Mixture Density Generative Adversarial Networks

Hamid Eghbal-zadeh[1] Werner Zellinger[2,3] Gerhard Widmer[1]

[1] LIT AI Lab & Institute of Computational Perception, Johannes Kepler University Linz
[2] Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz
[3] Software Compenetce Center Hagenberg GmbH
{hamid.eghbal-zadeh, werner.zellinger, gerhard.widmer}@jku.at

Abstract

Generative Adversarial Networks have a surprising ability to generate sharp and realistic images, but they are known to suffer from the so-called mode collapse problem. *In this paper, we propose a new GAN variant called Mixture Density GAN that overcomes this problem by encouraging the Discriminator to form clusters in its embedding space, which in turn leads the Generator to exploit these and discover different modes in the data. This is achieved by positioning Gaussian density functions in the corners of a simplex, using the resulting Gaussian mixture as a likelihood function over discriminator embeddings, and formulating an objective function for GAN training that is based on these likelihoods. We show how formation of these clusters changes the probability landscape of the discriminator and improves the mode discovery of the GAN. We also show that the optimum of our training objective is attained if and only if the generated and the real distribution match exactly. We support our theoretical results with empirical evaluations on three mode discovery benchmark datasets (Stacked-MNIST, Ring of Gaussians and Grid of Gaussians), and four image datasets (CIFAR-10, CelebA, MNIST, and Fashion-MNIST). Furthermore, we demonstrate (1) the ability to avoid mode collapse and discover all the modes and (2) superior quality of the generated images (as measured by the Fréchet Inception Distance (FID)), achieving the lowest FID compared to all baselines.*

1. Introduction

Generative Adversarial Networks (GANs) [11] learn an implicit estimate of the Probability Density Function (PDF) underlying a set of training data, and can learn to generate realistic new samples. One of the known issues in GANs is the so-called *mode collapse* [1, 10, 22], where the generator misses many modes when trained on a multi-modal dataset, and achieves a low diversity in generating samples.

In this paper, we propose **Mixture Density GAN (MD-GAN)**, which is capable of generating high-quality samples, and in addition copes with the mode collapse problem and enables the GAN to generate samples with a high variety. The central idea of MD-GAN is to enable the discriminator to create several clusters in its output embedding space for real images, and therefore provide better means for distinguishing not only real and fake, but also between different *kinds* of real images.

The discriminator in MD-GAN forms a number of clusters[1] over embeddings of real images which represent clusters in the real data. To fool the discriminator, the generator then has to generate images that the discriminator has to embed close to the center of these clusters to increase the likelihood of the generated images using the mixture density function. As there are multiple clusters, the generator can discover various modes by generating images that end up in various clusters in the discriminator embedding space.

MD-GAN's Discriminator uses a d-dimensional embedding space and is provided with an objective function that pushes it towards forming clusters in this space that are arranged in the form of a *Simplex*[2]: each cluster center is located in one of the vertices of this simplex.

In our experiments, we use seven benchmark datasets to demonstrate the ability of MD-GAN to generate samples with good quality and high variety, and to avoid mode collapse. We show that MD-GAN outperforms the state of the art, discovering the most number of modes in three benchmark datasets designed for evaluating mode collapse. Comparing our results to state-of-the-art methods in terms of the Fréchet Inception Distance (FID) [14] using only the basic DCGAN [28] architecture, we will demonstrate that MD-GAN also achieves state-of-the-art image quality.

[1] The number of clusters is a parameter that can be set.

[2] A simplex is a generalization of the notion of a tetrahedron with d dimensions and $d+1$ vertices [25]. The cluster centers are thus equidistant.

2. Related Work

GANs exhibit a surprising ability to generate sharp and realistic images – in contrast to the blurry images generated via other techniques such as VAEs [15] –, but they are known to be difficult to train. Since the advent of the first GAN (*'vanilla'*) [11], many variations have been proposed to make training easier by optimizing an alternative objective function. The Energy-Based GAN (EBGAN) [35] and its improved variant Boundary-Equilibrium GAN (BEGAN) [3] use an auto-encoder as discriminator and reconstruction loss as an energy function that assigns low energies to the regions near the data manifold and higher energies to others. The Wasserstein GAN (WGAN) [2] and Wasserstein GAN with Gradient Penalty (WGAN-GP) [12] minimize the Wasserstein distance between the activations of real and fake images in a Lipschitz-constraint discriminator. The McGAN [26] minimizes moment distances as proposed in [33, 34]. SpectralNormGAN [24] controls the Lipschitz constant of the discriminator using a novel weight normalization technique and achieves training stability. InfoGAN [5] disentangles the latent space of the generator by maximising the mutual information between a subset of the noise and the generated image, and stabilizes the training. DeliGAN [13] increases the intra-class diversity in low-data regimes by using a more expressive noise distribution drawn from a Gaussian Mixture.

Some authors have investigated solutions to the *mode collapse* problem. A Bayesian formulation is used [27] to optimize a GAN using stochastic gradient Hamiltonian Monte Carlo, which makes for more diverse generated samples. In [7] multiple discriminators are used to avoid mode collapse while [9] laverages multiple generators to improve the mode discovery. VEEGAN [29], uses a discriminator that auto-encodes Gaussian noise and succeeds in discovering the modes in the data, though it does not assess the quality of generated images. Unrolled GAN [23] "unrolls" several gradient steps ahead when computing the gradients for the generator to tell the generator how the discriminator will behave in the next updates.

Our work is different from [7, 9] since it uses a single discriminator and generator, in contrast to the multi-discriminator and multi-agent GANs which uses an ensemble of discriminators and generators. MD-GAN differs from [27], which requires time-consuming MC sampling, and from [5] that minimises mutual information. MD-GAN uses equal updates in discriminator and generator, hence different from [30, 23, 24], which require several additional updates to compute the gradients for a single update of generator/discriminator. MD-GAN uses a simple uniform noise and a vanilla generator, hence differs from [13]. Hence, MD-GAN is computationally more efficient and memory-friendly. Also, MD-GAN differs from VEEGAN [29] as it does not require an additional inference model for auto-encoding: MG-GAN does not auto-encode. VEEGAN's objective is built upon a Gaussian distribution used in both the discriminator and the generator, which has to match. This limits VEEGAN since choosing the distribution of the noise in the generator can indeed negatively affect the quality and diversity of a GAN as discussed in [31]. In contrast, MD-GAN allows the noise distribution in the generator to be selected independently from the embedding distribution in the discriminator.

3. Background

3.1. Generative Adversarial Networks

A GAN usually consists of two neural networks competing with each other: (1) A generator network G that decodes noise vectors into images, and (2) a discriminator network D that encodes images into a notion of probability of an image being real or fake. A *real* image is one from a dataset of training images; an image generated by the generator is considered a *'fake'*.

The generator tries to fool the discriminator by learning to generate images that in the discriminator produce high probabilities of being real and that thus cannot be distinguished from real images by the discriminator. For this game to go on, the discriminator is trained to produce high probabilities for images coming from the real image dataset, and low probabilities for images coming from the generator. As training continues, the generator learns to generate images that produce high probabilities in the discriminator, which usually look also realistic to humans and are visually similar to samples from the training dataset.

3.2. The Vanilla GAN Objectives

The loss of discriminator D and generator G in the two-player minimax game of the Vanilla GAN was introduced in [11] as

$$\min_G \max_D \mathcal{L}(G,D) = \min_G \max_D \big(\mathbb{E}_{\boldsymbol{x}\sim p_{data}}\big[\log D(\boldsymbol{x})\big] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\big[\log(1-D(G(\boldsymbol{z})))\big]\big) \quad (1)$$

where for a given input image $\boldsymbol{x}$, the discriminator D outputs an estimated probability of the image coming from the dataset of real images. p_{data} represents the distribution of real images, and $p_{\boldsymbol{z}}$ the distribution of noise. $\boldsymbol{z}$ is an observation from a random distribution $p_{\boldsymbol{z}}$; generator G creates a fake image using this $\boldsymbol{z}$. These two objectives and their differences are discussed in detail in [10, 8]. In the present paper, the GAN proposed in [11] will be called *vanilla GAN*.

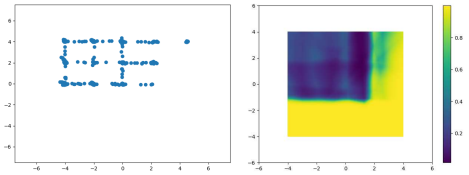
3.3. Mode Collapse Problem

As explained above, mode collapse happens when the generator generates only samples from a number of modes

in the data. To understand the reasons for mode collapse, we first have to keep in mind that the objective of the GAN is a minimax objective, and two networks play against each other, where the generator tries to fool the discriminator by generating samples that are similar to the real samples, while the discriminator tries to distinguish the fake samples from the real ones. However, the fake generated samples do not need to represent all the data modes; the discriminator can be fooled even if the generator generates samples from only parts of the data space. The generator receives its training signals directly from the discriminator, hence it is important to know how the discriminator interacts with the data space. As we discussed in Section 3.1, the discriminator can be basically seen as a binary classifier that tries to separate the real samples (with label 1) from fake samples (label 0). Consequently, the discriminator has to assign high probabilities to the areas of the data space where real examples are located, and low probabilities to the rest. These probabilities can be also seen as the training signal sent to the generator.

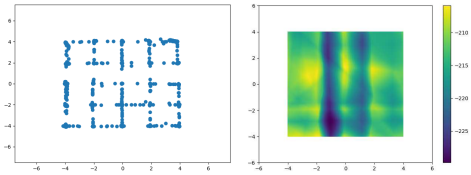
In Figure 1 we visualize the probability landscape of the discriminator where we trained the GAN on data sampled from a 2D grid of 25 Gaussians as real data. As can be seen in Figure 1a, the vanilla GAN misses many of the clusters and can only discover a small number of modes. Looking at the probability landscape[3] of the vanilla discriminator, it can be seen that the discriminator produces very high probabilities in the areas of the missing modes. This shows that the generator already lost the game in those areas and can no longer fool the discriminator in those areas.

We will tackle this issue by empowering the discriminator to better cover the data space and provide a more uniform probability landscape. In the following sections, we show that by creating several clusters in the discriminator embedding space and using these for the separation of real from fake, MD-GAN can create a more uniform probability landscape (Figure 1d) and provide better training signals to the generator to recover all the modes (Figure 1c).



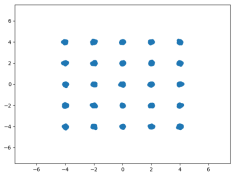
(a) Generated samples from vanilla GAN.

(b) Probability landscape of the vanilla discriminator.



(c) Generated samples from MD-GAN.

(d) Probability landscape of the MD-GAN discriminator.



(e) Real data.

Figure 1: Comparison of probability landscape and generated data between vanilla and MD-GAN.

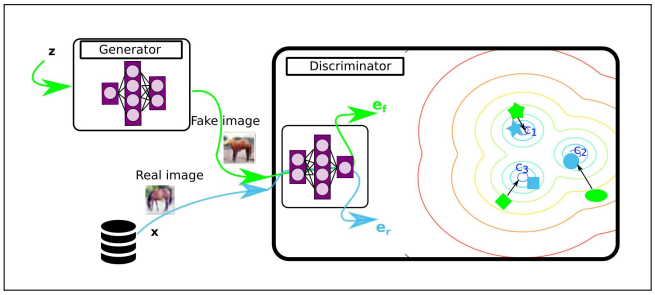


Figure 2: Block diagram of Mixture Density GAN. This figure should be viewed in color.

4. Mixture Density Generative Adversarial Networks

4.1. Mixture Density GAN: The Intuition

As explained in the introduction, the basic idea in Mixture Density GAN is to encourage the discriminator to form a number of clusters over the embeddings of real images. As also mentioned, these clusters will be positioned in an equi-distant way, their center vectors forming a *simplex*. Each cluster is represented by a Gaussian kernel. The whole collection thus makes up a *Mixture of Gaussians*, which we will call a *Simplex Gaussian Mixture Model (SGMM)*. Each of the clusters draw embeddings of fake images towards their center. This is achieved by using the SGMM as a likelihood function. Each Gaussian kernel spreads its density over the embedding space: the closer an embedding to the high probability areas (e.g, center of a cluster), the more density it gets and, therefore, the more likelihood reward it receives.

By defining a likelihood function via the parameters of a SGMM, in each update we train the discriminator to en-

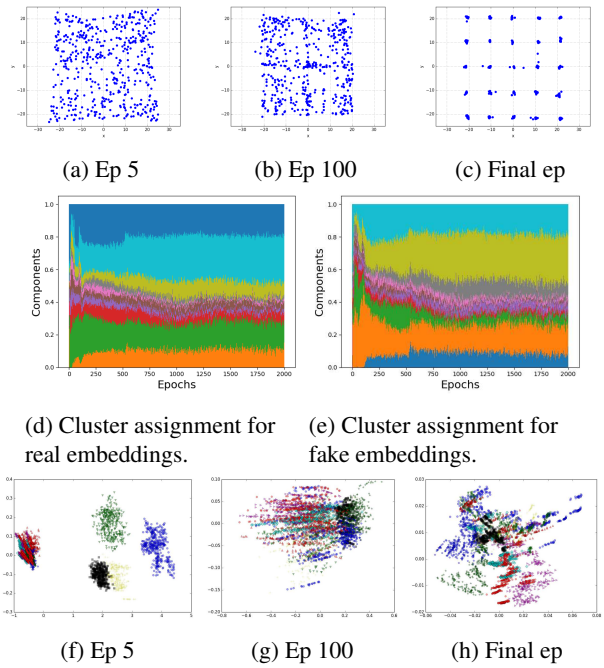
[3]The probability landscape is computed by feeding a mesh of data points from the 2D space and computing the output probability for each point. The probability is then shown in the position of that point, via a color code.

code real images to the centers of the clusters. The resulting SGMM creates a mixture of clusters that draws the real embeddings towards the cluster centers (see Figure 2). Likewise, the generator will be rewarded if it generates samples that end up in any of these clusters. Thus, if the fake embeddings are well spread around the cluster space – which they are likely to be at the beginning of training, when they are essentially just random projections –, it is likely that most of the clusters will 'catch' some fake embeddings. Therefore, the generator will tend to learn to generate samples with more variety to cover all of the clusters, which ideally results in discovering the modes present in the data. On the other hand, it is reasonable to expect the discriminator to create such clusters based on relevant similarities in the data, since it is trained as a classifier and therefore needs to learn a meaningful distance in its embedding space.

To demonstrate this, we trained MD-GAN on a 2D dataset of a grid of 25 Gaussians (Figure 1e) and visualized some aspects of the resulting model in Figure 1 and Figure 3. Figure 1 shows an example where MD-GAN can discover more modes in the data, compared to *vanilla* GAN. In particular, comparing Figure 1d and Figure 1b shows the significant differences in the respective probability landscapes. In Figure 3, the first row (a-c) shows the samples generated in various epochs. As can be seen, the data spreads nicely through data space and results in discovering all the modes. The second row (d-e) shows the percentages of real and fake embeddings assigned to each Gaussian component in different epochs. Here, the x axis represents the epochs and y the different components. Each color represents a different Gaussian component; the width of each color shows the percentage of embeddings assigned to that component in the given epoch. As can be seen, the embeddings of both real and fake are spread among all the components, and the clusters we were aiming to form are already created in the embedding space of MD-GAN discriminator. The third row of Figure 3 shows a 2D PCA projection of the embeddings from the 9D space of the discriminator. Embeddings from the different Gaussian components are shown in different color. As we expected, these clusters reflect similarities/distances in input data space: as the discriminator further differentiates the data into more specific clusters (Fig. 3h), we see that data points from the same Gaussian component (color) in the input data tend to be embedded in coherent clusters in the embedding space of the discriminator. In the next sections, we explain the technical and theoretical details of MD-GAN.

4.2. Mixture Density GAN: The Model

As in the vanilla GAN, MD-GAN consists of a generator G and a discriminator D. MD-GAN uses a mixture of Gaussians in its objective functions whose mean vectors are placed in the vertices of a d-dimensional simplex, where d is a parameter.



(a) Ep 5 (b) Ep 100 (c) Final ep

(d) Cluster assignment for real embeddings.

(e) Cluster assignment for fake embeddings.

(f) Ep 5 (g) Ep 100 (h) Final ep

Figure 3: a-c) Samples generated by Mixture Density GAN's generator in different epochs. d-e) Assignment percentages of embeddings to Gaussian components for real and fake data. Each color represents a Gaussian cluster; the width of the color in each column corresponds to the proportion of embeddings that were assigned to that cluster after a whole training epoch. f-h) $2D$ PCA-projected embeddings of MD-GAN's discriminator for real data in different epochs. Embeddings of data points coming from the same input Gaussian have the same color.

Discriminator: The discriminator D in MD-GAN is a neural network with d-dimensional output. For an input image $\boldsymbol{x}$, the discriminator creates an embedding $\boldsymbol{e}$ which is simply the activation of the last layer of the network for input $\boldsymbol{x}$. The SGMM in MD-GAN is a Gaussian mixture with the following properties:

1. The individual components are d-dimensional multivariate Gaussians (where d is the output/embedding dimensionality of the discriminator network).
2. The model comprises $d + 1$ Gaussian components, whose mean vectors are exactly the coordinates of the vertices of a simplex.
3. The covariance matrices are diagonal and have equal values on the main diagonal, in all the components. Thus, all components are spherical Gaussians.

For an embedding $\boldsymbol{e}$ produced by the discriminator D, we define the following *likelihood function:*

$$lk(\boldsymbol{e}) = \sum_{i=1}^{C} \frac{1}{d+1} \cdot \Phi\Big(\boldsymbol{e}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\Big) \tag{2}$$

where Φ is the Gaussian PDF, $\boldsymbol{\mu}_i$ is the mean vector, and $\boldsymbol{\Sigma}_i$ is the covariance matrix for Gaussian component i, and C is the number of Gaussian components in the mixture. Note that each mixture weight equals $\frac{1}{d+1}$.

When a discrimination between real and fake images is needed, the discriminator first encodes the input image $\boldsymbol{x}$ into the embedding $\boldsymbol{e}$. Then, a likelihood $lk(\boldsymbol{e})$ is calculated for this embedding. $lk(\boldsymbol{e})$ will be interpreted as the probability of $\boldsymbol{e}$ being an embedding of a real image, given the current model.

Generator: The generator G in MD-GAN is a regular neural network decoder, decoding a random noise z from a random distribution $p_{\boldsymbol{z}}$ into an image.

4.3. The Mixture Density GAN Objectives

Denoting the encoding (output of the encoder, also referred to as the embedding) of an image $\boldsymbol{x}$ by discriminator D as $D(\boldsymbol{x})$, we propose MD-GAN's objectives as follows:

$$\begin{aligned} &\min_G \max_D \mathcal{L}(G, D) = \\ &\quad \min_G \max_D \big(\mathbb{E}_{\boldsymbol{x}\sim p_{data}}\big[\log(lk(D(\boldsymbol{x})))\big] \\ &\quad + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\big[\log(\lambda - lk(D(G(\boldsymbol{z}))))\big]\big) \end{aligned} \tag{3}$$

where the likelihood $lk(\boldsymbol{e})$ for the given image embedding $\boldsymbol{e} = D(\boldsymbol{x})$ is as defined in Eq.(2).

We set λ to be the maximum value of the likelihood function lk (in Eq.(2)) in order to have only positive values in the logarithm in Eq.(3) (see also Experimental Setup Section). As discussed in [4], a Gaussian mixture can have more high-probability peaks than its components [4]. Hence, we compute the maximum value of the likelihood function using gradient descent. This is achieved by minimizing $-\log(lk(D(\boldsymbol{x})))$ where $\mathrm{D}(\boldsymbol{x})$ is a feed-forward neural network with only one dense layer and trained on a single and fixed data point. The likelihood value of the data point then converges to the necessary maximum [4].

5. Theoretical Discussion

Recall that our goal is to allow multiple data clusters in the discriminator's embedding space while preserving the discriminative power of the generative adversarial model proposed in [11]. Let p_{gen} represent the distribution of generated images, then the following holds:

[4] http://www.cs.toronto.edu/~miguel/research/GMmodes.html

Proposition 1 (Goodfellow et al. 2014) *For G fixed and $\mathcal{L}(G, D)$ being the discriminator loss as described in Eq.(1), the optimum discriminator*[5] *$D^*_G := \arg\max_D \mathcal{L}(G, D)$ is given by*

$$D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_{gen}(\boldsymbol{x})}$$

Let λ be the maximum of the likelihood function. We then define a new discriminator function $\tilde{D}$ by putting $\tilde{D}(\boldsymbol{x}) := lk(D(\boldsymbol{x}))/\lambda$, which normalizes the likelihood function and yields output values in the unit interval. By applying Proposition 1 we obtain that

$$lk(D^*_G(\boldsymbol{x})) = \lambda \cdot \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{gen}}(\boldsymbol{x})} \tag{4}$$

for an optimum $D^*_G := \arg\max_D \mathcal{L}(G, D)$ of the MD-GAN objective function in Eq.(3) with fixed G. Eq.(4) characterizes the multiple solutions of the discriminator objective. In the case of $\lambda = 1$ and lk being the identity function, the discriminator solution of [11] is obtained, where the value of $D^*_G(\boldsymbol{x})$ is one if and only if $p_{\text{gen}}(\boldsymbol{x}) = 0$. In contrast, in our case, $p_{\text{gen}}(\boldsymbol{x}) = 0$ allows $D^*_G(\boldsymbol{x})$ to be such that the Gaussian likelihood is maximized. Thus, Eq.(4) directly implements our idea of splitting up the optimal solutions to create more meaningful clusters in the embedding space.

As an example, let us consider the one-dimensional case with the likelihood function defined by

$$lk(e) := \frac{1}{2}\Phi(e; -1, \frac{1}{4}) + \frac{1}{2}\Phi(e; 1, \frac{1}{4})$$

Then, for a point x_0 with $p_{\text{gen}}(x_0) = 0$, Eq.(4) reduces to $lk(D^*_G(x_0)) = \lambda$. This implies that the optimal discriminator D^*_G fulfills either $D^*_G(x_0) \simeq 1$ or $D^*_G(x_0) \simeq -1$, which, for suitable covariance of the likelihood function, are near the means of the component Gaussian densities. For x_0 with $p_{\text{gen}}(x_0) \neq 0$ four different global optimal solutions for $D(x_0)$ exist.

The following Theorem 1 follows from [11, Theorem 1] and shows that the optimum of our approach is achieved if and only if the distribution of the real images and the distribution of the generated images match exactly.

Theorem 1 *Under the assumption of an optimal discriminator D^*_G as characterized by Eq.(4), the global optimum of the training criterion $\min_G \mathcal{L}(G, D^*_G)$ in Eq.(3) is achieved if and only if $p_{data} = p_{gen}$.*

Proof Under the assumption of an optimal discriminator D^*_G, characterized by Eq.(4), our objective $\min_G \mathcal{L}(G, D^*_G)$ can be reformulated as

[5] We keep the notation consistent with [11, Proposition 1]

$$
\begin{aligned}
&\min_G \mathcal{L}(G, D_G^*) \\
&\quad = \min_G \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log\left(\lambda \cdot \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{gen}}(\boldsymbol{x})}\right)\right] \\
&\quad + \mathbb{E}_{\boldsymbol{x}\sim p_{\text{gen}}}\left[\log\left(\lambda - \lambda \cdot \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{gen}}(\boldsymbol{x})}\right)\right] \\
&\quad = 2\log(\lambda) \\
&\quad + \min_G \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log\left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{gen}}(\boldsymbol{x})}\right)\right] \\
&\quad + \mathbb{E}_{\boldsymbol{x}\sim p_{\text{gen}}}\left[\log\left(\frac{p_{\text{gen}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{gen}}(\boldsymbol{x})}\right)\right].
\end{aligned}
$$

The optimality of our approach then follows from Theorem 1 in [11]. ■

6. Empirical Results

6.1. Mode collapse evaluation

We use three different datasets and seven baselines to compare with MD-GAN on all the aforementioned datasets: Vanilla GAN [11], Adversarially Learned Inference (ALI) [6], Unrolled GAN [23], VEEGAN [29], DeliGAN [13], InfoGAN [5] and SpectralNormGAN [24].

Stacked-MNIST: This dataset augments the MNIST into a 1000 classes dataset. Gray-scale 28×28 images from MNIST are randomly selected and filled into 3 channels of an RGB image. Since MNIST consists of 10 classes, Stacked MNIST has $10 \times 10 \times 10 = 1000$ classes. To evaluate Stack-MNIST experiments, we use the two measures that were also used in [29, 23]. First, the number of classes that a GAN can generate samples for: a CNN is trained on 1000 classes of Stacked MNIST and then is used to predict the classes of generated samples. The maximum number of classes that GAN can generate based on this classifier after 25000 generated samples is reported. As a second measure, the KL divergence between the label distribution of these 25000 samples and the real label distribution of Stacked MNIST (uniform) is reported.

2D Grid of Gaussians: 25 Gaussian components with a fixed σ of 0.05 and mean vectors placed in a 5×5 grid (Figure 1e) are used to sample training data from. For evaluation, as proposed in [19, 29], we sample the generator for 2500 times and then report the number of Gaussian components discovered. A Gaussian component is considered "discovered" if a sample is generated within an L_2 distance of 3σ from the component's mean. In addition to the number of discovered modes, we report the percentage of these 'high quality' samples with $L_2 \leq 3\sigma$.

2D Ring of Gaussians: 8 Gaussian components with fixed $\sigma = 0.05$ and means placed in a ring are used to sample training data from. The evaluation is analogous to the 2D grid of Gaussians.

6.2. Image quality evaluation

We use four standard datasets to compare MD-GAN to: Vanilla GAN [11], Wasserstein GAN (WGAN) [2], Wasserstein with Gradient Penalty (WGAN-GP) [12], DRAGAN [16], and BEGAN [3]. For these experiments, the *Fréchet Inception Distance (FID)* [14] is used to evaluate the quality of the generated images. The FID results are computed using the provided code and trained inception model from the github repository[6] of the main author of the FID paper.

Since we use results from [21] as our baselines on the 4 real image datasets, we also follow their FID computation procedure, which includes 10k sampling of real and 10k sampling of generated images.

MNIST [18] is a widely-used 28×28 image benchmark dataset consisting of 60k images of hand-written digits in 10 classes.

Fashion-MNIST [32] comprises 28×28 gray-scale images of 70,000 fashion products from 10 categories, with 7,000 images per category.

CIFAR-10 [17] consists of 60,000 32×32 RGB images in 10 classes, with 6,000 images per class.

CelebA [20] is a large-scale dataset with more than 200K images of celebrity faces, each with 40 attribute annotations. As done in [21], we use a 64×64 cropped and centered version of CelebA.

6.3. Experimental Setup

6.3.1 Network architectures

To evaluate the merits of different GAN objectives, it is important to keep the architecture the same. A standard strategy in this context is to use the basic DCGAN architectures [28]. These architectures have a reasonable performance, but are limited in terms of parameters and often used for evaluating new objectives in GANs. Hence, to keep our work comparable to the baselines, for the architectures in our experiments on MNIST, Fashion-MNIST, CelebA and CIFAR10, we use the same architectures as the other baselines used in [21].

To provide comparable results with our baselines for the experiments on *Stacked MNIST*, we use three DCGAN architectures: 1) the architecture used in [29] that has more parameters (shown as **B** in Table 3), 2) the architecture used in [23] with half the parameters in the discriminator (shown as $\mathbf{S}_{\frac{1}{2}}$) and finally 3) the architecture used in [23] with a quarter of parameters in the discriminator (shown as $\mathbf{S}_{\frac{1}{4}}$).

[6] https://github.com/bioinf-jku/TTUR

Again, it is important to note that limiting the number of parameters in a GAN makes mode discovery more difficult and demonstrates the ability of the objective in discovering the modes. The architectures for 2D Ring and Grid of Gaussians are the same as in [29, 23]. The last layer in the discriminator of MD-GAN has the dimensionality of the chosen simplex. All architectures used in our experiments are detailed in the appendix.

6.3.2 Hyper-parameters and reproducibility

The vertices of our simplex have a distance of one from one another. The variances in the diagonals of the covariance matrices in all experiments with real images are set to 0.25, and a 9D simplex (with 10 Gaussian components) is used. The results of a grid search over the number of components are provided in the appendix.

We use the noise distributions and dimensionality, batch sizes and optimization methods reported in [21] for MNIST, F-MNIST, CIFAR-10 and CelebA. The noise distribution and dimensionality, as well as the architecture and optimizations for the 2D datasets and Stacked MNIST are the same as in [29, 23]. The learning rates in mode-discovery experiments for the MD-GAN as well as the implemented baselines are tuned for each method separately. These details are provided in the appendix. To ensure reproducibility, the source code of our method is available online.[7]

6.4. Results

6.4.1 Analysis of mode collapse behavior

The results of our 2D mode collapse experiments are provided in Tables 1 and 2. As can be seen, MD-GAN discovers all the modes and at the same time, manages to generate significantly more high-quality data points compared to all the baselines.

Looking at the results on Stacked MNIST in Table 3, we see that again, MD-GAN outperforms all the baselines using architectures with the same number of parameters. Also MD-GAN achieves the lowest KL divergence of the predicted labels, which shows that not only did MD-GAN discover the most modes, but it also discovered them uniformly. These results are in line with the observations presented earlier in Figure 1d, which showed a more uniform probability landscape in the discriminator that results in a more uniform mode discovery.

6.4.2 Real image data and quality of generated images

The quantitative results of our experiments with the four real image data sets are summarized in Table 4, where we compare the FIDs achieved by MD-GAN with the results of other GAN variants reported in [21]. As can be seen, MD-GAN achieved the lowest FID among all the baselines in all datasets. Samples of the generated images are provided in Figure 4.

[7] https://github.com/eghbalz/mdgan

Table 1: Results of mode collapse experiments on 2D-Grid of 25 Gaussians. †: results taken from [29]. ‡: results taken from [29]. II: our implementation. All results are averages over 5 runs.

method \measure	modes (25)	% hq ($\leq 3 \times std$)
Vanilla [11]†	3.3	0.5
ALI [6]†	15.8	1.6
Unrolled GAN [23]‡	23.6	16
VEEGAN [29]†	24.6	40
DeliGAN [13]II	21±2	74.92±2.74
InfoGAN [5]II	17.2 ±4.95	75.12 ±30.64
SpecNorm [24]II	23.8 ±1.59	90.96 ±4.04
MD-GAN	**25**	**99.36±2.28**

Table 2: Results of mode collapse experiments on 2D-Ring of 8 Gaussians. †: results taken from [29]. ‡: results taken from [29]. II: our implementation. All results are averages over 5 runs.

method \ measure	modes (8)	% hq ($\leq 3 \times std$)
Vanilla [11]†	1	99.3
ALI [6]†	2.8	0.13
Unrolled GAN [23]‡	7.6	35.6
VEEGAN [29]†	8	52.9
DeliGAN [13]II	6.4 ±1.85	98.28±0.4
InfoGAN [5]II	3 ±1.54	98.88 ±1.51
SpecNorm [24]II	6.8 ±1.16	86.64 ±9.76
MD-GAN	**8**	**89.03±3.69**

6.5. Discussion

As can be seen in the mode collapse experiments above, MD-GAN manages to discover all the modes in the data and significantly outperforms all the baselines. From the results on real images, it can also be observed that MD-GAN achieved the lowest FID in all datasets.

We showed in Figure 1 how the probability landscape differs in MD-GAN and provides better training signals for discovering the modes. Figure 3d and 3e showed how the clusters in MD-GAN's discriminator draw the embeddings of real and fake towards them and create a more uniform distribution among components. Additionally, we reported results on several mode discovery datasets and demonstrated that using the mixture density function and

Table 3: Results of mode collapse experiments on Stacked MNIST. All results are averages over 5 runs.
†: results taken from [29]. ‡: results taken from [29]. II: our implementation. B: Big DCGAN architecture, $S_{\frac{1}{2}}$: Small DCGAN architecture with half disc. parameters. $S_{\frac{1}{4}}$: Small DCGAN architecture with quarter disc. parameters.

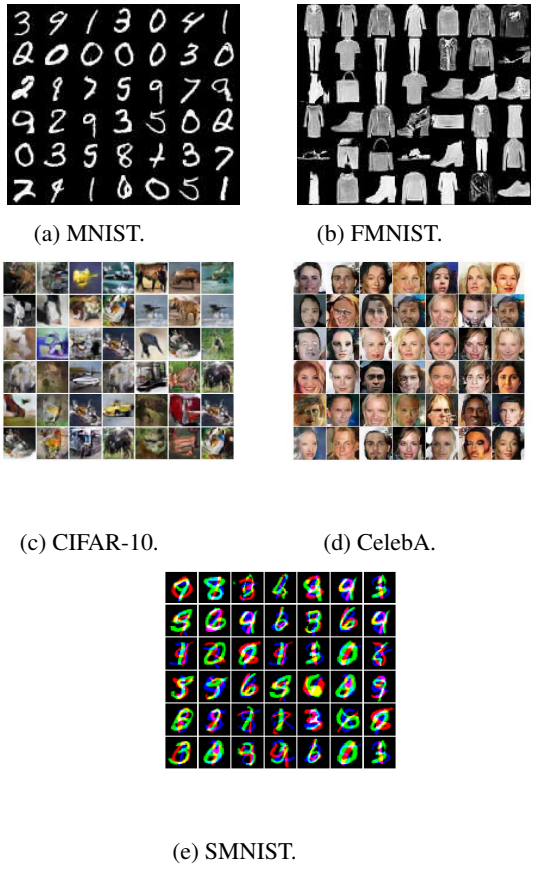
method \ measure	arch.	modes (1000)	KL (labels)
Vanilla [11]†	B	99	3.4
ALI [6] †	B	16	5.4
VEEGAN [29] †	B	150	2.95
Unrl GAN [23] †	B	48.7	4.32
MD-GAN	B	**1000**	**0.046±0.001**
Unrl GAN [23] ‡	$S_{\frac{1}{2}}$	817.4 ±37.91	1.43 ±0.12
Deli [13] II	$S_{\frac{1}{2}}$	125.60 ±144.65	3.77 ±1.97
InfoGAN [5] II	$S_{\frac{1}{2}}$	796.40 ±76.51	0.90±0.11
SpecNorm [24] II	$S_{\frac{1}{2}}$	678.80±270.98	1.45 ±.81
MD-GAN	$S_{\frac{1}{2}}$	**921.0±3.0**	**0.80±0.06**
Unrl GAN [23] ‡	$S_{\frac{1}{4}}$	327.2 ±74.67	4.66 ±0.46
Deli [13] II	$S_{\frac{1}{4}}$	158.6 ±84.21	3.22 ±0.96
InfoGAN [5] II	$S_{\frac{1}{4}}$	237.20 ±284.38	2.87 ±0.94
SpecNorm [24] II	$S_{\frac{1}{4}}$	354.60 ±248.0	2.44 ±1.10
MD-GAN	$S_{\frac{1}{4}}$	**696.0±10.0**	**1.32±0.015**

Table 4: FIDs on different datasets from different methods. †: Results taken from [21] which are best FIDs obtained in a large-scale hyper-parameter search for each data set. Lower FID values represent higher quality for generated images.

method \ db	MNT	FMNT	CFR	Clb
Real images*dagger*	1.2	2.6	5.1	2.2
Vanilla [11]†	6.7	26.6	58.6	58.0
Wasserstein [2]†	6.8	18.0	55.9	42.9
Wasserstein GP t[12]†	8.9	20.6	52.9	26.8
DRAGAN [16]†	7.7	26.0	68.5	41.4
BEGAN [3]†	12.3	33.2	71.4	38.1
MD-GAN	**6.29**	**11.79**	**36.80**	**24.51**

the clusters formed, MD-GAN outperformed several baselines.

To show that this mode discovery property does not negatively effect the quality of the generated images, using standard architectures and four benchmark datasets, we showed that MD-GAN achieved the lowest FID among all the baselines, hence is capable of generating high quality images.



(a) MNIST. (b) FMNIST.

(c) CIFAR-10. (d) CelebA.

(e) SMNIST.

Figure 4: Randomly chosen samples from MD-GAN.

7. Conclusion

We have proposed the Mixture Density GAN, which succeeds in alleviating the mode collapse problem and generating high-quality images by allowing the Discriminator to form separable clusters in its embedding space, which in turn leads the Generator to generate data with more variety. We analysed the optimum discriminator and showed that it is achieved when the generated and the real distribution match exactly. We demonstrated the ability of MD-GAN to deal with mode collapse and generate realistic images using seven benchmark datasets. We demonstrated that MD-GAN achieved the best results of all compared baselines on all datasets, in terms of FID and the number of discovered modes with high-quality generated data.

8. Acknowledgement

We would like to thank Susanne Saminger-Platz and Martin Heusel from Johannes Kepler University of Linz for helpful discussions. This work has been partly funded by the Austrian COMET Center SCCH. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, 2017. 1

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2, 6, 8

[3] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 2, 6, 8

[4] Miguel A. Carreira-Perpinan. Mode-finding for mixtures of gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1318–1323, 2000. 5

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016. 2, 6, 7, 8

[6] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016. 6, 7, 8

[7] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *International Conference on Learning Representations*, 2017. 2

[8] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: GANs do not need to decrease adivergence at every step. *arXiv preprint arXiv:1710.08446*, 2017. 2

[9] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[10] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 1, 2

[11] 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*, 2014. 1, 2, 5, 6, 7, 8

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. *Advances in Neural Information Processing Systems*, 2017. 2, 6, 8

[13] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 6, 7, 8

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017. 1, 6

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014. 2

[16] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017. 6, 8

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 6

[18] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 6

[19] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. 2017. 6

[20] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, 2015. 6

[21] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? a large-scale study. *Advances in Neural Information Processing Systems*, 2018. 6, 7, 8

[22] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. *arXiv preprint arXiv:1705.10461*, 2017. 1

[23] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *International Conference on Learning Representations*, 2017. 2, 6, 7, 8

[24] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*, 2018. 2, 6, 7, 8

[25] G. K. Monatsh. Mehrdimensionale geometrie. *Monatshefte für Mathematik und Physik*, 1907. 1

[26] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. *arXiv preprint arXiv:1702.08398*, 2017. 2

[27] Yunus Saatchi and Andrew Gordon Wilson. Bayesian GAN. In *Advances in Neural Information Processing Systems*, 2017. 2

[28] 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*, 2016. 1, 6

[29] Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. VEEGAN: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, 2017. 2, 6, 7, 8

[30] Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, Martin Heusel, Hubert Ramsauer, and Sepp Hochreiter. Coulomb gans: Provably optimal nash equilibria via potential fields. *International Conference on Learning Representations*, 2018. 2

[31] Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016. 2

[32] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 6

[33] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central

moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2017. 2

[34] Werner Zellinger, Bernhard A. Moser, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Robust unsupervised domain adaptation for neural networks via moment alignment. *Information Sciences*, 483:174–191, May 2019. 2

[35] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. 2