https://doi.org/10.1016/j.knosys.2020.106102

Sketch-then-Edit Generative Adversarial Network

Wei Li
School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, Jiangsu, P.R.China.
Jiangsu Key Laboratory of Media Design and Software Technology, Wuxi, Jiangsu, P.R.China. Email: umass.weili@gmail.com.
Linchuan Xu*
Corresponding author. Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P.R.China. Email: linchuan.xu@connect.polyu.hk.
Zhixuan Liang
Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P.R.China. Email: zhixuan.liang@connect.polyu.edu.hk.
Senzhang Wang
Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, P.R.China. Email: szwang@nuaa.edu.cn.
Jiannong Cao
IEEE Fellow. Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P.R.China. Email: csjcao@comp.polyu.edu.hk.
Chao Ma
School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, P.R.China. Email: whmachao@ieee.org.
Xiaohui Cui*
Corresponding author. School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, P.R.China. Email: xcui@whu.edu.cn.

Abstract

Generative Adversarial Network (GAN) has been widely used to generate impressively plausible data. However, it is a non-trivial task to train the original GAN model in practice due to the vanishing gradient problem. This is because the JS divergence could be a constant (i.e., log2) when original data distribution and generated data distribution hold a negligible overlapping area. Under such a scenario, the gradient of generator is 0. Most efforts

have been devoted to designing a more proper difference measure while few attentions have been paid to the former aspect of the issue.

In this paper, we propose a new method to design a noise distribution having a guaranteed non-negligible overlapping area with raw data distribution. The key idea is to transform the noise from the randomized space into the raw data space. We propose to obtain the transformation as the basis matrix in non-negative matrix factorization because the basis matrix has the underlying features of the raw data. The proposed idea is instantiated as Sketch-then-Edit GAN (SEGAN) where sketches are the noises after transformation and are adopted as the name since they contains basic features of the raw data. Moreover, a new generator for editing the sketches into realistic-like data is designed. We mathematically prove that SEGAN solves the gradient vanishing problem, and conduct extensive experiments on the MNIST, CIFAR10, SVHN and Celeba datasets to demonstrate the effectiveness of SEGAN.

Keywords: Generative adversarial network; vanishing gradient problem; non-negative matrix factorization.

---

1. Introduction

Recently, generative adversarial network (GAN) [13] has been widely used for generating synthetic but realistic-like data in various machine learning tasks [42] [24] [37] [47] [23] [45]. In general, a typical GAN model consists of two components, a discriminator D and a generator G. The discriminator D can be viewed as a detective who can determine whether the current data are from the realistic dataset or generated by the generator G, and the generator G specializes in generating the simulation data to fool the discriminator D into accepting them as real ones [8]. In particular, the generator G is modelled such that it can transform a vector of noises z sampled from an easy-to-sample distribution (e.g., Uniform distribution) into a synthetic instance, and the discriminator D takes a sample as the input and outputs a probability that the input is real. D and G are adversarially trained until they reach a Nash equilibrium [9] where the discriminator cannot tell the difference between the real data and the synthetic data.

Despite of the effectiveness of the elegant framework, the original GAN has been known hard to train in practice due to the vanishing gradient problem, which is caused by a negligible overlapping area between generated data

and raw data distributions, because the Jensen–Shannon (JS) divergence is a constant and the gradient of generator is 0 under such a scenario [13] [3]. In particular, both the raw data distribution (p_r) and the generating distribution (p_G) lie in low-dimensional manifolds, and they have very limited chances to overlap because they usually have disjoint supports [2]. Under such a scenario, the Jensen–Shannon (JS) divergence is a constant ($log2$) when p_r and p_G have a negligible overlapping area, and the gradients of the generator can be zero. More details about the vanishing gradient problem in the traditional generator are shown in Section 3.

Many methods have been proposed to tackle the issue by utilizing a more proper difference measure. Energe-based GAN (EBGAN) [46] uses the margin loss function to replace the original loss function. However, a study [3] has proved that minimizing the margin loss equals to minimize the total variation distance [34], and such modification cannot address the vanishing gradient problem when p_r and p_G have no overlapping area. Least Squares GAN (LSGAN) [26] proposes to utilize the least square loss function. We found that LSGAN cannot address this problem when there is a negligible overlapping area between p_r and p_G because the generator's loss is a fixed value as $p_r = 0$ and $p_G \neq 0$ (and vice verse), and the gradients are still 0 in such cases [2]. Wasserstein GAN (WGAN) [3] proposes to utilize the Wasserstein distance [31] to measure the dissimilarity between p_r and p_G even though there is no overlapping area between the two distributions. However, WGAN needs more iterations to push p_G to p_r as p_G is far away from p_r at the initial stage. Moreover, there still exists the challenge on addressing the issue when one uses the *Adam* optimizer [3].

Few efforts have been devoted to approaching the issue from the aspect of the overlapping area, which may be because the raw data distribution is usually unknown and lies in a low-dimensional manifold in practice such that it is very difficult to determine a generating distribution having a negligible overlapping area with the raw data distribution. Note that the generating distribution depends on both the noise distribution and the parameters in the generator (i.e., a neural network), but the noise distribution takes the dominant role in the generating distribution since the support of the generating distribution is contained in the noise distribution [2]. One heuristic strategy is to introduce the same continuous noises into both the raw data distribution and the generating distribution because the noises can make the two distributions match [2].

In this paper, we attempt to propose a new method to address the issue

of a negligible overlapping area. In particular, the idea is to build a noise distribution which is similar to the raw data distribution such that a non-negligible overlapping area can be guaranteed. Note that we do not attempt to learn a parametric noise distribution that fits the raw data since the raw data distribution is unknown and the task of fitting the raw data is actually to be performed by the generator. Instead, we propose a non-parametric approach which can produce samples. In particular, we firstly capture the raw data features through the basis matrix obtained by the Non-negative Matrix Factorization (NMF) [12]. NMF factorizes a data matrix X into a basis matrix W and a latent data representation matrix H via $X \approx W \times H$. Each column of W is a latent feature of the data and H can be considered as the new representations of data in terms of the basis W. We employ the non-negative matrix factorization because the real data should be the addition of basic features [12].

Afterwards, we produce a new noise input via multiplying W by a sample z drawn from the standard Uniform distribution (0, 1), which realizes the transformation of the original noises from a randomized space into the raw data space. The standard Uniform distribution is employed to guarantee the non-negativity of the addition of basic features. In this way, the new input is a randomly linear combination of the basis features of X, and hence it mimics a sample drawn from the raw data distribution. In this paper, the new noise input is named as a sketch, and the idea is instantiated in a novel model named Sketch-then-Edit GAN (SEGAN).

The way of constructing the sketch requires redesigning the architecture of the generator. Because the size of the sketch is the same as that of the raw data, the strided-fractional convolution [1], which is the major component of a typical generator, is no longer needed. Instead, the generator should focus on editing the sketch with basis features into complete and realistic-like data. Note that the editing is necessary because the random combination of the basic features may be far away from realistic data, e.g., a combination of the tail of fish and the feet of cats can never be found in real world.

Note that our proposed idea belongs to the paradigm of *Transfer Learning* [33], but, to the best of our knowledge, the proposed SEGAN adopts a novel strategy to realize the *Transfer*. Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. In our paper, we use NMF model to store the knowledge gained from raw data and apply the combination of knowledge and noise to GAN model to generate simulation data.

In summary, the major innovations and contributions of this paper are as follows:

- To the best of our knowledge, this paper for the first time proposes a new method to construct a noise distribution having a guaranteed non-negligible overlapping area with the raw data distribution in the GAN literature.
- This paper proposes a novel SEGAN model which learns a mapping function parameterized by the generator of SEGAN from the aforementioned noise distribution into the raw data distribution without suffering from the vanishing gradient problem.
- Through comprehensive experiments on four datasets, we demonstrate the effectiveness of the proposed approach.

The rest part of this paper is organized as follows. In Section 2 some related methods are discussed. We review the GAN model and NMF method in Section 3, and present our model in Section 4. In Section 5 we show our experimental results and the conclusion is made in Section 6.

2. Related work

The GAN model has gained a lot of attentions due to its powerful generative capabilities, and plenty of variants have been proposed. Here, we mainly review those studies for tackling the vanishing gradient problem. It is known that the vanishing gradient problem is the joint consequence of two factors which are the negligible overlapping area and the JS divergence, and the issue may be solved with either a non-negligible overlapping area or a more proper difference measure. Hence, we categorize existing studies according to the factor with which they deal.

Overlapping Area. A study [2] has shown that the negligible overlapping area between p_r and p_G may be inevitable if p_G lies in a random low-dimensional manifold. Hence, from this aspect, we need to make some manipulations on p_r or p_G or both of them. Wu et al. [39] and Roth et al. [30] propose a similar heuristic, which is to add some noises to both of the two distributions in the early phases of training, because the added noises can enforce the two distributions to match. And then they gradually remove the noises using annealing methods as the training continues. However, when

one uses the JS divergence to measure p_r and p_G with additional noises, the results are affected by those noises and cannot represent the true difference between p_r and p_G [3].

Difference Measure. More researchers focus their attentions on designing difference measures [46] [3] [2] [26]. EBGAN [46] found that the margin loss function can get better quality of gradients when p_G is far away from p_r. It has proven that if the discriminator and the generator reach to the Nash equilibrium [9] ($p_G = p_r$ almost everywhere), $V(G, D)$ (the loss function of the GAN model is defined as $V(G, D)$ shown in Eq.(1)) equals to a positive margin (m). However, the study [3] has proved that EBGAN can be though of as the generative approach to the total variation distance [34], and also gives that total variation distance displays the same regularity as the JS divergence. Thus, EBGAN still suffers from the problem of vanishing gradients.

Margin Adaptation GAN (MAGAN) [36] adopts the adaptive margin loss function to replace the original loss function to stabilize training. Thus, this model is very similar to EBGAN. The difference is that MAGAN can converge under more relaxed assumptions. Since this study adopts the same loss function, it still cannot address the vanishing gradient problem.

WGAN [3] is one of the state-of-the-art algorithms for addressing the vanishing gradient problem, because of the superior smoothing characteristics of the Wasserstein distance [31]. Such characteristics can quantify the distance between p_r and p_G even though the supports of them are disjoint. In WGAN, the discriminator is attached with a function $f_w^*(x)$, which has been limited over the 1-Lipshitz continuity. For satisfying the Lipshitz continuity, $\frac{\partial f_w^*(x)}{\partial x}$ has been limited to a specific range (e.g., (-0.01, 0.01)). That is to say, WGAN clips the weights w after each training epoch. The limitation of WGAN is that only certain optimizers (e.g., $RMSProp$ or SGD) are suitable for optimizing WGAN [3] but momentum based ones (e.g., $Adam$) may even turn the gradients into being negative.

Loss-Sensitive GAN (LS-GAN) [29] is also based on the Lipshitz continuous assumption. The difference between WGAN and LS-GAN is that WGAN is to maximize the difference of expectation of $f_w^*(x)$ between p_r and p_G while LS-GAN makes pairwise comparisons between the loss of real and that of generated samples. LS-GAN is effective even without batch normalization. However, the derivation and implementation of LS-GAN are very complicated, which makes the adoption of LS-GAN not cost-effective.

Least Squares GAN (LSGAN) [26] argues that the Sigmoid Cross Entropy

loss function [44] for the discriminator may lead to the problem of vanishing gradient problem when updating the generator using the fake samples. Therefore, LSGAN uses Least Squares to measure the difference between p_r and p_G rather than the Sigmoid Cross Entropy. However, such measure metric still cannot address the vanishing gradient when one of p_G or $p_r \to 0$ and the other one $\neq 0$ as the discriminator reaches the optimal state.

The Relevance of Sketch-then-Edit GAN to Existing Studies. The model proposed in this paper approaches the vanishing gradient problem from the aspect of the overlapping area. But instead of another heuristic, we propose a new method to make the p_G overlap the p_r. In particular, we transform the noises sampled from a randomized space, e.g., a Uniform distribution, into the raw data space. Note that other studies [7] [27] use the term "sketch" to indicate "outline" of an image and utilize GAN to render color of outline. In our study, the "sketch" indicates a combination of randomized noise and salient features of original data. More details are shown in Section 4.

3. Preliminaries

3.1. Generative Adversarial Network

The original GAN is formally described as below to establish the continuity. The GAN model was first proposed by Goodfellow [13] as a novel generative model to simultaneously train a generator and a discriminator using the following objective function:

$$\begin{aligned} \min_G \max_D \; V(G,D) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] \\ + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \end{aligned} \tag{1}$$

D is the discriminator, G is the generator, and both are neural networks. x comes from a distribution $p_r(x)$ underlying the raw dataset and z comes from a pre-defined noise distribution $p_z(z)$ which is usually an easy-to-sample distribution, e.g., Uniform distribution or Gaussian distribution. The discriminator is a binary classifier which takes an input from either the raw data or the generator and produces a probability that the input comes from the raw data. The generator is designed to build a mapping function from $p_z(z)$ to the raw data distribution $p_r(x)$ such that it can produce a synthetic instance with a noise input z. Maximizing Eq. (1) with respect to

D enhances the capability of the discriminator to tell the difference between a real instance and a synthetic one while minimizing Eq. (1) with respect to G improves the mapping function such that the synthetic data are more realistic-like. During the training process, the discriminator and the generator are alternatively optimized. In the initial stages, the discriminator is usually stronger than the generator and can easily distinguish the real data from synthetic data. As the training continues, the mapping function is gradually improved, and the model gets the convergence when the discriminator and the generator reach a Nash equilibrium [9] where $D(G(z)) = D(x) = 0.5$.

3.2. Vanishing Gradient Problem

In the initial stage of training a GAN model, the generated data are basically noises. The discriminator reaches to the optimal status ($D^* = \frac{p_r}{p_r+p_G}$) with a high probability under such scenario [13]. We then substitute the optimal discriminator (D^*) back into Eq. (1), and we obtain $\max_D V(G, D) = -log4$ if and only if $p_r = p_G$. We continue to subtract this expression from $\min_G V(G, D^*)$, and we obtain:

$$V(G, D^*) = -2log2 + 2JSD(p_r(x)||p_G(x)) \tag{2}$$

That is to say, the difference between p_r and p_G is determined by the JS divergence (JSD), so minimizing the difference is to minimize the JS divergence when the discriminator is optimal. The smaller the JS divergence is, the closer the two distributions (p_r and p_G) are.

The JS divergence works well for updating the generator's parameters if there is an overlapping area between p_r and p_G. And the generator obtains the minimum value ($-2log2$) when $p_r(x) = p_G(x)$ ($JSD = 0$ under such scenario). The challenge is that $JSD(p_r(x)||p_G(x)) = log2$ if there is no overlapping area between p_r and p_G. Unfortunately, it is of high probability that in practice two distributions may be disjoint because both of p_r and p_G lie in low-dimensional manifolds in the mapping space [2]. In other words, $JSD = log2$ is often the case. $V(G, D^*)$ is 0 when we substitute $JSD = log2$ into Eq. (2). The generator's gradients vanish under such scenario because the derivative of a constant (0 in this case) is 0.

3.3. Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a linear dimensionality reduction technique [12] [35] [41] [43]. Given a data matrix $X \in \mathbb{R}^{p \times n}$, NMF amounts to computing a basis matrix $W \in \mathbb{R}^{p \times r}$ that only contains basis elements (or features) of X, and a latent representation matrix $H \in \mathbb{R}^{r \times n}$ through $X \approx W \times H$. Here, $p \in \mathbb{R}$ is the number of dimensions of the raw data, $n \in \mathbb{R}$ is the number of data points, and $r \in \mathbb{R}$ is the number of reduced dimensions of the latent representations with $r < min(p, n)$. In this way, each data point $X(:, j)$ $(1 \leq j \leq n)$ can be approximated by a linear combination of the columns of W where the corresponding coefficients of the columns are in $H(:, j)$. Since both W and H are restricted to be nonnegative, NMF can automatically extract sparse and interpretable features (e.g., eyes, noses) from the raw data [12].

The motivations of using NMF are two-folds: 1). The real data are usually non-negative (e.g., image datasets) in practice, and the NMF method can keep non-negativity and automatically extract interpretable non-negative features [20]. 2). NMF has a strong interpretability for the regional features from the raw data [21] due to the fact that each basis feature is part of the original sample (e.g., a nose or mouth).

There are some other unsupervised dimensionality reduction methods such as Principal Component Analysis (PCA) [38], Singular Value Decomposition (SVD) [10], Independent Component Analysis (ICA) [22], and they are widely adopted in practice. PCA and SVD have no assumptions on the basis matrix and the representation matrix, but instead aim to obtain an optimal factorization. Note that the SVD is equivalent to principal component analysis (PCA) after mean centering of the data points. ICA enforces each column of the basis matrix to be independent of all the others. In this way, all these methods may produce both positive and negative basis features, and hence are not as suitable as NMF for SEGAN.

4. The proposed SEGAN

4.1. Overview

The major innovation of SEGAN is the idea of constructing a noise distribution having a non-negligible overlapping area with the raw data distribution. However, it is difficult to learn a parametric function as the noise distribution that fits the raw data because no information about the raw

data distribution is known as prior knowledge. Moreover, the task of capturing the raw data distribution is actually expected to be performed by the generator. To solve this challenge, we note that a parametric function is essentially not necessary since the input into the generator is samples instead of the parametric function. Hence, the problem is simplified to how to obtain samples from such an unknown noise distribution. To this end, we propose to firstly draw noise samples from a Uniform distribution, and then transform the samples into the raw data space. To guarantee the successful transformation into the raw data space, the basis matrix in the non-negative matrix factorization is employed. In particular, a new sample is obtained as the multiplication of the basis matrix by a vector of noises from the Uniform distribution. Samples constructed in this way are essentially random combinations of basis features of the raw data, and hence they indeed lie in the raw data space. The samples are referred to as sketches in this paper. To produce synthetic data based on sketches, the generator with a new architecture is required. Hence, the following subsections explain how the sketches are constructed in details and what the new generator should be.

4.2. Sketch Construction

We illustrate the process of constructing sketches in Fig. 1 where the MNIST dataset is employed as the example. On the one hand, a matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ is firstly constructed from the raw data (sub-figure (a)) where $p \in \mathbb{R}$ is the number of dimensions of images after vectorization and $\boldsymbol{n} \in \mathbb{R}$ is the number of images. Then, the basis matrix is obtained via non-negative matrix factorization as $\boldsymbol{W} \in \mathbb{R}^{p \times r}$ where each column of $\boldsymbol{W}$ is a basis feature and r is the total number of basis features. Here r is set to 10 for illustrating how to create the sketch. The visualization of each basis feature is conducted after reconstructing it into an image in sub-figure (b) of Fig. 1. It shows that each image is a salient feature of a digit.

On the other hand, a vector of noises ($z^{r \times 1}$) is sampled from a Uniform distribution ($p_z(z)$) in Fig. 1. Then, the vector is multiplied to the basis matrix $\boldsymbol{W}$ to form a sketch. The multiplication realizes the transformation of each noise vector from a randomized space corresponding to the chosen $p_z(z)$ into the raw data space. More concretely, each sketch is a random combination of all the basis features denoted as follows:

$$\text{Sketch} = \boldsymbol{W} \cdot \boldsymbol{z} \tag{3}$$

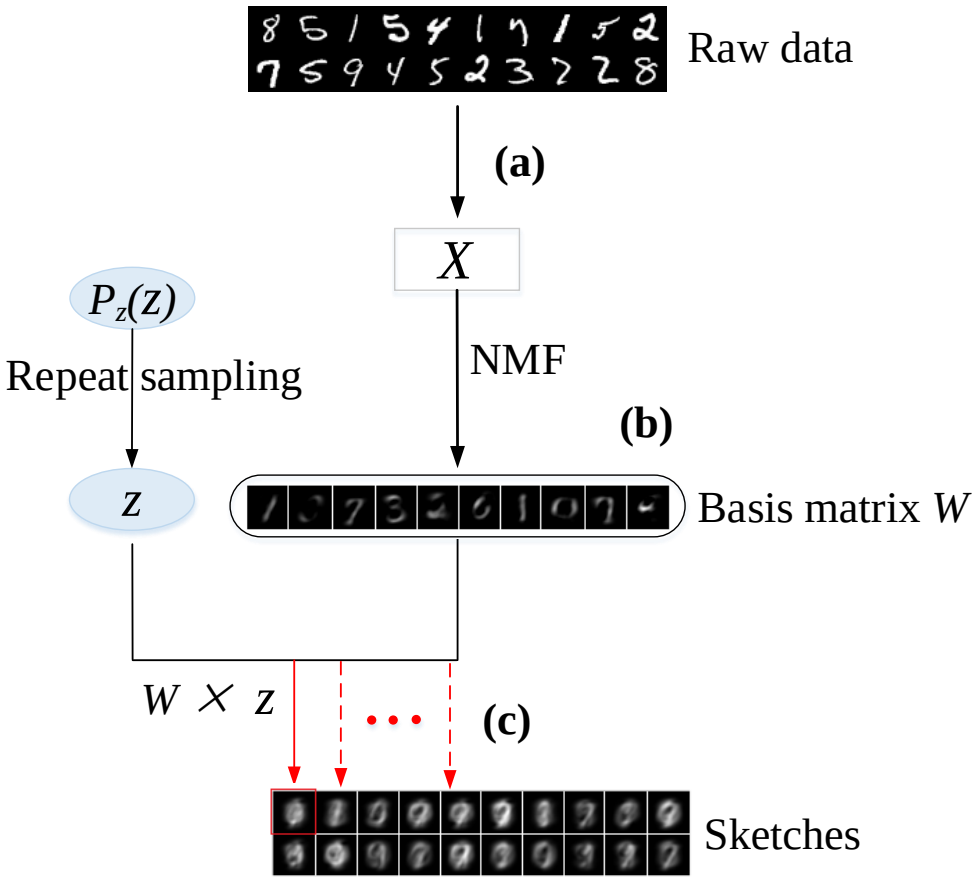


Figure 1: The illustration of constructing sketches where each sketch corresponds to a vector $\boldsymbol{z}$ sampled from $P_z(z)$.

where $\boldsymbol{W}$ indicates the basis matrix with size $p \times r$ while z refers to the noise code with size $r \times 1$. 20 sketches are presented in the sub-figure (c) of Fig. 1. Note that we repeat sample the noise code z from $p_z(z)$ 20 times, and each time we utilize Eq. (3) to create a sketch. It is observed that some sketches are already realistic-like digits while others have features of more than one digit. The former is because the corresponding noise vector sampled from the Uniform distribution are similar to a particular column of $\boldsymbol{H}$ while the latter may be because the corresponding vector is a particular combination of more than one column of $\boldsymbol{H}$ because each column of $\boldsymbol{H}$ can reconstruct a particular digit. It also confirms that sketches constructed in this way need

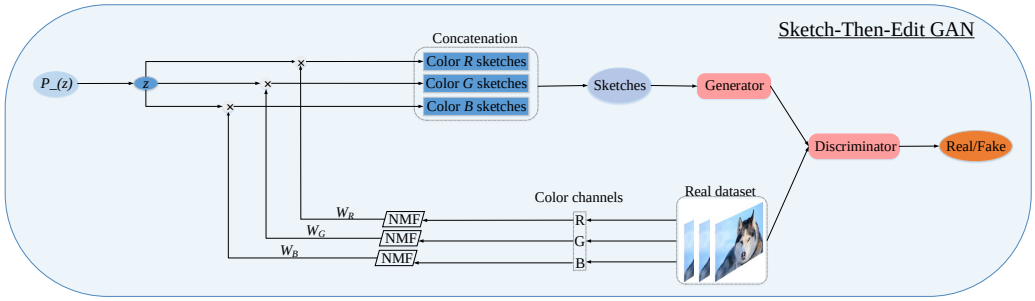


Figure 2: The architecture of SEGAN.

to be further edited into realistic digits, which is to be performed by the generator.

It is worth mentioning that when dealing with RGB images where the three colors are in three channels, three matrices ($\boldsymbol{X}_R \in \mathbb{R}^{p\times n}$,$\boldsymbol{X}_G \in \mathbb{R}^{p\times n}$,$\boldsymbol{X}_B \in \mathbb{R}^{p\times n}$) are required to be constructed for the three channels. This is because we need to maintain the three channels as input. We then simply apply NMF to each of the channel-specific matrices and obtain three basis matrices. According to Eq. (3), three channel-specific sketches can be constructed. After that, the three channel-specific sketches are concatenated in the vertical axis to create a complete sketch as the new noise input (See Fig. 2). Note that there is no channel sketches for gray-scale image dataset (e.g., MNIST) because the gray-scale image just holds one color channel.

Since the number of basis features (i.e., the value of r) needs to be predetermined, we discuss here the impact of r on the performance of SEGAN as the guidance for choosing an appropriate value. When the number of basis features is very small, e.g., 1 or 2 in extreme cases, the diversity of sketches may be very limited according to Eq. (3). In this case, the diversity of synthesis data produced by the generator is also very limited, which leads to model collapse due to the similar inputs (i.e., similar sketches). When r takes a very large value, each basis feature obtained in $\boldsymbol{W}$ may be too fine-grained such that it may not be a salient feature. In other words, each W_i captures the trivial details. For example, when we use the NMF method to extract the salient features of a person using a moderate value r (e.g., the number of r exactly equals to that of features), the salient features captured in the basis matrix are hair, eyes, nose, etc. Instead, if r takes a very large value, the features may focus on the individual pixels of eyes, hair, and etc. If we utilize

those features to create the sketches, they basically have no difference from the original noises sampled from the randomized space. This is validated by the experiments shown in Section 5.

Note that a non-negligible overlapping area between the generating distribution and the raw data distribution can be guaranteed when we only design a noise distribution having a non-negligible overlapping area with the raw data distribution, which is because the support of the generating distribution is contained in the input noise distribution when the mapping function parameterized by the generator is composed of affine transformations and pointwise nonlinearities [2].

4.3. Generator Design

For the traditional generator, it focuses on the upsampling method with the fractionally-strided convolutions [1] to produce high-dimensional instances. This is because the original noise input is usually a low-dimensional vector. As we will explain in the following paragraph, the fractionally-strided convolutions are not suitable for SEGAN, and a new design is required.

First of all, it is not necessary to increase the size of the input in each layer of the generated network since the size of the input, i.e., sketches, is already the same as that of the raw data. Moreover, to increase the size of original noises, the fractionally-strided convolution at each layer interpolates zeros among the elements of its input. If the fractionally-strided convolution is applied on the sketch, the basis features in the sketch may be contaminated by the zero values. Therefore, to transform a sketch into a synthetic instance, the generator should better only focus on editing the content of the sketch.

To this end, the new generator relies on the standard convolution method [11] to support the editing operation on sketches, which is in line with that of convolving a normal picture. Because the size of the sketch decreases after the convolution, the *padding* function is needed to pad the margin of the sketch with zero values for maintaining the size of the sketch as that of the raw data. Moreover, to minimize the negative impact of *padding* on the sketch, the number of both rows and columns to be padded to the margin is limited to one, which can be realized by carefully designing *kernel size*, *stride* and *padding* in the convolution. Note that for editing the sketch into a synthetic instance, we also need more than one layer of convolution to be consistent with the convention of deep learning. Here, the configuration of the convolution of each layer is the same to keep invariance of the size of input and output of each layer.

	DCGAN	SEGAN
1	Using fractionally-strided convolutions.	Using standard convolutions.
2	Using $Tanh$ activation function in the output.	Using *Sigmoid* activation function in the same layer.
3	The numbers of *stride* and *kernel_size* are not fixed within each layer, because of interpolation.	The numbers of *stride* and *kernel_size* are fixed within each layer for keeping the invariance of the shape and size of sketch.

Table 1: Comparison between DCGAN's generator and SEGAN's generator.

The ReLU activation function has been adopted in the new generator because it is beneficial for not only capturing the non-linear concept but also keeping the non-negativity of the generated data after each convolution layer. Besides, the last layer for the traditional generator always adopts the $Tanh$ function [25] for covering the color space of the training dataset [1], while the generator of SEGAN adopts the traditional *Sigmoid* function [16] due to its non-negativity. Since the generator of DCGAN is widely adopted in other GAN variants such as BEGAN[5], EBGAN [46], WGAN [3], LSGAN [26] and DeLiGAN [14], it becomes the *de facto* standard for designing the generator. In Table. 1, the generator of SEGAN is compared with that of DCGAN in terms of the convolution operation, activation function and the key parameters of the convolution.

4.4. The Algorithm

With the proposed sketches and the customized generator, the architecture of SEGAN is shown in Fig. 2. We use RGB images as the illustration. The architecture is the same as a typical GAN model except for how the noise inputs are sampled. Note that we directly employ an existing design of the discriminator, e.g., DCGAN's discriminator, since the output of the generator has not been changed. In this way, the formula of SEGAN is shown as follows.

$$\begin{aligned} \min_G \max_D \; V(G,D) = & \mathbb{E}_{x\sim p_r(x)}[\log D(x)] \\ & + \mathbb{E}_{Wz\sim p_{Wz}(Wz)}[\log(1-D(G(Wz)))] \end{aligned} \tag{4}$$

where W is the basis matrix factorized by NMF and it represents the features of raw data X. z comes from an easy-to-sample noise distribution, e.g., $p_z(z) = U(0,1)$. $p_{Wz}(Wz)$ jointly forms a new sampling space through multiplying W by z, and SEGAN transforms Wz from this new space into the raw data space $\mathcal{X}$. According to the *Radon-Nikodym* theorem (it is also known as *law of the unconscious statistician*) [28] [15], we can derive the following equation.

$$\mathbb{E}_{Wz\sim p_{Wz}(Wz)}[\log(1 - D(G(Wz)))] = \mathbb{E}_{x\sim p_G(x)}[\log(1 - D(x))]$$

where such a transformation facilitates the deduction of the optimal discriminator (i.e., $D^*(x) = \frac{p_r(x)}{p_r(x)+p_G(x)}$). More detailed deductions can be found in the original GAN paper [13]. Since the basis matrix W usually follows the Exponential distribution [6], such that Wz also follows the Exponential distribution.

Theorem 1. *Let W be a basis matrix and it is drawn from the original data with NMF. Let the noise code z be a low-dimensional vector and it comes from the uniform distribution with (0, 1). Then, the combination of Wz jointly forms a new distribution, and this new distribution follows Exponential distribution.*

Proof. Let $Sketch = Wz$ and we term *Sketch* as S, i.e., $S = Wz$. Then, the corresponding function is $F(S)$, so we get:

$$\begin{aligned} F(S) &= P(S \leqslant s) = P(UW \leqslant s) = P(W \leqslant \frac{s}{U}) \\ &= \int_0^1 P(W \leqslant \frac{s}{u}) f_U(u) du \\ &= \int_0^1 \int_0^{\frac{s}{u}} \lambda e^{-\lambda w} dw du \\ &= \int_s^\infty \lambda e^{-\lambda w} \frac{s}{w} dw + \int_0^s \lambda e^{-\lambda w} dw \\ &= \lambda e^{-\lambda} s \int_s^\infty \frac{1}{w} e^{-w} dw + \int_0^s \lambda e^{-\lambda w} dw \\ &= 1 - e^{-\lambda s} \end{aligned} \tag{5}$$

This shows that the proposed SEGAN still performs distribution transformation. Different from the vanilla GAN, it usually transforms the Gaussian

or Uniform into original data distribution, SEGAN transforms the Exponential into original data distribution. Then, we explain how to address the gradients vanishing problem durning training.

Study [2] has shown that if the dimension of the randomized space is less than that of raw data space $\mathcal{X}$, $G(z)$ will be a measure 0 in $\mathcal{X}$, which causes p_r and p_G having a negligible overlapping area, resulting in gradients vanishing. In SEGAN, the randomized space is formed by Wz where W is factorized from raw data X. Thus, the dimension of Wz is no longer less than that of $\mathcal{X}$. In this way, p_r and p_G have a non-negligible overlapping area, rendering that JS divergence works during training. We now state this properly in the following Theorem.

Theorem 2. *Let G: $Wz \rightarrow \mathcal{X}$ be a function composed by affine transformation, e.g., rectifiers or sigmoid. Then, the dimension of $G(Wz)$ is no longer less than that of $\mathcal{X}$. Therefore, $G(Wz)$ does not have a measure 0 in $\mathcal{X}$.*

Proof. In SEGAN, p_G is defined via sampling from $p_{Wz}(Wz)$ in which the raw data X ($X \in \mathbb{R}^{p \times n}$) is factorized into a basis matrix W ($W \in \mathbb{R}^{p \times r}$) and an representation matrix H ($h \in \mathbb{R}^{r \times n}$). Moreover, we set $sketch = Wz$ where z ($z \in \mathbb{R}^{r \times 1}$) is still from a simple, easy-to-sample distribution $p_z(z)$ (i.e., Uniform distribution) which is independent of $p_r(x)$. Note that we normalize the raw images into into the range of [0, 1] in the preprocess stage. In this way, the range of elements within H falls into such a specific range. When we repeatedly sample noise code z from the Uniform with [0, 1] and perform Wz operation during training, the sketches at least approach a raw data sample. Under such a scenario, the dimension of Wz is no less than that of X. Moreover, the details of our designed G are made up with $Conv$ and $Relu$, and such a combination does not reduce the dimension of Wz. Considering such a case, $G(Wz) = D_n A_n ... D_1 A_1 Wz$ where A_i indicates the activation function (i.e., $Relu$) and D_i indicates the $Conv$. $Conv$ operation extracts the linear information while $Relu$ function extracts the non-linear information. With that, these chunks (i.e., $D_i A_i$) focus on editing the sketch rather than changing the dimension of Wz. Hence, $G(Wz)$ keeps the same dimension as X. Since the dimension of $G(Wz)$ is no longer less than that of X in the space, $G(Wz)$ has a non-zero measure in $\mathcal{X}$. Under this case, p_r and p_G hold a non-negligible overlapping area, avoiding the gradients vanishing.

We present the pseudo-codes of the algorithm for training SEGAN in **Algorithm 1**. In **Algorithm 1**, SEGAN still plays the minimax game in a spirit to the vanilla GAN, and it has a global optimum on $p_G = p_r$. The

Algorithm 1 Sketch-then-Edit GAN.

Input:

Raw dataset;
noise z, $p_z(z)$;
r for the basis matrix W in NMF.

Output:

Synthetic data.

Creating sketches by using NMF to factorize the real dataset to get the basis matrix W and adopting *Adam* optimizer to update the parameters of the model.
for number of iterations **do**

- Sampling minibatch of m noise samples $z^1, ..., z^m$ from Uniform distribution with (0,1).
- Creating minibatch of m sketches $s^1, ..., s^m$ using Eq.(3).
- Sampling minibatch of m original samples $x^1, ...x^m$ from the real dataset.
- Updating the discriminator's parameters by ascending its stochastic gradient.
- $$\bigtriangledown_{\theta_D} \frac{1}{m} \sum_1^m \{logD(x^{(i)}) + log(1 - D(G(s^{(i)})))\}.$$
- Sampling minibatch of m noise samples $z^1, ..., z^m$ and creating minibatch of m sketches $s^1, ..., s^m$.
- Updating the generator's parameters by descending its stochastic gradient.
- $$\bigtriangledown_{\theta_D} \frac{1}{m} \sum_1^m \{log(1 - D(G(s^{(i)}))\}.$$

end for

optimal discriminator D is:

$$D_G^*(x) = \frac{p_r(x)}{p_r(x) + p_G(x)} \tag{6}$$

We substitute Eq.(6) into Eq.(4) to deduce the optimal G, and we get:

$$C(G) = \max_D V(G, D) = -2log2 + 2JSD(p_r(x)||p_G(x)) \tag{7}$$

We refer the proofs for optimal D and G to Eq.(6) and Eq.(7) in study [13]. In this way, the global optimality for D and G is achieved if and only if $p_G = p_r$, and the optimality of D is $\frac{1}{2}$ and that of G is -2log2 respectively.

5. Empirical Evaluation

5.1. Experiment settings

Four recent GAN variants are employed as baselines, which are **DCGAN** [1], **WGAN** [3], **LSGAN** [26] and **DeLiGAN** [14]. To make a fair comparison, the standard Uniform distribution is employed as the original noise distribution for all the models except that a Mixture of Gaussian distributions are for DeliGAN as required. For training all the GAN models, the batch size of is set as 20. For SEGAN, we set r, the number of basis features in NMF, as the number of categories of the studied datasets according to the literature of NMF [48] if no other values are specified. Note that all models are achieved by *PYTORCH* framework.

Four commonly used public image datasets, the MNIST, CIFAR-10, SVHN and Celeba, are studied. The MNIST dataset contains 50000 gray-scale images with size 28×28. CIFAR-10, SVHN and Celeba datasets contain RGB images. The number of images in the CIFAR-10 and SVHN is 50000 and the size is $3 \times 32 \times 32$. The number of images in the Celeba is 202599 and the size is $3 \times 178 \times 218$. The MNIST, CIFAR-10 and SVHN datasets belong to labeled datasets and each of them consists of 10 categories. The Celeba dataset is a face dataset, and it has no explicit categories because of containing the face images only. Nevertheless, the Celeba dataset has been labeled with the specific characters (e.g., wearing hat or not, wavy hair or bangs). One thus can group Celeba images depending on their preferences. Here, we also assume that there are 10 categories.

5.2. Generation Performance Comparison

In the experiments, the training has been set to the same number of epochs for all models (e.g., $Epoch = 10$ for the MNIST dataset and $Epoch = 30$ for the CIFAR10, SVHN and Celeba datasets) and the dimension of a noise input z for these GAN variants is set to 100. For SEGAN, we use the

Adam gradient method [18] and *BCE* [19] to update the parameters of both components. We set the parameter of *LeakyRelu* [40] as 0.02 and that of *Dropout* [32] as 0.5. The activation of last layer for the generator and the discriminator is *Sigmoid* [16]. The generated images are shown in Fig. 3. In Fig. 3, the generated images on the top (sub-figures (a) - (e)) are based on the MNIST dataset, the images on the second row (sub-figures (f)-(j)) are based on the CIFAR10 dataset, and the third row (sub-figures (k)-(o)) are based on SVHN dataset, the bottom images (sub-figures (p)-(t)) are based on the Celeba dataset. For each dataset, the first four images (sub-figures (a) - (d), (f) - (i), (k) - (n) and (p)-(s)) are from the baselines, and the last one (sub-figures (e), (j), (o) and (t)) is generated by SEGAN. Furthermore, the MNIST Scores and the SVHN Scores [17], similar to the Inceptions Score [4] (the higher the better), obtained by our proposed SEGAN and baselines are shown in Table.2. By comparison, we can see that SEGAN shows obviously better performance than baselines.

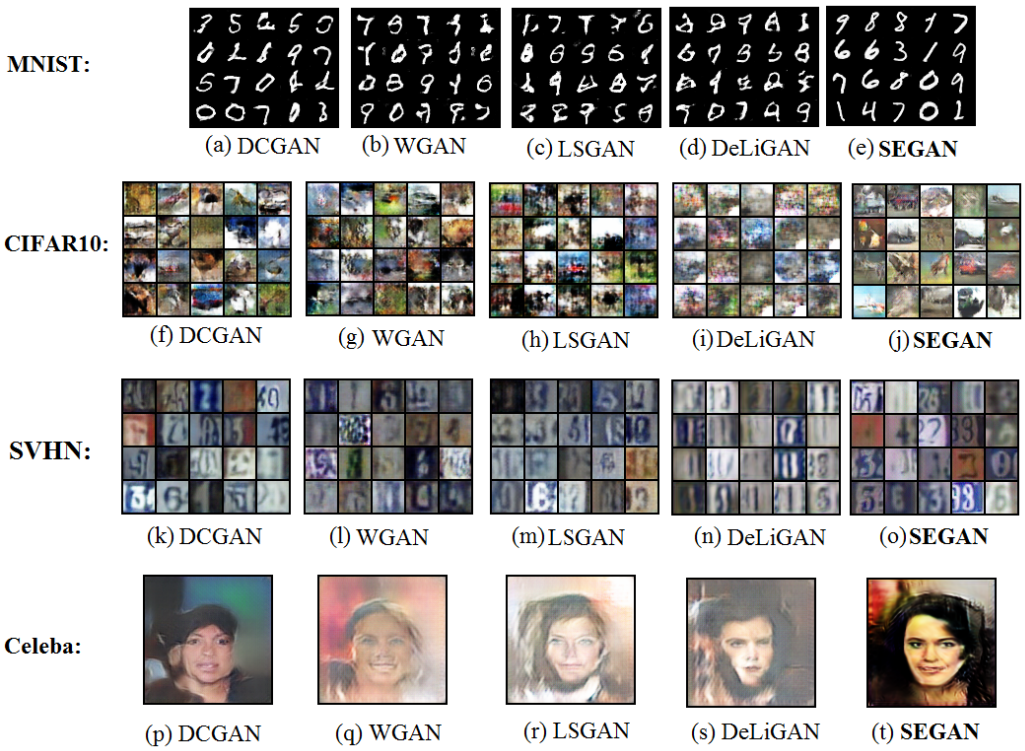


Figure 3: Generated images by baselines and SEGAN.

Models	MNIST Scores	Inception Scores	SVHN Scores
DCGAN	4.94	3.04	1.76
WGAN	4.87	2.50	1.79
LSGAN	5.10	1.92	1.80
DeLiGAN	5.12	1.64	1.84
SEGAN	**5.54**	**3.23**	**1.90**

Table 2: MNIST scores, Inception scores and SVHN scores obtained from baselines and SEGAN on MNIST, CIFAR10 and SVHN generated images shown in Fig.3

5.3. Addressing the Vanishing Gradient Problem

Now we empirically illustrate how well SEGAN tackles the vanishing gradient problem. We have known that the vanishing gradient problem is the consequence of a negligible overlapping area between p_r and p_G and a constant JS divergence for measuring the difference between the two distributions. Instead of replacing the difference measure, SEGAN innovatively creates the new noise input (i.e., sketch) which lies in the raw data space to guarantee a non-negligible overlapping area between p_r and p_G. For validating this point, we plot the distributions of the raw data and the generated data to observe the overlapping area on the MNIST dataset (two other datasets display a similar result and are thus omit). The overlapping area at different epochs is shown in Fig. 4. Sub-figure (a) shows that the generating distribution of SEGAN has a non-negligible overlapping area with the raw data distribution at $Epoch = 1$ while the other two have a negligible overlapping area with the raw data distribution (we omit WGAN for the sake of clearance because the generating distribution of WGAN has no essential difference from that of DCGAN). At $Epoch = 10$, sub-figure (b) shows that the generating distribution of SEGAN has the largest overlapping area with the raw data distribution. Hence, the claim that a non-negligible overlapping area can be guaranteed by transforming noises from a randomized space into the raw data space has been validated.

Moreover, we directly examine the gradients during the training, and present the values of gradients at different iterations in Fig. 4(c). It shows that the gradients of SEGAN are sufficient larger than zero even at the initial iterations while these of baselines are close to zero. For baselines, the gradients vanish because the generating distribution does not overlap the raw

data distribution in the initial stage and the discriminator can easily reach the optimal state [2].

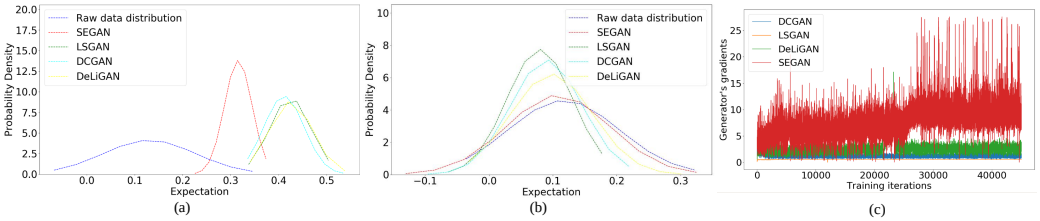


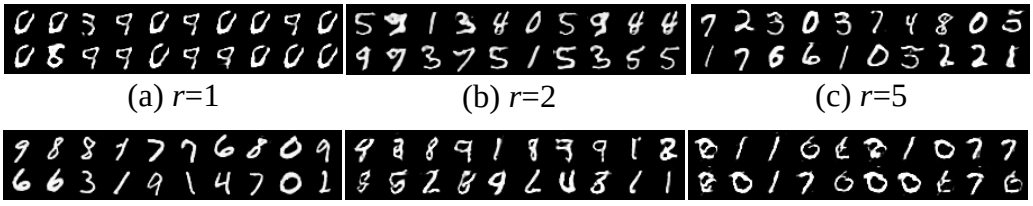
Figure 4: Sub-figures (a) and (b) indicate the overlapping area between p_r and p_G at $Epoch = 1$ and $Epoch = 10$ respectively. The blue line indicates the raw data distribution, and other lines indicate the generated data distributions. Sub-figure (c) indicates the gradients produced by the SEGAN and baselines.

5.4. The Influence of r on SEGAN

Note that the value r for the basis matrix in NMF has been set to 10 in previous experiments. And we have discussed in Section 4 the influence of different values of r on SEGAN. Here, we empirically study the influence by setting different values of r and examining the generated images on the MINST dataset (other two datasets show similar patterns). Six different values, which are 1, 2, 5, 10, 30, and 50, are selected and studied, and the corresponding generated images are presented in Fig. 5.

We see that the diversity of the generated images is poor when r takes small values (e.g., 1, 2). This is because when the number of basis features is small, the sketches which are the combinations of basis features tend to be similar. When r takes large values (e.g., 30 and 50), the quality of the generated images is poor. This is because the basis features may be too fine-grained and contain individual pixels of the raw data, which means that the sketches may be essentially not different from the original noises. Fig.6 shows the influence of different r on the generated data quality. In addition, we also present the basis features when r takes 5, 10 and 50 in Fig. 7 to visually demonstrate our hypothesis. We can see that each basis feature in Fig. 7 (a) and (b) corresponds to a salient feature of a digit, which is the expected result. But each basis feature in Fig. 7 (c) looks like trivial information. Moreover, it seems that there are many identical basis features (e.g., the first three rows). Therefore, many generated images may also be identical as shown in Fig. 5 (f). According to the study, we also recommend

that r should take a value around the number of categories of the given data as in the NMF literature. If the value of r is far away from the number of categories, the quality of generated data is not good as shown in Fig. 5 (a) (e) and (f).



(a) r=1 (b) r=2 (c) r=5

(d) r=10 (e) r=30 (f) r=50

Figure 5: The value of r in basis matrix has been set as 1, 2, 5, 10, 30 and 50, respectively. The six sub-figures are the corresponding results produced by SEGAN.

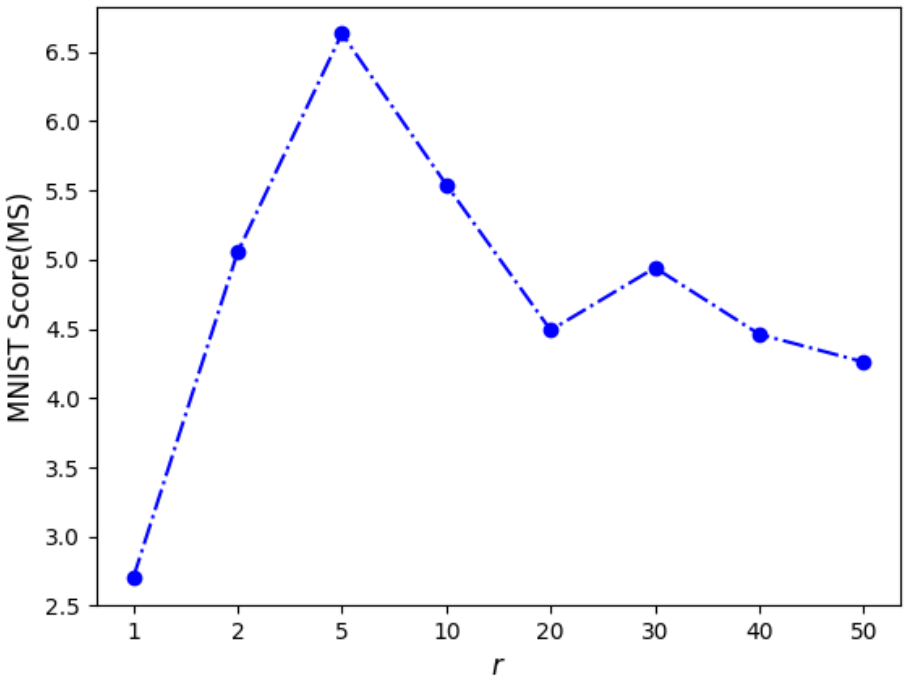
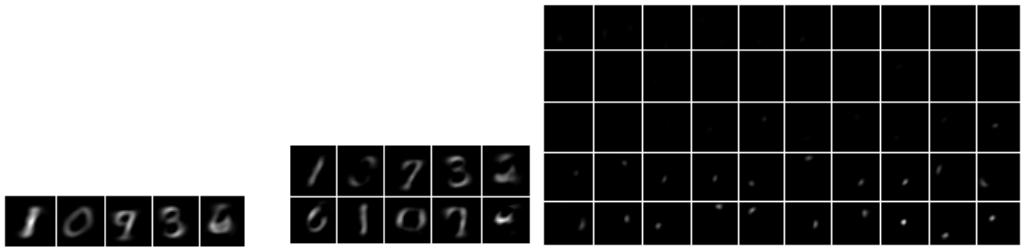


Figure 6: Different values of r have an influence on the generated data quality.



(a) r=5 (b) r=10 (c) r=50

Figure 7: Visualization of basis features when $r = 5$, $r = 10$ and $r = 50$, respectively, where each grid contains a feature.

5.5. Efficiency Analysis

As discussed in Section 4.4, SEGAN needs more convolution operations in generating a synthetic instance due to the increased size of an input. As the same time, since an input, i.e., a sketch, already has salient features of raw data, SEGAN needs less number of training epochs to reach the Nash equilibrium. Note that it is not easy to quantitatively compare the added computation costs and the reduced computation costs. This section thus empirically studies the efficiency of SEGAN.

In particular, we examine the quality of generated images at different epochs and compare the respective running time needed by SEGAN and DCGAN (other baselines have similar efficiency with DCGAN since they have the same generating mechanism). We study the MNIST and the Celeba dataset, and present the generated images at different epochs in Fig. 8 and Fig. 9, respectively. Through examining the quality of generated images, we have two major observations. First, the quality of images at the initial epoch of SEGAN is significantly better than DCGAN. We can see that the images at Epoch=1 are just noises by DCGAN while the images by SEGAN have meaningful outlines. Second, DCGAN needs several more epochs to generate images of similar quality with SEGAN. For example, the quality of the Celaba image at Epoch=10 by DCGAN is about similar to that of image at Epoch=2 by SEGAN.

According to the observations above, it is unfair to compare the running time of SEGAN with DCGAN if we assign the same number of epochs for the training. Instead, we need to compare the running time with which SEGAN and DCGAN generate images of similar quality. For example, we may compare the running time of SEGAN after 2 epochs with that of DCGAN after

10 epochs. For the MNIST dataset, the running time of SEGAN is about 0.32h and that of DCGAN is about 0.65h. For the Celaba dataset, the running time of SEGAN is 3.82h and that of DCGAN is 3.17h. In this sense, the efficiency of SEGAN is comparable to that of DCGAN. For a complete comparison, we still present the running time of SEGAN and DCGAN after the same number of epochs. The number of epochs for the MINST dataset and the Celeba dataset are set as 10 and 30, respectively, as studied in Fig. 3. For the MNIST dataset, the running time of DCGAN approximates to 0.65h while that of SEGAN nearly takes 1.59h. As for the Celeba dataset, the running time of DCGAN is about 3.17h while that of SEGAN approximates to 19.04h. Hence, we can see that the running time needed by SEGAN is only several times larger than that needed by DCGAN. Note that all experiments of this study are conducted on a Geforce 1080TI GPU.

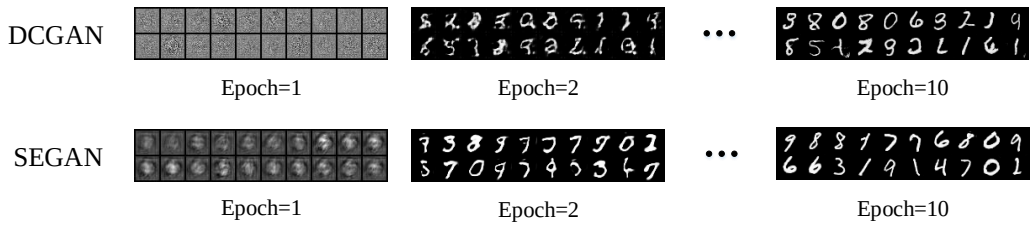


Figure 8: The two sets of results show the quality of generated MINST images on different training epochs by DCGAN and SEGAN, respectively. Apparently, SEGAN generates images with higher quality than DCGAN at each epoch.

6. Conclusion

In this paper, to deal with the vanishing gradient problem from the perspective of the negligible overlapping area between the generating distribution and the raw data distribution, we propose the sketch-then-edit GAN (SEGAN). The major novelty of SEGAN is the input into the generator which is sketches instead of the original noises. In particular, each sketch is a multiplication of two factors, the basis features W, which is drawn from the raw data using the NMF method, and z, which is sampled from the Uniform distribution with (0, 1). In this way, the sketch holds a similar distribution of the raw data such that a non-negligible overlapping area could be guaranteed. In addition, for editing the sketch, we modify the generator

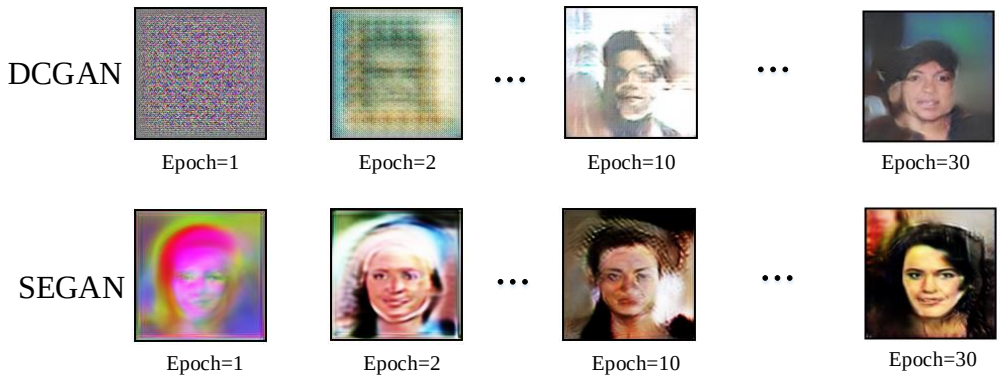


Figure 9: The two sets of results show the quality of generated Celeba images on different training epochs by DCGAN and SEGAN, respectively. Apparently, SEGAN generates images with higher quality than DCGAN at each epoch.

by replacing the fractionally-strided convolutions with the standard convolutions. We conducted extensive experiments with the MNIST, the CIFAR10, the SVHN and the Celeba datasets to validate our proposed model. These results show that our approach is effective.

Acknowledgment

This work was supported in part by the National Key R&D Program of China (No.2018YFC1604000), in part by the RGC Collaborative Research Fund (CRF) 2018/19 - Group Research Grant (RGC No.:C5026-18G), and in part by RGC General Research Fund (GRF) 2017/18 (B-Q61D RGC No.:PolyU 152199/17E).

[1] Luke Metz Alec Radford and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

[2] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[5] David Berthelot, Thomas Schumm, and Luke Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[6] Thomas Brouwer and Pietro Lio. Prior and likelihood choices for bayesian matrix factorisation on small datasets. *arXiv preprint arXiv:1712.00288*, 2017.

[7] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018.

[8] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[9] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. *The complexity of computing a Nash equilibrium.* ACM, 2009.

[10] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[11] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[12] Nicolas Gillis. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12(257), 2014.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[14] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *CVPR*, pages 4941–4949, 2017.

[15] Paul R Halmos. *Measure theory*, volume 18. Springer, 2013.

[16] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.

[17] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. *arXiv preprint arXiv:1811.03850*, 2018.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Dirk P Kroese, Reuven Y Rubinstein, Izack Cohen, Sergey Porotsky, and Thomas Taimre. Cross-entropy method. In *Encyclopedia of Operations Research and Management Science*, pages 326–333. Springer, 2013.

[20] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[21] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[22] Te-Won Lee. Independent component analysis. In *Independent component analysis*, pages 27–66. Springer, 1998.

[23] Wei Li, Wei Ding, Rajani Sadasivam, Xiaohui Cui, and Ping Chen. His-gan: A histogram-based gan model to improve data generation quality. *Neural Networks*, 119:31–45, 2019.

[24] Wei Li, Pengqiu Meng, Yi Hong, and Xiaohui Cui. Using deep learning to preserve data confidentiality. *Applied Intelligence*, pages 1–13, 2019.

[25] Willy Malfliet and Willy Hereman. The tanh method: I. exact solutions of nonlinear evolution and wave equations. *Physica Scripta*, 54(6):563, 1996.

[26] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821. IEEE, 2017.

[27] Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Cross-domain generative learning for fine-grained sketch-based image retrieval. In *British Machine Vision Conference (BMVC)*, volume 82, page 85, 2017.

[28] Gert K Pedersen, Masamichi Takesaki, et al. The radon-nikodym theorem for von neumann algebras. *Acta Mathematica*, 130:53–87, 1973.

[29] Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264*, 2017.

[30] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[31] Ludger Rüschendorf. The wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, 1985.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[33] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[34] S Verdú. Total variation distance and the distribution of relative information. In *Information Theory Applications Workshop*, 2014.

[35] Jing Wang, Feng Tian, Hongchuan Yu, Chang Hong Liu, Kun Zhan, and Xiao Wang. Diverse non-negative matrix factorization for multiview data representation. *IEEE transactions on cybernetics*, 48(9):2620–2632, 2018.

[36] Ruohan Wang, Antoine Cully, Hyung Jin Chang, and Yiannis Demiris. Magan: Margin adaptation for generative adversarial networks. *arXiv preprint arXiv:1704.03817*, 2017.

[37] Senzhang Wang, Jiannong Cao, Hao Chen, Hao Peng, and Zhiqiu Huang. Seqst-gan: Seq2seq generative adversarial nets for multi-step urban crowd flow prediction. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*.

[38] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics Intelligent Laboratory Systems*, 2(1):37–52, 1987.

[39] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

[40] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *Computer Science*, 2015.

[41] Linchuan Xu, Jiannong Cao, Xiaokai Wei, and Philip Yu. Network embedding via coupled kernelized multi-dimensional array factorization. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[42] Shan Yang, Lei Xie, Xiao Chen, Xiaoyan Lou, Xuan Zhu, Dongyan Huang, and Haizhou Li. Statistical parametric speech synthesis using generative adversarial networks under A multi-task learning framework. *CoRR*, abs/1707.01670, 2017.

[43] Qianyi Zhan, Jiawei Zhang, Philip Yu, and Junyuan Xie. Community detection for emerging social networks. *World Wide Web*, 20(6):1409–1441, 2017.

[44] Ning Zhang, Evan Shelhamer, Yang Gao, and Trevor Darrell. Fine-grained pose prediction, normalization, and recognition. *arXiv preprint arXiv:1511.07063*, 2015.

[45] Yuxuan Zhang, Senzhang Wang, Bing Chen, Jiannong Cao, and Zhiqiu Huang. Trafficgan: Network-scale deep traffic prediction with generative adversarial nets. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[46] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[47] Shizhan Zhu, Sanja Fidler, Raquel Urtasun, Dahua Lin, and Chen Change Loy. Be your own prada: Fashion synthesis with structural coherence. *arXiv preprint arXiv:1710.07346*, 2017.

[48] Linlin Zong, Xianchao Zhang, Long Zhao, Hong Yu, and Qianli Zhao. Multi-view clustering via multi-manifold regularized non-negative matrix factorization. *Neural Networks*, 88:74–89, 2017.