Inspirational adversarial image generation

Exploring the latent space of GANs

Morgane Riviere · Olivier Teytaud · Jérémy Rapin · Yann LeCun · Camille Couprie

Abstract The task of image generation started to receive some attention from artists and designers to inspire them in new creations. However, exploiting the results of deep generative models such as Generative Adversarial Networks can be long and tedious given the lack of existing tools. In this work, we propose a simple strategy to inspire creators with new generations learned from a dataset of their choice, while providing some control on them. We design a simple optimization method to find the optimal latent parameters corresponding to the closest generation to any input inspirational image. Specifically, we allow the generation given an inspirational image of the user choice by performing several optimization steps to recover optimal parameters from the model's latent space. We tested several exploration methods starting with classic gradient descents to gradient-free optimizers. Many gradient-free optimizers just need comparisons (better/worse than another image), so that they can even be used without numerical criterion, without inspirational image, but with only with human preference. Thus, by iterating on one's preferences we could make robust Facial Composite or Fashion Generation algorithms. High resolution of the produced design generations are obtained using progressive growing of GANs. Our results on four datasets of faces, fashion images, and textures show that satisfactory images are effectively retrieved in most cases.

Keywords Optimization · Generative adversarial networks · Similarity search

1 Introduction

Generative models, and in particular adversarial ones [16], are becoming prevalent in computer vision as they enable enhancing artistic creation [8,45], inspire designers [38,46], or prove usefulness in semi-supervised learning [6,9,32].

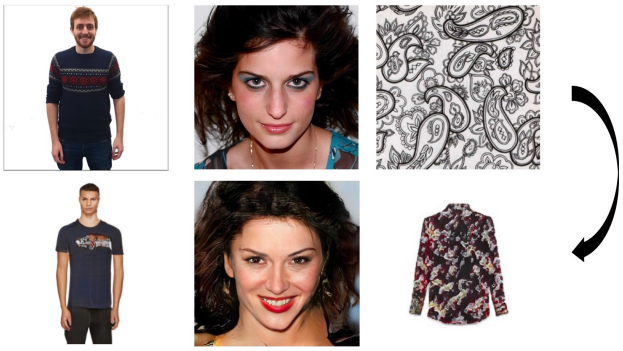


Fig. 1 Generative networks may be a source of inspiration for designers and artists, but the current approaches lack control on generations. Our approach allows to unearth generations (bottom line) from trained networks and either (i) an inspirational image (top line) or (ii) preferences among presented images provided by a user.

GAN models take a random vector as input and output an image. Due to the high dimensionality of their latent space, we know little about what kind of picture we will get. To reduce the burden of creative people compelled to brute-force GANs generations and cherry pick nice examples, we propose to generate at test time *image*-inspired generations or *preferences*-based generation:

- Given an inspirational image and a trained model, we recover via a gradient descent the input vector that minimizes a reconstruction criterion defined between the given image and the generation.
- Or, iteratively, based on human preferences at each generation, we converge to the best choice of a user.

Despite a highly non-convex optimization, we show that it is possible to obtain accurately related generations. In addition to the gradient descent, we tested several gradient-free optimizers in this procedure.

M. Riviere, O. Teytaud, J. Rapin, Y. LeCun, C. Couprie
Facebook AI Research
E-mail: mriviere@fb.com, oteytaud@fb.com, jrapin@fb.com, yann@fb.com, coupriec@fb.com

Our contributions are the following:

- We study the problem of GAN latent space exploration, define and validate a criterion of generation retrieval given a target image, and suggest a method based on user preferences for retrieval of GAN generations without any target image.
- We identify the best exploration methods to solve these problems.
- We present different applications for clothes, textures retrieval, and facial composites.

It is worth noticing that our procedure is done at test time and does not require any additional update of the model's weights.

Being able to control the output of a GAN is a first step toward numerous applications of this framework. Use cases may include: (i) Designers who want to invent a new clothes similar to successful items of past collections. (ii) Designers having a piece of textile and wanting to see a potential rendering in a clothe inspired by it. (iii) An artist looking for inspiration, given a picture. (iv) An artist looking for inspiration, iteratively choosing the μ best out of several proposals. (v) Facial composites, with a person iteratively choosing the μ most similar out of several faces. (vi) Artificial texture generation could be an interesting technology for 3D engines. (vii) Fake items generations could be used for in-painting features in image and photo-edition softwares. (viii) Inspired faces could serve as avatar or for anonymization purposes.

The control provided by our approach is both simple, intuitive, and computationally efficient. After introducing the related work in Section 2, we present our approach to retrieve inspired generations from trained GANs in Section 3, and experiments that include the comparison of various similarity criteria, several gradient based and gradient free optimization approaches on diverse datasets in Section 4.

2 Related work

In this section, after motivating our choice of GAN setting, we discuss works exploring the latent space of GANs.

State-of-the-art GANs. There have been large progress towards the image quality and the stabilization of adversarial training lately. The Wasserstein training objective was introduced along with gradient clipping to prevent too strong gradients [1]. Then the clipping was replaced by a gradient penalty loss [17]. Progressive growing of GAN architectures [21] employs this loss, leading to stable models that reach high resolution in a reasonable training time. Other approaches like [28] also reach high resolution but in our experience with a slower convergence. Class Conditional approaches [33] based on attention mechanism such as SAGAN [43] and spectral normalization [30], or BigGAN [3] display remarkable results but have not been assessed yet in high resolution image generation.

Conditional generation with GANs. Odena & al. [33] proposed a simple method to enforce label conditioning at training time on GANs: labels are encoded as a one-hot vector concatenated to the continuous generator's latent space, the generator is trained to generate data coherent with their given labels and the discriminator is trained to recognize them.

Latent space exploration of GANs. The exploration of the latent space of GANs was popularized by the DCGAN work presenting latent space interpolations and arithmetic operation results [34]. Learning a mapping projecting data back in the latent space of GANs has been studied in the context of bi-directional GANs [6], with an emphasis on a utility in semi-supervised learning. Similarly, image generation may improve zero shot learning tasks [47]. In Fader Networks [25], image manipulation is made possible by learning an image representation disentangled from its attributes with adversarial training. Recently, the Style-based generator architecture of Karras et al. [22] improved the coherency of the generator's internal representation by creating an intermediate latent space and enforcing feature statistics proximity between neighbor codes. The criterion of feature similarity is borrowed from Texture networks [41].

A number of works focus on neural visualization of trained networks for image classification [42,7,31]. A somehow related task lies in membership inference, where the goal is to determine if an image has been seen for training [37,24]. The notion of feature similarity in image generation is widely employed, for instance in style transfer [14], and generation quality assessment [44]. The most similar work in spirit to our image inspiration strategy is the Inference-via-optimization approach defined to evaluate the severity of mode collapse in Unrolled GANs [29]. We can also cite the approach of [2] that also matches vectors in the latent space of GANs with precise pictures at training time using a Nesterov gradient.

In contrast to these works, we design an appropriate similarity metric based on semantic features that also exploits the discriminator. Moreover, we benchmark various search strategies to obtain better generations including gradient-free approaches which do not always need a target image.

3 Approach

3.1 Background on progressive growing of GANs

GAN models are built as follows: given two networks, a generator G and a discriminator D, the discriminator is trained to differentiate real data from fake ones, while G is rewarded if it fools D. In this work, we have chosen to use the Progressive Growing architecture [21]. In order to reach generation in high resolutions, the progressive growing methods trains both networks partially, one resolution at a time. In other words, it starts with two small networks trained on 4×4 images; once the results are satisfying, the layers corresponding to the 8×8 resolution are added and the training continues. This method goes on until the desired resolution is reached. For example, in [21], Karras et al. could build realistic 1024×1024 images.

The progressive growing method happens to be very stable: even in high resolution we did not notice any mode collapse in any of our training sessions. Besides, we show that it gives convincing results even with small datasets (≈ 4000 images). Our implementation of Progressive Growing and the inspiration method described in this paper is open-sourced[1].

3.2 Multi-class conditioning

For models trained on labelled datasets, we apply a variation of AC-GAN [33]. If a class c has k_c possible values, then it can be encoded in a one-hot vector $(c_1, \ldots, c_{k_c})$ with $c_i = 1$ codes for label i. When $C > 1$ groups of classes exist, we concatenate the encoding vectors of all classes to get a label vector $\hat{c}$. This label vector is then fed to the generator with the latent input noise z.

The classification loss becomes:

$$\mathcal{L}_{\text{class}}(\hat{c}, x) = -\sum_{c}^{} \sum_{i=1}^{k_c} \log \left(\frac{e^{D_{c_i}(x)}}{\sum_{q=1}^{k_c} e^{D_{c_q}(x)}} \right), \tag{1}$$

The discriminator D must minimize:

$$\mathcal{L}_D = \mathcal{L}_{PGAN(D)} + \lambda_D^c \mathcal{L}_{\text{class}}(\hat{c}_{real}, x_{real})$$

While the generator G must minimize:

$$\mathcal{L}_D = \mathcal{L}_{PGAN(G)} + \lambda_G^c \mathcal{L}_{\text{class}}(\hat{c}_{noise}, z)$$

With $\mathcal{L}_{PGAN(D)}$ and $\mathcal{L}_{PGAN(G)}$ the loss penalties of Progressive Growing. $\hat{c}_{real}$ is the label vector of the input image x_{real} and $\hat{c}_{noise}$ the random label vector fed to G along with z.

[1] `https://github.com/facebookresearch/pytorch_GAN_zoo`

3.3 Image-inspired image generation

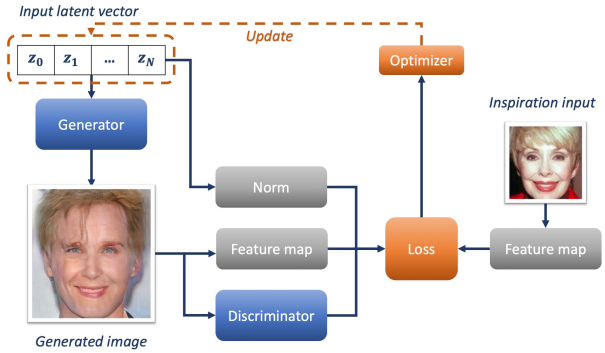


Fig. 2 Our inspiration based approach. On a trained GAN, we perform a search on the latent vector fed to the generator in order to find the closest generation to a given reference image.

The different directions of the latent space of a GAN model usually do not make sense from a human point of view. We wanted to know if it was possible to explore this space to perform what we call an "inspirational generation". In other words, given a reference image I we would like to force the generator to produce an output as similar as possible to I.

To do so, we considered performing a gradient descent on the noise vector z fed to the generator. This idea raises two issues:

- How do we define an objective criterion for image similarity ?
- Can we converge to an optimal vector ?

3.3.1 Similarity

We first need a similarity measure between images. Therefore, we tried two different approaches.

Pixel-based similarity The simplest way to estimate the similarity between two pictures is to perform their difference pixel-wise. This give us a similarity criterion $\mathcal{C}_L$:

$$\mathcal{C}_L(z, I) = \frac{1}{N_p} \sum_{p \in pixels} \|G(z)_p - I_p\|^2 \tag{2}$$

This loss however is not invariant by translation, rotation or illumination change. Also, we do not expect to capture abstract concept (gender, posture etc...) with it.

Feature-based similarity We found that a variation of the style criterion used in [19] performs well for nearest neighbors search in several considered datasets. This criterion $\mathcal{C}_S$ is defined as:

$$\mathcal{C}_S(z, I) = \sum_i \|\mu(\phi_i(G(z))) - \mu(\phi_i(R))\|^2 + \sum_i \|\sigma(\phi_i(G(z)))^2 - \sigma(\phi_i(R))^2\| \quad (3)$$

Where:

- ϕ_i represents the ith layer of a fully convolutional neural network trained on image net.
- μ and σ are the mean and standard deviation computed across the image

In our case, we considered a VGG-19 architecture. The references of the chosen layers are written in the appendix. Relying on statistics rather than on a spatial criterion makes the score invariant by translation of different components of the reference image. We used both low level or relatively high-level features of VGG-19. As shown in [42], low level layers catch very simple patterns like edges, specific shades of color, transitions, etc. The higher a layer is in the network, the more abstract, large and complex the pattern it reveals is. Some of these patterns are very specific, leading to some neurons mostly activated by faces for example.

3.3.2 Latent vector penalty

The Progressive growing model normalizes the latent vector before feeding it to the generator. That is why, without any further penalty on the latent vector, our problem is not properly defined and thus our algorithms cannot possibly converge. To correct this issue, we add a criterion constraining the norm of the latent vector to be one:

$$\mathcal{C}_\nu = (\frac{1}{N}\|z\|^2 - 1)^2, \quad (4)$$

3.3.3 Realism penalty

Finally, if $\mathcal{C}_S$ forces the generated image to show some statistical similarity with the reference, nothing compels the generator to produce a realistic output. That is why we complete our score with a realism criterion $\mathcal{C}_R$, that exploits the trained discriminator as well:

$$\mathcal{C}_R = -D(G(z)). \quad (5)$$

Fig. 3 Random samples sorted by increasing value of D. We observe that most images containing artifacts have a small D, highlighting the importance of the discriminator part in our criterion.

3.3.4 Final penalty

Our criterion becomes:

$$\mathcal{C} = \lambda_S\mathcal{C}_S + \lambda_\nu\mathcal{C}_\nu + \lambda_R\mathcal{C}_R + \lambda_L\mathcal{C}_L \quad (6)$$

where $\lambda_S, \lambda_\nu, \lambda_L, \lambda_R$ are positive scalars. Our goal is to find the optimal solution

$$z^* = \arg\min_z \mathcal{C}(z, I). \quad (7)$$

3.3.5 Inspiration with a class-conditioned generator.

When a generator was trained using a class input in addition to z (AC-GAN [33]), the class conditioning part of the latent vector is supposed to take discrete values. To overcome this problem, we tried two strategies:

1. Performing the search normally to find a continuous-valued vector consisting in the concatenation of z and input class vector c. Unless specified otherwise, we always adopt this strategy in our experiments.
2. Using the user provided input class kept fixed, and perform the gradient descent only on the latent part z. We denote this setting as the "conditioned" one.

3.4 Algorithm selection

3.4.1 Gradient based retrieval given a target

We experimented classical optimization algorithms such as the Adam method [23], and LBFGS [26], a well know quasi-newton algorithm approximating second order information with successive values of the gradient. The choice of Adam as an optimizer may seem odd since our problem is not stochastic. However, the function we consider in this problem shows a lot of local variations similar to noise. Using Adam is a way to smooth the optimization process. However, LBFGS has a better convergence rate than Adam: $O(1/t)$ with t the number of iterations. Therefore we expect it to show better performances. We also included Nesterov's momentum method, with various learning rates and momentum 0.9, advocated in [2].

3.4.2 Gradient free optimization

In addition to the optimizers tested above, we tried out several gradient-free optimizers in our image retrieval task depicted in Figure 2. Indeed, for solving a non-convex problem, gradient free optimizers could outperform more standard methods. Besides, at equal number of steps, they are computationally faster than gradient-based methods. After several tests, we selected three gradient-free optimizers from the Nevergrad optimization [35] library:

- Two variants of the Differential Evolution method [40] (DE): 2-points DE (2PDE) which replaces the classical pointwise crossover of DE by a two-point crossover [18] as advocated in [35], and DDE (Discrete-DE) which uses in DE the crossover rate $1/d$ with d the dimension of z, which is classical in discrete optimization.
- The discrete $(1+1)$-evolution strategy [4] (DOPO). DOPO is a special case of the $(\mu/\mu+\lambda)$-optimization described in Algorithm 1, with $\mu = \lambda = 1$. $1+\lambda$ candidate latent vectors are generated among which the μ best are selected according to the loss criterion defined above. From this selection, a new value of the optimum vector $\hat{z}$ is inferred. The process is repeated until $G(\hat{z})$ is satisfying.

Algorithm 1 $(\mu/\mu + \lambda)$-optimization, with a latent space of dimension d.

$\hat{z} := (0, \dots, 0) \in \mathbb{R}^d$.
while True **do**
  for $i \in \{1, \dots, \lambda\}$ **do**
    $z_i := \hat{z}$
    for $p \in \{1, \dots, d\}$ **do**
      With probability $1/d$ replace $z_i(p)$ with an independent standard normal value.
    end for
    repeat if no variable $z_i(p)$ was modified.
  end for
  $z_{\lambda+1} := \hat{z}$
  Select the μ best images $G(z_{\alpha_1}), \dots G(z_{\alpha_\mu})$ among $G(z_1), \dots G(z_{\lambda+1})$
  $\hat{z} := \frac{1}{\mu} \sum_{i=1}^{\mu} z_{\alpha_i}$
end while
return $\hat{z}$

Numerical scores usually do not perform as well as human judgment: with a human input we could hope to reach more visually convincing results. This is why we consider a last optimizer, HEVOL (Human in the EVOlution Loop), where a human user chooses manually at each iteration the μ best out of $1+\lambda$ possibilities. Typically for face generation, we took $\mu = 5$ and $\lambda = 27$. A target image is not even necessary with this method: the user can also try to reconstruct an image he would have in mind, or just provide preferences. This is convenient for applications such as facial composites or fashion generation (Section 5.5 and Section 5.5.2).

4 Experimental setting

4.1 Datasets.

We trained generators on four datasets: two small ones and two large ones publicly available:

- The Describable Textures Dataset (DTD) : a dataset of texture patterns [5] consisting of 5640 images labeled into 47 different categories.
- The RTW dataset described in [38], consisting in 4157 images of plain clothes in front of a white background, labeled with 7 shapes and 7 texture categories.
- The Celeba-HQ dataset introduced by Karras et al. [21], consisting in 30 000 images of celebrity faces. We do not consider the available labels for this database.
- The FashionGen dataset [36] containing 293 008 labelled fashion images. Each sample contains one item, wore by a model in the case of clothes, in front of a white background. We trained our model on the clothing sub-dataset with about 200 000 items.

4.2 Target images: reconstruction, semi-specified, miss-specified cases.

We distinguish the reconstruction case (the target image is randomly generated by the generator trained on the dataset), the semi-specified case (the target belongs to the training dataset), and the miss-specified one (the target comes from another source, e.g. Wikipedia, FFHQ [22] or DeepFashion [27]). Table 2 shows that scores degrade from specified to miss-specified, which is close to semi-specified. This is confirmed by visual inspection.

4.3 Scaling

When judging the similarity between a pair of images, we are more interested on the main patterns of the image rather on the tiniest details. That is why both I and $G(z)$ are resized to 128×128 before being fed to the feature network. Therefore, both images can be of arbitrary size and a high resolution generation can be made out of a lower resolution reference.

	L2	L2 + VGG	VGG
λ_L	50	50	0
λ_R	0	0.1	0.1
λ_S	0	1	1

Table 1 Default parameters for each criterion.

4.4 Hardware details

The progressive GAN networks were trained on two NVIDIA Tesla V100-SXM2 and took less than a week per model. We used the same hardware for the inspired generation. Our program relies on the Pytorch python framework.

4.5 Trained GAN models

We used default values of progressive growing of GANs. The dimension of z is 512. Most models have been trained until a 256×256 resolution though we ran generations up to 512×512 on the RTW and CelebA-HQ datasets.

4.6 Hyperparameters

For all gradient based optimizers, we worked with a batch size of 1. For Adam, we kept the hyperparameters used for the training of progressive growing ($(\beta_1, \beta_2) = (0, 0.99)$) and tested various learning rates. For Nesterov momentum, we use momentum 0.9 and various learning rates. In all cases we started with a base gradient step equal to one. We perform gradient decay twice, dividing the gradient step by 10 at one third of the iteration budget, and at the two thirds. To obtain better results, we reset the model's state to the one associated with the smallest loss found so far before each decay of the gradient step. We performed a grid search to select the best gradient step for Adam, Nesterov, and LBFGS. Unless specified otherwise, all optimizations are launched with 1000 iterations. All gradient-free optimization methods have been tested with their default parameters in [35].

5 Results

5.1 Criterion balance

We use various combinations of similarity measures, namely the simple ℓ_2 loss $\mathcal{C}_L$, the feature similarity $\mathcal{C}_S$ loss, the discriminator realism criterion $\mathcal{C}_R$. They are presented in Table 1.

We found in our trials that $\lambda_\nu = 1$ was an acceptable value for all models. The adequate value of λ_R depends on the training configuration of the GAN and more precisely on the discriminator's output. If D is "too strong" and returns very high scores in absolute value, then λ_R must be reduced accordingly. For Celeba-HQ and FashionGen-clothing, $\lambda_R \approx 0.1$ gave us fine outputs.

Human study. To validate the best criteria and algorithm for each case we conducted a human study. For each of 100 given targets, we asked five different participants to pick the most similar image between images generated with different criteria, or optimizers. For each dataset, we first identified by visual inspection two best performing algorithms and conducted the study to identify the best criterion between the different possibilities of Table 1. Table 3 presents the results on the DTD (semi-specified setting) and FashionGen dataset (miss-specified setting). The best identified criterion for DTD is the VGG without Realism penalty, and on FashionGen, the L2+VGG setting.

5.2 Impact of the realism penalty

For faces and fashionGen generation, using a high value of λ_R leads to less artifacts (see Figure 3). However, as shown in Figure 4, a compromise must be found between similarity and desired realism.

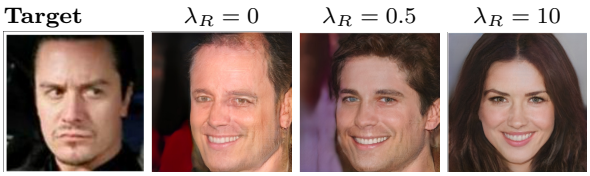


Fig. 4 Impact of the realism coefficient on a generation with a GAN trained on CelebaHQ.

5.3 Impact of the VGG and L2 terms

As far as reconstruction is concerned, the feature-based loss gives fair results when combined with Adam or LBFGS (see Figure 5). However, to perfectly reconstruct the reference image, L2 is the best option (see Figure 6): in 3000 to 5000 steps we retrieve the original output. In this case, it is worth noticing that the realism penalty should be set to zero. With a VGG criterion, we observe similar results with a greater budget.

Algorithm	Adam	RS	DOPO	DDE	LBFGS
reconstruction	17.0	52.2	37.9	34.8	14.5
semi-specified	23.5	63.0	50.6	46.5	20.6
miss-specified	25.5	64.0	49.5	46.5	19.6

Table 2 Best score (defined criterion) obtained by various algorithms with L2 + VGG loss for recovering faces with a generator trained on Celeba. Budget 1000, average over 10 images. Semi- or miss- specified cases are similar, both harder than the specified case. Visual inspection confirms numbers: the reconstruction configuration leads to more convincing results. RS:Random search.

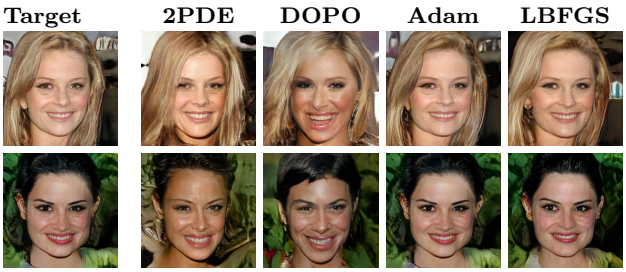


Fig. 5 Reconstruction with 3000 iterations, VGG loss using different optimizers on CelebA-HQ images. We observe superior performance of gradient-based optimizers.

Fig. 6 Reconstruction without realism penalty and LBFGS optimizer. Each configuration was iterated until perfect reproduction of the target image.

Fig. 7 Targets (first line) and images retrieved (second line) using an L2 similarity criterion and the LBFGS algorithm with 5000 iterations with a model trained on CelebaHQ.

For the semi-specified and misspeficied cases, feature-based similarity criterion are necessary to catch semantically meaningful data regardless of the target's background : haircut, hair color, pose, facial expression, etc. Besides, an L2 loss yields blurry results (see Figure 8). However, it is interesting to notice that with L2 similarity GANs can output a rough approximation of pictures sometimes very far from their training dataset (see Figure 7).

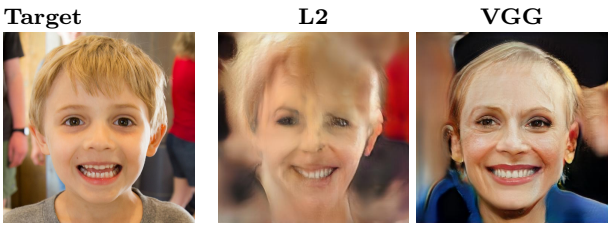


Fig. 8 Misspecified generation from FFHQ using a GAN trained on CelebaHQ and a LBFGS optimizer. Targeting features instead of pixel-wise similarity allows cleaner and sharper generations.

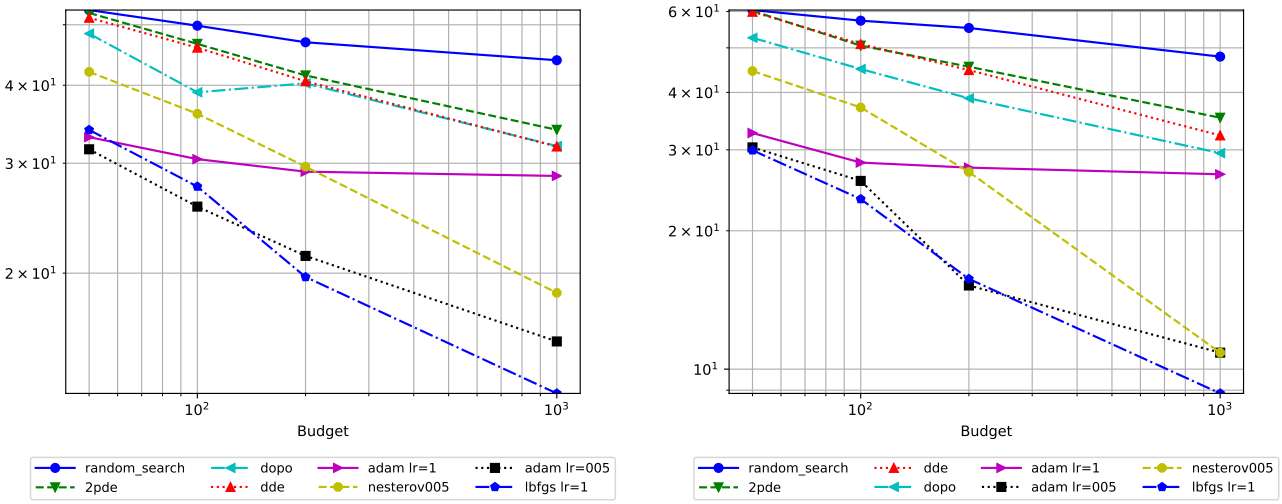
5.4 Optimization method

Figure 9 presents a benchmark of the best performing algorithms we tested. Numerically, LBFGS is the best algorithm asymptotically in most cases. But DOPO performed best among comparison-based methods, compatible with user preferences (i.e. when no criterion or no target image exists). As far as the loss is concerned, all optimizers show similar behaviors on different datasets with different loss criteria. Numerically, random search is the worst performer, followed by gradient free approaches (2PDE, DOPO, DDE), and finally gradient based approaches. As far as rendering is concerned, all optimization methods except Nesterov gave fair results with all training datasets (see Figures 10, 15, 11 and 13).

Table 4 presents results of a human study conducted to evaluate the best optimizer for GANs trained on different datasets. Surprisingly, LBFGS, which has the lowest criterion scores is outperformed by other approaches. On DTD, 2PDE ranks first, followed by DOPO. These methods ranks similarly when using samples from DeepFashion on a dataset trained on fashionGen. We attribute this phenomenon to the tendency of evolutionary algorithms to prefer stable robust optima [20]. Meanwhile, Adam seems to be the best option for face retrieval. The generative function may be more regular, besides FFHQ shows many similarities with CelebaHQ. These reasons could explain Adam's good performances.

Dealing with class conditioning. When working with a labelled generator, setting the values of the discrete part of the latent vector resulted in more stable and visually more pleasant generations (See Table 4a). And that, no matter the optimizer or the loss criterion considered.

Computation time. The time per iteration is essentially the same for all comparison-based algorithms on the one hand, and for all gradient-based methods on the other hand because most of the computation time lies in the



(a) on the CelebA dataset with criterion L2 + VGG.

(b) On the DTD dataset with criterion L2 + VGG without realism penalty.

Fig. 9 Retrieval performance measured with the criterion $\mathcal{C}$ (Reconstruction case). LBFGS dominates on both CelebA and DTD datasets. Here the x-axis is the number of calls to the generator. The methods rank similarly when working with other training datasets.

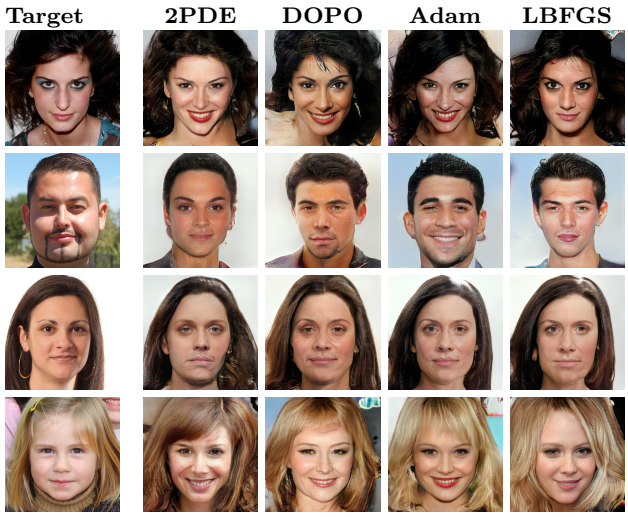


Fig. 10 Generation inspired from FFHQ samples with a model trained on CelebaHQ using VGG loss. Budget: 3000 iterations.

Fig. 11 Generation inspired from DeepFashion samples with a model trained on FashionGen using VGG loss. Budget: 3000 iterations.

computation of $G(z)$ or $\nabla G(z)$. Gradient-based methods are 5 times slower than comparison-based methods due to the additional cost for computing $\nabla G(z)$.

5.5 Preference based generation with HEVOL: Human evolution

5.5.1 Facial composite

Facial composites were originally based on a reconstruction methodology based on questions and answers on individual facial features. [10,15,39] pointed out that holistic methods, based on global faces, are competitive. None of the most recent systems [11,12,13] includes deep generators. Hair typically remains a specific dedicated feature, and masking mechanisms are included in such software. For the present research project, we focus on a global facial composite system. In our experiments, we worked with HEVOL combined with a generator trained on CelebaHQ.

We use a random human (Fig. 12, left) as a target for our facial recognition experiment. This a miss-specified case as the target's face is not in the CelebaHQ dataset. At each iteration, the user was asked to select $\mu = 5$ favorite among $1 + \lambda = 28$ images. We found out that we could get better performance and convenience

Fig. 12 A randomly chosen human (left image) and his reconstructions (right, 3 independent experimental results) obtained with HEVOL after 5, 6 and 7 iterations (about 3 minutes of user time).

Fig. 13 Images from the RTW network retrieved from texture images (DTD) with a budget of 1000 iterations using the VGG loss.

if we allowed the user to select the same image several times. We obtained convincing outputs in as few as 6 iterations.

This budget is far too low for any other non-human algorithm. LBFGS fails to reach this quality even with 400 iterations.

5.5.2 Fashion image retrieval

The same algorithm designed for facial composite can be used for fashion generation: the user just specifies which images best match his current preferences. We used this approach on a model trained on fashionGen and worked with 4 batches of 16 images. Results are presented in Fig. 14. Contrary to facial recognition, averaging performs poorly so the user only selects one picture at each iteration.

Fig. 14 User-chosen images obtained with HEVOL. The instruction was respectively to produce "sportswear", "Clothes for cold weather", "light clothes", "Sophisticated". 61 images were generated in each case, i.e. 4 generations of 15 images plus the initial one.

	L2	L2+VGG	VGG	VGG no R
cond LBFGS	0.8	13.6	5.0	26.1
cond DOPO	2.3	12.7	10.4	**29.2**

a) Criterion ablation study on DTD.

	L2	L2+VGG	VGG
LBFGS	7.8	13.5	14.7
2PDE	8.5	**35.2**	20.3

b) Criterion ablation study on the FashionGen dataset

Table 3 Selection of the best criterion setting for DTD (semi-specified) and (miss-specified) FashionGen image retrieval: human scores (% of retrieved images judged most similar to the target).

		LBFGS	Adam	DOPO	2PDE
DTD targets from DTD	cond	14.4	9.6	13.8	**25.3**
	no cond	3.1	10.5	12.6	10.7
	sum	17.5	20.1	26.4	**36.0**
CelebA-HQ targets from FFHQ		29.2	**37.2**	15.6	18.0
FashionGen targets from DeepFashion		11.7	20.9	**41.2**	26.3

Table 4 Comparison of different retrieval algorithms on various datasets (% of retrieved images judged most similar to the target). DTD and FashionGen images retrieved by evolutionary algorithms are found more related to targets, whereas CelebA-HQ generations work better with Adam.

6 Conclusion

We have shown several ways to control the output of a trained generative model. With a simple L2 loss and LBFGS, a GAN can build rough approximation of almost any image and retrieve its own generations. We have also shown that it was possible to look for semantic items (hair, expression, etc.) in a GAN's latent space using vision features and a gradient descent, like LBFGS or Adam. Unexpectedly, evolutionary methods like DOPO and 2PDE led to generations more success-

Fig. 15 Generated images for which there was a perfect human agreement. We used a VGG loss without realism penalty and the DTD dataset.

Fig. 16 Generated images with VGG loss for which there was a perfect human agreement on the FashionGen dataset.

ful when put to human inspection, whereas they numerically fail compared to LBFGS. A possible explanation is the natural tendency of such algorithms to find wide stable robust minima.

With a human user in the loop, classical $(\mu/\mu + \lambda)$ evolution strategy provided in a few minutes reasonable facial composites or acceptable fashions. This allows generations without using a target image and with an arbitrary criterion decided by the human.

It is worth noticing that these facial composites were more realistic than generations obtained by any criterion or any algorithm - confirming that humans are still better than programs at measuring similarity. Using evolutionary algorithms for facial composites is not new; but to our knowledge the present work is the first combination of evolution and deep learning.

Acknowledgements We would like to thank Matthijs Douze, Arthur Szlam, Diane Bouchacourt and Louis Martin for their useful input to this work.

References

1. Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *Arxiv preprint*, 1701.07875, 2017.
2. Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In *ICML*, 2018.
3. Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *Arxiv preprint*, 1809.11096, 2018.
4. Jianbo Cai and Georg Thierauf. Evolution strategies for solving discrete optimization problems. *Advances in Engineering Software*, 25(2):177 – 183, 1996. Computing in Civil and Structural Engineering.
5. Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.
6. Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *ICLR*, 2017.
7. Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *NIPS*, 2016.
8. Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Creative adversarial networks. In *ICCC*, 2017.
9. Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.
10. Charlie D. Frowd, Peter J. B. Hancock, Vicky Bruce, Alex H. McIntyre, Melanie Pitchford, Rebecca Atkins, Andrea Webster, John Pollard, Beverly Hunt, Emma Price, Sandra Morgan, Andrew Stoika, Romeo Dughila, Sergiu Maftei, and Gabriel Sendrea. Giving crime the 'evo': Catching criminals using evofit facial composites. In *International Conference on Emerging Security Technologies*, 2010.
11. Charlie D. Frowd, Peter J. B. Hancock, and Derek Carson. Evofit: A holistic, evolutionary facial imaging technique for creating composites. *ACM Trans. Appl. Percept.*, 1(1):19–39, 2004.
12. Charlie D. Frowd, Faye Skelton, Gemma Hepton, Laura Holden, Simra Minahil, Melanie Pitchford, Alex McIntyre, Charity Brown, and Peter J.B. Hancock. Whole-face procedures for recovering facial images from memory. *Science & Justice*, 53(2):89 – 97, 2013.
13. Ruben Garcia-Zurdo. Creation of facial composites from user selections using image gradient. *Informatica*, 04 2019.
14. Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.
15. Stuart J. Gibson, Chris J. Solomon, Matthew I. S. Maylin, and Clifford Clark. New methodology in facial composite construction: from theory to practice. *Int. J. Electron. Secur. Digit. Forensic*, 2(2):156–168, 2009.
16. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.
17. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein GANs. In *NIPS*, 2017.
18. John H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105, 1973.

19. Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.
20. Kenneth A. De Jong. Genetic algorithms are not function optimizers. In *Foundations of Genetic Algorithms*, pages 5 – 17. Elsevier, 1993.
21. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *ICLR*, 2018.
22. Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Arxiv preprint*, 1812.04948, 2018.
23. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Arxiv preprint*, 1412.6980, 2014.
24. Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *ICML*, 2017.
25. Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *NIPS*, 2017.
26. Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(1-3):503–528, 1989.
27. Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.
28. Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for GANs do actually converge? In *ICML*, 2018.
29. Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *ICLR*, 2017.
30. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.
31. Anh Mai Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *NIPS*, 2016.
32. Dong Nie, Roger Trullo, Jun Lian, Caroline Petitjean, Su Ruan, Qian Wang, and Dinggang Shen. Medical image synthesis with context-aware generative adversarial networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2017.
33. Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.
34. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016.
35. J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.
36. Negar Rostamzadeh, Seyedarian Hosseini, Thomas Boquet, Wojciech Stokowiec, Ying Zhang, Christian Jauvin, and Chris Pal. Fashion-Gen: The Generative Fashion Dataset and Challenge. *Arxiv preprint*, 1806.08317, June 2018.
37. Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Deja vu: an empirical evaluation of the memorization properties of convnets. *Arxiv preprint*, 1809.06396, 2018.
38. Othman Sbai, Mohamed Elhoseiny, Antoine Bordes, Yann LeCun, and Camille Couprie. DesIGN: Design Inspiration from Generative Networks. In *ECCV workshop on Fashion, Art and Design*, 2018.
39. Christopher Solomon, Stuart Gibson, and Matthew Maylin. A new computational methodology for the construction of forensic, facial composites. In *Computational Forensics*, pages 67–77, 2009.
40. Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, 1997.
41. Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.
42. Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.
43. Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *Arxiv preprint*, 1805.08318, 2018.
44. Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018.
45. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.
46. Shizhan Zhu, Sanja Fidler, Raquel Urtasun, Dahua Lin, and Chen Change Loy. Be your own prada: Fashion synthesis with structural coherence. *ICCV*, 2017.
47. Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. Imagine it for me: A generative adversarial approach for zero-shot learning from noisy texts. In *CVPR*, 2018.

A Layers considered in vgg19

When using vgg19 as a feature extractor, we always consider the output of 3 ReLU layers:

– The first one after the first MaxPooling operation
– The first one after the second MaxPooling operation
– The first one after the third MaxPooling operation