Auto-Encoding GAN for Reducing Mode Collapse and Enhancing Feature Representation

Xiaoxiang Lu, Yang Zou*, Xiaoqin Zeng, Xiangchen Wu, Pengfei Qiu
Institute of Intelligence Science and Technology, School of Computer and Information,
Hohai University, Nanjing, China
{luxx0824, yzou, xzeng, wxc}@hhu.edu.cn

Abstract—**Generative Adversarial Nets (GAN) has been a popular research topic in processing of images, speech, texts, and videos, and many other fields. However, GAN still has some drawbacks such as unstable training and mode collapse. To address these challenges, this paper proposes an auto-encoding GAN, which is composed of a set of generators, a discriminator, an encoder and a decoder. A set of generators is responsible for learning different modes, accelerating the convergence of the model and preventing model collapse. The discriminator is used to distinguish between real samples and generated ones. In order to improve feature representation of the encoder and prevent multiple generators from covering a certain mode, an approach consisting of three phases is proposed accordingly. First, a clustering algorithm is presented to perceive the distribution of real and generated samples. Then, cluster center matching is utilized to keep consistency of the distribution of real and generated samples. Finally, the encoder and decoder are jointly optimized by the generated and real samples. Therefore, the encoder can map the generated and real samples to the embedding space so as to encode distinguishable features, and the decoder can distinguish from which generator the generated samples come and from which mode the real samples come. Experiments are conducted on image datasets to verify effectiveness of the auto-encoding GAN for reducing mode collapse and enhancing feature representation.**

Keywords-GAN; mode collapse; feature representation; cluster

I. Introduction

GAN [1] is a generative model proposed by Goodfellow, and it has a wide range of applications in data generation, style transfer, image inpainting, etc. [2,3,4,5]. However, GAN suffers from several problems such as unstable training, blurred images and mode collapse, etc. Among them, mode collapse has been a crucial problem that needs to be addressed for GAN. Mode collapse means that GAN only generates some of all modes, and others are missing. In order to reduce mode collapse, researchers have made a number of improvements, which can be summarized in the following four classes.

1) Adding constraints: Unrolled GAN [6] stabilizes the training of GAN by K-step cyclic training to avoid the generator falling into a single mode. CGAN [7] strengthens the relationship between input and output data by using conditional information, forcing the generator to learn different modes. InfoGAN [8] maximizes the mutual information between the input data and the hidden code to obtain interpretable and distinguishable features and avoid mode collapse. Mixture Density GAN [9] encourages the discriminator to conform to Gaussian mixture distribution in the embedded space, which ensures that the generator fits Gaussian mixture distribution as much as possible, covering different modes.

2) Adding generators: The typical models are MADGAN [10] and MGAN [11]. Unlike GAN, their discriminators not only need to distinguish between real samples and generated samples, but also distinguish from which generator the generated samples come.

3) Modifying loss function: WGAN [12] uses the EM distance to measure the difference between generated distribution and real distribution, avoiding the problem of gradient disappearance of the generator, and making sure the generated distribution is as close to real distribution as possible to cover multiple modes.

4) Imposing gradient penalty: DRAGAN [13] imposes a gradient penalty on the training samples to the discriminator, and tries to construct a linear function on the training samples to find the global optimal solution.

In classes *1)* and *2)*, the discriminator only considers the distribution of the generated samples, but neglects the distribution of the real samples, which may lead to a problem that multiple generators cover different parts of a certain mode, resulting in mode collapse. As for *3)* and *4)*, although the generator can cover multiple modes, it still retains the characteristics of continuous mapping in GAN, which may cover the blank area between modes and generate poor samples, resulting in training instability.

In order to address the issues of modes collapse and training instability, this paper proposes an auto-encoding GAN, which consists of a set of generators, a discriminator, an encoder and a decoder. The network architecture is shown in Fig. 1, where a set of generators is responsible for covering different modes, and the discriminator, like GAN, is used to distinguish between real samples and generated samples, ensuring that the generated distribution does not deviate from real distribution. In order to prevent multiple generators from covering different parts of a certain mode, different from the above-mentioned multi-generator model, a clustering algorithm is introduced to perceive

*Corresponding author: yzou@hhu.edu.cn (Y. Zou)

DOI reference number: 10.18293/SEKE2022-152

the distribution of real samples and generated samples, and an algorithm of cluster center matching is presented to keep consistency of the distribution of real and generated samples. Then, the encoder and decoder are jointly optimized by the generated and real samples. Therefore, the encoder can map generated samples and real samples to the embedding space so as to encode distinguishable feature, and the decoder can distinguish from which generator the generated samples come and from which mode the real samples come. Experimental results show that the trained auto-encoding GAN can not only reduce mode collapse, but also have preferable capability in feature representation.

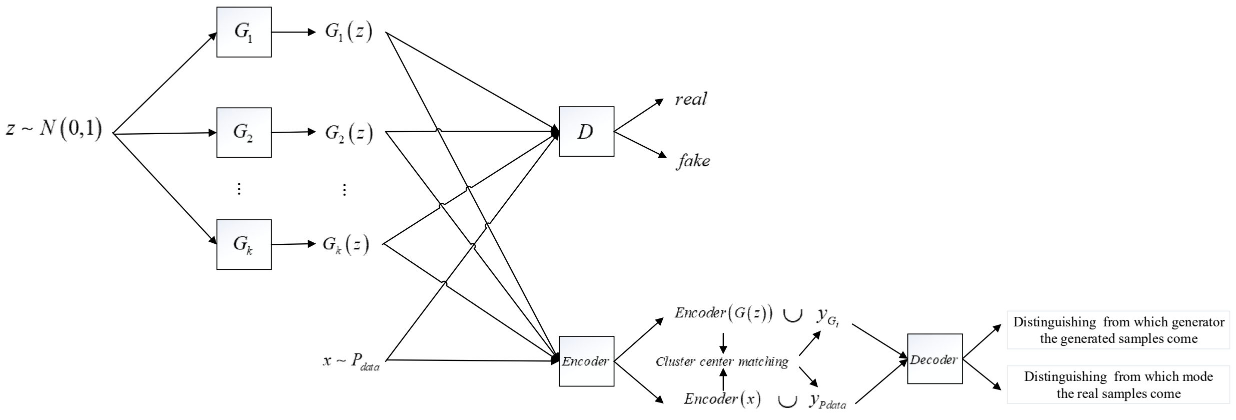
$z \sim N(0,1)$
G_1
G_2
G_k
$G_1(z)$
$G_2(z)$
$G_k(z)$
$x \sim P_{data}$
D
real
fake
Encoder
$Encoder(G(z)) \cup y_{G_i}$
Cluster center matching
$Encoder(x) \cup y_{Pdata}$
Decoder
Distinguishing from which generator the generated samples come
Distinguishing from which mode the real samples come

Figure 1. Architecture of Auto-Encoding GAN. $z \sim N(0,1)$ is input to different $G_1, G_2, ..., G_k$ in equal amounts. The generated samples $G_{i\,(1 \le i \le k)}(z)$ and real samples x are input to the discriminator and encoder-decoder respectively, and the discriminator distinguishes the real and the fake and the encoder extract features. The pseudo labels y_{data} and y_{G_i} of real and generated samples are determined by the cluster center matching algorithm. The decoder distinguishes from which generator the generated samples come and from which mode the real samples come by cross-entropy loss.

The technical contributions of the paper are summarized as follows:

1) From the perspective of data distribution, it verifies that adding generators is an effective way to tackle training instability and mode collapse.

2) An algorithm for cluster center matching is proposed to keep consistency of the distribution of real and generated samples, prevent multiple generators from covering different parts of a certain mode and reduce mode collapse.

3) The encoder and decoder are jointly optimized by generated samples and real samples, which reduces the mode collapse of generator and enhancing the feature representation of encoder.

II. Auto-Encoding GAN

This section elaborates on the auto-encoding GAN's network architecture, objective function and algorithm of cluster center matching in detail.

A. Adding Generators to Reduce Training Instability and Mode Collapse

The training instability of GAN means that the learning processes of generator G and discriminator D are difficult to converge together. From the perspective of data distribution, the reason is that the complexity of the real samples' distribution affects the speed of G and D convergence. If the real samples are a single-mode simple distribution, G and D converge together soon. If the real samples are a multi-mode complex distribution, G and D are difficult to converge together, as shown in the Fig. 2 and Fig. 3.

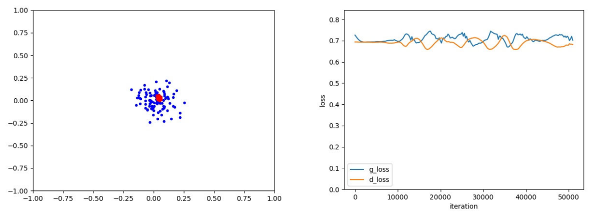

Figure 2. Left is mode coverage of GAN on single-mode simple distribution, where blue and red points represent real and generated samples, respectively. Right is the loss curve of G and D of GAN on the single-mode distribution.

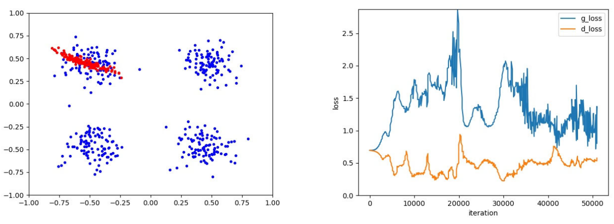

Figure 3. Left is mode coverage of GAN on four-mode complex distribution, where blue and red points represent real and generated samples respectively. Right is the loss curve of G and D of GAN on the four-mode distribution.

As shown in Fig. 3, it can be seen from the loss curve of four-mode complex distribution that the training of G and D is unstable and difficult to converge together. However, in Fig. 2, G and D of GAN converge easily together for single-mode simple distribution. Obviously, if one generator is used to learn multiple modes, it will be difficult to converge and the training is unstable. Therefore, if a set of generators is employed to learn multiple modes, trying to ensure that a single generator covers a single mode, the training instability of GAN can be effectively settled, as shown in Fig. 4. Accordingly, it can be seen from Fig. 3 and Fig. 4 that the generators and discriminator of auto-encoding GAN converge and tend to be stable much more quickly than that of GAN.

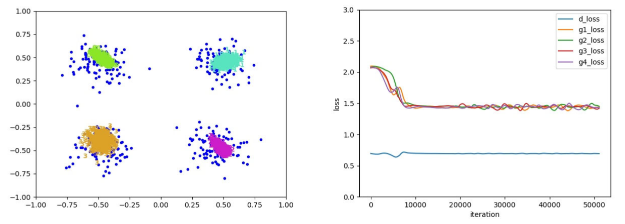

Figure 4. Left is mode coverage of Auto-Encoding GAN on four-mode complex distribution, where blue points represent real samples, and others represent generated samples in which samples generated from different generators are denoted by different colors. Right is the loss curve of G and D of Auto-Encoding GAN on the four-mode distribution.

The mode collapse of GAN refers to that GAN tends to concentrate continuously mapped values on a single mode during the training process. The main reason for the phenomenon is GAN can only approximate continuous mapping, on the contrary, but the multi-mode distribution belongs to the discrete distribution that is not continuous, so GAN is difficult to cover discrete multi-mode distributions with continuous map. If a continuous map is forced to cover all modes, the values of the continuous map will inevitably cover some blank areas outside the mode, so GAN may generate some samples that have no realistic meaning, which explains why GAN may generate some poor samples. For a theoretical proof of GAN's mode collapse, please refer to [14].

Therefore, the key to addressing mode collapse is to build a discontinuous map. Adding generators can discretize the generation distribution, so that each generator covers a mode, which is essentially similar the way that multiple GAN utilizes to achieve the discontinuous mapping of multi-mode distribution, but the difference is that simply adding generators can do.

In summary, auto-encoding GAN adopts the way of adding generators to realize discontinuous mapping of multi-mode distribution, which can not only speed up convergence of the model, but also cover multiple modes.

B. Objective Function of Auto-Encoding GAN

Assume real samples is subject to distribution of real samples $x \sim P_{data}$, where P_{data} is the distribution of real samples, containing k modes, and noise samples is subject to standard normal distribution $z \sim N(0,1)$. According to the architecture of the auto-encoding GAN, k generators $G=\{G_1, G_2, ..., G_k\}$ are designed to try to cover various k modes. A discriminator D is responsible for distinguishing between real and generated samples. An *Encoder* is used to map the generated samples and real samples to the embedding space. A *Decoder* not only distinguishes from which generator the generated samples come, but also distinguishes from which mode the real samples come, where G, D, *Encoder-Decoder*(ED) are deep networks. y_{G_i} represents the label corresponding to the generated samples by the *i-th* generator, and $y_{P_{data}}$ represents the label corresponding to the real sample, which is mainly a pseudo-label after cluster center matching. The purpose of auto-encoding GAN training is to obtain a set of $G=\{G_1, G_2, ..., G_k\}$ that can cover different modes and an *Encoder* with capability of feature representation. Therefore, the objective function that needs to be optimized for auto-encoding GAN can be written as follows.

$$\begin{aligned}\min_{G,ED}\max_{D} L(G,D,ED) = E_{x\sim P_{data}}\left[log\left(D(x)\right)\right] + \sum_{i=1}^{k} E_{z\sim P_{G_i}}\left[log\left(1-D\left(G_i(z)\right)\right)\right] \\ -\lambda\left[\sum_{i=1}^{k} y_{G_i} E_{z\sim P_{G_i}}\left[log\left(ED\left(G(z)\right)\right)\right] + y_{P_{data}} E_{x\sim P_{data}}\left[log\left(ED(x)\right)\right]\right]\end{aligned} \quad (1)$$

where λ is the weight to keep the balance between D and ED.

Assume G, D, and ED have sufficient capacity and training time, the conditions of convergence of them are given below.

Proposition 1. For G fixed, the optimal D is

$$D(x) = \frac{P_{data}}{P_{data} + \sum_{i=1}^{k} P_{G_i}}$$

Proposition 2. For G fixed, the optimal ED is

$$\begin{cases} ED_G\left(G(z)\right) = \sum_{i=1}^{k} y_{G_i} \\ ED_x(x) = y_{P_{data}} \end{cases}$$

For the convenience of description, ED can be divided into $ED_G(x)$ and $ED_x(x)$, which are responsible for the representation and classification of generated samples and real samples, respectively.

Obviously, when $P_{data} = \sum_{i=1}^{k} P_{G_i}$, D converges to the optimum. At this time, it can be deduced that $G(z) = x$, that is $\sum_{i=1}^{k} y_{G_i} = y_{P_{data}}$. When $\sum_{i=1}^{k} y_{G_i} = y_{P_{data}}$, it can be deduced that $ED_G\left(G(z)\right) = ED(x) = y_{P_{data}}$, so the convergence condition of ED is also reached.

Proposition 3. For the optimized D and ED fixed, the optimized generator G is

$$G = -log4 + 2JSD\left(P_{data} \,||\, \sum_{i=1}^{k} P_{G_i}\right) - 2\lambda \int_x \sum_{i=1}^{k} y_{G_i} log\left(y_{G_i}\right)$$

When $P_{data} = \sum_{i=1}^{k} P_{G_i}$, $JSD\left(P_{data} \,||\, \sum_{i=1}^{k} P_{G_i}\right) = 0$ .since both y_{G_i} and $y_{P_{ada}}$ are known, the generator G converges to $G = -log4 - 2\lambda \int_x \sum_{i=1}^{k} y_{G_i} log\left(y_{G_i}\right)$.

Due to limited space, the proofs of convergence of D, ED and G are omitted, which is similar to that of GAN.

C. *Minimizing Cluster Center Matching*

If the decoder only distinguishes from which generator the generated samples come, it may cause the generated samples to cover different parts of a certain mode, resulting in modes collapse. Here, the training results of the InfoGAN and MGAN on 2D dataset are taken as an example to show the case where the modes collapse, as shown in the Fig. 5.

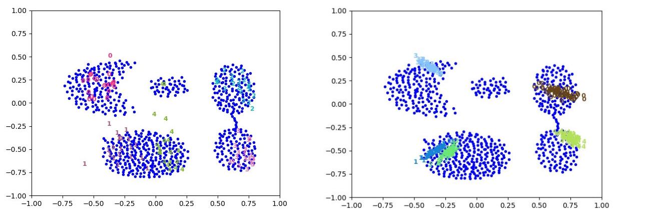

Figure 5. Left is the results of InfoGAN on 2D dataset, and right is the results of MGAN on 2D dataset.

If the mode distribution of real samples is taken into account, the classifier can effectively avoid mode overlap. However, the mode distribution of real samples is unknown, so how to approximate the distribution of real samples is a key issue. Both the generated distribution and real distribution are distributions of the embedding space output by the encoder. Since the same mode is usually clustered in the embedding space due to the similarity between real samples, a clustering algorithm is employed to perceive the mode distribution of real samples.

At the same time, in order to prevent perceived results from deviating from the real distribution, we take the generated distribution as a reference, and balance the deviation by minimizing the cluster centers matching of the generated distribution and real distribution. The reason why the generated distribution can be used as a reference is that it is constantly approaching the real distribution during the training process. Theoretically, when the model converges, the generated distribution approximates the real distribution. However, in actual training the encoder cannot accurately learn the characteristics of the real distribution, since the generated distribution cannot fully represent the real distribution. If some prediction information of the real distribution is added, the learning capability of the encoder for the real distribution will be considerably improved.

According to the above analysis, assuming that the generated samples and the real samples are fed into the encoder, the embedded features output by the encoder are respectively $h_x = Encoder(x)$ and $h_{G(z)} = Encoder(G(z))$. The cluster centers of real samples and generated samples obtained by the clustering algorithm are $\mu_x = \{\mu_{x,1}, \mu_{x,2}, \ldots, \mu_{x,k}\}$ and $\mu_{G(z)} = \{\mu_{G_1(z),1}, \mu_{G_2(z),2}, \ldots, \mu_{G_k(z),k}\}$. In order to minimize the matching of the cluster centers of the real distribution and the generated distribution, the loss function that needs to be satisfied is given as follows.

$$minL_c(G, Encoder) = \sum_{i=1}^{k} E_{x\sim P_{data}} \left\| h_x - \mu_{x,i} \right\|^2 + E_{z\sim P_{G_i}} \left\| h_{G_i(z)} - \mu_{G_i(z),i} \right\|^2 + \left\| \mu_{x,i} - \mu_{G_i(z),i} \right\|^2 \quad (2)$$

Combined with the loss function, the process of minimizing cluster center matching is introduced in detail, which includes the following three steps.

1) First, the encoder is utilized to take the generated and real samples as input, and outputs the embedded features. Then, k-means++ algorithm is used to cluster the generated samples and real samples in the embedded space respectively, and obtain the clustering center sets μ_x and $\mu_{G(z)}$ of the generated samples and real samples respectively.

2) To minimize the cluster center matching, the distance matrix between the center sets μ_x and $\mu_{G(z)}$ is calculated, and *i-th* $\mu_{x,i}$ the closest matching center $\mu_{G(z),i}$ is found to ensure $\left\| \mu_{x,i} - \mu_{G_i(z),i} \right\|^2$ is the smallest.

3) To unify the cluster assignment of the real samples and the generated samples, the generated samples are used as a reference to keep the matching center pairs $\left(\mu_{x,i}, \mu_{G_i(z),i}\right)$ consistent with the corresponding cluster labels.

After the above process, the current loss of centers matching between real distribution and the generated distribution can be obtained, and the generator and encoder can be further optimized.

D. *Training of Auto-Encoding GAN*

Combined with the above introduction of auto-encoding GAN, the specific algorithm of auto-encoding GAN is given.

Algorithm 1 Training of Auto-Encoding GAN

```
Input: Dataset: $X$, Number of generator: $k$, Training epochs: $n\_epochs$,
Output: Generator: $G$, Encoder-Decoder: $ED$
1:  Initialize generator $G$, discriminator $D$, Encoder-Decoder: $ED$.
2:  for $epoch = 1$ to $n\_epochs$ do
3:      Samples a batch $x_i$ from $X$.
4:      Samples $z \sim N(0,1)$.
5:      Calculate $L_D = E_{x\sim P_{data}}[log(1-D(x))] + \sum_{i=1}^{k} E_{z\sim P_{G_i}}[log(1-D(G_i(z)))]$
         according to $Eq.(1)$.
6:      Update D by descending along their gradient $\nabla_D(L_D)$.
7:      Clustering $h_x, h_{(G(z))}$ using k-means++ to get clustering centers $\mu_x, \mu_{G(z)}$.
8:      Matching $\mu_x$ the nearest cluster centers from $\mu_{G(z)}$.
9:      Setting clustering label $y_G, y_{p_{data}}$ of $h_x, h_{(G(z))}$ according to matching result.
10:     Calculate $L_{ED} = \left[\sum_{i=1}^{k} y_{G_i} E_{z\sim P_{G_i}}[log(ED(G_i(z)))] + y_{P_{data}} E_{x\sim P_{data}}[log(ED(x))]\right]$.
11:     Update $ED$ by descending along their gradient $\nabla_{ED}(L_{ED})$.
12:     for $i = 1$ to $k$ do
13:         Calculate $L_{G_i} = \sum_{i=1}^{k} E_{z\sim P_{G_i}}[log(1-D(G_i(z)))]$
                $-\lambda\left[\sum_{i=1}^{k} y_{G_i} E_{z\sim P_{G_i}}[log(ED(G_i(z)))]\right]$.
14:         Updata $G_i$ by descending along their gradient $\nabla_{G_i}(L_{G_i})$.
15:     end for
16: end for
```

III. Experiments

In this section, we conduct experiment on image datasets and demonstrate the effectiveness and of auto-encoding GAN for reducing mode collapse and enhancing feature representation.

A. Implementation Details

For the convenience of experiment reproduction, we provide the experimental details. In the process of cluster center matching, it is necessary to obtain the cluster center for matching by k-means++ algorithm (or GMM). The generated and real samples are clustered in each iteration, where k-means++ adopts default parameters of a python package "sklearn" except for the number of clusters that need to be specified.

It can be seen from the loss function that the hyperparameter involved in this paper is mainly λ, and it is used to balance the weight between the discriminator and the encoder-decoder. Their losses can be unified to magnitude according to the actual training value. In our experiments, λ set to 0.5. In addition, k determines the number of generators, and is generally equal to the number of categories of the dataset in experiments.

B. Experiments of Image Datasets

In this subsection, we conduct experiments on the image datasets to verify the effectiveness of auto-encoding GAN. We choose some frequently used datasets, USPS, MNIST, Fashion-MNIST, Coil-20, and Cifar-10, to conduct experiments.

Image Datasets. USPS and MNIST [15] are digital image datasets. USPS contains 9298 images (16×16) of 10 categories, where 7291 images for training and 2007 images for testing. MNIST contains 70000 images (28×28), including 60000 training images and 10000 testing images, with 10 categories. Fashion-MNIST is similar to MNIST, except that it contains different categories. Coil-20[16] contains 1440 images (128×128), with 20 categories. Cifar-10[17] involves 60,000 color images (32×32) of 10 categories, including 50000 for training and 10000 for testing. Only the training images is used in the experiments.

Evaluation Indicators. The generator is evaluated from the quality and diversity of the generated images, and FID [18] is often used to serve this purpose. The lower the FID score, the closer the generated distribution is to the real distribution, which means that the quality of the generated images is higher and the diversity is better.

As for the evaluation of the encoder-decoder, the feature representation of the encoder is verified from the classification of images. Clustering accuracy (ACC) and normalized mutual information (NMI) is usually exploited as the evaluation indicators of feature representation (especially unsupervised). The larger the value of ACC and NMI, the better the capability of feature representation.

Network Architecture. For USPS, MNIST and Fashion-MNIST datasets, both the generator and discriminator choose the DCGAN network architecture. The encoder has the same convolution architecture as the discriminator. The decoder is coded as a fully connected layer, and the activation function is softmax for classification of generated and real samples. The other parameters of the above datasets are set as follows: the optimizer is uniformly Adam, the learning rate is 0.0004, the batch size is 128, and the epochs is 500.

Experimental Results. Comparative experiments are implemented from the coverage degree of the mode and the feature representation of the model on image datasets, and each class is regarded as a mode in the image datasets. The models selected for comparison are WGAN, DCGAN, InfoGAN and MGAN, as they are representative in each category and more relevant to our model.

1) Experiments are implemented for the mode coverage of different models on the image datasets. The degree of model coverage is mainly evaluated by the FID. The experimental results are shown in Table I.

TABLE I. FID (lower is better) of different models on Image Datasets

Datasets / Models	USPS	MNIST	Fashion-MNIST	Coil-20	Cifar-10
DCGAN	51.41	28.70	72.78	41.39	95.47
WGAN	60.74	83.86	82.79	57.46	100.25
InfoGAN	34.06	26.71	72.92	37.41	**80.82**
MGAN	30.85	25.29	81.24	36.92	87.23
Ours	**28.42**	**22.07**	**64.63**	**35.27**	85.46

From the experimental results in Table I, it can be concluded that auto-encoding GAN is better than other models on USPS, MNIST, Fashion-MNIST and Coil-20. On Cifar-10, InfoGAN is the best and better than auto-encoding GAN.

From the average of FID over all datasets, auto-encoding GAN is 10.77 lower than DCGAN, 27.06 lower than WGAN, 4.01 lower than InfoGAN, and 4.53 lower than MGAN. It is further demonstrated that auto-encoding GAN is significantly better other models in reducing mode collapse.

In addition to the above FID comparison experiments, some images generated by various models on MNIST are shown in Fig. 6.

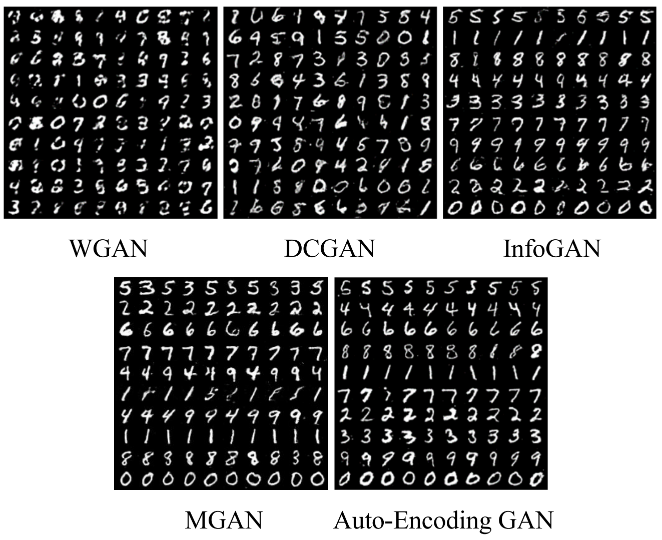

Figure 6. Images generated by various models on MNIST

From the generated images on MNIST, it can be seen that the quality of the images generated by WGAN and DCGAN is poor, and the images are generated in a random way. Both

InfoGAN and MGAN generate a variety of modes, but partial modes have a phenomenon of overlap, such as '1' of MGAN, and the purity of the generated images is low, such as '9' and '4'. Compared with InfoGAN and MGAN, auto-encoding GAN generate all modes, and the purity of generated images is higher.

2) Experiments are implemented for feature representation capabilities of different models on the image datasets. Because WGAN and DCGAN do not have multi-mode representation abilities, only InfoGAN, MGAN and auto-encoding GAN are selected in the experiments. NMI and ACC are used to evaluate the feature representation capabilities of a model. The experimental results are shown in Table II and Table III.

TABLE II. NMI (%) (higher is better) of different models on Image Datasets

Datasets / Model	USPS	MNIST	Fashion-MNIST	Coil-20	Cifar-10
InfoGAN	72.57	85.27	59.49	74.70	**20.49**
MGAN	75.99	85.01	54.11	66.49	17.26
Ours	**76.21**	**87.24**	**59.83**	**79.34**	18.34

TABLE III. ACC (%) (higher is better) of different models on Image Datasets

Datasets / Model	USPS	MNIST	Fashion-MNIST	Coil-20	Cifar-10
InfoGAN	71.42	92.64	**58.44**	60.41	**34.80**
MGAN	74.13	92.52	56.09	56.53	32.17
Ours	**74.92**	**93.27**	58.11	**68.94**	33.34

From the experimental results in Table II, it can be concluded that auto-encoding GAN is better than other models on USPS, MNIST and Coil-20 and Fashion MNIST. On Cifar-10, InfoGAN performs better than auto-encoding GAN. The possible reason for this is InfoGAN can randomly select the hidden code to prevent overfitting of the model compared with multi-generator models.

From the average of NMI over all datasets, auto-encoding GAN is 1.69% better than InfoGAN, and 4.22% better than MGAN. It is further demonstrated that auto-encoding GAN have preferable capability of feature representation.

IV. CONCLUTION

This paper proposed an auto-encoding GAN to reduce the mode collapse of the generator and enhance feature representation of the encoder. It consists of a set of generators, a discriminator, an encoder and a decoder. A set of generators is responsible for learning different modes, accelerating the convergence of the model and preventing mode collapse. The discriminator is used to distinguish between real samples and generated samples. The encoder maps the generated samples and real samples to the embedding space, encoding distinguishable feature information among modes. The decoder distinguishes from which generator the generated samples come and from which mode the real samples come. Different from other multi-generator models, in order to improve the feature representation of the encoder and prevent multiple generators from covering a certain mode, an approach consisting of three phases is proposed accordingly. First, a clustering algorithm is presented to perceive the distribution of real and generated samples. Then, cluster center matching is utilized to keep consistency of the distribution of real and generated samples. Finally, the encoder and decoder are jointly optimized by the generated and real samples. We have conducted experiments on image datasets to fully demonstrate the effectiveness of auto-encoding GAN in reducing mode collapse and enhancing feature representation.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley and S. Ozair. "Generative adversarial nets," Advances in neural information processing systems, 2014.

[2] A. Radforda, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.

[3] T. Karras, S. Laine and T. Aila. "A style-based generator architecture for generative adversarial networks," In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4401-4410, 2019.

[4] JY. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," In Proceedings of the IEEE international conference on computer vision, pp. 2223-2232, 2017.

[5] D. Pathak, P. Krahenbuhl, J. Donahue J, T. Darrell and A.A. Efros, "Context encoders: Feature learning by inpainting". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2536-2544, 2016.

[6] L. Metz, B. Poole, D. Pfau , and J. Sohl-Dickstein, "Unrolled generative adversarial networks," arXiv preprint arXiv:1611.02163, 2016.

[7] M Mirza and S. Osindero, "Conditional generative adversarial nets," Computer Science, arXiv preprint arXiv:1411.1784, 2014.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," Advances in neural information processing systems, 2016.

[9] H. Eghbal-Zadeh, W. Zellinger, and G. Widmer, "Mixture density generative adversarial networks," In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5820-5829, 2019.

[10] A. Ghosh, V. Kulharia, V. Namboodiri, V. P. Namboodiri, P. H. Torr, and P. K. Dokania, "Multi-agent diverse generative adversarial networks," In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8513-8521, 2018.

[11] Q. Hoang, T.D. Nguyen, T. Le, and D. Phung, "MGAN: Training generative adversarial nets with multiple generators," In International conference on learning representations, February 2018.

[12] M. Arjovsky, S. Chintala, and L.Bottou, "Wasserstein generative adversarial networks," In International conference on machine learning, pp. 214-223, July 2017.

[13] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning gan for pose-invariant face recognition," In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1415-1424, 2017.

[14] Y. Guo, D. An, X. Qi, Z. Luo, S. T. Yau, and X.Gu, "Mode collapse and regularity of optimal transportation maps," arXiv preprint arXiv:1902.02934., 2019.

[15] Y. Lecun, and L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition,". Proceedings of the IEEE, 1998, 86(11):2278-2324.

[16] S. Nene, S. Nayar, and Murase, "Columbia object image library (coil-20)," Technical Report, 1996.

[17] A. Krizhevsky, and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," Advances in neural information processing systems, pp. 6629-6640, 2017.