CONTROLLING GENERATIVE MODELS WITH CONTINUOUS FACTORS OF VARIATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent deep generative models are able to provide photo-realistic images as well as visual or textual content embeddings useful to address various tasks of computer vision and natural language processing. Their usefulness is nevertheless often limited by the lack of control over the generative process or the poor understanding of the learned representation. To overcome these major issues, very recent works have shown the interest of studying the semantics of the latent space of generative models. In this paper, we propose to advance on the interpretability of the latent space of generative models by introducing a new method to find meaningful directions in the latent space of any generative model along which we can move to control precisely specific properties of the generated image like position or scale of the object in the image. Our method is weakly supervised and particularly well suited for the search of directions encoding simple transformations of the generated image, such as translation, zoom or color variations. We demonstrate the effectiveness of our method qualitatively and quantitatively, both for GANs and variational auto-encoders.

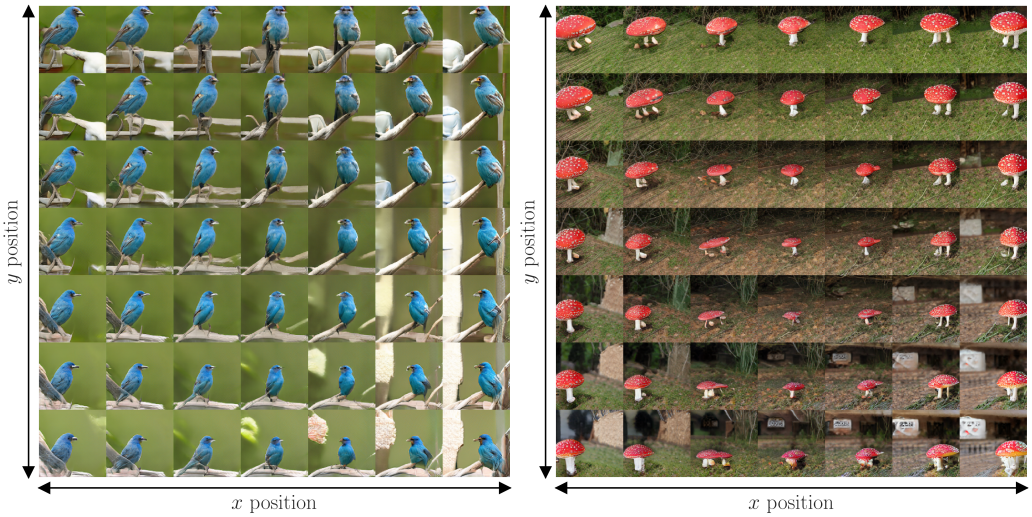

Figure 1: Images generated with our approach and a BigGAN model (Brock et al., 2018), showing that the position of the object can be controlled within the image.

1 INTRODUCTION

With the success of recent generative models to produce high-resolution photo-realistic images (Karras et al., 2018; Brock et al., 2018; Razavi et al., 2019), an increasing number of applications are emerging such as image in-painting, dataset-synthesis, deep-fakes... However, the use of generative models is often limited by the lack of control over the generated images. Indeed, more control could, for instance, be used to improve existing approaches which aim at generating new training examples (Bowles et al., 2018) by allowing the user to choose more specific properties of the generated images.

First attempts in this direction showed that one can modify an attribute of a generated image by adding a learned vector on its latent code (Radford et al., 2015) or by combining the latent code of two images (Karras et al., 2018). Moreover, the study of the latent space of generative models provides insights about its structure which is of particular interest as generative models are also powerful tools to learn unsupervised data representations. For example, Radford et al. (2015) observed on auto-encoders trained on datasets with labels for some factors of variations, that their latent spaces exhibit a vector space structure where some directions encode the said factors of variations.

We suppose that images result from underlying *factors of variation* such as the presence of objects, their relative positions or the lighting of the scene. We distinguish two categories of factors of variations. *Modal factors of variation* are discrete values that correspond to isolated clusters in the data distribution, such as the category of the generated object. On the other hand, the size of an object or its position are described by *Continuous factors of variations*, expressed into a range of possible values. As humans, we naturally describe images by using *factors of variations* suggesting that they are an efficient representation of natural images. For example, to describe a scene, one likely enumerates the objects seen, their relative positions and relations and their characteristics (Berg et al., 2012). This way of characterising images is also described in Krishna et al. (2016). Thus, explaining the latent space of generative models through the lens of factors of variation is promising. However, the control over the image generation is often limited to discrete factors and requires both labels and an encoder model. Moreover, for continuous factors of variations described by a real parameter t, previous works do not provide a way to get precise control over t.

In this paper, we propose a method to find meaningful directions in the latent space of generative models that can be used to control precisely specific continuous factors of variations while the literature has mainly tackled semantic labeled attributes like gender, emotion or object category (Radford et al., 2015; Odena et al., 2016). We test our method on image generative models for three factors of variation of an object in an image: vertical position, horizontal position and scale. Our method has the advantage of not requiring a labeled dataset nor a model with an encoder. It could be adapted to other factors of variations such as rotations, change of brightness, contrast, color or more sophisticated transformations like local deformations. However, we focused on position and scale as these are quantities that can be evaluated, allowing us to measure quantitatively the effectiveness of our method. We demonstrate both qualitatively and quantitatively that such directions can be used to control precisely the generative process and show that our method can reveal interesting insights about the structure of the latent space. Our main contributions are:

- We propose a new method to find interpretable directions in the latent space of generative models, which correspond to parametrizable continuous *factors of variations* of the generated image.
- We show that properties of generated images can be controlled precisely by sampling latent representations along linear directions.
- We propose a novel reconstruction loss for inverting generative models with gradient descent.
- We give insights of why inverting generative models with optimization can be difficult by reasoning about the geometry of the natural image manifold.
- We study the impacts of disentanglement on the controllability of generative models.

2 Latent Space Directions of a Factor of Variation

We argue that it is easier to modify a property of an image than to obtain a label describing that property. For example, it is easier to translate an image than to determine the position of an object within said image. Hence, if we can determine the latent code of a transformed image, we can compute its difference with the latent code of the original image to find the direction in the latent space which corresponds to this specific transformation as in Radford et al. (2015).

Let us consider a generative model $G : \boldsymbol{z} \in \mathcal{Z} \rightarrow \mathcal{I}$, with $\mathcal{Z}$ its latent space of dimension d and $\mathcal{I}$ the space of images, and a transformations $\mathcal{T}_t : \mathcal{I} \rightarrow \mathcal{I}$ characterized by a continuous parameter t. For example if $\mathcal{T}$ is a rotation, then t could be the angle, and if $\mathcal{T}$ is a translation, then t could be a component of the vector of the translation in an arbitrary frame of reference. Let $\boldsymbol{z}_0$ be a vector of $\mathcal{Z}$ and $I = G(\boldsymbol{z}_0)$ a generated image. Given a transformation $\mathcal{T}_T$, we aim at finding $\boldsymbol{z}_T$ such

that $G(\boldsymbol{z}_T) \approx \mathcal{T}_T(I)$ to then use the difference between $\boldsymbol{z}_0$ and $\boldsymbol{z}_T$ in order to estimate the direction encoding the factor of variation described by $\mathcal{T}$.

2.1 Latent Space Trajectories of an Image transformation

Given an image $I \in \mathcal{I}$, we want to determine its latent code. When no encoder is available we can search an approximate latent code $\hat{\boldsymbol{z}}$ that minimises a reconstruction error $\mathcal{L}$ between I and $\hat{I} = G(\hat{\boldsymbol{z}})$ ($\hat{I}$ can be seen as the projection of I on $G(\mathcal{Z})$) i.e.

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z} \in \mathcal{Z}}{\arg\min}\, \mathcal{L}(I, G(\boldsymbol{z})) \tag{1}$$

Solving this problem by optimization leads to solutions located in regions of low likelihood of the distribution used during training. It causes the reconstructed image $\hat{I} = G(\hat{\boldsymbol{z}})$ to look unrealistic[1]. Since $\boldsymbol{z}$ follows a normal distribution $\mathcal{N}(0, \boldsymbol{I}_d)$ in a d-dimensional space, we have $||\boldsymbol{z}|| \sim \chi_d$. Thus, $\lim_{d \to +\infty} \mathbb{E}\left[||\boldsymbol{z}||\right] = \sqrt{d}$ and $\lim_{d \to +\infty} Var\left(||\boldsymbol{z}||\right) = 0$. Hence, when d is large, the norm of $\boldsymbol{z}$ is approximately equal to $\sqrt{d}$. This can be used to regularize the optimization by constraining $\boldsymbol{z}$ to verify $||\boldsymbol{z}|| \leq \sqrt{d}$:

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z} \in \mathcal{Z}, ||\boldsymbol{z}|| \leq \sqrt{d}}{\arg\min}\, \mathcal{L}(I, G(\boldsymbol{z})) \tag{2}$$

2.1.1 Choice of the reconstruction error $\mathcal{L}$

One of the important choice regarding this optimization problem is that of $\mathcal{L}$. In the literature, the most commonly used are the pixel-wise Mean Squared Error (MSE) and the pixel-wise cross-entropy as in Lipton & Tripathi (2017) and Creswell & Bharath (2016). However in practice, pixel-wise losses are known to produce blurry images. To address this issue, other works have proposed alternative reconstruction errors. However, they are based on an alternative neural network (Boesen Lindbo Larsen et al., 2015; Johnson et al., 2016) making them computationally expensive.

The explanation usually given for the poor performance of pixel-wise mean square error is that it favors the solution which is the expected value of all the possibilities (Mathieu et al., 2015)[2]. We propose to go deeper into this explanation by studying the effect of the MSE on images in the frequency domain. In particular, our hypothesis is that due to its limited capacity and the low dimension of its latent space, the generator can not produce arbitrary texture patterns as the manifold of textures is very high dimensional. This uncertainty over texture configurations explains why textures are reconstructed as uniform regions when using pixel-wise errors. In Appendix A, by expressing the MSE in the Fourier domain and assuming that the phase of high frequencies cannot be encoded in the latent space, we show that the contribution of high frequencies in such a loss is proportional to their square magnitude pushing the optimisation to solutions with less high frequencies, that is to say more blurry. In order to get sharper results we therefore propose to reduce the weight of high frequencies into the penalisation of errors with the following loss:

$$\mathcal{L}(I_1, I_2) = ||\mathcal{F}\{I_1 - I_2\}\mathcal{F}\{\sigma\}||^2 = ||(I_1 - I_2) * \sigma||^2 \tag{3}$$

where $\mathcal{F}$ is the Fourier transform, $*$ is the convolution operator and σ is a Gaussian kernel. With a reduced importance given to the high frequencies to determine $\hat{\boldsymbol{z}}$ when one uses this loss in equation 2, it allows to benefit from a larger range of possibilities for $G(\boldsymbol{z})$, including images with more details (i.e with more high frequencies) and appropriate texture to get more realistic generated images. A qualitative comparison of some reconstruction errors and choices of σ can be found in Appendix C.

[1] We could have used a L^2 penalty on the norm of z to encode a centered Gaussian prior on the distribution of $\boldsymbol{z}$. However the L^2 penalty requires an additional hyper-parameter β that can be difficult to choose.

[2] Indeed, if we model the value of pixel by a random variable x then $\arg\min_x \mathbb{E}\left[(x - \mathrm{x})^2\right] = \mathbb{E}\left[\mathrm{x}\right]$. In fact, this problem can easily generalized at every pixel-wise loss if we assume that nearby pixels follows approximately the same distribution as $\arg\min_x \mathbb{E}\left[\mathcal{L}(x, \mathrm{x})\right]$ will have the same value for nearby pixels.

Algorithm 1: Create a dataset of trajectories in the latent space which corresponds to a transformation $\mathcal{T}$ in the pixel space. The transformation is parametrized by a parameter δt which controls a degree of transformation. We typically use $N = 10$ with $(\delta t_n)_{(0 \leq n \leq N)}$ distributed regularly on the interval $[0, T]$.

Input: number of trajectories S, generator G, transformation function $\mathcal{T}$, trajectories length N, threshold Θ.
Result: dataset of trajectories D
for $i \in [\![1, S]\!]$ **do**
  $D \leftarrow \{\}$;
  $\boldsymbol{z}_0 \sim \mathcal{N}(0, \boldsymbol{I})$;
  $I_0 \leftarrow G(\boldsymbol{z}_0)$;
  $\boldsymbol{z}_{\delta t} \leftarrow \boldsymbol{z}_0$;
  for $n \in [1, N]$ **do**
    $\boldsymbol{z}_t \leftarrow \arg\min_{\boldsymbol{z}} \mathcal{L}(G(\boldsymbol{z}), \mathcal{T}_{\delta t_n}(I_0))$;
    if $\mathcal{L}(G(\boldsymbol{z}), \mathcal{T}_{\delta t_n}(I_0)) < \Theta$ **then**
      $D \leftarrow D \cup \{(\boldsymbol{z}_0, \boldsymbol{z}_{\delta t}, \delta t_n)\}$;
    end
  end
end

2.1.2 Recursive Estimation of the Trajectory

Using equation 2, our problem of finding $\boldsymbol{z}_T$ such that $G(\boldsymbol{z}_T) \approx \mathcal{T}_T(I)$, given transformation $\mathcal{T}_T$, can be solve through the following optimisation problem:

$$\boldsymbol{z}_T = \underset{\boldsymbol{z} \in \mathcal{Z}, ||\boldsymbol{z}|| \leq \sqrt{d}}{\arg\min} \mathcal{L}(G(\boldsymbol{z}), \mathcal{T}_T(I)) \tag{4}$$

In practice, this problem is difficult and an "unlucky" initialization can lead to a very slow convergence. Zhu et al. (2016) proposed to use an auxiliary network to estimate $\boldsymbol{z}_T$ and use it as initialization. Training a specific network to initialize this problem is nevertheless costly. One can easily observe that a linear combination of natural images is usually not a natural image itself, this fact highlights the highly curved nature of the manifold of natural images in pixel space. In practice, the trajectories corresponding to most transforms in pixel space may imply small gradients of the loss that slowdown the convergence of problem of Eq. (2) (see Appendix E). To address this, we guide the optimization on the manifold by decomposing the transformation $\mathcal{T}_T$ into smaller transformations $[\mathcal{T}_{\delta t_0}, \ldots, \mathcal{T}_{\delta t_N}]$ such that $\mathcal{T}_{\delta t_0 = 0} = Id$ and $\delta t_N = T$ and solve sequentially:

$$\boldsymbol{z}_n = \underset{\boldsymbol{z} \in \mathcal{Z}}{\arg\min} \mathcal{L}\left(G\left(\boldsymbol{z}; \boldsymbol{z}_{init} = \boldsymbol{z}_{n-1}\right), \mathcal{T}_{\delta t_n}\left(G\left(\boldsymbol{z}_0\right)\right)\right) \quad \text{for } n = 1, \ldots, N \tag{5}$$

each time initializing $\boldsymbol{z}$ with the result of the previous optimization. In comparison to Zhu et al. (2016), our approach does not require extra training and can thus be used directly without training a new model. We compare qualitatively our method to a naive optimisation in Appendix C.

A transformation on an image usually leads to undefined regions in the new image (for instance, for a translation to the right, the left hand side is undefined). This is why we designed $\mathcal{L}$ to ignore the value of the undefined regions of the image. Another difficulty is that often the generative model cannot produce arbitrary images. For example a generative model trained on a given dataset is not expected to be able to produce images where the object shape position is outside of the distribution of object shape positions in the dataset. This is an issue when applying our method because as we generate images from a random start point, we have no guarantee that the transformed images is still on the data manifold. To reduce the impact of such outliers, we discard latent codes that give a reconstruction error above a threshold in the generated trajectories. In practice, we remove one tenth of the latent codes which leads to the worst reconstruction errors. It finally results into Algorithm 1 to generate trajectories in the latent space.

2.2 ENCODING MODEL OF THE FACTOR OF VARIATION IN THE LATENT SPACE.

After generating trajectories with Algorithm 1, we search regularities in these trajectories to find a direction $\boldsymbol{u} \in \mathcal{Z}$ which encodes a the parameter t of specific factor of variation in the latent space. Assuming a model $f : \mathcal{Z} \to \mathbb{R}$ that describes how factors of variations are encoded in the latent space, we pose $t = f(\boldsymbol{z}) = g(\langle \boldsymbol{z}, \boldsymbol{u} \rangle)$, with $g : \mathbb{R} \to \mathbb{R}$ and $\langle \cdot, \cdot \rangle$ the euclidean scalar product in $\mathbb{R}^d$, to formulate the hypothesis that a factor of variation is encoded in a direction $\boldsymbol{u}$ in the latent space.

For instance, if we consider the factor corresponding to the horizontal position of an object x in an image, and the dSprite dataset, we have x that follows a uniform distribution $\mathcal{U}(-0.5, 0.5)$ while the projection of $\boldsymbol{z}$ onto an axis $\boldsymbol{u}$ follows a normal distribution $\mathcal{N}(0, 1)$. Thus, it is natural to choose a model of the form:

$$x = f(\boldsymbol{z}) = g(\langle \boldsymbol{z}, \boldsymbol{u} \rangle) = \int_0^{\langle \boldsymbol{u}, \boldsymbol{z} \rangle} \mathcal{N}(t; 0, 1) dt = \frac{1}{2} \operatorname{erf}\left(\frac{\langle \boldsymbol{u}, \boldsymbol{z} \rangle}{\sqrt{2}} \right) \text{ with } ||\boldsymbol{u}|| = 1 \tag{6}$$

In general, the distribution of the parameter t of a factor of variation is not known. A simple approach is to suppose that t follows a centered Gaussian distribution (which results in a linear model as z is sampled from a centered Gaussian too). It is also possible to use a model which does not rely on such prior by learning g:

$$t = f_{(\theta, \boldsymbol{u})}(\boldsymbol{z}) = g_\theta \left(\langle \boldsymbol{u}, \boldsymbol{z} \rangle \right) \text{ with } ||\boldsymbol{u}|| = 1 \tag{7}$$

with $g_\theta : \mathbb{R} \to \mathbb{R}$ and (θ, $\boldsymbol{u}$) trainable parameters of the model. We typically used piece-wise linear functions for g_θ. However, this model cannot be trained directly as we do not have access to t (in the case of horizontal translation the x-coordinate for example) but only to the difference $\delta t = t_{G(\boldsymbol{z}_1)} - t_{G(\boldsymbol{z}_0)}$ between an image $G(\boldsymbol{z}_0)$ and its transformation $G(\boldsymbol{z}_1)$ (δx or δy in the case of translation). We solve this issue by reformulating the problem as:

$$\delta t = f_{(\theta, \boldsymbol{u})}(\boldsymbol{z}_1) - f_{(\theta, \boldsymbol{u})}(\boldsymbol{z}_0) \text{ with } ||\boldsymbol{u}|| = 1 \text{ and } g_\theta(0) = 0 \tag{8}$$

Hence, $\boldsymbol{u}$ and θ are estimated by training $f_{(\theta, \boldsymbol{u})}$ with gradient descent on a dataset produced by Algorithm 1 for a given transformation.

An interesting application of this method is the estimation of the distribution of the images generated by G. Indeed, when $\boldsymbol{z}$ is normally distributed, then $\langle \boldsymbol{u}, \boldsymbol{z} \rangle \sim \mathcal{N}(0, 1)$. With our model, $t = g_\theta(\langle \boldsymbol{u}, \boldsymbol{z} \rangle)$ thus, when g_θ is an monotonic differentiable function, the distribution $\varphi : \mathbb{R} \to \mathbb{R}_+$ of t can be estimated by:

$$\varphi(t) = \mathcal{N}(g_\theta^{-1}(t); 0, 1) \left| \frac{d}{dt} g_\theta^{-1}(t) \right| \tag{9}$$

With the knowledge of g_θ we can also choose how to sample images. For instance, let say that we want to have $t \sim \phi(t)$, with $\phi : \mathbb{R} \to \mathbb{R}_+$ an arbitrary distribution, we can simply transform $z \sim \mathcal{N}(0, 1)$ as follows:

$$\boldsymbol{z} \leftarrow \boldsymbol{z} - \langle \boldsymbol{z}, \boldsymbol{u} \rangle \, \boldsymbol{u} + (h_\phi \circ \psi)(\langle \boldsymbol{z}, \boldsymbol{u} \rangle) \boldsymbol{u} \tag{10}$$

with $h_\phi : [0, 1] \to \mathbb{R}$ and ψ such that:

$$\psi(x) = \int_{-\infty}^{x} \mathcal{N}(t; 0, 1) dt \ ; \ h_\phi^{-1}(x) = \int_{-\infty}^{x} \phi(g_\theta(t)) \left| \frac{d}{dt} g_\theta(t) \right| dt \tag{11}$$

These results are interesting to bring control not only on a single output of a generative model but also on the distribution of its outputs. Moreover, since generative models reflect the datasets on which they have been trained, the knowledge of these distributions could be applied to the training dataset to reveal potential bias.

3 EXPERIMENTS

Datasets: We performed experiments on two datasets. The first one is dSprites (Matthey et al., 2017), composed of 737280 binary 64×64 images containing a white shape on a dark background. Shapes can vary in position, scale and orientations making it ideal to study disentanglement. The second

dataset is ILSVRC (Russakovsky et al., 2015), containing $1.2M$ natural images from one thousand different categories.

Implementation details: All our experiments have been implemented with TensorFlow 2.0 (Abadi et al., 2015) and the corresponding code is available at `https://anonymised.for.review`. We used a BigGAN model (Brock et al., 2018) whose weights are taken from TensorFlow-Hub allowing easy reproduction of our results. The BigGAN model takes two vectors as inputs: a latent vector $\boldsymbol{z} \in \mathbb{R}^{128}$ and a one-hot vector to condition the model to generate images from one category. The latent vector $\boldsymbol{z}$ is then splited into six parts which are the inputs at different scale levels in the generator. The first part is injected at the bottom layer while next parts are used to modify the *style* of the generated image thanks to AdaIN layers (Huang & Belongie, 2017). We also trained several β-VAEs (Higgins et al., 2017) to study the importance of disentanglement in the process of controlling generation. The exact β-VAE architecture used is given in Appendix B. The models were trained on dSprites (Matthey et al., 2017) with an Adam optimizer during $1\mathrm{e}5$ steps with a batch size of 128 images and a learning rate of $5\mathrm{e}{-4}$.

3.1 Quantitative evaluation method

Evaluating quantitatively the effectiveness of our method on complex datasets is intrinsically difficult as it is not always trivial to measure a factor of variation directly. We focused our analysis on two factors of variations: position and scale. On simple datasets such as dSprites, the position of the object can be estimated effectively by computing the barycenter of white pixels. However, for natural images sampled with the BigGAN model, we have to use first saliency detection on the generated image to produce a binary image from which we can extract the barycenter. For saliency detection, we used the model provided by Hou et al. (2016) which is implemented in the PyTorch framework (Paszke et al., 2017). The scale is evaluated by the proportion of salient pixels. The evaluation procedure is:

1. Get the direction $\boldsymbol{u}$ which should describe the chosen factor of variation with our method.
2. Sample latent codes $\boldsymbol{z}$ from a standard normal distribution.
3. Generate images with latent code $\boldsymbol{z} - \langle \boldsymbol{z}, \boldsymbol{u} \rangle \, \boldsymbol{u} + t\boldsymbol{u}$ with $t \in [-T, T]$.
4. Estimate the real value of the factor of variation for all the generated images.
5. Measure the standard deviation of this value with respect to t.

Jahanian et al. (2019) proposed an alternative method for quantitative evaluation that relies on an object detector. Similarly to us, it allows an evaluation for x and y shift as well as scale but is restricted to image categories that can be recognized by a detector trained on some categories of ILSVRC. We thus gain in genericity with our approach.

3.2 Results on BigGAN

We performed quantitative analysis on ten chosen categories for which the object can be easily segmented by using saliency detection approach. Results are presented in Figure 2 (top). We observe that for the chosen categories of ILSVRC, we can control the position and scale of the object relatively precisely by moving along directions of the latent space found by our method.

However, one can still wonder whether the directions found are independent of the category of interest. To answer this question, we merged all the datasets of trajectories into one and learned a common direction on the resulting datasets. Results for the ten test categories are shown in Figure 2 (bottom). This figure shows that the directions which correspond to some factors of variations are indeed shared between all the categories. Qualitative results are also presented in Figure 3 for illustrative purposes. We also checked which parts of the latent code are used to encode position and scale. Indeed, BigGAN uses hierarchical latent code which means that the latent code is split into six parts which are injected at different level of the generator. We wanted to see by which part of the latent code these directions are encoded. The squared norm of each part of the latent code is reported in Figure 4 for horizontal position, vertical position and scale. This figure shows that the directions corresponding to *spatial factors of variations* are mainly encoded in the first part of the latent code. However, for the y position, the contribution of level 5 is higher than for the x position and the scale.

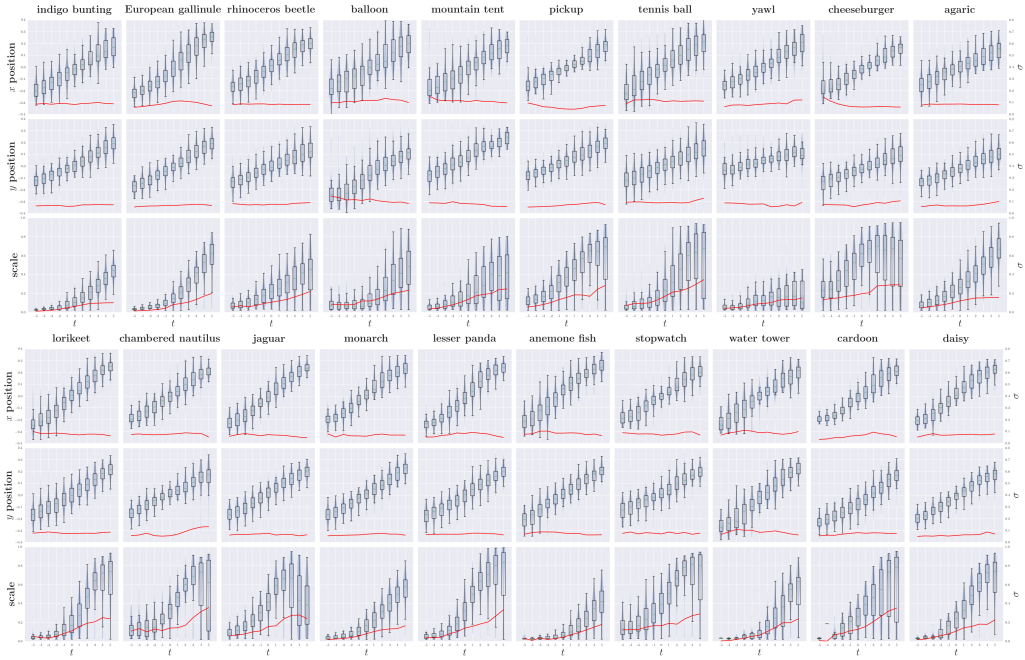

Figure 2: Quantitative results on the ten categories of the ILSVRC dataset used for training (Top) and for ten other categories used for validation (Bottom) for three geometric transformations: horizontal and vertical translations and scaling. In blue, the distribution of the measured transformation parameter and in red the standard deviation of the distribution with respect to t. Note that for large scales the algorithm seems to fail. However, this phenomenon is very likely due to the poor performance of the saliency model when the object of interest covers almost the entire image (scale ≈ 1.0). (best seen with zoom)

We suspect that it is due to correlations between the vertical position of the object in the image and its background that we introduced by transforming the objects because the background is not invariant by vertical translation because of the horizon.

3.3 The importance of disentangled representations

To test the effect of disentanglement on the performance of our method, we trained several β-VAE (Higgins et al., 2017) on dSprites (Matthey et al., 2017), with different β values. Indeed, β-VAE are known for having more disentangled latent spaces as the regularization parameter β increases. Results can be seen in Figure 5. The figure shows that it is possible to control the position of the object on the image by moving in the latent space along the direction found with our method. As expected, the effectiveness of the method depends on the degree of disentanglement of the latent space since the results are better with a larger β. Indeed we can see on Figure 5 that as β increases, the standard deviation decreases (red curve), allowing a more precise control of the position of the generated images. This observation motivates further the interest of disentangled representations for control on the generative process.

4 Related works

Our work aims at finding interpretable directions in the latent space of generative models to control their generative process. We distinguish two families of generative models: GAN-like models which do not provide an explicit way to get the latent representation of an image and auto-encoders which provide an encoder to get the latent representation of images. From an architectural point of view, conditional GANs (Odena et al., 2016) and InfoGan (Chen et al., 2016) allow the user to choose the category of a generated object or some chosen properties of the generated image but these approaches require a labeled dataset and use models which are explicitly designed to allow control. Our method

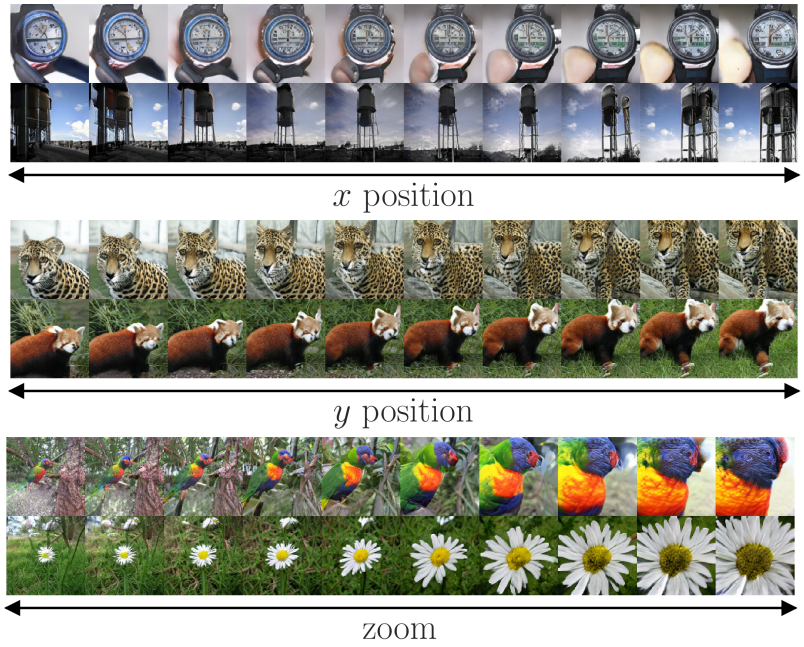

Figure 3: Qualitative results for some categories of ILSVRC dataset for three geometric transformations: horizontal and vertical translations and scaling.

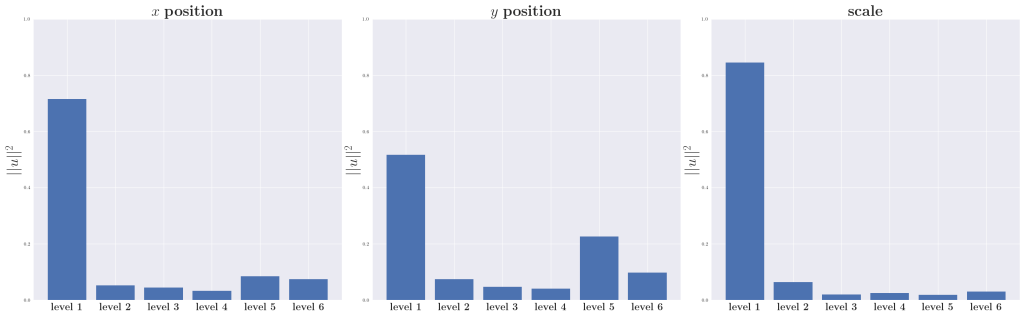

Figure 4: Squared norm of each part of the latent code for horizontal position, vertical position and scale.

has the advantage of being model agnostic and of not directly requiring labels. A more recent attempt at controlling generative models is described in Bau et al. (2018) in which the authors analyze the activations of the network's neurons to know which ones are causing the presence of specific objects in the generated image. With their method, they managed to control the presence of objects in the generated image. On the other side, our work focuses on the latent space and not on the intermediate activations inside the generator.

One of our contribution and a part of our global method is a procedure to find the latent representation of an image when an encoder is not available. Several previous works have studied how to invert the generator of a GAN to find the latent code of an image. Creswell & Bharath (2016) showed on simple datasets (MNIST (Lecun et al., 1998) and Omniglot (Lake et al., 2015)) that this inversion process can be achieved by optimising the latent code to minimize the reconstruction error between the generated image and the target image. Lipton & Tripathi (2017) introduced tricks to improve the results on a more challenging dataset (CelebA (Liu et al., 2015)). However we observed that these methods fail when applied on a more complex datasets (ILSVRC (Russakovsky et al., 2015)). In this paper we introduce a new reconstruction loss adapted to this particular problem which improves the

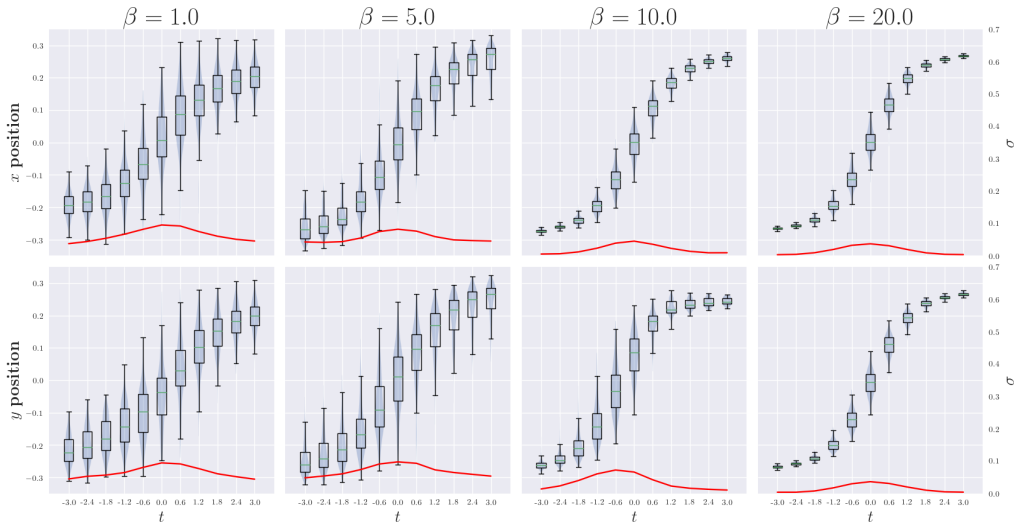

Figure 5: Results of our evaluation procedure with four β-VAE for $\beta = 1, 5, 10, 20$. Note the erf shape of the results which indicates that the distribution of the shape positions has been correctly learned by the VAE. See Figure 2 for additional information on how to read this figure.

quality of reconstructions significantly. We also give a hint about why the problem of inverting a generative model can be difficult compared to other optimization problems.

The closest works were released on ArXiv very recently (Goetschalckx et al., 2019; Jahanian et al., 2019) indicating that finding interpretable directions in the latent space of generative models to control their output is of high interest for the community. In these papers, the authors describe a method to find interpretable directions in the latent space of the BigGAN model (Brock et al., 2018). If their method exhibits similarities with ours (use of transformation, linear trajectories in the latent space), it also differs on several points. From a technical point of view our training procedure differs in the sense that we first generate a dataset of interesting trajectories to then train our model while they train their model directly. Our evaluation procedure is also more general as we use a saliency model instead of a MobileNet-SSD v1 Liu et al. (2016) trained on specific categories of the ILSVRC dataset allowing us to measure performance on more categories. We provide additional insight on how auto-encoders can also be controlled with the method, the impact of disentangled representations on the control and on the structure of the latent space of BigGAN. Moreover we also propose an alternative reconstruction error to invert generators. However, the main difference we identify between the two works is the model of the latent space used. Our model allows a more precise control over the generative process and can be being adapted to more cases.

5 Conclusions

Generative models are increasingly more powerful but suffer from little control over the generative process and the lack of interpretability in their latent representations. In this context, we propose a method to extract meaningful directions in the latent space of such models and use them to control precisely some properties of the generated images. We show that a linear subspace of the latent space of BigGAN can be interpreted in term of intuitive factors of variation (namely translation and scale). It is an important step toward the understanding of the representations learned by generative models.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1811.10597, 11 2018.

A. C. Berg, T. L. Berg, H. Daumé, J. Dodge, A. Goyal, X. Han, A. Mensch, M. Mitchell, A. Sood, K. Stratos, and K. Yamaguchi. Understanding and predicting importance in images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3562–3569, June 2012.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv e-prints*, art. arXiv:1512.09300, Dec 2015.

Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1810.10863, Oct 2018.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018. URL `http://arxiv.org/abs/1809.11096`.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *arXiv e-prints*, art. arXiv:1606.03657, 06 2016.

Antonia Creswell and Anil Anthony Bharath. Inverting The Generator Of A Generative Adversarial Network. *arXiv e-prints*, art. arXiv:1611.05644, 11 2016.

Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. GANalyze: Toward Visual Definitions of Cognitive Image Properties. *arXiv e-prints*, art. arXiv:1906.10112, Jun 2019.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representation*, 2017.

Qibin Hou, Ming-Ming Cheng, Xiao-Wei Hu, Ali Borji, Zhuowen Tu, and Philip Torr. Deeply supervised salient object detection with short connections. *arXiv e-prints*, art. arXiv:1611.04849, 11 2016.

Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *arXiv e-prints*, art. arXiv:1703.06868, Mar 2017.

Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks. *arXiv e-prints*, art. arXiv:1907.07171, Jul 2019.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *arXiv e-prints*, art. arXiv:1603.08155, Mar 2016.

Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1812.04948, 12 2018.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *arXiv preprint arXiv:1602.07332, 2016.*, 2016.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL `https://science.sciencemag.org/content/350/6266/1332`.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Zachary C. Lipton and Subarna Tripathi. Precise Recovery of Latent Vectors from Generative Adversarial Networks. In *ICLR workshop*, 2017.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 12 2015.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv e-prints*, art. arXiv:1511.05440, Nov 2015.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv e-prints*, art. arXiv:1610.09585, 10 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1511.06434, 11 2015.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. *arXiv e-prints*, art. arXiv:1906.00446, Jun 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. doi: 10.1109/TIP.2003.819861.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative Visual Manipulation on the Natural Image Manifold. *arXiv e-prints*, art. arXiv:1609.03552, 09 2016.

A PENALTY ON THE AMPLITUDE OF FREQUENCIES DUE TO MSE

In Section 2.1, we consider a target image $I \in \mathcal{I}$ and a generated image $\hat{I} = G(\hat{z})$ to be determined according to a reconstruction loss $\mathcal{L}$ (Equation 1). Let us note $\mathcal{F}\{\cdot\}$ the Fourier transform. If $\mathcal{L}$ is the usual MSE, from the Plancherel theorem, we have $||\hat{I} - I||^2 = ||\mathcal{F}\{\hat{I}\} - \mathcal{F}\{I\}||^2$. Let us consider a particular frequency ω in the Fourier space and compute its contribution to the loss. The Fourier transform of I (resp. $\hat{I}$) having a magnitude r (resp. $\hat{r}$) and a phase θ (resp. $\hat{\theta}$) at ω, we have:

$$\begin{aligned} |\mathcal{F}\{\hat{I}\}(\omega) - \mathcal{F}\{I\}(\omega)|^2 &= |\hat{r}e^{i\hat{\theta}} - re^{i\theta}|^2 \\ &= (\hat{r}cos(\hat{\theta}) - rcos(\theta))^2 + (\hat{r}sin(\hat{\theta}) - rsin(\theta))^2 \\ &= \hat{r}^2 + r^2 - 2\hat{r}r\left(cos(\hat{\theta})cos(\theta) + sin(\hat{\theta})sin(\theta)\right) \\ &= \hat{r}^2 + r^2 - 2\hat{r}r\left(cos(\hat{\theta})cos(\theta) + sin(\hat{\theta})sin(\theta)\right) \end{aligned} \tag{12}$$

If we model the disability of the generator to model every high frequency patterns as an uncertainty on the phase of high frequency of the generated image, i.e by posing $\hat{\theta} \sim \mathcal{U}([0, 2\pi])$, the expected value of the high frequency contributions to the loss is equal to:

$$\begin{aligned} \mathbb{E}\left[|\mathcal{F}\{\hat{I}\}(\omega) - \mathcal{F}\{I\}(\omega)|^2\right] &= \hat{r}^2 + r^2 - 2\hat{r}r\left(\underbrace{\mathbb{E}\left[cos(\hat{\theta})\right]}_{=0} cos(\theta) + \underbrace{\mathbb{E}\left[sin(\hat{\theta})\right]}_{=0} sin(\theta)\right) \\ &= \hat{r}^2 + r^2 \end{aligned} \tag{13}$$

The term r^2 is a constant w.r.t the optimization of $\mathcal{L}$ and can thus be ignored. The contribution to the total loss $\mathcal{L}$ thus directly depends on $\hat{r}^2$. While minimizing $\mathcal{L}$, the optimization process tends to favour images $\hat{I} = G(\hat{z})$ with smaller magnitudes in the high frequencies, that is to say smoother images, with less high frequencies.

B β-VAE ARCHITECTURE

The β-VAE framework was introduced by Higgins et al. (2017) to discover interpretable factorised latent representations for images without supervision. For our experiments, we designed a simple convolutional VAE architecture to generate images of size 64x64, the decoder network is the opposite of the encoder with transposed convolutions.

Encoder
Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Dense + ReLU units=256
Dense + ReLU units=256
μ: **Dense + Identity** σ: **Dense + Exponential** units=10

Decoder
Dense + ReLU units=256
Dense + ReLU units=256
Reshape shape=4x4x32
Transposed Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Transposed Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Transposed Convolution + ReLU filters=32 size=4 stride=2 pad=SAME
Transposed Convolution + Sigmoid filters=1 size=4 stride=2 pad=SAME

Table 1: β-VAE architecture used during experiments with the dSprites dataset.

C QUALITATIVE EXPERIMENTS WITH OUR RECONSTRUCTION ERROR

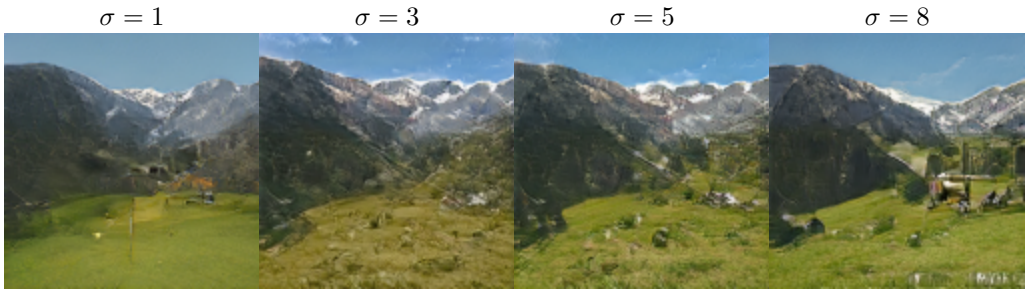

Figure 6: Reconstruction results with different σ values. We typically used a standard deviation of 3 pixels for the kernel.

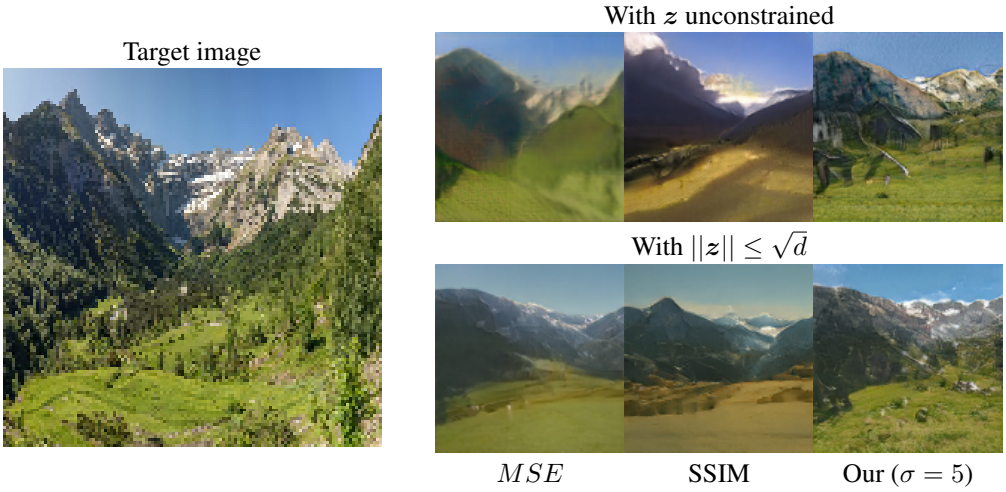

Figure 7: Reconstruction results obtained with different reconstruction errors: MSE, SSIM (Zhou Wang et al., 2004) and our loss. With or without the constraint on $||\boldsymbol{z}||$. Note the artifacts when using our loss without constraining $\boldsymbol{z}$ (best seen with zoom).

D QUALITATIVE COMPARISON BETWEEN OUR OPTIMIZATION METHOD AND THE NAIVE METHOD.

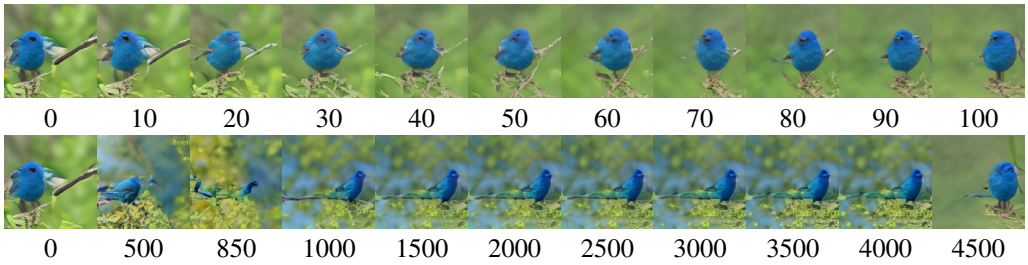

Figure 8: Comparison of the speed of convergence on a single example for our method (top) given by equation 5 and a naive approach (bottom) given by equation 4. The numbers indicate the step of optimization. Both experiences have been conducted with Adam optimizer with a learning rate of $1\text{e}{-1}$.

E ON THE DIFFICULTY OF OPTIMISATION ON THE NATURAL IMAGE MANIFOLD.

The curvature of the natural image manifold makes the optimisation problem of Equation 2 difficult to solve. This is especially true for factors of variation which correspond to curved walks in pixel-space (for example translation or rotation by opposition to brightness or contrast changes which are linear).

To illustrate this fact, we show that the trajectory described by an image undergoing common transformations is curved in pixel space. We consider three types of transformations, namely translation, rotation and scaling, and get images from the dSprites (Matthey et al., 2017) dataset which correspond to the progressive transformation (interpolation) of an image. To visualize, we compute the PCA of the resulting trajectories and plot the trajectories on the two main axes of the PCA. The result of this experiment can be seen in Figure 9.

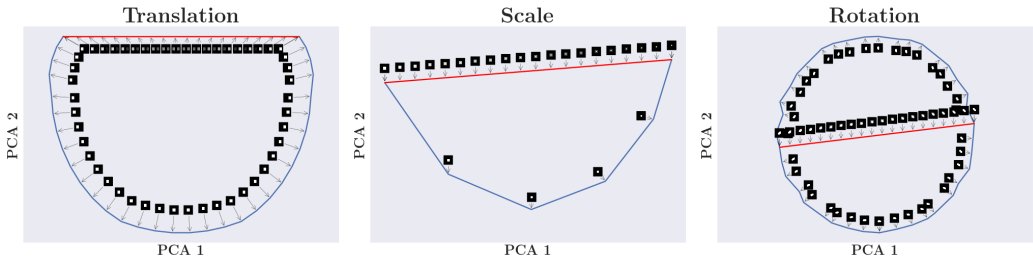

Figure 9: Two trajectories are shown in the pixel space, between an image and its transformed version, for three types of transformations: translation, scale and orientation. Red: shortest path (interpolation) between the two extremes of the trajectory. Blue: trajectory of the actual transformation. At each position along the trajectories, we report the corresponding image (best seen with zoom).

In this figure, we can see that for large translations, the direction of the shortest path between two images in pixel-space is near orthogonal to the manifold. The same problem occurs for rotation and, at a smaller extent, for scale. However this problem does not exist for brightness for example, as its change is a linear transformation in pixel-space. This is problematic during optimization of the latent code because the gradient of the reconstruction loss with respect to the generated image is tangent to this direction. Thus, when we are in the case of near orthogonality, the gradient of the error with respect to the latent code is small.

Indeed, let us consider an ideal case where G is a bijection between $\mathcal{Z}$ and the manifold of natural images. Let be $\boldsymbol{z} \in \mathcal{Z}$, a basis of vectors tangent to the manifold at point $G(\boldsymbol{z})$ is given by $\left(\frac{\partial G(\boldsymbol{z})}{\partial \boldsymbol{z}_1}, ..., \frac{\partial G(\boldsymbol{z})}{\partial \boldsymbol{z}_d}\right)$.

If $\nabla_{G(\boldsymbol{z})}\mathcal{L}(G(\boldsymbol{z}), I_{target})$ is near orthogonal to the manifold then:

$$\forall i \in 1, ..., d : \langle \nabla_{G(\boldsymbol{z})}\mathcal{L}(G(\boldsymbol{z}), I_{target}), \frac{\partial G(\boldsymbol{z})}{\partial \boldsymbol{z}_i} \rangle = \epsilon_i \text{ with } \epsilon_i \approx 0 \tag{14}$$

Thus,

$$\|\nabla_z \mathcal{L}(G(\boldsymbol{z}), I_{target})\| = \left\| \frac{\partial G(\boldsymbol{z})}{\partial \boldsymbol{z}}^* \nabla_{G(\boldsymbol{z})}\mathcal{L}(G(\boldsymbol{z}), I_{target}) \right\| = \sqrt{\sum_{i=1}^{d} \epsilon_i^2} \approx 0 \tag{15}$$

It shows that when the direction of descent in pixel space is near orthogonal to the manifold described by the generative model, optimization gets slowed down and can stop if the gradient of the loss with respect to the generated image is orthogonal to the manifold.

For example, let assume we have an ideal GAN which generates a small white circle on a black background, with a latent space of dimension 2 that encodes the position of the circle. Let consider a generated image with the circle on the left of the image and we want to move it to the right. Obviously, we thus have $\nabla_{\boldsymbol{z}}||G(\boldsymbol{z}) - \mathcal{T}_T(G(\boldsymbol{z}_1))||^2 = 0$ if the intersection of the two circles is empty (see Figure 9) since a small translation of the object does not change the reconstruction error.

F ADDITIONAL QUALITATIVE EXAMPLES

We show qualitative examples for images generated with the BigGAN model for position, scale and brightness. The images latent codes are sampled in the following way: $\boldsymbol{z} - \langle \boldsymbol{z}, \boldsymbol{u} \rangle \boldsymbol{u} + \alpha \boldsymbol{u}$ with $\alpha \in [-3, 3]$ and $\boldsymbol{u}$ the learned direction. We have chosen the categories to produce interesting results: for position and scale categories are objects, for brightness categories are likely to be seen in a bright or dark environment. Notice that for some of the chosen categories, we failed to control the brightness of the image. It is likely due to the absence of dark images for these categories in the training data. for position and scale, the direction is learned on the ten categories presented here while for brightness only the five top categories are used.

Figure 10: Qualitative results for 10 categories of ILSVRC dataset for three geometric transformations (horizontal and vertical translations and scaling) and for brightness.