Evolutionary Latent Space Exploration of Generative Adversarial Networks

Paulo Fernandes, João Correia, and Penousal Machado

CISUC, Department of Informatics Engineering
University of Coimbra
pcastillo@student.dei.uc.pt, jncor@dei.uc.pt, machado@dei.uc.pt

Abstract. Generative Adversarial Networkss (GANs) have gained popularity over the years, presenting state-of-the-art results in the generation of samples that follow the distribution of the input training dataset. While research is being done to make GANs more reliable and able to generate better samples, the exploration of its latent space is not given as much attention. The latent space is unique for each model and is, ultimately, what determines the output from the generator. Usually, a random sample vector is taken from the latent space without regard to which output it produces through the generator. In this paper, we move towards an approach for the generation of latent vectors and traversing the latent space with pre-determined criteria, using different approaches. We focus on the generation of sets of diverse examples by searching in the latent space using Genetic Algorithms and Map Elites. A set of experiments are performed and analysed, comparing the implemented approaches with the traditional approach.

Keywords: Generative Adversarial Network, Evolutionary Computation, Latent Space Exploration.

1 Introduction

GANs have been presenting state-of-the-art results in the generation of samples that follow the distribution of the input training dataset [1]. In general, this model of adversarial learning works by having a generator and a discriminator training together and competing against each other in a min-max game. The discriminator is trained with real data as well as fake data created by the generator from latent space. The generator evolves from the feedback given by the discriminator on the generated data Figure 1. Although able to produce results with high quality, GANs are often really hard to train, requiring a lot of fine-tuning through trial and error.

While a lot of research is being made in order to make GANs more reliable and able to generate better samples, the exploration of its latent space does not have as much attention. The latent space is unique for each model and is, ultimately, what determines the output from the generator since, in simple terms, we have $g(z)=x'$, being x' the output of a latent vector z through the

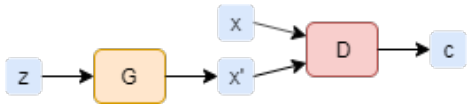

Fig. 1. Base model of a Generative Adversarial Network. G - generator; D - discriminator; z - latent vector; x - real samples; x' - generated samples

generator g. Typically we use the trained model to draw random samples from the latent space without any particular criteria.

In this work, we move towards the search and exploration of solutions from the latent space according to pre-determined criteria. The overall idea is to enable the search and generation of sets of latent vectors that accomplish a certain objective. We will start from exploring approaches that enable us to draw samples and traverse the latent space moved by a certain objective function. We will use Genetic Algorithms (GAs) and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) to assist on such task. Thus, the contributions of this paper are the following: (i) modelling of the latent space exploration as a search problem, enabling the use Genetic Algorithms and Map Elites (ii) a generalised approach to generate sets of images from a GAN according to with different objectives, e.g. generate diverse sets of images; (iii) a comparison analysis of the implemented latent space search algorithms with the conventional approach. Without lacking generalisation, we apply the ideas of this paper to the image domain, using Deep Convolutional Generative Adversarial Networkss (DCGANs) [2].

The remaining of the paper goes as follows: in the next Section, we cover approaches relate to this work (Section 2). In Section 3, we describe our approaches to this search problem. We describe the experimental setup in Section 4 and analyse and discuss the results in Section 5. In Section 6 we draw overall conclusions.

2 Related Work

GANs are generative models that are trained through a face-off between a generator and a discriminator, mostly used to train a generator that can produce realistic images. In order to generate an image, the generator is usually given a random noise vector, a high dimensional vector that, in training, is randomly sampled from a distribution, for example, a gaussian distribution, called the prior. The high dimensional space from which images are created is called latent space. Some work has already been made in this area, and not only with GANs, which is the type of model that we are going to work within this work. For instance, latent space exploration was performed in generative models using Kernel Principal Component Analysis (KPCA) [3], showing navigation through image features and novelty detection; but also in Variational Autoencoders (VAE), in, for example, mapping genes into a lower-dimensional space in order uncover underlying gene expression features in cases of tumour or cancer [4].

There are different ways that we can explore generative models latent space. White, T. shows three different arithmetical ways we can use to navigate in it [5]. The first, interpolation, is used to disclose the path between 2 samples in the space. Usually, a linear interpolation is used, but White suggests a new way, the spherical linear interpolation - slerp - which takes into account its prior distribution used for sampling noise. Analogy, the second approach, can be simplified as "King – Man + Woman = Queen", meaning that it is possible to perform operations between images to form new images with generally predictable results. Finally, Manifold Interpolated Neighbor Embedding (MINE), is an exploration model which makes use of nearest neighbours and interpolation to construct a manifold of the space. Also worth noting is that the authors showed that by combining generative models with labelled data, attribute vectors can be computed using simple arithmetic, like, for example, a smile vector which, by traversing uncovers several states of smiling in an image.

As for the use of Evolutionary Computation (EC) in order to evolve images, some works can be mentioned. For example, evolving master print templates [6] that, like a master key, could be able to open multiple fingerprints closed locks. In their work, Roy et al. compared four different Evolutionary Algorithms (EA), namely Hill-Climbing, Covariance Matrix Adaptation Evolution Strategy, Differential Evolution and Particle Swarm Optimization to evolve Synthetic MasterPrints according to the metric proposed by them, the Modified Marginal Success Rate. The samples were generated from two datasets, namely Authentec AES3400, with latter algorithm getting the best results, and FVC 2002 DB1-A, for which the second approach had the most success. Moreover, and with a two-stage workflow similar to what we implemented in this paper which includes first the unsupervised training of GANs and second the evolution of latent space, there are two works. One that implements Interactive Evolutionary Computation [7] for image generation and the other, in the topic of video-games that uses GANs and latent space evolution to learn and improve Mario Levels [8] using Covariance Matrix Adaptation Evolution Strategy.

There is also a new approach to generative models which was inspired by GANs, the Generative Latent Optimization [9]. This approach takes away the adversarial discriminator and replaces it with simple reconstruction losses where the focus is to evolve the latent space to match the one learnable noise vector to each one of the images in the training dataset.

3 The Approach

Our goal in this paper is to use approaches that can explore the latent space and create a set of latent vectors towards an objective. In particular, we set the objective to: find a set of images that can maximize a diversity measure. We pursuit the objective by exploring the latent space of GAN models using EC. Therefore, this experiment was separated into two main parts:

– Training of the GANs - develop a generative model that can produce images which follow the distribution of a certain training input

– Exploration of the latent space via EC - navigate the latent space of the generative model in order to find a possible solution for our problem.

The exploration of the latent space and set of latent vectors has the potential to promote the generation of samples according to pre-determined criteria to solve and adapt to other problems. It could be used during the training of the GAN, to have a few latent vectors generated that maximize the loss of the model, instead of randomly generating all of them. It could be used to generate samples of a particular type. Some metrics of evaluation of GANs must draw samples from the latent space, generate the samples and test them on another model [10]– "what if we guide that generation?" We can use this approach to spot problems on the training of models.

Also, this problem was applied to 3 distinct sets of images in order to comprehend how well our approach would work for different situations:

– handwritten digits – MNIST
– clothing – Fashion – MNIST
– faces – Facity

3.1 Model definition

We are using a type of GANs suitable for generating images, the DCGAN [2]. They make use of deep convolutional layers to better explore space correlation in images, producing more realistic images. For every GAN, the latent space, which is the input for all generators, is an array of floats of shape 1×100. The generators used in both MNIST datasets are built with the following model, generating output with shape $28 \times 28 \times 1$:

```
Dense((7, 7, 128), activation="relu")
UpSampling2D()
Conv2D(128, kernel_size=3, padding="same"))
BatchNormalization(momentum=0.8))
Activation("relu"))
UpSampling2D())
Conv2D(64, kernel_size=3, padding="same"))
BatchNormalization(momentum=0.8))
Activation("relu"))
Conv2D(1, kernel_size=3, padding="same"))
Activation("tanh"))
```

The generator used with the facity dataset, has an output shape of $128 \times 128 \times 3$ and is built with the following model:

```
Dense((8, 8, 128), activation="relu")
UpSampling2D()
Conv2D(256, kernel_size=3, padding="same")
BatchNormalization(momentum=0.8)
Activation("relu")
```

```
UpSampling2D()
Conv2D(128, kernel_size=3, padding="same")
BatchNormalization(momentum=0.8)
Activation("relu")
UpSampling2D()
Conv2D(64, kernel_size=3, padding="same")
BatchNormalization(momentum=0.8)
Activation("relu")
Conv2D(3, kernel_size=3, padding="same")
Activation("tanh")
```

The discriminators used for each GAN, all follow the same model, having an input shape defined by the output of the generators.

```
Conv2D(32, kernel_size=3, strides=2, padding="same")
LeakyReLU(alpha=0.2)
Dropout(0.25)
Conv2D(64, kernel_size=3, strides=2, padding="same")
ZeroPadding2D(padding=((0, 1), (0, 1)))
BatchNormalization(momentum=0.8)
LeakyReLU(alpha=0.2)
Dropout(0.25)
Conv2D(128, kernel_size=3, strides=2, padding="same")
BatchNormalization(momentum=0.8))
LeakyReLU(alpha=0.2))
Dropout(0.25))
Conv2D(256, kernel_size=3, strides=1, padding="same")
BatchNormalization(momentum=0.8)
LeakyReLU(alpha=0.2)
Dropout(0.25)
Flatten()
Dense(1, activation='sigmoid')
```

After building both the model of the discriminator and of the generator, they are combined into a single, combined, model which receives the same input as the generator and has the output of the discriminator.

3.2 GAN Training

The adversarial neural networks are trained by having the discriminator learn to distinguish the real samples, which come from the input datasets, from the fake samples, that are generated by the generator, and by having the generator learn to produce images that successfully trick the discriminator into classifying them as real samples [1].

The input datasets used Figure 2, were the following:

- MNIST − dataset of handwritten digit with 70000 (60000 for training and 10000 for testing) 28x28x1 (black and white) images, divided into 10 classes[11].

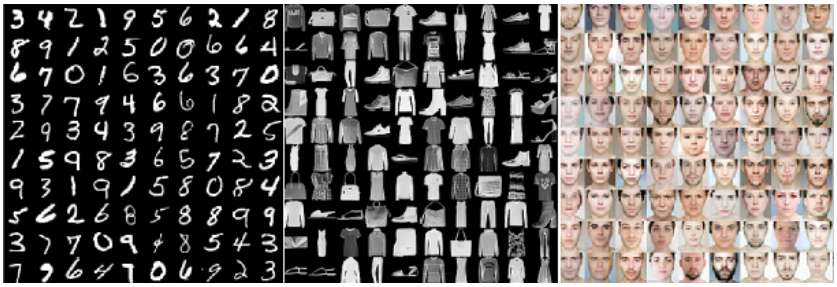

Fig. 2. Samples from the datasets used for training. From left to right: MNIST, Fashion-MNIST, Facity

- Fashion MNIST − dataset of pieces of clothing with 70000 (60000 for training and 10000 for testing) 28x28x1 (black and white) images, divided into 10 classes [12].
- Facity − dataset of faces with 4024 128x128x3 (rgb) images, not divided by training or classes.

3.3 Latent Space Exploration

In order to traverse the latent space, we require to generate sets of individuals. To pursuit our objectives, we used the following approaches:

- Random Sampling (RS)
- GA
- MAP-Elites

The three approaches were thought to evolve set of individuals, but they vary in the way they promote change along with iterations.

In the RS approaches a completely new random set of individuals is created at the beginning of each generation and evaluated by an evaluation function, as shown in 1.

Algorithm 1 Random Sampling

```
1: procedure randomSampling(iterations, popsize, objective, fitnessfunc)
2:     for i ← 1 to iterations do
3:         population = randomPopulation(popsize)
4:         population = fitnessfunc(population)
5:         best = best(population, objective)
       return best(population)
```

The GA is an EC approach where we start with a randomly generated population of individuals, but a new population is created trough variation operators

Algorithm 2 Genetic Algorithm

```
1: procedure geneticAlgorithm(generations, popsize, problemargs)
2:     population = randomPopulation(popsize)
3:     for i ← 1 to generations do
4:         population = variationOperators(popsize, problemargs)
5:         population = problemargs.fitnessfunc(population)
6:         population = problemargs.sort(objective)
       return best(population)
```

(mutation and crossover) which are applied to the individuals of the old population evaluated by a certain fitness function (refer to algorithm 2).

The last algorithm, the MAP-Elites, works a little bit differently. It is an illumination algorithm, and it was made for exploring the search space of solutions as much as possible [13]. A map of plausible combinations of feature dimensions for the individual's phenotype, which is previously defined, is maintained throughout the training. The algorithm starts by generating a random number of individuals and placing them on the feature map. Afterwards, MAP-Elites runs by iterations. At each iteration, a single new individual is created from applying variation operators to individuals already placed in the map. Each new individual is then evaluated and placed in the map according to its features, though in each cell of the map, only the best is kept according to the fitness function.

Algorithm 3 MAP-Elites

```
1: procedure mapElites(iterations, initpopsize, map, problemargs)
2:     for i ← 1 to initpopsize do
3:         newind = randomIndividual(problemargs)
4:         placeInMap(newind, map)
5:     for i ← 1 to iterations do
6:         newind = variationOperators(map, problemargs)
7:         placeInMap(newind, map)
       return best(map)
```

Individuals, Initialization For this problem, we the decided to work with sets of 50 images, meaning that, since we are working with a latent dimension of size 100, each individual has a genotype of size 5000 that is represented in an 1×5000 array. In order to maintain consistency, The initialization of individuals follows the same Gaussian distribution as the prior used to generate fake images in the training of the generative models.

Evaluation, Metrics and Variation Operators All algorithms use the same method to evaluate the fitness of the individuals, an average similarity function.

Through the use of the specified GAN generator, the genotype of the individuals is transformed into a set of images. Afterwards the images are compared to one another using a similarity metric, averaging between all values at the end. Since we want the most diversity possible, we have modelled the problem for minimization. In terms of the feature dimensions for MAP-Elites, both the average similarity and max similarity are used map individuals.

To measure similarity, 2 distinct metrics were used, namely Root-Mean-Squared Error (RMSE) and Normalized Cross-Correlation (NCC) [14]. The RMSE metric is calculated as

$$\text{RMSE} = 1 - \sqrt{\frac{\sum((A-B) \odot (A-B))}{\text{size}}} \tag{1}$$

while NCC is calculated as

$$\text{NCC} = \frac{\sum(A-B) \odot (A-B)}{\sqrt{(\sum A \odot A) \times (\sum B \odot B)}} \tag{2}$$

Where A and B are images, size is a function that measures the size of the images and $\odot$ is the Hadamard product. We selected these metrics for their fast calculation time and to observe the impact of both since they work in different ways. With RMSE we have a strict and direct pixel by pixel comparison whereas with NCC we are looking for certain contrast in the pixel intensities. These are options out of a number of different similarity metrics [14], but covering all of them is out of the scope of .

Two variation operators were used for the experiments: crossover and mutation. In the crossover, the two individuals are chosen from the set of available individuals using tournament selection in the GA and random choice in MAP-Elites. In both algorithms, the algorithm used is the uniform crossover [15].

In the case of the mutation operator, it was used a random reset mutation, where each gene has a probability of being mutated. The mutation resets the selected genes with completely new values that are taken from the same Gaussian distribution used in the initialization of the individuals.

4 Experimental Setup

The experimental setup was thought to analyse how different algorithms and distinct metrics affect, for different situations, the end result in terms of diversity, which is the main goal. Moreover, by analysing the observable characteristics of the sets of images obtained, we also wanted to see if the diversity measured by an algorithm, is consistent to what we, as humans, perceive as a diverse.

To perform the experiments three generative models were trained, 1 for each dataset to be used Figure 3. Most of the parameters are the same Table 1, the only thing that changes is the number of epochs of training: (i) MNIST - 200; (ii) Fashion-MNIST - 800; (iii) Facity - 1000.

So that the results would be comparable, we keep the conditions of the algorithms as close as possible Table 2, for instance, performing a number of

Table 1. GAN Parameters

Parameter	Setting
optimizer	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
batch size	32
loss function	Binary Cross-Entropy
noise distribution	N(0,1)

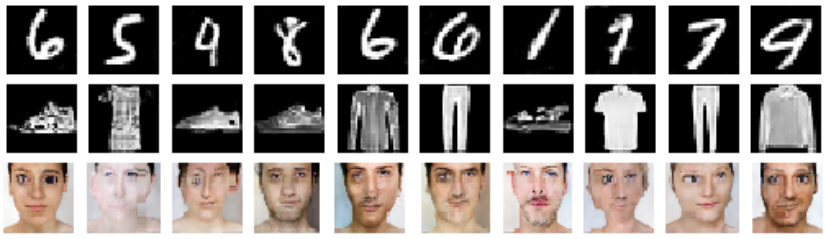

Fig. 3. Random sample of images generated from the 3 generative models

iterations on MAP-Elites that would produce the same number of evaluations as in the GA, maintaing the same number of generated images and attempts to reach the diverse set of images.

5 Experimental Results

In this section, we analyse the results from the experiments in terms of optimisation of the fitness function and the visual outputs from each algorithm. In order to analyse the results for comparison, we provide graphics with aggregated

Table 2. GA and MAP-Elites Parameters

Parameter	Setting
Population size	50 in RS/GA, initial$\leq$50 in MAP-Elites
Number of generations	500 in GA, 25K in RS/MAP-Elites
Genotype length	50$\times$ size of latent space
Elite size	1
Tournament size	3
Crossover operator	uniform crossover
Crossover rate	0.7 in GA/MAP-Elites
Mutation operator	gene replacement
Mutation rate per gene	0.02 in GA/MAP-Elites
feature dimensions	avg similarity : bins=[0:0.01:1], max similarity bins=[0:0.01:1] in MAP-Elites

the values of evaluations. We group the values for Random Sampling and Map Elites in iterations, that corresponds to 50 evaluations. This way, it is possible to compare with the information of the GA. Basically, one iteration is equivalent to a generation on the GA, which in turn is equivalent to 50 evaluations on Random Sampling and Map Elites algorithms. We also separate the analysis of image diversity in two groups based on the similarity metric: RMSE and NCC.

In figure 4, we can observe the results of the different algorithms across the iterations. It is clear that the GA is able to optimise the fitness function. The same does not happen with MAP-Elites though, we get little improvement compared to the random sampling, and that does not change when working with different datasets, the difference between the actually gets narrower the more complex is the type of images.

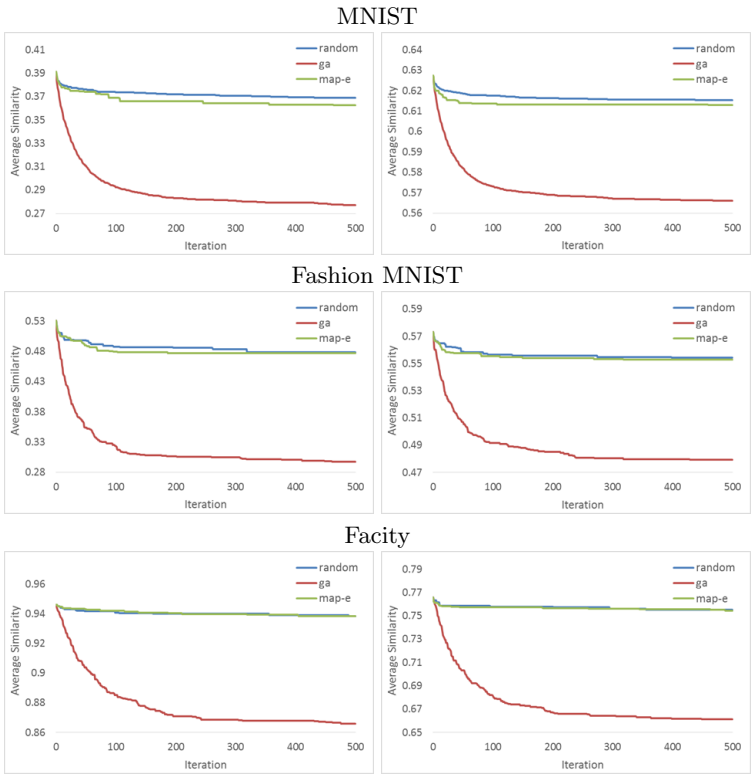

Fig. 4. Average Maximum values across iterations for the different datasets and similarity metrics: on the left NCC and on the right RMSE. The values are averages of 10 runs.

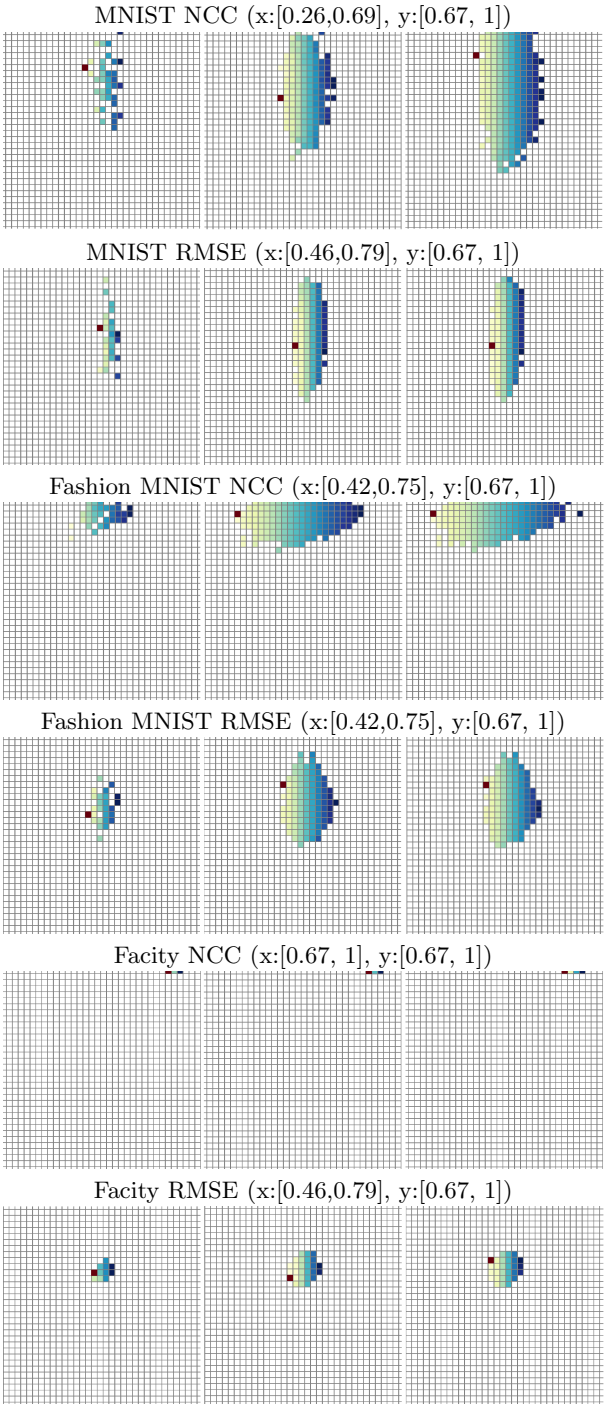

Fig. 5. Heatmap of the Map constructed with MAP-Elites for each dataset and similarity metric. Images correspond to 0th, 12000th and 25000th iterations of the algorithm. The color represent fitness, yellow for better fitness and blue for worse, while the red represents the best individual. The x axis corresponds to the average similarity and the y axis to the maximum similarity.

When analysing the Map Elites algorithm, we are also concerned with the mapping of the individuals. As such, we analysed the heatmaps of the algorithm across iterations for the datasets and metrics, as presented in figure 5, to understand how much of the space was being explored by the algorithm, which helps us to understand the distribution of samples along with the iterations.

We can observe that space exploration is limited and that it generally seems to take increasingly more time to expand and, therefore, more time to find better solutions. It is noticeable the existence of a cluster of solutions in this approach and that the expansion is easier for less fit (dark blue) zones, which corroborates the evolution graphs. Here we clearly see how much MAP-Elites struggles to find better solutions when the mapping is not favourable to the problem. With the facity dataset and the NCC metric, the random initialisation values were clustered into a very small area, which made it really difficult for the algorithm to expand. Even the variation operations caused changes so small that they still fall in the same area.

The results suggest that this algorithm is not suitable for this type of problem. However, it could be related to the selection of feature descriptor and algorithm's parametrisation. The mapping functions tend to be simplistic and become a bottleneck that does not allow the ideal exploration of the search space and expansion of the mapped area, something that we will further investigate.

In terms of visuals the outputs from Figure 6 and Figure 7 showcase the best set of images that maximize the pre-determined criteria. For this last analysis, we only focused on the NCC metric and RMSE metric, respectively. It is noticeable the differences between the different approaches and metrics. In general, every approach, for each metric, ended up having a different set of images at the end.

Between the NCC and RMSE results, one aspect is noticeable, and it showcases the particularity of each similarity metric. In the RMSE, we observe that it tended to pick a set of variables that generated images which minimize the overlap of elements and with a dissimilar background (facity). The NCC, on the other hand, tended to promote contrast between the different images. This is a clear example of the success of the approach of searching and achieving the pre-determined objective. Among the different approaches, we can observe that the GA was able to generate the most visually diverse set of images.

6 Conclusions

This work presents multiple methods that enable the user to explore the latent space with a pre-determined objective. We have implemented Genetic Algorithms and Map Elites and compared with the traditional approach of random sampling from the latent space. Some of the results obtained with were unexpected but suggested that we are able to generate diverse sets of latent variables that translate into samples that correspond to certain criteria. The conducted experiments, in the image domain, with different datasets, point out that it is possible to apply our approach to GANs of different types of datasets, ranging from grayscale to colour images. The overall results worked as a proof of con-

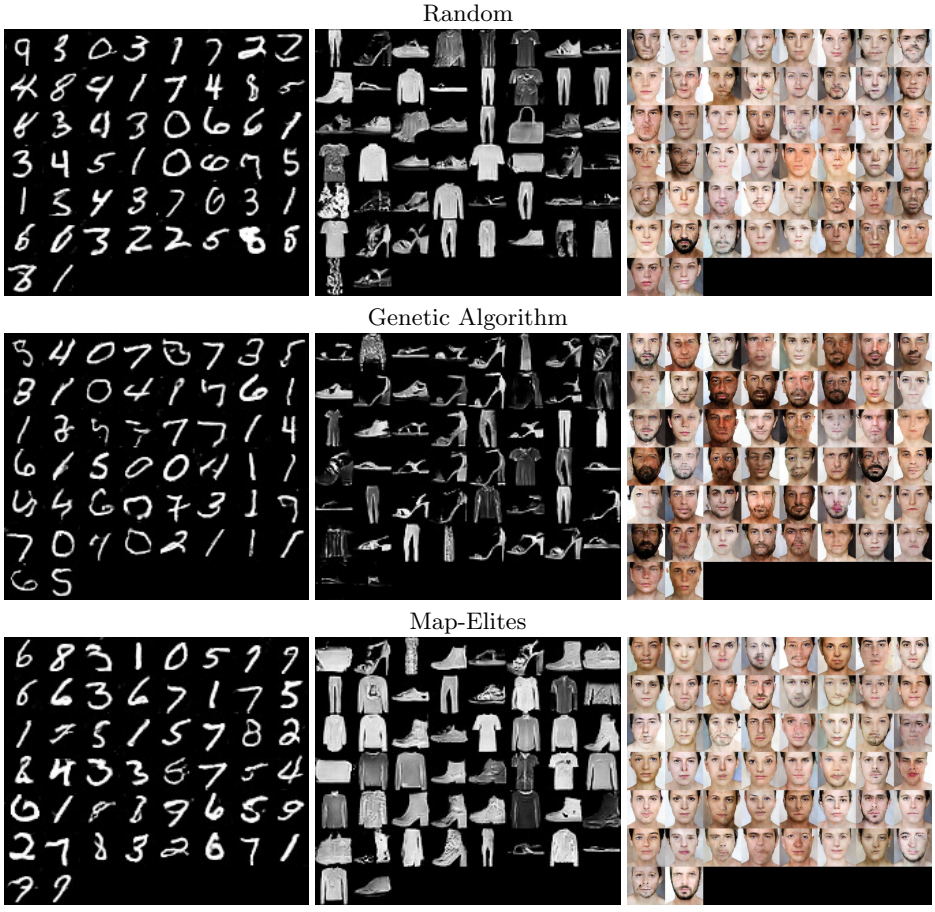

Fig. 6. The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the NCC metric.

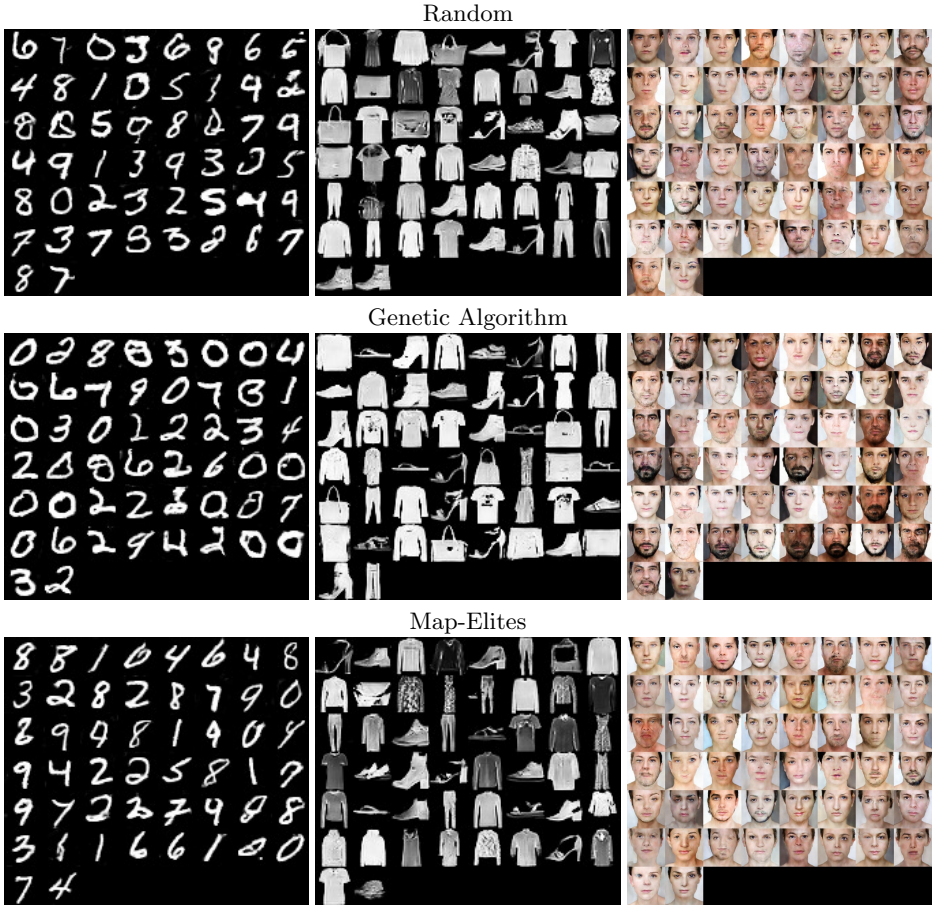

Fig. 7. The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the RMSE metric.

cept that is possible to guide the generation of latent variables towards certain criteria.

Future work could include other strategies and objectives for navigation, such as the usage of different metrics; fine-tuning of parameters, application of different criteria for illuminating the latent space in map-elites, which may include working with more than 2, usage of multiple seeds per set as well as adding a pre-processing layer that could help focus on the regions of interest in the images (for example the removal of the background). Moreover, the generation of groups of images uncovers problems of generalization. Therefore research could be done in order to explore reinforcement of training datasets.

7 Acknowledgments

This work is partially supported by national funds through the Foundation for Science and Technology (FCT), Portugal, within the scope of the project UID/CEC/00326/2019 and it is based upon work from COST Action CA15140: Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO).

References

1. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS. pp. 2672–2680 (2014)
2. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015)
3. Winant, D., Schreurs, J., Suykens, J.: Latent space exploration using generative kernel pca. In: Proc. of the 28th Belgian Dutch Conference on Machine Learning (Benelearn2019). BNAIC/Benelearn (2019)
4. Way, G.P., Greene, C.S.: Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders, pp. 80–91 (2018)
5. White, T.: Sampling generative networks. CoRR abs/1609.04468 (2016), `http://arxiv.org/abs/1609.04468`
6. Roy, A., Memon, N., Togelius, J., Ross, A.: Evolutionary methods for generating synthetic masterprint templates: Dictionary attack in fingerprint recognition. In: 2018 International Conference on Biometrics (ICB). pp. 39–46 (Feb 2018)
7. Bontrager, P., Lin, W., Togelius, J., Risi, S.: Deep interactive evolution. In: Liapis, A., Romero Cardalda, J.J., Ekárt, A. (eds.) Computational Intelligence in Music, Sound, Art and Design. pp. 267–282. Springer International Publishing, Cham (2018)
8. 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 (2018)
9. Bojanowski, P., Joulin, A., Lopez-Paz, D., Szlam, A.: Optimizing the latent space of generative networks (2017)
10. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. pp. 6626–6637 (2017)

11. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010)
12. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
13. Mouret, J.B., Clune, J.: Illuminating search spaces by mapping elites (2015)
14. Goshtasby, A.A.: Similarity and dissimilarity measures. In: Image registration, pp. 7–66. Springer (2012)
15. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series, Springer Berlin Heidelberg, Berlin, Heidelberg (2003)