$$\mathcal{L}_{\text{factorvae}} = \underbrace{E_{q_\phi(z|x_i)}[\log p_\theta(x_i|z)]}_{\text{reconstruction loss}} - \lambda \cdot \underbrace{D_{KL}(q_\phi(z|x_i)||p(z))}_{\text{Mutual Information}} - \gamma \cdot \underbrace{D_{KL}(q_\phi(z)||\overline{q}(z))}_{\text{actual KL divergence}} \tag{2.6}$$

In this way the terms are independent from one another and the trade-off between the reconstruction loss and the KL Divergence with respect to the prior become more manageable.

2.4. Supervised Disentangled Representation Learning

All the methods mentioned in section 2.3 are totally unsupervised. This means that no additional information other than the images themselves is shown to the model during training. The achieved disentanglement in those methods only comes from the statistical properties that the losses carry. However, the only dataset over which those methods evaluated disentanglement are toy datasets, principally *DSprites* [26] and some modifications of it, treated in section 4.2.1.

The achievement of effective disentanglement in these models has to be attributed mainly to the exploitation of the inherent biases of the toy datasets they've been trained on. This is because the generative model of the dataset is precisely aligned with the underlying data FoVs, thus making the framework not generalizable. So as *Locatello et al.* argue in [22], it is essentially impossible for disentangled representation learning to capture the desired properties without exploiting inductive biases or adding an explicit supervision. In fact, relying solely on unsupervised approaches falls short of providing a meaningful isolation of semantic generative factors, especially on real-world dataset. Below we showcase some of the works that introduced some kind of supervision to enhance the disentanglement level.

2.4.1. Multi Level VAE

Bouchacourt et al. proposed Multi-Level Variational Autoencoder (ML-VAE) [3], a non-adversarial approach to disentangle factors of variation based on group-level supervision. ML-VAE is a novel probabilistic model designed to learn a disentangled representation from grouped observations. It works on both group and observation levels, separating latent representations into meaningful components such as style and content. This disentanglement process is facilitated through a grouping operation where samples within the same group share the same content but can differ in style. During encoding, style variations are naturally captured in one part of the latent code while content, shared within groups, is represented in a different part. The grouping operation also enhances content certainty when multiple samples are present within a group. Notably, the grouping operation doesn't require prior knowledge of the groupings' meaning; it only relies on the organization of data into groups. This enables the ML-VAE to achieve semantically meaningful disentanglement.

2.4.2. Group VAE

The GroupVAE [13] method introduces a framework that utilizes supervision to learn representations that are disentangled by groups. It works with paired observations that have k=1 shared underlying factors of variation. While ML-VAE [3] uses a product of the posteriors, GroupVAE employs an empirical average of the parameters of the approximate target posteriors to disentangle the FoVs. As in ML-VAE, the framework consist in showing at each iteration pairwise observations that shares one factor to the model. After having obtained the encoded latent representation of both images within the pair, the two latents z_1 and z_2 are modified with an aggregate function a:

$$\begin{cases} \tilde{q}_\phi(\hat{z}_i|x_1) = a(q_\phi(\hat{z}_i\|x_1), q_\phi(\hat{z}_i\|x_2)) & \forall i \in FoVs \\ \tilde{q}_\phi(\hat{z}_i|x_1) = q_\phi(\hat{z}_i\|x_1) & \text{otherwise} \end{cases} \tag{2.7}$$

where a in this case in a plain average.

In this way it is forcing the target dimension to encode the shared FoV within the pair.

2.5. Generative Adversarial Networks

Generative Adversarial Networks [8] (GAN) are deep neural networks that have represented the state of the art in image generation for many years. The architecture is summarized in Figure 2.5.

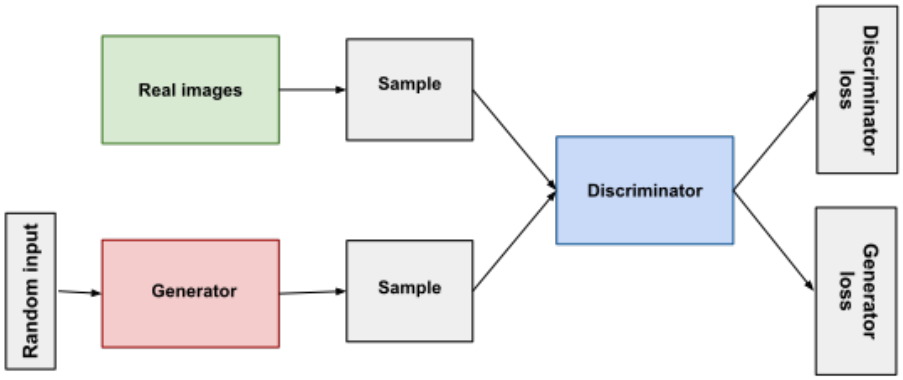


Figure 2.5: GAN architecture, from googledevs.

They consist of two separate networks, the generator G and the discriminator D. The generator represents a differentiable function $G(z, \theta_g)$, where θ_g are the parameters of the generator network, which takes a randomly sampled latent vector z from an initial data distribution $p_z(z)$ and transforms it into an image belonging to a certain distribution p_G. The discriminator instead represents a differentiable function $D(I, \theta_D)$ that takes an input image I and outputs the probability of it coming from the training probability distribution p_{data}.

The goal of the Discriminator is to distinguish images generated by G from real images coming from the training data set. Thus, its objective function is to maximise the probability of correctly classifying samples from the training data, from the generated ones. Instead, the goal of the Generator is to mimic the real distribution of the input data and to generate increasingly realistic images. These two networks are jointly trained competitively in a zero-sum game whose ultimate goal is to obtain a good estimator of p_{data}. We can think of the objective function in a game theory fashion, where players D and G compete in a min-max game. Finally, the objective function for the two players:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))], \tag{2.8}$$

In other words, the goal of the generator is to fool the discriminator by generating realistic images so that it cannot distinguish the source. In fact, the global optimum is found when p_G becomes equal to the original input training distribution p_{data}. When this is achieved, the Discriminator is fooled, and it can be discarded, while the Generator can be kept and used to create unseen realistic images coming from a distribution similar to the training one.

2.5.1. StyleGAN

In recent years, GANs has seen many developments along with huge quality improvements on the generated data. One of the most important twists in the GAN field has been carried out by NVIDIA's StyleGAN [17] architecture, which borrows from the Style Transfer [14] literature. In fact, the major intuition that made them the state-of-the-art in the generative AI field for years has been the introduction of the AdaIN layer, described in detail in Section 2.5.2.

StyleGAN's training method is builded on top of Progressive GAN's [16] one. The generator and the discriminator start from low resolution images, in this case 4x4, and once stable, they are upsampled to double its height and width, thus quadrupling the total area. In other words, it gradually generates feature maps of higher resolution as they

get upsampled by successive layers. This is said to enhance network stability and increasingly add details at different levels of resolution of the image. Another StyleGAN's peculiarity is the multiplicity of the initial latent spaces with which feed the generator. In the original GANs, a vector is randomly sampled from an uniform distribution Z and passed directly into the generator G. Instead in StyleGAN the z vector passes first into a Mapping Network M, composed of 8 fully connected layers, giving rise to another latent space W. The resulting vector w is claimed to be more disentangled since it emulates better the final data distribution, stepping away from the initial uniform Z uninformative one. Once obtained the w vector, it is passed to the Synthesis Network S, which is the actual generator responsible for the upsampling part.

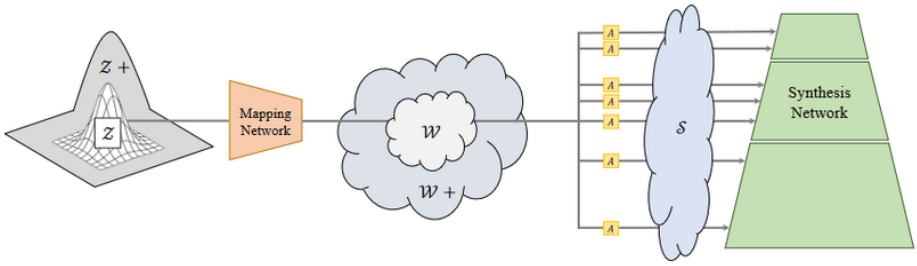


Figure 2.6: Visualization of the StyleGAN framework and of the identified latent spaces, from [27]

Some [34] claim that there exists another intermediate space S even more disentangled, which can be extracted after the affine transformations A that retrieves the style of the image.

2.5.2. Adaptive Instance Normalization

In the literature, normalizations have always been seen as training regularizers; the style transfer literature transformed its use into a style-content separator. In fact, talking about style, StyleGAN's major innovative contribution comes from the addition of the Adaptive Instance Normalization (AdaIN) [14] layer before the Upsampling one. Let's have a look at the most famous normalization techniques/layers used in deep learning frameworks, all the way to AdaIN.

Batch Normalization [15] takes as input a batch of images x and it normalizes the mean and standard deviation for each individual feature channel.

$$\mathrm{BN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta \tag{2.9}$$

Instance Normalization [33] normalizes each element x of the batch independently. It performs a form of style normalization by normalizing feature statistics, namely the mean and variance.

$$\mathrm{IN}(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta \tag{2.10}$$

So, where BN computes one mean and standard deviation, thus making the distribution of the whole layer Gaussian, IN computes n of them, making each individual image distribution look Gaussian, but not jointly.

As we can notice from both equations, the output is then rescaled by γ and shifted by β, two fixed parameters representing the mean and the standard deviation of the target style. Therefore, we may alter the output's style by adjusting γ and β in the normalization layer. The fact that in each normalization layer there is only one set of the variables limits its ability to learn more than one style.

Adaptive Instance Normalization [14] enables the network to learn diverse styles. It receives a content input x and a style input y, and it simply aligns the channel-wise mean and variance of x to match those of y. Here, γ and β are trainable vectors. AdaIN adaptively computes affine parameters from the style input y, discarding the fixed parameters γ and β with which output is scaled and shifted.

$$\mathrm{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \tag{2.11}$$

AdaIN allows for generation of realistic and varied images by controlling the its style and content in a separated way. By adjusting the mean and standard deviation of the feature maps, the AdaIN layer can transfer the statistical characteristics of one image onto another, creating unique variations of the same input image.

2.6. Semantic Facial Attribute Editing

Generating random realistic images has seen an impressive and continuously higher interest over the past few years. The main reason behind its success is the growth of Deep Convolutional Neural Network based methods. Facial images is one of the most prominent subjects targeted by these generative models. Along with facial image generation, in Semantic Facial Attribute Editing (SFAE) various applications has seen a great potential. We refer as SFAE the process of editing a single (or a few) facial attributes such as the *Nose*, *Mouth*.., while keeping the others unchanged. See Figure 2.7 for an example.

Figure 2.7: Example of Semantic Facial Attribute Editing on the attribute *Bangs*.

Various real-world applications could benefit from SFAE: enhancing the effectiveness of automatic Face Recognition systems, Face Data Augmentation in facial image processing, adding effects to filters in social media platforms, Video Editing, producing and animating faces in the gaming and animation industries etc.

As Nickabadi et al. [28] pointed out, many challenges are be faced when editing facial attributes:

- **Attributes Entanglement**: an edit in a single attribute could cause an unwanted but sometimes almost inevitable modification of another attribute. These two attributes are said to be entangled. An example could be the attributes *Young* and *Grey Hair*.
- **Low data resources**: almost all the method used to perform facial editing employs data-hungry deep neural networks. The whole knowledge about facial features comes from the training data, which are scarce and do not cover the whole human facial diversity. Therefore obviously bad results could come up from these exacts biases in the data. This limitation is one of the caused of the next one.

- **Uncommon face conditions**: when the face to be edited comes in an extreme pose or lightning condition, the edit will result more difficult. In general, many methods could be applied only on images similar to the ones in training set.
- **Evaluation metrics**: facial editing is a very challenging problem to be evaluated through objective metrics. Humans in the loop are always a viable yet limited and biased solution, while no quantitative special-purpose metric exists. We will address in detail this issue in section 4.4.

Nickabadi et al. [28] provided an insightful overview of the literature methods presented to perform Semantic Facial Attribute Editing, that can be summarized with the scheme in Figure 2.8.

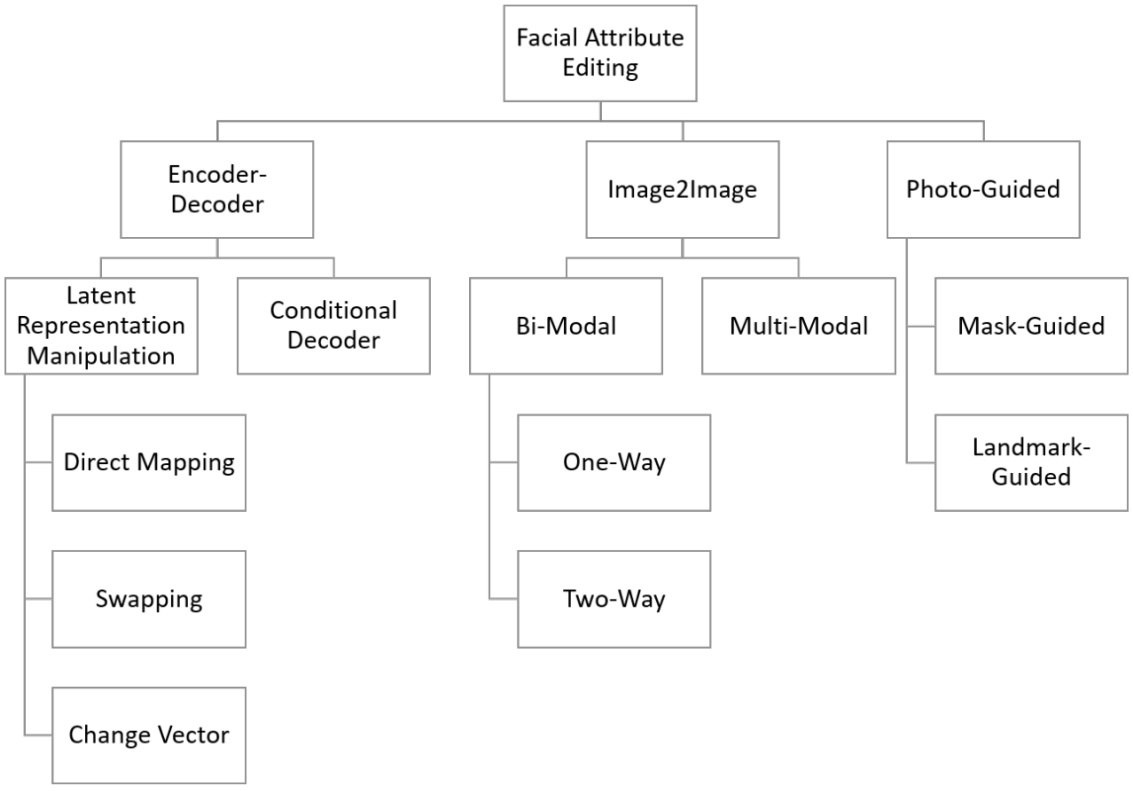


Figure 2.8: Taxonomy of the literature's methods for Semantic Facial Attribute Editing, from [28]

Below is presented a detailed outline of the first diramation level.

2.6.1. Encoder-Decoder methods

The most common and broadly adopted architecture for Semantic Facial Attribute Editing is the one that employs an encoder-decoder like framework. Generally, in this method an image I is passed through an encoder E, whose output is a latent vector z containing the most relevant features of the image I. The decoder's job is to remap the latent vector back into the image space. Here, the editing comes at the latent level; different techniques are adopted to find a direction to which modify the latent vector to obtain an edited code:

$$\hat{z} = a + \Delta direction. \tag{2.12}$$

where the target attribute vector a could be z itself or a different point in the latent space as well, depending on the method.

This latent vector is then fed again through the decoder D to obtain the edited final image. The whole innovation and differentiation in this category of methods comes in finding the direction of editing of the latent vector. Studies of the latent spaces of the various models, generally GAN's generators, lead to the different approaches. Nitzan et al. [29] have proposed a facial attribute editing model which takes two images I_{id} and I_{attr} as input and produces a face image whose identity is taken from I_{id} and its all other attributes such as pose, expression and illumination are coming from I_{attr}. This approach's limitation comes from the fact that in this case the editing direction is given by the second input image. In general it could be challenging to edit the image in the needed single direction with this method.

The state of the art for what concerns Semantic Facial Attribute Editing is represented by **InterFaceGAN** [32]. Their technique employs the labelled dataset CelebA [21], which has 40 binary labels for each data point, indicating the presence or absence of a facial attribute. In particular, separately for each attribute, they divided the whole dataset into the positive subset and the negative one, and trained a linear Support Vector Machine (SVM) to predict the best boundary that separates the two classes. Once finished training, the normal hyperplane to the decision boundary represents the direction of change of that single attribute on which the two subset have been divided into. Following this hyperplane, one can add or remove the target attribute.

2.6.2. Image to Image methods

Another set of methods to perform Semantic Facial Attribute Editing is Image to Image translation. The source and target domains correspond to sets of face image with two

different values of a specific attribute. For instance, a collection of images with *Eyeglasses* and a collection of images without *Eyeglasses* are gathered. Here the condition (attribute) vector a is no longer required as input of the generator, as it is trained to perform the same edit on all input images. In this method, models are classified into two main categories: bi-modal models and multi-modal models. *Bi-modal* models are designed to perform a specific edit on individual attributes, while *multi-modal* models are capable of manipulating multiple attributes simultaneously.

2.6.3. Photo-Guided methods

The fundamental component in these model architectures involves employing a mask to instruct the generator G to concentrate solely on the sections of the input image that might be impacted by the editing request, while leaving the rest unchanged. To achieve this, two generators, namely G_{color} and $G_{attention}$, are simultaneously trained. G_{color} 's role is to modify the input image in accordance with the intended attributes, producing the corresponding color image Ic. On the other hand, $G_{attention}$ predicts an attention mask A, which determines the contributions of both the edited and original images in generating each pixel of the output image. Ultimately, an alpha blending operation is executed on the input and edited images, guided by the attention mask, to generate the target image as follows:

$$target = \alpha \cdot I_c + (1 - \alpha) \cdot I \tag{2.13}$$

This approach eliminates the need for the generator to recreate static parts of the input image, allowing it to focus specifically on the attribute-specific regions, resulting in more realistic changes.

3 | Methods

3.1. Problem Formulation

As we discussed, explainability could be benefit from disentangled representation. In this thesis, we tackle the problem of generating an interpretable model by leveraging on the disentanglement property of its latent space.

The reference model for disentangled representation learning is the Beta Variational Autoencoder [10] described in Section 2.3.3. It is composed of:

- an encoder $q_\phi(z|x)$, that maps the input x to a latent gaussian distribution with mean and variance $\{\mu, \sigma\} = \{[\mu_1, \sigma_1], ..., [\mu_n, \sigma_n]\}$;
- a decoder $p_\phi(x|z)$ that reconstruct the input in $\tilde{x}$ after sampling the actual latent vector $z \sim \mathcal{N}(\mu, \sigma) = \{z_1, ..., z_n\}$;

being n the dimension of the latent space.

If we denote as $\mathcal{F} = \{f_1, ..., f_n\}$ the data factors of variations, z is said to be perfectly disentangled when there exists a one-to-one mapping between $\mathcal{F}$ and the latent vector z, i.e. where a change in each latent dimension z_i affects only a factor f_i.

Reaching a satisfactory level of disentanglement is essentially impossible without any supervision of inductive bias in non-toy datasets [22]. In particular, when dealing with non-toy datasets, its FoVs are usually unknown a priori, since the data generation process is unavailable or too complicated to deal with. Since with the increase of data complexity unsupervised methods struggle to build efficient latent representation that are non-redundant and disentangled, our goal is to guide the latent representation providing a supervision to enhance the model's disentanglement, by guiding it in encoding each semantic data FoV f_i in a dedicated dimension z_i of the latent space z.

In the following, we discuss at first a proposed method **E2Editor** to generate the required supervision in a selected real-world dataset, and then our **SVAE** framework to enhance

the latent space disentanglement.

3.2. Supervised VAE

We here present SVAE, our framework to provide a supervision to the Beta Variational Autoencoder [10] in order to enhance its latent space disentanglement.

3.2.1. Settings

Our method exploits the supervision given from paired observation to force the disentanglement in the latent space of a Beta Variational Autoencoder. By forcing disentanglement, we are actually guiding the SVAE to encode different factors of variation (FoV) in separate latent dimensions. Our framework requires paired observations of the form:

$$\{x_1, x_1\}_f \tag{3.1}$$

such that each x_2 differs from each x_1 in a single FoV f.

In the paired images below we showcase examples of the non-shared FoVs and of the relative pairs:

Figure 3.1: Paired observation sample from *Dfaces* and *SDSprites*, described in sections 4.3.2 and 4.3.1. The first couple's non-shared FoV is *Smiling*, while for the second is *Shape*

From the image above it should be clear what we mean by "we treated the facial attributes as FoV" of our facial dataset. They are the factors that best defines the images, and over which we decided to put the variability.

From the difference of the two images we help the model the understand the variation in the image, in order to confine it to a dedicated latent dimension.

3.2.2. Architecture

Below we showcase the architecture of our SVAE model.

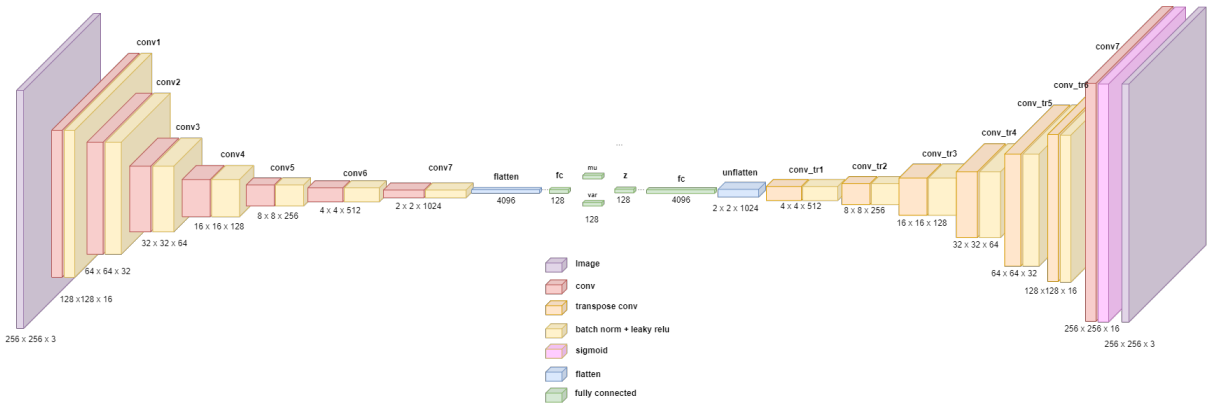


Figure 3.2: SVAE architecture.

The architecture of both Encoder and Decoder follow the traditional architectures of Convolutional Neural Networks. The usual convolutional block is adopted, formed of a Convolutional Layer, Batch Normalization Layer and finally an Activation.

After the dowsampling phase performed by the Encoder, a Flatten layer is used. Since we are in a variational framework, two fully connected layers connects the mean and the variance of the latent representation. Then a sampling is applied to generate the intermediate latent vector, and the reparametrization trick [19] is used to make the network trainable.

The Decoder part mirrors the Encoding, using Transposed Convolutions instead of Convolutions, to reconstruct the original image.

As last activation function has been chosen a Sigmoid, since it maps values in the range [0,1], which resembles probability values.

3.2.3. Method

Our method's goal is to force disentanglement, by encoding in separate latent dimensions different factors of variation.

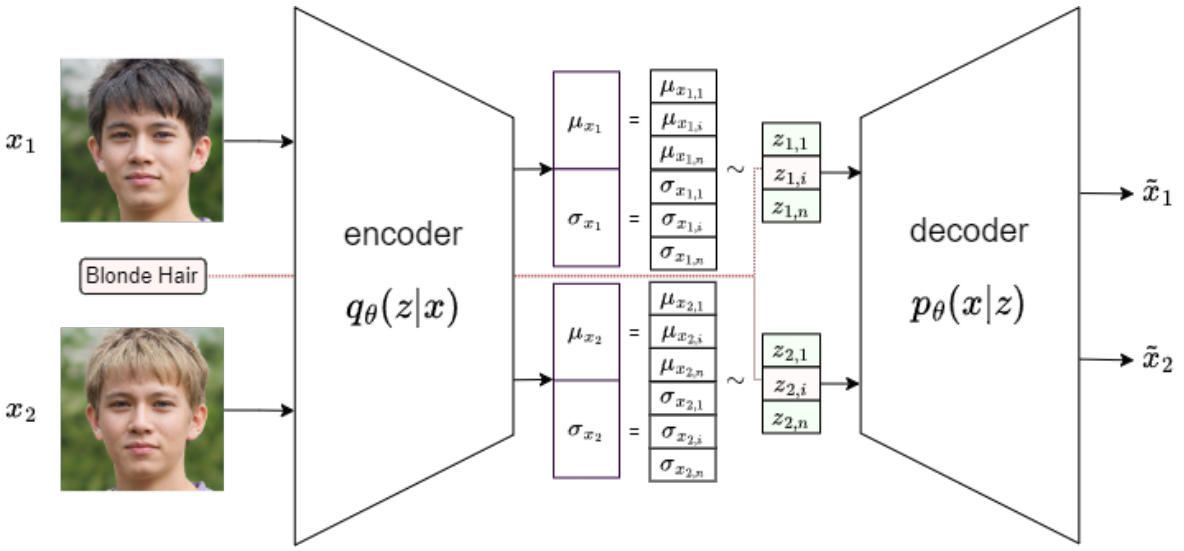


Figure 3.3: Our SVAE framework to enhance the latent space disentanglement with paired observations. Input images x_1 and x_2 have one non-shared factor f (*Blonde Hair* here), whose latent dimensions z_{1_i} and z_{2_i} have to be pushed apart, forcing in that dimension the encoding of factor f.

As we can see from Figure 3.3, S-VAE accepts paired observations (x_1, x_2) that differs in a single FoV, it encodes them into mean and variances components, respectively μ_{x_1}, σ_{x_1} and μ_{x_2}, σ_{x_2}, and with the reparametrization trick z_1 and z_2 are sampled to form the reconstructed $\hat{x_1}$ and $\hat{x_2}$:

It works like a normal Variational Autoencoder, but the paired observations play a role when computing our custom loss function. We modified the plain βVAE's loss function 2.4 adding to the reconstruction loss and the KL divergence a new term, namely the *pair loss* (or disentanglement loss). It's composed of two sub-losses:

- **target loss**: this loss is responsible for pushing away the target FoV, i.e. the non shared factor of variation between x_1 and x_2. After identifying the dimension that should encode that target FoV (more on this choice later), a mean squared error loss is employed to distance the two latent variables. Since we aim to maximize this distance, we put a minus term in front.
- **non-target loss**: this loss mirrors the target one, since it forces a small distance

among the others non-target dimensions. For this reason, a plus sign in front is needed to minimize this term.

As an example, we take the paired observations of Figure 3.1: in this cases the $loss_{target}$ should force respectively the difference in the *Smiling*/*Shape* FoV, while the $loss_{non\_target}$ should keep the others FoVs similar, for example the orientation, size, positionX and positionY in the second pair.

The resulting loss:

$$loss_{svae} = \underbrace{\|x_1 - \hat{x_1}\|_2 + \|x_2 - \hat{x_2}\|_2}_{\text{reconstruction loss}} + \underbrace{\beta \cdot D_{KL}(\mathcal{N}(\mu_{x_1}, \sigma_{x_1})\|\mathcal{N}(0, \mathbb{I}))\beta \cdot D_{KL}(\mathcal{N}(\mu_{x_2}, \sigma_{x_2})\|\mathcal{N}(0, \mathbb{I}))}_{\text{KL divergence}}$$
$$\underbrace{\underbrace{-\gamma \cdot (z_{1_{target}} - z_{2_{target}})^2}_{\text{target loss}} + \underbrace{\eta \cdot \frac{1}{N-1} \sum_{k \neq target}^{N} (z_{1_k} - z_{2_k})^2}_{\text{non target loss}}}_{\text{pair loss}} \tag{3.2}$$

in which x_1, x_2 are the input pair data, μ_{x_1}, σ_{x_1} and μ_{x_2}, σ_{x_2} are respectively their mean and variances components, while *target* is the dimension of the latent vector in which we aim to encode the FoV that isn't shared between the paired observations. In the pair loss term, z_1 and z_2 are their derived latent vectors obtained with the reparametrization trick:

$$\begin{cases} z_1 = q_\phi(z_1|x_1) = \mu_1 + \epsilon \cdot \sigma_1 & (3.3) \\ z_2 = q_\phi(z_2|x_2) = \mu_2 + \epsilon \cdot \sigma_2 & (3.4) \end{cases}$$

Both sub-losses are weighted by hyperparameters γ andη.

The heuristic for selecting the latent dimension responsible for the encoding of the tg FoV is to choose the one with the highest KL divergence:

$$\text{argmax}\left(\frac{\sigma_1}{\sigma_2} + \frac{(\mu_2 - \mu_1)^2}{\sigma_2} - 1 + (\log \sigma_2 - \log \sigma_1)\right) \tag{3.5}$$

With this ulterior loss, we expect the model to grasp the difference between each given pair and to be able to generalize and confine the span of variation of the given non-shared factor to a single latent dimension. Once trained the model, the desiderata is visualizing that the latent traversals over each dimension *target* moves ideally only the correspondent semantic FoV.

3.3. E2Editor

Since our focus wants to be real-world datasets, we below present our contributed method to perform Semantic Facial Attribute Editing, in order to generate a facially disentangled dataset to be used as a supervision for our SVAE of Section 3.2.

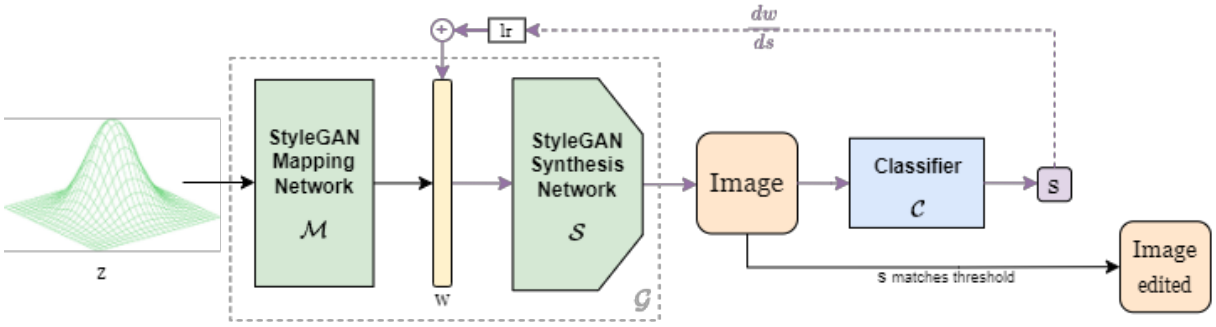


Figure 3.4: *E2Editor*'s architecture to perform Semantic Facial Attribute Editing.

3.3.1. Architecture

As we can notice in figure 3.4, the full pipeline is built from two main principal components: a pretrained StyleGAN and an Attribute Classifier.

A **StyleGAN** Generator G pretrained on CelebA dataset [21] is employed as backbone Generator. Being a StyleGAN, it can be divided into two separate components, namely a Mapping Network M and a Synthesis Network S. M is an 8-layer fully connected neural network that takes as input z, a latent vector randomly sampled from a normal Gaussian distribution, and outputs another vector w of the same dimensionality (512 in this case). This two vectors z and w respectively identify the two latent spaces Z and W. Space Z is conceptually simpler and easier to sample from, since it follows a simple Gaussian distribution, but it doesn't exhibit a high degree of disentanglement, which is a desirable property when operating a facial edit with an encoder-decoder method (See section 2.6.1). Instead, W is designed to be disentangled to a higher extent compared to space Z. One of the key promoter is the orthogonality regularizer: M encourages the dimensions of space W to be orthogonal or uncorrelated. The model is thus incentivized to assign separate semantics to each dimension by reducing the inner product between different dimensions, thus promoting disentangled representation learning. For this reasons, the space Z is only used to sample the first vector that is going to give the initial base image I to be edited. Instead, the correspondent vector w is the one over which the modification will

be iteratively performed. Once edited the vector w, it is passed through the Synthesis Network S to be mapped into the image space. Therefore by controlling the vector w we can obtain different images I, once passed through S. In this work has been used Anycost GAN's implementation [20]. It's worth a mention the fact that the StyleGAN network's weights have been kept freezed during the whole editing process.

For the *E2Editor* method, an **Attribute Classifier** is needed to compute the edit in the target attribute direction for the w vector mentioned above. The attributes considered for this task are the 40 attributes on which the CelebA [21] dataset's images have been labeled on; namely: $5\_o\_Clock\_Shadow$, $Arched\_Eyebrows$, $Attractive$, $Bags\_Under\_Eyes$, $Bald$, $Bangs$, Big_Lips, Big_Nose, $Black\_Hair$, $Blond\_Hair$, $Blurry$, $Brown\_Hair$, $Bushy\_Eyebrows$, $Chubby$, $Double\_Chin$, $Eyeglasses$, $Goatee$, $Gray\_Hair$, $Heavy\_Makeup$, $High\_Cheekbones$, $Male$, $Mouth\_Slightly\_Open$, $Mustache$, $Narrow\_Eyes$, No_Beard, $Oval\_Face$, $Pale\_Skin$, $Pointy\_Nose$, $Receding\_Hairline$, $Rosy\_Cheeks$, $Sideburns$, $Smiling$, $Straight\_Hair$, $Wavy\_Hair$, $Wearing\_Earrings$, $Wearing\_Hat$, $Wearing\_Lipstick$, $Wearing\_Necklace$, $Wearing\_Necktie$, $Young$. Some trials have been carried out to train solid binary classifiers, one for each of the above attribute, indicating its presence or absence. An initial network composed of 8 stacked traditional convolutional blocks (Conv + ReLU + Pooling + BatchNorm) followed by a Sigmoid activation has been trained from scratch over the CelebA-HQ dataset. Other tries has been done with transfer learning: using some Resnet50-like architecture [9] pretrained on Imagenet [5] or VggFaces [4]. Both have been trained by freezing the weights of the feature extractor part, while activating the remaining classification layers' weights. Both these methods reached over 90% accuracy both in training and validation mode over CelebA-HQ dataset. Even an holdout testing set has been kept apart, on which both models scored over 90%. However, the issue with these models was that they didn't manage to generalize well on other facial images other than CelebA samples. In particular, our goal was to perform facial edits on StyleGAN's generated images, therefore robust classifiers with high accuracy were requested when testing on those images. Plenty of works [30], [20], [1] recurred to GAN inversion to train an encoder that maps images directly into StyleGAN's latent space. In this way the edits could start from a chosen image, maybe exhibiting a needed attribute presence knowing its annotations. This could be beneficial to the overall edit, but it also clearly affects the reconstruction quality due to the encoder bottleneck. Because of the scarcity of massively annotated datasets consisting of deep generated images, our generalization capability couldn't be improved signifincantly, even recurring to data augmentation or hyperparameters adjustments. In the end, a single pretrained massive Attribute Classifier has been employed in this work, from [20]. It has

been trained on CelebA-HQ and predicts 80 logits labels, each indicating the presence and absence of every of the 40 attributes

3.3.2. Method

E2Editor is an end-to-end Semantic Facial Attribute Editing method whose goal is to modify a/some facial attribute(s) while keeping the others unchanged. An example of edit can be found in Figure 3.5.

Figure 3.5: Steps of an edit example on the attribute *Beard*

The main two components needed for this method are an Attribute Classifier (C), and a StyleGAN generator (G), composed in turn of a Mapping Network (M) and a Synthesis Network (S). Let's start with defining the overall pipeline.

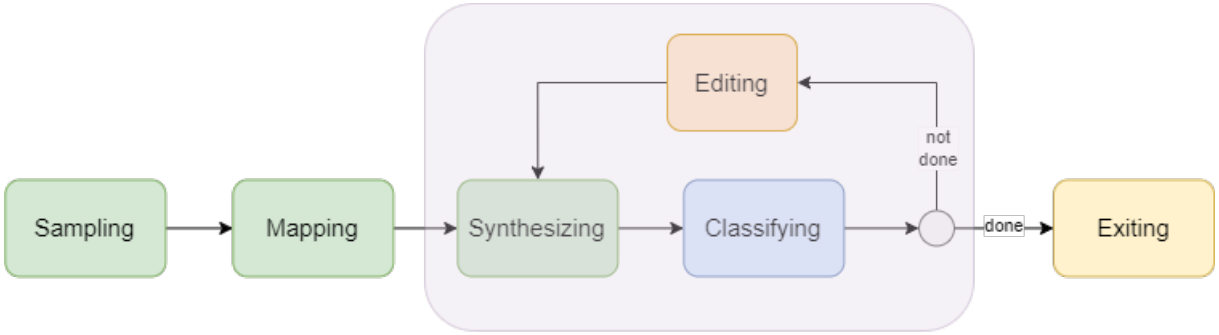


Figure 3.6: *E2Editor*'s pipeline to perform an edit

First we need to generate a random image: a sampling from a Gaussian distribution is performed to extract a vector z. It is then mapped through M into a w vector, which is claimed to exhibit more disentanglement. From now on, the network will work on this base w vector. Passing w through S, the base facial image I is then created.

At this point the editing loop begins: it terminates when the classification score s of the needed attribute a (s_a) meets the chosen condition. The condition is met when the s_a is below or above the threshold, as explained hereafter. If the sampled image's ($I_{initial}$) classification score is below 0.5, then the goal becomes the maximization of the attribute

presence and consequently the threshold is set to 0.9. Otherwise if s_a is above 0.5, we assume the attribute being present in the $I_{initial}$; accordingly we need to minimize its presence, thus the threshold is set to 0.1. Here is a formalization of the threshold t:

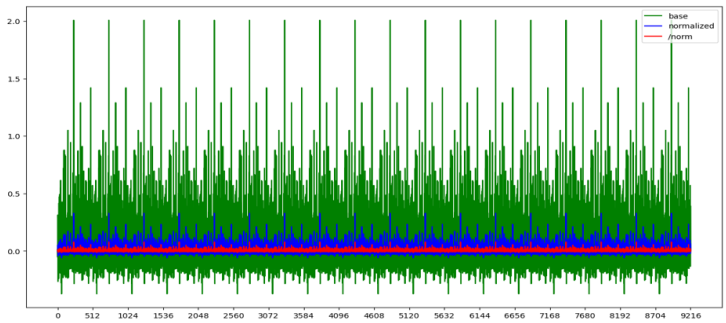
$$t = \begin{cases} 0.9, & \text{if } s_a \leq 0.5 \\ 0.1, & \text{otherwise} \end{cases} \tag{3.6}$$

At each iteration, an inplace modification of the vector w is performed; precisely:

$$w_i = w_{i-1} \pm lr \cdot \frac{\nabla w}{\|\nabla w\|} \tag{3.7}$$

Here ∇w represents the gradient of w, which provides the real direction along which modify the vector w to maximize the presence/absence of attribute a in the correspondent I_{edited}. The gradient gets populated because of the interdependence among w, $I_{edited} = S(w)$ and $c = C(I_{edited})$ after computing the classification score of the newly edited w.

Figure 3.7: Examples of gradient's normalizations



Moreover, different normalization of the gradient vector has been experimented. In figure 3.7 we showcase two different types against the unnormalized version (the green). First of all, being w and ∇w (512x18)-dimensional vectors, the flattened version are showed for visualization purposes. In blue we can see a standardized version of ∇w obtained by removing the mean and scaling to unit variance, so that the vector will have mean value 0 and standard deviation of 1. Instead in red we can see the gradient vector divided by its norm, thus obtaining an unit norm gradient vector. Empirically we noticed that using different normalizations yields no difference in the editing mechanism, even not using any of them is fine. The unit norm gradient has been chosen so that each edit has the same magnitude; in this way the number of iterations used to perform an edit that reachs the desired threshold can be used as metric when comparing results coming from different λ values, as we will see later.

Moreover, in the formula 3.7 lr stands for learning rate, which is a tunable parameter that specifies the magnitude of the step in the direction pointed by the gradient. Experiments

have been carried out to define a learning rate scheduler, but none of them proved to be qualitatively better than the others; a constant lr has been finally opted for. The sign indicates the objective function goal, whether we have to maximize or minimize it due to the initial s_a, as discussed in 3.6. It's also the motivation of the block *dir* in figure 3.2.

The pseudo code of the *E2Editor* method is illustrated in Algorithm section 3.3.2:

Algorithm 3.1 e2edit

Input: stylegan generator G, attribute classifier C, attribute a, loss' weighting factor λ, threshold t, learning rate lr
Sample $z \sim \mathcal{N}(0, \mathbb{I})$
$w, I_{initial} \leftarrow G(z)$
$s \leftarrow C(I_{initial}, a)$
if $s \leq 0.5$ **then** $direction = 1$ **else** $direction = -1$
while s doesn't reach t **do**
    $loss_{target} = bce(y_{target}, t)$
    $loss_{others} = bce(y_{others}, p)$
    $loss_{e2editor} = \lambda \cdot loss_{target} + (1 - \lambda) \cdot loss_{others}$
    $loss.backward()$
    $w \leftarrow w + direction \cdot lr \cdot w.grad$
    $I_{edited} \leftarrow G(w)$
    $s \leftarrow C(I_{edited}, a)$
end while
Output: $I_{initial}$,I_{edited}

3.3.3. Loss Function

The backpropagation step happens after this custom loss function is calculated:

$$\mathbf{loss_{e2editor}} = \lambda \cdot \mathbf{loss}_{\boldsymbol{target}} + (\mathbf{1} - \lambda) \cdot \mathbf{loss}_{\boldsymbol{others}} \tag{3.8}$$

where

$$\boldsymbol{loss_{target}} = bce(y_{target}, p) = -y_{target} \log(p) - (1 - y_{target}) \log(1 - p) \tag{3.9}$$

$$\boldsymbol{loss_{others}} = bce(y_{others}, p) = -y_{others} \log(p) - (1 - y_{others}) \log(1 - p) \tag{3.10}$$

where *bce* stands for binary_cross_entropy, y_{target} is 1 or 0 indicating the classification

score we aim to reach for the current input data point, y_{others} is the initial classification score of the others (non targets) attributes for the current input data point, and p is the softmax probability for the current input data point.

The $loss_{target}$ term is the loss responsible for bringing the edit in the direction of changing the target attribute a. Instead $loss_{target}$ term is the loss responsible for keeping the others non-target attributes unchanged, by maintaining them at the same classification score of the $I_{initial}$.

In the overall loss $loss_{e2editor}$, the λ weighting factor plays a crucial role: it manages the trade-off between the disentanglement quality of the edit and the number of iterations taken, i.e. the quickness. Its value can span in the range $(0, 1]$; the value 0 is not considered since no learning in the target attribute direction would happen. Let's look at the extreme cases:

- **low** λ: with an λ value equal to 1, the $loss_{e2editor}$ would be reduced to the $loss_{target}$. In this way the edit will end up not caring about the maintenance of the non-target attributes, but only pushing towards the target attribute edit. The edit will result fast, taking few iterations, but in general it could result not particularly disentangled due to the entanglement of the attributes with respect to the target one. In other words, all the non-target attributes are free to change proportionally to their entanglement degree with respect to the target one, without the extra regularizer loss.
- **high** λ: with a low λ value equal for example to 0.1, an higher importance is given to the $loss_{others}$, which will result in a more precise but slower edit. The gradient steps would be shorter, but significantly more accurate in not trying to perform unwanted modifications to non-target attributes.

With intermediate λ values, both the $loss_{target}$ and the $loss_{others}$ contribute to the overall $loss_{e2editor}$. As explained above, the trade-off could be managed by tuning this λ parameter. Experiment showed λ=0.5 being a good fit, guaranteeing a relatively fast and nice quality edit.

Figure 3.8 is a clear example of the lambda trade-off.

(a) Base sampled image. (b) Edit on *Blonde* (λ=1) (c) Edit on *Blonde* (λ=0.1)

Figure 3.8: The λ trade-off explained: the edit (3.8b) with λ=1 unwantedly modifies also *Beard*, whereas the edit (3.8c) with λ=0.1 keeps the *Beard* desirably unchanged.

It's clear to see the crucial role player by the $loss_{target}$ that doesn't let the non-target attributes vary. In this case the *Beard* attribute is the one that undergoes an undesirable variation when λ=1, while λ=0.1 manages to maintain it at its original value.

3.3.4. Discussion

Being a deep generative model, we expect StyleGAN's latent space W to be well regularized. Close latent space's regions should exhibit similar facial characteristics, resulting in visually smooth morphs when traversed. No meaningful edits would occur if this condition was not met. In general we also expect the space to be clustered by attributes, so that within each cluster reside facial images having the same attribute. It's then interesting to inspect the edit behaviour when "entering/exiting a cluster", i.e. respectively adding or removing an attribute. As discussed before, *E2Editor* is a **two-way** Semantic Facial Attribute Editing: it works both in case of adding a target attribute, in case its classification score is initially low, but also in removing it in the complementary case.

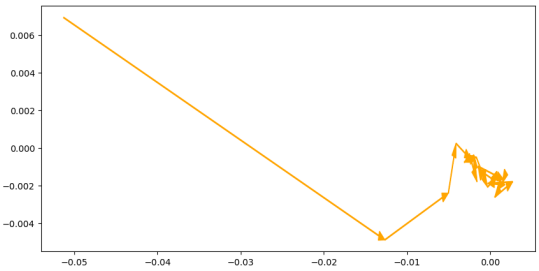
Our intuition is that gradient flow would behave differently in the two cases, following logarithmic magnitude steps. Empirical results met our intuitions which are explained below, along with examples.

- **Attribute addition**: in this case we aim to enter the target cluster. So we expect the first gradient steps to be bigger (and in the low λ case, orthogonal to the non-target attributes) to approach the cluster and the followings to be littler adjustments within the cluster.

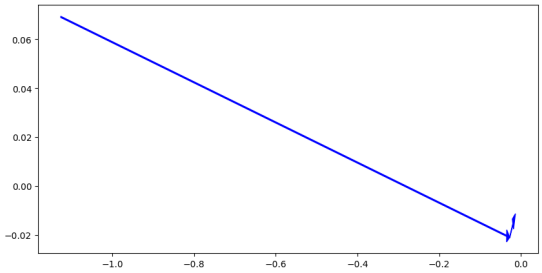
(a) Base image

(b) *Blonde* edit (λ=0.1)

(c) *Blonde* edit (λ=1)



(d) Gradient steps 2D visualization with λ=0.1



(e) Gradient steps 2D visualization with λ=1

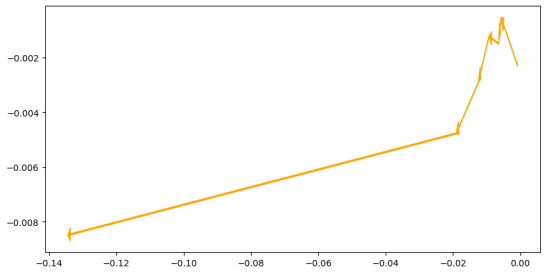
Figure 3.9: Attribute *Blonde_Hair* addition example. In the first row we have a randomly sampled image, along with both edits with λ=0.1 and with λ=1. Notice the smaller and preciser steps that lead to Figure 3.9b versus the longer yet impreciser steps that lead to the "blondier" Figure 3.9c. In the second row the connected arrows represent the subsequent gradient steps. We performed TSNE dimensionality reduction to be able to visualize the high dimensional vectors. In this example unnormalized gradients have been used.

- **Attribute removal**: in this case we aim to exit the target cluster. So we expect the first gradient steps to be smaller be accurately guided out from the cluster and the followings to be bigger once exited.

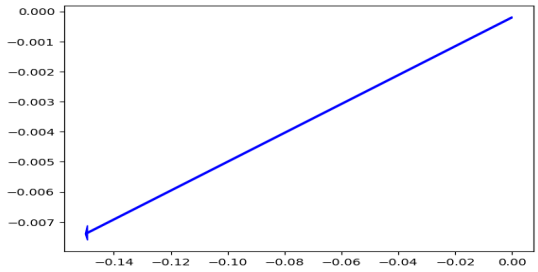
(a) Base image

(b) *Goatee* edit (λ=0.1)

(c) *Goatee* edit (λ=1)



(d) Gradient steps 2D visualization with λ=0.1



(e) Gradient steps 2D visualization with λ=1

Figure 3.10: Attribute *Goatee* removal example. In the first row we have a randomly sampled image, along with both edits with λ=0.1 and with λ=1. Notice the smaller steps that exits from the cluster and lead to Figure 3.10b versus the longer step that lead to the Figure 3.10c. In the second row the connected arrows represent the subsequent gradient steps. We performed TSNE dimensionality reduction to be able to visualize the high dimensional vectors. In this example unnormalized gradients have been used.

Empirically it has been noticed that when removing an attribute, an higher value of λ usually perform equally, if not better, that a lower one. An example can be seen in Figure 3.10c. We have confidence in saying that the reasons could reside in the above discussion. It's easier to remove an attribute since we need less precision when exiting the cluster. Moreover it could happen that we already are placed at the margin of the cluster, making the edit simpler..

Another *E2Editor*'s characteristic not yet discussed is **multi-modality**: it supports the joint manipulation of several attributes. For example, in a single edit pass we can edit not just one, but many target attributes together. For simplicity, until now the only case taken into account is the single-modal one. With the described method, the derivation to a multi-modal setting is pretty straightforward. We just need to take into account multiple indexes, corresponding to the target attributes, when computing $loss_{target}$ in Equation 3.9. Clearly each multiple-attributes edit is more difficult than a single-attribute one, but the principle of following the classification gradient doesn't change at all. One unavoidable factor is attribute entanglement: we could not perform an edit on attributes entangled by definition such as *Beard* and *Goatee*.

Figure 3.11 showcases a multiple-attributes editing, in particular on *Blonde_Hair* and *Smiling*.

(a) Base image

(b) Edit obtained increasing *Blonde* and decreasing *Smiling*, with λ=0.1

Figure 3.11: An example of an edit towards multiple attributes. One the left, the original image; on the right, the image edited towards both the attributes *Blonde_Hair* and *Smiling*. Since the base image has originally not blonde hair, this attribute is increased, while the opposite holds for the smile.

3.3.5. Considerations

To conclude this section about *E2Editor*'s method, we below highlight its pros and cons:

Pros:

- **Target and Non-Target Aware**: unlike many other methods whose objective function only considers the edit in the target attribute direction, which could result in an undesired entangled edit, ours also provides a way to control the variation of the non-target ones with the analytical help of an Attribute Classifier. See an example in figure 3.12.
- **Instance Aware**: this method doesn't provide a general editing direction for the target attribute like Instance Agnostic methods, which could bee too general and imprecise for specific cases, but a custom tailored direction for each single sample.
- **Multi modal**: it can handle multiple-attributes editing at the same time.
- **Two-Way**: both removing and adding attribute(s) is supported.

Cons:

- **Attribute Classifier need**: a solid unbiased classifier is crucial for the edits' outcome. The more robust it is, the more robust will the edits be, exactly because of the fact that the latent space edit depends on the classification score.

- **Time**: edits can be time-consuming in some settings, especially when a low λ value is chosen. Nonetheless, the time lost is gained in the edit disentanglement quality as discussed in Section 3.3.3.

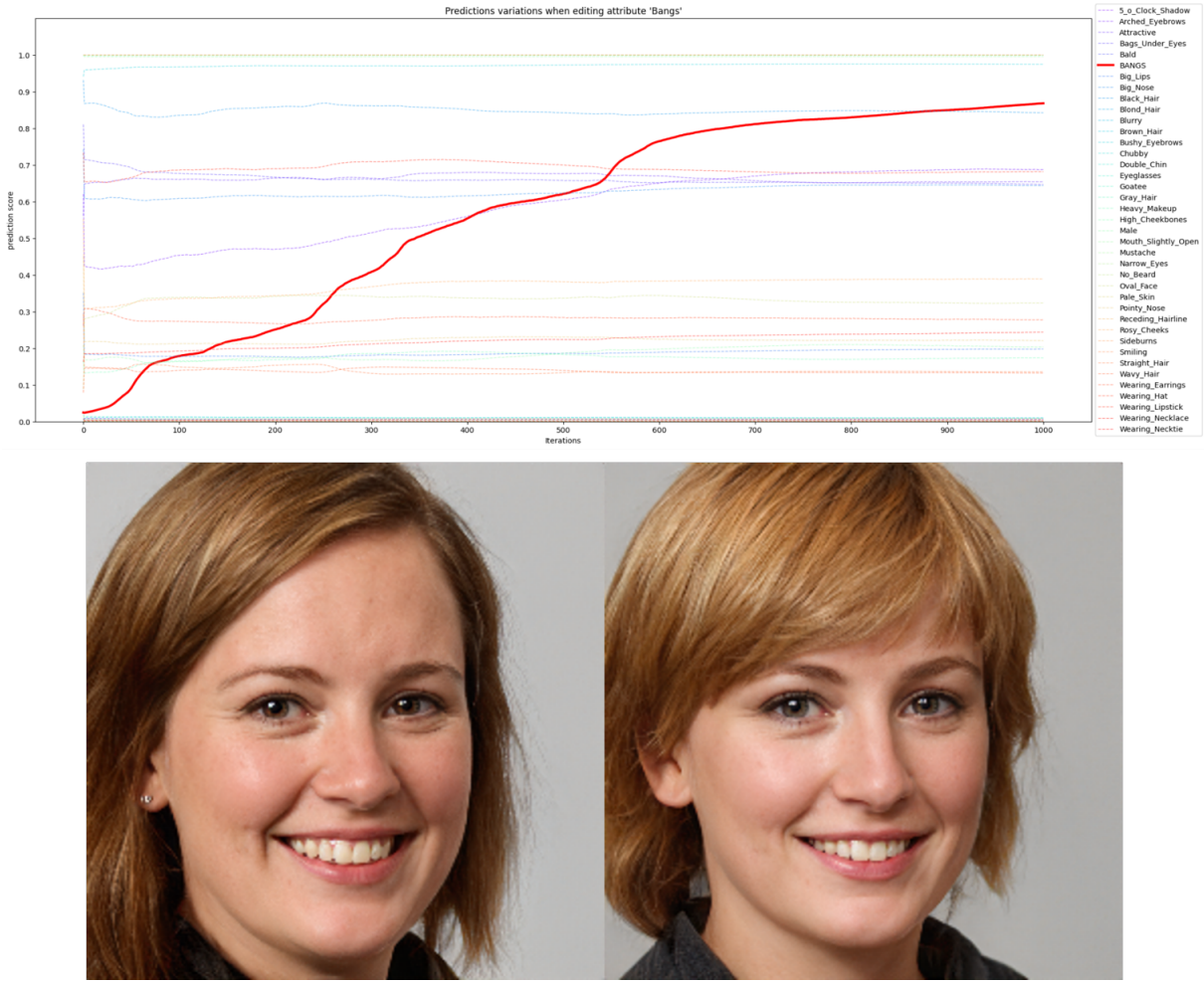


Figure 3.12: An example of edit on *Bangs*, where on the x-axis we record the iterations to reach the final image, while on the y-axis are recorded the scores of all the attributes at each step. We can see how the target attribute in the red line in the only one that undergoes a substancial variation, while the others stays almost invariate.

We believe that given this outline of *E2Editor*'s strengths and weaknesses, this method can definitely be competitive. In the next Section we will discuss more about quantitative results and comparisons with other literature methods.

4 | Experiments & Results

The final goal of this work is to train an explainable model which correctly disentangles non-trivial factors of variation (FoVs).

In general, the unsupervised learning of disentangled representation is proved to work only on biased toy datasets [10] [18]. It does work exactly because they exploit the dataset bias, but *Locatello et al.* [22] clearly stated that achieving disentanglement on real dataset without providing any explicit supervision is essentially impossible. For this reasons we designed a new learning framework, described in section 3.2.2, whose aim is to provide the supervision to encourage the model to correctly disentangle the known factors of variation, also on real non-toy datasets.

Following, we will describe the experimental setup, the basic and custom dataset used in the experiments, and discuss the metrics adopted; finally, we will have a look at the results obtained.

4.1. Experimental Setup

The categories of experiments that will follow are divided into two learning settings:

- **Unsupervised setting**: in these experiments we will test the limited unsupervised learning case. As baseline model we will use a β-VAE [10], with β=4, whose architecture is detailed in section 3.2.2.
- **Supervised setting**: in these experiments we will add the supervision described in our method in section 3.2 to bridge the gaps left by the unsupervised methods.

The above methodologies will be tested on two types of datasets:

- **Toy Dataset**: DSprites will be used to show the deficits of the unsupervised methods to correctly disentangle the latent FoVs. We will then present SDSprites, which is just a selection of DSprites couples of images in which only a FoV varies within the couple. Therefore, we will use it in our unsupervised method to demonstrate its soundness and improvements in disentanglement capabilites with respect to the

base case.

- **Non-Toy Dataset**: *Celeba* [21] is the focus of our work. We will provide the supervision to SVAE with the *DFaces* image couples, a novel custom dataset created with our *E2Editor* method, described in Section 3.3.

In Figure 3.1 we showcase paired examples for both the datasets.

Below we list all the others common parameters, hyperparameters, devices used and assumptions made throughout the whole process, for replicability purposes:

- The GPU used for all the experiment was a NVIDIA RTX A5000.
- All the datasets' images *dimension* has been kept at 64x64.
- The *latent dimension* of the bottleneck layer has been kept at 5 for DSprites, equal to the number of FoVs; it should be a wide enough bottleneck to allow for a valid reconstruction, given the triviality of the dataset and 128 for Celeba. Those values were found empirically, looking at the models' reconstruction capabilities.
- The *batch size* has been chosen at 34 for DFaces, since the number of facial FoVs was chosen at 17; in this way at each iteration the model sees one couple of images for each FoV. For all other datasets it has been chosen at 32.
- The β hyperparameter of the SVAE was fixed at the value 4 for all the experiments [10].

4.2. Datasets

We below present the two main dataset on which we focused our experiments.

4.2.1. DSprites

DSprites [1] is a dataset of 2D shapes procedurally generated from 6 ground truth independent latent factors. These factors are color, shape, scale, rotation, x and y positions of a sprite. All possible combinations of these latents are present exactly once, generating 737280 total images; an example of images is shown below.

[1] `https://github.com/deepmind/dsprites-dataset`

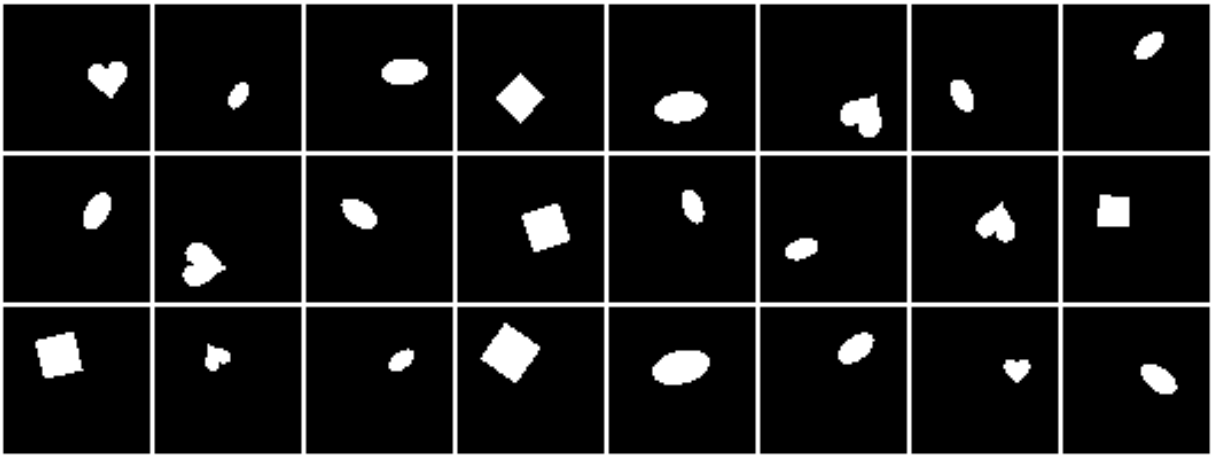


Figure 4.1: Random samples taken from DSprites dataset.

The latent factor values are precisely:

- Color: white
- Shape: square, ellipse, heart
- Scale: 6 values linearly spaced in [0.5, 1]
- Orientation: 40 values in [0, 2 pi]
- Position X: 32 values in [0, 1]
- Position Y: 32 values in [0, 1]

The aim of this dataset is testing the disentanglement properties of unsupervised models. It can be used to determine how well models recover the ground truth latents presented above.

4.2.2. CelebA

CelebFaces Attributes Dataset (CelebA) [2] [21] is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including:

- 10,177 number of identities,
- 202,599 number of face images,
- 5 landmark locations, 40 binary attributes annotations per image.

[2] https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

In this preliminary phase, we perform an overall exploration of the dataset.

Being a real face dataset labelled by humans, this dataset represents one of the best reflection of the real human faces data distribution available. Therefore, with the following analysis we aim to find principally which of the attributes are more prone to be positive/negative, and which ones should be correlated/entangled with each other.

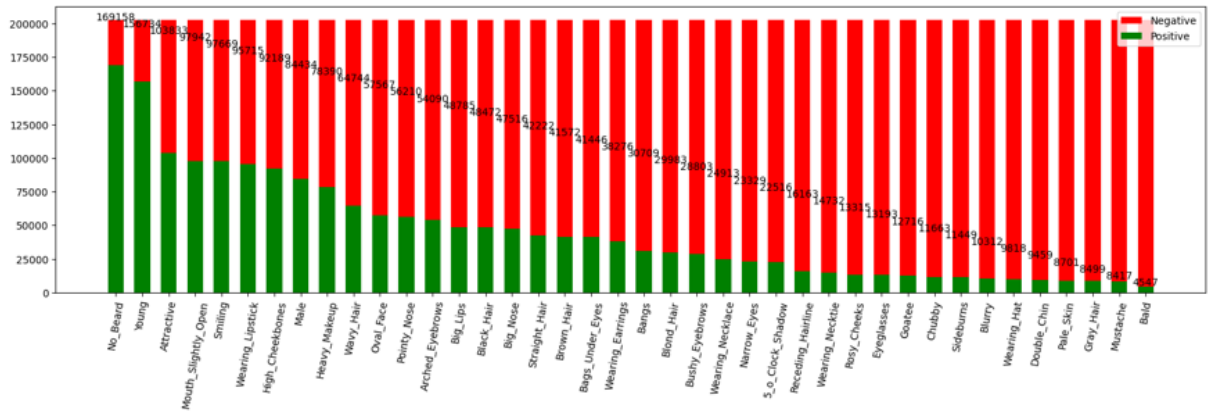


Figure 4.2: Quantities of positively and negatively labelled data points for CelebA [21], for each attribute.

From Figure 4.2, we observe that almost half the attributes are relatively below 20% of positivity rate. Only 3 attributes (*No_Beard*, *Young* and *Attractive*) are above 50%. This in turn influenced the generalization failure of the Attribute Classifier training described in Section 3.3.1.

Later, an attribute correlation analysis has been performed to see which of the attributes are "entangled" with some others. Using the PearsonR correlation coefficient on each couple of attributes, a kind of normalized measurement of the covariance, the resulting heatmap is depicted below. The results came out as expected: some attributes are highly positively correlated, such as *Heavy_Makeup* and *Wearing_Lipstick*, or *Chubby* and *Double_Chin*, and some others are highly negatively correlated like *Goatee* and *No_Beard* or *Male* and *Wearing_Lipstick*. This measurements has been useful to extract only 17 out of the 40 attributes to train our VAE, as we will see later in Section 3.2.

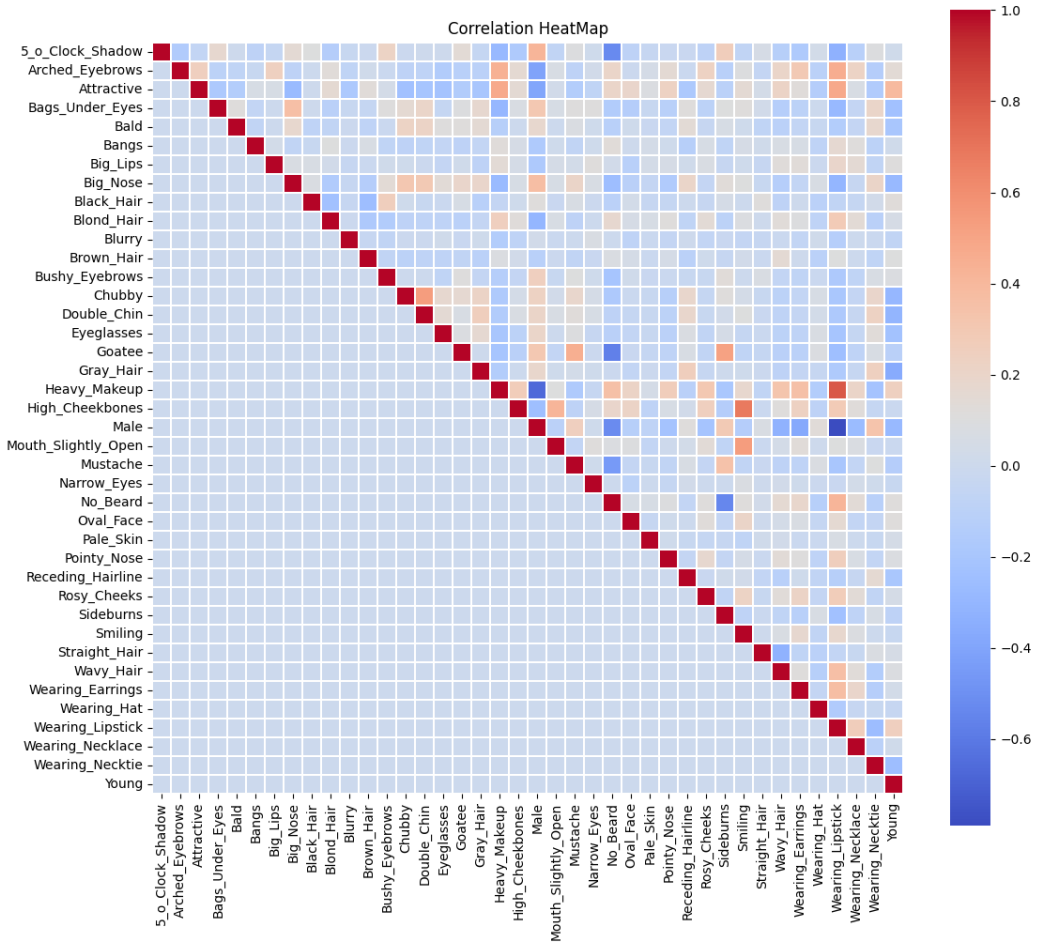


Figure 4.3: Correlation Heatmap between each attribute couple of CelebA. Red symbolizes an high correlation, blue a low one. Both, however, imply entanglement.

4.3. Datasets generation for Disentanglement enhancement

To add the required supervision needed to promote the latent space disentanglement to the β-VAE, we required some custom datasets. We below present them.

4.3.1. SDSprites

Given the poor results obtained by traversing the unsupervised β-VAE case described in 4.5.1, we decided to extract couples of images varying in a single generative factors in order to apply our method SVAeE described in section 3.2 also to DSprites. To generate

this dataset, which we named SDSprites, we applied the following mechanism:

- we fixed one generative factor
- for each DSprites image we selected another image in the dataset such that the only FoV that changes is the fixed one;
- these two images constitute a pair, with the additional information of the varying FoV.

Example: given the order of the FoV being [*shape*, *size*, *orientation*, *posX*, *posY*] and being the first image [*square*,0.5,0,1,0], the second image of the pair, given the fixed FoV *orientation* could be [*square*,0.5,π,1,0].

We below showcase some image couples, ready to be provided to our SVAE.

(a) Examples of dsprites' image couples that differs only in *shape*.

(b) Examples of dsprites' image couples that differs only in *size*.

(c) Examples of dsprites' image couples that differs only in *orientation*.

(d) Examples of dsprites' image couples that differs only in *positionX*.

(e) Examples of dsprites' image couples that differs only in *positionY*.

Figure 4.4: SDSprites image couples with one single varying FoV.

4.3.2. DFaces

With the *E2Editor* method described in Section 3.3, we managed to create a new visually facial attributes-wise disentangled dataset, called **DFaces**.

Figure 4.5: Paired samples from *DFaces* dataset; side by side images have one non-shared attribute like, respectively from top-down left-right, *Blonde_Hair, Goatee, Rosy_Cheek.*

It's composed of 200022 human facial images:

- 100011 couples of images, where each couple $(I_{base}, I_{edited,a})$ is composed of a base image I_{base} and an edited one $I_{edited,a}$ on the attribute a.
- 5883 couples of images for each one of the 17 chosen attributes, namely *Bald*, *Bangs*, *Big_Lips*, *Big_Nose*, *Black_Hair*, *Blond_Hair*, *Bushy_Eyebrows*, *Chubby*, *Double_Chin*, *Eyeglasses*, *Goatee*, *Gray_Hair*, *Heavy_Makeup*, *Male*, *Mustache*, *No_Beard*, *Pointy_Nose*, *Rosy_Cheeks*, *Sideburns*, *Smiling*, *Wearing_Lipstick*, *Young*.

To assess the results of the SFAE's edits, *Nickabadi et al.* in [28] provide a rundown of the various evaluation metrics, that could be:

- **Qualitative**: this kind of evaluation needs humans in the loop, mostly through surveys.
- **Quantitative**: this type is divided in many subtypes, regarding which aspect we aim to evaluate:
  - **Image aspect**: metrics like *FID Score*, *Sim*, *Mssim* can be used. Those metrics establish the general image quality and it's commonly used to assess the quality of the image generated by GANs.
  - **Edit quality**: these metrics assess the quality of the single edit on an attribute, which could be differenciated in:

- **Target modification**: with the use of attribute classifiers, this metric analyzes the change in the predictions score on the target attribute between the base and the edited image.
- **Non-Target preservation**: a common approach to this goal is to perform face verification between the input and the modified face images. To do so, the distance between the feature vectors of the initial and manipulated face images, extracted by pretrained face recognition models, are measured using a Euclidean or cosine distance and compared with a predetermined threshold to verify if the two images belong to the same person or not

Quantitative experiments have been performed to assess analytically the resulting edits with different α configurations (for details visit section 3.3). The chosen configurations were the extreme opposites, namely α=1 and α=0.1. The metric kept into account is the sum of the MSE value of all the non-target attributes predictions between 1000 base images and the correspondent edit, obtained when editing on the target attribute. Its aim is to monitor how much an edit of a target attribute influences the other attributes predictions. Obviously, the lower the better.

For each attribute a_{target}, the considered metric is formally defined as:

$$v_{a_{target}} = \frac{1}{N} \sum_{i}^{N} \underbrace{\sum_{a \neq a_{target}}^{A} |\widetilde{s}_a - s_a|^2}_{\text{MSE editing on } a_{target}} \tag{4.1}$$

where $\widetilde{s}_a$ and s_a are respectively the classifier $\mathcal{C}$ score of attribute a for the edited image on a_{target} and the base image.

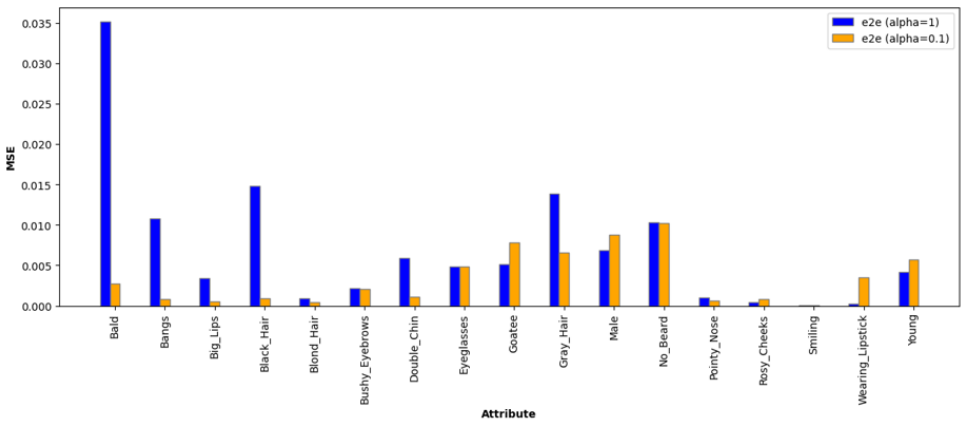


Figure 4.6: Sum of MSE of the non-target attributes between the prediction of 1000 base images versus their edit on an attribute (the lower the better).

In each column of Figure 4.6, we show the metric 4.1 attribute by attribute. Only a subset of 17/40 attributes has been used in this entire work. We can see that the configuration α=0.1 most of the times performs better than the configurations with α=1. As we discussed in Section 3.3.4, sometimes α=1 could be preferable.

Here is the tabular version of the results:

	E2Editor (α=1)	**E2Editor (α=0.1)**
Bald	0.03518942	**0.00271607**
Bangs	0.01077762	**0.00078402**
Big_Lips	0.00344190	**0.00052207**
Black_Hair	0.01483328	**0.00091163**
Blond_Hair	0.00095059	**0.00047568**
Bushy_Eyebrows	0.00215917	**0.00209541**
Double_Chin	0.00587940	**0.00112743**
Eyeglasses	0.00485284	**0.00480959**
Goatee	**0.00511453**	0.00781297
Gray_Hair	0.01387716	**0.00655545**
Male	**0.00685158**	0.00882778
No_Beard	0.01028384	**0.01022305**
Pointy_Nose	0.00102964	**0.00060121**
Rosy_Cheeks	**0.00046284**	0.00086657
Smiling	3.61024006	**3.52134591**
Wearing_Lipstick	**0.00026791**	0.00353027
Young	**0.00414855**	0.00575577

Table 4.1: Sum of MSE of the non-target attributes between the prediction of 1000 base images versus their edit on a target attribute (the lower the better).

Thus, to have a final cumulative metric, we averaged each one of the attribute-wise MSE by method:

$$v_A = \frac{1}{A}\sum_{a}^{A} v_a \tag{4.2}$$

In this case, we also added the InterFaceGAN [32] method to perform Semantic Facial Attribute Editing, to have a comparison with the state-of-the-art method. As we can see from this figure below, $E2editor(\alpha = 0.1)$ in average performs better.

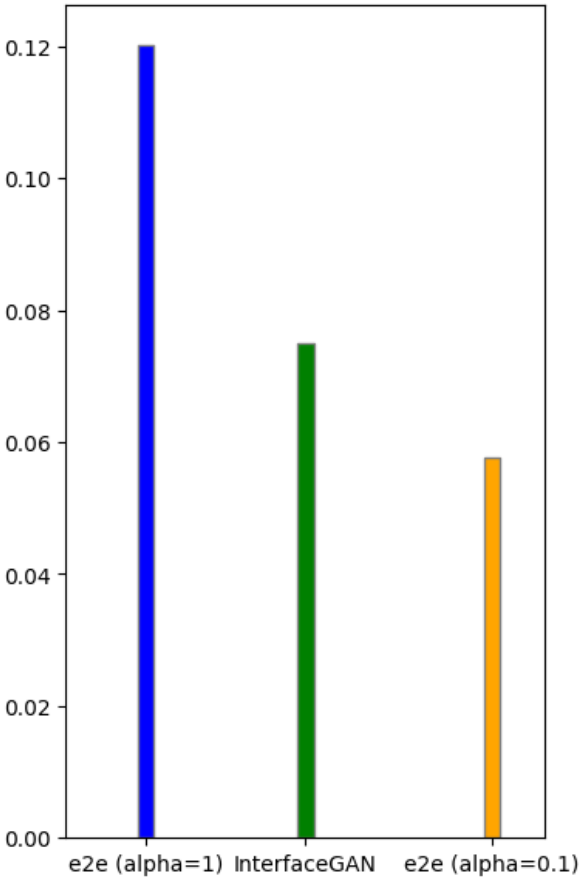
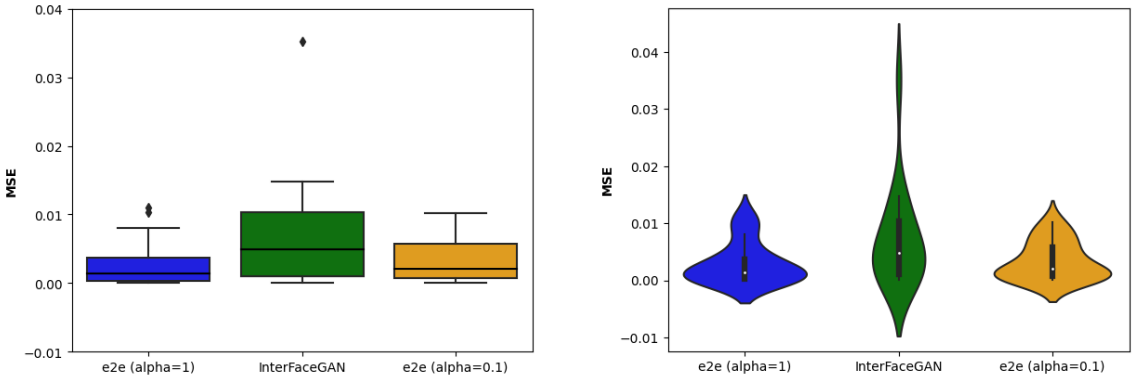


Figure 4.7: Plot comparition of the sum of the MSE over each attribute.

	Avg Sum of MSE
E2Editor (α=1)	0.120156
InterFaceGAN	0.074868
E2Editor (α=0.1)	**0.057652**

Table 4.2: SFAE methods quality comparison.

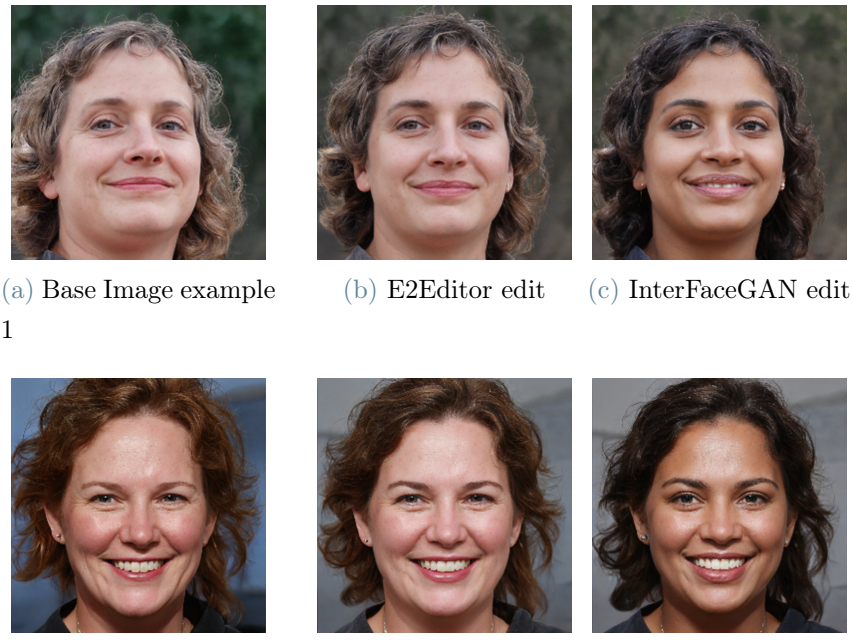
Last, to have deeper insights about the distributions divided by attribute, instead of a simple MSE, below we showcase the Box Plot and the Violin Plot of the MSE distribution by attributes in Figure 4.8.



(a) Box Plot (b) Violin Plot

Figure 4.8: Box Plot and Violin plot of the MSE distributions by attribute.

When comparing our results with InterFaceGAN's ones, we notice some peculiar improvements. In particular, we noticed that when editing on attribute *Big_Lips* with their method, the edit actually ended up changing the skin color. This is probably due to a bias in the CelebA dataset on which InterFaceGAN has been trained on; this bias incorporates the correlation between big lips and dark skin color. Our method instead, as shown in the figure below, is able to overcome the bias and identify the right gradient direction for the edit in question.



(a) Base Image example 1 (b) E2Editor edit (c) InterFaceGAN edit

(d) Base Image example 2 (e) E2Editor edit (f) InterFaceGAN edit

Figure 4.9: Examples of the edit bias on the attribute *Big_Lips*

4.4. Metrics

To evaluate the disentanglement level of our results, many metrics have been proposed [35]. Three of the most famous ones have been adopted in this work, namely:

- **Z-diff**: often known as the βVAE metric, chooses pairs of instances to group together into batches. A factor v_i in a batch is randomly selected. Then, samples v_1 and v_2 are paired together if their values for the selected factor are the same ($v_i{}^1 = v_i{}^2$). The absolute difference between the samples' codes is used to describe pairs ($p = |z_1 - z_2|$) of data. The fixed factor's related code dimensions are supposed to have the same value, which entails a smaller difference than those of the other code dimensions. A point in the final training set is created by taking the mean of all pairwise differences in the subset. To create a substantial training set, the technique is done several times. The data set is then used to train a linear classifier to determine which factor was fixed. The Z-diff score represents the classifier's accuracy. The accuracy of a random classifier of $\frac{1}{M}$, where M is the number of factors, can be used to scale the output closer to the $[0, 1]$ range.

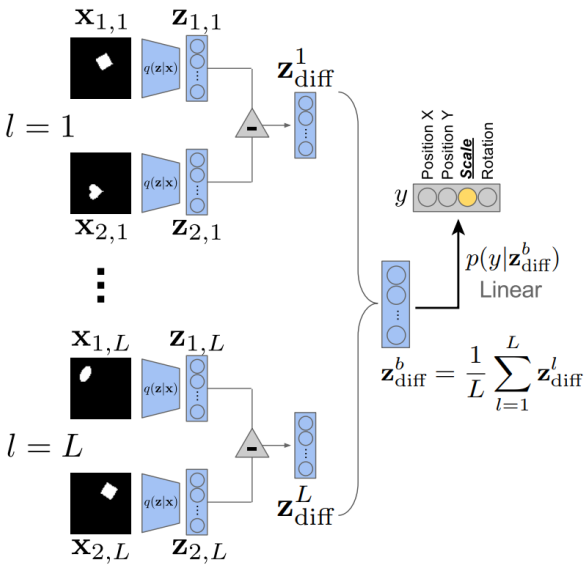


Figure 4.10: Z-diff disentanglement metric calculation process, proposed in [10]

- **z-min variance**: often known as FactorVAE metric, this metric was proposed to address some of the shortcomings of Z-diff metric. The logic is the same as for Z-diff: if the factor value is the same, then the code dimensions encoding the factor should also be similar. All codes are first normalized using their standard deviation, which is calculated over the whole collection of data. A factor is chosen at random and

fixed at a random value for the subset. The subset consists of sampled occurrences for which the chosen factor is set to the chosen value. Over the normalized codes in the subset, variance is calculated. The fixed factor is connected to the code dimension with the lowest variation. The factor-code relationships are employed as data points in a majority vote classifier, and many subsets are produced. Z-min The classifier's mean accuracy is measured by the variance score. Similar to Z-diff, scaling the output closer to the $[0, 1]$ range may be accomplished by using a random classifier accuracy of $\frac{1}{M}$.

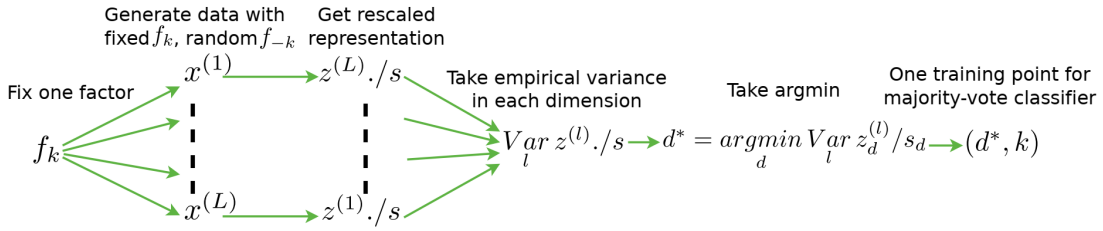


Figure 4.11: Z-min var disentanglement metric calculation process, proposed in [18]

- **DCI**: The authors in [7] presented a comprehensive framework for assessing disentangled representations, replacing a single metric with a multifaceted evaluation. They introduce distinct measures for Modularity, compactness, and explicitness, denoted as Disentanglement, Completeness, and Informativeness, respectively. To evaluate these aspects, they employ regressors trained to predict factors from codes. For Modularity and compactness estimation, they utilize importance weights, denoted as R_{ij}, derived from the regressor's internal parameters, which relate to each factor and code dimension pair. Linear Lasso regressors or random forests are employed for non-linear mappings. The lasso regressor's importance weights are the magnitudes of its learned weights, while Gini importance is used with random forests to evaluate code dimensions.

  Compactness for a specific factor vi is calculated using $C_i = 1 + \sum_{j=1}^{d} p_{ij} \log_d(p_{ij})$, where p_{ij} represents the probability that code dimension z_j is essential for predicting v_i. These probabilities are obtained by dividing each importance weight by the sum of all importance weights related to that factor: $p_{ij} = \frac{R_{ij}}{\sum_{k=1}^{d} R_{ik}}$. The overall compactness of the representation is the average compactness across all factors.

  Similarly, Modularity for code dimension z_j is computed as $D_j = 1 + \sum_{i=1}^{M} p_{ij} \log_M(p_{ij})$, where p_{ij} signifies the probability that z_j is critical only for predicting v_i. The importance weights here are normalized with respect to codes: $p_{ij} = \frac{R_{ij}}{\sum_{i=1}^{M} R_{kj}}$. The representation's Modularity score is a weighted average of individual code dimen-

sion Modularity scores, weighted by ρ_j to account for less important codes. The weight ρ_j is calculated as the total importance for z_j normalized by the sum of all importance weights: $\rho_j = \frac{\sum_{i=1}^{M} R_{ij}}{\sum_{k=1}^{d} \sum_{i=1}^{M} R_{ik}}$.

Explicitness of the representation is assessed using the prediction error of the regressor. Normalized inputs and outputs enable the computation of the estimation error, which is then normalized between 0 and 1. An explicit representation is one where the mean squared error (MSE) of the predictor is lower than the expected MSE between two uniformly distributed random variables (X and Y), where $MSE = E[(X - Y)^2] = \frac{1}{6}$. Therefore, explicitness is quantified as $1 - 6 * MSE$. Values below 0 are reported as 0 in their implementation.

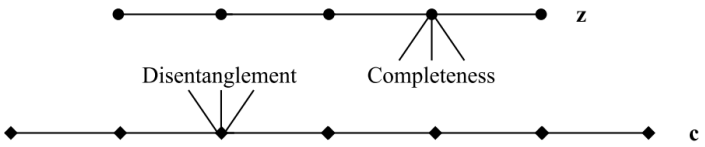


Figure 4.12: Visualization of Disentanglement and Completeness, from [7]

4.5. Experiments

Below are listed all the experiments performed for the different types of datasets along with the correspondent settings, described in section 4.1.

4.5.1. DSprites

As analyzed in section 4.2.1, DSprites is a toy dataset purposefully built for testing the disentanglement capabilities of a model. It has six known generative factors that fully defines the images.

Although its FoVs are so easy to be disentangled, plain β-VAE can only perform well in the simplified case where the factors to be disentangled are only the posX and posY of the white dot. So, for this experiment we simply fixed values for the other FoVs (namely shape as circle, orientation and size) so that the only factors to vary were the dot coordinates. In this way, the only latent traversals that would show any meaningful variations would be the latent variables encoding the posX and posY FoVs, as can be seen in the full figure 4.15, while others latent traversals show that non encoded factor resides in others dimensions. Below we highlighted in detail the only two meaningful traversals, in figure 4.13 and 4.14.

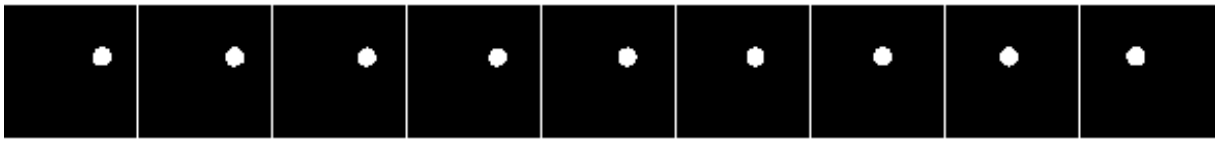


Figure 4.13: Latent Traversal on the dimension that encodes the FoV *posX*.

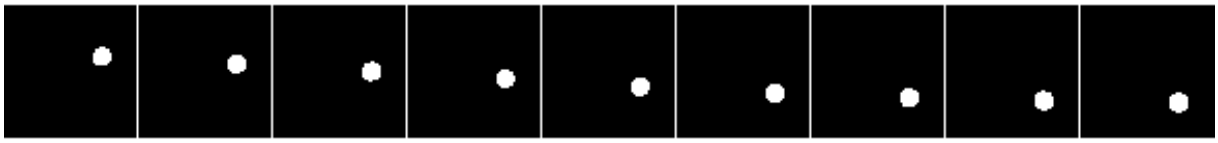


Figure 4.14: Latent Traversal on the dimension that encodes the FoV *posY*.

The results on disentanglement when adding other generative factors are unclear: we can see that the traversals are not clean, so several generative factors are mixed into one (e.g. position X and orientation), so by definition they are not well disentangled by the model. We can infer that the model cannot keep up with the increasing complexity of the generative factors, ergo the need for a supervision.

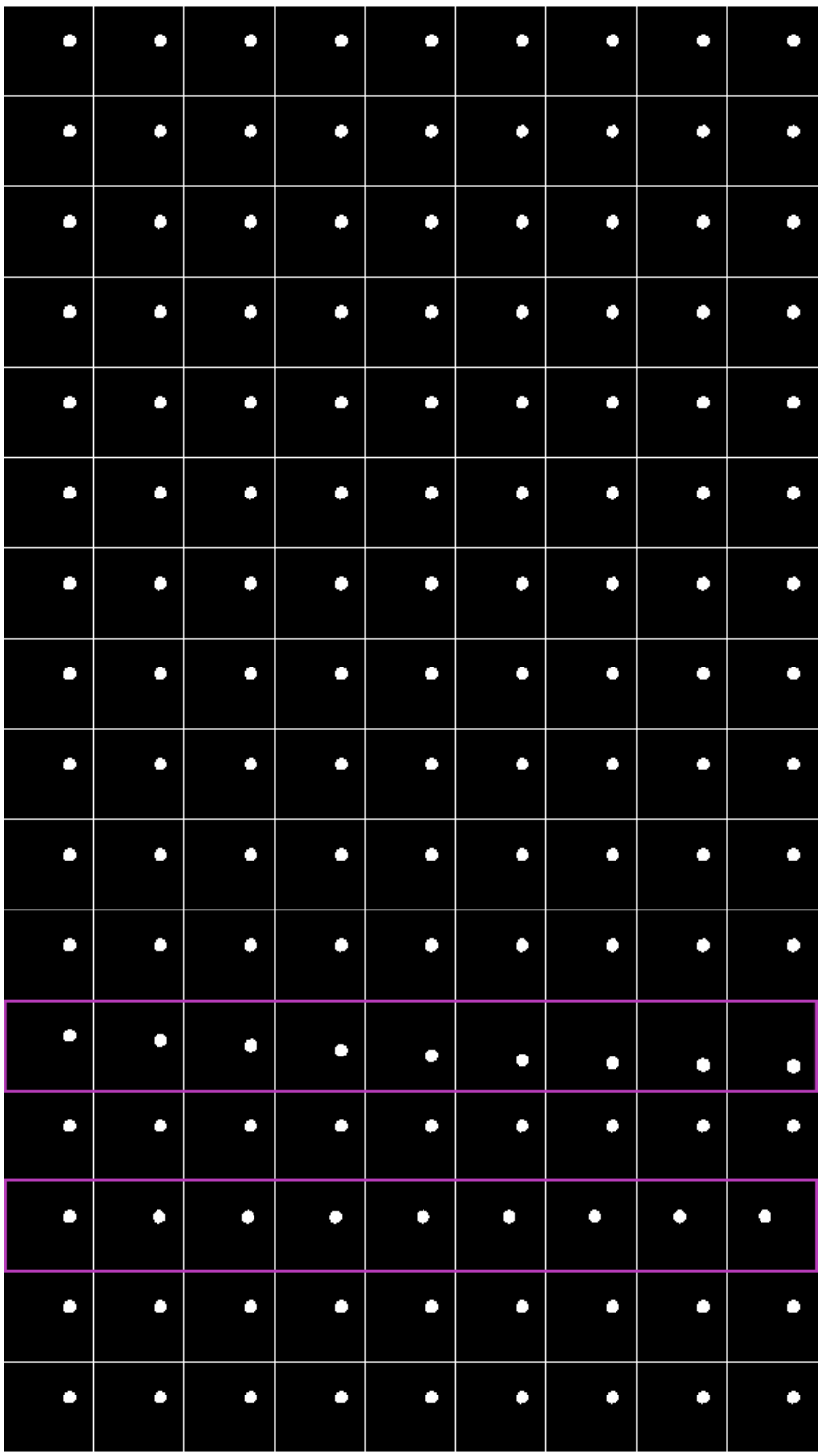


Figure 4.15: Latent traversal of a plain β-VAE with a 16-dimensional latent space, with FoVs posX and posY. We can notice how it correctly disentangles the factor posX in the 14th dimension and posY in the 12th, leaving the others unchanged as they shouldn't (and don't) ancode any FoV.

4.5.2. SDSprites

By providing SDSprites supervision, the model correctly grasps the FoV to be encoded in each dimension, as can be seen in Figure 4.16 showing the latent traversal. Given the same sampled image of the first column, each row displays a traversal on a different dimension, semantically represented by the FoV on the left. In particular:

- *Shape*: in the first dimension is encoded the shape FoV. It's the most difficult to notice since the training epochs used were very little, and the reconstruction capabilitiy of the network didn't reach great results, since they were trading-off with the kl-loss and the pair loss.
- *Scale*: the second dimension is clearly encoding the scale factor, as the dot is getting bigger.
- *Orientation*: the third dimension encodes the orientation factor, as the dot is varying direction.
- *Position X*: the fourth dimension is encoding the x coordinate factor, as the dot moving from left to right.
- *Position Y*: the fifth dimension is encoding the y coordinate factor, as the dot moving up.

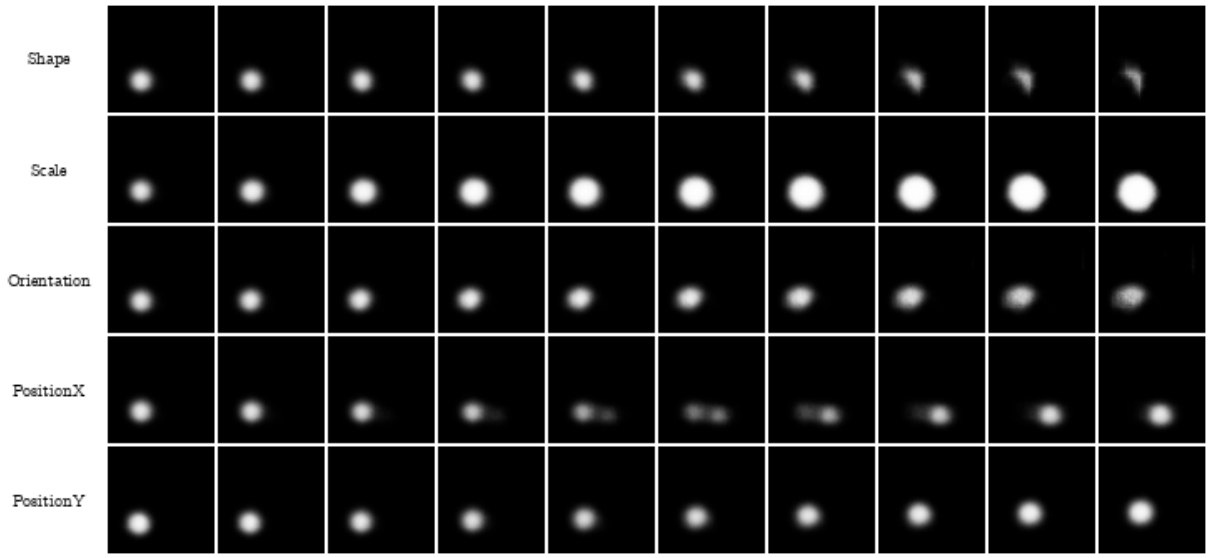


Figure 4.16: Latent Traversal of SVAE trained on *SDSprites*. Given the same sampled image of the first column, each row displays a traversal on a different dimension, that semantically encodes the FoV on the left.

Comparing SVAE (trained for 3k epochs) with the state-of-the-art methods (each trained for 300k epochs in Google's Official Disentanglement Library) [22], we show competitive results given the significantly shorter training duration, highlighting the efficiency and practicality of our approach for disentanglement representation learning. Results are shown in Table 4.3. In this table, we compare we major methods described in 2.3 and 2.4 on the metrics described in 4.4 on DSprites dataset.

	z-diff	**z-min**	**DCI**
βVAE	0.823	0.660	0.186
FactorVAE	0.853	0.750	0.256
MLVAE	0.896	0.701	0.294
GVAE	**0.923**	**0.847**	**0.479**
SVAE	0.652	0.711	0.313

Table 4.3: Disentanglement score comparisons with state-of-the-art methods.

We believe that with equal training epochs, or adjustments in the losses trade-off, our method can achieve more competitive results.

4.5.3. DFaces

Since we are dealing with a non-toy real-world dataset, the data generation process doesn't define clear FoVs. Thus we decided to treat some facial attributes as factors of variation. The attributes chosen are the 17 out of 40 of the most representative labels over which CelebA dataset [21] has been labelled, namely: *Bald*, *Bangs*, *Big_Lips*, *Black_Hair*, *Blond_Hair*, *Bushy_Eyebrows*, *Double_Chin*, *Eyeglasses*, *Goatee*, *Gray_Hair*, *Male*, *No_Beard*, *Pointy_Nose*, *Rosy_Cheeks*, *Smiling*, *Wearing_Lipstick*, *Young*. Here the *latent dimension* has been set to 128, but only the forced 17 dimensions carry an interpretable semantic FoV.

Figure 4.17: Latent traversals oh the SVAE latent space. The first row contains the sampled images, and each row traverses a single FoV, defined on the left. Only the clearest results are shown..

We can see how traversing a single dimension, it changes the image aspect on the leading FoV written on the right. Clearly the reconstructions are a bit blurry; longer trainings should fix this issue.

Given the obtained results, we managed to train an explainable encoder, paired with its decoder, on which we know how to perform edits in its latent space thanks to the forced disentanglement property, showed in its latent representation by this latent traversals. This could be employed in various scenarios, for example to explain the outcomes of task like face recognition.

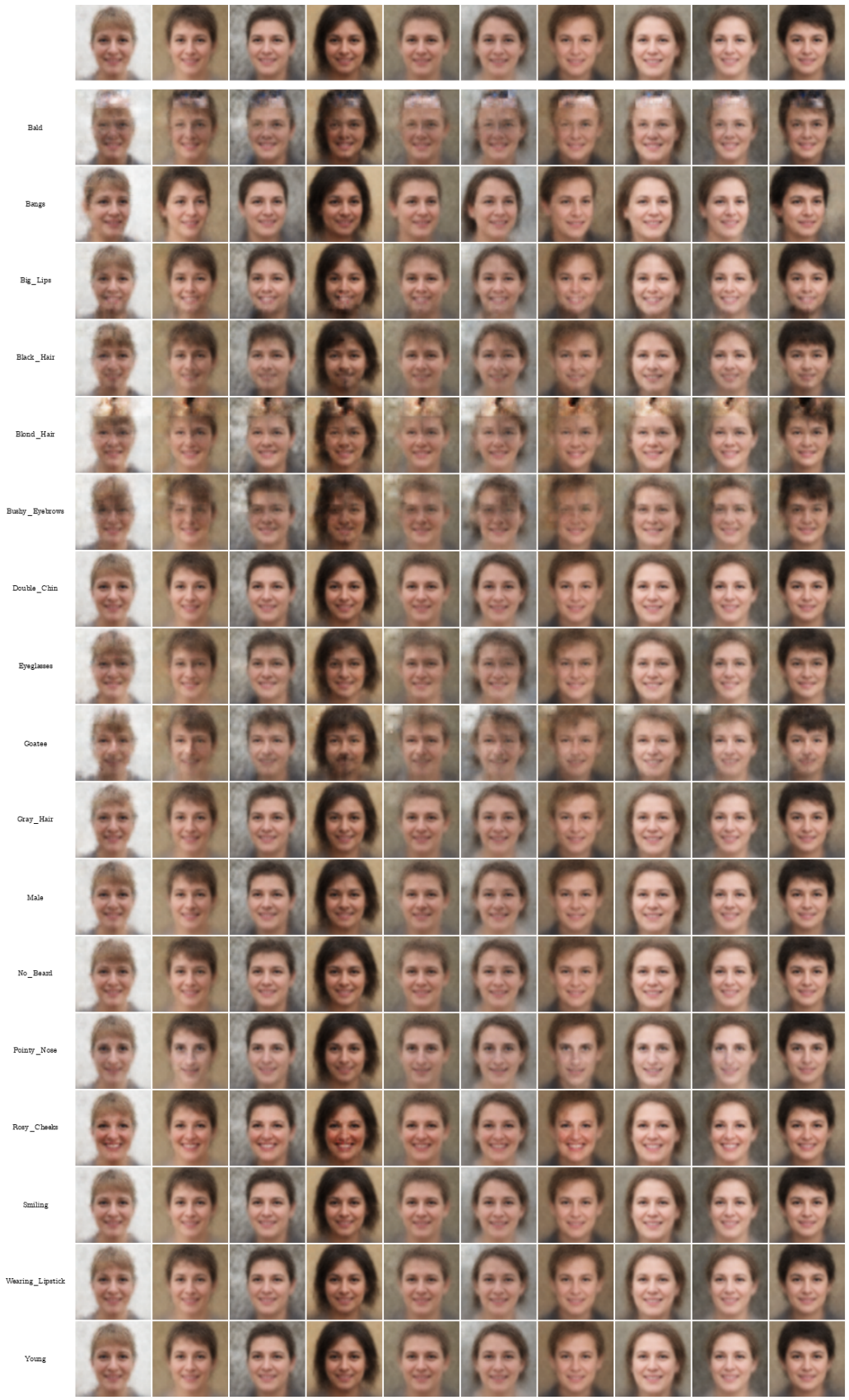


Figure 4.18: Full Traversal on all the semantically meaningful latent dimensions of SVAE trained in *DFaces*.

5 | Conclusions & Future Works

In summary, we addressed the problem of generating explainable models by leveraging on the forced latent space disentanglement. The proposed solution consists of a novel Variational framework in which paired observations differing in a single factor of variation induce the model into encoding that non-shared attribute in a dedicated latent dimension. To test our method's effectiveness on real-world scenarios, we developed a novel Semantic Facial Attribute Method to generate a tailored dataset. However, the aim of this work was to prove that disentanglement could be forced and then exploited also in real-world scenarios. The aim of our test on facial images was just to verify its effectiveness, and demonstrate its applicability in whichever real-world scenario. The gathered results show the viability of the suggested strategy and its competitiveness compared to the main solutions in the literature.

Future works should:

- carry out more intensive trainings on both the tested datasets;
- improve the reconstruction quality, essential for a valid evaluation of the disentanglement outcomes especially in the latent traversals.
- test out the SVAE method on other datasets, such as the variations of the DSprites dataset existing in the literature;

Bibliography

[1] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space?, 2019.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

[3] D. Bouchacourt, R. Tomioka, and S. Nowozin. Mlvae: Multi-level variational autoencoder: Learning disentangled representations from grouped observations, 2017.

[4] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age, 2018.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[6] E. Dupont. Learning disentangled joint continuous and discrete representations, 2018.

[7] C. Eastwood and C. K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=By-7dz-AZ`.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[10] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[11] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a definition of disentangled representations, 2018.

[12] M. D. Hoffman and M. J. Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, 2016.

[13] H. Hosoya. Gvae: Group-based learning of disentangled representations with generalizability for novel contents, 2021.

[14] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[16] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[17] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks, 2019.

[18] H. Kim and A. Mnih. Disentangling by factorising, 2019.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022.

[20] J. Lin, R. Zhang, F. Ganz, S. Han, and J.-Y. Zhu. Anycost gans for interactive image synthesis and editing, 2021.

[21] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[22] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations, 2019.

[23] F. Locatello, B. Poole, G. Raetsch, B. Schölkopf, O. Bachem, and M. Tschannen. Weakly-supervised disentanglement without compromises. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6348–6359. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/locatello20a.html`.

[24] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem. Disentangling factors of variation using few labels, 2020.

[25] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders, 2016.

[26] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[27] A. Melnik, M. Miasayedzenkau, D. Makarovets, D. Pirshtuk, E. Akbulut, D. Holzmann, T. Renusch, G. Reichert, and H. Ritter. Face generation and editing with stylegan: A survey, 2023.

[28] A. Nickabadi, M. S. Fard, N. M. Farid, and N. Mohammadbagheri. A comprehensive survey on semantic facial attribute editing using generative adversarial networks, 2022.

[29] Y. Nitzan, A. Bermano, Y. Li, and D. Cohen-Or. Face identity disentanglement via latent space mapping, 2020.

[30] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[31] L. Schott, J. von Kügelgen, F. Träuble, P. Gehler, C. Russell, M. Bethge, B. Schölkopf, F. Locatello, and W. Brendel. Visual representation learning does not generalize strongly within the same domain, 2022.

[32] Y. Shen, J. Gu, X. Tang, and B. Zhou. Interpreting the latent space of gans for semantic face editing, 2020.

[33] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017.

[34] Z. Wu, D. Lischinski, and E. Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation, 2020.

[35] J. Zaidi, J. Boilard, G. Gagnon, and M. Carbonneau. Measuring disentanglement: A review of metrics. *CoRR*, abs/2012.09276, 2020. URL `https://arxiv.org/abs/2012.09276`.

List of Figures

List of Tables

Ringraziamenti

Se sono arrivato fino a qui è doveroso fare dei ringraziamenti.

In primis devo ringraziare il professor Giacomo Boracchi, per avermi permesso di lavorare nel suo team di ricerca e di intraprendere questo percorso di crescita personale. Voglio ringraziare anche il mio advisor Loris Giulivi per il costante ed essenziale supporto dato nel corso dei lunghi mesi di brainstorming, studio, implementazione e scrittura. Un grosso grazie anche al professor Francesco Locatello, per il prezioso e non scontato aiuto ricevuto.

Devo ringraziare la mia famiglia, che più di tutti mi ha supportato nel corso di questi cinque anni; a mio padre Fabio, per essere sempre il mio punto di riferimento ed il mio esempio da seguire; a mia mamma Paola, per essere la persona amorevole che sempre ha creduto in me e che mi ha sempre messo al primo posto; senza voi due non sarei niente. A mio fratello Fra, per essere la roccia su cui posso sempre poggiarmi, da tutta la vita; a mia sorella Gaia, per essere la mia piccola complice.

Ai miei nonni; al nonno Franco e alla nonna Vittoria, che piu di tutti avrebbero voluto vedermi tagliare questo traguardo, siate sempre fieri di me; alla nonna Olga e al nonno Cesare, per la serenità che avete sempre trasmesso e per essere sempre stati i nostri secondi genitori.

A Giulia, per essere la spalla su cui posso sempre contare, per essere la persona con cui voglio condividere la vita, e per essere la persona che amo.

Ai miei amici, quelli di sempre, che ci sono sempre stati: a Tinor, GhiGho, Jek, Ale, Guazzo, Ste, Lori, Sofi, Alessia, Sonia, Marti, Sara, Giada, Dado, Galbio, Rubi, Fabio.

Ai miei amici di uni, per avermi accompagnato, aiutato e aver condiviso momenti che non torneranno mai, ma che ricorderemo sempre; a Fra, Filop, Paolo, Fede, Serena, Simo, Richi.

Grazie.