MULTI-GENERATOR GENERATIVE ADVERSARIAL NETS

Quan Hoang
University of Massachusetts-Amherst
Amherst, MA 01003, USA
qhoang@umass.edu

Tu Dinh Nguyen, Trung Le, Dinh Phung
PRaDA Centre, Deakin University
Geelong, Australia
{tu.nguyen,trung.l,dinh.phung}
@deakin.edu.au

ABSTRACT

We propose in this paper a novel approach to address the mode collapse problem in Generative Adversarial Nets (GANs) by training many generators. The training procedure is formulated as a minimax game among many generators, one classifier, and one discriminator. Generators produce data to fool the discriminator while staying within the decision boundary defined by the classifier as much as possible; the classifier estimates the probability that a sample comes from each of the generators; and the discriminator estimates the probability that a sample comes from the training data rather than from generators. We develop theoretical analysis to show that at an equilibrium of this system, the Jensen-Shannon divergence between the equally weighted mixture of all generators' distributions and the empirical data distribution is minimal while the Jensen-Shannon divergence among generators' distributions is maximal. Generators can be trained efficiently by utilizing parameter sharing, thus adding minimal cost to the basic GAN model. We conduct extensive experiments on synthetic and real-world large-scale datasets (CIFAR-10, STL-10 and ImageNet) to evaluate the effectiveness of our proposed method. Experimental results demonstrate the state-of-the-art performance of our approach in generating diverse and visually appealing samples over the latest GAN's variants.

1 INTRODUCTION

Generative Adversarial Nets (GAN) is a novel approach to train generative models that directly generate data samples while avoiding the cost of approximating the data generating distribution (Goodfellow et al., 2014). GANs have received great interests within the Deep Learning research community thanks to its ability to generate high quality images and capture semantic attributes of the training images (Radford et al., 2015). However, GANs suffer from the mode collapsing problem in which the generator concentrates on only a few modes instead of the whole data space (Goodfellow, 2016), (Metz et al., 2016). Therefore, GANs generated impressive samples when trained on data from specific domains such as bedroom scenes (Radford et al., 2015) but unsatisfactory samples when trained on more diverse data sets such as CIFAR-10 (Krizhevsky & Hinton, 2009) or ImageNet (Russakovsky et al., 2015).

Attempts to improve GANs have been mostly to modify the discriminator network or augment the objective function to provide the generator with better gradient signals. Improvements have been limited and, therefore, addressing the mode collapse problem remains a research frontier (Goodfellow, 2016).

In this paper, we propose a different approach to improve the generation diversity by training many generators that discover different modes of the data. At equilibrium, the equally weighted mixture of all generators' distributions approximates the unknown data generating distribution while the Jensen-Shannon divergence among them is maximal. The generators can be trained efficiently by utilizing parameter sharing and using small batch size. In our experiments, we used the Deep Convolutional Generative Adversarial Nets (DCGAN) architecture proposed in (Radford et al., 2015) to train our Multi-Generator Generative Adversarial Nets (MGGAN) model with 10 generators that

differ only in the input layers. The models trained on the CIFAR-10, STL-10 and ImageNet data sets successfully generated diverse, recognizable objects and achieved state-of-the-art Inception scores (Salimans et al., 2016). The model trained on the CIFAR-10 even outperformed the Semi-supervised Improved GAN model (Salimans et al., 2016). This confirms that training many generators is a promising approach to address the mode collapse problem. In short, our main contributions are:

- A robust framework to efficiently train many generator networks.
- A theoretical analysis that our objective functions are optimized towards minimizing the Jensen-Shannon divergence (JSD) between the equally weighted mixture of all generators' distributions and the real data distribution, while maximizing the JSD among generators.
- A comprehensive evaluation on the performance of our method on diverse data sets including CIFAR-10, STL-10, and ImageNet.

2 BACKGROUND

2.1 GENERATIVE ADVERSARIAL NETS (GAN)

GAN consists of a generator network G and a discriminator network D. G is trained to generate samples indistinguishable from real samples while D is trained to estimate the probability that a data sample came from the training data rather than the generator. GAN's original training procedure can be formulated as a minimax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim P_{data}(\mathbf{x})} \left[\log D\left(\mathbf{x}\right)\right] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \left[1 - \log\left(1 - D\left(G\left(\mathbf{z}\right)\right)\right)\right] \tag{1}$$

At equilibrium, the generator captures the real data distribution and $D\left(\mathbf{x}\right) = \frac{1}{2}$ almost everywhere (Goodfellow et al., 2014).

In practice, the convergence however is elusive. The mode collapse problem was identified as the most common cause of non-convergence in the GAN training procedure (Goodfellow, 2016). Many researchers believe the mode collapse problem is because learning the optimal G minimizes the Jensen-Shannon divergence, which is more similar to minimizing $D_{KL}(p_{model}\|p_{data})$, often referred to as the reverse KL divergence, than to minimizing the KL divergence $D_{KL}(p_{data}\|p_{model})$. A model trained with minimizing the reverse KL divergence would prefer to generate samples that are around only a few modes of the training distribution while ignoring other modes(Huszár, 2015)(Theis et al., 2015). The mode collapse problem was also shown to arise from a defect in the training procedure. When the generator and discriminator are trained simultaneously using gradient descent, the minimax game may become a maximin game (Goodfellow, 2016) .

2.2 RELATED WORK

Many GAN variants have been developed to solve the non-convergence and mode collapse problems. The main approach is to yield better gradient signal. Feature matching was proposed in (Salimans et al., 2016). The idea is to augment the generator's objective function with the L_2 distance between the average activations of generated data on an immediate layer of the discriminator and those of real data. Gradient signal from the L_2 loss term can guide the generators to generate samples with similar representations to the real data's.

Inspired by the idea of feature matching, Denoising Feature Matching (DFM) uses a denoising autoencoder (DAE)'s reconstruction error of generated data's activations in place of the L_2 loss (Warde-Farley & Bengio, 2016). The idea was to take advantage of the observation made in (Alain & Bengio, 2014) that for a DAE trained on data from a distribution $q(h)$, the reconstruction error $r(h) - h$ can estimate the gradient of the true log-density, $\frac{\partial \log q(h)}{\partial h}$. Therefore, gradient signal from a DAE's reconstruction error can guide the generator towards generating samples whose activation near the manifold of the real data's activations. The theoretical foundation of this approach is questionable due to the issue with spurious probability maxima discussed in (Alain & Bengio, 2014). When h is far away from the data manifold, $r\left(h\right) - h$ points to arbitrary directions because the noise used in training a DAE is often (as should be) so small that the DAE is trained with few samples that are far away from the data manifold. Therefore, the reconstruction error in DFM may send misleading signals. Even if DAE's reconstruction error behaves as desired, following its signals may cause

the model to collapse to local density maxima in the representation manifold, worsening the mode collapse problem. In practice, DFM was reported to achieve great performance on the CIFAR-10, STL-10 and ImageNet datasets (Warde-Farley & Bengio, 2016).

Another close idea was Energy-Based GAN (EBGAN), which uses an autoencoder as the discriminator (Zhao et al., 2016). In EBGAN, the generator's loss function is the reconstruction errors of generated samples as the generator's loss function. In theory, EBGAN is more robust than DFM because the autoencoder is trained with not only the real data but also with contrastive samples.

There were also some attempts to directly address the mode collapse problem. One idea was to use minibatch discrimination to allow the discriminator to detect samples that are a noticeably similar to other generated samples (Salimans et al., 2016). Minibatch discrimination works well and generate visually appealing samples but is computationally expensive and hard to train. Unrolled GANs attempts to address the defect of training the generator and discriminator simultaneously by including several optimization steps of the discriminator into the computational graph. It was shown that unrolling a small number of steps, such as 10 or fewer, could noticeably reduce the mode collapsing problem (Metz et al., 2016). This approach is however computationally expensive.

Unlike these attempts to improve generation diversity of a single generator, our method improves generation diversity through using many generators. Therefore, we expect that improvement in training a single generator could be easily integrated into our framework.

Attempts to train the discriminator in a semi-supervised manner achieved great improvement in quality of generated images (Salimans et al., 2016). Adding label information helps the discriminator learn representations that align with the features humans emphasize. This approach is related to our method in the use of a softmax classifier. However, our method does not require label information but let the discriminator and generators together classify the data space.

3 THE MGGAN MODEL

In MGGAN, K generators map noise vectors sampled from the same prior distribution $p(z)$ to the data space. Generator G_k implicitly defines a distribution density p_{G_k}. We wish to formulate a minimax game for MGGAN such that at equilibrium, the mixture of all distribution p_{G_k}s approximates the real data distribution p_{data} while the generators cover different modes of the data. The former goal can be achieved by following the derivation of the original GANs. The latter goal can be achieved by ensuring that optimizing the minimax game will maximize the Jensen-Shannon divergence (JSD) among the generators. We propose *three* alternatives to define the JSD among the generators: (i) *general* JSD, defined as the JSD for K distributions p_{G_1}, p_{G_2}, ..., p_{G_K} with equal weights; (ii) *one-vs-all* JSD, defined as the sum of the JSD between each p_{G_i} and the average $\frac{1}{K-1}\sum_{j\neq i} p_{G_j}$; (iii) and *pairwise* JSD, defined as the sum of the JSD between any two distributions p_{G_i} and p_{G_j}. All three versions of JSD, at their maximal values, will ensure that any two generators almost never produce the same data.

In the MGGAN model using the general JSD, the minimax game includes K generators G_ks, a classifier C, and a discriminator D. C estimates the probability that a sample came from each of the generators and D estimates the probability that a sample came from the training data rather than from Gs. Training many generators is made feasible by parameter sharing. Through experiment, we found that separating the first layer while sharing all other layers led to the best performance. This confirmed our intuition that a generator's low layers generates high-level features while high layers map abstract representation into real data. The mapping should be consistent across generators so it can be shared. Moreover, the classifier C and the discriminator D perform similar tasks so they can share parameters and differ only in the output layer. Figure 1 illustrates the general architecture of our model.

The minimax game is formulated as:

$$\min_{G,C}\max_{D} \mathcal{J}(G,C,D) = \mathbb{E}_{p_{data}}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{p_{model}}\left[\log\left(1 - D(\mathbf{x})\right)\right] - \frac{\beta}{K}\sum_{k=1}^{K}\mathbb{E}_{\mathbf{x}\sim p_{G_k}}\left[\log C_k(\mathbf{x})\right] \quad (2)$$

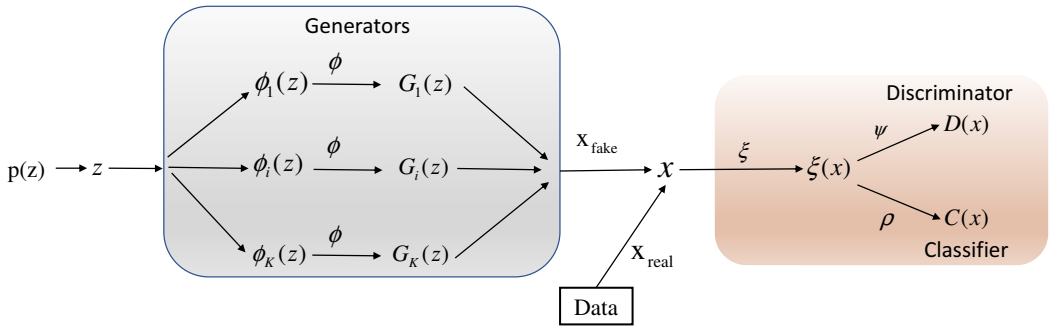


Figure 1: An illustration of DGGAN. K generators differ only in the input layer. The classifier and discriminator differ only in the output layer.

where $p_{model}(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} p_{G_k}(\mathbf{x})$, $\mathbb{E}_p f(\mathbf{x})$ is the expectation of $f(\mathbf{x})$ with respect to $p(\mathbf{x})$, and $C_k(\mathbf{x})$ is C's estimated probability that $\mathbf{x}$ is generated by G_k. We use the notation G to denote all generators. The interaction between G and D is similar to that of the original GAN. The interaction between G and C is interesting as both minimize the last term in 2. This term encourages each generator to generate data separable from those produced by other generators, and to stay within the decision boundary defined by C. β can be seen as the diversity hyper-parameter.

The MGGAN models using one-vs-all JSD and pairwise JSD are similar but differ in the classifier. The MGGAN model using one-vs-all JSD has K binary classifiers. Each classifier C_k estimates the probability that a sample is generated by G_k. The minimax game in this case is formulated as:

$$\min_{G,C}\max_{D}\mathcal{J}(G,C,D) = \mathbb{E}_{p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{p_{model}}[\log(1-D(\mathbf{x}))] - \beta\sum_{k=1}^{K}\left(\mathbb{E}_{\mathbf{x}\sim p_{G_k}}[\log C_k(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_{G_{-k}}}[\log(1-C_k(\mathbf{x}))]\right) \quad (3)$$

where C denotes all classifiers, and $p_{G_{-k}}(\mathbf{x}) = \frac{1}{K-1}\sum_{j\neq k} p_{G_j}(\mathbf{x})$.

The MGGAN model using pairwise JSD has $\frac{K\times(K-1)}{2}$ binary classifiers. Each classifier $C_{i,j}$ takes samples generated by generator G_i and G_j and estimates the probability that each sample is generated by G_i. The minimax game in this case is formulated as:

$$\min_{G,C}\max_{D}\mathcal{J}(G,C,D) = \mathbb{E}_{p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{p_{model}}[\log(1-D(\mathbf{x}))] - \beta\sum_{i<j}\left(\mathbb{E}_{\mathbf{x}\sim p_{G_i}}[\log C_{i,j}(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_{G_j}}[\log(1-C_{i,j}(\mathbf{x}))]\right) \quad (4)$$

where C denotes all classifiers.

4 Theoretical Analysis

The theoretical analysis in this section is done in a non-parametric setting, e.g. we assume that C, D and G have infinite capacity. We will show that at equilibrium, $p_{data} = p_{model}$ and the Jensen-Shannon divergence (JSD) among K generators is maximal, e.g. any two generators almost never produce the same data. The analyses for all three MGGAN variants are almost the same so we will present only the analysis for the MGGAN model using the general JSD.

Proposition 1. *For fixed generators G_1, G_2, ..., G_K, the optimal discriminators C^* and D^* are:*

$$C_k^*(\mathbf{x}) = \frac{p_{G_k}(\mathbf{x})}{\sum_{j=1}^{K} p_{G_j}(\mathbf{x})} \quad (5)$$

$$D^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_{model}(\mathbf{x})} \quad (6)$$

Proof. It can be easily seen that 5 is the general case of 6. The optimal D^* was proved in (Goodfellow, 2016). We will present a similar proof for the optimal C^*. Assuming that C^* can be optimized in the functional space, we can calculate the functional derivatives of $\mathcal{J}(G, C, D)$with respect to each $C_k(\mathbf{x})$ for $k \in (2, ..., K)$and set them equal to zero:

$$\begin{aligned}\frac{\delta \mathcal{J}(G,C,D)}{\delta C_k(\mathbf{x})} &= -\frac{1}{K}\frac{\delta}{\delta C_k(\mathbf{x})}\int\left(p_{G_1}(\mathbf{x})\log\left(1-\sum_{k=2}^{K}C_k(\mathbf{x})\right)+\sum_{k=2}^{K}p_{G_k}(\mathbf{x})\log C_k(\mathbf{x})\right)dx \\ &= -\frac{1}{K}\left(\frac{p_{G_k}(\mathbf{x})}{C_k(\mathbf{x})}-\frac{p_{G_1}(\mathbf{x})}{C_1(\mathbf{x})}\right) \end{aligned} \tag{7}$$

Setting $\frac{\delta \mathcal{J}(G,C,D)}{\delta C_k(\mathbf{x})}$ to 0 for $k \in (2, ..., K)$, we get:

$$\frac{p_{G_1}(\mathbf{x})}{C_1^*(\mathbf{x})} = \frac{p_{G_2}(\mathbf{x})}{C_2^*(\mathbf{x})} = ... = \frac{p_{G_K}(\mathbf{x})}{C_K^*(\mathbf{x})} \tag{8}$$

5 results from 8 due to the fact that $\sum_{k=1}^{K} C_k^*(\mathbf{x}) = 1$. □

Replacing the optimal C^* and D^* into 2, we can reformulate the objective function for the generators:

$$\begin{aligned}\mathcal{L}(G) &= \mathcal{J}(G, C^*, D^*) \\ &= \mathbb{E}_{p_{data}}\left[\log\frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})+p_{model}(\mathbf{x})}\right]+\mathbb{E}_{p_{model}}\left[\log\frac{p_{model}(\mathbf{x})}{p_{data}(\mathbf{x})+p_{model}(\mathbf{x})}\right] \\ &\quad -\frac{\beta}{K}\sum_{k=1}^{K}\mathbb{E}_{p_{G_k}}\left[\log\frac{p_{G_k}(\mathbf{x})}{\sum_{j=1}^{K}p_{G_j}(\mathbf{x})}\right]\end{aligned} \tag{9}$$

The sum of the first two terms in 9 was shown in (Goodfellow et al., 2014) to be $2 \cdot JSD(p_{data}\|p_{model}) - \log 4$. The second term of 9 is related to the Jensen-Shannon divergence for K distributions with equal weights:

$$\begin{aligned}-\frac{\beta}{K}\sum_{k=1}^{K}\mathbb{E}_{p_{G_k}}\left[\log\frac{p_{G_k}(\mathbf{x})}{\sum_{j=1}^{K}p_{G_j}(\mathbf{x})}\right] &= -\frac{\beta}{K}\sum_{k=1}^{K}\mathbb{E}_{p_{G_k}}\log p_{G_k}(\mathbf{x})+\frac{\beta}{K}\sum_{k=1}^{K}\mathbb{E}_{p_{G_k}}\log\sum_{j=1}^{K}p_{G_j}(\mathbf{x}) \\ &= \beta\sum_{k=1}^{K}\frac{1}{K}H(p_{G_k})-\beta H\left(\sum_{k=1}^{K}\frac{p_{G_k}}{K}\right)+\beta\log K \\ &= \beta\log K-\beta JSD_{1/K}(p_{G_1}, p_{G_2}, ..., p_{G_K})\end{aligned} \tag{10}$$

where $H(p)$ is the entropy of p. Thus, 9 can be rewritten as:

$$\begin{aligned}\mathcal{L}(G) = &-\log 4+2\cdot JSD(p_{data}\|p_{model}) \\ &+\beta\log K-\beta JSD_{1/K}(p_{G_1}, p_{G_2}, ..., p_{G_K})\end{aligned} \tag{11}$$

11 shows that $\mathcal{L}(G)$ can be minimized by minimizing $JSD(p_{data}\|p_{model})$ or maximizing $JSD_{1/K}(p_{G_1}, p_{G_2}, ..., p_{G_K})$ or both. The following lemma shows that $JSD_{1/K}(p_{G_1}, p_{G_2}, ..., p_{G_K})$ is maximal when generators almost never produce the same data.

Lemma 2. *$JSD_{1/K}(P_1, ..., P_K)$ reaches its upper bound of $\log K$ if and only if $\mathbb{E}_{P_i}[P_j(\mathbf{x})] = 0\ \forall i, j \in (1, ..., K)$.*

Proof. From 10, we have:

$$JSD_{1/K}(P_1, ..., P_K) = \log K+\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{P_i}\left[\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})}\right] \tag{12}$$

Consider each of the expectations in 12:

$$\begin{aligned}\mathbb{E}_{P_i}\left[\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})}\right] &= \mathbb{E}_{P_i}\left[\mathbb{I}_{\sum_{j\neq i}P_j(\mathbf{x})>0}\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})}\right] \\ &\quad+\mathbb{E}_{P_i}\left[\mathbb{I}_{\sum_{j\neq i}P_j(\mathbf{x})=0}\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})}\right] \\ &= \mathbb{E}_{P_i}\left[\mathbb{I}_{\sum_{j\neq i}P_j(\mathbf{x})>0}\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})}\right]\end{aligned}$$

where $\mathbb{I}(\cdot)$ be the indicator function. As $\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})} < 0$ where $\sum_{j\neq i}P_j(\mathbf{x}) > 0$, $\mathbb{E}_{P_i}\left[\log\frac{P_i(\mathbf{x})}{\sum_{j=1}^{K}P_j(\mathbf{x})}\right]$ reaches its upper bound of 0 if and only if $\mathbb{E}_{P_i}\left[\mathbb{I}_{\sum_{j\neq i}P_j(\mathbf{x})>0}\right] = \mathbb{E}_{P_i}\left[\sum_{j\neq i}P_j(\mathbf{x})\right] = \sum_{j\neq i}\mathbb{E}_{P_i}[P_j(\mathbf{x})] = 0$. The last equation occurs if and only if $\mathbb{E}_{P_i}[P_j(\mathbf{x})] = 0\ \forall j$. □

Theorem 3. *The global minimum of $\mathcal{L}(G)$ is achieved if and only if $p_{data} = p_{model}$ and the Jensen-Shannon divergence $JSD_{1/K}(p_{G_1}, p_{G_2}, ..., p_{G_K})$ is maximum. At that point, $\mathcal{L}_G(D_1^*, D_2^*) = -\log 4$.*

Proof. Theorem 3 results directly from 11 and Lemma 2. □

Theorem 3 shows that progressing towards equilibrium is equivalently to minimizing $JSD(p_{data}\|p_{model})$ while maximizing $JSD_{1/K}(P_1, ..., P_K)$. In practice, C, D, and G are parameterized by neural networks and are optimized in the parameter space rather than in the function space. As all of the K generators share the same objective function, we can efficiently update weights for all K generators using the same backpropagation passes. Therefore, MGGAN is scalable.

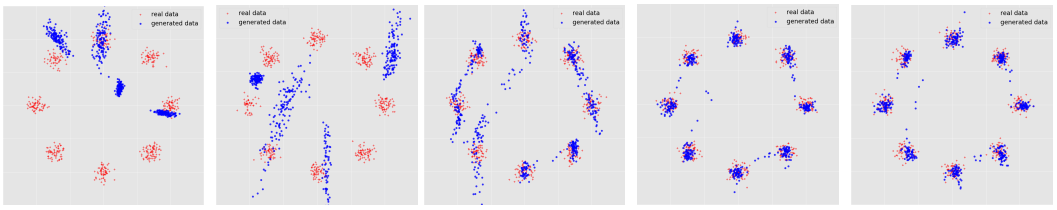
5 EXPERIMENTS

In this section, we will discuss the results of our experiments. We conducted experiments on both synthetic data and the diverse real-world datasets. We used the synthetic data to visualize the training behavior of the model while using the real-world datasets to assess the method's effectiveness in improving generation diversity and in generating visually appealing images. The experiments were implemented in TensorFlow. For all experiments, we used the MGGAN model with the general JSD as it is the most simple of the three variants. In addition, we used: (i): Adam optimizer (Kingma & Ba, 2014) with learning rate of 0.0002 and the first-order momentum of 0.5; (ii) mini-batch size of 64 samples for training the discriminators; (iii) ReLU activation function for the generator networks; (iv) Leaky ReLU with slope of 0.2 for the discriminator networks; weights initialized from an isotropic Gaussian: $\mathcal{N}(0, 0.02)$.

5.1 SYNTHETIC DATA

In the experiment on synthetic data, we reuse the experimental design proposed in (Metz et al., 2016) to investigate how well MGGAN can explore multiple modes in the data. The training data was sampled from a 2D mixture of 8 isotropic Gaussian distributions with a covariance matrix of $0.02I$ and means arranged in a circle of zero centroid and radius of 2.0. The low variance mixtures creates low density regions and separate the modes.

For the generator, we used a simple architecture of two fully connected hidden layers with 128 hidden units each. We train 4 generators that have separate weights connecting the input and the first hidden layer but share weights in the rest of the network. The discriminator has one hidden layer with 128 hidden units. The noise vector has 128 dimensions and is drawn from the standard Gaussian distribution.



(a) Epoch 1,000 (b) Epoch 5,000 (c) Epoch 10,000 (d) Epoch 15,000 (e) Epoch 20,000

Figure 2: The mode discovering progress of MGGAN. Generated data are in blue and data samples from the 8 Gaussians are in red..

Figure 2 shows the evolution of 512 samples generated by 4 generators through time. At epoch 1,000, the generators generated 4 separate clusters. At epoch 5,000, the generators started stretching out. At epoch 10,000, the each generator covers two modes but many points were scattered between the two modes of each pair. At epoch 15,000 and 20,000, generators produced few points between their two modes. Interestingly, we did not need 8 generators to discover all of the 8 modes.

5.2 Real-world Datasets

We trained MGGANs on three widely-adopted datasets: CIFAR-10 (Krizhevsky & Hinton, 2009), STL-10 (Coates et al., 2011) and Imagenet (Russakovsky et al., 2015). CIFAR-10 has 50,000 32×32 training images of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. STL-10 is a subset of ImageNet, containing about 100,000 unlabeled 96×96 images for unsupervised learning. STL-10 is much more diverse than CIFAR-10. Imagenet presents a much more larger and diverse space, consisting of over 1.2 million images from 1,000 classes. We train our model on the training set of the most widely used release, the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012). In order to ficiliate comparison with our baselines, we follow the procedure of (Krizhevsky et al., 2012) to we rescaled the STL-10 and ImageNet images down to 48×48 and 32×32, respectively.

To quantitatively evaluate the quality of images generated by MGGANs, we adopted the Inception score proposed in (Salimans et al., 2016), which computes $\exp\left(\mathbb{E}_{\mathbf{x}}\left[KL\left(p\left(y|\mathbf{x}\right)\|p\left(y\right)\right)\right]\right)$ where $p\left(y|\mathbf{x}\right)$ is the conditional label distribution for the image $\mathbf{x}$ estimated by the reference Inception convolutional neural network (Szegedy et al., 2015). Inception score rewards good and varied samples and was found to correlate very well with human judgment (Salimans et al., 2016) so we use this measure to quantitatively evaluate the generated images of our models. We use the code provided in (Salimans et al., 2016) to compute the Inception scores for 10 partitions of 50,000 randomly generated samples.

Our generator and discriminator architectures followed closely the DCGAN's design proposed in (Radford et al., 2015) except that we applied batch normalization (Ioffe & Szegedy, 2015) to all but the output layers in the networks. During training we observed two phenomenons. Occasionally in a mini-batch, the percentage of active neurons, which we define as ReLU units with positive activation for at least 10% of inputs in the batch, declined sharply from over 90% to around 60% and the generators simply output squares of a certain hue. Interestingly, after one or two weight updates, the generators output normal images and continued the training procedure as normal. The root cause of this problem is yet to be found. Another phenomenon was that the percentage of active neurons chronically declined. Therefore, the quality of generated images, after reaching the peak level, started declining. One possible cause was that the batch normalization center (offset) gradually shifted to the negative range, causing up to 45% of ReLU units of the generator networks to die. Our solution was to fix the offset at zero for all layers in the generator networks. The rationale is that for each feature map, the ReLU gates will open for 50% highest inputs in a mini-batch across all locations and generators, and close for the rest. Therefore, batch normalization can keep ReLU units alive even when most of their inputs are negative, and introduces a form of competition that encourages generators to "specialize" in different features. This measure significantly improves performance but does not totally solve the dying ReLUs problem. We found that late in the training, the input to generators' ReLU units became more and more right-skewed, causing the ReLU gates to

Model	CIFAR-10	STL-10	ImageNet
Real data	11.24±0.16	26.08±0.26	25.78±0.47
WGAN (Arjovsky et al., 2017)	3.82±0.06	–	–
MIX+WGAN (Arora et al., 2017)	4.04±0.07	–	–
Improved-GAN (Salimans et al., 2016)	4.36±0.04	–	–
ALI (Dumoulin et al., 2016)	5.34±0.05	–	–
BEGAN (Berthelot et al., 2017)	5.62	–	–
MAGAN (Wang et al., 2017)	5.67	–	–
DCGAN (Radford et al., 2015)	6.40±0.05	7.54	7.89
DFM (Warde-Farley & Bengio, 2016)	7.72±0.13	8.51±0.13	9.18±0.13
D2GAN (Nguyen et al., 2017)	7.15±0.07	7.98	8.25
MGGAN	**8.23**	**9.09**	**9.27**

Table 1: Comparison of Inception scores on different datasets.

open less and less often. The diversity hyper-parameter β was tuned to 0.1 for the CIFAR-10 dataset and 1.0 for the STL-10 48×48. For STL-10, we found that significantly increasing β help stabilize learning. For more details about the model architecture, we refer to the supplementary material.

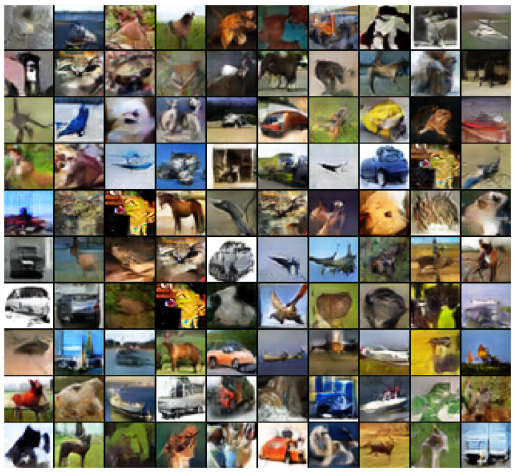
Table 1 compares the Inception scores on CIFAR10 32×32, STL10 48×48, and ImageNet 32×32 of our models and baselines collected from recent work in literature. Note that DCGAN's and D2GAN's Inception score on STL-10 were available only for the models trained on 32x32 images resolution. Overall, MGGAN outperformed other models by a wide margin on CIFAR-10 and STL-10. MGGAN achieved an Inception score of 8.23, outperforming even Semi-supervised Improved GAN's an Inception score of 8.09 as reported in (Salimans et al., 2016). Our proposed method also achieved the highest Inception score on ImageNet, however in this case our MGGAN outperformed DFM by only a small margin. It was tempted to combine DFM with MGGAN for further improvement but we did not successfully train a DFM model with the scores reported in (Warde-Farley & Bengio, 2016).

Finally we present samples randomly generated by our proposed model trained on the 3 datasets. Figure 3 shows 32×32 images generated from our MGGAN models trained on CIFAR-10 and STL-10. We can recognize many objects such as airplane, cars, truck, boat, deer, horse or dog. Figures 4 and 5 shows STL-10 48×48 and 64×64 images, respectively, generated by our MGGAN models. The images are diverse with objects such as bus, helicopter or parachute, and some animals look nice. In all there cases, there are still some broken images that we believe to result from the problems with ReLUs units. We hypothesize that solving this problem may further improve the quality of generated images.

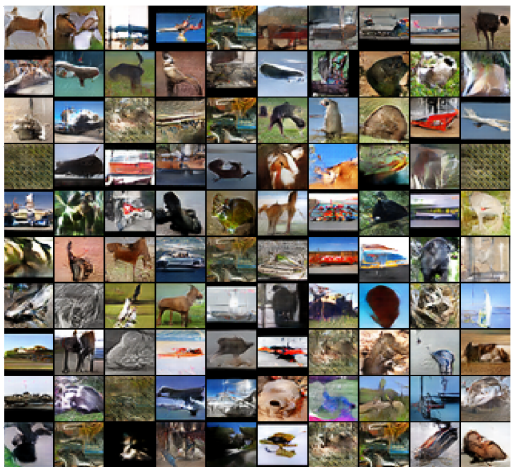
6 Conclusion

To summarize, we have presented a novel approach to alleviate the mode collapse problem for GANs. Our idea is to train many generators to explore different modes of the data. To achieve this goal, we proposed a minimax game in which the optimization is equivalent to minimize the reverse Kullback-Leibler divergence between p_{model} and p_{data} while maximizing Jensen-Shannon divergence (JSD) among generators. We proposed three variants using three different definition of the JSD among generators, including the general JSD, one-vs-all JSD, and pairwise JSD. For all three variants, the generators, at equilibrium, generate different images but together recover the data generating distribution. The generators can be efficiently trained by utilizing parameter sharing. We have implemented experiments to evaluate the effectiveness of our method in generating diverse natural images. The trained models achieved the state-of-the-art inception scores on the CIFAR-10 and STL-10, and generated some visually appealing images.

There are, however, some weaknesses. Our approach introduced a diversity hyper-parameter, which might be computationally expensive to tune. Training many generators also add more parameters to the networks. Most importantly, we noticed problems with ReLUs units during training. We have not fully identified the root cause. Fixing batch normalization offset at zero helped alleviate the problem but not totally. We hypothesize that solving this problem might significantly boost the performance of MGGAN.



(a) CIFAR-10 generated samples.



(b) STL-10 generated samples.

Figure 3: 32×32 images generated by our MGGAN models.

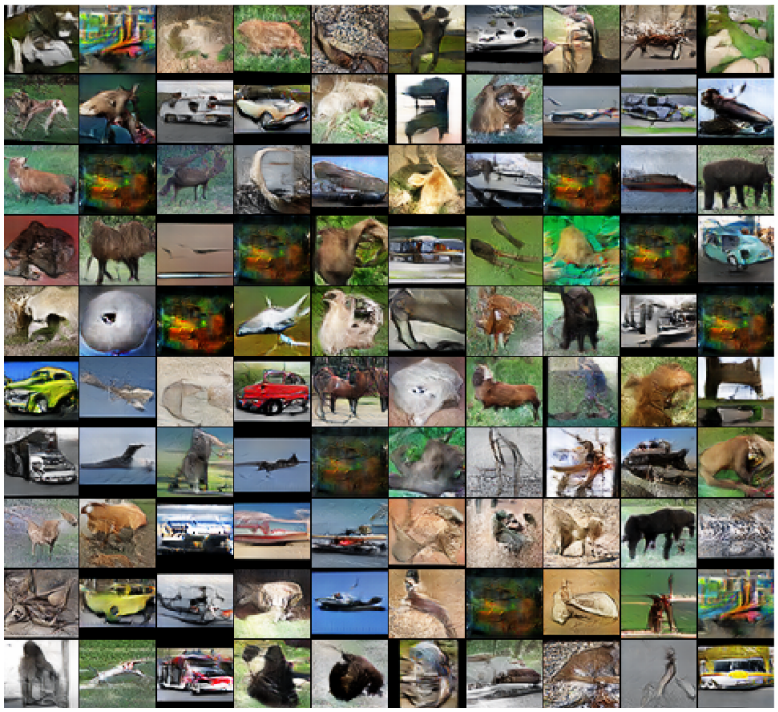


Figure 4: STL-10 48×48 images generated by an MGGAN model. .

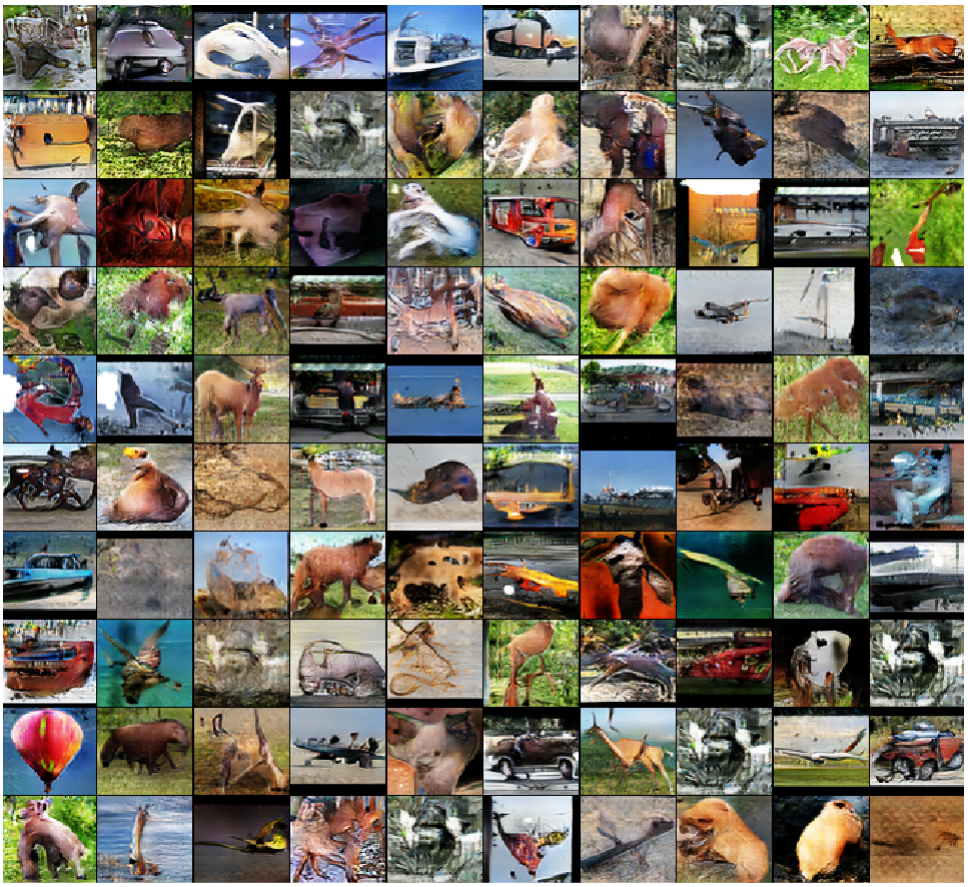


Figure 5: STL-10 64×64 images generated by an MGGAN model..

Our approach was not to directly directly solve the mode collapse problem for a single generator but forcing the divergence among generators might indirectly help the shared generator networks learn better. Finally, we hypothesize that any advancement in training a single generator can be easily integrated in our framework.

REFERENCES

Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014. 2.2

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. **??**

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017. **??**

David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. **??**

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, 2011. 5.2

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. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 1, 2.1, 4

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. 1, 2.1, 4

Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015. 2.1

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015. 5.2

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 1, 5.2

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012. 5.2

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016. 1, 2.2, 5.1

Tu Dinh Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pp. accepted, 2017. **??**

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1, 5.2, **??**

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 5.2

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016. 1, 2.2, 5.2, **??**

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015. 5.2

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015. 2.1

Ruohan Wang, Antoine Cully, Hyung Jin Chang, and Yiannis Demiris. Magan: Margin adaptation for generative adversarial networks. *arXiv preprint arXiv:1704.03817*, 2017. **??**

David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. 2016. 2.2, **??**, 5.2

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. 2.2