**Mathematics and Computer Science**
2020; 5(1): 31-41
http://www.sciencepublishinggroup.com/j/mcs
doi: 10.11648/j.mcs.20200501.14
ISSN: 2575-6036 (Print); ISSN: 2575-6028 (Online)

# A Survey of Generative Adversarial Networks Based on Encoder-Decoder Model

**Yi Yin[1, *], Lin Ouyang[1], Zhixiang Wu[1], Shuifang Yin[2]**

[1]School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, P. R. China

[2]School of Science, Wuhan University of Science and Technology, Wuhan, P. R. China

## Email address:

yinyi@wust.edu.cn (Yi Yin), ouyanglin@wust.edu.cn (Lin Ouyang), wuzhixiang@wust.edu.cn (Zhixiang Wu), yinsf_wkd@163.com (Shuifang Yin)

[*]Corresponding author

## To cite this article:

Yi Yin, Lin Ouyang, Zhixiang Wu, Shuifang Yin. A Survey of Generative Adversarial Networks Based on Encoder-Decoder Model. *Mathematics and Computer Science*. Vol. 5, No. 1, 2020, pp. 31-41. doi: 10.11648/j.mcs.20200501.14

**Received**: January 17, 2020; **Accepted**: February 11, 2020; **Published**: March 10, 2020

**Abstract:** The generative model is a very important type of model in the field of artificial intelligence in recent years. Such models can comprehend the data through the neural network, and then create data according to the probability distribution of the input data, predict the results according to the data characteristics. The whole processing process is "intelligent". At present, two typical applications of generative model are Generative Adversarial Networks (GAN) model and Encoder-Decoder model, which have strong ability to generate image data. In the model of GAN, the generator simulates real data, and the discriminator judges the authenticity of the samples. Its goal is to train a generator to fit the real data perfectly, so that the discriminator cannot distinguish. In the Encoding-Decoding model, it can be understood as a process of "Encoding→intermediate vector→Decoding", It is suitable for processing one kind of data to generate another kind of data with the same probability distribution as the original data. Since most of the data features are intermingled, they are encoded in a complex and disorderly way. But if these features are extractable, it shows that these features are interpretable, and it will be easier to use these features for coding. Based on this situation, some research results have been produced by incorporating the Encoder- Decoder into GAN. This paper systematically analyzes and summarizes the basic concepts and theories of GAN and Encoder-Decoder, as well as their respective advantages and disadvantages. On this basis, by combing the related work of the two types of models, the main technical routes of the three types of GAN based on the Encoder Decoder are summarized, the techniques and theories including variational inference, energy function and correlation transformation of different distribution data are summarized. Finally, the GAN based on the Encoder-Decoder is summarized.

**Keywords:** Generative Adversarial Networks, Encoder-Decoder, Variational Inference, Energy Function

## 1. Introduction

In recent years, with the remarkable improvement of computer processing ability and the explosive growth of data in industries, the rapid development in the field of deep learning knowledge has been brought forward. Among them, applying generative model to data processing is the most promising method at present [1, 2]. The Generative Adversarial Networks (GAN) model [3] and the Encoder-Decoder model [4] are two successful applications built on the generative model.

The GAN model consists of a Generator and a Discriminator. The Generator aims to learn the real data distribution as much as possible. The purpose of Generator is to learn the real data distribution as much as possible. The purpose of Discriminator is to try to determine whether the input data comes from real data or from generated noise data. In order to improve the generation ability of Generator, the generated data is not discriminated by Discriminator. In order to improve its discriminating ability, Discriminator can more accurately judge whether the input data is from real data or generated noise data. Both need to be continuously optimized to reach the Nash equilibrium state [3, 5].

GAN can effectively solve the problem of the generation of interpretable data, especially for high- dimensional data, the

neural network method used does not limit the dimension of the data, nor does it limit the data type, which greatly broadens the scope of sample selection for generating data. At the same time, the neural network can integrate a variety of loss functions and increase the degree of freedom of design [6].

GAN transforms a random noise vector from a probability distribution to a probability distribution of the real data set. GAN creatively trains two neural networks during the training process. The training process does not require approximate inference, which improves the training difficulty and improves the training efficiency.

Encoder-Decoder is a kind of framework, end-to-end is one of its most prominent features [7]. Encoder denotes that the input data is converted into a fixed-length intermediate vector, and Decoder denotes that the previously generated intermediate vector is converted into output data. Encoder and Decoder are two kinds of neural network models, which can adopt arbitrary neural network algorithm according to the application requirements.

In Encoder-Decoder framework, the dimension of the intermediate vector is generally much smaller than that of the input data. Therefore, Encoder can be used to reduce the dimension. Then, the noise of the intermediate vector can be increased and the framework can be trained to restore the original input data as much as possible [8]. In this way, the intermediate vector has deep features that the original input data can extract.

Intuitively, the two models have obvious advantages and disadvantages compared with each other. GAN does not need pre-modeling, that is, it does not need a pre-hypothesized data distribution, and the loss function design is easy. As long as there is a standard, it can be handed to the discriminator for confrontation training, and finally the generated data can be obtained. The final generated data and the original data of Encoder-Decoder need to be the same distribution. However, GAN cannot directly compare the difference between the generated data and the original data, but Encoder- Decoder can do these.

Based on this, scholars have put forward some methods of combining GAN with Encoder- Decoder and realized some applications. This paper revolves around the current mainstream application of Encoder-Decoder to GAN. Chapter 1 outlines the basic theories and concepts of the two models. Chapter 2 combs the variable-inference method for applying Encoder-Decoder to the Generator structure based on probability theory. Chapter 3 introduces the energy function method of applying Encoder-Decoder to Discriminator structure based on energy model. Chapter 4 introduces the association transformation method that combines GAN and Encoder-Decoder to different distributed data. Chapter 5 summarizes these methods.

## 2. Basic Concepts

The theory and model of GAN were proposed by Ian Goodfellow et al. In GAN, any differentiable function can be expressed as Generator and Discriminator, which are represented here by differentiable functions $G$ and $D$. The noise data is represented by $z$, and $G(z)$ represents the data generated by $G$ as far as possible from the probability distribution of the real data $x$. Generator continuously learns the probability distribution of real data, and the goal is to optimize the input random noise to fit the probability distribution of real data, which makes the Discriminator indistinguishable. Discriminator judges whether the received data is real data or the data $z'$ generated by the Generator through the noise data $s$, and distinguishes the "true" and "false" data generated by the generated model. The goal of $D$ here is to realize the two-class discrimination of data sources. If the data comes from real data $x$, it is judged as "true" and marked as 1, and if it is derived from $G(z)$, it is judged as "false" and marked as 0. The definition $D(z)$ is defined to represent the probability distribution $p_z(z)$ of the input $G(z)$ in the generated model, $D(x)$ is defined to represent the probability distribution $p_{data}(x)$ of $x$ from real data, and its optimized loss function is defined as:

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

and:

$$G^* = arg\ \min_G \max_D V(D,G) \quad (2)$$

The goal of the Generator is to make its own generated data $G(z)$ consistent with the real data $x$ when it is discriminated by the Discriminator.

When updating the parameters of the discriminative model, it hopes $logD(x)$ to be as large as possible from real data. For the $G(z)$ generated from noise data, the larger the $\log(1 - D(G(z)))$ is, the better the max$D$ needs to be solved. When updating the parameters of the generative model, it hopes $D(G(z))$ to be as large as possible. At this time, $V(D,G)$ will become smaller, that is, min$G$ is required for generating the model. These two confrontation and optimization processes make the performance of Generator and Discriminator improve constantly. The final ideal state is that when Discriminator's discriminating ability cannot correctly determine the data source, it can be considered that Generator has learned the real data distribution. The basic structure of GAN is shown in Figure 1.

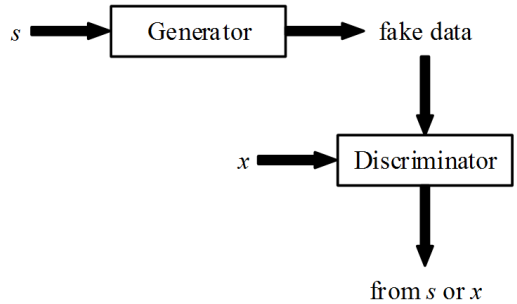


***Figure 1.*** *The structure of GAN. The noise data is generated by Generator, and the Discriminator is used to determine whether the data generated by Generator or the original real data.*

The idea and structure of Encoder-Decoder was proposed by Ilya Sutskever et al [9]. Encoder-Decoder can be understood as the process of "encoding→intermediate vector →decoding". The input data $x$ of the Encoder-Decoder structure is encoded by the Encoder to obtain the code vector $z$, Then the output data $\tilde{x}$ is obtained after Decoder processing. Neural network algorithms are often used here as the structure of the Encoder and Decoder. The model expects the input vector $x$ to obtain an intermediate vector $z$ through one neural network algorithm in the Encoder, which is a dimensionality reduction process. Then, another neural network in the decoder is used to Decode, and the dimensional reconstruction is added to obtain the generated data $\tilde{x}$ similar to the input data. By comparing the two sets of data $x$ and $\tilde{x}$, and minimizing the difference between them, the parameters of the encoder and decoder are trained. The basic framework of Encoder-Decoder is shown in Figure 2.

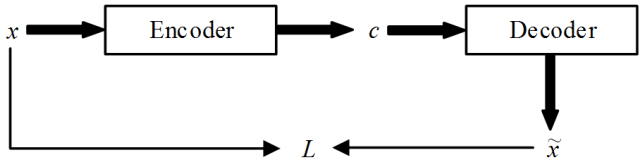


***Figure 2.*** *The structure of Encoder-Decoder. Data conversion based on a set of input sequences to generate another set of output sequence.*

Encoder-Decoder is an end-to-end model. It is a process of continuously improving the similarity $\mathcal{L}$ between $\tilde{x}$ and $x$ in training.

$$\begin{cases} E(x): x \to z \\ D(x): z \to \tilde{x} \\ z = \sigma(Wx + b) \\ \tilde{x} = \sigma'(W'c + \mathrm{b}') \\ \mathcal{L} = \min \parallel x - \tilde{x} \parallel^2 \end{cases} \tag{3}$$

Because the Encoder-Decoder model and GAN model are based on different theories, the data processing methods and the results will be different.

(1) The generated data in GAN comes from some random noise points, so it is possible to cause mode collapse. If GAN wants to generate some specific details, it must traverse the entire distribution of the input data to determine which part of the input data determines the details [10]. The Encoder-Decoder can obtain the same type of features as the input data through the encoding process of the output data, and then generate the desired content through specific noise, so that the generated data and input data have the same probability distribution [11].

(2) The training of GAN model needs to reach Nash equilibrium, and the method of gradient descent can guarantee the realization of Nash equilibrium only under the premise of convex function, and it is not easy to converge when the input data is discrete or sparse [12]. Although some GAN methods have been able to solve these problems, GAN is less trainable than Encoder-Decoder.

(3) GAN uses an Adversarial Networks, and Encoder-Decoder optimizes not the likelihood itself, which results in much less identifiable data generated by Encoder-Decoder than data generated by GAN [13].

(4) If the discriminator in the GAN is well trained, the generator can perfectly learn the distribution of the training samples, that is, GAN is progressively consistent, while Encoder-Decoder is biased.

Based on this, some GAN models based on Encoder-Decoder are obtained by combining the advantages of Encoder-Decoder and GAN.

# 3. Variational Inference Model

Considering GAN as a Possibility Based Model (PBM) is a commonly recognized method [13]. Since the essence of Discriminator is to calculate the conditional probability $p(x|y)$ that $x$ belongs to a certain class $y$, and the essence of Generator is to calculate the joint probability $p(xy)$ of $x$ in the whole distribution. For PBM, Generator is the core part, because in PBM theory, Generator was designed before Discriminator, and then the divergence between positive and negative samples was missing in the calculation of Generator. Discriminator must learn to calculate this divergence to assist Generator, this also makes the structure design of the Generator in PBM relatively more complicated [14]. Therefore, the Encoder-Decoder model can be applied to the Generator structure based on the PBM theory.

## *3.1. Variational Self-coding*

Assuming that there is a distribution of real random variables, input data $x$ can be considered as samples extracted from the whole distribution, but the distribution of real data is unknown. John Paisley et al. [15] put forward an idea of approximating the distribution by controllable and known distribution, assuming that Z obeys some common distribution, such as normal distribution or uniform distribution, and then wished to train a model $\tilde{x} = g(z)$, and use the distribution of the model to approximate the distribution of real data, that is, by transforming the parts to make the two distributions overlap as much as possible. This is the idea of Variational Auto Encoder (VAE).

Anders Boesen Lindbo Larsen et al. [16] combined VAE and GAN to form VAEGAN. Here, VAE includes Encoder and Decoder, and GAN includes Generator and Discriminator. In VAE, $z$ is generated by $x$ through Encoder and $\tilde{x}$ by $z$ through Decoder. There are $z \sim \mathrm{Enc}(x) = q_\phi(z|x)$, $\tilde{x} \sim \mathrm{Dec}(z) = p_\theta(x|z)$. The Generator in GAN can be regarded as the Decoder in VAE. After converting $z$ into $\tilde{x}$, the $\tilde{x}$ produced by discriminator is “true” or “false”, and gives a “score”. Combining VAE and GAN, it can be seen that the discrimination effect of GAN is better than VAE, but the training process of VAE is easier than GAN.

Suppose that the input vector $x$ is encoded into an implicit variable $z$, and then decoded back to the output vector $\tilde{x}$. VAE wants to generate $z^{(i)}$ from a prior distribution $p_{\theta^*}(z)$ and then generate $x^{(i)}$ from a conditional distribution $p_{\theta^*}(x|z)$. However, the real parameter $\theta^*$ and the hidden variable $z^{(i)}$ are unknown, so a general solution is to use the maximum likelihood estimation to solve the unknown parameters. However, it is difficult to get $\theta$ either by solving

$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)\mathrm{d}z$ or by solving $p_\theta(z|x) = p_\theta(x|z)\,p_\theta(z)/p_\theta(x)$, so it needs to be solved by the variational Bayesian method.

$q_\phi(z|x)$ is introduced to approximate the true posterior distribution $p_\theta(z|x)$, and the distance from $q_\phi(z|x)$ to $p_\theta(z|x)$ is measured by KL divergence, which is denoted as $D_{KL}[q_\phi(z|x) \parallel p_\theta(z)]$, then $L_{\mathrm{VAE}}(\phi,\theta;x) = -D_{KL}[q_\phi(z|x) \parallel p_\theta(z)] + \mathbb{E}_{q(z|x)}[\log p_\theta(x|z)]$. Assuming that $p_\theta(z)$ is a known prior Gauss distribution function, $p_\theta(z)\sim N(0,I)$, then $q_\phi(z|x) = N(z;u(x,\phi),\sigma^2(x,\phi))$, then,

$$L_{REG}(z;\mu,\sigma) = D_{KL}[q_\phi(z|x) \parallel p_\theta(z)] \tag{4}$$

$$= \int q_\phi(z|x)\log p(z)dz - \int q_\phi(z|x)\log q_\phi(z|x)dz \tag{5}$$

$$= \int N(z;u,\sigma^2)\log N(z;0,1)dz - \int N(z;u,\sigma^2)\log N(z;u,\sigma^2)dz \tag{6}$$

$$= \frac{1}{2}(-log2\pi - (u+\sigma^2) + log2\pi + (1+\log\sigma^2)) \tag{7}$$

$$= \frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2 - \sigma_j^2 - \mu_j^2) \tag{8}$$

For $\mathbb{E}_{q(z|x)}[\log p_\theta(x|z)]$, the Monte Carlo method can be used to obtain,

$$\mathbb{E}_{q(z|x)}[\log p_\theta(x|z)] \approx \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x|z^{(l)}) \tag{9}$$

Where $z^{(l)} = u + \sigma\cdot\varepsilon^{(l)}$, $\varepsilon^{(l)}\sim N(0,I)$.
Finally, the loss function of VAE is,

$$\mathcal{L}_{\mathrm{VAE}} = \frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2 - \sigma_j^2 - \mu_j^2) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x|z^{(l)}) \tag{10}$$

The loss function of VAE is,

$$\mathcal{L}_{V(D,G)} = \log(Dis(x)) + \log(1 - Dis(Dec(z))) + \log(1 - Dis(Dec(Enc(x)))) \tag{11}$$

So, the loss function of VAE-GAN is,

$$\mathcal{L} = \mathcal{L}_{\mathrm{VAE}} + \mathcal{L}_{\mathrm{GAN}} \tag{12}$$

Initially, Generator is similar to the VAE. A vector is randomly generated, and then the Discriminator determines the true and false. After that, the Discriminator is fixed, the gradient descent is used to update the Generator parameters so that the Discriminator output is as close as possible to 1.

The basic framework of VAEGAN is shown in Figure 3.

### *3.2. Variational Mutual Information*

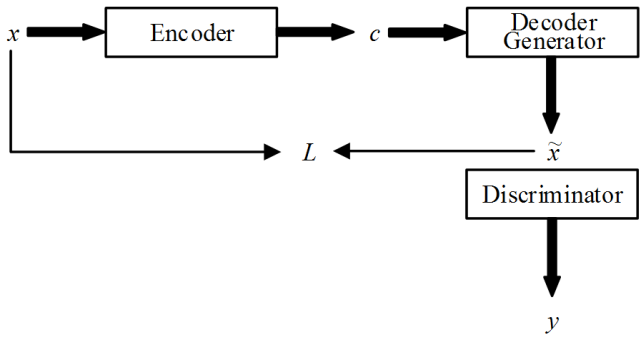


***Figure 3.*** *The structure of VAEGAN. After the input data is processed by Encoder-Decoder, the similarity between the generated data and the input data is compared, then the trained Decoder is used as the Generator in the GAN, and the quality of the generated data is improved by the Discriminator detection.*

From the perspective of feature learning, GAN does not impose any conditional restrictions on the input random noise, which means that there is no obvious feature representation of any dimension of $x$, so it is impossible to determine what kind of features can be generated by the noise dimension. Xi Chen et al. [17] improved the objective function of GAN and proposed a new GAN model -- InfoGAN, which represents mutual information. InfoGAN adds an implicit coding $c$ to the input vector of Generator and uses mutual information to indicate the degree of correlation between $x$ and $c$. In order to express the close relationship between $x$ and $c$, it is necessary to maximize the value of mutual information.

Furthermore, the objective function of GAN needs to add mutual information of $x$ and $c$ as a regularization term. Thus, based on the loss function of the original GAN, a regularization constraint is added: $I(c;G(x,c))$. Then there,

$$\min_G \max_D V(D,G) = V(D,G) - \lambda I(c;G(x,c)) \tag{13}$$

In fact, it is difficult to directly maximize $I(c;G(x,c))$. This involves a posterior probability distribution $p(c|x)$, which is difficult to obtain in practice. At the same time, because the generator has a high degree of freedom, it is possible to find an analytical solution during the learning process, so that $p(c|x) = p(x)$, which causes $c$ to lose its proper function. Therefore, we need to define an auxiliary probability distribution $q(c|x)$ to approximate $p(c|x)$ and obtain the lower bound of $p(c|x)$. This method is called variational mutual information maximization.

$$I(c;G(x,c)) = -H(c|G(x,c)) + H(c) \tag{14}$$

$$= \mathbb{E}_{x\sim G(x,c)}[\mathbb{E}_{c'\sim p(c|x)} \log p(c'|x)] + H(c) \tag{15}$$

$$= \mathbb{E}_{x\sim G(x,c)}[\underbrace{D_{KL}(p(\cdot|x) \parallel q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c'\sim p(c|x)} \log q(c'|x)] + H(c) \tag{16}$$

$$\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim p(c|x)} \log q(c'|x)] + H(c) \tag{17}$$

$$= \mathbb{E}_{c\sim p(c),x\sim G(z,c)}[\log q(c|x)] + H(c) \tag{18}$$

$$\triangleq L_I(G,Q) \tag{19}$$

When $H(c)$ is maximum, the maximum value of mutual information is obtained. The objective function is equivalent to:

$$\min_G \max_D V(D,G) = V(D,G) - \lambda L_I(G,Q) \tag{20}$$

InfoGAN model uses DCGAN as its network structure [18, 19]. Input $c$ and $z$ to Generator, and input the generated data and real data randomly to Discriminator for judgment. $Q$ and D share the convolution layer. The probability distribution of $Q$ output C. Here G and $Q$ can be seen as Encoder-Decoder structures, and $G$ and $D$ are models in GAN. The basic framework of InfoGAN is shown in Figure 4.

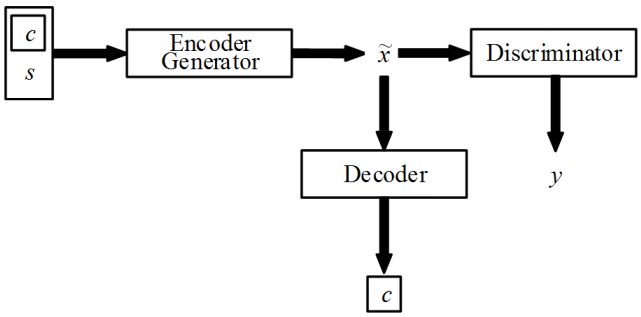


***Figure 4.*** *The structure of InfoGAN. Add an implicit encoding in addition to the noise data and maximize their mutual information.*

### *3.3. Adversary Inference*

The ALI (Adversarially Learned Inference) method proposed by Vincent Dumoulin et al. [20] and the BiGAN (Bidirectional GAN) method proposed by Jeff Donahue et al. [21] use an adversarial method to jointly train a generation network as Encoder and an inference network as Decoder. Encoder maps $x$ to a data space and encodes it to get $\hat{z}$. Decoder decodes the $z$ inverse map called the latent variable and get $\tilde{x}$. D needs to recognize $(x,\hat{z})$ and $(\tilde{x},z)$, judging that it is generated by Encoder or Decoder. Here, $z = \mu(x) + \sigma(x)\cdot\epsilon, \epsilon\sim N(0,I), q(z|x) = N(u(x),\sigma^2(x)I)$.

In ALI, Encoder and Decoder work independently, and their mapping and sampling processes are performed independently. They generate joint distributions respectively, and then give them to Discriminator to determine whether they are the same distribution. The joint distribution of the Encoder is $q(x,z) = q(x)q(z|x)$, and the joint distribution of the Decoder is $p(x,z) = p(z)p(x|z)$. When $q(x,z)$ and $p(x,z)$ match, all $q(x)$ and $p(z)$ matches and all $q(z|x)$ and $p(x|z)$ matches can be assumed. Thus, the loss function is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{q(x)}[\log(D(x,G_z(x)))] + \mathbb{E}_{p(z)}[\log(1 - D(G_x(z),z))] \tag{21}$$

$$= \iint q(x)q(z|x)\log(D(x,z))\,dxdz + \iint p(z)p(x|z)\log(1 - D(x,z))dxdz \tag{22}$$

The basic framework of ALI is shown in Figure 5.

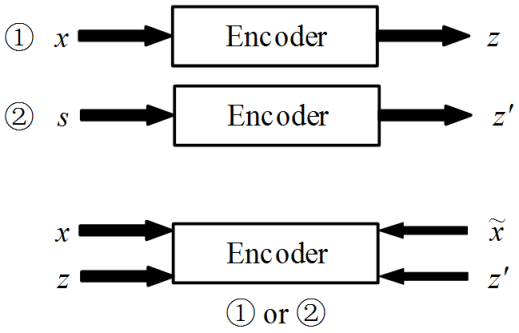


***Figure 5.*** *The structure of ALI/BiGAN. The input data is mapped by the Encoder to obtain the latent variable, and the noise data is converted into generated data by the Generator, and the two sets of data are discriminated by the Discriminator.*

### *3.4. Adversary Inference*

The IntroVAE method proposed by Huaibo Huang et al. [22] introduces GAN into the VAE, ignores the Discriminator structure in GAN and the Decoder structure in AutoEncoder, and achieves a Self-introspective learning. That is, the model itself can judge the quality of its generated samples and make improvements to improve the performance of the model.

$$L_{AE} = -E_{q_\phi(z|x)} log p_\theta(x|z) \tag{23}$$

$$L_{REG} = D_{KL}(q_\phi(z|x) \parallel p(z)) \tag{24}$$

Here, $L_{AE}$ is a loss function.

IntroVAE is implemented by training Encoder to make the hidden variables of real image close to the prior distribution, while the hidden variables of fake image far away from the prior distribution. At the same time, training generator can make the hidden variables of fake image close to the prior distribution. That is:

$$L_e = L_{REG}(z) + \alpha \sum_{s=r,p}[m - L_{REG}(z_s)]^+ + \frac{\beta}{2}\sum_{i=1}^{n} \parallel x_{r,i} - x_i \parallel_F^2 \tag{25}$$

$$L_G = \alpha \sum_{s=r,p} L_{REG}(Enc(x)) + \beta L_{AE}(x) \tag{26}$$

Here, $[\cdot]^+ = \max(0,\cdot)$, $\alpha$ and $\beta$ are superparametric, and $x_r$ is the reconstructed sample.

Unlike GAN and AutoEncoder [23], Encoder and generator in IntroVAE are adversarial, but they also ensure that the error between the output image and the real image is as small as possible. For real data, this method is consistent with the traditional VAE method and keeps the stability of AutoEncoder. For fake data, this method of confrontation improves the quality of the generated images.

# 4. Energy Function Model

Considering GAN as an energy-based model (EBM) is another design idea of GAN [24]. It treats the Discriminator as an energy function with no explicit probability interpretation and is trained to assign low energy values to regions of high energy values. The EBM is based on the construction of the Discriminator, and the Discriminator requires a negative sample in the calculation process, which is then provided by the Generator. This shows that the structure design of Discriminator in EBM is more abundant. Therefore, the Encoder-Decoder model can be applied to the Discriminator structure based on EBM theory.

## *4.1. Energy Function*

The EBGAN (Energy-Based GAN) method proposed by Junbo Zhao et al. [25] applies the concept of energy function to the Discriminator, and the Discriminator uses the Encoder-Decoder structure. $D$ is regarded as an energy function, which gives low energy to real data and high energy to generated data. Finally, a mean square error including real and fake data is output through Discriminator. The loss functions of Generator and Discriminator are:

$$f_D(x,z) = D(x) + [m - D(G(z))]^+ \tag{27}$$

$$= \| Dec\big(Enc(x)\big) - x \| + [m - \| Dec(Enc(G(z))) \|]^+ \tag{28}$$

$$f_G(z) = \| D(G(z)) \| = \| Dec(Enc(G(z))) - G(z) \| \tag{29}$$

Where $[\cdot]^+ = \max(0,\cdot)$ means that in $f_D$, when the reconstruction error of the fake data is less than a certain value $m$, $[\cdot]^+$ is positive, otherwise it is 0. Regarding the pattern collapse problem of GAN, EBGAN uses the Pulling-away Term method to generate diverse data from Generator:

$$f_{PT}(S) = \frac{1}{N(N-1)} \sum_i \sum_{j \neq i} \left( \frac{S_i' S_j}{\|S_i\| \|S_j\|} \right)^2 \tag{30}$$

The idea of EBGAN is that the generated fake data will generate a vector after Encoder. The cosine similarity between each two vectors is calculated separately, and then the mean value is calculated. The closer the two vectors are orthogonal, the smaller the value of $f_{PT}(S)$.

In EBGAN, Generator is based on Oakam's razor principle, while Discriminator is based on giving higher energy to the sample. The basic framework of EBGAN is shown in Figure 6.

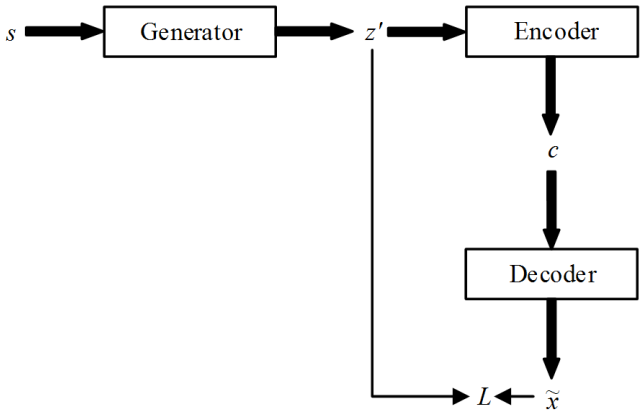


***Figure 6.*** *The structure of EBGAN. If the input data comes from real data, the data after decoding is basically lossless. If the input data comes from noise, the decoded data is quite different from the original data.*

## *4.2. Boundary Equilibrium*

David Berthelot et al. applied the encoder to the Discriminator in GAN and proposed BEGAN (Boundary Equilibrium GAN) [26]. Different from other GAN methods, most of them start with reconstructing the sample distribution, BEGAN starts from the perspective of reconstruction error distribution, it matches the loss between real samples and generated samples in self-encoder based on Wasserstein Distance.

The loss function between the input data in Discriminator passing through Encoder and the output data generated by Decoder is as follows:

$$\mathcal{L}(v) = |v - D(v)| \tag{31}$$

This formula is used to measure the difference between the two distributions. i.e. loss_real distribution formed by real data and loss_fake distribution formed by Generator's fake data. Wasserstein Distance is used to measure the distance between the two distributions, thus improving Discriminator's ability to distinguish between true and false.

Assuming that the Loss distributions generated by real data and fake data are represented by $\mu_1$ and $\mu_2$ respectively. It is found by experiments that the reconstruction errors of each input data are independent and identically distributed and obey normal distribution. Thus, two one-dimensional normal distributions, loss_real and loss_fake, are denoted as $u_1 = N(m_1, c_1)$, $u_2 = N(m_2, c_2)$, where $m_1$, $m_2$ and $c_1$, $c_2$ are the mean and variance of loss_real and loss_fake, respectively. Their Wasserstein Distance is defined as:

$$W(u_1, u_2) = \inf_{\gamma \in \Gamma(u_1, u_2)} \mathbb{E}_{(x_1, x_2) \sim \gamma}[|x_1 - x_2|] \tag{32}$$

Here, $x_1$ and $x_2$ are random samples subject to $u_1$ and $u_2$ respectively. Their joint distribution obeys $\gamma$, all possible forms of $\gamma$ constitute a probability space $\Gamma(u_1, u_2)$. The joint distribution of the smallest value in $\Gamma(u_1, u_2)$ is the target distribution. Its expected value $\mathbb{E}_{(x_1, x_2) \sim \gamma}[|x_1 - x_2|]$ is the distance required. Discriminator hopes that the distance becomes larger, but the optimal solution of $\gamma$ is unknown. It is very difficult to solve it directly, and it needs to be approximated by a variable lower bound. The lower bounds of

$W(u_1, u_2)$ can be determined by Jensen inequality.

$$\inf E[|x_1 - x_2|] \geq \inf |\mathbb{E}[x_1 - x_2]| = |m_1 - m_2| \quad (33)$$

so,

$$W(u_1, u_2) \geq |m_1 - m_2| \quad (34)$$

For the discriminator, there are two ways to increase the value of $W(u_1, u_2)$:

$$W(u_1, u_2) \geq m_1 - m_2, m_1 \to \infty \ \& \ m_2 \to 0 \quad (35)$$

$$W(u_1, u_2) \geq m_2 - m_1, m_2 \to \infty \ \& \ m_1 \to 0 \quad (36)$$

Intuitively, when the Discriminator is trained well, the error of real data is expected to be the smallest, that is, $m_1 \to 0$. Therefore, $W(u_1, u_2) \geq m_2 - m_1$. In this way, the goal of maximizing Discriminator can be equivalent to minimizing $m_1 - m_2$, while the goal of Gnerator is to minimize the distance between two distributions, which can be achieved by minimizing $m_2$.

Finally,

$$\mathcal{L}_D = m_1 - m_2 = \mathbb{E}(\mathcal{L}(x)) - \mathbb{E}(\mathcal{L}(G(z_D))) \quad (37)$$

$$\mathcal{L}_G = \mathbb{E}(\mathcal{L}(G(z_D))) \quad (38)$$

When the loss in Generator and Discriminator are respectively equalized, the Discriminator cannot distinguish between true and false. At this time, $W(u_1, u_2) \to 0$, so there are:

$$\mathbb{E}(\mathcal{L}(x)) = \mathbb{E}(\mathcal{L}(G(z_D))) \quad (39)$$

Since the Generator's generation process is slower than the Discriminator during the training process, which will lead to unstable training. Therefore, a coefficient $k_t$ is added to $\mathcal{L}_D$ to balance the difference between them.

$$\mathcal{L}_D = \mathbb{E}(\mathcal{L}(x) - k_t \cdot \mathbb{E}(\mathcal{L}(G(z_D))) \quad (40)$$

$$k_{t+1} = k_t + \lambda_k(\gamma\mathcal{L}(x) - \mathcal{L}(G(z_G))) \quad (41)$$

among of them:

$$\gamma = \frac{\mathbb{E}(\mathcal{L}(G(z_G)))}{\mathbb{E}(\mathcal{L}(x))} \quad (42)$$

$\gamma$ is a hyperparameter, taking 0.001.

# 5. Association Transformation of Different Distribution Data

For the problem of inter-association and transformation style of image datasets with different probability distributions, the integration of Encoder-Decoder with GAN has made great progress in recent years. The problem is actually to learn a mapping function. In the case that the underlying data does not change the distribution, the style conversion of different image data sets is performed by finding correspondences such as similar semantics. And this conversion is unsupervised learning.

Ming-Yu Liu et al. extended the distribution $p(x)$ to the joint distribution $p(x_1, x_2)$ and proposed CoGAN to deal with domain adaptation problems [27]. CoGAN considered learning a combination of two domain types to get a joint distribution with different attributes. CoGAN consists of two GANs, each for a domain type image. If the two GANs are directly trained independently to process image adaptation problems of two different domain types, the generated results will lead to the inner product $p(x_1)p(x_2)$ of the two edge distributions being not equal to their joint distribution $p(x_1, x_2)$, which makes the GAN unable to learn a joint distribution with different attributes. In COGAN, the weights of two gas in some layers of the generator are shared and constrained, so that COGAN can learn a joint distribution when there is no correlation between two domain types. Because in the deep network of generator, the weight sharing of high-level semantic information, it can be seen that the GAN 's Discriminator can decompose the semantic information of the high-level, and the bottom layer in the Generator is still the content in different domain types.

Specifically, Cogan is composed of GAN1 and GAN2. Each GAN generates and distinguishes images of its own domain type. In training, two Generators need to share a part of the weight of the upper layer in the deep network. which is:

$$g_1(z) = g_1^{(m_1)}(g_1^{(m_1-1)}(\cdots g_1^{(2)}(g_1^{(1)}(z)))) \quad (43)$$

$$g_2(z) = g_2^{(m_2)}(g_2^{(m_2-1)}(\cdots g_2^{(2)}(g_2^{(1)}(z)))) \quad (44)$$

The discriminator needs to share part of the weights in the lower layer of the deep network. which is:

$$d_1(x_1) = d_1^{(n_1)}(d_1^{(n_1-1)}(\cdots d_1^{(2)}(d_1^{(1)}(x_1)))) \quad (45)$$

$$d_2(x_2) = d_2^{(n_2)}(d_2^{(n_2-1)}(\cdots d_2^{(2)}(d_2^{(1)}(x_2)))) \quad (46)$$

finally:

$$V(D, G) = \mathbb{E}_{x_1 \sim p(x_1)}[\log D_1(x_1)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D_1(G_1(z)))] + \mathbb{E}_{x_2 \sim p(x_2)}[\log D_2(x_2)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D_2(G_2(z)))] \quad (47)$$

When the weights of all Generators of GAN1 and GAN2 in CoGAN are shared, Generator1 and Generator2 can be collectively regarded as an Encoder. The basic framework of CoGAN is shown in Figure 7.

Compared with the structure of Encoder-Decoder introduced by COGAN, Guillaume Lample et al. integrated GAN into Encoder-Decoder and proposed the structure of Fader Networks [28]. Fader networks combines Encoder and Discriminator to form a generation discrimination combination of GAN. After the Encoder generates the data $E_x$, the Encoder continuously discriminates whether $E_x$ is related to the attribute $y$ of the original data through the Discriminator. Continuously optimizing the process can achieve the stripping of $E_x$ and $y$. Finally, when the decoder generates the new image data, it can add some attributes of $y$ that meet the needs, thus achieving the purpose of controlling

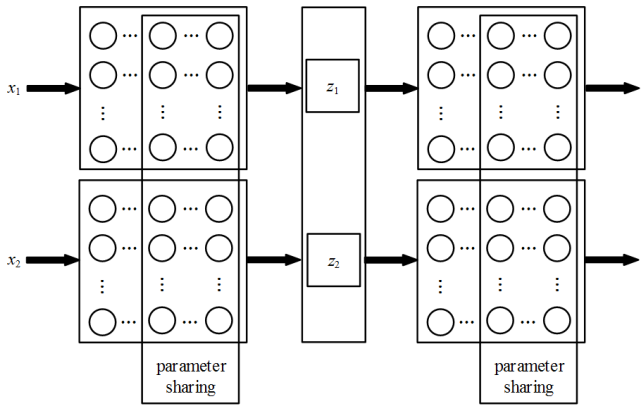


***Figure 7.*** *The structure of EBGAN. By sharing the weights of the latter layers of networks in Encoder1 and Encoder2, the structure of the two probability distributions in feature extraction can be similar. And the weights of the former layers of networks in Decoder1 and Decoder2 can be shared.*

the image to be generated as required.

Thus, the Discriminator discriminates whether $\mathbb{E}(x)$ is related to $y$ and its expression is:

$$\mathcal{L}_{dis}(\theta_{dis}|\theta_{enc}) = -\frac{1}{m}\sum_{(x,y)\in D} \log P_{\theta_{dis}}(y|\mathbb{E}_{\theta_{enc}}(x)) \quad (48)$$

$\theta_{dis}$ and $\theta_{enc}$ represent the weights of the Discriminator and Encoder, Discriminator is to maximize the correlation between $\mathbb{E}(x)$ and $y$. In addition, the final generated image is very similar to the original image, then:

$$\mathcal{L}_{AE}(\theta_{enc}, \theta_{dec}) = \frac{1}{m}\sum_{(x,y)\in D} \| D_{dec}(\mathbb{E}_{\theta_{enc}}(x), y) - x \|_2^2 \quad (49)$$

$$\mathcal{L}(\theta_{enc}, \theta_{dec}|\theta_{dis}) = L_{AE}(\theta_{enc}, \theta_{dec}) = -\lambda_E \log P_{\theta_{dis}}(1-y|E_{\theta_{enc}}(x)) \quad (50)$$

However, such an approach can easily lead to a pattern collapse problem, that is, $\mathrm{G}(x_1)$ converts a large number of images in $X_1$ space into some small images in $X_2$ space. In this regard, CycleGAN also learned the mapping from $X_2$ to $X_1$, namely $\mathrm{G}_1(\mathrm{G}_2(x_2)) \approx x_2$, $G_2(G_1(x_1)) \approx x_1$. This loss is called the cyclic consistency loss and is defined as:

Here $\lambda_E > 0$. $\lambda_E$ needs to be adjusted precisely. A larger $\lambda_E$ will limit the amount of information about $x$ contained in $\mathbb{E}(x)$, which resulting in image blurring. A smaller $\lambda_E$ will limit the discriminator's dependence on $y$. The basic framework of Fader Networks is shown in Figure 8.

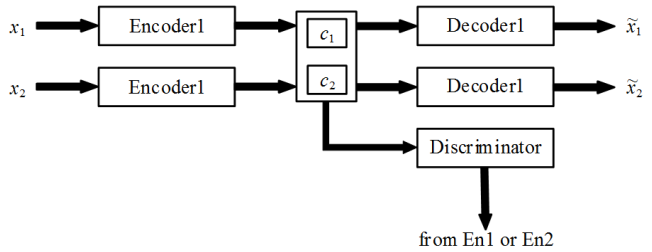


***Figure 8.*** *The structure of Fader Networks. Discriminator judges whether Encoder produces data from Encoder1 or Encoder2, forcing Encoder1 and Encoder2 to generate eigenvectors approximating the same probability distribution to deceive Discriminator.*

Jun-Yan Zhu et al. designed CycleGAN to convert images into styles [29]. That is, there are two differently distributed image data $x_1$ and $x_2$. CycleGAN can convert the distribution of samples in $x_1$ space into the distribution of $x_2$ space. Therefore, the goal of CycleGAN is to learn the mapping from $x_1$ to $x_2$.

There is a generator that can convert the images in the $x_1$ space to the distribution in the $x_2$ space, which is set to $G_{x_1}$. For $G_{x_1}$, it is necessary to determine whether it is real or not by Discriminator and set it to $D_{x_2}$, thus forming a GAN. The loss function expression is:

$$\mathcal{L}_{\mathrm{GAN}}(G_{x_1}, D_{x_2}, X_1, X_2) = \mathbb{E}_{x_2\sim p(X_2)}[\log D_{X_2}(x_2)] + \mathbb{E}_{x_1\sim p(X_1)}[\log(1 - D_{X_2}(\mathrm{G}(x_1)))] \quad (51)$$

$$\mathcal{L}_{\mathrm{cyc}}(G_{X_1}, G_{X_2}, X_1, X_2) = \mathbb{E}_{x_2\sim p(X_2)}[\| G_1(G_2(x_2)) - x_2 \|_1] + \mathbb{E}_{x_1\sim p(X_1)}[\| G_2(G_1(x_1)) - x_1 \|_1] \quad (52)$$

At the same time, we need to introduce a discriminator for $G_{X_2}$, which is set to $D_{X_1}$. Finally, the loss function consists of three parts:

$$\mathcal{L} = \mathcal{L}_{\mathrm{GAN}}(G_{X_1}, D_{X_2}, X_1, X_2) + \mathcal{L}_{\mathrm{GAN}}(G_{X_2}, D_{X_1}, X_1, X_2) + \mathcal{L}_{\mathrm{cyc}}(G_{X_1}, G_{X_2}, X_1, X_2) \quad (53)$$

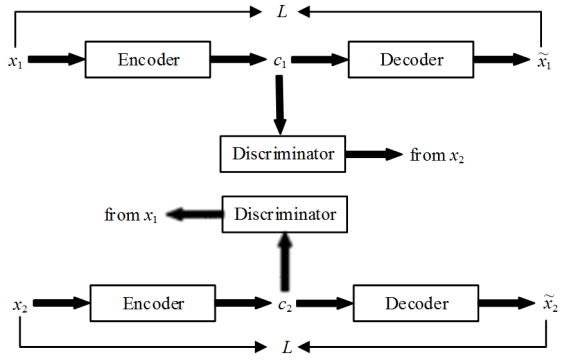


***Figure 9.*** *The structure of CycleGAN. Two mirrored GANs, one Generator generates a fake image, and the other generators need to generate images that can deceive Discriminator of different probability distributions, the generated face image and real image have reasonable loss values.*

The basic framework of CycleGAN is shown in Figure 9.

When using CycleGAN, it is found that although CycleGAN can achieve cross domain style transformation, but it is unstable and needs to be adjusted repeatedly. Amelie Royer et al. Introduced the application of Encoder-Decoder structure and combined it with CycleGAN to form XGAN [30].

There are two two encoders, decoder and discriminator in XGAN, and it has one classifier. The discriminator in XGAN acts similarly to CycleGAN. XGAN let $x_1$ get $E_{x_1}$ through encoder1, then generate $D_{x_2}$ through decoder2, then output $E_{x_2}$ with $D_{x_2}$ as the input of encoder2, final calculate the distance between $E_{x_1}$ and $E_{x_2}$ as loss function.

In XGAN, the loss function of two pairs of self-Encoders is:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{x_1 \sim p(X_1)} |x_1 - d_1(e_1(x_1)| + \mathbb{E}_{x_2 \sim p(X_2)} |x_2 - d_2(e_2(x_2))| \tag{54}$$

The classifier in XGAN is used to distinguish the images in different domains after coding.

If the encoded image is still distinguishable, the encoded information has not only feature information but also domain information. If it cannot be distinguished, it means that the encoded information is feature information common to the two domains.

$$\mathcal{L}_{\text{dann}} = \mathbb{E}_{x_1 \sim p(X_1)} \ell(1, c(e_1(x_1))) + \mathbb{E}_{x_1 \sim p(X_1)} \ell(2, c(e_2(x_2))) \tag{55}$$

$\ell$ is a loss function for classification.

Two encoders need to maintain feature consistency when encoding two fields. The loss of consistency is:

$$\mathcal{L}_{\text{sem}} = \mathbb{E}_{x_1 \sim p(X_1)} |e_1(x_1) - e_2(d_2(e_1(x_1)))| + \mathbb{E}_{x_2 \sim p(X_2)} |e_2(x_2) - e_1(d_1(e_2(x_2)))| \tag{56}$$

The Discriminator loss is:

$$\mathcal{L}_{\text{GAN},x_1 \to x_2} = \mathbb{E}_{x_1 \sim p(X_1)} [\log D(x_1)] + \mathbb{E}_{x_2 \sim p(X_2)} [\log(1 - D(d_1(e_2(x_2))))] \tag{57}$$

$$\mathcal{L}_{\text{GAN},x_2 \to x_1} = \mathbb{E}_{x_2 \sim p(X_2)} [\log D(x_2)] + \mathbb{E}_{x_1 \sim p(X_1)} [\log(1 - D(d_2(e_1(x_1))))] \tag{58}$$

XGAN introduces an optional function which means teacher loss. When there is prior knowledge, this function can be integrated into XGAN 's model. It is asymmetric.

$$\mathcal{L}_{\text{teach}} = \mathbb{E}_{x_1 \sim p(X_1)} \parallel \mathrm{T}(x_1) - e_1(x_1) \parallel \tag{59}$$

finally,

$$\mathcal{L}_{\text{XGAN}} = \mathcal{L}_{\text{rec}} + \omega_{\text{d}} \mathcal{L}_{\text{dann}} + \omega_{\text{s}} \mathcal{L}_{\text{sem}} + \omega_{\text{G}} \mathcal{L}_{\text{GAN}} + \omega_{\text{t}} \mathcal{L}_{\text{teach}} \tag{60}$$

The basic framework of XGAN is shown in Figure 10.

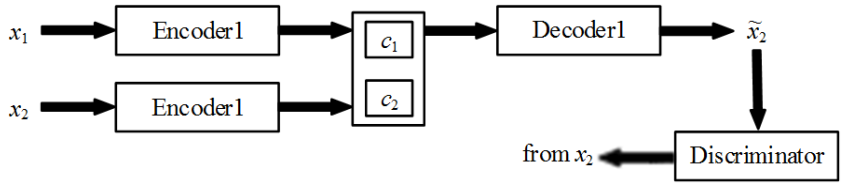


***Figure 10.*** *The structure of XGAN. XGAN maintains feature-to-feature consistency by calculating the similarity of latent through two loops.*

Asha Anoosheh et al. proposed ComboGAN [31] to apply the Encoder-Decoder structure to different distribution problems of image data. Let $x_1$ get $E_{x_1}$ through Encoder1, then generate $D_{x_2}$ through Decoder2, then output $E_{x_2}$ as input of Encoder2, and finally generate $D_{x_1}$ through Decoder1. The distance between $E_{x_1}$ and $D_{x_1}$ is calculated as a loss function. The basic framework of combogan is shown in Figure 11.

Jianlin Su proposed the model of O-GAN (original general advanced network) considering that the structure of Discriminator is similar to Encoder [32]. The Discriminator output is the scalar of classification, the Encoder output is a vector, so the Discriminator is written as a composite function $D(x) \triangleq T(e(x))$, where $T$ is the mapping from noise space to discrimination space. Here, in order to realize the functions of generator and discriminator, O-GAN adds a Pearson coefficient as the regularization term:

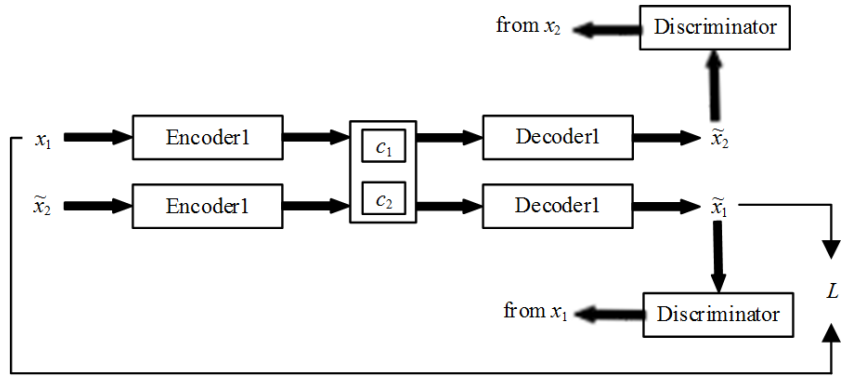


***Figure 11.*** *The structure of ComboGAN. ComboGAN uses cyclic consistency to maintain the similarity between pixel-to-pixel.*

$$T, e = \text{argmin}_{T,e} \, \mathbb{E}_{x \sim p(x), z \sim p(z)} [\log(T(e(x)) + \log(1 - T(e(G(z)))) - \lambda \rho(z, e(G(z)))] \tag{61}$$

$$G = \text{argmin}_G \, \mathbb{E}_{z\sim p(z)} \left[\log(T(e(G(z)))) - \lambda\rho(z, e(G(z)))\right] \tag{62}$$

Among them,

$$\rho(z, \hat{z}) = \frac{1}{std(z)\times std(\hat{z})} \cdot \sum_{i=1}^{n_z} (z_i - \text{avg}(z))(\hat{z}_i - \text{avg}(\hat{z}))/n_z \tag{63}$$

In this way, the discriminator is divided into two parts, where $e$ is the decoder. For $T$, $\text{avg}(e(x))$ is directly used in O-GAN. That is, $T(e(x)) = \text{avg}(e(x))$.

# 6. Summary

GAN and Encoder-Decoder are both generative models. Their output is determined by the input. A model can be used to learn a characteristic representation of the input. And both models are directly learning from input samples, so it is not necessary for label information. Specifically, for the generation model in GAN, it is no longer necessary to need a rigorously formatted generated data representation like the traditional model, which directly avoids the computability of the model due to the complexity of the data, and also avoids the inability of data input or generation due to the complexity of the model. However, GAN has the possibility of mode collapse, and the input data needs to be continuous. Encoder-Decoder is composed of two multilayer neural networks. Input data and output data need to express the same probability distribution with the same number of nodes. The significance of the model lies in the middle vector layer, which represents "dimension reduction" and maintains the feature distribution of input data to the greatest extent. The data generated by the Encoder-Decoder is lost due to dimensionality reduction after dimension reduction. Based on the advantages and disadvantages of the two models, a new model can be generated by combining them, which can overcome their shortcomings and produce data information that can meet the needs better.

# References

[1] Sabuncu M R, Yeo B T T, Van Leemput K, et al. A generative model for image segmentation based on label fusion [J]. IEEE transactions on medical imaging, 2010, 29 (10): 1714-1729.

[2] Wang C, Chang X, Xin Y, et al. Evolutionary Generative Adversarial Networks [J]. IEEE Transactions on Evolutionary Computation, 2019.

[3] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets [C]. Advances in neural information processing systems. 2014: 2672-2680.

[4] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation [J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39 (12): 2481-2495.

[5] Ledig C, Theis L, Huszár F, et al. Photo-realistic single image super-resolution using a generative adversarial network [C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4681-4690.

[6] Denton E L, Chintala S, Fergus R. Deep generative image models using a laplacian pyramid of adversarial networks [C]. Advances in neural information processing systems. 2015: 1486-1494.

[7] Kendall A, Badrinarayanan V, Cipolla R. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding [J]. arXiv preprint arXiv: 1511.02680, 2015.

[8] Sordoni, Alessandro, Bengio, et al. A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion [J]. Computer Science, 2015: 553-562.

[9] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks [C]. Advances in neural information processing systems. 2014: 3104-3112.

[10] Creswell A, White T, Dumoulin V, et al. Generative adversarial networks: An overview [J]. IEEE Signal Processing Magazine, 2018, 35 (1): 53-65.

[11] Zhang H, Goodfellow I, Metaxas D, et al. Self-attention generative adversarial networks [J]. arXiv preprint arXiv: 1805.08318, 2018.

[12] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation [J]. arXiv preprint arXiv: 1406.1078, 2014.

[13] Choi Y, Choi M, Kim M, et al. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8789-8797.

[14] Alonso-Monsalve S, Whitehead L H. Image-based model parameter optimisation using Model-Assisted Generative Adversarial Networks [J]. arXiv preprint arXiv: 1812.00879, 2018.

[15] Paisley J, Blei D, Jordan M. Variational Bayesian inference with stochastic search [J]. arXiv preprint arXiv: 1206.6430, 2012.

[16] Larsen A B L, Sønderby S K, Larochelle H, et al. Autoencoding beyond pixels using a learned similarity metric [J]. arXiv preprint arXiv: 1512.09300, 2015.

[17] Chen X, Duan Y, Houthooft R, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets [C]. Advances in neural information processing systems. 2016: 2172-2180.

[18] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks [J]. arXiv preprint arXiv: 1511.06434, 2015.

[19] Kurutach T, Tamar A, Yang G, et al. Learning plannable representations with causal infogan [C]. Advances in Neural Information Processing Systems. 2018: 8733-8744.

[20] Dumoulin V, Belghazi I, Poole B, et al. Adversarially learned inference [J]. arXiv preprint arXiv: 1606.00704, 2016.

[21] Donahue J, Krähenbühl P, Darrell T. Adversarial feature learning [J]. arXiv preprint arXiv: 1605.09782, 2016.

[22] Huang H, He R, Sun Z, et al. Introvae: Introspective variational autoencoders for photographic image synthesis [C]. Advances in Neural Information Processing Systems. 2018: 52-63.

[23] Ng A. Sparse autoencoder [J]. CS294A Lecture notes, 2011, 72 (2011): 1-19.

[24] Poultney C, Chopra S, Cun Y L. Efficient learning of sparse representations with an energy-based model [C]. Advances in neural information processing systems. 2007: 1137-1144.

[25] Zhao J, Mathieu M, LeCun Y. Energy-based generative adversarial network [J]. arXiv preprint arXiv: 1609.03126, 2016.

[26] Berthelot D, Schumm T, Metz L. Began: Boundary equilibrium generative adversarial networks [J]. arXiv preprint arXiv: 1703. 10717, 2017.

[27] Liu M Y, Tuzel O. Coupled generative adversarial networks [C]. Advances in neural information processing systems. 2016: 469-477.

[28] Lample G, Zeghidour N, Usunier N, et al. Fader networks: Manipulating images by sliding attributes [C]. Advances in Neural Information Processing Systems. 2017: 5967-5976.

[29] Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks [C]. Proceedings of the IEEE international conference on computer vision. 2017: 2223-2232.

[30] Royer A, Bousmalis K, Gouws S, et al. Xgan: Unsupervised image-to-image translation for many-to-many mappings [J]. arXiv preprint arXiv: 1711.05139, 2017.

[31] Anoosheh A, Agustsson E, Timofte R, et al. Combogan: Unrestrained scalability for image domain translation [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018: 783-790.

[32] Su J. O-GAN: Extremely Concise Approach for Auto-Encoding Generative Adversarial Networks [J]. arXiv preprint arXiv: 1903.01931, 2019.