Multi-Agent Diverse Generative Adversarial Networks

Arnab Ghosh*
University of Oxford
arnab.ghosh@eng.ox.ac.uk

Viveka Kulharia*
University of Oxford
viveka@robots.ox.ac.uk

Vinay Namboodiri
IIT Kanpur
vinaypn@iitk.ac.in

Philip H.S. Torr
University of Oxford
philip.torr@eng.ox.ac.uk

Puneet K. Dokania
University of Oxford
puneet@robots.ox.ac.uk

Abstract

We propose an intuitive generalization to the Generative Adversarial Networks (GANs) and its conditional variants *to address the well known* mode collapse *problem. Firstly, we propose a multi-agent GAN architecture incorporating multiple generators and one discriminator. Secondly, to enforce different generators to capture diverse high probability modes, we modify discriminator's objective function where along with finding the real and fake samples, the discriminator has to identify the generator that generated the fake sample. Intuitively, to succeed in this task, the discriminator must learn to push different generators towards different identifiable modes. Our framework (MAD-GAN) is generalizable in the sense that it can be easily combined with other existing variants of GANs to produce diverse samples. We perform extensive experiments on synthetic and real datasets and compare MAD-GAN with different variants of GAN. We show high quality diverse sample generations for the challenging tasks such as image-to-image translation (known to learn delta distribution) and face generation. In addition, we show that MAD-GAN is able to disentangle different modalities even when trained using highly challenging multi-view dataset (mixture of forests, icebergs, bedrooms* etc.*). In the end, we also show its efficacy for the unsupervised feature representation task. In the appendix we introduce a similarity based competing objective which encourages the different generators to generate varied samples judged by a user defined similarity metric. We show extensive evaluations on a 1-D setting of mixture of gaussians for non parametric density estimation. The theoretical proofs back the efficacy of the framework and explains why various generators are pushed towards distinct clusters of modes.*

*Joint first author

Figure 1: Multi-view data generation using MAD-GAN (a highly challenging task). Notice that each of the three generators (each row shows generation from different generators) generate images of different modalities when trained on a highly challenging multi-view data (mixture of forests, icebergs, bedrooms *etc*.). Each column represents generations for a given random noise input z.

1. Introduction

Generating new data points given an unlabeled dataset is a core challenging problem of machine learning. In this paper we address this task using generative models. The underlying idea behind such models is to generate high-dimensional data such as images and texts using low/high-dimensional interpretable latent space. Though these models are highly useful in various applications, it is computationally challenging to efficiently train them as it normally requires intractable integration in a very high-dimensional space. Recently, there have been remarkable progress in this field with the development of generative models that do not explicitly require integration and can be trained using back-propagation algorithm. Two such famous examples are Generative Adversarial Networks [13] and Variational Autoencoders [17].

In this work we focus on GANs as they are known to produce sharp and plausible images. Briefly, the objective function of GANs employ a generator and a discriminator where both are involved in a minimax game. The task of the discriminator is to learn the difference between 'real' (from true data distribution p_d) and 'fake' samples

(from generator distribution p_g). However, the task of the generator is to maximize the mistakes of the discriminator. At convergence, the generator learns to produce real looking images. A few successful applications of GANs are video generation [29], image inpainting [24], image manipulation [32], 3D object generation [30], interactive image generation using few brush strokes [32], image super-resolution [19], diagrammatic abstract reasoning [11] and conditional GANs [22, 26].

Despite the remarkable success of GANs, one of its major drawback is the problem of 'mode collapse' [2, 7, 8, 21, 27]. Even though, theoretically, at convergence the generator should be able to learn the true data distribution, practically this is not observed because of the difficulties involved in optimizing GANs which makes it very hard to reach the true equilibrium. Broadly speaking, there are two directions in which this issue has been addressed: (1) improving the learning aspect of GANs, similar to [2, 21, 27]; and (2) enforcing GANs to capture diverse modes, similar to [7, 8, 20]. In this work we focus on the latter.

Inspired by the multi-agent algorithm [1] and coupled GAN [20], we propose to use multiple generators with one discriminator, and allow generators to share information. We call this the Multi-Agent GAN architecture as shown in Figure 2. In more detail, similar to the standard GAN, the objective of each generator here is to maximize the mistakes of the *common* discriminator. To allow different generators to share information with each other, we share few initial layer parameters among all the generators. Another reason behind sharing these parameters is the fact that initial layers capture high-frequency structures which is almost the same for a particular type of dataset (for example, faces), therefore, sharing them reduces redundant computations. Naively using this simple approach may lead to the *trivial solution* where all the generators learn to generate *similar* samples. To resolve this issue and generate different visually plausible samples capturing diverse high probability modes, we propose a very simple and intuitive solution. *We propose to modify the objective function of the discriminator, in which, along with finding the real and the fake samples, the discriminator also has to correctly predict the generator that generated the given fake sample.* Intuitively, in order to succeed in this task, the discriminator must learn to push generations corresponding to different generators towards different identifiable modes. Experimentally, we show that this approach is highly effective and outperforms existing approaches in terms of diverse generations. We also provide theoretical analysis of this approach and show that the proposed modification in the discriminator's objective function allows generators to learn together as a mixture model where each generator represents a mixture component. We show that at convergence, the global optimum value of $-(k+1)\log(k+1)+k\log k$ is achieved, where k is the number of generators. Combining the Multi-Agent GAN architecture with the diversity enforcing term allows us to generate diverse plausible samples, thus the name Multi-Agent Diverse GAN (MAD-GAN).

We analyze MAD-GAN through extensive experiments. First of all, we use synthetic datasets (mixture of Gaussians) to show that MAD-GAN is able to capture diverse clustered modes. We then move towards a more complicated Stacked/Compositional MNIST dataset with 1000 hand engineered modes and compare MAD-GAN with several variants of GANs. Using KL divergence and number of modes recovered as a criterion, we show that our approach outperforms all other GAN variants we compare with, and is able to generate high quality samples while capturing very high number of modes. To evaluate our approach on real world and much more challenging datasets, we show high quality diverse sample generations for the challenging tasks of *image-to-image translation* [14] (conditional GAN), *multi-view generations* and *face generation* [8, 25]. In the end, using the SVHN dataset [23], we show that our framework is capable of learning a better feature representation in an unsupervised setting compared to DCGAN [25].

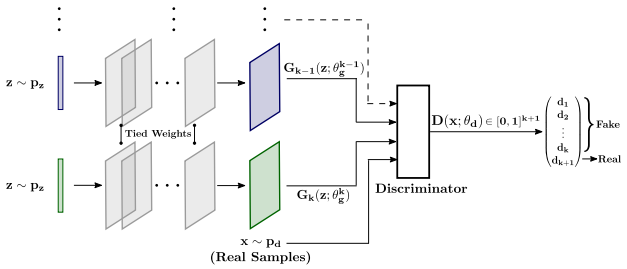


Figure 2: Multi-Agent Diverse GAN (MAD-GAN). The discriminator outputs $k+1$ softmax scores signifying the probability of its input sample being from either one of the k generators or real distribution. Thus it follows generator identification based objective. All but the last layer parameters of the generators are shared.

2. Related Work

The recent work called InfoGAN [8] proposed an information-theoretic extension to GANs in order to address the problem of mode collapse. Briefly, InfoGAN disentangles the latent representation by assuming a factored representation of the latent variables. In order to enforce the generator to learn factor specific generations, InfoGAN maximizes the mutual information between the factored latents and the generator distribution. Che *et al*. [7] proposed a mode regularized GAN (ModeGAN) which uses an encoder-decoder paradigm. The basic idea behind ModeGAN is that if a sample from the true data distribution p_d belongs to a particular mode, then the sample generated by

the generator (fake sample) when the true sample is passed through the encoder-decoder is likely to belong to the same mode. ModeGAN assumes that there exists enough true samples from a mode for the generator to be able to capture it. Another work by Metz *et al*. [21] proposed a surrogate objective for the update of the generator with respect to the unrolled optimization of the discriminator (UnrolledGAN) to address the issue of convergence of the training process of GANs. This improves the training process of the generator which in turn allow the generators to explore wide coverage of the true data distribution.

Liu *et al*. [20] presented Coupled GAN, a method for training two generators with shared parameters to learn the joint distribution of the data. The shared parameters guide both the generators towards similar subspaces but since they are trained independently on two domains, they promote diverse generations. Durugkar *et al*. [10] proposed a model with multiple discriminators whereby an ensemble of multiple discriminators have been shown to stabilize the training of the generator by guiding it to produce better samples.

W-GAN [3] is a recent technique which employs integral probability metrics based on the earth mover distance rather than the JS-divergences that the original GAN uses. BEGAN [5] builds upon W-GAN using an autoencoder based equilibrium enforcing technique alongside the Wasserstein distance. DCGAN [25] was an iconic technique which used fully convolutional generator and discriminator for the first time and was able to generate compelling generations along with the introduction of batch normalization thus stabilizing the training procedure. GoGAN [16] introduced a training procedure for the training of the discriminator using a maximum margin formulation alongside the earth mover distance based on the Wasserstein-1 metric. [4] introduced a technique and theoretical formulation stating the importance of multiple generators and discriminators in order to completely model the data distribution.

In terms of employing multiple generators, our work is closest to [4, 20, 12]. However, while using multiple generators, our method explicitly enforces them to capture diverse modes.

3. Preliminaries

Here we present a brief review of GANs [13]. Given a set of unlabelled samples $\mathcal{D} = (x_i)_{i=1}^n$ from the true data distribution p_d, the GAN learning problem is to obtain the optimal parameters θ_g of a generator $G(z;\theta_g)$ that can sample from an approximate data distribution p_g, where $z \sim p_z$ is the prior input noise (*e.g*. samples from a normal distribution). In order to learn the optimal θ_g, the GAN objective (Eq. (1)) employs a discriminator $D(x;\theta_d)$ that learns to differentiate between a 'real' (from p_d) and a 'fake' (from p_g) sample x. The overall GAN objective function is as follows:

$$\min_{\theta_g}\max_{\theta_d} V(\theta_d,\theta_g) := \mathbb{E}_{x\sim p_d}\log D(x;\theta_d) + \mathbb{E}_{z\sim p_z}\log\big(1-D(G(z;\theta_g);\theta_d)\big) \quad (1)$$

The above objective is optimized in a block-wise manner where θ_d and θ_g are optimized one at a time while fixing the other. Intuitively, for a given sample x (either from p_d or p_g) and the parameter θ_d, the function $D(x;\theta_d) \in [0,1]$ produces a score that represents the probability of x belonging to the true data distribution p_d (or probability of it being real). Hence, the discriminator's objective is to learn parameters θ_d that maximizes this score for the true samples (from p_d) while minimizing it for the fake ones $\tilde{x} = D(z;\theta_g)$ (from p_g). On the other hand, the generator's objective is to minimize $\mathbb{E}_{z\sim p_z}\log\big(1-D(G(z;\theta_g);\theta_d)\big)$, equivalently maximize $\mathbb{E}_{z\sim p_z}\log D(G(z;\theta_g);\theta_d)$. Thus, the generator learns to maximize the scores for the fake samples (from p_g), which is exactly the opposite to what discriminator is trying to achieve. In this manner, the generator and the discriminator are involved in a minimax game where the task of the generator is to maximize the mistakes of the discriminator. Theoretically, at equilibrium, the generator learns to generate real samples, which means $p_g = p_d$.

4. Multi-Agent Diverse GAN

In the GAN objective, generator's task is much harder than that of the discriminator as it has to produce real looking images to maximize the mistakes of the discriminator. This, along with the minimax nature of the objective raise several challenges with GANs such as [2, 7, 8, 21, 27]: (1) mode collapse; (2) difficult optimization; and (3) trivial solution. In this work we propose a new framework to address the first challenge of 'mode collapse' by increasing the capacity of the generator while we use the well known optimization tricks to partially avoid other challenges [2].

Briefly, we propose a *Multi-Agent GAN architecture* that employs multiple generators and one discriminator in order to generate different samples while capturing high probability regions of the true data distribution. In order to direct different generators towards diverse modes we propose to modify the objective function of the discriminator where along with finding fake samples, the discriminator has to identify the generator that produced the given fake sample. Thus it follows generator identification based objective. Theoretically, we show that this objective allows generators to act as a mixture model with each generator capturing one component. We also provide conditions for global optimality. Our framework is general in the sense that it can be used with different variants of GAN.

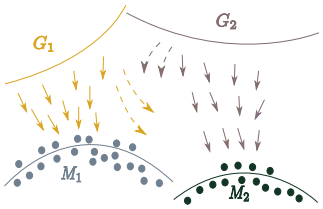


Figure 3: Visualization of how different generators are pushed towards different modes. Note that M_1 and M_2 could be a cluster of modes where each cluster themselves contain different modes. The arrows abstractly represent generator specific gradients for the purpose of building intuition.

4.1. Multi-Agent GAN Architecture

Here we describe our proposed architecture (Figure 2). It involves k generators and one discriminator. In the case of single-view data (*e.g.* faces or birds), we allow all the generators to share information by tying most of the initial layer parameters. This is essential to avoid redundant computations as initial layers of a generator captures high-frequency structures which is almost the same for a particular type of dataset. This also allows different generators to converge faster. However, in the case of *multi-view* data (*e.g.* mixture of different views such as forests, icebergs *etc.*), it is necessary to avoid sharing these parameters to allow each generator to capture view specific structures.

More specifically, given $z \sim p_z$ for the i-th generator, similar to the standard GAN, the first step involves generating a sample (for example, an image) $\tilde{x}_i$. Since each generator receives the same latent input sampled from the same distribution, naively using this simple approach may lead to the *trivial solution* where all the generators learn to generate similar samples. In what follows we propose an intuitive solution to avoid this issue to capture diverse modes.

4.2. Enforcing Diverse Modes

Inspired by the discriminator formulation for the semi-supervised learning [27], we use a generator identification based objective function that, along with minimizing the score $D(\tilde{x};\theta_d)$, requires the discriminator to identify the generator that generated the given fake sample $\tilde{x}$. In order to do so, as opposed to the standard GAN objective function where the discriminator outputs a scalar value, we modify it to output $k+1$ soft-max scores. In more detail, given the set of k generators, the discriminator produces a soft-max probability distribution over $k+1$ classes. The score at $(k+1)$-th index $(D_{k+1}(.))$ represents the probability that the sample belongs to the true data distribution and the score at $j \in \{1,\ldots,k\}$-th index represents the probability of it being generated by the j-th generator. Under this setting, while learning θ_d, we optimize the cross-entropy between the soft-max output of the discriminator and the Dirac delta distribution $\delta \in \{0,1\}^{k+1}$, where for $j \in \{1,\ldots,k\}$, $\delta(j) = 1$ if the sample belongs to the j-th generator, otherwise $\delta(k+1) = 1$. Therefore, the objective function (1) for the optimization of θ_d while keeping θ_g constant becomes:

$$\max_{\theta_d} \mathbb{E}_{x\sim p}\, H(\delta, D(x;\theta_d)) \tag{2}$$

where, $Supp(p) = \cup_{i=1}^{k} Supp(p_{g_i}) \cup Supp(p_d)$ and $H(.,.)$ is the negative of the cross entropy function. Intuitively, in order to correctly identify the generator that produced a given fake sample, the discriminator must learn to push different generators towards different identifiable modes. However, the objective of each generator remains the same as in the standard GAN. Thus, for the i-th generator, the objective is to minimize the following:

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x;\theta_d) + \mathbb{E}_{z\sim p_z} \log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_d))$$

To update the parameters, the gradient for each generator is simply computed as $\nabla_{\theta_g^i} \log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_d)))$. Notice that all the generators in this case can be updated in parallel. For the discriminator, given $x \sim p$ (can be real or fake) and corresponding δ, the gradient is $\nabla_{\theta_d} \log D_j(x;\theta_d)$, where $D_j(x;\theta_d)$ is the j-th index of $D(x;\theta_d)$ for which $\delta(j) = 1$. Therefore, using this approach requires *very minor modifications to the standard GAN optimization algorithm* and can be easily used with different variants of GAN.

Theorem 1 shows that the above objective function actually allows generators to form a mixture model where each generator represents a mixture component and the global optimum of $-(k+1)\log(k+1) + k\log k$ is achieved when $p_d = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$. Notice that, at $k = 1$, which is the case with one generator, we obtain exactly the same Jensen-Shannon divergence based objective function as shown in [13] with the same optimal value of $-\log 4$.

Theorem 1. *Given the optimal discriminator, the objective function for training the generators boils down to minimizing*

$$KL\Big(p_d(x)||p_{avg}(x)\Big) + kKL\Big(\frac{1}{k}\sum_{i=1}^{k} p_{g_i}(x)||p_{avg}(x)\Big) - (k+1)\log(k+1) + k\log k \tag{3}$$

where, $p_{avg}(x) = \frac{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)}{k+1}$. *The above objective function obtains its global minimum if* $p_d = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$ *with the objective value of* $-(k+1)\log(k+1) + k\log k$.

Proof. The objective function of the set of generators is to minimize the following:

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x) + \sum_{i=1}^{k} \mathbb{E}_{x\sim p_{g_i}} \log(1 - D_{k+1}(x))$$

Using Corollary 1, we substitute the optimal discriminator in the above equation and obtain:

$$\mathbb{E}_{x\sim p_d}\log\left[\frac{p_d(x)}{p_d(x)+\sum_{i=1}^{k}p_{g_i}(x)}\right]+$$
$$\sum_{i=1}^{k}\mathbb{E}_{x\sim p_{g_i}}\log\left[\frac{\sum_{i=1}^{k}p_{g_i}(x)}{p_d(x)+\sum_{i=1}^{k}p_{g_i}(x)}\right]$$
$$:=\mathbb{E}_{x\sim p_d}\log\left[\frac{p_d(x)}{p_{avg}(x)}\right]+k\mathbb{E}_{x\sim p_g}\log\left[\frac{p_g(x)}{p_{avg}(x)}\right]$$
$$-(k+1)\log(k+1)+k\log k \tag{4}$$

where, $p_g=\frac{\sum_{i=1}^{k}p_{g_i}}{k}$ and $p_{avg}(x)=\frac{p_d(x)+\sum_{i=1}^{k}p_{g_i}(x)}{k+1}$. Note that Equation (4) is exactly the same as Equation (3). When $p_d=\frac{\sum_{i=1}^{k}p_{g_i}}{k}$, both the KL terms become zero and the global minimum is achieved. □

Corollary 1. *For fixed generators, the optimal distribution learned by the discriminator D has following form:*

$$D_{k+1}(x)=\frac{p_d(x)}{p_d(x)+\sum_{i=1}^{k}p_{g_i}(x)}, \tag{5}$$
$$D_i(x)=\frac{p_{g_i}(x)}{p_d(x)+\sum_{i=1}^{k}p_{g_i}(x)}, \forall i\in\{1,\cdots,k\}.$$

where, $D_i(x)$ represents the i-th index of $D(x;\theta_d)$, p_d the true data distribution, and p_{g_i} the distribution learned by the i-th generator.

Proof. For fixed generators, the objective function of the discriminator is to maximize

$$\mathbb{E}_{x\sim p_d}\log D_{k+1}(x)+\sum_{i=1}^{k}\mathbb{E}_{x_i\sim p_{g_i}}\log D_i(x_i) \tag{6}$$

where, $\sum_{i=1}^{k+1}D_i(x)=1$ and $D_i(x)\in[0,1],\forall i$. The above equation can be written as:

$$\int_x p_d(x)\log D_{k+1}(x)dx+\sum_{i=1}^{k}\int_x p_{g_i}(x)\log D_i(x)dx$$
$$:=\int_{x\in p}\sum_{i=1}^{k+1}p_i(x)\log D_i(x)dx \tag{7}$$

where, $p_{k+1}(x):=p_d(x)$, $p_i(x):=p_{g_i}(x),\forall i\in\{1,\cdots,k\}$, and $Supp(p)=\bigcup_{i=1}^{k}Supp(p_{g_i})\bigcup Supp(p_d)$, Therefore, for a given x, the optimum of objective function defined in Equation 7 with constraints defined above can be obtained using Proposition 1. □

Proposition 1. *Given $\mathbf{y}=(y_1,\cdots,y_n)$, $y_i\geq 0$, and $a_i\in\mathbb{R}$, the optimal solution for the objective function defined below is achieved at $y_i^*=\frac{a_i}{\sum_{i=1}^{n}a_i},\forall i$*

$$\max_{\mathbf{y}}\sum_{i=1}^{n}a_i\log y_i,\quad s.t.\ \sum_{i}^{n}y_i=1 \tag{8}$$

Proof. The Lagrangian of the above problem is:

$$L(\mathbf{y},\lambda)=\sum_{i=1}^{n}a_i\log y_i+\lambda(\sum_{i=1}^{n}y_i-1) \tag{9}$$

Differentiating w.r.t y_i and λ, and equating to zero, we obtain

$$\frac{a_i}{y_i}+\lambda=0\ ,\ \sum_{i=1}^{n}y_i-1=0$$

Solving the above two equations, we obtain $y_i^*=\frac{a_i}{\sum_{i=1}^{n}a_i}$. □

Discussion The generators' objective is minimized when the discriminator is optimal and $p_d=\frac{1}{k}\sum_{i=1}^{k}p_{g_i}$ holds true but if we analyze it carefully we can see that there are infinitely many configurations for the generators to assign probabilities to different x such that $p_d=\frac{1}{k}\sum_{i=1}^{k}p_{g_i}$ still holds true. For different configurations the Cross Entropy Error of the discriminator is different and the minimum cross entropy occurs when for each point x $\exists i$ such that $p_{g_i}(x)=kp_d(x)$ and $\forall j\neq i; p_{g_j}(x)=0$. It means that given enough capacity of the generators and the discriminator and a good enough optimizer the modes are distributed among the generators and the generators are tasked with mastering distinct modes of the data distribution.

One obvious question that could arise is that *is it possible that all the generators learn to capture the same mode?*. The short answer is, theoretically *yes* and in practice *no*. Let us begin with the discussion to understand this. Theoretically, according to Theorem 1, if $p_{g_i}=p_d$, for all i, then also the minimum objective value can be achieved. This implies, in worst case, MAD-GAN would perform same as the standard GAN. However, as discussed below, this is possible in following *highly unlikely* situations:

- all generators *always generate exactly similar* samples so that the discriminator is not able to differentiate them. In this case, the discriminator will learn a uniform distribution over the generator indices, thus, the gradients passed through the discriminator will be exactly the same for all the generators. However, this situation in general is not possible as all the generators are initialized differently. Even a slight variation in the samples from the generators will be enough for the

discriminator to identify them and pass different gradient information to each generator. In addition, the objective function of generators is *only* to generate 'real' samples, thus, there is nothing that encourage them to generate exactly the same samples.

- the discriminator does not have enough capacity to learn the optimal parameters. This is in contrast to the assumption made in Theorem 1, which is that the discriminator is *optimal*. Thus, it should have enough capacity to learn a feature representation such that it can correctly identify samples from different generators. In practice, this is a very easy task and we did not have to modify anything up to the feature representation stage of the architecture of the discriminator. We used the standard architectures (explained in Section 8) for all the tasks.

Hence, with random initializations and sufficient capacity generator/discriminator, we can easily avoid the trivial solution in which all the generators focus on exactly the same region of the true data distribution. This has been very clearly supported by various experiments showing diverse generations by MAD-GAN.

5. Experiments

We show the efficacy of our framework on both synthetic and real-world datasets. First of all, we use a simple 1D mixture of Gaussian distribution to show that our framework is able to capture diverse modes. Next, we perform experiments on Stacked/Compositional MNIST dataset (1000 modes) and compare our methods with several known variants of GANs such as DCGAN [25], WGAN [3], BEGAN [5], GoGAN [16], Unrolled GAN [21] and Mode-Reg GAN [7]. Furthermore, we also created another baseline, called *MA-GAN (Multi-Agent GAN)*, which is a trivial extension of GAN with multiple generators and one discriminator. As opposed to MAD-GAN, MA-GAN is simple Multi-Agent architecture without the modification in the discriminator's objective function. This comparison will allow us to better understand the effect of explicitly enforcing diversity in MAD-GAN's objective. We use KL-divergence [18] and number of modes recovered [7] as the criterion for comparison and show superior results compared to all other methods. In addition, we show diverse generations for the challenging tasks of 'image-to-image translation' [14] (we compare with InfoGAN in this setup), 'multi-view data generation', and 'face generation'. *Note that the image-to-image translation objective is known to learn the delta distribution, thus, it is agnostic to the input noise vector. However, we show that MAD-GAN is able to produce highly plausible diverse generations for this task.* In the end, we show that MAD-GAN is able to learn better feature space for the unsupervised representation learning task. We provide detailed overview of the CNN architectures, datasets, and the parameters used in our experiments in their respective references in Section 8 at the end of this paper.

GAN Variants	**Chi-square** $(\times 10^5)$	**KL-Div**
DCGAN	0.90	0.322
WGAN	1.32	0.614
BEGAN	1.06	0.944
GoGAN	2.52	0.652
Unrolled GAN	3.98	1.321
Mode-Reg DC-GAN	1.02	0.927
MA-GAN (without diversity enforcement)	1.39	0.526
MAD-GAN (Our)	**0.24**	**0.145**

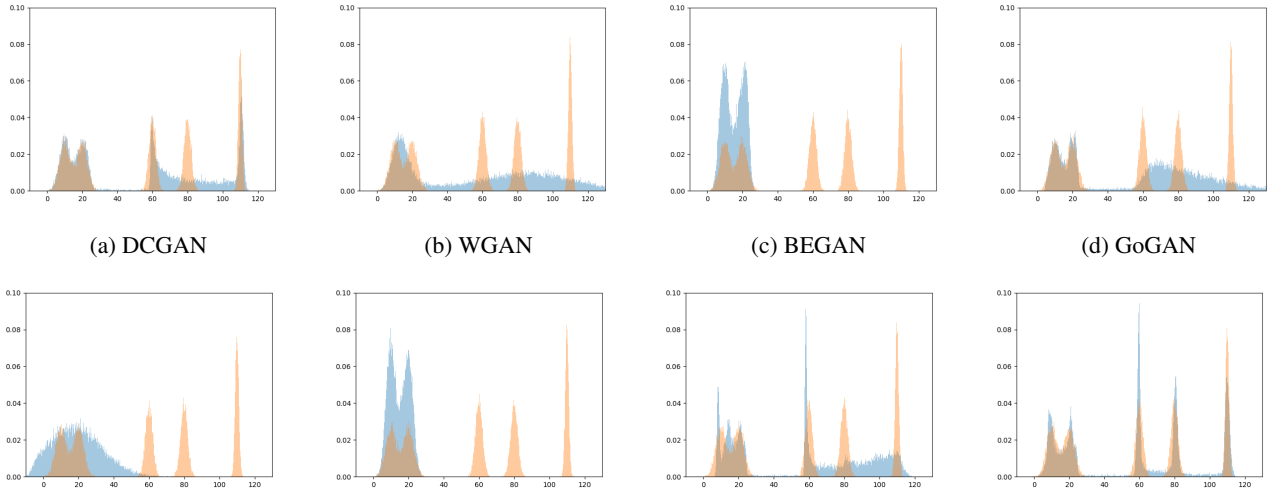
Table 1: Synthetic experiment on 1D GMM as Figure 4.

Generators	**Chi-square** $(\times 10^7)$	**KL-Div**
1	1.27	0.57
2	1.38	0.42
3	3.15	0.71
4	0.39	0.28
5	3.05	0.88
6	0.54	0.29
7	0.97	0.78
8	4.83	0.68

Table 2: Synthetic experiment with different number of MAD-GAN generators as Figure 5.

5.1. Non-Parametric Density Estimation

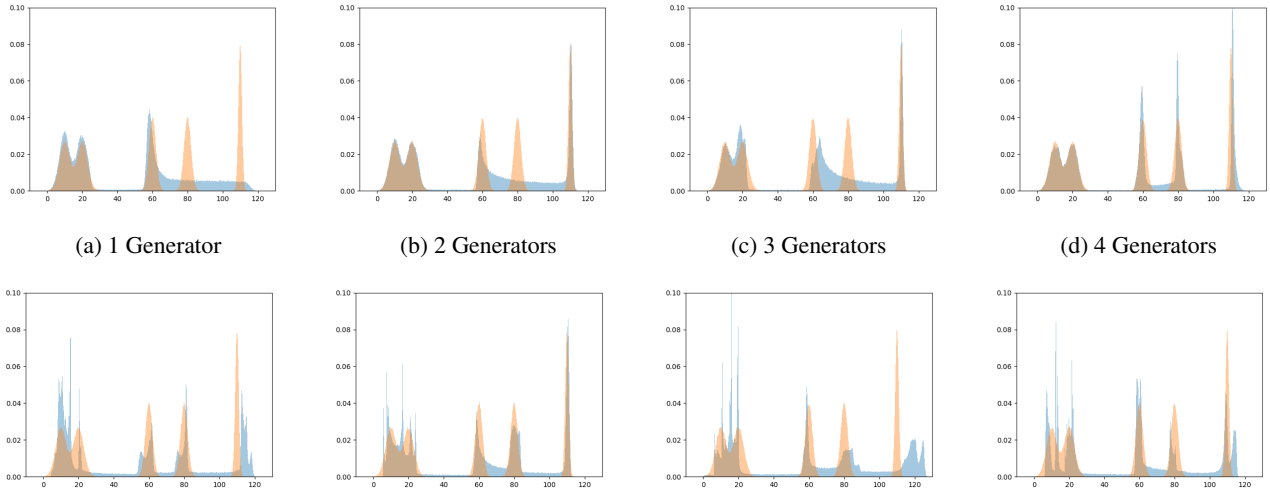
In order to understand the behaviour of MAD-GAN and different state-of-the-art GAN models, we first perform a very simple synthetic experiment, much easier than generating high-dimensional complex images. We consider a distribution of 1D GMM [6] having five mixture components with modes at 10, 20, 60, 80 and 110, and standard deviation of 3, 3, 2, 2 and 1, respectively. While the first two modes overlap significantly, the fifth mode stands isolated as evident from Figure 4. We train different GAN models using $200,000$ samples from this distribution and generate $65,536$ data points from each model. In order to compare the learned distribution with the ground truth distributions, we first estimate them using bins over the data points and create the histograms. These histograms are carefully created using different bin sizes and the best bin (found to be 0.1) is chosen. Then, we use Chi-square distance and the KL-divergence to compute distance between the two histograms. From Figure 4 and Table 1 it is evident that MAD-



(a) DCGAN (b) WGAN (c) BEGAN (d) GoGAN

(e) Unrolled GAN (f) Mode-Reg DCGAN (g) MA-GAN (h) MAD-GAN (Our)

Figure 4: A toy example to understand the behaviour of different GAN variants in order to compare with MAD-GAN (each method was trained for 198000 iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1.



(a) 1 Generator (b) 2 Generators (c) 3 Generators (d) 4 Generators

(e) 5 Generators (f) 6 Generators (g) 7 Generators (h) 8 Generators

Figure 5: A toy example to understand the behavior of MAD-GAN with different number of generators (each method was trained for 198000 iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1.

GAN is able to capture all the clustered modes which includes significantly overlapped modes as well. MAD-GAN obtains the minimum value in terms of both Chi-square distance and the KL-divergence. In this experiment, both MAD-GAN and MA-GAN used four generators. We provide in-depth analysis about the effect of varying generators in the supplementary material.

To understand the effect of varying the number of generators in MAD-GAN, we use the same synthetic experiment setup, i.e. the real data distribution is same GMM with 5 Gaussians. We use 1 million sample points from real distribution (instead of $65,536$). We generate equal number of points from each of the generators such that they sum up to 1 million. The results are shown in Figure 5 and cor-

responding Table 2. It is quite clear that as the number of generators are increased up to 4, the sampling keeps getting more realistic. For this real data distribution, four generators are enough to capture all the modes. As we increase the number of generators further, more than one generator try to capture the same mode. Hence, during sampling, more samples are obtained from that mode. As can be seen from Figure 5, when the number of generators is more than 4, more generators focus on the overlapping modes than the non-overlapping ones, so, there are more samples from the region at which they overlap.

Other works using more than one generator [20, 4] also use the number of generators as a hyper-parameter as knowing a-priori the number of modes in a real-world data (*e.g.* images) in itself is an open problem.

5.2. Stacked and Compositional MNIST

We now move to a more challenging setup, similar to [7, 21], in order to examine and compare MAD-GAN with other GAN variants. [21] created a Stacked MNIST dataset with 25,600 samples where each sample has three channels stacked together with a random MNIST digit in each of them. Thus, creating 1000 distinct modes in the data distribution. [21] used a stripped down version of the generator and discriminator pair to reduce the modeling capacity. We follow the same path for fair comparisons and used the same architecture as mentioned in their paper. Similarly, [7] created Compositional MNIST whereby they took 3 random MNIST digits and place them at the 3 quadrants of a 64×64 dimensional image. This also resulted in a data distribution with 1000 modes. The distribution of the resulting generated samples was estimated using a pretrained MNIST classifier to classify each of the digits either in the channels or the quadrants to decide which of the 1000 modes a particular generation referred to.

In Tables 3 and 4, we compare our method with many other variants of GAN in terms of KL divergence and the number of modes recovered for the Stacked and Compositional MNIST datasets, respectively. It is evident from Table 3 that MAD-GAN outperforms all other variants of GAN in terms of both the number of modes recovered and the KL divergence. Interestingly, in the case of Compositional MNIST experiments, as shown in Table 4, MAD-GAN, WGAN and Unrolled GAN were able to recover all the 1000 modes. However, in terms of KL divergence, the distribution generated by MAD-GAN is the closest to the true data distribution.

GAN Variants	**KL Div**	**# Modes Covered**
DCGAN	2.15	712
WGAN	1.02	868
BEGAN	1.89	819
GoGAN	2.89	672
Unrolled GAN	1.29	842
Mode-Reg DC-GAN	1.79	827
MA-GAN (without diversity enforcement)	3.4	700
MAD-GAN (Our)	**0.91**	**890**

Table 3: Stacked-MNIST Experiment. There are 1000 modes in the dataset. MAD-GAN recovered maximum number of modes (890) and also the generated distribution is closest to the true data distribution compared to all other variants of GAN in terms of KL divergence.

GAN Variants	**KL Div**	**# Modes Covered**
DCGAN(reproduced)	0.18	980
WGAN	0.25	1000
BEGAN	0.19	999
GoGAN	0.87	972
Unrolled GAN	0.091	1000
Mode-Reg DC-GAN (reproduced)	0.12	992
MA-GAN (without diversity enforcement)	1.62	997
MAD-GAN (Our)	**0.074**	**1000**

Table 4: Compositional-MNIST Experiment. There are 1000 modes in the dataset. MAD-GAN, WGAN and Unrolled GAN recovered all the modes (1000). However, in terms of KL divergence, the distribution generated by MAD-GAN is closest to the true data distribution.

5.3. Diverse Samples for Image-to-Image Translation and Comparison to InfoGAN

In this section we show diverse generations for the challenging task of image-to-image translation [14] which uses patch based Conditional GANs [22]. Conditional GANs for this task are known to learn the delta distribution, thus, generates the same image irrespective of the variations in the input noise vector. *Generating diverse samples in this setting in itself is an open problem.* In this set of experiments, we show that MAD-GAN is able to generate diverse samples for this task. We do not claim to capture all the possible modes as knowing a priori the number of modes in the data distribution is not possible. However, we show that with MAD-GAN we can generate as many diverse samples as the number of generators we use. Note that, adding more generators only requires adding one additional last layer. We follow the same approach as [14] and employ patch based conditional GAN for this task.

Closest to our approach is InfoGAN [8]. Theoretically,

Figure 7: Diverse generations for 'edges to handbags' generation task. In each sub-figure, the first column represents the input, columns 2-4 represents generations by MAD-GAN (using three generators), and columns 5-7 are generations by InfoGAN. It is evident that different generators are able to produce very diverse results capturing different colors (blue, green, black etc), textures (leather, fabric, plastic etc), design patterns, among others. However, InfoGAN generations are visually same, which clearly indicated that it is not able to capture diverse modes.

Figure 9: InfoGAN for 'edges to handbags' task by using three categorical code values. In each sub-figure, the first column represents the input, columns 2-4 represents generations when input is categorical code besides conditioning image, and columns 5-7 are generations with noise as an additional input. The generations for each of the two architectures are visually the same irrespective of the categorical code value, which clearly indicates that it is not able to capture diverse modes.

latent codes in InfoGAN should enable diverse generations. However, InfoGAN can only be used in situations where the bias introduced by the categorical variables have significant impact on the generator network. For image-to-image translation and high resolution generations, the categorical variable does not have sufficient impact on the generations. As will be seen shortly, we validate this hypothesis by comparing our method with InfoGAN for this task.

For InfoGAN generator, to capture three kinds of distinct modes, the categorical code can take three values. Hence, in this case, the categorical code is a 2D matrix in which one third of entries are set to 1 and remaining to 0 for each category. The generator is fed input image along with categorical code appended channel wise to the image. Q network's architecture is the same as that of pix2pix discriminator [14], except that the output is a vector of size 3 for the prediction of the categorical codes.

Figure 7 shows generations for MAD-GAN and InfoGAN for the 'edges to handbags' task, where given the edges of handbags, the objective is to generate real looking handbags. Clearly, each MAD-GAN generator is able to produce meaningful and diverse images in terms of 'color', 'texture', and 'patterns'. However, InfoGAN generations are almost the same for all the three categorical code values. The results shown for InfoGAN are obtained by not sharing the discriminator and Q network parameters.

To make our baseline as strong as possible, we did some more experiments with InfoGAN for the 'edges to hand-

Figure 11: Diverse generations for 'night to day' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing different lighting conditions, different sky patterns (cloudy vs clear sky), different weather conditions (winter, summer, rain), different landscapes, among many other minute yet useful cues.

Figure 12: Face generations using MAD-GAN. Each generator employed is DCGAN. Each row represents a generator. Each column represents generations for a given random noise input z. The first generator is generating faces pointing to left. The second generator is generating female faces with long hair, while the third generator generates the images with light background.

bags' task. For Figure 9, we did two experiments by sharing all the initial layers of the discriminator and Q network. In the first experiment, the input is categorical code besides the conditional image. In the second experiment, noise is also added as an input. The architecture details are given in Section 8.3.2. In Figure 9, we show the results of both these experiments side by side. There are still not many perceivable changes as we vary the categorical code values. Generator simply learn to ignore the input noise as was also pointed by [14].

In addition, in Figure 11, we show diverse generations using MAD-GAN for 'night to day' task, where given night images of places, the objective is to generate their equivalent day images. The generated day images in Figure 11 differ in terms of lighting conditions, sky patterns, weather conditions, and many other minute yet useful cues.

5.4. Multi-View Data Generation

To further explore the mode capturing capacity of MAD-GAN, we experimented with a much more challenging task of multi-view data generation. In detail, we trained MAD-GAN (three generators) on a combined dataset consisting of various highly diverse images such as *islets*, *icebergs*, *broadleaf-forest*, *bamboo-forest* and *bedroom*, obtained from the Places dataset [31]. To create the training data, images were randomly selected from each of them, creating a dataset consisting of $24,000$ images. The generators have the same architecture as that of DCGAN. In this case, as the images belong to very diverse modalities, the generator parameters were not shared. As shown in Figure 1, to our surprise, we found that, even in this highly challenging setting, each generator is able to generate samples from different modality. This clearly indicates that MAD-GAN is able to disentangle diverse modes even if the training dataset is highly challenging given that it contains images from varying modalities.

5.5. Diverse Face Generation

In this section we show diverse face generations (CelebA dataset) using MAD-GAN where we use DCGAN [25] as our generators. Again, we use the same setting as provided in DCGAN. The high quality face generations are shown in the Figure 12.

To get better understanding about the possible diversities, we repeated the experiment. The resulting generations are shown in Figure 14.

5.6. Unsupervised Representation Learning

Similarly to DCGAN [25], we train our framework using SVHN dataset [23]. The trained discriminator is then used to extract features. Using these features, we train an SVM for the classification task. For the MAD-GAN we obtain misclassification error of 17.5% which is almost 5% better than the results reported by DCGAN (22.48%). This clearly indicates that our framework is able to learn better feature space in an unsupervised setting.

6. Conclusion

We presented a very simple and effective framework, Multi-Agent Diverse GAN, for generating diverse and meaningful samples. We showed the efficacy of our approach and compared it with various variants of GAN using extensive experiments and showed that it captures diverse modes while producing high quality samples. We presented theoretical analysis of MAD-GAN with conditions for global optimality. An interesting future direction would be to estimate a priori the number of generators needed.

Figure 14: Face generations using MAD-GAN. Each sub-figure represents generations by a single generator. The first generator is generating faces with very dark background. The second generator is generating female faces with long hair in very light background, while the third one is generating faces with colored background and casual look (based on direction of viewing and expression) .

References

[1] M. Abadi and D. Andersen. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*, 2016. 2

[2] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. Technical report, 2017. 2, 3

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 3, 6

[4] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017. 3, 8

[5] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 3, 6

[6] C. Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007. 6

[7] T. Che, Y. Li, A. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. Technical report, 2017. ICLR. 2, 3, 6, 8

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016. 2, 3, 8

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 16

[10] I. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016. 3

[11] A. Ghosh, V. Kulharia, A. Mukerjee, V. Namboodiri, and M. Bansal. Contextual rnn-gans for abstract reasoning diagram generation. In *AAAI*, 2017. 2

[12] A. Ghosh, V. Kulharia, and V. Namboodiri. Message passing multi-agent gans. *arXiv preprint arXiv:1612.01294*, 2016. 3

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 1, 3, 4

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016. 2, 6, 8, 9, 10, 15

[15] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 2009. 13

[16] F. Juefei-Xu, V. N. Boddeti, and M. Savvides. Gang of gans: Generative adversarial networks with maximum margin ranking. *arXiv preprint arXiv:1704.04865*, 2017. 3, 6

[17] D. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[18] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 1951. 6

[19] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 2

[20] M. Liu and O. Tuzel. Coupled generative adversarial networks. *arXiv preprint arXiv:1606.07536*, 2016. 2, 3, 8

[21] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017. 2, 3, 6, 8, 14

[22] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2, 8

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. N. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 2, 10, 17

[24] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. *arXiv preprint arXiv:1604.07379*, 2016. 2

[25] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2, 3, 6, 10, 14, 16, 17

[26] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016. 2

[27] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016. 2, 3, 4, 13

[28] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004. 13

[29] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016. 2

[30] J. Wu, C. Zhang, T. Xue, W. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *arXiv preprint arXiv:1610.07584*, 2016. 2

[31] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 10, 16

[32] J. Zhu, P. Krähenbühl, E. Shechtman, and A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2, 16

7. Similarity based competing objective

We have discussed the MAD-GAN architecture using generator identification based objective. In this section, we propose a different extension to the standard GAN : *similarity based competing objective (SCO)*. Here, we augment by the GAN objective function with a diversity enforcing term. It ensures that the generations from different generators are diverse where the diversity depends on a user-defined task-specific function. We call this approach MAD-GAN-Sim: similarity based competing objective for MAD-GAN.

The architecture is same as MAD-GAN discussed in Section 4.1 (refer Figure 15).

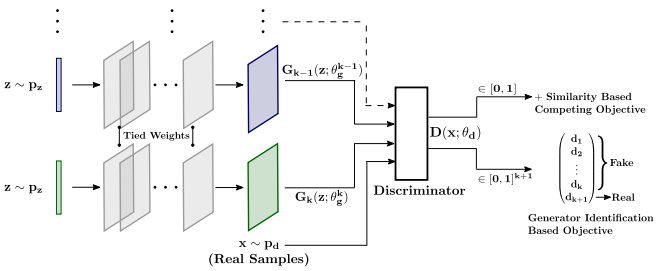


Figure 15: MAD-GAN-Sim compared with MAD-GAN. All the generators share parameters of all the layers except the last one. Two proposed diversity enforcing objectives, 'competing' (MAD-GAN-Sim) and 'generator identification' (MAD-GAN), are shown at the end of the discriminator

7.1. Approach

The approach presented here is motivated by the fact that the samples from different modes must look different. For example, in the case of images, these samples should differ in terms of texture, color, shading, and various other cues. Thus, different generators must generate dissimilar samples where the dissimilarity comes from a task-specific function. Before delving into the details, let us first define some notations in order to avoid clutter. We denote θ_g^i as the parameters of the i-th generator. The set of generators is denoted as $K = \{1, \cdots, k\}$. Given random noise z to the i-th generator, the corresponding generated sample $G_i(z; \theta_g^i)$ is denoted as $g_i(z)$. Using these notations and following the above discussed intuitions, we impose following constraints over the i-th generator while updating its parameters:

$$D(G_i(z;\theta_g^i);\theta_d) \geq D(G_j(z;\theta_g^j);\theta_d) + \Delta\big(\phi(g_i(z)), \phi(g_j(z))\big), \quad \forall j \in K \setminus i \tag{10}$$

where, $\phi(g_i(z))$ denotes the mapping of the generated image $g_i(z)$ by the i-th generator into a feature space and $\Delta(.,.) \in [0,1]$ is the similarity function. Higher the value of $\Delta(.,.)$ more similar the arguments are. Intuitively, the

above set of constraints ensures that the discriminator score for each generator should be higher than all other generators with a margin proportional to the similarity score. If the samples are similar, the margin increases and the constraints become more active. We use unsupervised learning based representation as our mapping function $\phi(.)$. Precisely, given a generated sample $g_i(z)$, $\phi(g_i(z))$ is the feature vector obtained using the discriminator of our framework. This is motivated by the feature matching based approach to improve the stability of the training of GANs [27]. The $\Delta(.,.)$ function used in this work is the standard cosine similarity based function. The above mentioned constraints can be satisfied by maximizing an equivalent unconstrained objective function as defined below:

$$U(\theta_g^i, \theta_d) := f\Big(D(G_i(z;\theta_g^i);\theta_d) - \frac{1}{k-1}\sum_{j\in K\setminus i}\big(D(G_j(z;\theta_g^j);\theta_d) + \Delta(\psi_i,\psi_j)\Big) \quad (11)$$

where, $f(a) = \min(0, a)$, $\psi_i = \phi(g_i(z))$, and $\psi_j = \phi(g_j(z))$. Intuitively, if the argument of $f(.)$ is positive, then the desirable constraint is satisfied and there is no need to do anything. Otherwise, maximize the argument with respect to θ_g^i. Note that instead of using all the constraints independently, we use the average of all of them. Another approach would be to use the constraint corresponding to the j-th generator that maximally violates the set of constraints shown in Equation 10. Experimentally we found that the training process of the average constraint based objective is more stable than the maximum violated constraint based objective. The intuition behind using these constraints comes from the well know 1-slack formulation of the structured SVM framework [15, 28]. Thus, the overall objective for the i-th generator is:

$$\min_{\theta_g^i} \; V(\theta_d, \theta_g^i) - \lambda\, U(\theta_g^i, \theta_d) \quad (12)$$

where $\lambda \geq 0$ is the hyperparameter. Algorithm 1 shows how to compute gradients corresponding to different generators for the above mentioned objective function. Notice that, once sampled, the same z is passed through all the generators in order to enforce constraints over a *particular generator* (as shown in Equation 10). However, in order for constraints to not to contradict with each other while updating another generator, a different z is sampled again from the p_z. The Algorithm 1 is shown for the batch of size one which can be trivially generalized for any given batch sizes. In the case of discriminator, the gradients will have exactly the same form as the standard GAN objective. The only difference is that in this case the fake samples are being generated by k generators, instead of one.

Algorithm 1 Updating generators for MAD-GAN-Sim

input θ_d; $p(z)$; $\theta_g^i, \forall i \in \{1, \cdots, k\}$; λ.
for each generator $i \in \{1, \cdots, k\}$ **do**
  Sample noise from the given noise prior $z \sim p_z$.
  Obtain the generated sample $G_i(z;\theta_g^i)$ and corresponding feature vector $\psi_i = \phi(G_i(z;\theta_g^i))$.
  $\nu \leftarrow 0$.
  for each generator $j \in \{1, \cdots, k\} \setminus i$ **do**
    Compute feature vector $\psi_j = \phi(G_j(z;\theta_g^j))$.
    $\nu \leftarrow \nu + D(G_j(z;\theta_g^j);\theta_d) + \Delta(\psi_i, \psi_j)$.
  end for
  $\nu \leftarrow D(G_i(z;\theta_g^i);\theta_d) - \frac{\nu}{k-1}$.
  if $\nu \geq 0$ **then**
    $\nabla_{\theta_g^i} \log(1 - D(G_i(z;\theta_g^i);\theta_d)))$.
  else
    $\nabla_{\theta_g^i}\big(\log(1 - D(G_i(z;\theta_g^i);\theta_d))) - \lambda U(\theta_g^i, \theta_d)\big)$.
  end if
end for
output

7.2. Experiments

We present the efficacy of MAD-GAN-Sim on the real world datasets.

7.3. Diverse Samples for Image-to-Image Translation

We show diverse and highly appealing results using the diversity promoting objective. We use cosine based similarity to enforce diverse generations, an important criteria for real images. As before, we show results for the following two situations where diverse solution is useful: (1) given the edges of handbags, generate real looking handbags as in Figure 17; and (2) given night images of places, generate their equivalent day images as in Figure 19. We clearly notice that each generator is able to produce meaningful and diverse images.

7.4. Unsupervised Representation Learning

We do the same experiment using SVHN as done in Section 5.6. For MAD-GAN-Sim we obtained the misclassification error of 18.3% which is better than DCGAN (22.48%). It clearly indicates that MAD-GAN-Sim is able to learn better feature representation in an unsupervised setting.

8. Network Architectures and Parameters

In this section we give all the details about the architectures and the parameters used in various experiments shown in the main paper.

DCGAN, Unrolled GAN, MA-GAN Disc	Mode-Reg DCGAN Disc	Mode-Reg DCGAN Enc	WGAN, GoGAN Disc	BEGAN Enc	BEGAN Dec	MAD-GAN Disc
Input: 1	1	1	32	1	1	1
fc: 128, leaky relu						
fc: 128, leaky relu						
fc: 1	1	64	1	32	1	5(nGen + 1)
sigmoid		identity				softmax

Table 5: Non-Parametric density estimation architecture for discriminators (Disc), encoders (Enc) and decoders (Dec). nGen is number of generators, fc is fully connected layer.

Figure 17: MAD-GAN-Sim: Diverse generations for 'edges to handbags' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing color (brown, pink, black), texture, design pattern, shininess, among others.

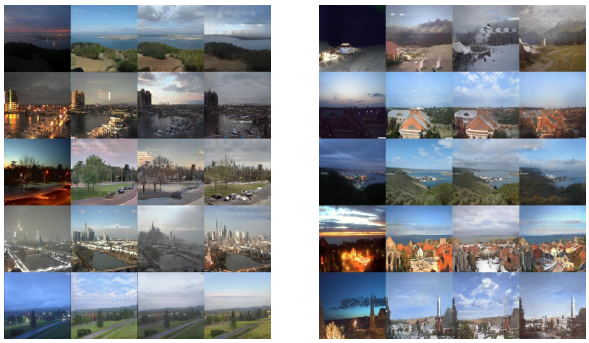


Figure 19: MAD-GAN-Sim: Diverse generations for 'night to day' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing different lighting conditions, different sky patterns (cloudy vs clear sky), different weather conditions (winter, summer, rain), different landscapes, among many other minute yet useful cues.

8.1. Non-Parametric Density Estimation

The architecture and dataset details are given below.

Architecture Details: The generator has two fully connected hidden layers with 128 neurons, each followed by exponential linear, and fully connected outer layer. In case of MAD-GAN and MA-GAN we used 4 generators with parameters of first two layers shared. Generator generates 1D samples. Input to each generator is a uniform noise $U(-1, 1)$ of 64 dimension. The discriminator architecture for respective networks is shown in Table 5. Mode-Regularized GAN architecture has encoder, BEGAN has encoder and decoder whose details are also present in Table 5.

MAD-GAN has multi-label cross entropy loss. MA-GAN has binary cross entropy loss. For training, we use Adam optimizer with batch size of 128 and learning rate of $1e-4$. In each mini batch, for MAD-GAN we have 128 samples from each of the generators as well as real distribution, while for MA-GAN 128 samples are chosen from real distribution as well as all the generators combined.

Dataset Generation We generated synthetic 1D data using GMM with 5 Gaussians and select their means at $10, 20, 60, 80$ and 110. The standard deviation used is $3, 3, 2, 2$ and 1. The first two modes overlap significantly while the fifth one is peaky and stands isolated.

8.2. Stacked and compositional MNIST Experiments

Architecture details: The architecture for stacked MNIST is similar to the one used in [21]. Please refer to the Table 6 for generator architecture and Table 7 for discriminator architecture. The architecture for compositional MNIST experiment is same as DCGAN [25].

	number outputs	stride
Input: $z \sim \mathcal{N}(0, I_{256})$		
Fully connected	4 * 4 * 64	
Reshape to image 4,4,64		
Transposed Convolution	32	2
Transposed Convolution	16	2
Transposed Convolution	8	2
Convolution	3	1

Table 6: Generator architecture for 1000 class stacked MNIST experiment.

	number outputs	stride
Input: $x \sim p_{data}$ or G		
Convolution	4	2
Convolution	8	2
Convolution	16	2
Flatten		
Fully Connected	1	

Table 7: Discriminator architecture for 1000 class stacked MNIST experiment.

Dataset preparation: MNIST database of hand written digits are used for both the tasks.

8.3. Image to Image Translation

8.3.1 MAD-GAN / MAD-GAN-Sim

Architecture details: The network architecture is adapted from [14] and the experiments were conducted with the U-Net architecture and patch based discriminator.

In more detail, let Ck denote a Convolution-BatchNorm-ReLU layer with k filters and CDk represent a Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of 50%. All Convolutions are 4x4 spatial filters with a stride of 2. Convolutions in the encoder, and in the discriminator, downsample by a factor of 2, whereas in the decoder they upsample by a factor of 2.

Generator Architectures We used the U-Net generator based architecture from [14] as follows:

- U-Net Encoder: C64-C128-C256-C512-C512-C512-C512-C512
- U-Net Decoder: CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128. Note that, in case of MAD-GAN, the last layer does not share parameters with other generators.

After the last layer in the decoder, a convolution is applied to map to the number of output channels to 3, followed by a tanh function. BatchNorm is not applied to the first C64 layer in the encoder. All ReLUs in the encoder are leaky, with a slope of 0.2, while ReLUs in the decoder are not leaky. The U-Net architecture has skip-connections between each layer i in the encoder and layer $n - i$ in the decoder, where n is the total number of layers. The skip connections concatenate activations from layer i to layer $n - i$. This changes the number of channels in the decoder.

Discriminator Architectures The patch based 70x70 discriminator architecture was used in this case : C64-C128-C256-C512.

Diversity term

- MAD-GAN: After the last layer, a convolution is applied to map the output layer to the dimension of $k+1$ (where k is the number of generators in MAD-GAN) followed by the softmax layer for the normalization.
- MAD-GAN-Sim: After the last layer, a convolution is applied to map to a 1 dimensional output followed by a Sigmoid function. For the unsupervised feature representation $\phi(.)$, the feature activations from the penultimate layer C256 of the discriminator was used as the feature activations for the computation of the cosine similarity.

For the training, we used Adam optimizer with learning rate of $2e{-}4$ (for both generators and discriminator), $\lambda_{L1} = 10$ (hyperparameter corresponding to the L_1 regularizer), $\lambda = 1e - 3$ (corresponding to MAD-GAN-Sim), and batch size of 1.

8.3.2 InfoGAN

The network architecture is adapted from [14] and the experiments were conducted with the U-Net architecture and patch based discriminator.

Generator Architectures The U-Net generator is exactly same as in [14] except that the number of input channels are increased from 3 to 4. For the experiment done for Figure 9 to take noise as input, input channels are increased to 5 (one extra input channel for noise).

Discriminator Architectures The discriminator is exactly same as in [14]: C64-C128-C256-C512

Q network Architectures The Q network architecture is C64-C128-C256-C512-Convolution3-Convolution3. Here first Convolution3 gives a output of 30×30 patches with 3 channels while second Convolution3 just gives 3 dimensional output. To perform the experiments for Figure 9 by sharing layers, all the layers except last two are shared with the discriminator.

Diversity term To capture three kinds of distinct modes, the categorical code can take three values. Hence, in this case, the categorical code is a 2D matrix in which one third of entries are set to 1 and remaining to 0 for each category. The generator is fed input image along with categorical code appended channel wise to the image. For the experiment done for Figure 9, to take noise as input, the generator input is further channel wise appended with a 2D matrix of normal noise.

For the training, we used Adam optimizer with learning rate of $2e-4$ (for both generator and discriminator), $\lambda_{L1} = 10$ (hyperparameter corresponding to the L_1 regularizer) and batch size of 1.

Dataset Preparation:

- Edges To Handbags: We used 137k Amazon handbag images from [32]. The random split into train and test was kept the same as done by [32].
- Night To Day: We used 17823 training images extracted from 91 webcams. We thank Jun-Yan Zhu for providing the dataset.

8.4. Multi-View Data Generation

Architecture details: The network architecture is adapted from DCGAN [25]. Concretely, the discriminator architecture is described in Table 10 and the generator architecture in Table 9. We use three generators without sharing any parameter. The residual layers helped in improving the image quality since the data manifold was much more complicated and the discriminator needed more capacity to accommodate it.

Diversity terms For the training, we used Adam optimizer with the learning rate of $2e-4$ (both generator and discriminator) and batch size of 64.

Dataset preparation: Training data is obtained by combining dataset consisting of various highly diverse images such as *islets*, *icebergs*, *broadleaf-forest*, *bamboo-forest* and *bedroom*, obtained from the Places dataset [31]. To create the training data, images were randomly selected from each of them, creating a dataset consisting of $32,000$ images.

Discriminator D
Input 64x64 Color Image
4x4 conv. 64 leakyRELU. stride 2. batchnorm
4x4 conv. 128 leakyRELU. stride 2. batchnorm
4x4 conv. 256 leakyRELU. stride 2. batchnorm
4x4 conv. 512 leakyRELU. stride 2. batchnorm
4x4 conv. output leakyRELU. stride 1

Table 8: DCGAN Discriminator: It is adapted to have $k+1$ dimensional last layer output for MAD-GAN with k generators. (normalizer is softmax).

Generator G
Input $\in \mathbb{R}^{100}$
4x4 upconv. 512 RELU.batchnorm.shared
4x4 upconv. 256 RELU. stride 2.batchnorm.shared
4x4 upconv. 128 RELU. stride 2.batchnorm.shared
4x4 upconv. 64 RELU. stride 2.batchnorm.shared
4x4 upconv. 3 tanh. stride 2

Table 9: DCGAN Generator: All the layers except the last one are shared among all the three generators.

8.5. Diverse Face Generations with DCGAN

Architecture details: The network architecture is adapted from DCGAN [25]. Concretely, the discriminator architecture is described in Table 10 and the generator architecture in Table 9. In this case all the parameters of the generators except the last layer were shared. The residual layers helped in improving the image quality since the data manifold and the manifolds of each of the generators was much more complicated and the discriminator needed more capacity to accommodate it.

Diversity terms For the training, we used Adam optimizer with the learning rate of $2e-4$ (both generator and discriminator) and batch size of 64.

Dataset preparation: We used CelebA dataset as mentioned for face generation based experiments. For Image generation all the images (14,197,122) from the Imagenet-1k dataset [9] were used to train the DCGAN with 3 Generators alongside the MAD-GAN objective. The images from both CelebA and Imagenet-1k were resized into 64x64.

Residual Discriminator D
Input 64x64 Color Image
7x7 conv. 64 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 64 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 128 leakyRELU. stride 2.pad 1. batchnorm
3x3 conv. 256 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 512 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 512 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 512 leakyRELU. stride 2. pad 1. batchnorm
RESIDUAL-(N512, K3, S1, P1)
RESIDUAL-(N512, K3, S1, P1)
RESIDUAL-(N512, K3, S1, P1)

Table 10: Multi-View Data Generation and Diverse Face Generation: The last layer output is $k+1$ dimensional for MAD-GAN with k generators (normalizer is softmax). RESIDUAL layer is elaborated in Table 11.

RESIDUAL-Residual Layer
Input: previous-layer-output
c1: CONV-(N512, K3, S1, P1), BN, ReLU
c2: CONV-(N512, K3, S2, P1), BN
SUM(c2,previous-layer-output)

Table 11: Residual layer description for Table 10.

Technique	**2 Gen**	**3 Gen**	**4 Gen**
MAD-GAN	20.5%	18.2%	17.5%
MAD-GAN-Sim	20.2%	19.6%	18.3%

Table 12: The misclassification error of MAD-GAN and MAD-GAN-Sim on SVHN with different number of generators are shown.

8.6. Unsupervised Representation Learning

Architecture details: Our architecture uses the one proposed in DCGAN [25]. Similar to the DCGAN experiment on SVHN dataset (32x32x3) [23], we removed the penultimate layer of generator (second last row in Table 9) and first layer of discriminator (first convolution layer in Table 8).

Classification task: We trained our model on the available SVHN dataset [23]. For feature extraction using discriminator, we followed the same method as mentioned in the DCGAN paper [25]. The features were then used for training a regularized linear L2-SVM. The ablation study is presented in Table 12.

Dataset preparation: We used SVHN dataset [23] consisting of $73,257$ digits for the training, $26,032$ digits for the testing, and $53,1131$ extra training samples. As done in DCGAN [25], we used 1000 uniformly class distributed random samples for training, $10,000$ samples from the non-extra set for validation and 1000 samples for testing.

For the training, we used Adam optimizer with learning rate of $2e-4$ (both generator and discriminator), $\lambda = 1e-4$ (competing objective), and batch size of 64.