Generative Adversarial Autoencoder Networks

Ngoc-Trung Tran, Tuan-Anh Bui, and Ngai-Man Cheung
Singapore University of Technology and Design (SUTD)
8 Somapah Rd, 487372, Singapore
{ngoctrung_tran,tuananh_bui,ngaiman_cheung}@sutd.edu.sg

Abstract

In this paper, we introduce a simple but effective way to overcome the problem of mode collapse when training Generative Adversarial Networks (GAN). First, we constrain the generator by an independent Autoencoders (AE), and consider the reconstructed samples as "real" to constrain the discriminator. This consideration is important to systematically slow down the convergence of discriminator, that enables to avoid the gradient vanishing problem and stabilize the model. Second, the AE provides mappings between latent and data spaces, which can be potentially used to tackle explicitly the mode collapse. Based on this, we further regularize the AE by relative distance between the latent and data spaces to prevent the generator falling into mode collapse setting. Our proposed model, namely Generative Adversarial Autoencoders Networks (GAAN), is stable and has suffered neither gradient vanishing nor mode collapse problems as empirically demonstrated on synthetic, MNIST, MNIST-1K, CelebA and CIFAR-10 datasets. Experimental results also show that our method can infer well multi-modal distribution and achieve better results than state-of-the-art methods on these benchmark datasets. In addition, we open sources of our models for research purpose.

1. Introduction

Generative Adversarial Networks [12] (GAN) has recently been being a dominant approach for learning generative models. It can produce very visually appealing samples, but require few assumption about the model. The core idea behind its huge success is the 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 into high-dimensional data samples, $\mathrm{x} \sim P_{\mathrm{x}}$. 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 [11]. For example, the imbalance between discriminator and generator capacities often leads to convergence issues, such as gradient vanishing, or mode collapse. Gradient vanishing happens when the discriminator provides no informative gradient for the generator to learn. It often occurs as the discriminator distinguishes so well between the 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 [20, 23, 4] to stabilize GAN, amongst of them are Autoencoders (AE) based GAN.

The advantage of using AE is we can explicit encode data samples to latent space and control them from the latent space. It is not only potential for stabilizing GAN, but also applicable for other applications, such as dimensional reduction. AE was also used in another prominent class of generative models, Variational Autoencoders (VAE) [15, 22, 6], which are attractive for learning inference/generative models that can lead to better log-likelihoods [24]. These encouraged many recent works following this direction. They applied either encoders/decoders as an inference model to improve GAN training [10, 9, 17], or use AE to define the objective function of discriminators [25, 5] or generators [7]. There are also other attempts at combining AE and GAN [19, 16]. 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 problems, and approximating better data distribution. Our main contributions are three-fold: (i) We propose a new model of AE based GAN, in which we use AE to constrain both generator and discriminator to overcome the gradient vanishing, and reduce drastically mode collapse (ii) We provide a new intuitive analysis of mode collapse in latent point-of-view. (iii) Based on this analysis, we introduce a new regularization term for AE to explicitly tackle the mode collapse. 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 work well as considered either inference model or generation model.

2. Related Works

The issue of non-convergence still remains open for research direction, in which gradient vanishing, mode collapse are its most common forms [11, 3]. Many important variants of GAN have been proposed to tackle these limitations by improving GAN training. Improved GAN [23] introduced some techniques, *e.g.* feature matching, minibatch discrimination, and historical averaging, which drastically reduced the mode collapse. Unrolled GAN [20] tried to change optimization process to address convergence and mode collapse. WGAN [4] 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 [12]. 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 [13] 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 [19] learned the inference by AE and matched the encoded latent distribution to given prior distribution. The decoder or generator was constrained by reconstruction loss, which does not guarantee the generator can well approximate data distribution and avoid mode collapse. VAE/GAN [16] combined VAE and GAN into one single model and used feature-wise distance to achieve better quality images. Due to depending on VAE [15], VAEGAN also needed reparameterization tricks for back-propagation or required access to exact functional form of prior distribution. InfoGAN [8] learned the disentangled representation by maximizing the mutual information for inducing latent codes. EBGAN [25] introduced the energy-based model, in which the discriminator is considered as energy function minimized via reconstruction errors. BEGAN [5] extended EBGAN by optimizing Wasserstein distance between AE loss distributions. ALI [10] and BiGANs [9] encoded the data into latent and learned 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 [7] 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.

VAEGAN and MDGAN are most related to our method in term of using AE to improve the generator. However, we remarkably differ from them: (1) VAEGAN used KL divergence for AE regularization. By this way, it requires exact form of prior distribution and re-parameterization tricks for solving the optimization via back-propagation. Our method constrains AE by relative distance of data and latent space, which requires no trick for optimization, and stably works well with arbitrary distribution (2) We do not require two discriminators for our model like MDGAN (3) VAEGAN considers reconstructed samples as "fake", and MDGAN adopts this similarly in its manifold step. In contrast, we consider it as "real" can be helpful to constrain the discriminator and avoid gradient vanishing. It is an important observation being discussed later (4) Two methods regularizes G by simply AE loss, which is likely unable to overcome the mode collapse problem; we will discuss on this and show why we need further the regularization term for AE. In addition, we empirically found that MDGAN is not easy 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 has specifically studied on it. In this section, we first contribute a way to visualize the mode collapse on MNIST dataset. And based on observations, 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." [23]. We often see the mode collapse happened in data space as visualizing samples as mapped from random noise (of a prior (latent) distribution) as shown in Fig. 1a. However, the data space is high-dimensional that is

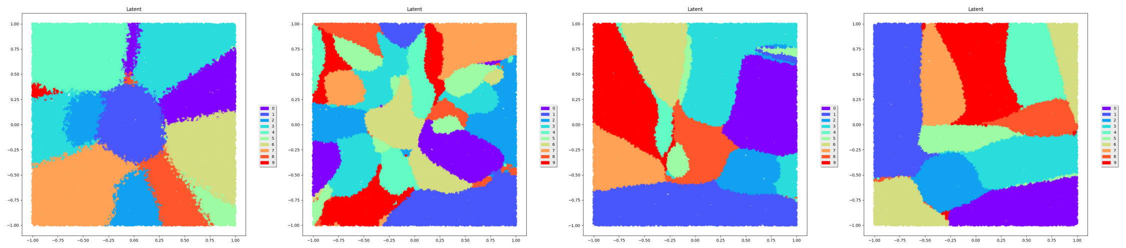
Figure 2. The label of 55K 2D latent variables by (a) DCGAN, (b) WGANGP, (c) our GAAN without our AE regularization and (d) our GAAN with our AE regularization. The latent labels predicted by using a pre-trained classifier on samples generated at the iteration of 70K.

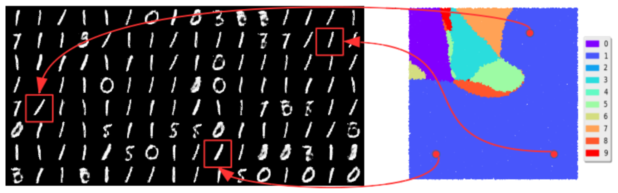
Figure 1. (a) Mode collapse observed at data space, and (b) its corresponding latent variables of uniform distribution. This happens when considering the reconstruction as "fake" in our model. The similar cases can easily be reproduced when using small network capacity or with some unbalance design of discriminators of GAN on this dataset.

difficult to visualize. Nevertheless, the latent space is lower-dimensional and controllable that we can handle. Problem is that GAN is not invertible to map from the data space to latent space. Therefore, we propose to use one off-the-shelf classifier to predict labels of the generated samples and visualize these labels according to their latent variables in latent space. It is possible on MNIST dataset because the pre-trained classifiers can achieve high accuracy on it, *e.g.* 0.04%. Fig. 1b is the result when applying this technique on 2D latent space of $[-1, 1]$ uniform distribution. As shown in Fig. 1, different latents z in the collapsed mode can be mapped into the same digit number, *e.g.* '1', although they lie far from each other in the latent 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 $\forall i, j \in [1, N]$:

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

where f is one distance metric in the data space, and δ_x is a certain small threshold can be found in this space. What we expect to solve the mode collapse if there exists a distance metric in latent space g and threshold δ_z of this metric such that:

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

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 constrains to GAN is difficult since GAN is only mapping from latent sample to data sample. However, this observation gives us the good intuition that if we can control generator mapping regions from latent space and data space we can avoid the mode collapse. This idea motivated us to find a solution that will be discussed in the next section.

We also try to apply the same technique on two state-of-the-art methods: DCGAN [21], WGANGP [13] on the MNIST dataset (using open source published by [13]). 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. In Fig. 2a, we see that DCGAN generates very few digits '5' and '9' according to their latent variables near the bottom-right top-left corners of the distribution. This figure shows that DCGAN is partially collapsed. In contrast, WGANGP can generate all digit numbers as shown Fig. 2b, yet the latent variables representing one digit are fragmented in many a few sub-regions. It is interesting observation on WGANGP, but it is not what we focus on this moment. Hence, we let further analysis in future work.

3.2. Generative Adversarial Autoencoder Networks

We realize the idea formulated by equations (1) and (2) by using an AE. We propose to use AE to encode data samples into latents and use these encoded latents to direct how generator would generate samples 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 encoder, generator and discriminators 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 samples of training set. Furthermore, we know how data samples are arranged in the latent space. This information enables to better control the sampling of generator, will realized by *an regularization* for AE. Our *regularized* AE objective can be 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 objective of classical AE whose goal is to obtain encoded latents, and direct the generator. And the regularization $\mathcal{W}(\omega,\theta)$ is to regularize θ and prevent it 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 [16] or similarly feature matching [23]: $\mathcal{R}(\omega,\theta) = ||\Phi(\mathrm{x}) - \Phi(G_\theta(E_\omega(\mathrm{x})))||_2^2$. $\Phi(\mathrm{x})$ 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 train D_γ to maximize (Eqn. 4), and (iii) fixing E_ω, D_γ and train G_θ to maximize (Eqn. 4). The generator objective is similar to GAN, but we constrain the discriminator as discussed in the next section.

3.2.1 Discriminator objective

The objective function of our discriminator is written in Eqn. 4. It is different from original discriminator of GAN at two main points. First, 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" is able to systematically slow down the convergence of discriminator, so that we can avoid gradient vanishing. In other words, we constrain the discriminator by our reconstructed samples, so that 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 can help the discriminator converging faster than both generator and encoder that will lead to the similar issues of gradient vanishing from D_γ or mode collapse like GAN. Fig. 1 is, for instance, the result when we considered the reconstructed sample as "fake" in our model. Second, we apply the gradient penalty for the discriminator objective (Eqn. 4), 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

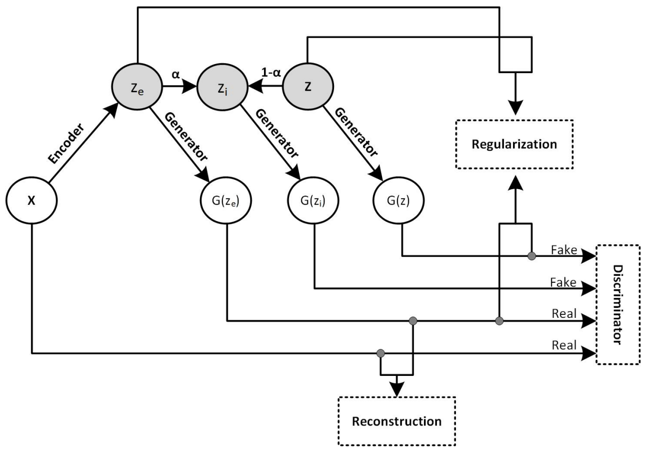

Figure 3. Our model architecture.

Lipschitz constraint of Wasserstein-1 distance [13]. In this work, we explore that this term can provide the good gradient to direct the learning of our generator. So, we empirically found that adding this term to our model makes our model more stable. However, using this gradient penalty alone doesn't solve common issues of GAN, *e.g.* mode collapse, for example on MNIST dataset. Therefore, we propose the new regularization term of AE $\mathcal{W}(\omega,\theta)$. Using this gradient penalty and the new proposed regularization in combination significantly stabilize our model. In all our experiments, this penalty will be applied, and we just discuss on the AE regularization, which can effectively solve the mode collapse in our work.

3.2.2 Autoencoder regularization

In last section, we use reconstructed samples to constrain the discriminator convergence. In this section, we present our proposed regularization $\mathcal{W}(\omega,\theta)$ which constrain AE to overcome the mode collapse. First, we observe the encoded latents obtained by our GAAN model without AE regularization ($\lambda_{\mathrm{r}} = 0$) on MNIST dataset. Its 55K latent codes of standard MNIST digits is shown in Fig. 2c, which is obtained by the similar technique as in Fig. 2a, 2b. This figure reveals partially mode collapse, *e.g.* small regions of latent variables generating digits '4' and '5'. We see the regions covered by MNIST digits are significantly unbalanced in this latent space. For many other cases, if we use small network architectures, the mode collapse is even more serious, *e.g.* generating mostly digits '1' for most of noise inputs. Intuitively, in order to avoid this problem and achieve the mode balance, the AE needs to create more latent space for some digits, that allows random noise z can fall into those regions with higher probability. In other words, some latent variables need to be spread out more space and occupy similar areas in this space as others, so that the probability that random noise z fall in these digit regions is to generate the

$$\mathcal{L}(\omega,\theta,\gamma) = \mathbb{E}_{\mathrm{x}} \log\sigma(D_\gamma(\mathrm{x})) + \mathbb{E}_{\mathrm{z}} \log(1-\sigma(D_\gamma(G_\theta(\mathrm{z})))) + \mathbb{E}_{\mathrm{x}} \log\sigma(D_\gamma(G_\theta(E_\omega(\mathrm{x})))) + \lambda_{\mathrm{p}}\mathbb{E}_{\hat{\mathrm{x}}}(||\nabla_{\hat{\mathrm{x}}} D_\gamma(\hat{\mathrm{x}})||_2^2 - 1)^2 \tag{4}$$

equivalent number of digits.

In order to do that, we can use simultaneously noise input to constrain the encoder, and real samples to constrain the generator. We know that the mode collapse occurs 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:

$$\mathcal{W}(\omega,\theta) = ||f(\mathrm{x}, G_\theta(\mathrm{z})) - \lambda_{\mathrm{w}} g(E_\omega(\mathrm{x}), \mathrm{z})||_2^2 \tag{5}$$

where f and g are distance functions computed in data space and latent space. 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, L1 norm, etc, we instead constrain the matching score $f(\mathrm{x}, G_\theta(\mathrm{z}))$ of real and fake distributions by 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:

$$f(\mathrm{x}, G_\theta(\mathrm{z})) = \mathrm{M_d}(\mathbb{E}_{\mathrm{x}\sim P_{\mathrm{x}}} \mathrm{x} - \mathbb{E}_{\mathrm{z}\sim P_{\mathrm{z}}} G_\theta(\mathrm{z})) \tag{6}$$

$$g(E_\omega(\mathrm{x}), \mathrm{z}) = \mathrm{M_d}(\mathbb{E}_{\mathrm{x}\sim P_{\mathrm{x}}} E_\omega(\mathrm{x}) - \mathbb{E}_{\mathrm{z}\sim P_{\mathrm{z}}} \mathrm{z}) \tag{7}$$

where $\mathrm{M_d}$ computes the average of all dimensions of latent or data samples. Because latent and data samples have different dimension, we mapping each samples into 1D dimension by $\mathrm{M_d}$. Fig. 4a shows the density of 10000 random samples mapped from different $[-1, 1]$ uniform space to 1D dimension by $\mathrm{M_d}$. We see that the higher dimensional spaces has higher density, smaller deviation. Therefor, matching two different densities of data and latent samples requires λ_{w}. Empirically, we found $\lambda_{\mathrm{w}} = \sqrt{\frac{d}{D}}$, where d and D are dimensions of latent and data samples respectively. Intuitively, $\mathcal{W}(\omega, \theta)$ can enforce the relative distance between two spaces. We can see that if the generator is collapsed, $G(\mathrm{z})$ always produces the same output 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 be always large as $f(\mathrm{x}, G_\theta(\mathrm{z}))$ when minimizing the regularized AE. Fig. 4b shows the density of one serious collapse mode case, in which the density of collapsed generated samples is much different from the real data density. As a result, enforcing f and g can reduce drastically the mode collapse problem. Fig. 4c compares 1D densities of 55K MNIST samples generated by different methods and also the density of real samples. Our GAAN method can estimate better 1D density than DCGAN and WGANGP based on computing KL divergence between the density of generated samples and density of real samples. This 1D mapping is not bijective, hence it is 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 one good of evidence for us to believe our method is better than others in term of approximating the data distribution and handle the mode collapse issue. The diagram of our network architecture is illustrated in the Fig. 3 and the detailed algorithm is presented in Alg. 1.

4. Experimental Results

In this section, we demonstrate our method that can improve the mode coverage on our synthetic datasets and the MNIST datasets and can generate quality faces on CelebA dataset. We also measure some standard scores and compare to the state of the art. All experiments are in the unsupervised setting.

4.1. Synthetic data

The purpose of experiments on synthetic data is the see how well the generator can approximate the data distribution. For that, we create our synthetic dataset has 25 Gaussian modes in grid layout similar to [10], Fig 6. Our dataset contains 50K training points in 2D, and we draw 2K generated samples for testing. For 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 [20] on the 8-Gaussian dataset, but here we use one more hidden layer. We use only fully-connected layers, and use Rectifier Linear Unit (ReLU) activation for input and hidden layers, sigmoid for output layers. The network structure of encoder, generator and discriminator are presented in Table 1, where d_{in}, d_{out}, d_{h} are dimension of input, output and hidden layers

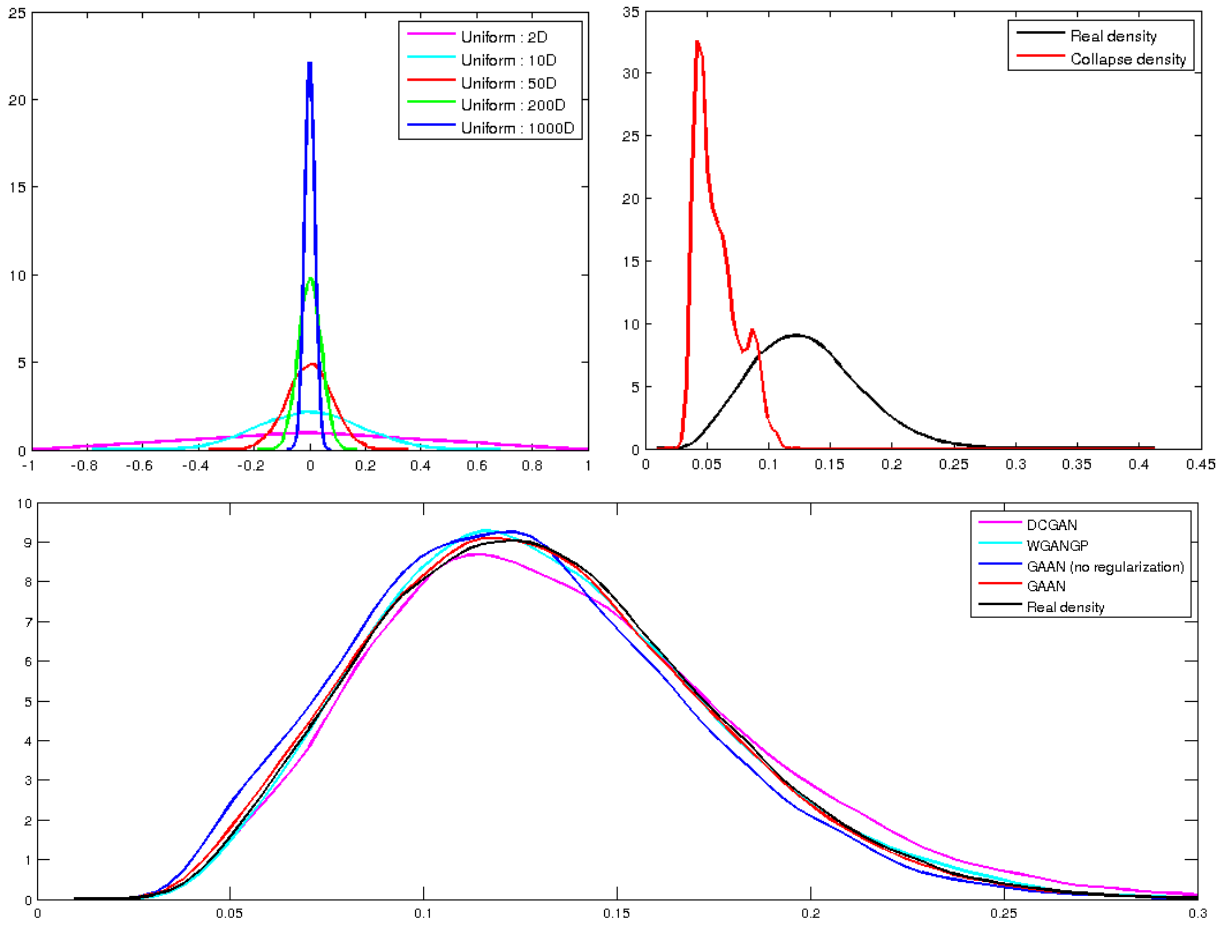

Figure 4. (a) The 1D density of different dimensional uniform. (b) One example of the density of mode collapse case. (c) The 1D density of real data and generated densities obtained by different methods: DCGAN (kldiv: 0.00979), WGANGP (kldiv: 0.00412), GAAN without regularization (kldiv: 0.01027), and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

Algorithm 1 Generative Adversarial Autoencoder Network

```
 1: Initialize discriminators, encoder and generator $D_\gamma, E_\omega, G_\theta$
 2: **repeat**
 3:     $\mathrm{x^m} \leftarrow$ Random minibatch of $m$ data points from dataset.
 4:     $\mathrm{z^m} \leftarrow$ Random $m$ samples from noise distribution $P_\mathrm{z}$
 5:     // *Learn encoder and generator on* $\mathrm{x^m}$ *and* $\mathrm{z^m}$ *by Eqn. 3*
 6:     $\omega, \theta \leftarrow \min_{\omega,\theta} \mathcal{R}(\omega,\theta) + \lambda_\mathrm{r} \mathcal{W}(\omega,\theta)$
 7:     // *Learn discriminators of data and latent by Eqn. 4 on* $\mathrm{x^m}, \mathrm{z^m}$
 8:     $\gamma \leftarrow \max_\gamma \mathcal{L}(\omega,\theta,\gamma)$
 9:     // *Learn once more the generator.*
10:     $\theta \leftarrow \max_\theta \mathbb{E}_\mathrm{z} \log(D_\gamma(G_\theta(\mathrm{z^m})))$
11: **until**
12: **return** $E, G, D$
```

respectively. N_h is the number of hidden layers. The output dimension of the encoder is the dimension of the latent variable. Our prior noise is uniform distribution 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 as best as possible results

	d_{in}	d_{out}	N_{h}	d_{h}
Encoder (E)	2	2	3	128
Generator (G)	2	2	3	128
Discriminator (D)	2	1	3	128

Table 1. The network structures are used for evaluating on synthetic data.

on the synthetic data. For evaluation, a mode is collapsed if there are less than 20 generated samples registered into this mode, which is measured by its mean and variance of 0.01 [17, 20]. For all experiments, we fix the parameters $\lambda_{\text{r}} = 0.1$ (Eqn. 3), $\lambda_{\text{p}} = 0.1$ (Eqn. 4), $\lambda_{\text{w}} = 1.0$ (Eqn. 5). We conduct eight runs for one method and the final results for comparison are their average.

At first, we highlight the capability of our model to approximate the distribution P_{x} through experiments on our synthetic dataset. We evaluate our proposed method in different settings to understand the influence of each proposed component to the overall performance of our method. We define Setting 1 of our method (GAAN$_1$) of using AE without the regularization term and the gradient penalty in our discriminator objective. Setting 2 (GAAN$_2$) improves from GAAN$_1$ by adding the regularization to AE. The quantitative results are shown in the first row of Fig. 5. The left figures are the number registered modes changing over the training. GAAN$_2$ cover all 25 modes faster than GAAN$_1$, and can cover all the modes after about 100 epochs. GAAN$_1$ sometimes missed one mode. Because they almost don't miss many modes, it's fair to compare also on the number of registered points as on the right figures. GAAN$_2$ demonstrates the effectiveness of using AE constraints over the first model of GAAN$_1$.

We compare our method to previous works, where ALI [10], and DAN-2S [17] are recent works using encoder/decoder in their method, VAE-GAN [16] introduces a similar model, and WGAN-GP [13] is one of the current state of the art of GAN. The number of covered modes and registered points is presented in Fig 5, and the quantitative numbers are in Table 2. In this table, we report also Total Variation scores to measure the mode balance. The reported results for each method are the average of eight runs. Our method outperforms GAN [12], DAN-2S [17], ALI [10], and VAE/GAN [16] on the number of covered modes. While WGAN-GP sometimes misses one mode, we show our stability as not suffering 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 all others in Total Variation (TV) [17]. Fig. 6 visualizes generated samples of compared methods. Fig. 7 gives the detail of the proportion of generated samples of 25 modes.

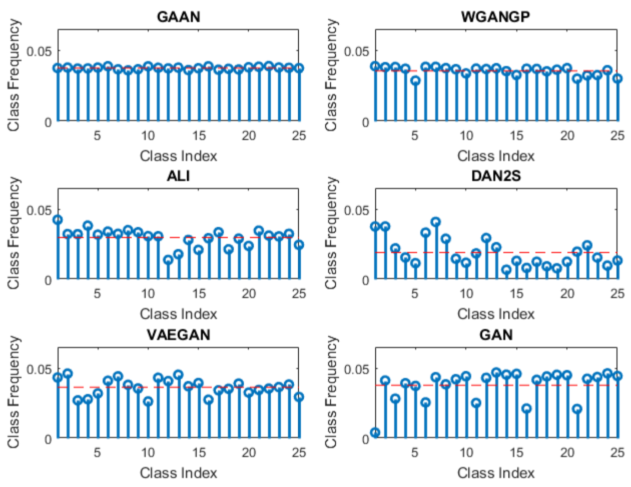

Figure 7. The mode balance obtained by compared methods.

4.2. MNIST-1K

For image datasets, we use $\Phi(\text{x})$ instead x for the reconstruction loss and the AE regularization in order to avoid the blurry problem. We fix the parameters $\lambda_{\text{p}} = 1.0$, and $\lambda_{\text{r}} = 1$ for all datasets that work consistently well. The λ_{w} cane be automatically computed based on dimensions of features $\Phi(\text{x})$ and latent samples.

Our implementation on MNIST based on the open source published by WGAN-GP [13]. Fig. 8 is the generated samples and mode frequency of each digit generated by our method on standard MNIST. It 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 even by the human. The standard MNIST seems trivial to compare methods on mode collapse. Hence, we follow a more challenging technique [20] 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 modes is covered (at least one sample). We also compute 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 [20]. Table 3 presents the mode number and KL divergence of our method comparing to other published methods. Compare to others on MNIST-1K, our method outperforms all others at a number of covered modes, especially with low-capacity discriminator (K/4 architecture), that is higher about 150 modes than the next one. Our method achieves the small gap between two architectures (*e.g.* differ about only ~ 60 mode), and this gap is small as compared with other works. Our

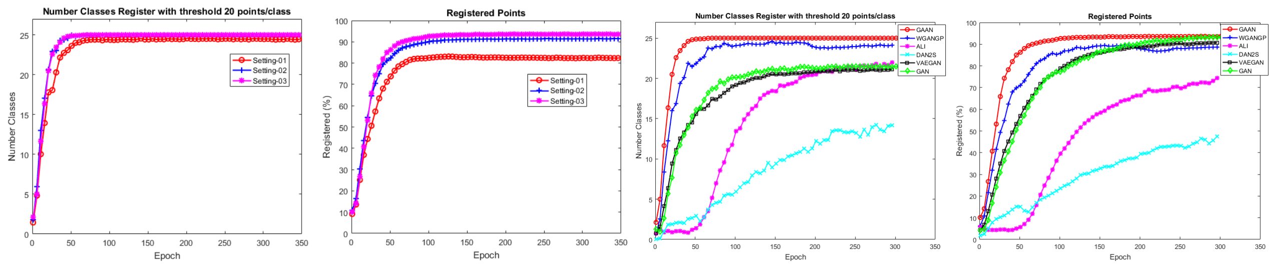

Figure 5. Comparison the number of registered modes (left) and points (right) on the synthetic dataset. The first row indicates the influence of each of our components. The second row compares to other methods.

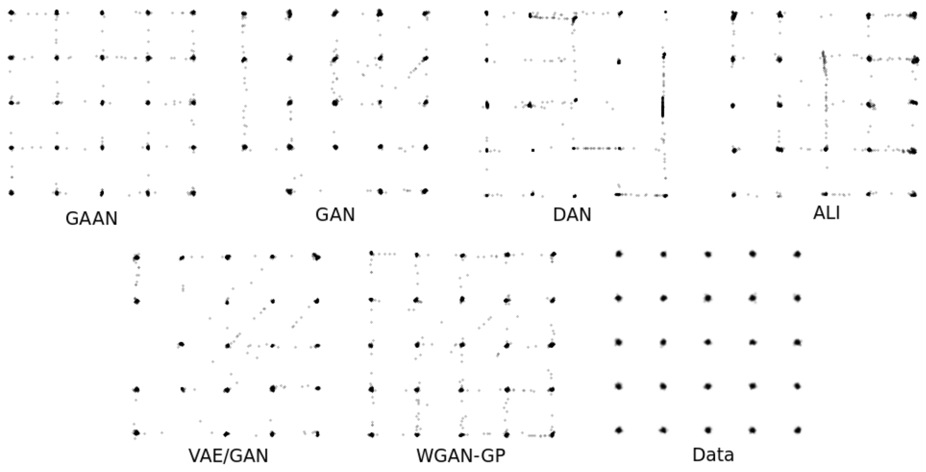

Figure 6. Visualizing samples generated by compared methods on the synthetic dataset.

method achieves similar KL divergence as the state-of-the-art method, WGAN-GP. All results support the demonstration of our capability to handle a large number of modes and the stability of our training even in case of the imbalance between generator and discriminator.

5. CelebA and Cifar10 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 [1, 2], which provided some available qualitative results of DCGAN and WGAN-GP. It's interesting to choose the architecture that DCGAN collapsed and we show that we are robust with the same architecture. Fig. 9 shows some samples generated by our GAAN, DCGAN, and WGANGP. While DCGAN is slightly collapsed at epoch 50, and WGAN-GP sometimes generates broken faces (not look like human faces). Our method doesn't suffer such problems and can generate recognizable and look-realistic faces. We interpolate the latent variables to generate faces as shown in Fig. 10 which smoothly changes the generated faces. That confirms the semantic mapping that our method can infer.

We also report our method on Cifar-10 dataset based on DCGAN architecture from the open source published by WGANGP's authors [13]. Fig. 11 shows the generated samples of our method trained on this dataset. We also report our FID scores [14] on two datasets. FID can detect intra-class mode dropping, and measure diversity, quality of generated samples. We follow the experimental procedure [18], except the difference from [18] 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 Cifar10, shown in Table 4. We also report also 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 most of them. Therefore, we do not report its result in our paper.

6. Conclusion

In this paper, we propose a new GAN model by leveraging the AE and its latent variables. First, we introduce the model can simply/effectively integrating AE into GAN. In this model, we constrain the discriminator by the reconstructed samples. In addition, we introduce a way to ob-

Method	#registered modes	#registered points	TV (True)	TV (Differential)
GAN [12]	21.50 ± 2.00	1875.63 ± 42.53	1.00 ± 0.00	0.94 ± 0.02
DAN-2S [17]	14.07 ± 2.85	986.33 ± 160.80	0.99 ± 0.02	0.70 ± 0.09
VAE-GAN [16]	21.13 ± 2.33	1816.19 ± 61.23	0.98 ± 0.15	0.93 ± 0.21
ALI [10]	22.00 ± 2.63	1490.25 ± 194.38	0.93 ± 0.11	0.68 ± 0.15
WGAN-GP [13]	24.04 ± 1.16	1766.25 ± 79.48	0.54 ± 0.35	0.43 ± 0.31
GAAN$_1$	24.52 ± 1.00	1652.89 ± 148.22	0.47 ± 0.26	0.31 ± 0.19
GAAN$_2$	25.00 ± 0.00	1827.38 ± 25.60	0.20 ± 0.03	0.12 ± 0.02

Table 2. The network structures are used for evaluating synthetic data. Columns indicate the number of covered modes, and the number of registered samples among 2000 generated samples, and two types of Total Variation (TV).

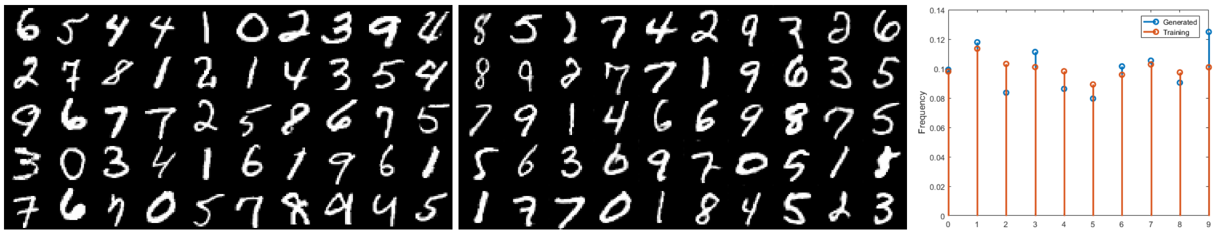

Figure 8. The real samples and our generated samples of one mini-batch. And the number of generated samples per class obtained by our method on the standard MNIST dataset. We compare our frequency of generated samples to the ground-truth via KL divergence: KL = 0.01.

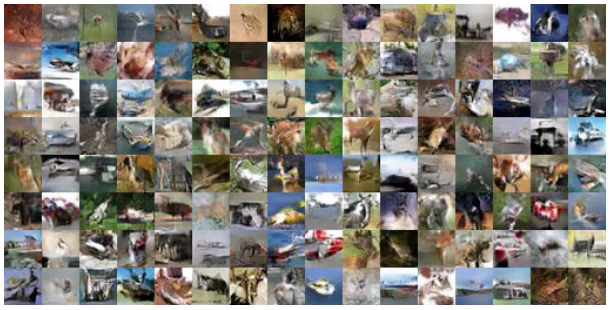

Figure 11. Generated samples of our method trained on Cifar10.

serve the mode collapse from latent viewpoint. The observation of this visualization motivated us to propose a new robust model for this problem. Based on that, we introduce to use the regularization for AE enables to constrain better the generator that can achieve better mode balance and avoid the mode collapse. Throughout experiments, we show that our method can approximate well the multi-modal distribution of training data on synthetic and MNIST-1K. Our method does not suffer mode collapse issues as evaluating on both the synthetic, MNIST, MNIST-1K, CelebA and Cifar10 datasets. Furthermore, we achieve better results than previous works on all benchmark datasets. We demonstrate the efficiency of using good regularizations can direct better the generator of GAN to produce better samples.

References

[1] `https://github.com/carpedm20/DCGAN-tensorflow`.

[2] `https://github.com/LynnHo/DCGAN-LSGAN-WGAN-WGAN-GP-Tensorflow`.

[3] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[5] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[6] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[7] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. *CoRR*, 2016.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

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

[10] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[11] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NISP*, pages 2672–2680, 2014.

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

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

	NS GAN	LSGAN	WGANGP	BEGAN	VAEGAN	Ours
CelebA	58.0 ± 2.7	53.6 ± 4.2	26.8 ± 1.2	38.1 ± 1.1	27.5 ± 0.98	23.7 ± 0.3
Cifar10	58.6 ± 2.1	67.1 ± 2.9	52.9 ± 1.3	71.4 ± 1.1	58.1 ± 3.2	45.6 ± 1.2

Table 4. Comparing FID score to other methods.

[14] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, 2017.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[17] C. Li, D. Alvarez-Melis, K. Xu, S. Jegelka, and S. Sra. Distributional adversarial networks. *arXiv preprint arXiv:1706.09549*, 2017.

[18] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study. *CoRR*, 2017.

[19] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

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

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

[22] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286, 2014.

[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NISP*, pages 2234–2242, 2016.

[24] Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

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

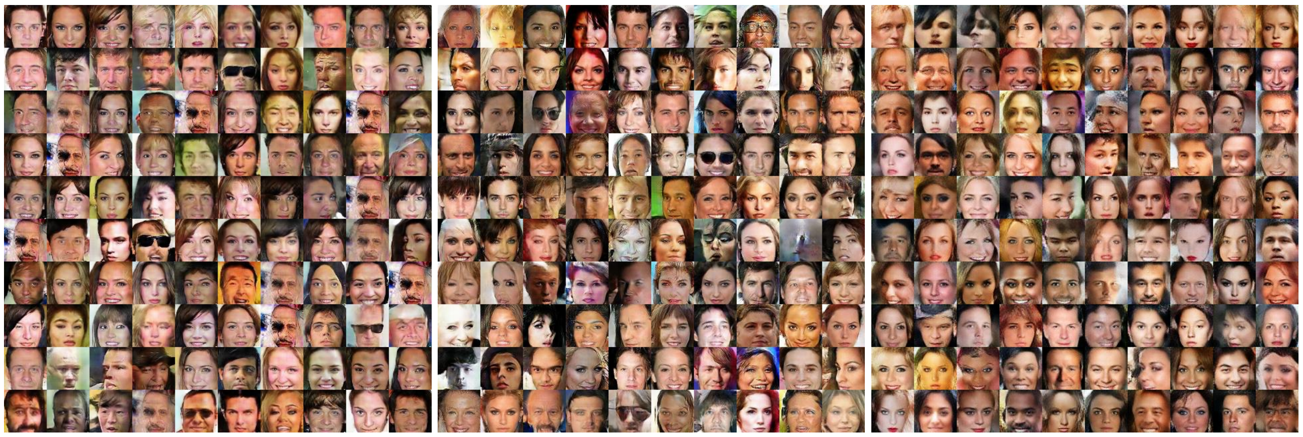
Figure 9. Generated samples of DCGAN (50 epochs, courtesy of [2]), WGAN-GP (50 epochs, courtesy of [2]) and our GAAN (50 epochs).

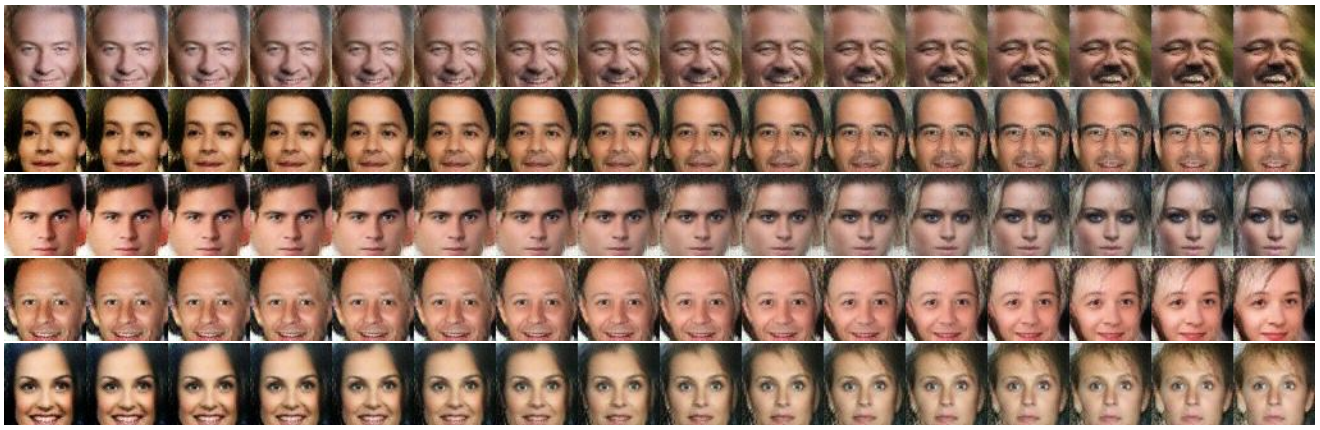
Figure 10. Interpolated samples by our method.