Generative Adversarial Autoencoder Networks

Ngoc-Trung Tran Tuan-Anh Bui Ngai-Man Cheung

Information Systems Technology and Design (ISTD)
Singaore University of Technology and Design (SUTD)
{ngoctrung_tran,tuananh_bui,ngaiman_cheung}@sutd.edu.sg

Abstract. We introduce an effective model to overcome the problem of mode collapse when training Generative Adversarial Networks (GAN). Firstly, we propose a new generator objective that finds it better to tackle mode collapse. And, we apply an independent Autoencoders (AE) to constrain the generator and consider its reconstructed samples as "real" samples to slow down the convergence of discriminator that enables to reduce the gradient vanishing problem and stabilize the model. Secondly, from mappings between latent and data spaces provided by AE, we further regularize AE by the relative distance between the latent and data samples to explicitly prevent the generator falling into mode collapse setting. This idea comes when we find a new way to visualize the mode collapse on MNIST dataset. To the best of our knowledge, our method is the first to propose and apply successfully the relative distance of latent and data samples for stabilizing GAN. Thirdly, our proposed model, namely Generative Adversarial Autoencoder Networks (GAAN), is stable and has suffered from neither gradient vanishing nor mode collapse issues, as empirically demonstrated on synthetic, MNIST, MNIST-1K, CelebA and CIFAR-10 datasets. Experimental results show that our method can approximate well multi-modal distribution and achieve better results than state-of-the-art methods on these benchmark datasets. Our model implementation is published here.[1]

Keywords: Generative Adversarial Network, Auto-encoders, Generative model, Variational Auto-encoders

1 Introduction

Generative Adversarial Networks [1] (GAN) have recently been being a dominant approach for learning generative models. It can produce very visually appealing samples but require few assumptions about the model. The core idea behind its huge success is that this approach can produce samples *without* explicitly estimating data distribution, e.g. in analytical forms. GAN has two main components competing for each other to improve themselves through the competition. The first component is the generator G taking low-dimensional random noise $\mathrm{z} \sim P_{\mathrm{z}}$ as an input and maps them into high-dimensional data samples,

[1] https://github.com/tntrung/gaan

$\mathrm{x} \sim P_{\mathrm{x}}$. The prior distribution P_{z} is often uniform or normal. Simultaneously, GAN uses the second component as a discriminator D to distinguish whether samples are drawn by the generator distribution P_G or data distribution P_{x}. Training GAN is an adversarial process that while the discriminator D learns to better distinguish real or fake samples, the generator G attempts to confuse the discriminator D into accepting its outputs as being real. The generator G uses discriminator's scores as a feedback mechanism, to improve itself over time and eventually can approximate the data distribution. Despite their success, GAN is known to be hard to train and requires the careful designs of model architectures [2,3]. For example, the imbalance between discriminator and generator capacities often leads to convergence issues, such as gradient vanishing, or mode collapse. Gradient vanishing is when the discriminator provides no informative gradient for the generator to learn. It often occurs as the discriminator distinguishes so well between "real" and "fake" samples before the generator can approximate the data distribution. Mode collapse is another crucial issue, in which the generator is collapsed into a typical parameter setting that always generates a low diversity of samples. Many GAN variants were proposed [4,5,6] trying to solve these problems, amongst of them are Autoencoders (AE) based GAN.

The advantage of using AE is that we can explicitly encode data samples into latent space and represent data samples with lower dimensions. It is not only potential for stabilizing GAN, but also applicable for other applications, such as dimensional reduction. AE was also used as part of a prominent class of generative models, Variational Autoencoders (VAE) [7,8,9], which are attractive for learning inference/generative models that can lead to better log-likelihoods [10]. These encouraged many recent works following this direction. They applied either encoders/decoders as an inference model to improve GAN training [11,12,13], or use AE to define the objective function of discriminators [14,15] or generators [16,17]. There are also other attempts at combining AE and GAN [18,19]. Those fusions are interesting; however, their answers for above problems remain unclear, e.g. what is the impact of AE? or how can their model discourage vanishing gradient and mode collapse problems?

In this work, we follow this fusion direction, but propose a new form of unifying AE and GAN. Moreover, we explain more explicitly how our model can stabilize better GAN training, e.g. avoiding the gradient vanishing, mode collapse issues, or approximating better data distribution. Our main contributions are three-fold: (i) We propose a new model of AE-based GAN, which consists of a new generator objective, and an independent AE to constrain the generator and using reconstructed samples to restrain the discriminator that enables the unified model to be stable (ii) To the best of our knowledge, none of earlier works ever really visualize the mode collapse in order to understand how it looks like rather than just viewing some collapsed data samples. Hence, we introduce a simple way to visualize mode collapse on MNIST which gives us a new intuitive analysis of mode collapse from latent point-of-view. (iii) Based on this analysis, we propose a new regularization term for AE to explicitly tackle this issue. Comparing to state of the art on synthetic and benchmark datasets, our method achieves better

stability, balance, and competitive standard scores. Furthermore, our model can potentially work well as considered either inference model or generation model.

2 Related Works

The issue of non-convergence still remains open for GAN research, in which gradient vanishing and mode collapse are its most common forms [2,20]. Many important variants of GAN have been proposed to tackle these limitations by improving GAN training. Improved GAN [5] introduced some techniques, e.g. feature matching, mini-batch discrimination, and historical averaging, which drastically reduced the mode collapse. Unrolled GAN [4] tried to change optimization process to address convergence and mode collapse. WGAN [6] provided better theoretical analysis of convergence properties for GAN. This GAN variant leveraged the Wasserstein distance, which shows converging better than Jensen Shannon (JS) divergence used in original GAN [1]. However, WGAN required that the discriminator must lie on the space of 1-Lipschitz functions, therefore, it had to enforce norm critics to the discriminator by weight-clipping tricks. WGAN-WP [21] stabilized WGAN by alternating the weight-clipping by penalizing the gradient norm of the interpolated samples.

An alternative approach is to integrate AE into the GAN. AAE [18] learned the inference by AE and matched the encoded latent distribution to given prior distribution by GAN. Constraining the generator with AE loss, which does not assure that the generator is able to approximate well data distribution and overcome mode missing. VAE/GAN [19] combined VAE and GAN into one single model and used feature-wise distance for reconstruction. Due to depending on VAE [7], VAEGAN also needed re-parameterization tricks for back-propagation or required access to an exact functional form of prior distribution. InfoGAN [22] learned the disentangled representation by maximizing the mutual information for inducing latent codes. EBGAN [14] introduced the energy-based model, in which the discriminator is considered as energy function minimized via reconstruction errors. BEGAN [15] extended EBGAN by optimizing Wasserstein distance between AE loss distributions. ALI [11] and BiGAN [12] encoded the data into latent and trained jointly the data/latent samples in GAN framework. This model can learn encoder/decoder models after training although no explicit form of AE was introduced. MDGAN [16] required two discriminators for two separate steps: manifold and diffusion. The manifold step tended to learn good AE, and the diffusion worked like original GAN excepts the constructed samples are used instead of real samples.

In the literature, VAEGAN and MDGAN are most related to our work in term of using AE to improve the generator. However, we remarkably differ from them: (1) VAEGAN combined KL divergence and reconstruction loss to train the inference model. By this way, it requires an exact form of prior distribution and re-parameterization tricks for solving the optimization via back-propagation. Our method constrains AE by the relative distance of data and latent spaces, which requires no tricks for optimization, and is effective with any given prior

Fig. 1. (a) Mode collapse observed by data samples of MNIST dataset, and (b) their corresponding latent variables of uniform distribution. This collapse is produced when considering the reconstruction as "fake" ($GAAN_1$-fake, defined the section of Experimental Results). The similar cases can easily be reproduced when using the small capacity of networks or with the unbalance design of generator/discriminator networks of GAN.

distribution (2) We do not need two discriminators for our model like MDGAN (3) VAEGAN considered the reconstructed samples as "fake", and MDGAN adopts this similarly in its manifold step. In contrast, we use them as "real" samples, which is important to constrain the discriminator in order to avoid gradient vanishing and reduce the mode collapse. (4) Two these methods regularize G by simply reconstruction loss, which is certainly not possible to solve the mode collapse. We will discuss this and show why we need further the regularization term for AE. In addition, we empirically found that MDGAN is hard to train and collapses seriously with many datasets, and VAEGAN diverges if learning rate does not decay. Our method has none of these problems.

3 Proposed method

Mode collapse is the common issue of GAN; however, there are very few works in the literature have specifically studied on it. In this section, we first contribute a way to visualize the mode collapse on MNIST dataset. And based on observation, we propose a new model, namely Generative Adversarial Autoencoder Networks (or GAAN), to solve this problem.

3.1 Mode collapse from latent viewpoint

Mode collapse occurs when "the generator collapses to a parameter setting where it always emits the same point. When collapse to a single mode is imminent, the gradient of the discriminator may point in similar directions for many similar points." [5]. We often look at the mode collapse by visualizing a few numbers of collapsed samples (mapped from the unknown random noises of a prior (latent) distribution). Fig. 1a is an example. However, the data space is high-dimensional that is difficult to imagine where data points are in the space. In contrast, the latent space is lower-dimensional and controllable, which is possible to draw the entire space. The problem is that GAN is not invertible to map the data samples

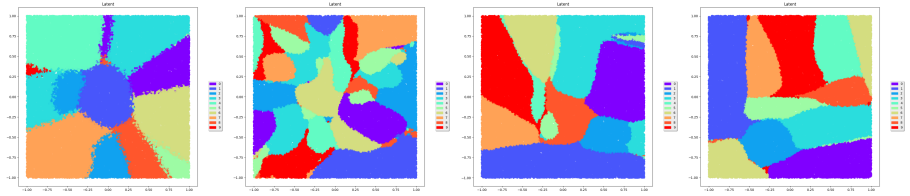

Fig. 2. The labels of 55K 2D latent variables obtained by (a) DCGAN, (b) WGANGP, (c) our $GAAN_2$ (without AE regularization) and (d) our $GAAN_3$ (with AE regularization). These GAAN settings are defined in the section of Experimental Results.

back the latent space. In order for that, we propose to use one off-the-shelf classifier. This classifier predicts labels of the generated samples and visualizes these labels according to their latent inputs. It is possible on MNIST dataset because the pre-trained classifiers now can achieve high accuracy, e.g. 0.04%. Note that the same technique is also likely applicable on CelebA if finding a pre-trained classifier with good enough accuracy. CIFAR-10 is more challenge as current GAN methods unable to generate realistic samples. Fig. 1b is the result of this technique applied to 2D latent space of $[-1, 1]$ uniform distribution. As shown in Fig. 1b, many latents in a large region of latent space are collapsed into the same digit number, e.g. '1', although some of them lie very far from each other in this space. In other words, a generator G_θ whose parameter θ is collapsed when there exists a large number N of latent variables $\{\mathrm{z_i}\}_{i=1}^{N}$ (amongst the number of random noise sampled in the latent space, e.g. $\geq 70\%$ in Fig. 1a) are mapped into a small region of data space $\{\mathrm{x_i}\}_{i=1}^{N}$. It means $\forall i, j \in [1, N]$:

$$\mathrm{x_i} = G_\theta(\mathrm{z_i}), \mathrm{x_j} = G_\theta(\mathrm{z_j}) : f(\mathrm{x_i}, \mathrm{x_j}) < \delta_\mathrm{x} \tag{1}$$

where f is one distance metric in the data space, and δ_x is a certain small threshold can be found in this space. There are surely many other ways to handle mode collapse. However, at this point-of-view, we expect to solve the mode collapse if there exists a distance metric in latent space g and small threshold δ_z of this metric enforcing G_θ such that:

$$g(\mathrm{z_i}, \mathrm{z_j}) > \delta_\mathrm{z} \rightarrow f(\mathrm{x_i}, \mathrm{x_j}) > \delta_\mathrm{x} \tag{2}$$

However, finding out good functions f, g for two different high-dimensional spaces and their thresholds $\delta_\mathrm{x}, \delta_\mathrm{z}$ is challenging. Moreover, applying these constraints to GAN is not simple as it is just mapping from latent to data samples. But, this observation gives us the good intuition that if we can regularize this mapping, we are able to overcome the mode collapse. This idea motivated us to find a solution being discussed in the next section.

We also try to apply the same technique on two state-of-the-art methods: DCGAN [3], WGANGP [21] on the MNIST dataset (using published code of [21]). Note that all of our experiments are in the unsupervised setting. Fig. 2a and Fig. 2b represent the labels of the 55K latent variables of DCGAN and

WGANGP respectively at iteration of 70K. From Fig. 2a, we understand that DCGAN is partially collapsed as it generates very few digits '5' and '9' according to their latent variables near the bottom-right top-left corners of the prior distribution. In contrast, WGANGP is un-collapsed as shown Fig. 2b, yet the latent variables representing each digit are fragmented in many sub-regions. It is an interesting observation on WGANGP, but it is not our current focus. Hence, we let further analysis for the future works.

3.2 Generative Adversarial Autoencoder Networks

We formulate the idea of Eqn. 2 by using an AE. We propose to use AE to encode data samples into latents and use these encoded latents to direct the generator's mapping from entire latent space. First, we train an independent AE (encoder E_ω and decoder G_θ) before training the discriminator D_γ and the generator G_θ once again later. Here, the generator is the decoder of AE and ω, θ, γ are the parameters of the encoder, generator, and discriminator respectively. Two main reasons of training an independent AE are: (i) to constrain the parameter θ at each training iteration, and (ii) to direct the generator generating samples similar to real training samples. Furthermore, knowing the arrangement of data samples in the latent space enables to better control the sampling of the generator. The idea will be formed as *a regularization* for AE, and the objective is written:

$$\min_{\omega,\theta} \mathcal{R}(\omega,\theta) + \lambda_{\mathrm{r}} \mathcal{W}(\omega,\theta) \tag{3}$$

where $\mathcal{R}(\omega,\theta) = ||\mathrm{x} - G_\theta(E_\omega(\mathrm{x}))||_2^2$ is the classical AE objective, whose goal is to compute encoded latents, and direct the generator. $\mathcal{W}(\omega,\theta)$ is to regularize θ and prevents it from being collapsed. This term will be discussed later. Here, λ_{r} is an regularization constant. The reconstructed samples $G_\theta(E_\omega(\mathrm{x}))$ can be approximated by $G_\theta(E_\omega(\mathrm{x})) = \mathrm{x}+\varepsilon$, where ε is the reconstruction error. Assume that the capacity of E and G are large enough so that ϵ is small (like noise) that means it's possible to consider those reconstructed samples as "real" samples (plus noise ε). However, this requirement is not always met, especially as working with high-quality images in reality. The pixel-wise reconstruction may cause the blurry issue. To circumvent this, we instead use feature-wise distance [19] or similarly feature matching [5]: $\mathcal{R}(\omega,\theta) = ||\Phi(\mathrm{x}) - \Phi(G_\theta(E_\omega(\mathrm{x})))||_2^2$. $\Phi(\mathrm{x})$ is the high-level feature can be obtained at some middle layers of deep networks. In our implementation, $\Phi(\mathrm{x})$ is the feature output from the last convolution layer of discriminator D_γ. Our framework is shown in Fig. 3. We propose to train encoder E_ω, generator G_θ and discriminator D_γ following the order: (i) fixing D_γ and train E_ω and G_θ to minimize the reconstruction loss Eqn. 3 (ii) fixing E_ω, G_θ, and training D_γ to maximize (Eqn. 5), and (iii) fixing E_ω, D_γ and training G_θ to minimize (Eqn. 4).

Generator and discriminator objectives Empirically, we find that when maximizing the alternative loss of the generator $\mathbb{E}_{\mathrm{z}}\sigma(D_\gamma(G_\theta(\mathrm{z})))$ [1], the generator tends to produce samples at high-density mode that easily leads to mode

collapse. Here, σ denotes the sigmoid function and $\mathbb{E}$ denotes the expectation. Instead, we straightforwardly target generator distribution to real distribution with the ℓ_1 distance of Eqn. 4. Ideally, the generator tries to generate samples as similar as possible the samples drawn from the real distribution, which is able to reduce part of missing mode issues.

$$\mathcal{L}_G(\theta) = |\mathbb{E}_{\mathrm{x}}\sigma(D_\gamma(\mathrm{x})) - \mathbb{E}_{\mathrm{z}}\sigma(D_\gamma(G_\theta(\mathrm{z})))| \tag{4}$$

The objective function of the discriminator is written in Eqn. 5. It is different from original discriminator of GAN at two main points. Firstly, we indicate the reconstructed samples as "real", represented by the term $\mathbb{E}_{\mathrm{x}} \log \sigma(D_\gamma(G_\theta(E_\omega(\mathrm{x}))))$. Considering the reconstructed samples as "real" to systematically slow down the convergence of discriminator, so that the gradient from discriminator is not saturated too quickly. In other words, the convergence of the discriminator depends on the convergence of AE. It's an important constraint. In contrast, if we consider the reconstruction as "fake" in our model, it speeds up the discriminator converging even faster than both generator and encoder that will lead to gradient saturation of D_γ. Fig. 1 is, for instance, the result, which shows serious mode collapse in this case with the same input of 2D latent as above. Secondly, we apply the gradient penalty for the discriminator objective (Eqn. 5), where λ_{p} is penalty coefficient, and $\hat{\mathrm{x}} = \epsilon \mathrm{x} + (1-\epsilon)G(\mathrm{z})$, ϵ is a uniform random number $\epsilon \in U[0,1]$. This penalty was used to enforce Lipschitz constraint of Wasserstein-1 distance [21]. In this work, we also find this term to be useful for JS divergence and make our model more stable. It forces the gradient of the discriminator not to be saturated. However, using this gradient penalty alone causes GAN diverged similar to WGANGP. Combining it with our proposed generator objective reduces this problem, but this combination does not perfectly solve, e.g. mode collapse on MNIST dataset with 2D latent inputs as Fig. 2c. Therefore, we propose the new regularization term for AE, $\mathcal{W}(\omega, \theta)$, and use them all together will effectively solve mode collapse.

$$\begin{aligned}\mathcal{L}_D(\omega,\theta,\gamma) =& \mathbb{E}_{\mathrm{x}} \log \sigma(D_\gamma(\mathrm{x})) + \mathbb{E}_{\mathrm{z}} \log(1 - \sigma(D_\gamma(G_\theta(\mathrm{z})))) \\ &+ \mathbb{E}_{\mathrm{x}} \log \sigma(D_\gamma(G_\theta(E_\omega(\mathrm{x})))) - \lambda_{\mathrm{p}} \mathbb{E}_{\hat{\mathrm{x}}}(||\nabla_{\hat{\mathrm{x}}} D_\gamma(\hat{\mathrm{x}})||_2^2 - 1)^2\end{aligned} \tag{5}$$

Autoencoder regularization In this section, we present the AE regularization $\mathcal{W}(\omega, \theta)$ to further stabilize our model. First, we observe the encoded latents obtained by our GAAN_2 model ($\lambda_{\mathrm{r}} = 0$ or without AE regularization) on MNIST dataset via the same technique as in Fig. 2a, 2b. Its 55K latent codes are shown in Fig. 2c. Although GAAN_2 worked pretty well on synthetic data, this figure reveals it is still partially collapsed, e.g. small regions of latent variables generating digits '4' and '5'. The areas covered by MNIST digits are significantly different in this space. If we use smaller network architectures, the mode collapse is even more serious. Intuitively, in order to avoid this problem, AE needs to assign more spaces for some digits, that allows random noise z falling into those digits' regions with higher probability.

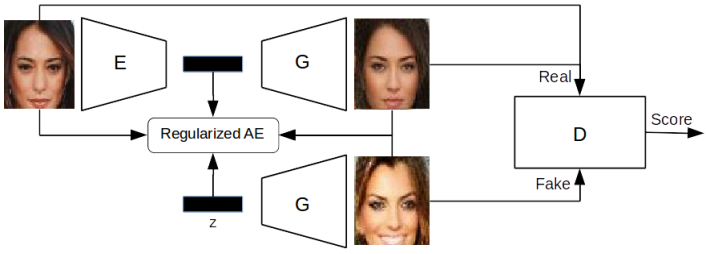

Fig. 3. The architecture of GAAN includes Encoder (E), Generator (G) and Discriminator (D). Reconstructed samples are considered as "real" ones. The input, reconstructed, and generated samples as well as the input noise and encoded latent are all used to form the regularized AE.

In order to do that, we can use noise input to constrain encoder's outputs, and simultaneously real samples to constrain the generator's outputs. We know that the mode collapse is when the generator creates the low diversity of samples in data space given different latent inputs. Therefore, by this constraint, we expect if the distance of any two latent variables $g(\mathrm{z_i}, \mathrm{z_j})$ is "close/far" in the latent space, the distance $f(\mathrm{x_i}, \mathrm{x_j})$ of their mappings into data space should be relatively "close/far" respectively, and vice versa. We propose a new regularization $\mathcal{W}(\omega, \theta)$ as the relative distance, describing our words in the written formula:

$$\mathcal{W}(\omega, \theta) = ||f(\mathrm{x}, G_\theta(\mathrm{z})) - \lambda_\mathrm{w} g(E_\omega(\mathrm{x}), \mathrm{z})||_2^2 \tag{6}$$

where f and g are distance functions computed in data and latent manifolds. λ_w is the scale factor required because of the dimensional difference between two spaces. In practice, it's hard to compare distances for two different high-dimensional manifolds. Therefore, rather than using the direct distance functions, e.g. Euclidean, ℓ_1-norm, etc, we instead compare the matching score $f(\mathrm{x}, G_\theta(\mathrm{z}))$ of real and fake distributions and the matching score $g(E_\omega(\mathrm{x}), \mathrm{z})$ of two latent distributions encoded from these data samples (matching scores measured by their means). The detail functions are defined as follows:

$$f(\mathrm{x}, G_\theta(\mathrm{z})) = \mathrm{M_d}(\mathbb{E}_\mathrm{x}\mathrm{x} - \mathbb{E}_\mathrm{z} G_\theta(\mathrm{z})) \tag{7}$$

$$g(E_\omega(\mathrm{x}), \mathrm{z}) = \mathrm{M_d}(\mathbb{E}_\mathrm{x} E_\omega(\mathrm{x}) - \mathbb{E}_\mathrm{z}\mathrm{z}) \tag{8}$$

where $\mathrm{M_d}$ computes the average of all dimensions of the input. Fig. 4a illustrates the density of 10000 random samples mapped by $\mathrm{M_d}$ from low to high dimensions of $[-1, 1]$ uniform space to one dimension. We see that the higher dimensional spaces get higher densities, smaller deviations. Therefore, matching two different densities of data and latent samples requires λ_w. Empirically, we found $\lambda_\mathrm{w} = \sqrt{\frac{d_\mathrm{z}}{d_\mathrm{x}}}$, where d_z and d_x are dimensions of latent and data samples respectively. Intuitively, $\mathcal{W}(\omega, \theta)$ can enforce the relative distance between

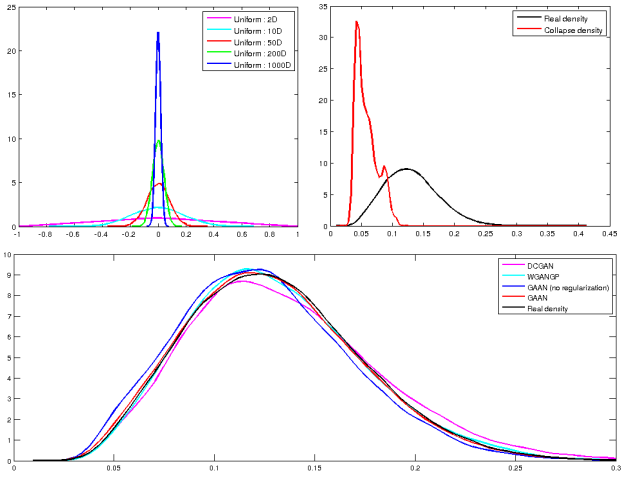

Fig. 4. (a) The 1D density of different dimensional uniform. (b) One example of the density of mode collapse case. (c) The 1D density of real data and generated densities obtained by different methods: DCGAN (kldiv: 0.00979), WGANGP (kldiv: 0.00412), GAAN without regularization (kldiv: 0.01027), and GAAN (kldiv: 0.00073).

two manifolds. When the generator is collapsed, $G_\theta(\mathrm{z})$ always produces similar outputs for any given inputs z that leads $f(\mathrm{x}, G_\theta(\mathrm{z}))$ to be large. Otherwise z is random and $E_\omega(\mathrm{x})$ are encoded from data, which are fixed value at each training epoch, it's difficult for $g(E_\omega(\mathrm{x}), \mathrm{z})$ to match the score of $f(\mathrm{x}, G_\theta(\mathrm{z}))$ to minimize the regularized AE objective. As a result, enforcing f and g can reduce drastically the mode collapse problem. Fig. 4b shows the density of one serious collapse mode case, in which the density of collapsed generated samples is significantly different from the real data density. Fig. 4c compares 1D densities of 55K MNIST samples generated by different methods from the density of real samples. Our GAAN method can estimate better 1D density than DCGAN and WGANGP based on computing KL divergence (kldiv) between the density of generated samples and density of real samples. This 1D mapping is not bijective, hence it is likely not always true that our method can approximate better data distribution than others. However, we see empirically that the generated 1D density is far from the real data density, it is more likely in collapse mode. It is a good evidence along with later empirical experiments to believe that our method is better than others in term of approximating the data distribution and handling the mode collapse issue. The diagram of our network architecture is illustrated in the Fig. 3 and the algorithm is presented in Algorithm. 1.

4 Experimental Results

In this section, we show that our method can improve the mode coverage on our synthetic datasets and the MNIST datasets, and generate quality faces on

Algorithm 1 Generative Adversarial Autoencoder Networks

Initialize discriminators, encoder and generator $D_\gamma, E_\omega, G_\theta$
repeat
  $\mathrm{x}^{\mathrm{m}} \leftarrow$ Random minibatch of m data points from dataset.
  $\mathrm{z}^{\mathrm{m}} \leftarrow$ Random m samples from noise distribution P_{z}
  // Training encoder and generator on x^{m} *and* z^{m} *by Eqn. 3*
  $\omega, \theta \leftarrow \min_{\omega,\theta} \mathcal{R}(\omega,\theta) + \lambda_{\mathrm{r}}\mathcal{W}(\omega,\theta)$
  // Training discriminators according to Eqn. 5 on $\mathrm{x}^{\mathrm{m}}, \mathrm{z}^{\mathrm{m}}$
  $\gamma \leftarrow \max_\gamma \mathcal{L}_D(\omega,\theta,\gamma)$
  // Training the generator on $\mathrm{x}^{\mathrm{m}}, \mathrm{z}^{\mathrm{m}}$ *according to Eqn. 4.*
  $\theta \leftarrow \max_\theta \mathcal{L}_G(\theta)$
until
return $E_\omega, G_\theta, D_\gamma$

CelebA dataset. We also measure standard scores and compare to the state of the art on MNIST-1K, CelebA, and CIFAR-10. All experiments are in the unsupervised setting.

4.1 Synthetic data

The purpose of experiments on synthetic data is to understand clearly how well a GAN method can approximate the data distribution. For that, we create our synthetic dataset of 25 Gaussian modes in grid layout similar to [11]. Our dataset contains 50K training points in 2D, and we draw 2K generated samples for testing. For a fair comparison, we use the equivalent architectures and setup for all methods in the same experimental condition if possible. The architecture and network size are similar to [4] on the 8-Gaussian dataset, except we use one more hidden layer. We use fully-connected layers and Rectifier Linear Unit (ReLU) activation for input and hidden layers, sigmoid for output layers. The network size of encoder, generator and discriminator are presented in Table 1 of Supplementary Material, where $d_{\mathrm{in}} = 2$, $d_{\mathrm{out}} = 2$, $d_{\mathrm{h}} = 128$ are dimensions of input, output and hidden layers respectively. $N_{\mathrm{h}} = 3$ is the number of hidden layers. The output dimension of the encoder is the dimension of the latent variable. Our prior distribution is the uniform of $[-1, 1]$. We use Adam optimizer with learning rate $\mathrm{lr} = 0.001$, and the exponent decay rate of first moment $\beta_1 = 0.8$. The learning rate is decayed very $10K$ steps with a base of 0.9. The mini-batch size is 128. The training stops after 500 epochs. To be fair, we carefully fine-tune other methods (and use weight decay during training if it gets better results) to ensure they achieve their best possible results on the synthetic data. For evaluation, the mode is missed if there are less than 20 generated samples registered into this mode, which is measured by its mean and variance of 0.01 [13,4]. A method has mode collapse if there are missing modes. In this experiment, we fix the parameters $\lambda_{\mathrm{r}} = 0.1$ (Eqn. 3), $\lambda_{\mathrm{p}} = 0.1$ (Eqn. 5), $\lambda_{\mathrm{w}} = 1.0$ (Eqn. 6). We conduct eight runs for each method and the final results for comparison are their average.

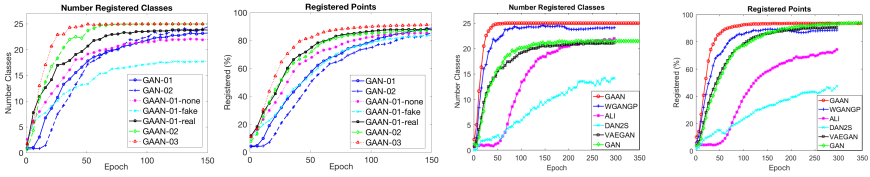

Fig. 5. From left to right figures: (a), (b), (c), (d). The number of registered modes (a) and points (b) of our method with two different settings on the synthetic dataset. We compare our GAAN to other methods on the same dataset measured on registered modes (classes) (c) and points (d).

At first, we highlight the capability of our model to approximate the distribution P_{x} of synthetic data. We carry out the ablation experiment to understand the influence of each proposed component. We define Setting 1 of our method (GAAN_1) without using AE regularization term and gradient penalty. This setting has three different versions as using reconstructed samples as "real", "fake" or "none" (not use it) in the discriminator objective. Setting 2 (GAAN_2) improves from GAAN_1 (real) by adding the gradient penalty. Setting 3 (GAAN_3) improves the GAAN_2 by adding the AE regularization. The quantitative results are shown in Fig. 5. Fig. 5a is the number of registered modes changing over the training. GAAN_1 misses few modes while GAAN_2, GAAN_3 recovers all 25 modes after about 50 epochs. Since they almost don't miss many modes, it's fair to compare also on the number of registered points as in Fig. 5b. Making use of reconstructed samples as "real" achieves better results than that of "fake" or "none". Using "fake" reconstructed samples, the model gets stuck and miss many modes. That highlights our choice of using "real" reconstructed samples. It's reasonable that GAAN_1 gets similar results as GAN when not using the reconstructed samples in discriminator objective ("none" option). Other results show the improvement when adding the gradient penalty (GAAN_2) into discriminator. GAAN_3 demonstrates the effectiveness of using AE constraints over the model of GAAN_2. To demonstrate the efficiency of the proposed generator objective, we use it to improve GAN as GAN_1. Then, we propose GAN_2 to improve GAN_1 by adding the gradient penalty. Although the combination of the new generator objective and gradient penalty makes GAN stable, there is a slight mode collapse. In contrast, our best GAAN settings can recover all modes and converge faster than accordingly improved GAN versions.

We compare our best setting (GAAN_3) to previous works, where ALI [11], and DAN-2S [13] are recent works using encoder/decoder in their model, VAE-GAN [19] introduces a similar model, and WGAN-GP [21] is one of the current state of the art of GAN. The numbers of covered modes and registered points are presented in Fig. 5c and Fig 5d respectively. The quantitative numbers of last epochs are shown in Table 2 of Supplementary Material. In this table, we report also Total Variation scores to measure the mode balance. The result for each method is the average of eight runs. Our method outperforms GAN [1], DAN-

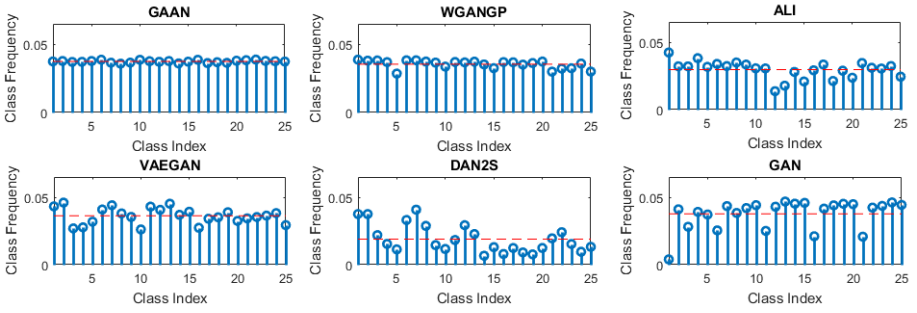

Fig. 6. The mode balance obtained by compared methods.

2S [13], ALI [11], and VAE/GAN [19] on the number of covered modes. While WGAN-GP sometimes misses one mode and diverges, our method ($GAAN_2$) does not suffer from any mode collapse on all eight runs. Furthermore, we achieve a higher number of registered samples than WGAN-GP and all others. Our method is also better than the rest with Total Variation (TV) [13]. Fig. 6 gives the detail proportion of generated samples of 25 modes. (See more visualization of generated samples in Section 2 of Supplementary Material).

4.2 MNIST-1K

For image datasets in general, we use $\Phi(\mathrm{x})$ instead x for the reconstruction loss and the AE regularization in order to avoid the blur. We fix the parameters $\lambda_\mathrm{p} = 1.0$, and $\lambda_\mathrm{r} = 1.0$ for all image datasets that work consistently well. The λ_w is automatically computed from dimensions of features $\Phi(\mathrm{x})$ and latent samples.

Our model implementation for MNIST based on the published code of WGAN-GP [21]. Fig. 7 from left to right are the real samples, the generated samples and mode frequency of each digit generated by our method on standard MNIST. It again confirms that our method can approximate well the MNIST digit distribution. Moreover, our generated samples looks realistic with different styles/strokes that is difficult to be distinguished from the real ones even by the human. The standard MNIST seems trivial to evaluate GAN methods on mode collapse. Hence, we follow a more challenging technique [4] to construct 1000-class MNIST (MNIST-1K) dataset. This dataset is built by stacking three random digits to form an RGB image (one digit per a channel). It can be assumed to have 1000 modes from 000 to 999. The digits are classified by a pre-train classifier (0.4% in our case). We create a total of 25,600 images and measure methods by counting the number of covered modes (having at least one sample [4]), and computing KL divergence. To be fair, we adopt the equivalent network architecture (low-capacity generator and two crippled discriminators K/4 and K/2) as proposed by [4]. Table 1 presents the mode number and KL divergence of compared methods. Results show that our method outperforms all others at a number of covered modes, especially with low-capacity discriminator (K/4 architecture),

Table 1. The comparison on MNIST-1K of methods. We follow the setup and network architectures from Unrolled GAN.

Architecture	GAN	Unrolled GAN	WGAN-GP	GAAN
K/4, #	30.6 ± 20.7	372.2 ± 20.7	640.1 ± 136.3	859.5 ± 68.7
K/4, KL	5.99 ± 0.04	4.66 ± 0.46	1.97 ± 0.70	1.04 ± 0.29
K/2, #	628.0 ± 140.9	817.4 ± 39.9	772.4 ± 146.5	917.9 ± 69.6
K/2, KL	2.58 ± 0.75	1.43 ± 0.12	1.35 ± 0.55	1.06 ± 0.23

Table 2. Comparing FID score to other methods.

	NS GAN	LSGAN	WGANGP	BEGAN	VAEGAN	GAAN
CelebA	58.0 ± 2.7	53.6 ± 4.2	26.8 ± 1.2	38.1 ± 1.1	27.5 ± 1.9	23.7 ± 0.3
CIFAR-10	58.6 ± 2.1	67.1 ± 2.9	52.9 ± 1.3	71.4 ± 1.1	58.1 ± 3.2	45.6 ± 1.2

that is higher about 150 modes than the next one. Our method reduces the gap between two architectures (e.g. about only $\sim$ 60 mode), which is smaller than than other works. On two architectures, we get better than others in terms of both KL divergence and the number of recovered modes. All results support our demonstration that we can handle better mode collapse and our model is robust even in case of the imbalance design of generator and discriminator.

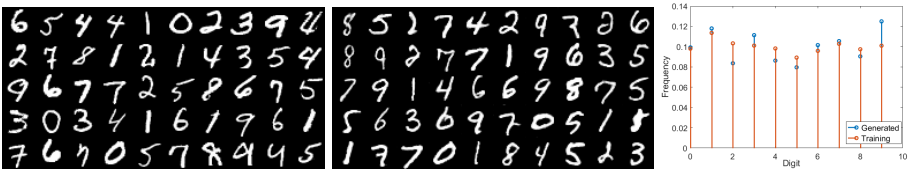

Fig. 7. The real samples and our generated samples of one mini-batch. And the number of generated samples per class obtained by our method on the standard MNIST dataset. We compare our frequency of generated samples to the ground-truth via KL divergence: KL = 0.01.

5 CelebA and CIFAR-10 datasets

We extend experiments of our GAAN on CelebA dataset and compare to DCGAN and WGAN-GP. Our implementation is based on the open source [24,23], which provided some available qualitative results of DCGAN and WGAN-GP. It's interesting to choose the architecture that DCGAN gets collapsed but our model is robust. Fig. 8 shows samples generated by DCGAN, WGANGP and our GAAN. While DCGAN is slightly collapsed at epoch 50, and WGAN-GP sometimes generates broken faces (look unlike human faces). Our method doesn't suffer from such problems and can generate recognizable and look-realistic faces.

Fig. 8. Generated samples of DCGAN (50 epochs, courtesy of [23]), WGAN-GP (50 epochs, courtesy of [23]) and our GAAN (50 epochs).

We also report our method on CIFAR-10 dataset based on DCGAN architecture of same published code [21]. See the generated samples of our method trained on this dataset in Section 4 of Supplementary Material. For quantitative numbers, we also report the FID scores [25] on two datasets. FID can detect intra-class mode dropping, and measure diversity, quality of generated samples. We follow the experimental procedure and model architecture [26], except the difference from [26] is that we only use the default parameter settings of our method rather than performing the wide range hyper-parameter search for the best setting. However, our method still outperforms others for both CelebA and CIFAR-10, shown in Table 2. We also report FID score of VAEGAN on these datasets. We again show our method is better than VAEGAN. Note that we tried MDGAN as well, but it is seriously collapsed on both two these datasets. Therefore, we do not report its result in our paper.

6 Conclusion

In this paper, we proposed a robust AE-based GAN model can handle effectively the mode collapse. Our model differs remarkably published GAN models on three important points: (i) We make use of independent AE, which can leverage its latent variables for further improvements. (ii) We propose a better generator objective for handling mode collapse and slow down the convergence of the discriminator by the reconstructed samples as considering them as "real" samples. (iii) We introduce a way to observe better the mode collapse from the latent viewpoint. The mode collapse visualization gives us the idea of regularizing AE in order to effectively prevent the generator falling into mode collapse settings. Extensive experiments prove that our method can approximate well the multi-modal distribution of synthetic data, and reduce drastically the mode collapse on MNIST-1K. Importantly, our model is stable and does not suffer from mode collapse as evaluated on MNIST, CelebA, and CIFAR-10 datasets. Furthermore, we achieve better results than previous works on all these benchmark datasets, which strongly demonstrates the efficiency of the proposed model.

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NISP. (2014) 2672–2680
2. Goodfellow, I.: Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160 (2016)
3. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
4. Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.: Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163 (2016)
5. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: NISP. (2016) 2234–2242
6. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
7. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
8. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: ICML. (2014) 1278–1286
9. Burda, Y., Grosse, R., Salakhutdinov, R.: Importance weighted autoencoders. arXiv preprint arXiv:1509.00519 (2015)
10. Wu, Y., Burda, Y., Salakhutdinov, R., Grosse, R.: On the quantitative analysis of decoder-based generative models. arXiv preprint arXiv:1611.04273 (2016)
11. Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., Courville, A.: Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016)
12. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)
13. Li, C., Alvarez-Melis, D., Xu, K., Jegelka, S., Sra, S.: Distributional adversarial networks. arXiv preprint arXiv:1706.09549 (2017)
14. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016)
15. Berthelot, D., Schumm, T., Metz, L.: Began: Boundary equilibrium generative adversarial networks. arXiv preprint arXiv:1703.10717 (2017)
16. Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W.: Mode regularized generative adversarial networks. CoRR (2016)
17. Warde-Farley, D., Bengio, Y.: Improving generative adversarial networks with denoising feature matching. (2016)
18. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)
19. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300 (2015)
20. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862 (2017)
21. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028 (2017)
22. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems. (2016) 2172–2180

23. https://github.com/LynnHo/DCGAN-LSGAN-WGAN-WGAN-GP-Tensorflow
24. https://github.com/carpedm20/DCGAN-tensorflow
25. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a nash equilibrium. CoRR (2017)
26. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are gans created equal? a large-scale study. CoRR (2017)