EvolGAN: Evolutionary Generative Adversarial Networks

Baptiste Rozière, Fabien Teytaud, Vlad Hosu, Hanhe Lin, Jeremy Rapin, Mariia Zameshina, Olivier Teytaud

EvolGAN: Evolutionary Generative Adversarial Networks

Baptiste Roziere[1⋆], Fabien Teytaud[⋆2], Vlad Hosu[3], Hanhe Lin[3], Jeremy Rapin[1], Mariia Zameshina[4], and Olivier Teytaud[1]

[1] Facebook AI Research {broz,jrapin,oteytaud}@fb.com
[2] Univ. Littoral Cote d'Opale teytaud@univ-littoral.fr
[3] Univ. Konstanz, Germany {hanhe.lin,vlad.hosu}@uni-konstanz.de
[4] Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, France

Abstract. We propose to use a quality estimator and evolutionary methods to search the latent space of generative adversarial networks trained on small, difficult datasets, or both. The new method leads to the generation of significantly higher quality images while preserving the original generator's diversity. Human raters preferred an image from the new version with frequency 83.7% for Cats, 74% for FashionGen, 70.4% for Horses, and 69.2% for Artworks - minor improvements for the already excellent GANs for faces. This approach applies to any quality scorer and GAN generator.

1 Introduction

Generative adversarial networks (GAN) are the state-of-the-art generative models in many domains. However, they need quite a lot of training data to reach a decent performance. Using off-the-shelf image quality estimators, we propose a novel but simple evolutionary modification for making them more reliable for small, difficult, or multimodal datasets. Contrarily to previous approaches using evolutionary methods for image generation, we do not modify the training phase. We use a generator G mapping a latent vector z to an image $G(z)$ built as in a classical GAN. The difference lies in the method used for choosing a latent vector z. Instead of randomly generating a latent vector z, we perform an evolutionary optimization, with z as decision variables and the estimated quality of $G(z)$ — based on a state-of-the-art quality estimation method— as an objective function. We show that:

- The quality of generated images is better, both for the proxy used for estimating the quality, i.e., the objective function, as well as for human raters. For example, the modified images are preferred by human raters more than 80% of the time for images of cats and around 70% of the time for horses and artworks.
- The diversity of the original GAN is preserved: the new images are preferred by humans and still similar.

⋆ Equal contribution

- The computational overhead introduced by the evolutionary optimization is moderate, compared to the computational requirement for training the original GAN.

The approach is simple, generic, easy to implement , and fast. It can be used as a drop-in replacement for classical GAN provided that we have a quality estimator for the outputs of the GAN. Besides the training of the original GAN, many experiments were performed on a laptop without any GPU. Fig. 1 shows examples

Fig. 1. For illustration , random images generated using StyleGAN2 (left) and EvolGAN (right). Horses were typically harder than cats. The images generated by EvolGAN are generally more realistic. The top-left example of a generated cat by StyleGAN2 has blood-like artifacts on its throat and the other is blurry. Three of the four StyleGAN2 horses are clearly unrealistic: on the bottom right of the StyleGan2 results, the human and the horse are mixed, the bottom left shows an incoherent mix of several horses, the top left looks like the ghost, and only the top right is realistic. Overall, both cats, and 3 of the 4 horses generated by EvolGAN look realistic. We show more examples of horses, as they are more difficult to model.

of generations of $EvolGAN_{StyleGAN2}$ compared to generations by StyleGAN2. Fig. 2 presents our general approach, detailed in the Section 3.

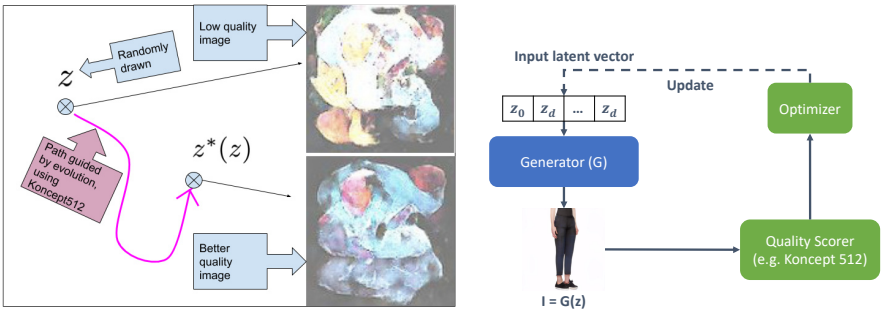

Fig. 2. General EvolGAN approach (left) and optimization loop (right). Our method improves upon a pre-trained generator model G, which maps a latent vector z to an image G(z). In classical models, the images are generated by sampling a latent vector z randomly. We modify that randomly chosen z into a better $z^*(z_0)$ evolved to improve the quality of the image, typically estimated by Koncept512. For preserving diversity, we ensure that $z^*(z_0)$ is close to the original z_0. The original image is generated using PokeGan (Pokemon dataset) and the improved one is generated using EvolGAN (superposed on top of PokeGan): we see an elephant-style Pokemon, hardly visible in the top version.

2 Related Works

2.1 Generative Adversarial Networks

Generative Adversarial Networks [1] (GANs) are widely used in machine learning [2,3,4,5,6,7,8,9] for generative modeling. Generative Adversarial Networks are made up of two neural networks: a Generator G, mapping a latent vector z to an image $G(z)$ and a Discriminator D mapping an image I to a realism value $D(I)$. Given a dataset $\mathcal{D}$, GANs are trained using two training steps operating concurrently:

(i) Given a randomly generated z, the generator G tries to fool D into classifying its output $G(z)$ as a real image, e.g. by maximizing $\log D(G(z))$. For this part of the training, only the weights of G are modified.
(ii) Given a minibatch containing both random fake images $F = \{G(z_1), \ldots, G(z_k)\}$ and real images $R = \{I_1, \ldots, I_k\}$ randomly drawn in $\mathcal{D}$, the discriminator learns to distinguish R and F , e.g. by optimizing the cross-entropy.

The ability of GANs to synthesize faces [10] is particularly impressive and of wide interest. However, such results are possible only with huge datasets for each modality and/or after careful cropping, which restricts their applicability.
Here we consider the problem of improving GANs trained on small or difficult datasets. Classical tools for making GANs compatible with small datasets include:

- Data augmentation, by translation, rotation, symmetries, or other transformations.
- Transfer from an existing GAN trained on another dataset to a new dataset [11].
- Modification of the distribution in order to match a specific request as done in several papers. [12] modifies the training, using quality assessement as we do; however they modify the training whereas we modify inference. In the same vein, [13] works on scale disentanglement: it also works at training time. These works could actually be combined with ours. [14] optimizes the injected noise in a super resolution GAN using a criterion combining the Koncept512 and the discriminator of the GAN. However, their method is only applied to super resolution and their method cannot be applied directly to any generative model. [15] generates images conditionally to a classifier output or conditionally to a captioning network output. [16] and [17] condition the generation to a playability criterion (estimated by an agent using the GAN output) or some high-level constraints. [18] uses a variational autoencoder (VAE), so that constraints can be added to the generation: they can add an attribute (e.g. black hair) and still take into account a realism criterion extracted from the VAE: this uses labels from the dataset. [19] uses disentanglement of the latent space for semantic face editing: the user can modify a specific latent variable. [20] allows image editing and manipulation: it uses projections onto the output domain.
- Biasing the dataset. [21] augments the dataset by generating images with a distribution skewed towards the minority classes.
- Learning a specific probability distribution, rather than using a predefined, for example Gaussian, distribution. Such a method is advocated in [22].

The latter is the closest to the present work in the sense that we stay close to the goal of the original GAN, i.e. modeling some outputs without trying to bias the construction towards some subset. However, whereas [22] learn a probability distribution on the fly while training the GAN, our approach learns a classical GAN and modifies, a posteriori, the probability distribution by considering a subdomain of the space of the latent variables in which images have better quality. We could work on an arbitrary generative model based on latent variables, not only GANs. As opposed to all previously mentioned works, we improve the generation, without modifying the target distribution and without using any side-information or handcrafted criterion - our ingredient is a quality estimator. Other combinations of deep learning and evolutionary algorithms have been published around GANs. For instance, [23] evolves a population of generators, whereas our evolutionary algorithm evolves individuals in the latent space. [24] also evolves individuals in the latent space, but using human feedback rather than the quality estimators that we are using. [25] evolves individuals in the latent space, but either guided by human feedback or by using similarity to a target image.

2.2 Quality estimators: Koncept512 and AVA

Quality estimation is a long-standing research topic [26,27] recently improved by deep learning [28]. In the present work, we focus on such quality estimation tools based on supervised convolutional networks. The KonIQ-10k dataset is a large publicly available image quality assessment dataset with 10,073 images rated by humans. Each image is annotated by 120 human raters. The Koncept512 image quality scorer [28] is based on an InceptionResNet-v2 architecture and trained on KonIQ-10k for predicting the mean opinion scores of the annotators. It takes as input an image I and outputs a quality estimate $K(I) \in \mathbb{R}$. Koncept512 is the state of the art in technical quality estimation [28], and is freely available. We use the release without any modification. [29] provides a tool similar to Koncept512, termed AVA, but dedicated to aesthetics rather than technical quality. It was easy to apply it as a drop-in replacement of Koncept512 in our experiments.

3 Methods

3.1 Our algorithm: EvolGAN

We do not modify the training of the GAN. We use a generator G created by a GAN. G takes as input a random latent vector z, and outputs an image $G(z)$. While the latent vector is generally chosen randomly (e.g., $z \leftarrow \mathcal{N}(0, I_d)$), we treat it as a free parameter to be optimized according to a quality criterion Q. More formally, we obtain $z^*(z_0)$:

$$z^*(z_0) = \arg\max_z Q(G(z)) \text{ in the neighborhood of a random } z_0. \quad (1)$$

In this paper, Q is either Koncept512 or AVA. Our algorithm computes an approximate solution of problem 1 and outputs $G(z^*(z_0))$. Importantly, we do not want a global optimum of Eq. 1. We want a local optimum, in order to have essentially the same image – $z^*(z_0)$ must be close to z_0, which would not happen without this condition. The optimization algorithm used to obtain z^* in Eq. 1 is a simple $(1+1)$-Evolution Strategy with random mutation rates [30], adapted as detailed in Section 3.2 (see Alg. 1). We keep the budget of our algorithm low, and the mutation strength parameter α can be used to ensure that the image generated by EvolGAN is similar to the initial image. For instance, with $\alpha = 0$, the expected number of mutated variables is, by construction (see Section 3.1), bounded by b. We sometimes use the aesthetic quality estimator AVA rather than the technical quality estimator Koncept512 for quality estimation. We consider a coordinate-wise mutation rate: we mutate or do not mutate each coordinate, independently with some probability.

3.2 Optimization algorithms

After a few preliminary trials we decided to use the $(1+1)$-Evolution Strategy with uniform mixing of mutation rates [30], with a modification as described

Algorithm 1: The $EvolGAN_{G,b,\alpha}$ algorithm

Parameters:
- A probability distribution $\mathcal{P}$ on $\mathbb{R}^d$.
- A quality estimator Q, providing an estimate $Q(I)$ of the quality of some $I \in E$. We use $Q = Koncept512$ or $Q = AVA$.
- A generator G, building $G(z) \in E$ for $z \in \mathbb{R}^d$.
- A budget b.
- A mutation strength $0 \leq \alpha \leq \infty$.
- A randomly generated $z \leftarrow random(\mathcal{P})$. $I = G(z)$ is the baseline image we are trying to improve.

```
1  for i ∈ {1, ..., b} do
2  |  r := Clip(1/d, 1, α × uniform([0, 1]))
3  |  z' := z
4  |  for j ∈ {1, ..., d} do
5  |  |  with probability r, z'_i ← random(P)_i (i^th marginal of P).
6  |  end
7  |  if Q(G(z)) < Q(G(z')) then
8  |  |  z ← z'
9  |  end
10 end
   Output : Optimized image I' = G(z)
```

in algorithm 1. This modification is designed for tuning the compromise between quality and diversity as discussed in Table 1. We used $Clip(a, b, c) = \max(a, \min(b, c))$. Optionally, z_0 can be provided as an argument, leading to $EvolGAN_{G,b,\alpha,z_0}$. The difference with the standard uniform mixing of mutation rates is that $\alpha \neq 1$. With $\alpha = 0$, the resulting image I' is close to the original image I, whereas with $\alpha = \infty$ the outcome I' is not similar to I. Choosing $\alpha = 1$ (or $\alpha = \frac{1}{2}$, closely related to FastGA[31]) leads to faster convergence rates but also to less diversity (see Alg.1, line 2). We will show that overall, $\alpha = 0$ is the best choice for EvolGAN. We therefore get algorithms as presented in Table 1.

3.3 Open source codes

We use the GAN publicly available at `https://github.com/moxiegushi/pokeGAN`, which is an implementation of Wasserstein GAN [32], the StyleGAN2 [10] available at `thispersondoesnotexist.com`, and PGAN on FashionGen from Pytorch GAN zoo [33]. Koncept512 is available at `https://github.com/subpic/koniq`. Our combination of Koncept512 and PGAN is available at DOUBLEBLIND. We use the evolutionary programming platform Nevergrad [34].

$0 \leq \alpha \leq \infty$	behavior of $EvolGAN_{G,b,\alpha}$	$\mathbb{E}\|\|z^*(z_0) - z\|\|_0$ with budget b
$0 \leq \alpha \leq \frac{1}{d}$	standard $(1+1)$ evol. alg. with mutation rate $r = \frac{1}{d}$.	$\leq b$
$\alpha = 1$	uniform mixing of mutation rates [30] (also related to [31]).	
$\alpha = \infty$	all variables mutated: equivalent to random search	
intermediate values α	intermediate behavior	$\leq \min(\max(\alpha, 1/d)bd, d)$

Table 1. Optimization algorithms used in the present paper. The last setting is new compared to [30]. We modified the maximum mutation rate α for doing a local or global search depending on α, so that the diversity of the outputs is maintained when α is small (Sect. 4.3). $||x||_0$ denotes the number of non-zero components of x.

4 Experiments

We present applications of EvolGAN on three different GAN models: (i) StyleGAN2 for faces, cats, horses and artworks (ii) PokeGAN for mountains and Pokemons (iii) PGAN from Pytorch GAN zoo for FashionGen.

4.1 Quality improvement on StyleGAN2

The experiments are based on open source codes [34,33,35,28,29]. We use the StyleGAN2 [10] trained on a horse dataset, a face dataset, an artwork dataset, and a cat dataset[5]. **Faces.** We conducted a human study to assess the quality of EvolGAN compared to StyleGAN, by asking to 10 subject their preferred generations (pairwise comparisons, double-blind, random positioning). There were 11 human raters in total, including both experts with a strong photography background and beginners. 70% of the ratings came from experts. Results appear in Table 2. Faces are the most famous result of StyleGAN2. Although the results are positive as the images generated by EvolGAN are preferred significantly more than 50% of the time, the difference between StyleGAN2 and EvolGAN is quite small on this essentially solved problem compared to wild photos of cats or horses or small datasets. **Harder settings.** Animals and artworks are a much more difficult setting (Fig. 3) - StyleGAN2 sometimes fails to propose a high quality image. Fig. 3 presents examples of generations of $StyleGAN2$ and $EvolGAN_{StyleGAN2,b,\alpha}$ in such cases. Here, $EvolGAN$ has more headroom for providing improvements than for faces: results are presented in Table 3. The case of horses or cats is particularly interesting: the failed generations often contain globally unrealistic elements, such as random hair balls flying in the air or unusual positioning of limbs, which are removed by EvolGAN. For illustration

[5] `https://www.thishorsedoesnotexist.com/`,`https://www.thispersondoesnotexist.com/`,`https://www.thisartworkdoesnotexist.com/`,`https://www.thiscatdoesnotexist.com/`

	$EvolGAN_{1,\infty} = G$	$EvolGAN_{10,\infty}$	$EvolGAN_{20,\infty}$	$EvolGAN_{40,\infty}$
$EvolGAN_{10,\infty}$	60.0			
$EvolGAN_{20,\infty}$	50.0	57.1		
$EvolGAN_{40,\infty}$	75.0	44.4	66.7	
$EvolGAN_{80,\infty}$	53.8	53.8	40.0	46.2
10-80 aggregated	60.4% ± 3.4% (208 ratings)			

Table 2. Human study on faces dataset. $\alpha = \infty$, quality estimator $q = Koncept512$. Row X, col. Y: frequency at which human raters preferred $EvolGAN_{X,\infty}$ to $EvolGAN_{Y,\infty}$. By construction, for all α, $EvolGAN_{1,\alpha}$ is equal to the original GAN. The fifth row aggregates all results of the first four rows for more significance.

Dataset	Budget b	Quality estimator	score
Cats	300	Koncept512	83.71 ±1.75% (446 ratings)
Horses	300	Koncept512	70.43 ± 4.27% (115 ratings)
Artworks	300	Koncept512	69.19 ± 3.09% (224 ratings)

Table 3. Difficult test beds. $\alpha = \infty$; same protocol as in Tables 2 i.e. we check with which probability human raters prefer an image generated by $EvolGAN_{G,b,\alpha}$ to an image generated by the original Gan G. By definition of EvolGAN, $\forall \alpha, G = EvolGAN_{G,1,\alpha}$. Number are above 50%: using EvolGAN for modifying the latent vector z improves the original StyleGAN2.

purpose, in Fig. 3 we present a few examples of generations which go wrong for the original StyleGan2 and for $EvolGAN_{StyleGan2,b=100,\alpha=0}$; the bad examples in the case of the original StyleGan2 are much worse.

4.2 Small difficult datasets and $\alpha = 0$

In this section we focus on the use of EvolGAN for small datasets. We use the original pokemon dataset in PokeGAN [35] and an additional dataset created from copyright-free images of mountains. The previous section was quite successful, using $\alpha = \infty$ (i.e. random search). The drawback is that the obtained images are not necessarily related to the original ones, and we might lose diversity (though Section 4.3 shows that this is not always the case, see discussion later). We will see that $\alpha = \infty$ fails in the present case. In this section, we use α small, and check if the obtained images $EvolGAN_{G,b,\alpha,z}$ are better than $G(z_0)$ (see Table 5) and close to the original image $G(z_0)$ (see Fig.4). Fig. 4 presents a Pokemon generated by the default GAN and its improved counterpart obtained by $EvolGAN$ with $\alpha = 0$. Table 5 presents our experimental setting and the results of our human study conducted on PokeGAN. We see a significant improvement when using Koncept512 *on real-world data* (as opposed to drawings such as Pokemons, for which Koncept512 fails), whereas we fail with AVA as in

Context	LPIPS score
PGAN	0.306 ± 0.0003
$EvolGAN_{PGAN,b=40,\alpha=0}$	0.303 ± 0.0003
$EvolGAN_{PGAN,b=40,\alpha=1}$	0.286 ± 0.0003
$EvolGAN_{PGAN,b=40,\alpha=\infty}$	0.283 ± 0.0002

Table 4. LPIPS scores on FashionGen. As expected, $\alpha = 0$ mostly preserves the diversity of the generated images, while higher values of α can lead to less diversity for the output of EvolGAN. The LPIPS was computed on samples of $50,000$ images for each setting.

(a) Cherry-picked poor StyleGAN2 generation

(b) Random EvolGAN generation

(c) Cherry-picked poor EvolGAN generation

(d) Cherry-picked poor StyleGAN2 generation

Fig. 3. For illustration, bad generations by StyleGAN2 and by EvolGAN. (a) Generation of a cat by StyleGAN2: we looked for a bad generation and found that one. Such bad cases completely disappear in the EvolGAN counterpart. (b) example of cat generation by EvolGAN: we failed to find a really bad EvolGAN generated image. (c) Example of bad horse generation by EvolGAN: the shape is unusual (looks like the muzzle of a pork) but we still recognize a horse. (d) Bad generation of a horse by StyleGAN2: some hair balls are randomly flying in the air.

previous experiments (see Table 2). We succeed on drawings with Koncept512 only with $\alpha = 0$: on this dataset of drawings (poorly adapted to Koncept512), α large leads to a pure black image.

4.3 Quality improvement

Pytorch Gan Zoo [33] is an implementation of progressive GANs (PGAN[36]), applied here with FashionGen [37] as a dataset. The dimension of the latent space is 256. In Table 6, we present the results of our human study comparing $EvolGAN_{PGAN,b,\alpha}$ to $EvolGAN_{PGAN,1,\alpha} = PGAN$. With $\alpha = 0$, humans prefer EvolGAN to the baseline in more than 73% of cases, even after only 40 iterations. $\alpha = 0$ also ensures that the images stay close to the original images when the budget is low enough (see Table 1). Fig. 5 shows some examples of generations using EvolGAN and the original PGAN.

Type of images	Number of of images	Number of training epochs	Budget b	Quality estimator	α	Frequency of image preferred to original
Real world scenes						
Mountains	84	4900	500	Koncept512	0	73.3% ± 4.5% (98 ratings)
Artificial scenes						
Pokemons	1840	4900	500	Koncept512	0	55%
Pokemons	1840	4900	2000	Koncept512	0	52%
Pokemons	1840	4900	6000	Koncept512	0	56.3 ± 5.2% (92 ratings)
Artificial scenes, higher mutation rates						
Pokemons	1840	4900	500	Koncept512	1/7	36.8%
Pokemons	1840	4900	20	Koncept512	∞	0%

Table 5. Experimental results with $EvolGAN_{PokeGAN,b,\alpha=0}$. Reading guide: the last column shows the probability that an image $EvolGAN_{PokeGAN,b,\alpha=0,z} = G(z^*(z_0))$ was prefered to the starting point $PokeGAN(z_0)$. The dimension of the latent space is $d = 256$ except for mountains ($d = 100$). Koncept512 performs well on real world scenes but not on artificial scenes. For Pokemon with $\alpha = \infty$, the 0% (0 success out of 24 tests!) is interesting: the code starts to generate almost uniform images even with a budget $b = 20$, showing that Koncept512 fails on drawings. On mountains (the same GAN, but trained on real world images instead of Pokemons), and to a lesser extent on Pokemons for small α, the images generated using EvolGAN are preferred more than 50% of the time: using EvolGAN for modifying the latent vector z improves the original PokeGan network.

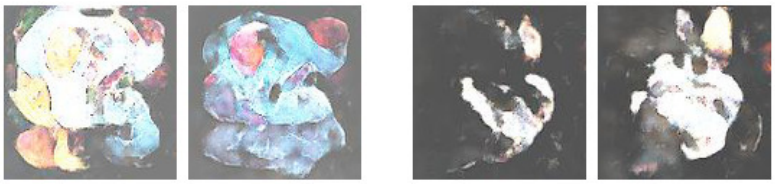

Fig. 4. In both cases, a Pokemon generated by the default GAN (left) and after improvement by Koncept512 (right). For the left pair, after improvement, we see eyes and ears for a small elephant-style pokemon sitting on his back. A similar transformation appears for the more rabbit-style pokemon on the right hand side. These cherry-picked examples (cherry-picked, i.e. we selected specific cases for illustration purpose) are, however, less convincing than the randomly generated examples in Fig. 5 - Pokemons are the least successful applications, as Koncept512, with α large or big budgets, tends to push those artificial images towards dark almost uniform images.

4.4 Consistency: preservation of diversity.

Here we show that the generated image is close to the one before the optimization. More precisely, given $z \mapsto G(z)$, the following two methods provide related outputs: method 1 (classical GAN) outputs $G(z_0)$, and method 2 (EvolGAN) outputs $EvolGAN_{G,b,\alpha,z} = G(z^*(z_0))$, where $z^*(z_0)$ is obtained by our evolutionary algorithms starting at z with budget b and parameter α (Sect. 3.2). Fig. 5 shows some example generated images using PGAN and EvolGAN. For most examples, $G(z^*(z_0))$ is very similar to $G(z_0)$ so the diversity of the original GAN is preserved well. Following [38,39], we measure numerically the diversity of the

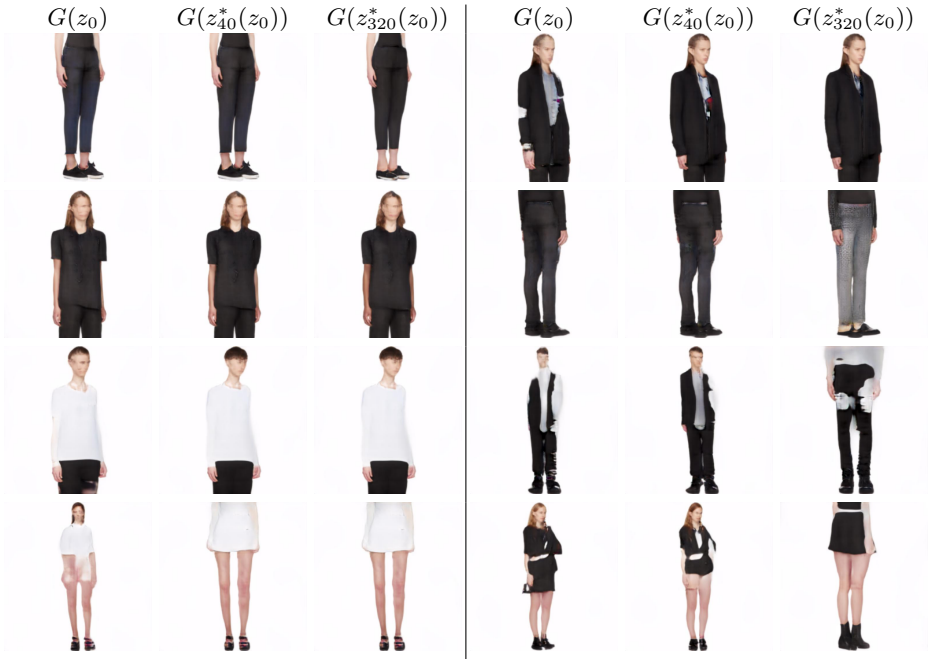

Fig. 5. Preservation of diversity, in particular with budget 40, when $\alpha = 0$. We present triplets $PGAN(z), PGAN(z^*_{40}(z)) = EvolGAN(PGAN, b = 40, \alpha = 0, z), PGAN(z^*_{320}(z)) = EvolGAN(PGAN, b = 320, \alpha = 0, z)$, i.e. in each case the output of PGAN and its optimized counterpart with budgets 40 and 320 respectively (images 1, 4, 7, 10, 13, 16, 19, 22 are the $G(z)$ and images 2, 5, 8, 11, 14, 17, 20, 23 are their counterparts $G(z^*_{40}(z))$ for budget 40; indices +1 for budget 320). With $\alpha = 1$ (unpresented) $PGAN(z)$ and $EvolGAN_{PGAN,b,\alpha,z}$ are quite different, so that we can not guarantee that diversity is preserved. With $\alpha = 0$ diversity is preserved with $b = 40$: for each of these 8 cases, the second image ($b = 40$) is quite close to the original image, just technically better — except the 8^{th} one for which $G(z)$ is quite buggy and EvolGAN can rightly move farther from the original image. Diversity is less preserved with $b = 320$: e.g. on the top right we see that the dress becomes shorter at $b = 320$.

b	Frequency
$\alpha = 0$	
40	**73.33 ± 8.21% (30 ratings))**
320	**75.00 ± 8.33% (28 ratings))**
40 and 320 aggreg.	**74.13 ± 5.79% (58 ratings))**
$\alpha = 1$	
40	48.27 ± 9.44% (29 ratings)
320	67.74 ± 8.53% (31 ratings)
40 and 320 aggreg.	58.33 ± 6.41% (60 ratings)
$\alpha = \infty$	
40	56.66 ± 9.20% (30 ratings)
320	66.66 ± 9.24% (27 ratings)
40 and 320 aggreg.	61.40 ± 6.50% (57 ratings)
All α aggregated	
40	59.55 ± 5.23% (89 ratings)
320	69.76 ± 4.98% (86 ratings)
40 and 320 aggreg.	64.57 ± 3.62% (175 ratings)

Table 6. Frequency (a.k.a score) at which various versions of $EvolGAN_{PGAN,b,\alpha,z} = PGAN(z^*(z_0))$ were preferred to their starting point $PGAN(z)$ on the FashionGen dataset. This experiment is performed with Koncept512 as a quality estimator. In most experiments we get the best results with $\alpha = 0$ and do not need more than a budget $b = 40$. The values are greater than 50%, meaning that EvolGAN improves the original PGAN network on FashionGen according to human preferences.

generated images from the PGAN model, and from EvolGAN models based on it, using the LPIPS score. The scores were computed on samples of $50,000$ images generated with each method. For each sample, we computed the LPIPS with another randomly-chosen generated image. The results are presented in Table 4. Higher values correspond to higher diversity of samples. EvolGAN preserves the diversity of the images generated when used with $\alpha = 0$.

4.5 Using AVA rather than Koncept512

In Table 7 we show that AVA is less suited than Koncept512 as a quality assessor in EvolGAN. The human annotators do not find the images generated using EvolGAN with AVA to be better than those generated without EvolGAN. We hypothesize that this is due to the subjectivity of what AVA estimates: aesthetic quality. While humans generally agree on the factors accounting for the technical quality of an image (sharp edges, absence of blur, right brightness), annotators often disagree on aesthetic quality. Another factor may be that aesthetics are inherently harder to evaluate than technical quality.

	$EvolGAN_{1,\infty} = G$	$EvolGAN_{10,\infty}$	$EvolGAN_{20,\infty}$	$EvolGAN_{40,\infty}$
$EvolGAN_{10,\infty}$	34.8			
$EvolGAN_{20,\infty}$	52.0	42.8		
$EvolGAN_{40,\infty}$	39.1	32.0	36.4	
$EvolGAN_{80,\infty}$	52.2	52.2	40.9	56.0%
$EvolGAN_{10-80,\infty}$ (aggregated)	44.5% ± 5.0%			
500	50.55 ± 3.05 %			

(a) Faces with StyleGAN2: reproducing Table 2 with AVA in lieu of Koncept512.

Dataset	Budget b	Quality estimator	score
Cats	300	AVA	47.05 ±7.05%
Artworks	300	AVA	55.71 ± 5.97 %

(b) Reproducing Table 3 with AVA in lieu of Koncept512.

Type of images	Number of images	Number of training epochs	Budget b	Quality estimator	α	Frequency of image preferred to original
Mountains	84	4900	500	AVA	0	42.5%
Pokemons	1840	4900	500	AVA	0	52.6%
Pokemons	1840	4900	500	AVA	1/13	52.6%

(c) Reproducing Table 5 with AVA rather than Koncept512.

Table 7. Testing AVA rather than Koncept512 as a quality estimator. With AVA, EvolGAN fails to beat the baseline according to human annotators.

5 Conclusion

We have shown that, given a generative model $z \mapsto G(z)$, optimizing z by an evolutionary algorithm using Koncept512 as a criterion and preferably with $\alpha = 0$ (i.e. the classical $(1+1)$-Evolution Strategy), leads to

- The generated images are preferred by humans as shown on Table 3 for StyleGAN2, Table 5 for PokeGan and Table 6 for PGAN on FashionGen
- The diversity is preserved, as shown on Fig. 5 and Table 4, when using a small value for α (see Section 3.2).

Choosing α. α small, i.e., the classical $(1+1)$-Evolution Strategy with mutation rate $1/d$, is usually the best choice: we preserve the diversity (with provably a small number of mutated variables, and experimentally a resulting image differing from the original one mainly in terms of quality), and the improvement compared to the original GAN is clearer as we can directly compare $EvolGAN_{G,b=d,\alpha=0,z}$ to $G(z)$ — a budget $b \simeq d/5$ was usually enough. Importantly, evolutionary algorithms clearly outperformed random search and not only in terms of speed: we completely lose the diversity with random search, as well as the ability to improve a given point. Our application of evolution is a case in which we provably preserve diversity — with a mutation rate bounded by $\max(\alpha, 1/d)$, and a budget $b = d/5$, and a dimension d, we get an expected ratio of mutated variables at most $b \times \max(\alpha, 1/d)$. In our setting $b = 40, d = 256, \alpha = 0$ so the maximum expected ratio of mutated variables is

at most 40/256 in Fig. 5. A tighter, run-dependent bound, can be obtained by comparing z_0 and $z^*(z_0)$ and could be a stopping criterion.

Successes. We get an improved GAN without modifying the training. The results are positive in all cases in particular difficult real-world data (Table 3), though the gap is moderate when the original model is already excellent (faces, Table 2) or when data are not real-world (Pokemons, Table 5.EvolGAN with Koncept512 is particularly successful on several difficult cases with real-world data — Mountains with Pokegan, Cats, Horses and Artworks with StyleGAN2 and FashionGen with Pytorch Gan Zoo.

Remark on quality assessement. Koncept512 can be used on a wide range of applications. As far as our framework can tell, it outperforms AVA as a tool for EvolGAN (Table 7). However, it fails on artificial scenes such as Pokemons, unless, we use a small α for staying local.

Computational cost. All the experiments with PokeGAN presented here could be run on a laptop without using any GPU. The experiments with StyleGAN2 and PGAN use at most 500 (and often just 40) calls to the original GAN, without any specific retraining: we just repeat the inference with various latent vectors z chosen by our evolutionary algorithm as detailed in Section 3.1.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Project-ID 251654672, TRR 161 (Project A05).

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. NeurIPS (2014)
2. Sbai, O., Elhoseiny, M., Bordes, A., LeCun, Y., Couprie, C.: DesIGN: Design Inspiration from Generative Networks. ECCV workshop on Fashion, Art and Design (2018)
3. Zhu, S., Fidler, S., Urtasun, R., Lin, D., Loy, C.C.: Be your own prada: Fashion synthesis with structural coherence. ICCV (2017)
4. Elgammal, A., Liu, B., Elhoseiny, M., Mazzone, M.: Creative adversarial networks. ICCC (2017)
5. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. ICCV (2017)
6. Park, T., Liu, M., Wang, T., Zhu, J.: Semantic image synthesis with spatially-adaptive normalization. CVPR (2019)
7. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. ICLR (2017)
8. Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., Greenspan, H.: GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing **321** (2018) 321–331
9. Nie, D., Trullo, R., Lian, J., Petitjean, C., Ruan, S., Wang, Q., Shen, D.: Medical image synthesis with context-aware generative adversarial networks. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. (2017)

10. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan (2019)
11. Noguchi, A., Harada, T.: Image generation from small datasets via batch statistics adaptation. CoRR **abs/1904.01774** (2019)
12. PARIMALA, K., Channappayya, S.: Quality aware generative adversarial networks. In: Advances in Neural Information Processing Systems. (2019) 2948–2958
13. Yi, Z., Chen, Z., Cai, H., Mao, W., Gong, M., Zhang, H.: Bsd-gan: Branched generative adversarial network for scale-disentangled representation learning and image synthesis. IEEE Transactions on Image Processing (2020)
14. Roziere, B., Rakotonirina, N.C., Hosu, V., Rasoanaivo, A., Lin, H., Couprie, C., Teytaud, O.: Tarsier: Evolving noise injection in super-resolution gans. arXiv preprint arXiv:2009.12177 (2020)
15. Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., Yosinski, J.: Plug & play generative networks: Conditional iterative generation of images in latent space (2016)
16. Volz, V., Schrum, J., Liu, J., Lucas, S.M., Smith, A., Risi, S.: Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '18, New York, NY, USA, Association for Computing Machinery (2018) 221–228
17. Giacomello, E., Lanzi, P.L., Loiacono, D.: Searching the latent space of a generative adversarial network to generate doom levels. In: 2019 IEEE Conference on Games (CoG). (2019) 1–8
18. Engel, J.H., Hoffman, M., Roberts, A.: Latent constraints: Learning to generate conditionally from unconditional generative models. CoRR **abs/1711.05772** (2017)
19. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing (2019)
20. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: European conference on computer vision, Springer (2016) 597–613
21. Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., Malossi, C.: Bagan: Data augmentation with balancing gan. arXiv preprint arXiv:1803.09655 (2018)
22. Gurumurthy, S., Sarvadevabhatla, R.K., Radhakrishnan, V.B.: Deligan : Generative adversarial networks for diverse and limited data. CoRR **abs/1706.02071** (2017)
23. Wang, C., Xu, C., Yao, X., Tao, D.: Evolutionary generative adversarial networks. CoRR **abs/1803.00657** (2018)
24. Bontrager, P., Lin, W., Togelius, J., Risi, S.: Deep interactive evolution. In: International Conference on Computational Intelligence in Music, Sound, Art and Design, Springer (2018) 267–282
25. Riviere, M., Teytaud, O., Rapin, J., LeCun, Y., Couprie, C.: Inspirational adversarial image generation. arXiv preprint 1906.11661 (2019)
26. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing (2004)
27. Ye, P., Kumar, J., Kang, L., Doermann, D.: Unsupervised feature learning framework for no-reference image quality assessment. IEEE Conference on Computer Vision and Pattern Recognition (2012)
28. Hosu, V., Lin, H., Sziranyi, T., Saupe, D.: Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment. IEEE Transactions on Image Processing (2020) 1–1

29. Hosu, V., Goldlucke, B., Saupe, D.: Effective aesthetics prediction with multi-level spatially pooled features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 9375–9383
30. Dang, D.C., Lehre, P.K.: Self-adaptation of mutation rates in non-elitist populations. In: International Conference on Parallel Problem Solving from Nature, Springer (2016) 803–813
31. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference. (2017) 777–784
32. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning. (2017) 214–223
33. Riviere, M.: Pytorch GAN Zoo. `https://GitHub.com/FacebookResearch/pytorch_GAN_zoo` (2019)
34. Rapin, J., Teytaud, O.: Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad` (2018)
35. Moxiegushi: Pokegan. `https://github.com/moxiegushi/pokeGAN` (2018)
36. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. ICLR (2018)
37. Rostamzadeh, N., Hosseini, S., Boquet, T., Stokowiec, W., Zhang, Y., Jauvin, C., Pal, C.: Fashion-Gen: The Generative Fashion Dataset and Challenge. Arxiv preprint **1806.08317** (2018)
38. Huang, X., Liu, M., Belongie, S.J., Kautz, J.: Multimodal unsupervised image-to-image translation. CoRR **abs/1804.04732** (2018)
39. Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: Advances in neural information processing systems. (2017) 465–476