PCA-AE: Principal Component Analysis Autoencoder for Organising the Latent Space of Generative Networks

Chi-Hieu Pham, Saïd Ladjal, Alasdair Newson

Noname manuscript No.
(will be inserted by the editor)

PCA-AE: Principal Component Analysis Autoencoder for organising the latent space of generative networks

Chi-Hieu Pham · Saïd Ladjal · Alasdair Newson

Received: date / Accepted: date

Abstract Autoencoders and generative models produce some of the most spectacular deep learning results to date. However, understanding and controlling the latent space of these models presents a considerable challenge. Drawing inspiration from principal component analysis and autoencoders, we propose the Principal Component Analysis Autoencoder (PCA-AE). This is a novel autoencoder whose latent space verifies two properties. Firstly, the dimensions are organised in decreasing importance with respect to the data at hand. Secondly, the components of the latent space are statistically independent. We achieve this by progressively increasing the latent space during training, and with a covariance loss applied to the latent codes. The resulting autoencoder produces a latent space which separates the intrinsic attributes of the data into different components of the latent space, in a completely unsupervised manner. We also describe an extension of our approach to the case of powerful, pre-trained GANs. We show results on both synthetic examples of shapes and on a state-of-the-art GAN. For example, we are able to separate the colour shade scale of hair, pose of faces and gender, without accessing any labels. We compare the PCA-AE with other state-of-the-art approaches, in particular with respect to the ability to disentangle attributes in the latent space. We hope that this approach will contribute to better understanding of the intrinsic latent spaces of powerful deep generative models.

Keywords Generative networks · autoencoders · image generation

Submitted to the editors August 5, 2021. This work was funded by the Labex DIGICOSME

Chi-Hieu Pham
LTCI, Télécom Paris, Institut Polytechnique de Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France
E-mail: chi-hieu.pham@telecom-paris.fr
Present address: LSL, L@BISEN, ISEN Yncrea, 20 rue Cuirassé Bretagne, CS42807, 29228, Brest Cedex, France of Chi-Hieu Pham

Saïd Ladjal
LTCI, Télécom Paris, Institut Polytechnique de Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France
E-mail: said.ladjal@telecom-paris.fr

Alasdair Newson
LTCI, Télécom Paris, Institut Polytechnique de Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France
E-mail: anewson@telecom-paris.fr

1 Introduction

The recent impressive results of deep generative models and autoencoder-type models rely on a core idea: uncovering a compact, powerful latent space where the original high-dimensional data can be better synthesised or manipulated. Some of the most astounding recent synthesis results in deep learning have come from generative models such as generative autoencoders [21,42,39,41,16,12] or Generative Adversarial Networks (GANs) [10,37,18,40,19,6,44,32]. However, in spite of their undoubted efficiency, the latent spaces created by these models are difficult to interpret. In particular, a common problem is that these spaces are *entangled*: several image characteristics are often combined into one dimension of the latent space, making understanding it difficult. Certain previous approaches have attempted to disentangle the space in a semi-supervised manner, that requires knowledge about the true underlying factors of the data [23,35,29,8,14,38]. However, we would like to achieve this organisation of the latent space in a non-supervised approach, letting the data tell us what variability exists in the database.

In this work, we propose a network which we refer to as the "Principal Component Analysis Autoencoder" (PCA-AE). An autoencoder is a neural network consisting of two sub-networks : an encoder and a decoder. These networks project data to and from the lower-dimensional latent space. Ideally, we would like this latent space to be interpretable

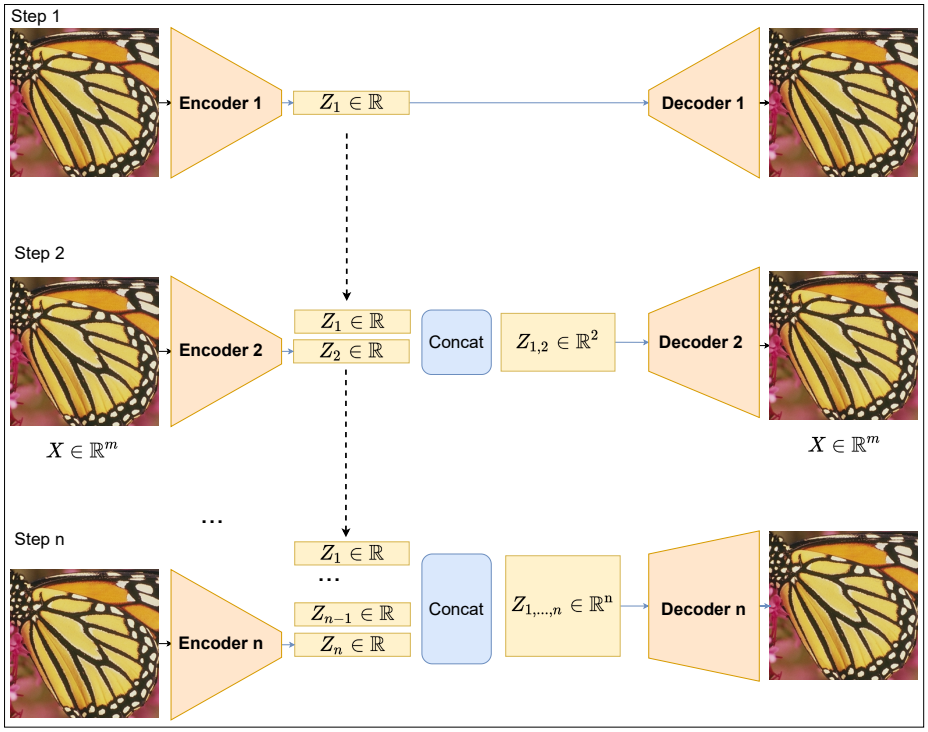


Fig. 1: Architecture of our PCA-AE. At the n^{th} step, the PCA-AE takes all previous pre-trained encoders, while the decoders are discarded. The parameters of these encoders are fixed. Their output are concatenated with those of the n^{th} encoder and trained with the new n^{th} decoder.

and navigable. By navigable, we mean that we have a method to move in the latent space such that some visual attributes of the output image are modified in a controlled manner, *i.e.* without changing all attributes at once or randomly. We propose to achieve this by creating an autoencoder which shares some of the desirable characteristics of the PCA. The classical PCA is a linear transformation to a space with two main properties. Firstly, the axes are organised in order of decreasing variability. So, along the first axis lies the greatest variability of the data, along the second orthogonal axis lies the second-greatest variability, and so on and so forth. Secondly, the axes are orthogonal to each other, which is necessary for interpretation and manipulation. Ideally, we would like to have the best of both worlds, *i.e.* the power of a non-linear transformation (a neural network here) with the aforementioned properties of PCA. This is the objective of this work. More precisely, our goal is to propose an autoencoder with the following two properties: i) the latent space components (axes) are ordered in terms of decreasing "importance" (this is defined shortly afterwards) and ii) each component of a code is statistically independent from the other components.

To achieve this, we start by training an autoencoder with a latent space of size 1. Once this is trained, we fix the values of this first element in the latent space, and train an autoencoder with a latent space of size 2, where only the second component is trained. At each step, the decoder is discarded, and a new one is trained from scratch. This continues until we reach the required latent space size (see Figure 1 for an illustration of this approach). Therefore, **"importance" in this context refers to the ℓ_2 reconstruction error** : the first element is the one which has the most impact on this reconstruction error.

Secondly, we add a latent space covariance loss term to the autoencoder loss **to ensure that each latent component is statistically independent from the others**. If the intrinsic characteristics of the data are distributed independently throughout the dataset, then this will be reflected in the PCA-AE latent space. The final objective is to create an autoencoder whose latent space efficiently separates (disentangles) independent characteristics of the data being considered. For example, this could be properties such as size, shape or colour, or more high-level characteristics such as gender or hair colour in the case of images of faces. We achieve this without any reference to labels relative to these characteristics. Instead, we aim to discover the latter in a completely unsupervised fashion, through the data itself.

To summarise, in this paper we propose the following contributions:

- An algorithm to create a autoencoder with a latent space where the components of the latent code are ordered in

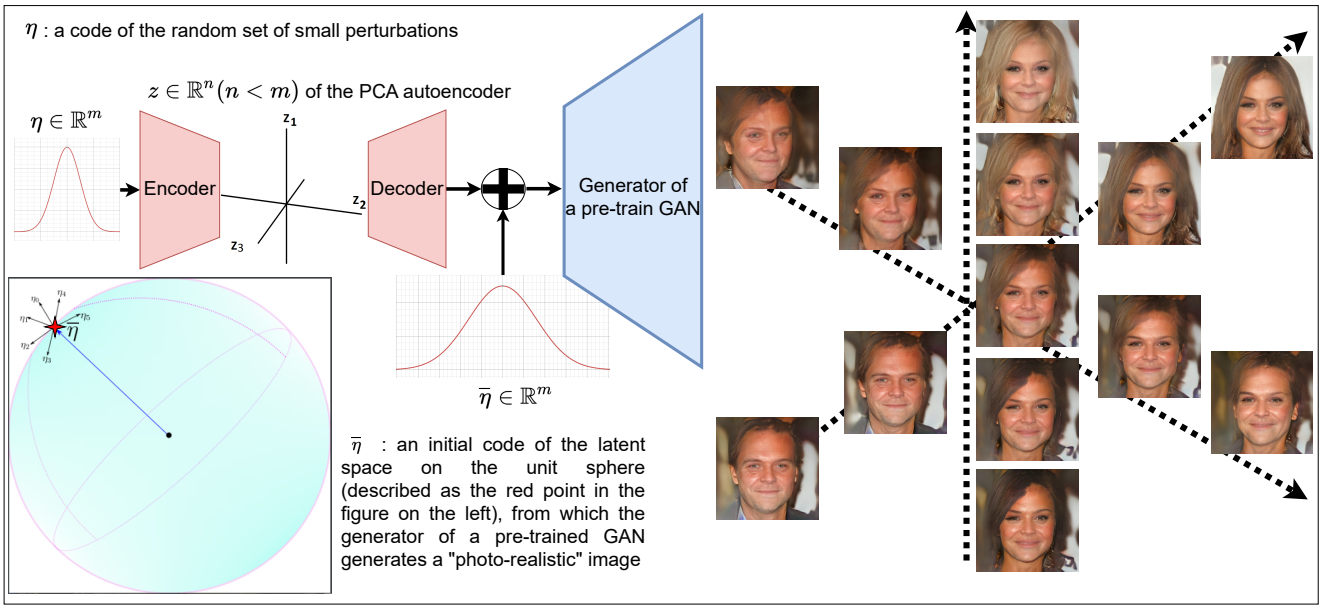


Fig. 2: PCA-AE is applied for navigating in the latent space of a pre-trained GAN. Each component of the PCA-AE attempts to control one attribute of generated images.

terms of decreasing importance to the data. This importance refers to the ℓ_2 reconstruction error
- We use a covariance loss term to encourage the components of the latent space to be statistically independent to decrease entanglement;
- We show how the PCA-AE can be used to organise and disentangle the latent space of a pre-trained generative network such as a GAN. An illustration of this can be seen in Figure 2

In other words, we wish both to impose an order on and disentangle the latent space. We demonstrate the efficiency of our autoencoder on synthetic examples of images of geometric shapes as well as on the more complex data of the CelebA dataset. In the first case, we show that the resulting autoencoder retrieves meaningful axes that can be manipulated to change different geometric characteristics (size, rotation) of the shapes. In the second, we automatically discover properties such as hair colour, gender and pose. We emphasise that this is done in a **completely unsupervised manner, without any access to the labels of these characteristics**. In this work, we wish to discover these underlying properties automatically, by letting the data indicate its different variable characteristics.

2 Previous work

Broadly speaking, there are two main categories of networks which are used for image editing and synthesis: autoencoders [21,34,9,28] and GANs [33,1,11,4,31,43,30,7]. The goal of autoencoders is to compress and decompress data to and from a compact, powerful latent space. GANs, on the other hand, fix the latent space with an a priori distribution (for example Gaussian), and attempt to create realistic data with the parallel action of a generator and an adversarial network. These models have produced impressive results, and therefore understanding and interpreting their latent spaces is now an extremely hot topic. Ideally, we would like to understand what kinds of hidden representations the model has learned. More precisely, the latent space should be disentangled so that one latent code represents one factor of the variation in the formation of the data space.

Many previous works concerning such models have the goal of improving the compactness and power of latent spaces. Firstly, a commonly remarked behaviour of autoencoders is that they fill up all the space allowed in their latent space, which is detrimental to interpretation and manipulation. A common solution to this problem is to allow the autoencoder more space than is likely necessary, and then try to impose some sort of structure on the latent space. Sparse autoencoders [34,9,28], for example, attempt to have as few active (non-zero) components as possible in the latent space. However, while this forces compactness, the autoencoder can still entangle several data characteristics in a single latent component. Generative autoencoders such as variational autoencoder [21], Wasserstein autoencoder [41] create an autoencoder whose latent space is encouraged to follow a certain predefined distribution. While this is very useful for the purposes of synthesis, this does not in itself improve the

interpretability of the latent space components, which can mix several characteristics.

Many previous works exist on the specific task of disentangling latent representations. Rifai et al [36] employ contractive autoencoders to learn locally invariant features at multiple resolutions, which is then given to a "contractive discriminant analysis" block for the purpose of emotion prediction. Reed et al [35] propose a Boltzmann machine to discover underlying variation in data with two strategies. Firstly, they include the data labels in their cost function for the Boltzmann machine, and secondly, they "clamp" (impose) a code for two data points which are known to share some characteristics. The work of Cheung et al [5], Kumar et al [24] and Lezama [26] are the most similar previous works to ours, in certain aspects. In particular, these works employ some form of covariance loss. Cheung et al use a semi-supervised autoencoder to output an image and at the same time predict a class. Kumar et al propose the covariance loss for the latent space to decorrelate its dimensions, leading to match the moments of the distributions of data and the latent space. Lezama et al use a loss on the Jacobian of an autoencoder output with respect to the latent code, to encourage the code to follow the desired class, as well as a prediction loss using binary classes. Lample et al [25] proposed Fader networks, which try to isolate a single image characteristic in a single latent component, with an innovative use of a discriminator network. This produces a network where the characteristic can be effectively controlled with a slider. In the case of the work of β-VAE$_B$ [13], β-VAE$_H$ [2], FactorVAE [20] and β-TCVAE [3], propose frameworks or regularisation to disentangle VAE by modelling and weighting the Kullback-Leibler divergence term to encourage factorised representations in the latent space.

3 Principal Component Analysis Autoencoder

Before describing the PCA-AE, we first set out some notation. Let $\mathcal{X}$ be the data space, in general, we will consider images x of size $m = s \times s$, so $\mathcal{X} = \mathbb{R}^m$. We note with $\mathcal{Z} = \mathbb{R}^n$ the latent space, n being the dimensionality of this latent space. We denote the encoder with $E : \mathcal{X} \to \mathcal{Z}$, and the decoder with $D : \mathcal{Z} \to \mathcal{X}$. Let $z \in \mathcal{Z}$ be a latent vector (or code). We denote with z_i the i^{th} *component* of z. We will refer to this as a *latent component*. Let $y = D \circ E(x)$ be the output of the autoencoder. The standard autoencoder loss, also called the *reconstruction loss*, is given by (for each sample x):

$$\|x - D \circ E(x)\|_2^2. \tag{1}$$

Now, we describe the core idea and algorithm of PCA-AE. As we explained above, we wish to organise the latent space according to two principles:

- Decreasing order of "importance"
- Statistical independence of the components

If we consider importance to mean variability, then in the case where the data is drawn from a multi-dimensional Gaussian distribution, the PCA achieves these two goals. Therefore, we shall start by describing an algorithm to perform the PCA as an illustrative example, which we will then translate directly into an algorithm to train our PCA-AE.

Suppose that a random variable x is drawn from a Gaussian distribution with 0 mean and a covariance matrix A. For the sake of simplicity, we suppose that A is positive definite with no repeated eigenvalues. We call these eigenvalues $\gamma_1 > \cdots > \gamma_n$, paired with (normed) eigenvectors $v_1, \ldots, v_n$. Performing the PCA is equivalent to discovering the vectors v and the eigenvalues γ (since $A = \sum \gamma_i v_i^T v_i$).

A possible algorithm to find the γ's and v's in decreasing order of importance is to find a unit vector v such that:

$$\mathbb{E}\left[\|x - \langle x|v\rangle\, v\|_2^2\right] \tag{2}$$

is minimal. The solution v to this optimisation problem will be v_1 (up to a sign), and the associated eigenvalue can be evaluated as:

$$\gamma_1 = \mathbb{E}\left[\langle x|v\rangle^2\right] \tag{3}$$

Once v_1 is found, we can look for a unit vector v such that

$$\mathbb{E}\left[\|x - \langle x|v_1\rangle\, v_1 - \langle x|v\rangle\, v\|_2^2\right] \tag{4}$$

is minimal and this will be the vector v_2. This process can be repeated as long as desired, until reaching dimension n at most, keeping in mind that smaller eigenvalues will be affected by noise and accumulated numerical errors.

With this in mind, translating this algorithm into a PCA-AE training algorithm is straightforward. If we define $E_1(x) = \langle x|v_1\rangle$ as an encoder and $D_1(t) = tv_1$ as a decoder, then the first step of the PCA would correspond to finding the parameter v_1 such that

$$\mathbb{E}\left[\|x - D_1 \circ E_1(x)\|_2^2\right] \tag{5}$$

is minimal. If we consider a database composed of N vectors x_i, this translates into the context of machine learning to minimising the following loss

$$\sum_{i=1}^{N} \|x_i - D_1 \circ E_1(x_i)\|_2^2 \tag{6}$$

Finally, the full generalisation of importance is achieved by letting the the encoder E_1 and the decoder D_1 have a more general and powerful architecture (ie. a neural network) than a simple projection of x onto a direction v_1. If we denote with θ_1 the set of parameters of the encoder and decoder, our loss will become

$$\mathcal{L}(\theta_1) = \sum_{i=1}^{N} \|x_i - D_1 \circ E_1(x_i)\|_2^2 \tag{7}$$

Algorithm 1 PCA-AE algorithm. Note, we have described the algorithm with a simple gradient descent, but any descent-based optimisation can be used (Adam, Adagrad etc)

Require:
Regularisation coefficient $\lambda_{cov} > 0$. Maximum latent space size n. Initialise the parameters θ_i of the encoders E_i and the decoders D_i.

while θ_1 *not converged* **do**
  Sample $\{x_1, \dots, x_N\}$ from the training set.
  Update E_1 and D_1 by minimising:

$$\mathcal{L}(\theta_1) = \frac{1}{mN} \sum_{i=1}^{N} \|x_i - D_1 \circ E_1(x_i)\|_2^2$$

for $k = 2, \dots, n$ **do**
  while θ_k *not converged* **do**
    Sample $\{x_1, \dots, x_N\}$ from the training set.
    Update E_i and D_i by minimising:

$$\mathcal{L}(\theta_k) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{m} \|x_i - D_k \circ (E_1(x_i), \dots, E_k(x_i))\|_2^2 \right) + \lambda_{cov} \mathcal{L}_{cov}(\theta_k)$$

In the case of PCA, the "encoders" $\langle x|v_1\rangle$, $\langle x|v_2\rangle$, ... are fixed when discovering the next eigenvector v_{k+1}. Following this idea, we freeze the encoders $E_1, \dots, E_k$, each of which providing one of the dimensions of the latent space, before training the encoder E_{k+1}.

To summarise, in the case of PCA-AE, the notion of "importance" corresponds to the ℓ_2 reconstruction error. We impose decreasing importance in a manner similar to PCA by training and then freezing the encoders (which represent the vectors v of the PCA) progressively until reaching the chosen latent space size.

Now we need to address the second requirement of our PCA-AE: how to impose statistical independence on the latent codes. This is done in the following manner: **we require that the covariance matrix of the vector z to be as close as possible to the identity matrix**. In other words, we minimise the correlations between the latent components.

In order to reduce the computational burden we can, without loss of generality, impose a *batch normalisation* [17] (BN) layer to the latent vector z, but with the training parameters α and β fixed, such that the mean of each latent component z_i of z is 0 mean and the variance is 1.

The magnitude of the off-diagonal entries of the covariance matrix can then be simply expressed as $\sum_{i \neq j} \left(\mathbb{E}(z_i z_j)\right)^2$ where i and j range through the dimensions of z. We recall that we are adding a new dimension to our latent space while freezing the first dimensions. Therefore, imposing the independence between the components of the vector z boils down to minimising:

$$\sum_{i<k} \left(\mathbb{E}(z_i z_k)\right)^2 \tag{8}$$

where k is the current dimension being added. This, in turn, can be translated into a loss term, by replacing the expectation by a mean over the whole dataset, giving our final covariance loss:

$$\mathcal{L}_{\text{cov}}(\theta_k) = \sum_{j=1}^{k-1} \left(\sum_{i=1}^{N} E_j(x_i) E_k(x_i) \right)^2 \tag{9}$$

In practice the sum over the dataset is replaced by a sum over the mini-batch, similarly to what is done in Batch Normalisation.

This gives the following loss to minimise at step k:

$$\mathcal{L}(\theta_k) = \sum_{i=1}^{N} \left(\|x_i - D_k \circ (E_1(x_i), \dots, E_k(x_i))\|_2^2 \right) + \lambda_{cov} \mathcal{L}_{\text{cov}}(\theta_k), \tag{10}$$

where the parameters θ_k are the parameters of the new 1-dimensional encoder E_k and the completely new decoder D_k, and where λ_{cov} is a weighting factor. We chose to discard the previous decoders because we wish to give the highest degree of freedom possible to the reconstruction part, while we keep the first computed dimensions of our latent space, since they were determined as being the most effective to reconstruct the input.

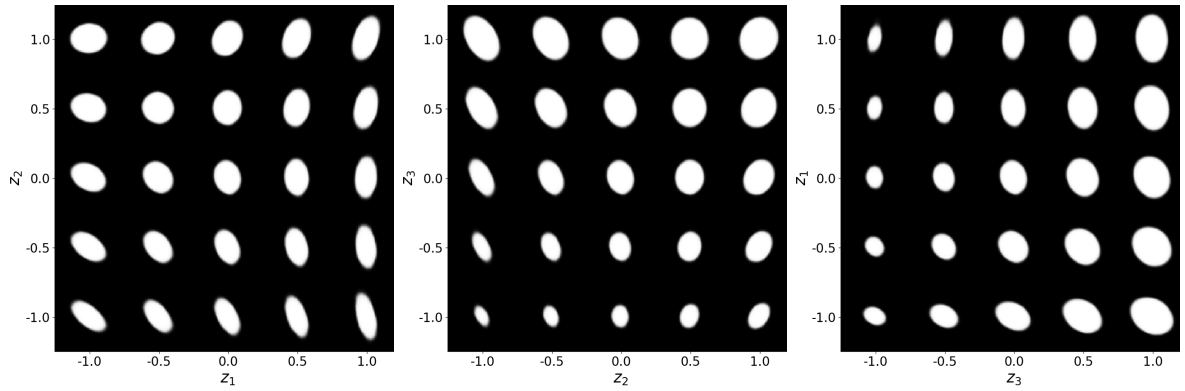
The pseudo-code for our algorithm can be seen in Algorithm 1. Note that we use the mean squared error (MSE), since it is the default setting for neural network packages (we used Pytorch), so we have added the normalisation factor m. In this pseudo-code, we do not specify the minimisation scheme, but any gradient-descent based algorithm can be used (we used the Adam optimiser [22]).

4 PCA-AE for GAN

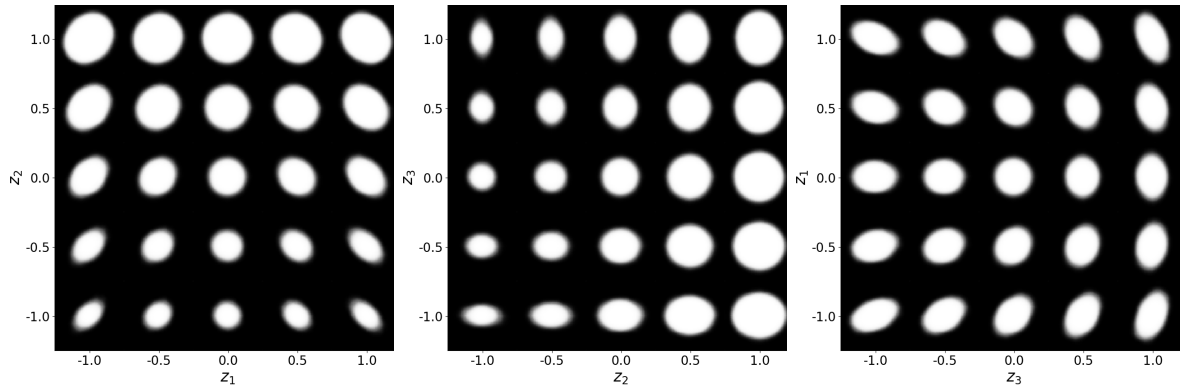
The objective of the generator of GANs is to find a mapping from the latent distribution p_z into the image data distribution

Fig. 3: Interpolation in the original latent space of PGAN (with 512 components). From the initial code $\overline{\eta} = [\overline{\eta}_1, \overline{\eta}_2, \overline{\eta}_3, ..., \overline{\eta}_{512}]$ as shown in the middle column, we adjust the n^{th} component by adding a constant shown above the image, other codes are not shown that are fixed. We can see that it is difficult to interpret this latent space. Several attributes such as hair colour or head pose are varied within the same component of the latent space of PGAN.

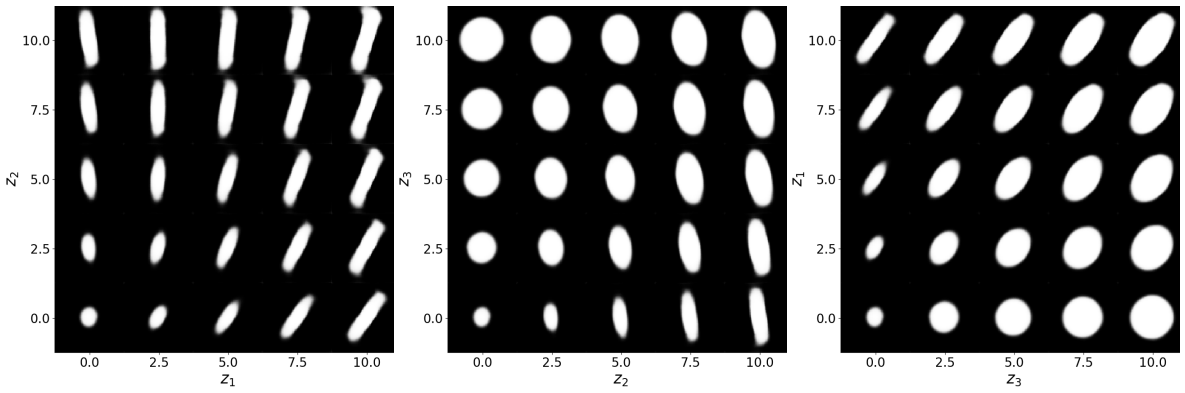
p_{data}. Ideally, we would like each latent component to correspond to one factor of variation in the data. In practice, the latent representations of GANs are entangled. In Figure 3, we show several examples of interpolation in the original latent space of Progressive Growing of GANs (PGAN) [18], which is a GAN-based approach for generating high quality images, before applying our PCA-AE. We visualise images generated by PGAN while varying one latent component at a time. We can see that it is difficult to interpret this latent space. For example, we can see a blond woman at both the first and the last components of the latent space, and the woman in the generated images changes the pose of her head when either the second and last components are varied. It is clear that this latent space is heavily entangled, with several characteristics modified by changing one component. This makes it difficult to understand, navigate and manipulate the latent space. Addressing these problems is precisely the goal of the present work. In order to organise and disentangle this latent representation, we apply the PCA-AE to the latent space of a pre-trained GAN. Indeed, we do not intend to create a new GAN architecture which can compete with state-of-the-art generators such as PGAN, rather we propose to use our PCA-AE to better understand and organise the latent space of a high quality, pre-trained GAN. In other words, since the



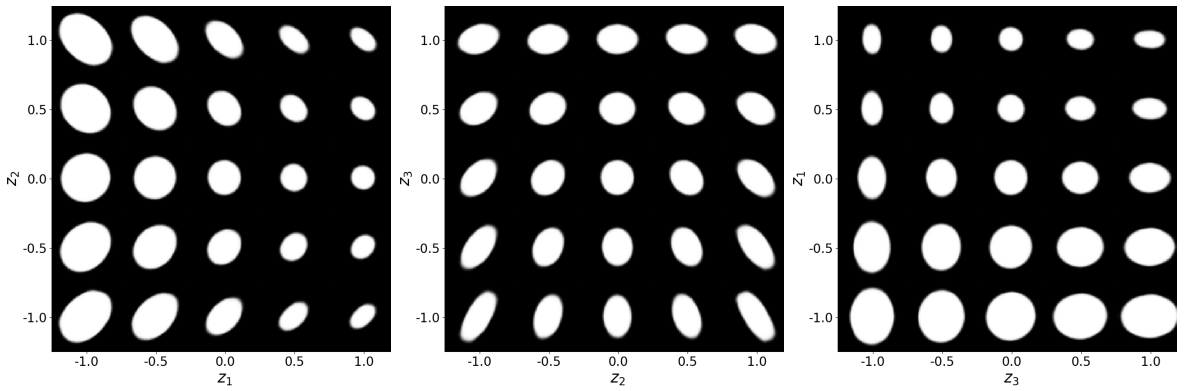
(a) VAE



(b) β-VAE



(c) β-TCVAE



(d) PCA-AE

Fig. 4: Interpolation in latent space w.r.t image reconstruction, ellipses with rotation (three parameters) of VAE, β-VAE, β-TCVAE and our proposed method. The PCA-AE can create a meaningful latent space where different geometric attributes are separated (*i.e.* the 1^{st} component corresponds to the area and the next two parameters are the ratios of the ellipses' axes in different directions).

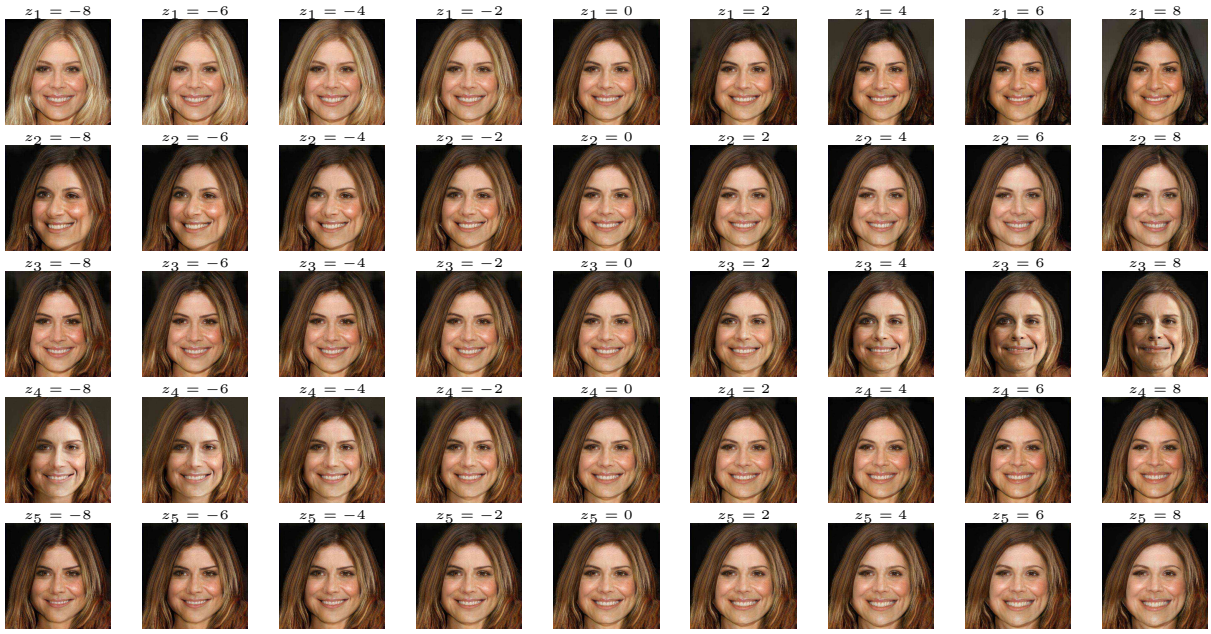


Fig. 5: Results of navigation in the latent space of the PCA-AE for a pre-trained PGAN. We trained this PCA-AE around the code $\overline{\eta}$ corresponding to the middle column. On each row, we have modified a single component (the other components are set to 0). We see that the component z_1 of the latent space z of the PCA-AE represents hair colour, while z_2 corresponds head poses, and in this case z_3 seems to correspond to gender and z_5 to the mouth posture.

problem of simultaneously learning and organising the latent space is too difficult, we propose to learn first and organise afterwards. The learning part is done during the training of the high-quality GAN. This difficulty may arise from the fact that more complex data, such as faces, may need larger increments of latent space size to achieve good results. In this case, it is easier to rely on the pre-trained GAN.

Let us highlight that the strategy we propose can be easily adapted to analyse any GAN, and we have chosen PGAN in the existing work due to its impressive performances. The input sample to the PGAN lives in $\mathbb{R}^{512}$ (the PGAN latent space), or more precisely is a random sample from the normal Gaussian distribution in dimension 512. Since the input is normalised in the first operation of PGAN's generator during testing, we can assume that the latent codes are drawn uniformly from a sphere, which is not convex. To make the job of the autoencoder easier, and since the latent space is not convex, we will apply our tool locally around a given point from the latent space. More precisely, let $\overline{\eta}$ be a fixed point of this sphere (see Figure 2). Let G be the generator of PGAN and η be a small perturbation vector (drawn randomly). Our goal is to design a low dimensional autoencoder E, D that minimises the following loss :

$$\begin{aligned}\mathcal{L}(\theta) =& \|G(\eta+\overline{\eta}) - G(D\circ E(\eta)+\overline{\eta})\|_2^2 \\ &+ \lambda_{cov}\mathcal{L}_{\text{cov}}(\theta)\end{aligned} \tag{11}$$

where θ are the parameters of the PCA-AE.

In other words, the autoencoder's goal is to produce a vector $D\circ E(\eta)+\eta$ which leads to an image that is as close as possible to $G(\eta+\overline{\eta})$. The vector $D\circ E(\eta)$ will have passed through the low dimensional internal representation of the autoencoder, which is well-organised, where $dim(E(\eta)) < dim(\eta)$. The covariance loss $\mathcal{L}_{\text{cov}}$ in Equation (11) is defined as in (9) and will encourage disentanglement of the latent space of the pair E, D. We apply the same training strategy that consists in iteratively increasing the number of latent components, while freezing the first components. This training process is illustrated in Figure 2.

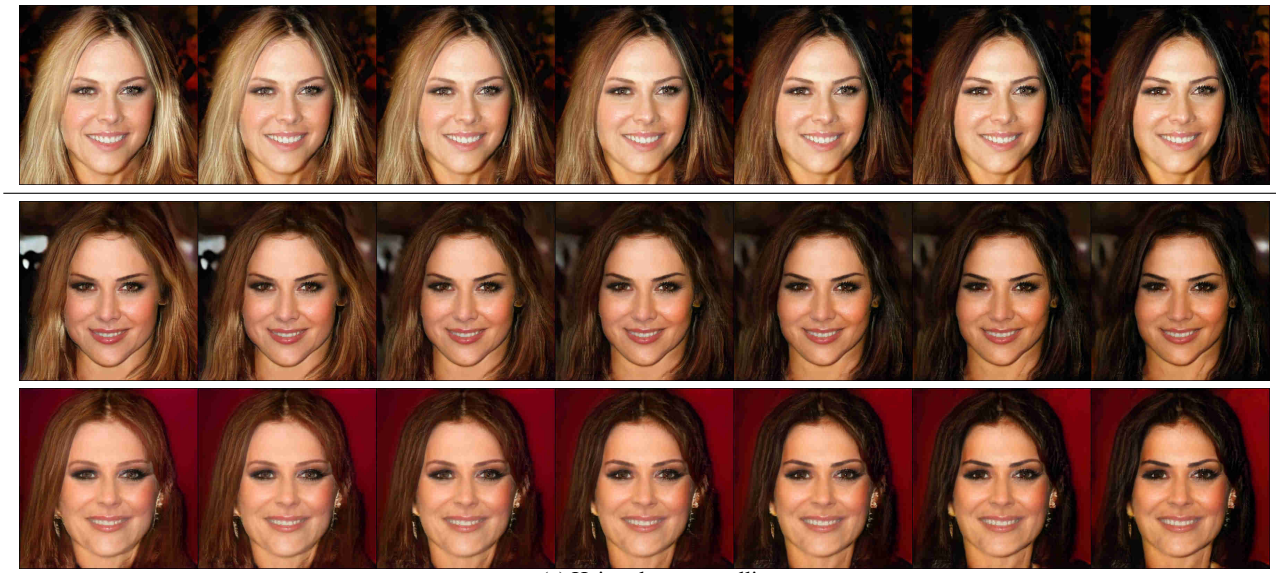
5 Results

In this section, we present the results of our PCA-AE, and we compare with those of VAE [22], β-VAE$_B$ [13], β-VAE$_H$ [2], FactorVAE [20] and β-TCVAE [3] [1]. Note that other approaches to disentangling the latent space use data labels, which we wish to avoid here : our goal is to discover the variability of the data in an unsupervised fashion.

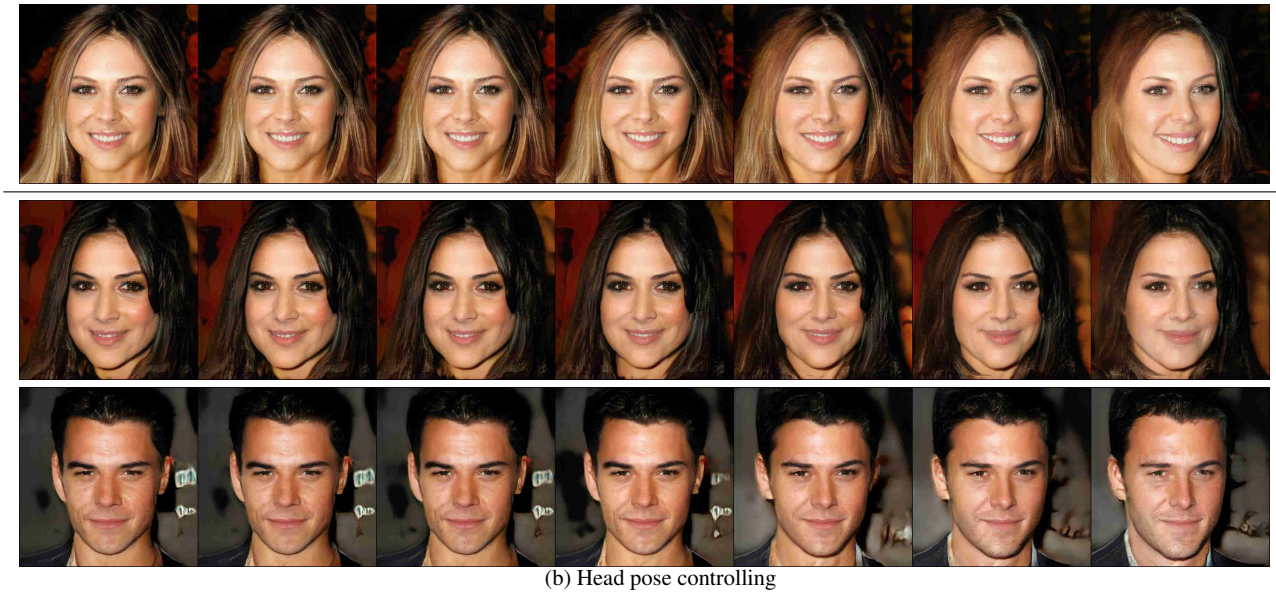
5.1 Disentanglement evaluation

We propose to use the absolute Pearson correlation coefficient (PCC) as a disentanglement evaluation to verify the

[1] https://github.com/YannDubs/disentangling-vae



(a) Hair colour controlling



(b) Head pose controlling

Fig. 6: Application of PCA-AE for PGAN: Transferring the learning attributes from the training code (the first row) to other testing codes (the last rows) for (a) hair colour and (b) head pose. The first row shows that we change the hair colour of the generated image from PGAN with respect to a training initial code $\overline{\eta}$ by adjusting the first component of the latent space of PCA-AE. We can see that the hair colours and the head pose of generated images from other testing initial codes are also changed as those of the training code.

relationship between the attributes of image data and the components of the trained latent space. Given a pair of random variables $(Attr(x), z_i)$ where $Attr(x)$ is the attribute of image x and z_i denotes the i^{th} component of the latent space z, the absolute PCC $\rho(Attr(x), z_i)$ is computed as:

$$\begin{aligned}\rho(Attr(x), z_i) &= \left| \frac{cov(Attr(x), z_i)}{\sigma_{Attr(x)}\sigma_{z_i}} \right| \\ &= \left| \frac{\mathbb{E}[(Attr(x) - \mu_{Attr(x)})(z_i - \mu_{z_i})]}{\sigma_{Attr(x)}\sigma_{z_i}} \right|\end{aligned} \tag{12}$$

where $\sigma_{Attr(x)}$ and σ_{z_i} denote the standard deviation of $Attr(x)$ and z_i, respectively. $\mu_{Attr(x)}$ and μ_{z_i} are the mean of $Attr(x)$ and z_i, respectively. The absolute PCC ranges from 0 to 1.

5.2 Experimental setup and results on synthetic data

In order to find out whether our PCA-AE is able to capture meaningful components which correspond to the parameters of visual objects, we have first tested our algorithm on syn-

(a) Generated images from StyleGAN.The first row displays several source images, which impose the content of the output images. This content is then mapped to the style of the left-bottom image. The output images are all those in the second row, starting from the second column.

(b) We show an example of the interpolation in the latent space $\mathcal{W}$ towards 16 dimensions of StyleGAN. Note that we use the model of resolution 512×512. Each row corresponds to a dimension w_i where $1 \leq i \leq 16$. The first column of a row is the original generated image from an initial point in the latent space $\mathcal{W}$. The other columns show the images that are generated by adding perturbations to the initial point towards the direction w_i.

Fig. 7: Results of StyleGAN.

thetic data of grayscale images of geometric shapes which are centred in the image, with a single shape per image. We have created images of ellipses in the case of three parameters: two axes, and rotation. The two ellipse axes a and b are sampled from a uniform distribution on the interval $(0, \frac{m}{2})$ (where $m \times m$ denotes the size of image), and the rotation angle Θ from a uniform distribution on the interval $(0, \frac{\pi}{2})$. In these experiments, we set n (maximum autoencoder dimension) to 3 (the number of parameters used to create the dataset). A drawback of using data with binary images of shapes is that we have a limited number of centred parametric shapes that we can create, even though we sample the parameters from a continuous space. To solve this problem, we blur the binary shapes slightly with a Gaussian filter with $\sigma = 0.8$ pixels, allowing us to create as many images as we wish.

Figure 4 shows decoded images of interpolated points in the latent space, in the case of ellipses. Table 1 shows the numeric evaluation based on the absolute PCC between the

	PCA-AE with respect to three considered attributes		
	A	R1	R2
z_1	**0.99**	0.15	0.16
z_2	0.00	0.06	**0.61**
z_3	0.00	**0.72**	0.06

	Area of ellipses (A)						
	AE	VAE	β-VAE$_B$	FactorVAE	β-TCVAE	PCA-AE ($\lambda_{cov} = 0$)	PCA-AE
z_1	0.52	0.00	0.15	0.01	0.10	**0.99**	**0.99**
z_2	0.00	0.33	0.90	0.07	0.14	0.01	0.00
z_3	0.64	0.83	0.06	0.89	0.86	0.55	0.00

Table 1: Evaluation of the absolute PCC between the attributes of ellipses with respect to three components (z_1, z_2 and z_3) of the trained latent space. We consider three attributes: the area (A), the ratio of two diameters towards vertical and horizontal directions (R1), the ratio of two diameters towards diagonal directions (R2). In the left table, bold font denotes the largest value among the components. In the right table, the strongest correlation (the PCA-AE's) is in bold font. We can see that each component of PCA-AE is strongly correlated with only one ellipse attribute.

attributes of ellipses with respect to three components of the trained latent space. We observe that the latent space of our PCA-AE corresponds to three principal attributes of ellipses : area (A), the ratio of two diameters towards vertical and horizontal directions (R1), the ratio of two diameters towards diagonal directions (R2). The compared methods also create a meaningful latent space whereas AE and VAE learn a latent space where the intrinsic parameters of the ellipses are mixed up. While these are not the parameters with which we created the images (indeed, the autoencoder has absolutely no way of knowing what representation to choose, and we cannot impose one in an unsupervised setting), they are indeed independent; for a given area, the ratio between the axes is an independent parameter, and vice versa. This gives us a way to interpolate in the latent space in a meaningful manner. These independent parameters are sufficient to describe the ellipse, and each axis is hierarchically more interpretable and navigable than in the case of other methods.

Table 1 also shows an ablation study which compares the PCA-AE with the baselines such as a standard AE and our PCA-AE with no covariance loss (*i.e.* $\lambda_{cov} = 0$). We can see that more than one component of the latent space of AE the PCA-AE with no covariance loss controls the area of the ellipses. In the case of the PCA-AE with no covariance loss, the first and the third component of its latent space correspond to the area attribute simultaneously. This confirms the need of the proposed covariance loss.

5.3 Experimental setup and results of the PCA-AE applied to the latent space of PGAN

Note that in Figure 9, we also display the results of our PCA-AE directly to the Celeba data. This gives very blurry results (since the task is very difficult), similar to the results of β-VAE [13] (Figure 4 of their paper), which lead us to our approach to using the PCA-AE applied to pretrained GANs. Therefore, to show the use of our PCA-AE on more high-level data, we take a pre-trained model of PGAN [18] [2] trained with the CelebA dataset [27]. Note that the pre-trained generator is fixed during the training of our PCA-AE. The latent space of PGAN is entangled (we show experiments to support this in Figure 3), so that a variation along one parameter of this initial code in the latent space can modify several characteristics of the generated images. The latent space size of this pre-trained network is 512. An initial code $\overline{\eta}$, from which the network generates a photo-realistic image, is chosen. In order to create the set of random perturbations, we sample from a multivariate Gaussian distribution :

$$\eta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}) \tag{13}$$

where $\boldsymbol{I}$ is the identity matrix.

We now show our results of PCA-AE for organising the latent space of the pre-trained PGAN [18]. We show an example of the navigation of the latent space of the PGAN in Figure 5. This is achieved by training our PCA-AE around the code generating the image at the middle of the three grids. We can see that for this example, the first component (z_1) corresponds to the hair colour from black to blond, the second one (z_2) controls the head pose and the third parameter (z_3) changes the gender.

In order to better visualise the results of the proposed method, we adjust two components which correspond specifically to hair colour and head poses of generated images from the training initial code as shown in the first row of each sub figure of Figure 6. Then, we apply the trained model to other initial codes of the latent space of PGAN. We can see that the attribute of generated images from testing initial code also change as those of the training code (described as the last rows).

We have also compared our method to the approach of StyleGAN[19], using an available code [3]. We denote $\mathcal{W}$ as the disentangled latent space of StyleGAN. The authors of StyleGAN propose a multi-scale architecture, which, they posit, encourages better separation of attributes in the latent space. They propose a style transfer algorithm that basically

[2] Pytorch GAN zoo: `https://github.com/facebookresearch/pytorch_GAN_zoo`

[3] StyleGAN Code: `https://github.com/rosinality/style-based-gan-pytorch`

Co.	AE			β-TCVAE			FactorVAE			VAE			PCA-AE		
	HC	HP	GE	HC	HP	GE	HC	HP	GE	HC	HP	GE	HC	HP	GE
z_1	0.20	**0.53**	0.02	0.35	**0.80**	0.04	0.07	0.46	**0.53**	0.35	**0.81**	0.07	**0.70**	0.03	0.14
z_2	**0.36**	0.21	0.04	0.05	0.26	0.25	0.04	0.17	0.03	0.05	0.24	0.24	0.07	**0.80**	0.13
z_3	0.28	0.36	0.23	0.13	0.04	0.04	0.09	**0.66**	0.37	0.13	0.02	0.02	0.16	0.15	**0.56**
z_4	0.20	0.19	**0.58**	**0.53**	0.19	**0.53**	**0.70**	0.11	0.14	**0.59**	0.23	**0.57**	0.03	0.24	0.05
z_5	0.34	0.21	0.49	0.07	0.15	0.33	0.01	0.28	0.12	0.07	0.11	0.35	0.05	0.22	0.09

Table 2: Quantitative evaluation of the correlation of latent components with high-level attributes. We have calculated the PCC between the latent components of AE, VAE-based methods and PCA-AE, and three attributes: head pose (HP), hair colour (HC) and gender (GE). We can see that the components of PCA-AE are correlated with one dominant attribute of the semantic feature.

mixes two latent codes. The result of this approach can be seen in Figure 7a. However, this does not provide a way to navigate in the latent space. To illustrate this, we have added a perturbation in one of the scales in the latent space $\mathcal{W}$ of StyleGAN, starting from a generated latent point. The perturbation is drawn from a normal distribution. In this manner, we can see if the attributes are effectively disentangled. The results of this experiment can be seen in Figure 7b. While some scales correspond to attributes such as colours (w_{16}), artefacts appear quickly, and other high-level attributes do not appear naturally. Our method, on the other hand, aims to impose each component of the sub-latent space which corresponds to one attribute of generated images.

In order to evaluate the disentanglement of the latent space of other methods and ours, we use pre-trained classifiers to determine an attribute of generated images. We choose three main attributes which the classifiers [15] can recognise well, corresponding to the head pose (*i.e.* turning left to right), hair colour (*i.e.* black, brunette and blond) and gender. To demonstrate the performance of our algorithm, we have trained the standard AE, the aforementioned VAE-based methods and our proposed PCA-AE, using the procedure described in Section 4. Table 2 shows the numeric evaluation of the methods and Figure 8 shows the generated images of the generator of PGAN from the latent spaces of the other approaches and of our proposed PCA-AE. The other methods construct a latent space where the attributes of the generated images are correlated with more than one component. For example, we can see that the latent space of AE mixes up the attributes. In addition, it can be seen that the fourth parameter of β-TCVAE controls the hair color and the gender of generated images simultaneously. Respectively, the first parameter of FactorVAE changes the head pose. Then, the third one of this model still corresponds to the head pose. Indeed, the absolute PCC of this model for the head pose is correlated to the first and third components of the latent space. Our proposed PCA-AE yields a disentangled latent space which is organised in a hierarchical fashion: the first component corresponds to the colour hair of the generated images, the second one represents head poses (*e.g.* turning left and right), the third parameter corresponds to hair thickness and the last one is mildly correlated to skin tone. Our PCA-AE is able to efficiently separate the different facial attributes and rank them according to their importance in the reconstruction. Thus, the latent space created by our method is easier to interpret and navigate than the original GAN latent space.

We highlight that this procedure can be applied to any pre-trained model, so that the disentangling and organisation of the latent space can be carried out after the initial, computationally expensive, training of a GAN.

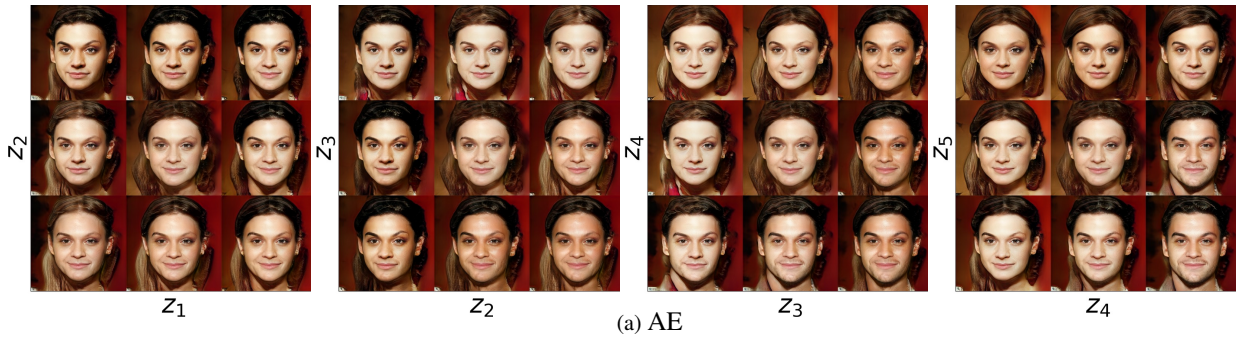
6 Discussion

In this paper, we have presented a novel autoencoder, the PCA-AE, where the latent space is organised according to decreasing importance, and where these components are statistically independent. The PCA-AE is trained with latent spaces of increasing sizes to ensure that we capture the properties of the data in decreasing order of importance, in an unsupervised manner. Furthermore, we have imposed statistical independence of the latent components by employing a covariance loss term, which we add to the standard autoencoder cost, to encourage a disentangled latent space. We have used synthetic data to illustrate that the PCA-AE learns a latent space which is interpretable and which can be interpolated in a meaningful manner with respect to the properties inherent in the data. We have applied our autoencoder to high quality face data, and have shown that this efficiently disentangles the latent space of a powerful pre-trained GAN by projecting it to another smaller, interpretable, latent space. The resulting model can manipulate one facial attribute on each component. Furthermore, the proposed method can be applied to any pre-trained generative model, so that the initial time-consuming training of a powerful model and the organisation of its latent space can be carried out separately. We hope that this work will contribute to the interpretation and manipulation of latent spaces of complex data.

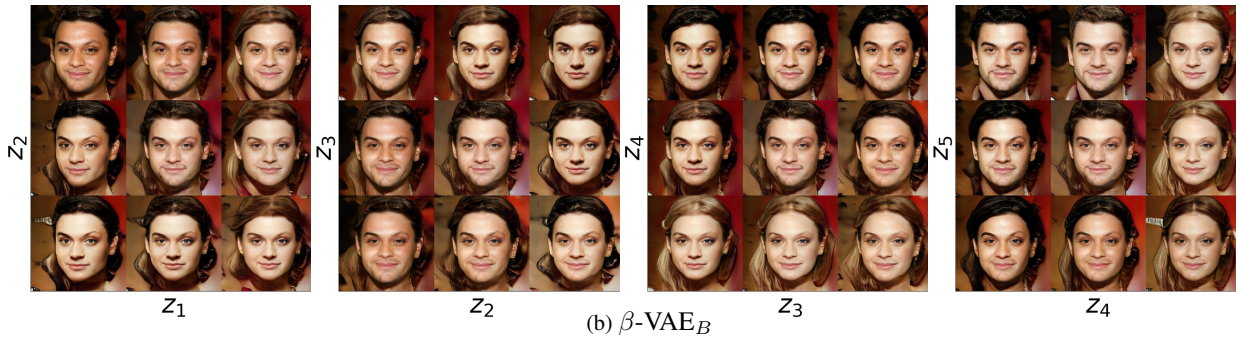
One limitation of our algorithm is that it is reliant on the hypothesis that the attributes which we wish to disentangle are distributed independently in the dataset. Indeed, if we have two attributes such as age and grey hair, and these are

naturally correlated in the data, it will be difficult for our algorithm to disentangle them, since it could decide to put both in one latent element. A possibility to address this problem is to take inspiration from recent multi-scale architectures [19], which can separate coarse from fine-grained details.

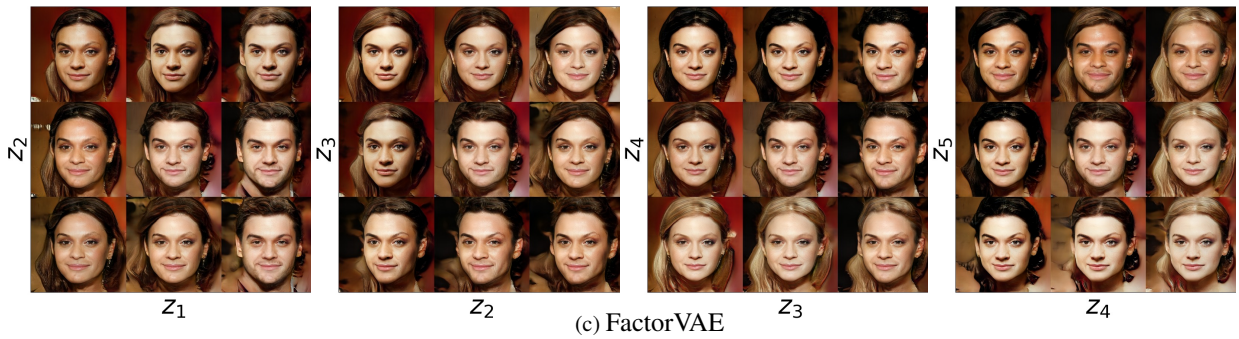
Another drawback of this approach is that we increase the latent space size by one at each step. This can be problematic in some cases, where the autoencoder needs a certain amount of freedom to learn a useful representation. Therefore, we could consider increasing the latent space by small packets of codes, to give it the freedom it needs. It is clear that the use of the ℓ_2 norm is not optimal to define the importance of a latent component. Indeed, in the case of the CelebA dataset as shown in Figure 9, applying the PCA-AE directly to the image data leads to very blurry results. Replacing the ℓ_2 norm reconstruction loss by an alternative, perceptual, metric could provide better results. Finally, the application of a PCA-AE trained in one region of a GAN latent space is not necessarily valid for another region. A future challenge will be to create a PCA-AE which is applicable to the whole space of the GAN.



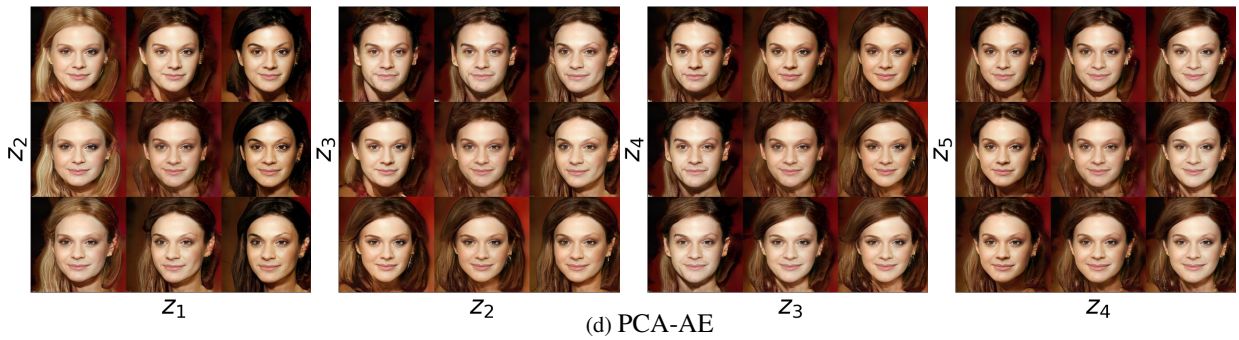
(a) AE



(b) β-VAE$_B$



(c) FactorVAE



(d) PCA-AE

Fig. 8: Interpolation in latent space of five components of AE, β-VAE$_B$, FactorVAE and the PCA-AE for the pre-trained PGAN [18]. Two components are adjusted along two axes, the others are set to zeros. We can see that VAE, β-VAE$_B$ mixes hair colour along two components z_1 and z_4, z_2 and z_5 respectively. Head pose corresponds to the components z_1 and z_4 of FactorVAE. Our method, on the contrary, shows that each component of our proposed latent space represents one attribute of the generated images. For example, z_1, z_2, z_3 correspond to hair colours, head poses and gender.

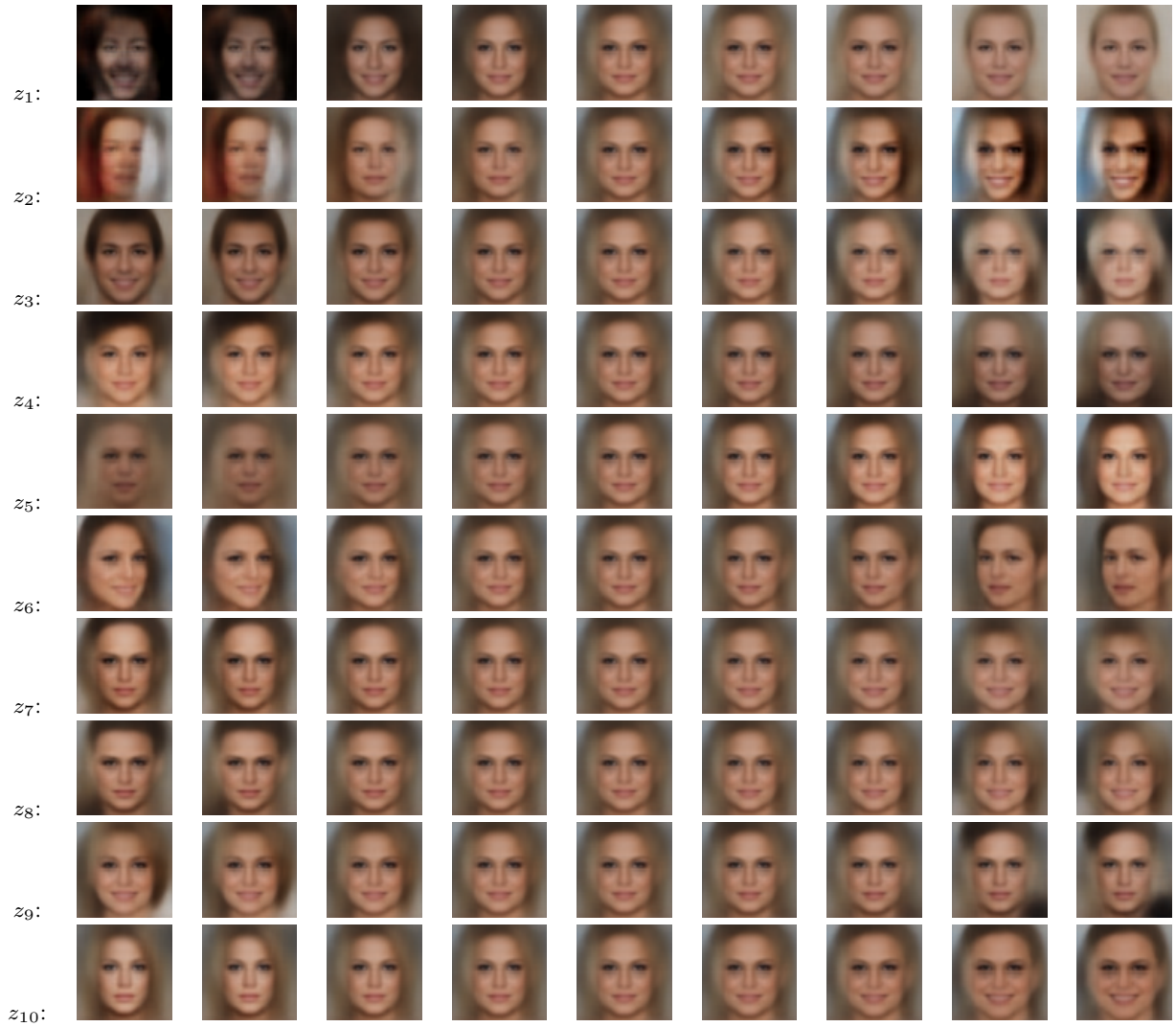


Fig. 9: Interpolation in latent space of ten components of a PCA-AE, applied directly to the CelebA dataset with the size image of 64×64. The code shown in the left side is used to adjusted, other codes are set to zeros. The middle column corresponding the images with the codes of all zeros. This leads to blurry results, which is why we chose to apply our PCA-AE a posteriori to a pre-trained GAN.

References

1. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.
2. Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. In *NIPS Workshop on Learning Disentangled Representations*, 2018.
3. Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.
4. Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.
5. Brian Cheung, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen. Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*, 2014.
6. Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
7. Quentin Delannoy, Chi-Hieu Pham, Clément Cazorla, Carlos Tor-Díez, Guillaume Dollé, Hélène Meunier, Nathalie Bednarek, Ronan Fablet, Nicolas Passat, and François Rousseau. SegSRGAN: Super-resolution and segmentation uing generative adversarial networks − Application to neonatal brain MRI. *Computers in Biology and Medicine*, page 103755, 2020.
8. Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems*, pages 4414–4423, 2017.
9. Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.
10. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
11. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.
12. Ari Heljakka, Arno Solin, and Juho Kannala. Towards photographic image manipulation with balanced growing of generative autoencoders. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3120–3129, 2020.
13. Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 2, page 6, 2017.
14. Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*, pages 1878–1889, 2017.
15. https://www.faceplusplus.com/. Face++ cognitive services.
16. Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems*, pages 52–63, 2018.
17. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
18. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.
19. Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
20. Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, 2018.
21. D. P. Kingma and M Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.
22. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
23. Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.
24. Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.
25. Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5967–5976, 2017.
26. José Lezama. Overcoming the disentanglement vs reconstruction trade-off via jacobian supervision. In *International Conference on Learning Representations*, 2019.
27. Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision*, 2015.
28. Alireza Makhzani and Brendan Frey. K-sparse autoencoders. In *International Conference on Learning Representations*, 2014.
29. Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.
30. Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4610–4617, 2019.
31. Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.
32. Chi-Hieu Pham, Carlos Tor-Díez, Hélène Meunier, Nathalie Bednarek, Ronan Fablet, Nicolas Passat, and François Rousseau. Simultaneous super-resolution and segmentation using a generative adversarial network: Application to neonatal brain MRI. In *International Symposium on Biomedical Imaging*, pages 991–994, 2019.
33. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.
34. Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann L Cun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1185–1192, 2008.
35. Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*, pages 1431–1439, 2014.

36. Salah Rifai, Yoshua Bengio, Aaron Courville, Pascal Vincent, and Mehdi Mirza. Disentangling factors of variation for facial expression recognition. In *European Conference on Computer Vision*, pages 808–822. Springer, 2012.
37. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.
38. Narayanaswamy Siddharth, Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.
39. Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738–3746, 2016.
40. Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.
41. Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.
42. Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.
43. Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016.
44. Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.