MODE REGULARIZED GENERATIVE ADVERSARIAL NETWORKS

[†]**Tong Che**,[*] [‡]**Yanran Li**,[*] [†,§]**Athul Paul Jacob**, [†]**Yoshua Bengio**, [‡]**Wenjie Li**
[†]Montreal Institute for Learning Algorithms, Université de Montréal, Montréal, QC H3T 1J4, Canada
[‡]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[§]David R. Cheriton School of Computer Science, University Of Waterloo, Waterloo, ON N2L 3G1, Canada
{tong.che,ap.jacob,yoshua.bengio}@umontreal.ca
{csyli,cswjli}@comp.polyu.edu.hk

ABSTRACT

Although Generative Adversarial Networks achieve state-of-the-art results on a variety of generative tasks, they are regarded as highly unstable and prone to miss modes. We argue that these bad behaviors of GANs are due to the very particular functional shape of the trained discriminators in high dimensional spaces, which can easily make training stuck or push probability mass in the wrong direction, towards that of higher concentration than that of the data generating distribution.

We introduce several ways of regularizing the objective, which can dramatically stabilize the training of GAN models. We also show that our regularizers can help the fair distribution of probability mass across the modes of the data generating distribution, during the early phases of training and thus providing a unified solution to the missing modes problem.

1 INTRODUCTION

Generative adversarial networks (GAN) (Goodfellow et al., 2014) have demonstrated their potential on various tasks, such as image generation, image super-resolution, 3D object generation, and video prediction (Radford et al., 2015; Ledig et al., 2016; Sønderby et al., 2016; Nguyen et al., 2016; Wu et al., 2016; Mathieu et al., 2015). The objective is to train a parametrized function (the generator) which maps noise samples (e.g., uniform or Gaussian) to samples whose distribution is close to that of the data generating distribution. The basic scheme of the GAN training procedure is to train a discriminator which assigns higher probabilities to real data samples and lower probabilities to generated data samples, while simultaneously trying to move the generated samples towards the real data manifold using the gradient information provided by the discriminator. In a typical setting, the generator and the discriminator are represented by deep neural networks.

Despite their success, GANs are generally considered as very hard to train due to training instability and sensitivity to hyper-parameters. On the other hand, a common failure pattern observed while training GANs is the collapsing of large volumes of probability mass onto a few modes. Namely, although the generators produce meaningful samples, these samples are often from just a few modes (small regions of high probability under the data distribution). Behind this phenomenon is the missing modes problem, which is widely conceived as a major problem for training GANs: many modes of the data generating distribution are not at all represented in the generated samples, yielding a much lower entropy distribution, with less variety than the data generating distribution.

This issue has been the subject of several recent papers proposing several tricks and new architectures to stabilize GAN's training and encourage its samples' diversity. However, we argue that a general cause behind these problems is the lack of control on the discriminator during GAN training. We would like to encourage the manifold of the samples produced by the generator to move towards that of real data, using the discriminator as a metric. However, even if we train the discriminator to distinguish between these two manifolds, we have no control over the shape of the discriminator function in between these manifolds. In fact, the shape of the discriminator function in the data

[*]Authors contributed equally.

space can be very non-linear with bad plateaus and wrong maxima and this can therefore hurt the training of GANs (Figure 1).

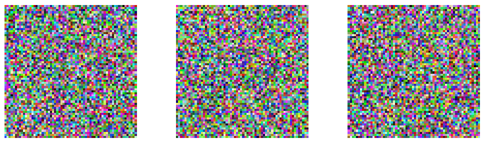
Figure 1: Samples with very high discrimination values (D=1.0) in DCGAN model trained on CelebA dataset.

To remedy this problem, we propose a novel regularizer for the GAN training target. The basic idea is simple yet powerful: in addition to the gradient information provided by the discriminator, we want the generator to take advantage of other similarity metrics with much more predictable behavior, such as the L_2 norm. Differentiating these similarity metrics will provide us with more stable gradients to train our generator. Combining this idea with an approach meant to penalize the missing modes, we propose a family of additional regularizers for the GAN objective. We then design a set of metrics to evaluate the generated samples in terms of both the diversity of modes and the distribution fairness of the probability mass. These metrics are shown to be more robust in judging complex generative models, including those which are well-trained and collapsed ones.

Regularizers usually bring a trade-off between model variance and bias. Our results have shown that, when correctly applied, our regularizers can dramatically reduce model variance, stabilize the training, and fix the missing mode problem all at once, with positive or at the least no negative effects on the generated samples. We also discuss a variant of the regularized GAN algorithm, which can even improve sample quality as compared to the DCGAN baseline.

2 Related Work

The GAN approach was initially proposed by Goodfellow et al. (2014) where both the generator and the discriminator are defined by deep neural networks.

In Goodfellow et al. (2014), the GAN is able to generate interesting local structure but globally incoherent images on various datasets. Mirza & Osindero (2014) enlarges GAN's representation capacity by introducing an extra vector to allow the generator to produce samples conditioned on other beneficial information. Motivated from this, several conditional variants of GAN has been applied to a wide range of tasks, including image prediction from a normal map Wang & Gupta (2016), image synthesis from text Reed et al. (2016) and edge map Isola et al. (2016), real-time image manipulation Zhu et al. (2016), temporal image generation Zhou & Berg (2016); Saito & Matsumoto (2016); Vondrick et al. (2016), texture synthesis, style transfer, and video stylization Li & Wand (2016).

Researchers also aim at stretching GAN's limit to generate higher-resolution, photo-realistic images. Denton et al. (2015) initially apply a Laplacian pyramid framework on GAN to generate images of high resolution. At each level of their LAPGAN, both the generator and the discriminator are convolutional networks. As an alternative to LAPGAN, Radford et al. (2015) successfully designs a class of deep convolutional generative adversarial networks which has led to significant improvements on unsupervised image representation learning. Another line of work aimed at improving GANs are through feature learning, including features from the latent space and image space. The motivation is that features from different spaces are complementary for generating perceptual and natural-looking images. With this perspective, some researchers use distances between learned features as losses for training objectives for generative models. Larsen et al. (2015) combine a variational autoencoder objective with a GAN and utilize the learned features from the discriminator in the GANs for better image similarity metrics. It is shown that the learned distance from the discriminator is of great help for the sample visual fidelity. Recent literature have also shown impressive results on image super-resolution to infer photo-realistic natural images for 4x upscaling factors Ledig et al. (2016); Sønderby et al. (2016); Nguyen et al. (2016).

Despite these promising successes, GANs are notably hard to train. Although Radford et al. (2015) provide a class of empirical architectural choices that are critical to stabilize GAN's training, it would be even better to train GANs more robustly and systematically. Salimans et al. (2016) propose feature matching technique to stabilize GAN's training. The generator is required to match the statistics of intermediate features of the discriminator. Similar idea is adopted by Zhao et al. (2016).

In addition to feature distances, Dosovitskiy & Brox (2016) found that the counterpart loss in image space further improves GAN's training stability. Furthermore, some researchers make use of information in both spaces in a unified learning procedure (Dumoulin et al., 2016; Donahue et al., 2016). In Dumoulin et al. (2016), one trains not just a generator but also an encoder, and the discriminator is trained to distinguish between two joint distributions over image and latent spaces produced either by the application of the encoder on the training data or by the application of the generator (decoder) to the latent prior. This is in contrast with the regular GAN training, in which the discriminator only attempts to separate the distributions in the image space. Parallelly, Metz et al. (2016) stabilize GANs by unrolling the optimization of discriminator, which can be considered as an orthogonal work with ours.

Our work is related to VAEGAN (Larsen et al., 2015) in terms of training an autoencoder or VAE jointly with the GAN model. However, the variational autoencoder (VAE) in VAEGAN is used to generate samples whereas our autoencoder based losses serves as a regularizer to penalize missing modes and thus improving GAN's training stability and sample qualities. We demonstrate detailed differences from various aspects in Appendix D.

3 Mode Regularizers for GANs

The GAN training procedure can be viewed as a non-cooperative two player game, in which the discriminator D tries to distinguish real and generated examples, while the generator G tries to fool the discriminator by pushing the generated samples towards the direction of higher discrimination values. Training the discriminator D can be viewed as training an evaluation metric on the sample space. Then the generator G has to take advantage of the local gradient $\nabla \log D(G)$ provided by the discriminator to improve itself, namely to move towards the data manifold.

We now take a closer look at the root cause of the instabilities while training GANs. The discriminator is trained on both generated and real examples. As pointed out by Goodfellow et al. (2014); Denton et al. (2015); Radford et al. (2015), when the data manifold and the generation manifold are disjoint (which is true in almost all practical situations), it is equivalent to training a characteristic function to be very close to 1 on the data manifold, and 0 on the generation manifold. In order to pass good gradient information to the generator, it is important that the trained discriminator produces stable and smooth gradients. However, since the discriminator objective does not directly depend on the behavior of the discriminator in other parts of the space, training can easily fail if the shape of the discriminator function is not as expected. As an example,Denton et al. (2015) noted a common failure pattern for training GANs which is the vanishing gradient problem, in which the discriminator D perfectly classifies real and fake examples, such that around the fake examples, D is nearly zero. In such cases, the generator will receive no gradient to improve itself.[1]

Another important problem while training GANs is mode missing. In theory, if the generated data and the real data come from the same low dimensional manifold, the discriminator can help the generator distribute its probability mass, because the missing modes will not have near-0 probability under the generator and so the samples in these areas can be appropriately concentrated towards regions where D is closer to 1. However, in practice since the two manifolds are disjoint, D tends to be near 1 on all the real data samples, so large modes usually have a much higher chance of attracting the gradient of discriminator. For a typical GAN model, since all modes have similar D values, there is no reason why the generator cannot collapse to just a few major modes. In other words, since the discriminator's output is nearly 0 and 1 on fake and real data respectively, the generator is not penalized for missing modes.

3.1 Geometric Metrics Regularizer

Compared with the objective for the GAN generator, the optimization targets for supervised learning are more stable from an optimization point of view. The difference is clear: the optimization target for the GAN generator is a learned discriminator. While in supervised models, the optimization targets are distance functions with nice geometric properties. The latter usually provides much easier training gradients than the former, especially at the early stages of training.

[1]This problem exists even when we use $\log D(G(z))$ as target for the generator, as noted by Denton et al. (2015) and our experiments.

Inspired by this observation, we propose to incorporate a supervised training signal as a regularizer on top of the discriminator target. Assume the generator $G(z) : Z \to X$ generates samples by sampling first from a fixed prior distribution in space Z followed by a deterministic trainable transformation G into the sample space X. Together with G, we also jointly train an encoder $E(x) : X \to Z$. Assume d is some similarity metric in the data space, we add $\mathbb{E}_{x\sim p_d}[d(x, G\circ E(x))]$ as a regularizer, where p_d is the data generating distribution. The encoder itself is trained by minimizing the same reconstruction error.

In practice, there are many options for the distance measure d. For instance, the pixel-wise L^2 distance, or the distance of learned features by the discriminator (Dumoulin et al., 2016) or by other networks, such as a VGG classifier. (Ledig et al., 2016)

The geometric intuition for this regularizer is straight-forward. We are trying to move the generated manifold to the real data manifold using gradient descent. In addition to the gradient provided by the discriminator, we can also try to match the two manifolds by other geometric distances, say, L^s metric. The idea of adding an encoder is equivalent to first training a point to point mapping $G(E(x))$ between the two manifolds and then trying to minimize the expected distance between the points on these two manifolds.

3.2 Mode Regularizer

In addition to the metric regularizer, we propose a mode regularizer to further penalize missing modes. In traditional GANs, the optimization target for the generator is the empirical sum $\sum_i \nabla_\theta \log D(G_\theta(z_i))$. The missing mode problem is caused by the conjunction of two facts: (1) the areas near missing modes are rarely visited by the generator, by definition, thus providing very few examples to improve the generator around those areas, and (2) both missing modes and non-missing modes tend to correspond to a high value of D, because the generator is not perfect so that the discriminator can take strong decisions locally and obtain a high value of D even near non-missing modes.

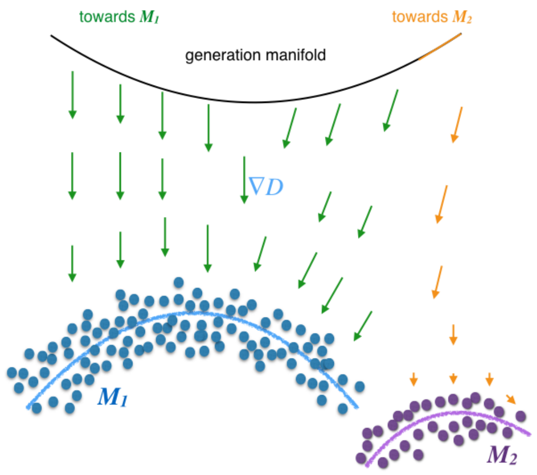

Figure 2: Illustration of missing modes problem.

As an example, consider the situation in Figure 2. For most z, the gradient of the generator $\nabla_\theta \log D(G_\theta(z))$ pushes the generator towards the major mode M_1. Only when $G(z)$ is very close to the mode M_2 can the generator get gradients to push itself towards the minor mode M_2. However, it is possible that such z is of low or zero probability in the prior distribution p_0.

Given this observation, consider a regularized GAN model with the metric regularizer. Assume M_0 is a minor mode of the data generating distribution. For $x \in M_0$, we know that if $G \circ E$ is a good autoencoder, $G(E(x))$ will be located very close to mode M_0. Since there are sufficient training examples of mode M_0 in the training data, we add the mode regularizer $\mathbb{E}_{x\sim p_d}[\log D(G \circ E(x))]$ to our optimization target for the generator, to encourage $G(E(x))$ to move towards a nearby mode of the data generating distribution. In this way, we can achieve fair probability mass distribution across different modes.

In short, our regularized optimization target for the generator and the encoder becomes:

$$T_G = -\mathbb{E}_z[\log D(G(z))] + \mathbb{E}_{x\sim p_d}[\lambda_1 d(x, G\circ E(x)) + \lambda_2 \log D(G\circ E(x))] \quad (1)$$

$$T_E = \mathbb{E}_{x\sim p_d}[\lambda_1 d(x, G\circ E(x)) + \lambda_2 \log D(G\circ E(x))] \quad (2)$$

3.3 Manifold-Diffusion Training for Regularized GANs

On some large scale datasets, CelebA for example, the regularizers we have discussed do improve the diversity of generated samples, but the quality of samples may not be as good without carefully tuning the hyperparameters. Here we propose a new algorithm for training metric-regularized GANs, which is very stable and much easier to tune for producing good samples.

The proposed algorithm divides the training procedure of GANs into two steps: a manifold step and a diffusion step. In the manifold step, we try to match the generation manifold and the real data manifold with the help of an encoder and the geometric metric loss. In the diffusion step, we try to distribute the probability mass on the generation manifold fairly according to the real data distribution.

An example of manifold-diffusion training of GAN (MDGAN for short) is as follows: we train a discriminator D_1 which separates between the samples x and $G \circ E(x)$, for x from the data, and we optimize G with respect to the regularized GAN loss $\mathbb{E}[\log D_1(G \circ E(x)) + \lambda d(x, G \circ E(x))]$ in order to match the two manifolds. In the diffusion step we train a discriminator D_2 between distributions $G(z)$ and $G \circ E(x)$, and we train G to maximize $\log D_2(G(z))$. Since these two distributions are now nearly on the same low dimensional manifold, the discriminator D_2 provides much smoother and more stable gradients. The detailed training procedure is given in Appendix A. See Figure 6 for the quality of generated samples.

3.4 Evaluation Metrics for Mode Missing

In order to estimate both the missing modes and the sample qualities in our experiments, we used several different metrics for different experiments instead of human annotators.

The inception score (Salimans et al., 2016) was considered as a good assessment for sample quality from a labelled dataset:

$$\exp\left(\mathbb{E}_x KL(p(y|\mathbf{x}) || p^*(y))\right) \tag{3}$$

Where $\mathbf{x}$ denotes one sample, $p(y|x)$ is the softmax output of a trained classifier of the labels, and $p^*(y)$ is the overall label distribution of generated samples. The intuition behind this score is that a strong classifier usually has a high confidence for good samples. However, the inception score is sometimes not a good metric for our purpose. Assume a generative model that collapse to a very bad image. Although the model is very bad, it can have a perfect inception score, because $p(y|x)$ can have a high entropy and $p^*(y)$ can have a low entropy. So instead, for labelled datasets, we propose another assessment for both visual quality and variety of samples, the MODE score:

$$\exp\left(\mathbb{E}_x KL(p(y|\mathbf{x}) || p(y)) - KL(p^*(y) || p(y))\right) \tag{4}$$

where $p(y)$ is the distribution of labels in the training data. According to our human evaluation experiences, the MODE score successfully measures two important aspects of generative models, i.e., variety and visual quality, in one metric.

However, in datasets without labels (LSUN) or where the labels are not sufficient to characterize every data mode (CelebA), the above metric does not work well. We instead train a third party discriminator between the real data and the generated data from the model. It is similar to the GAN discriminator *but is not used to train the generator*. We can view the output of the discriminator as an estimator for the quantity (See (Goodfellow et al., 2014) for proof):

$$D^*(s) \approx \frac{p_g(s)}{p_g(s) + p_d(s)} \tag{5}$$

Where p_g is the probability density of the generator and p_d is the density of the data distribution. To prevent D^* from learning a perfect 0-1 separation of p_g and p_d, we inject a zero-mean Gaussian noise to the inputs when training D^*. After training, we test D^* on the test set T of the real dataset. If for any test sample $t \in T$, the discrimination value $D(t)$ is close to 1, we can conclude that the mode corresponding to t is missing. In this way, although we cannot measure exactly the number of modes that are missing, we have a good estimator of the total probability mass of all the missing modes.

4 EXPERIMENTS

4.1 MNIST

We perform two classes of experiments on MNIST. For the MNIST dataset, we can assume that the data generating distribution can be approximated with ten dominant modes, if we define the term "mode" here as a connected component of the data manifold.

4.1.1 GRID SEARCH FOR MNIST GAN MODELS

In order to systemically explore the effect of our proposed regularizers on GAN models in terms of improving stability and sample quality, we use a large scale grid search of different GAN hyper-parameters on the MNIST dataset. The grid search is based on a pair of randomly selected loss weights: $\lambda_1 = 0.2$ and $\lambda_2 = 0.4$. We use the same hyper-parameter settings for both GAN and Regularized GAN, and list the search ranges in Table 1. Our grid search is similar to those proposed in Zhao et al. (2016). Please refer to it for detailed explanations regarding these hyper-parameters.

Table 1: Grid Search for Hyperparameters.

nLayerG	[2,3,4]
nLayerD	[2,3,4]
sizeG	[400,800,1600,3200]
sizeD	[256, 512, 1024]
dropoutD	[True,False]
optimG	[SGD,Adam]
optimD	[SGD,Adam]
lr	[1e-2,1e-3,1e-4]

For evaluation, we first train a 4-layer CNN classifier on the MNIST digits, and then apply it to compute the MODE scores for the generated samples from all these models. The resulting distribution of MODE score is shown in Figure 3. Clearly, our proposed regularizer significantly improves the MODE scores and thus demonstrates its benefits on stabilizing GANs and improving sample qualities.

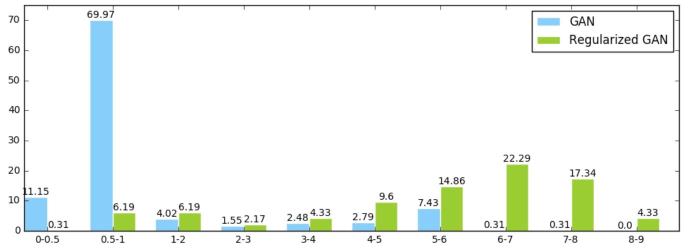

Figure 3: The distributions of MODE scores for GAN and regularized GAN.

To illustrate the effect of regularizers with different coefficients, we randomly pick an architecture and train it with different $\lambda_1 = \lambda_2$. The results are shown in Figure 4.

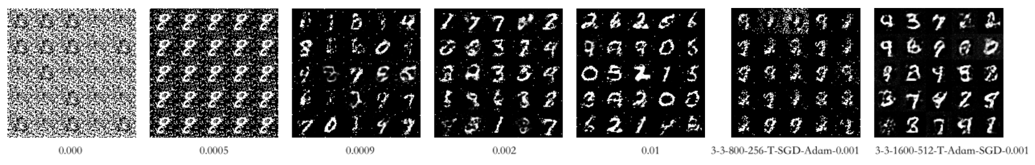

Figure 4: (Left 1-5) Different hyperparameters for MNIST generation. The values of the λ_1 and λ_2 in our Regularized GAN are listed below the corresponding samples. (Right 6-7) Best samples through grid search for GAN and Regularized GAN.

4.1.2 COMPOSITIONAL MNIST DATA WITH 1000 MODES

In order to quantitatively study the effect of our regularizers on the missing modes, we concatenate three MNIST digits to a number in [0,999] in a single 64x64 image, and then train DCGAN as a baseline model on the 1000 modes dataset. The digits on the image are sampled with different probabilities, in order to test the model's capability to preserve small modes in generation. We again use a pre-trained classifier for MNIST instead of a human to evaluate the models.

Table 2: Results for Compositional MNIST with 1000 modes. The proposed regularization (Reg-DCGAN) allows to substantially reduce the number of missed modes as well as the KL divergence that measures the plausibility of the generated samples (like in the Inception score).

	Set 1		Set 2		Set 3		Set 4	
	#Miss	KL	#Miss	KL	#Miss	KL	#Miss	KL
DCGAN	204.7	77.9	204.3	60.2	103.4	75.9	89.3	77.8
Reg-DCGAN	**32.1**	**62.3**	**71.5**	**58.9**	**42.7**	**68.4**	**31.6**	**67.8**

The performances on the compositional experiment are measured by two metrics. #Miss represents the classifier-reported number of missing modes, which is the size of the set of numbers that the model never generates. KL stands for the KL divergence between the classifier-reported distribution of generated numbers and the distribution of numbers in the training data (as for the Inception score). The results are shown in Table 2. With the help of our proposed regularizer, both the number of missing modes and KL divergence drop dramatically among all the sets of the compositional MNIST dataset, which again proves the effectiveness of our regularizer for preventing the missing modes problem.

4.2 CelebA

To test the effectiveness of our proposal on harder problems, we implement an encoder for the DCGAN algorithm and train our model with different hyper-parameters together with the DCGAN baseline on the CelebA dataset. We provide the detailed architecture of our regularized DCGAN in Appendix B.

4.2.1 Missing Modes Estimation on CelebA

We also employ a third party discriminator trained with injected noise as a metric for missing mode estimation. To implement this, we add noise in the input layer in the discriminator network. For each GAN model to be estimated, we independently train this noisy discriminator, as mode estimator, with the same architecture and hyper-parameters on the generated data and the training data. We then apply the mode estimator to the test data. The images which have high mode estimator outputs can be viewed as on the missing modes.

Table 3: Number of images on the missing modes on CelebA estimated by a third-party discriminator. The numbers in the brackets indicate the dimension of prior z. σ denotes the standard deviation of the added Gaussian noise applied at the input of the discriminator to regularize it. MDGAN achieves a very high reduction in the number of missing modes, in comparison to other methods .

σ	DCGAN (100)	DCGAN (200)	Reg-GAN (100)	Reg-GAN (200)	MDGAN (200)
3.5	5463	17089	754	3644	**74**
4.0	590	15832	42	391	**13**

The comparison result is shown in Table 3. Both our proposed Regularized-GAN and MDGAN outperform baseline DCGAN models on all settings. Especially, MDGAN suppresses other models, showing its superiority on modes preserving. We also find that, although sharing the same architecture, the DCGAN with 200-dimensional noise performs quite worse than that with 100-dimensional noise as input. On the contrary, our regularized GAN performs more consistently.

To get a better understanding of the models' performance, we want to figure out when and where these models miss the modes. Visualizing the test images associated with missed modes is instructive. In Figure 5, the left three images are missed by all models. It is rare to see in the training data the cap in the second image and the type of background in the third, which thus can be viewed as small modes under this situation. These three images should be considered as the hardest test data for GAN to learn. Nonetheless, our best model, MDGAN still capture certain small modes. The seven images on the right in Figure 5 are only missed by DCGAN. The sideface, paleface, black, and the berets are special attributes among these images, but our proposed MDGAN performs well on all of them.

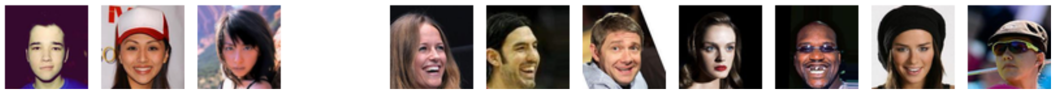

Figure 5: Test set images that are on missing mode. Left: Both MDGAN and DCGAN missing. Right: Only DCGAN missing.

4.2.2 Qualitative Evaluation of Generated Samples

After quantitative evaluation, we manually examine the generated samples by our regularized GAN to see whether the proposed regularizer has side-effects on sample quality. We compare our model with ALI (Dumoulin et al., 2016), VAEGAN (Larsen et al., 2015), and DCGAN (Radford et al., 2015) in terms of sample visual quality and mode diversity. Samples generated from these models are shown in Figure 6[2].

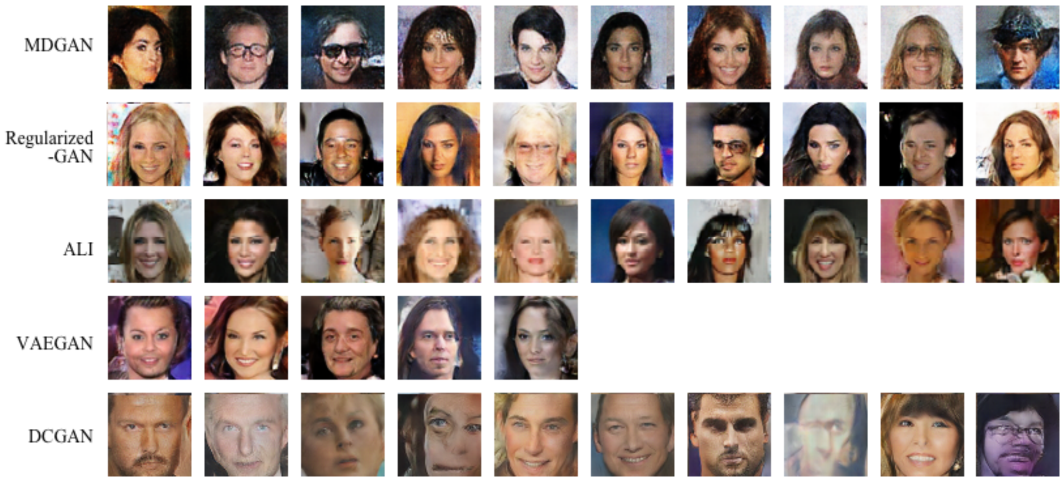

Figure 6: Samples generated from different generative models. For each compared model, we directly take ten decent samples reported in their corresponding papers and code repositories. Note how MDGAN samples are both globally more coherent and locally have sharp textures.

Both MDGAN and Regularized-GAN generate clear and natural-looking face images. Although ALI's samples are plausible, they are sightly deformed in comparison with those from MDGAN. The samples from VAEGAN and DCGAN seem globally less coherent and locally less sharp.

As to sample quality, it is worth noting that the samples from MDGAN enjoy fewer distortions. With all four other models, the majority of generated samples suffer from some sort of distortion. However, for the samples generated by MDGAN, the level of distortion is lower compared with the other four compared models. We attribute it to the help of the autoencoder as the regularizer to alter the generation manifolds. In this way, the generator is able to learn fine-grained details such as face edges. As a result, MDGAN is able to reduce distortions.

[2]For fair comparison, we also recommend readers to refer to the original papers Dumoulin et al. (2016); Larsen et al. (2015); Radford et al. (2015) for the reported samples of the compared. The ALI samples are from `https://github.com/IshmaelBelghazi/ALI/blob/master/paper/celeba_samples.png` and we reverted them to the original 64x64 size. The DCGAN samples are from `https://github.com/Newmu/dcgan_code/`

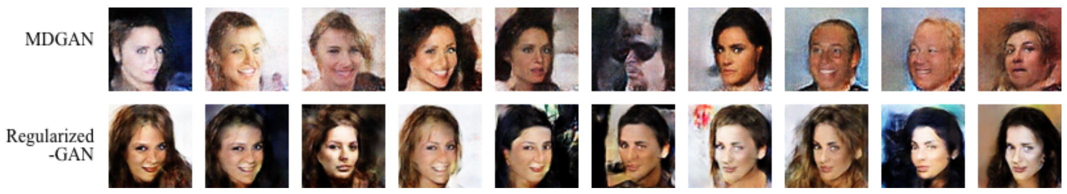

Figure 7: Sideface samples generated by Regularized-GAN and MDGAN.

In terms of missing modes problem, we instructed five individuals to conduct human evaluation on the generated samples. They achieve consensus that MDGAN wins in terms of mode diversities. Two people pointed out that MDGAN generates a larger amount of samples with side faces than other models. We select several of these side face samples in Figure 7. Clearly, our samples maintain acceptable visual fidelity meanwhile share diverse modes. Combined with the above quantitative results, it is convincing that our regularizers bring benefits for both training stability and mode variety without the loss of sample quality.

5 Conclusions

Although GANs achieve state-of-the-art results on a large variety of unsupervised learning tasks, training them is considered highly unstable, very difficult and sensitive to hyper-parameters, all the while, missing modes from the data distribution or even collapsing large amounts of probability mass on some modes. Successful GAN training usually requires large amounts of human and computing efforts to fine tune the hyper-parameters, in order to stabilize training and avoid collapsing. Researchers usually rely on their own experience and published tricks and hyper-parameters instead of systematic methods for training GANs.

We provide systematic ways to measure and avoid the missing modes problem and stabilize training with the proposed autoencoder-based regularizers. The key idea is that some geometric metrics can provide more stable gradients than trained discriminators, and when combined with the encoder, they can be used as regularizers for training. These regularizers can also penalize missing modes and encourage a fair distribution of probability mass on the generation manifold.

Acknowledgements

We thank Naiyan Wang, Jianbo Ye, Yuchen Ding, Saboya Yang for their GPU support. We also want to thank Huiling Zhen for helpful discussions, Junbo Zhao for providing the details of grid search experiments on his EBGAN model, as well as Anders Boesen Lindbo Larsen for kindly helping us on running VAEGAN experiments. We appreciate for the valuable suggestions and comments from the anonymous reviewers.

References

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pp. 1486–1494, 2015.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *arXiv preprint arXiv:1604.04382*, 2016.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

Masaki Saito and Eiichi Matsumoto. Temporal generative adversarial nets. *arXiv preprint arXiv:1611.06624*, 2016.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Neural Information Processing Systems (NIPS)*, 2016.

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

Yipin Zhou and Tamara L Berg. Learning temporal transformations from time-lapse videos. In *European Conference on Computer Vision*, pp. 262–277. Springer, 2016.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

A APPENDIX: PSEUDO CODE FOR MDGAN

In this Appendix, we give the detailed training procedure of an MDGAN example we discuss in Section 3.3.

```
Manifold Step:
1. Sample $\{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_m\}$ from data generating distribution $p_{data}(x)$.
2. Update discriminator $D_1$ using SGD with gradient ascent:
    $$\nabla_{\theta_d^1} \frac{1}{m} \sum_{i=1}^{m} [\log D_1(\mathbf{x}_i) + \log(1 - D_1(G(E(\mathbf{x}_i))))]$$
3. Update generator $G$ using SGD with gradient ascent:
    $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\lambda \log D_1(G(E(\mathbf{x}_i))) - ||\mathbf{x}_i - G(E(\mathbf{x}_i))||^2]$$
Diffusion Step:
4. Sample $\{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_m\}$ from data generating distribution $p_{data}(x)$.
5. Sample $\{\mathbf{z}_1, \mathbf{z}_2, \cdots \mathbf{z}_m\}$ from prior distribution $p_\sigma(z)$.
6. Update discriminator $D_2$ using SGD with gradient ascent:
    $$\nabla_{\theta_d^2} \frac{1}{m} \sum_{i=1}^{m} [\log D_2(G(E(\mathbf{x}_i))) + \log(1 - D_2(\mathbf{z}_i))]$$
7. Update generator $G$ using SGD with gradient ascent:
    $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log D_2(G(\mathbf{z}_i))]$$
```

Figure 8: The detailed training procedure of an MDGAN example.

B APPENDIX: ARCHITECTURE FOR EXPERIMENTS

We use similar architectures for Compositional MNIST and CelebA experiments. The architecture is based on that found in DCGAN Radford et al. (2015). Apart from the discriminator and generator which are the same as DCGAN, we add an encoder which is the ”inverse” of the generator, by reversing the order of layers and replacing the de-convolutional layers with convolutional layers.

One has to pay particular attention to batch normalization layers. In DCGAN, there are batch normalization layers both in the generator and the discriminator. However, two classes of data go through the batch normalization layers in the generator. One come from sampled noise z, the other one come from the encoder. In our implementation, we separate the batch statistics for these two classes of data in the generator, while keeping the parameters of BN layer to be shared. In this way, the batch statistics of these two kinds of batches cannot interfere with each other.

C APPENDIX: ADDITIONAL SYNTHESIZED EXPERIMENTS

To demonstrate the effectiveness of mode-regularized GANs proposed in this paper, we train a very simple GAN architecture on synthesized 2D dataset, following Metz et al. (2016).

The data is sampled from a mixture of 6 Gaussians, with standard derivation of 0.1. The means of the Gaussians are placed around a circle with radius 5. The generator network has two ReLU hidden layers with 128 neurons. It generates 2D output samples from 3D uniform noise from [0,1]. The discriminator consists of only one fully connected layer of ReLU neurons, mapping the 2D input to a real 1D number. Both networks are optimized with the Adam optimizer with the learning rate of 1e-4.

In the regularized version, we choose $\lambda_1 = \lambda_2 = 0.005$. The comparison between the generator distribution from standard GAN and our proposed regularized GAN are shown in Figure 9.

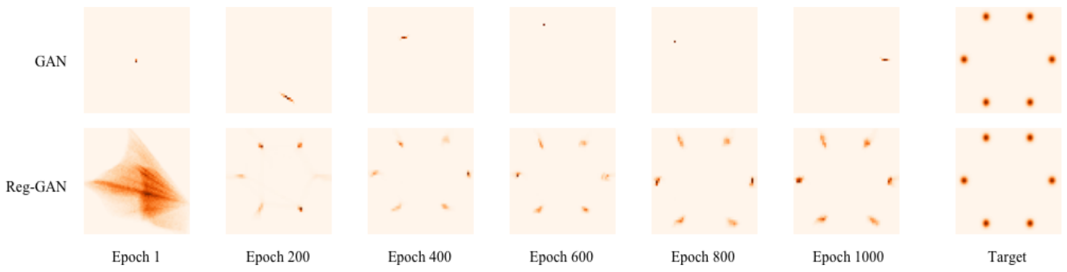

Figure 9: Comparison results on a toy 2D mixture of Gaussians dataset. The columns on the left shows heatmaps of the generator distributions as the number of training epochs increases, whereas the rightmost column presents the target, the original data distribution. The top row shows standard GAN result. The generator has a hard time oscillating among the modes of the data distribution, and is only able to "recover" a single data mode at once. In contrast, the bottom row shows results of our regularized GAN. Its generator quickly captures the underlying multiple modes and fits the target distribution.

D Appendix: Comparison with VAEGAN

In this appendix section, we demonstrate the effectiveness and uniqueness of mode-regularized GANs proposed in this paper as compared to Larsen et al. (2015) in terms of its theoretical difference, sample quality and number of missing modes.

With regard to the theoretical difference, the optimization of VAEGAN relies on the probabilistic variational bound, namely $p(x) \geq \mathbb{E}_{q(z|x)}[\log p(x|z)] - \text{KL}(q(z|x)||p(z))$. This variational bound together with a GAN loss is optimized with several assumptions imposed in VAEGAN:

1. In general, VAE is based on the assumption that the true posterior $p(z|x)$ can be well approximated by factorized Gaussian distribution q.
2. As to VAEGAN, It is also assumed that the maximum likelihood objectives does not conflict with GAN objective in terms of probabilistic framework.

The first assumption does not necessarily hold for GANs. We have found that in some trained models of DCGANs, the real posterior $p(z|x)$ is even not guaranteed to have only one mode, not to mention it is anything close to factorized Gaussian. We believe that this difference in probabilistic framework is an essential obstacle when one tries to use the objective of VAEGAN as a regularizer. However, in our algorithm, where we use a plain auto-encoder instead of VAE as the objective. Plain auto-encooders works better than VAE for our purposes because as long as the model $G(z)$ is able to generate training samples, there always exists a function $E^*(x)$ such that $G(E(x)) = x$. Our encoder can therefore be viewed as being trained to approximate this real encoder E^*. There are no conflicts between a good GAN generator and our regularization objective. Hence, our objectives can be used as regularizers for encoding the prior knowledge that good models should be able to generate the training samples. This is why our work is essentially different from VAEGAN. In our experiments, we also believe that this is the reason why VAEGAN generates worse samples than a carefully tuned regularized GANs.

In terms of sample quality and missing modes, we run the official code of VAEGAN [3] with their default setting. We train VAEGAN for 30 epochs [4] and our models for only 20 epochs. For fairness,

[3]`https://github.com/andersbll/autoencoding_beyond_pixels`

[4]Note that we also trained 20-epoch version of VAEGAN, however the samples seemed worse.

their model was run 3 times and the trained model with the best sample visual quality was taken for the comparison.

The generated samples are shown in Figure 10. The most obvious difference between our samples and VAEGAN's samples is the face distortion, which is consistent with our experimental results in Section 4.2.2. We conjecture that the distortions of VAEGAN's samples are due to the conflicts between the two objectives, as we present above. In other words, the way we introduce auto-encoders as regularizers for GAN models is different from VAEGAN's. The difference is that the second assumption mentioned above is not required in our approaches. In our framework, the auto-encoders helps alter the generation manifolds, leading to fewer distortions in fine-grained details in our generated samples.

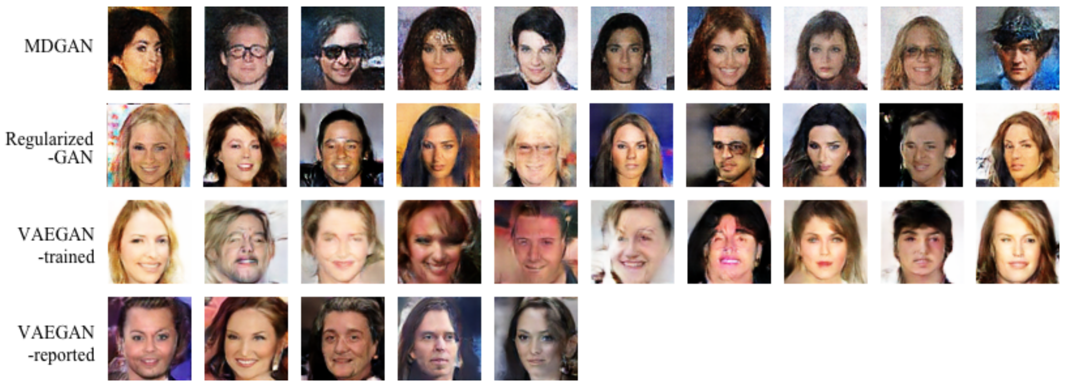

Figure 10: Samples generated by our models and VAEGAN. The third line are samples generated by our self-trained VAEGAN model, with default settings. The last line are generated samples reported in the original VAEGAN paper. We depict both of them here for a fair comparison.

In terms of the missing modes problem, we use the same method described in Section 4.2.1 for computing the number of images with missing modes. The results are shown below.

Table 4: Number of images on the missing modes on CelebA estimated by a third-party discriminator. The numbers in the brackets indicate the dimension of prior z. σ denotes the standard deviation of the added Gaussian noise applied at the input of the discriminator to regularize it. MDGAN achieves a very high reduction in the number of missing modes, in comparison to VAEGAN.

σ	VAEGAN (100)	Reg-GAN (100)	Reg-GAN (200)	MDGAN (200)
3.5	9720	754	3644	**74**
4.0	5862	42	391	**13**

We see that using our proposed regularizers results in a huge drop in the number of missing modes. We conjecture that the reason why VAEGAN performs very bad in our metric of missing modes is that the samples generated are of low quality, so the discriminator classified the samples as "not on mode". Namely, the data generated is too far away from many real data modes. Essentially if a model generates very bad samples, we can say that the model misses all or most modes.

To conduct more fair evaluation between VAEGAN and our methods, we also perform a blind human evaluation. Again we instructed five individuals to conduct this evaluation of sample variability. Without telling them which is generated by VAEGAN and which is generated by our methods, four people agrees that our method wins in terms of sample diversity. One people thinks the samples are equally diverse.

In conclusion, we demonstrate that our proposed mode-regularized GANs, i.e., Reg-GAN and MDGAN, are different from VAEGAN theoretically as discussed above. Such differences empirically result in better sample quality and mode preserving ability, which are our main contributions.