IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019

Generate Desired Images from Trained Generative Adversarial Networks

Ming Li*, Rui Xi*, Beier Chen†, Mengshu Hou*, Daibo Liu*, Lei Guo*,
*School of Computer Science and Engineering
University of Electronic Science and Technology of China
†Ohio State University, Columbus
Email: uestcliming@gmail.com, ruix.ryan@gmail.com, chen.8319@osu.edu, mshou@uestc.edu.cn,
dbliu.sky@gmail.com, leiguo@uestc.edu.cn

***Abstract*—The emerging of Generative Adversarial Networks (GANs) gives rise to a significant improvement in image generation. However, a controllable way of synthesizing images with specific characteristics still is a challenging issue. Many existing methods are not efficient enough that require additional information and pre-designed attributes, and are with much more human intervention. In this paper, we propose GAGAN, an extension method to the Generative Adversarial Network, that is the first work to generate specific images from a trained GAN model. To control the characteristics of images, a DNA pool of the trained GAN model is introduced and evolved by a genetic algorithm (GA). Then, with the DNA pool, GAGAN can generate the corresponding latent vector (DNA) of target images. Furthermore, GAGAN can synthesize images containing a single specific characteristic or multiple specific attributes (including AND and OR relation). Moreover, several fitness evaluation strategies are also proposed to make GAGAN flexible to control the target characteristics. Experiments on CelebA and MNIST are conducted, and results show that the proposed method is feasible and effective in specific image generation problem.**

I. Introduction

In recent years, image generation has become a hot topic of computer vision and has been widely used in many scenarios, such as advertising, painting, data augmentation. In order to generate realistic images, many models have been presented using traditional machine learning, including Deep Belief Networks (DBNs) [1], Variational Auto-encoder (VAE) [2], and auto-regressive [3]. Unfortunately, these models are very complex, and fail to generate high-resolution images. Benefiting from the great performance of Deep Learning, many researches are becoming to adopt Generative Adversarial Network(GAN) [4] to get higher quality images.

Since GAN was first proposed by Goodfellow in 2014, a variety of variants has been proposed, such as DCGAN [5], LAPGAN [6], ProgressiveGAN [7]. These models maybe could generate much more realistic and plausible images, nevertheless, they still can't satisfy the needs to flexibly control the attributes of generated images. Therefore, in order to specify the range of generated images, researchers present many variants of GAN like C-GAN [8], ACGAN [9] and Info-GAN [10]. These models introduce the label of images or latent code learned by unsupervised learning to the latent vector.

However, these models bring in additional information to the input vector, rather than using the latent vector directly, this will result in some serious problems. For example, the generated images must be within a predefined range, and all the target images or desired attributes should be defined in advance. Otherwise, it is needed to train another GAN model. And it is much more difficult for people to take every condition into account. In addition, most of these methods couldn't control the attributes of generated images independently. Once the number of attributes is large, an exploration of attributes combination may exacerbate this drawback. What's more, imposing additional information constraint simply on the latent vector may result in information loss, and may even limit the generation ability of GAN.

As we all know, a well-trained GAN model could generate a variety of different images without any additional information. Therefore, given a specific latent vector as input, it is possible to generate specific images only rely on the information hidden in the latent vector of the trained GAN. Inspired by the biological phenomenon that the DNA controls characteristics, we assume that the latent vector is the DNA of a GAN model, and its generator model can be regarded as an explainer of DNA. Then, the key issue of generating specific images is how to find the suitable DNAs of targets. However, the relation between the gene and the attributes is highly complex, and it's hard to separate the target information from the latent vector. Therefore, instead of decoupling the gene, we will focus on finding an end-to-end method to generate suitable latent vectors (DNAs).

In this paper, we present a GAGAN method that can generate suitable DNA of a trained GAN model and makes it possible to generate target images directly from the latent vector. Given a trained GAN model (generator), in order to obtain the suitable DNA of specific targets, we first introduce a randomized DNA pool, which has the same (normalize) distribution as the trained GAN model. Then, we choose to use Genetic Algorithm (GA) for optimizing the DNA pool. According to Darwinian evolution, along with the process of selection and perturbation of DNA, using the optimized DNA pool can control specific features in the generated results of a trained GAN. To be more specific, we need a pre-trained recognition model to evaluate the DNA's fitness, a value of

978-1-7281-2009-6/$31.00 paper N-19141.pdf

loss between the corresponding images of the gene and the target class, which is used for DNA selection. With the fitness of individual DNA, we can evolute the DNA pool by genetic heredity, variation, and elimination process, and then generate accurate targets. Furthermore, we can also generate new DNAs by crossover and mutation operation. For flexibility, we separate the generation problems into three classes: single attribute/characteristic, AND relation of multiple attributes and OR relation, and each is with a corresponding fitness evaluating strategy.

In comparison with traditional methods, GAGAN does not need to impose any additional information, and the pre-define is no longer required. When a new target is added, we only need to conduct GA several iterations to get the target. Benefiting from the genetic algorithm, GAGAN could flexibly control the combination of characteristics by modifying the fitness function. Moreover, even the GAN model is not well trained, GAGAN could still generate visually pleasing images with distinct target characteristics. Besides, it is noteworthy that GAGAN algorithm converges very fast,mainly because gradient computing and back propagation are removed, and the cost is much lower than retraining the whole GAN model. Most important of all, our proposed method is not just to get a single latent vector, but to get the DNA pool via the genetic algorithm. As mentioned, we can produce many DNAs with the DNA pool via crossover, to generate images with desired characteristics, so the number of results is not fixed once the evolution stops and this will make GAGAN more practical.

To sum up, this paper includes the following contributions:

- To the best of our knowledge, there is no previous work that can generate specific target images from a trained GAN model. By properly combining the Genetic Algorithm with GAN, our GAGAN could generate suitable latent vectors (DNAs) and use them to generate specific attributes images directly without any additional information or pre-design.
- To flexibly control the target characteristics, we formulate three types of generation problems and provide solutions for them by modifyng the fitness evaluation strategy.
- To validate the effectiveness of our proposed GAGAN, we conduct experiments on MNIST and CelebA. Results demonstrate that GAGAN could handle different types of target generation problem. Moreover, our methods could get visually pleasing images from not well trained GAN model.

II. Related work

Genetic Algorithm(GAs) [11] is a type of evolutionary algorithm using computational analogs of biological mutation and crossover to generate variants, and modeling a Darwinian using the fitness of variants. Variants with high fitness are selected for continued evolution. There is a legion of enhancements of GAs [12] [13] [14]. Genetic algorithms has been shown effective in many tasks. Recently, researchers use genetic algorithm to handle problems in Deep Learning such as training deep neural networks for reinforcement learning [15] and adversarial attacking [16]. But, there is no application in image generation problem with GAN.

Generative Adversarial Networks(GAN) [4] was proposed to train generative models via an adversarial process, it outperforms a huge superiority than tradition generation methods in quality of generated samples and the training progress. To generate high-quality images with high resolution, many variants of GAN has been proposed. For instance, DCGAN [5] firstly introduce CNN and batch normalization to GAN and generate images by using deconvolutions. Some works improve generating quality by sequence the GAN generating progress, LAPGAN [6] leverage Laplacian Pyramid to achieve this. Very recently, ProgressGAN [7] grow both generator and discriminator progressively by adding new layers, this allows us to produce images of unprecedented quality.

To control the results of the generator, many researchers have studied many transformed GAN models in conditional ways, by adding additional information to the latent vector of generator. CGAN [8], ACGAN [9] adopt label as information to control the generating process. StackGAN [17] studied the problem of generating realistic images, guided by text description. InfoGAN [10] decomposes the input lantent vector of GAN into information(latent representation code) and incompressible noise by maximizing the mutual information. Some impose different conditions for specific application, such as inpainting [18], video prediction [19], $etc.$ These GANs are always conditioned on labels or information extracted from pre-trained model, which would harm the generation ability, and the desirable attributes need to be defined in advance.

Image attribute editing seems to be a feasible way to get target images. Pix2Pix [20] first introduced a common framework, which could transfer a given image into the target image domain. Subsequently, inspired by dual learning paradigm in natural language translation (NLT) [21], DiscoGAN [22], DualGAN [23], CycleGAN [24] have been proposed to remove the paired data requirement of Pix2Pix framework. More recently, several specific methods were presented for facial attribute editing, such as ExprGAN [25], AttGAN [26]. These methods learn a latent representation of attributes and modify attributes in facial images with it. However, these methods can only modifying an existing image based on attributes or other forms of semantic information rather than generating the images with target characteristic.

III. Method

In this section, we will formally present our proposed method, GAGAN. Firstly, we give a brief review of the genetic algorithm and original GAN along with its optimization policy. Subsequently, on the basis of them, we give a detailed introduction of GAGAN. At last, we will describe how to define the fitness objective function to meet different generation requirements.

A. Background

Generative Adversarial Nets [4] were firstly proposed by Good-fellow $et\ all$ in 2014 to solve generation problems

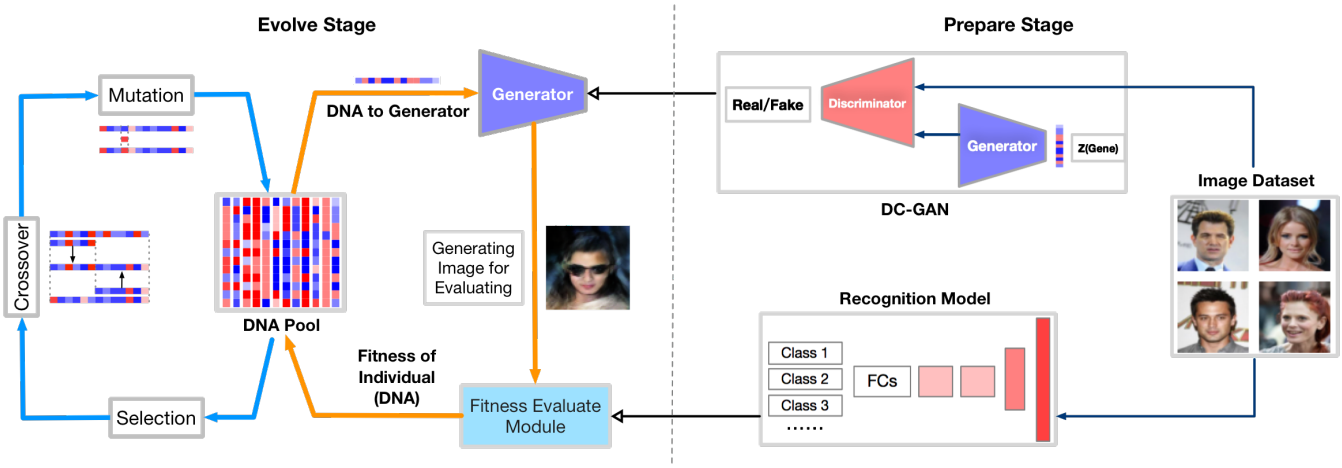


Fig. 1. The overall architecture of the proposed GAGAN method.

via an adversarial process. It consists of a generative model G and a discriminative model D. Assuming that data x is an image drawn from dataset distribution, p_{data}, and z is a random vector in $\mathbb{R}^d$. During the period of training, the generator G takes z as input and maps it into an image data space $G(z)$, whose distribution can be denoted as P_G. And, the discriminator D will distinguish real images from the image distribution P_{data} and synthetic images $G(z)$ from P_G. However, in GAN, the generator G and the discriminator D can seem to be two players of a game, that G tries its utmost to confuse D. Therefore, in order to train G and D simultaneously, a min-max optimization game is employed to optimize the following objective function,

$$\min_G \max_D V(G,D) = E_{x\sim P_{data(x)}}[\log D(x)] + E_{z\sim P_z}[\log(1 - D(G(z)))] \quad (1)$$

where x is the real images drawn from data distribution P_{data}, z is the noises input drawn from a prior distribution P_z.

The genetic algorithm (GA) evolves a population $\mathbb{P}$ of N individuals by imitating Darwinian evolution. Here, individuals are n-dimensional vectors θ. At every generation, each θ is evaluated by the fitness function F to get the fitness score $F(\theta)$. Individuals with high fitness score are selected as the parents of next generation. To produce next generation individuals, the parents are selected uniformly at random with placement and start crossover, which is exchanging parameters among parents to produce an offspring. And then, the offspring mutates by applying Gussian noise to the parameter to create new variants. Subsequently, the fitness of new individuals are calculated by F. According to the fitness, the top N individuals in the new produced and the previous generation make up for the next generation. The process repeats for several generations or until some other stopping conditions are triggered.

B. GAGAN

In order to generate specific images from a trained GAN model, instead of extracting information from the latent vector, we propose a GA-based GAN method, namely GAGAN, to generate corresponding latent vector, which can be used to generate images with desired characteristics. To combine GA and GANs, and to make it feasible to generate specific images, we regard the latent vector as the DNA of GAN, this is based on the fact that different images have a corresponding latent vector, which is exactly similar as a DNA controls the characteristics of organisms. Meanwhile, the generator can seem to be an interpreter of DNA, and transfers DNA into images. Through a fitness evaluation, the images with distinct target characteristics are valued high fitness and will be selected. In addition, according to the fitness function, GA is applied to evolve the DNA.

As illustrated in Fig. 1, GAGAN is comprised of two stages: the prepare stage and the evolve stage. From it, we can see that a generator, denoted as $G_{trained}$, is needed to be well trained in advance. For simplicity, we choose DCGAN[5] in this work. Additionally, a recognition model with high accuracy, denoted as R, is also required. It is used in the fitness evaluate module to indentify characteristics and generate the possibility that an image has desired characteristics. Note that, both the trained generator and recognition model can be obtained from the existing trained models.

In the evolve stage, GAGAN consist of three main components, a DNA pool $Pool$, a generator $G_{trained}$ and a fitness evaluate module F. For the $Pool$, we initiate N latent vectors θ (DNA), sampled from a Gaussian distribution $N(0,1)$ as same as inputs of the trained generator. Through the trained generator, each θ in the DNA pool $Pool$ will be mapped into an image $G_{trained}(\theta)$, which will be passed to the fitness evaluate module as inputs. As mentioned above, in the fitness evaluate module, a recognition network will be triggered to value the fitness of input images. Note that, according to different generation problems, the fitness evaluate module can be classified into three categories, and we will give a detailed description in the below Section. Given the fitness score, genetic algorithm could select the DNA θ with higher score as parents. Subsequently, the chosen parents exchange parameters to produce new DNA θ_{new} by crossover and mutation at a

given mutation rate. Then, mixing with original DNAs, the top-N fitness DNAs will be chosen to make up a new DNA pool. The whole process won't stop early until the percentage of desired images reaches a predefined level L_{traget}, which is determined empirically. And the overall evolution procedure of GAGAN is also presented in Algorithm.1.

After evolution, we get a DNA pool, in which the DNA θ has a high possibility to generate desired images via the trained generator. In addition, the DNA pool should be stored, in case of the same target image generation problem. It is worth noting that, as the evolution goes, the percentage of desired images increases, while the diversity of generated images decreases. In summary, there is a trade-off between the accuracy and diversity. Thus, the stopping criterion L_{target} needs to be set according to different requirements.

Algorithm 1: *GAGAN* procedure

Require: Pre-trained generator G;
Stoping level L_{target};
DNA pool $Pool$
Number of individuals in DNA pool N Target set T;
for number of generations **do**
  for DNA θ in $Pool$ **do**
    Generate image $G_{trained}(\theta)$
    Calculate fitness $F(G_{trained}, T)$
  end for
  for N **do**
    Select parents from $Pool$
    Create an offspring θ_{new} via crossover and mutation
    Calculate fitness $F(G_{trained}(\theta_{new}), T)$
    Put θ_{new} into $Pool_{new}$
  end for
  Select top N fitness from $Pool$ and $Pool_{new}$ as the next generation $Pool$
  Calculate the generation accuracy $Accuracy$
  if $Accuracy$ equals L_{target} **then**
    End the evolution
  end if
end for
Generate or get latent vector θ from $Pool$

C. Fitness function

As described above, the fitness evaluation function, providing reliable fitness score, is another key issue in our proposed GAGAN. In this section, we separate the image generation problems into three basic categories: single characteristic generation, multiple characteristics generation (including AND operation and OR operation). Next, we will introduce their corresponding fitness evaluation metrics.

According to the fitness evaluate module mentioned in Fig. 1, we define the notation $G_{trained}(\theta)$ and t as the generated images by the generator and a target characteristic, respectively. Then, $R(t, G_{trained}(\theta))$ is defined as its output, representing the probability that a given image has the target attributes or is the target class.

1) Single characteristic generation: It is the most common generation problem in specific image generating, through which we aim to get specific images with only one characteristic or images belong to the target class. Thus, the fitness of θ represents how likely the generated image $G_{trained}(\theta)$ have the desired characteristics. The more confidence of the recognition network is, the higher the fitness tend to be. Therefore, the fitness function can be expressed as follows:

$$F_s(\theta, t) = R(t, G_{trained}(\theta))$$

or,

$$F_s(\theta, t) = 1 - R(t, G_{trained}(\theta))$$

when we don't want to generate a specific kind of image.

2) Multiple characteristics generation: While single image generation only handle one characteristic or one target class, in order to extend to multiple characteristics, i.e., generating an image contains a 'man' with 'glasses', we also present how to formulate the fitness function according to the operation of multiple characteristics, including AND and OR.

AND: In fact, AND operation can be turned into a single characteristic generation problem by training a recognition model, which could tell the possibility that the generated image $G_{trained}(\theta)$ has all desired attributes. However, this would lead to some problems when the attributes set is large. In order to flexibly manipulate the attributes, we address it in an independent way, that θ's fitness is determined by the minimum value of all single attributes fitness. Therefore, the fitness objective is formulated as,

$$F_{AND}(\theta, t_1, t_2, ..., t_n) = \min F_s(\theta, t_1), F_s(\theta, t_2), ..., F_s(\theta, t_n)$$

where $t_1, t_2, ..., t_n$ are the desired characteristics.

OR: This aims to generate images that only satisfy one of the given characteristics constraints, such as, images contain digits '1' or '7'. In this case, the DNA θ, which could be mapped into images with any of desired classes, should be valued a high fitness score. To end this, we take the maximum fitness score as θ's fitness by the following equation,

$$F_{OR}(\theta, t_1, t_2, ..., t_n) = \max F_s(\theta, t_1), F_s(\theta, t_2), ..., F_s(\theta, t_n)$$

where $t_1, t_2, ..., t_n$ are the desired characteristics.

IV. Experiments

In order to validate the effectiveness of our proposed method, we conduct experiments on the CelebA [27] dataset and MNIST [28] dataset. In the experiments, we implement the GAN and recognition network using Pytorch [29], which is widely used in deep learning. And we use DEAP [30], an open-source algorithm framework, to realize the genetic algorithm.

A. Dataset and Model Preparation

1) CelebA: CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset, including 202,599 face images of 10,177 identities, that each image is with 40 binary

attributes annotations and 5 landmark locations. All the images are rescaled into a size of 64×64.

For simplicity, based on the design of DCGAN [5], we implement the generative model G and discriminative model D in the form of convolution/deconvolution-BatchNorm-Relu/LeakyReLu [31] [32] [33]. It has been proved that this architecture performs well in image generation. In our implementation, the slope of LeakyReLu is set to 0.2. And then, we train this GAN model using Adam optimizer [34], with a learning rate of 0.0005, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and mini-batch size is 128.

For a recognition model, which is required to identify the attributes exactly, we utilize a pre-trained ResNet [35] to extract features, convolution layers and fully-connected layers are then employed to output the predict results. Since the distribution of the binary annotations of attributes is very unbalanced, we select 4 attributes with relatively balanced annotations for our experiments, including "Gender", "Glasses", "Mouth Open" and "Smile". Furthermore, we train the recognition model using SGD [36] with a learning rate of 0.0001, momentum of 0.5 and mini-batch size is 50. Finally, we get a recognition model and the accuracy is shown in Table. I.

TABLE I
ACCURACY OF THE RECOGNITION MODEL

Attribute	Glasses	Gender	Mouth Open	Smile
Accuracy	99.5%	95.5%	93.1%	90.7%

2) MNIST: MNIST is a widely used handwriting digit images dataset, which consists of 60,000 training and 10,000 testing 28×28 grayscale images of digits from 0 to 9.

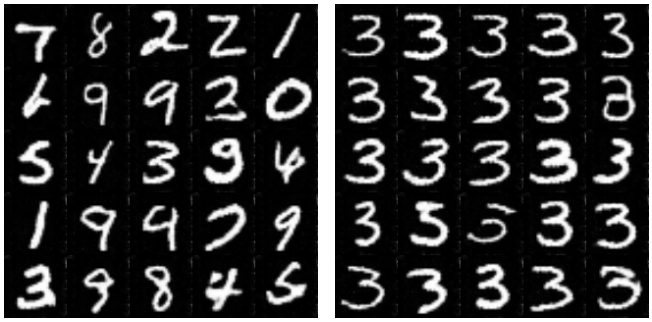
For generation, same as CelebA, we also use DCGAN as the base architecture to implement our MNIST GAN model. And, the training settings is exactly the same as DCGAN training settings described above, except the learning rate of 0.0002.

For recognition, we train a simple, yet effective 3-layer convolution networks that could predict image classes and compute the label probabilities. Note that this recognition network can achieve 99.3% accuracy on MNIST testing set.

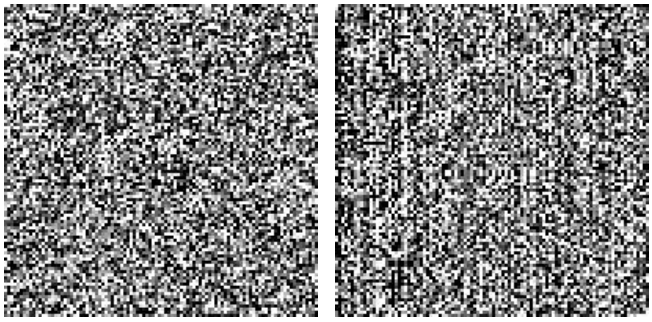
B. Single Digit Generation

Here, we will show the effectiveness of our proposed method and then visualize the evolution happened in DNA pool. To do this, we generate handwriting digit images with a specific class t (here, t = "3"). We use the corresponding fitness $F(\theta, 3)$ and evolve the DNA pool. Besides, we randomly choose 100 DNA θ from DNA pool, and then map them to a grayscale image, which helps to monitor the evolution process. The DNAs θ are fed into the trained generator G_{MNIST} to generate images. The results of our method are shown in Fig. 2(b), we can clearly see that almost all images generated by our method are the desired images, compared to the original images in Fig. 2(a). What's more, the origin DNA Pool seems to be very messy, while it seems organized after evolution.

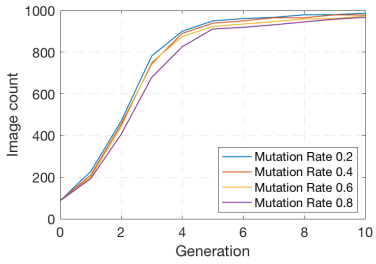
As shown in our experiment, our method only takes several generations to get the desired DNA pool (here, 4 generations).



(a) Original Images (b) GAGAN: 3



(c) Original DNAs (d) Optimized DNAs



(e) The process of evolution

Fig. 2. Results of Single Digit Generation

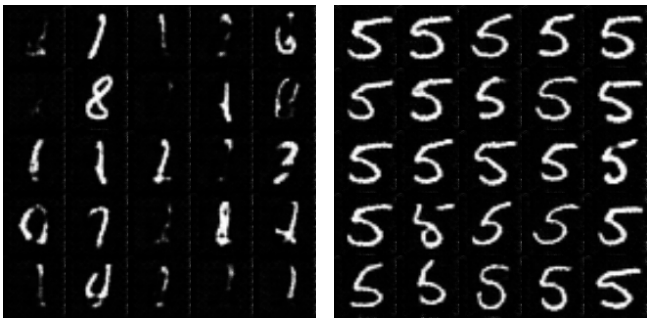
We also evaluate the time cost of our method, it only takes 2 minutes using a Core i5 CPU, while training a DCGAN costs about 10 hours using a GTX1080 GPU. This is mainly because our method doesn't need gradient computing and BP optimization, which is computationally expensive.

We also conduct this experiments with different mutation rate to verify the influence of mutation rate. Note that if the attribute of a generated image is predicted the same as the desired one by the recognition model, it is considered a correct generation, otherwise a wrong one. And the generation accuracy is the proportion of the correct generations. As Fig. 2(e) shows, regardless of mutation rate, the percentage of desired images increases, as the evolution goes and finally reach a high generation accuracy, above 98%. However, we notice that the converging speed of our method decreases with the increasing mutation rate. This is mainly because the mutation of DNA contributes to the diversity of images. Apart from mutation, we also explore the influence of different

crossover strategies, but we find that the result have only small changes, so it is not a key factor in this problem. Furthermore, as observed in the figure, our method is able to more distinguishable and clear images.

C. Generation by "Broken Generator"

In this part, we demonstrate our method's ability to improve the quality of generated images, which means we can also generate from not well trained generator. To achieve this, we first get a "Broken Generator" by changing the input distribution into $[0, 1]$. Then, we use our method to generate specific handwriting digits (here, "5"). As we can see from Fig. 3(a), the quality of original generated images is quite poor. On the contrary, in Fig. 3(b), the images generated by GAGAN are visually pleasing and distinguishable. Since recognition model is more confident on the images with distinct characteristic and clear edges, fitness evaluate module value it a high fitness score. Thus these high-quality images survive as the evolution goes and our method still performs well even the generator is broken or not well trained.

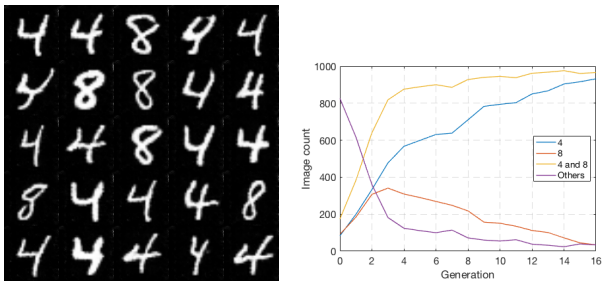


(a) Original Images (b) GAGAN: 5

Fig. 3. Results of Generation by "Broken Generator"

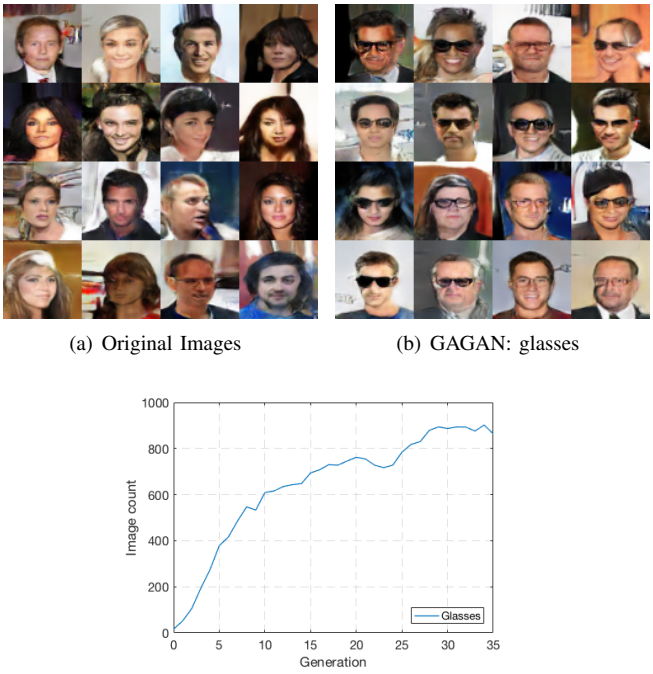
D. Two Digits Generation

According to the experiments above, we have evaluated GAGAN's performance on image generation with a single digit, next, we will conduct evaluation to reveal how GAGAN performs while generating multiple characteristics. In order to give a detailed description, we take an OR operation as example, such as handwriting digit generation with "4" or "8". Assuming t_1, t_2 represent digits "4" and "8", respectively, then, the fitness evaluate metrics can be presented as $F_{OR}(\theta, t_1, t_2)$. Moreover, we set the mutation rate to 0.2 in this experiment. As illustrated in Fig. 4(a), results show that the range of generated handwriting digit images is limited to "4" and "8" as expected. And Fig. 4(b) also reveal that the amount of desired images increases as the evolution goes. It further validates the effectiveness of our proposed GAGAN. Furthermore, we also find an interesting phenomenon that after the increase at the beginning of training, the proportion of "4" and "8" perform an inverse trend, that digit "8" decreases while the proportion of digit "4" continues to grow. This is mainly because the recognition ability of model R is imbalanced among different digits. The recognition model has more confidence in recognizing digit "4" than "8", results in a higher fitness score to "4". Therefore, digit "4" has a higher possibility to survive in the evolution process.



(a) GAGAN: 4 OR 8 (b) The process of evolution

Fig. 4. Results of Two Digits Generation



(a) Original Images (b) GAGAN: glasses

(c) The process of evolution

Fig. 5. Results of Single Attribute Generation

TABLE II
ATTRIBUTES OF 1000 ORIGINAL IMAGES

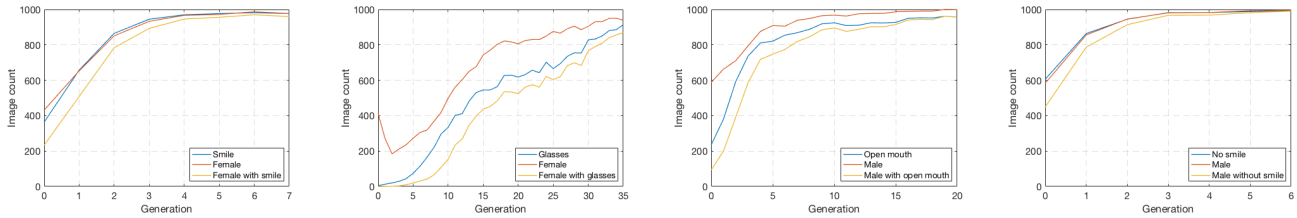
Attribute	Glasses	Male	Open Mouth	Smile
Count	7	583	262	394

E. Single Attribute Generation

In this part, we demonstrate the ability to generate facial images with a single attribute. As Table. II illustrates, we first analyze the attributes of original face images generated by G_{CelebA}. We can see that the proportion of people with glasses is very small, less than 1%. Therefore, in order to better validate the ability of our method, we choose the attribute

(a) female with smile (b) female with glasses (c) male with open mouth (d) male without smile

Fig. 6. Results of Multiple Attributes Generation



(a) female with smile (b) female with glasses (c) male with open mouth (d) male without smile

Fig. 7. The process of evolution

"Glasses" as our desired characteristic t. Then, fitness of $F_s(\theta, t)$ and mutation rate of 0.2 are applied in evolution. The generated results are shown in Fig. 5(b), from it, we can intuitively find that it is feasible to generate facial images with glasses. Moreover, the generation accuracy reaches 90% at the end of evolution. This reveals that our method performs well in the single attribute generation task.

Note that a reduction of diversity might occur when the generation model reaches a high accuracy. As explained above, there is a trade-off between the generation accuracy and diversities of generated images. In this experiment, we empirically use the DNA pool, with generation accuracy of 85%, to produce suitable DNAs for generation.

F. Multiple Attributes Generation

Similar to MNIST, we also conduct an experiment to generate facial images with multiple attributes, and the AND operation is chosen as example to make a detailed description. In this experiment, we introduce four kinds of attributes combination: "female with smile", "female with glasses", "male with open mouth" and "male without smile". It is natural to use F_{AND} as the fitness evaluate metric. Results are given in Fig. 6, it can be easily observed that both the generated images contain the desired attributes. As Fig. 7 shows the corresponding quantitative change of images for each generation task, it shows that both achieve a high generation accuracy and the single attributes also increases as the evolution goes. And the rate of convergence is different because of different starts.

Furthermore, we also intuitively observe that in Fig. 7(b), a decrease in the amount of 'female' images happens in the first two generations, which is supposed to be an increase. This is mainly due to that there are 9 images with glasses in original generated images, but only one of them is the female. In comparison with the gender, the glasses has a much more influence on the fitness. Therefore, the corresponding DNAs of images that male wears glasses dominate the beginning of the evolution.

V. Conclusions

In this paper, we propose GAGAN method for generating specific images. With GAGAN, we also address the specific image generation problem under the condition that given trained GAN model and recognition model but without the dataset. To the best of our knowledge, it is the first method that can generate images with specific characteristics from a trained GAN model. By combining Genetic Algorithm with GAN properly, it could produce a DNA pool, which could generate the corresponding latent vectors (DNAs) of desired images. Meanwhile, it could also generate visually pleasing images directly from a not well trained generator. Furthermore, to make GAGAN more flexible, we separate generation problems into three meta classes and introduce the corresponding fitness evaluation metrics. Detailed experiments and analysis are conducted on MNIST and CelebA dataset and the extensive and qualitative results demonstrate the effectiveness of our proposed method.

VI. Acknowledgment

This work is supported by the National Key Research and Development Program No.2018YFC0831800.

REFERENCES

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *ICML*, 2016, pp. 1558–1566.

[3] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks," in *ICML*, 2014, pp. 1242–1250.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *ICRL*, 2016.

[6] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *NIPS*, 2015, pp. 1486–1494.

[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *ICLR*, 2018.

[8] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[9] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *ICML*, 2017, pp. 2642–2651.

[10] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NIPS*, 2016, pp. 2172–2180.

[11] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[12] J.-B. Mouret and S. Doncieux, "Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 1161–1168.

[13] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria, "On the performance of indirect encoding across the continuum of regularity," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, p. 346, 2011.

[14] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[15] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[16] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015, pp. 427–436.

[17] H. Zhang, T. Xu, and H. Li, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *ICCV*, 2017, pp. 5908–5916.

[18] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016, pp. 2536–2544.

[19] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *NIPS*, 2016, pp. 64–72.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*. IEEE, 2017, pp. 5967–5976.

[21] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma, "Dual learning for machine translation," in *NIPS*, 2016, pp. 820–828.

[22] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, 2017, pp. 1857–1865.

[23] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *ICCV*, 2017, pp. 2868–2876.

[24] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017, pp. 2242–2251.

[25] H. Ding, K. Sricharan, and R. Chellappa, "Exprgan: Facial expression editing with controllable expression intensity," *AAAI*, 2018.

[26] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Arbitrary facial attribute editing: Only change what you want," *arXiv preprint arXiv:1711.10678*, 2017.

[27] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015, pp. 3730–3738.

[28] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, p. 18, 2010.

[29] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 2017.

[30] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.

[32] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

[33] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2014.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[36] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.