MULTI-MODAL GENERATIVE ADVERSARIAL NETWORKS FOR DIVERSE DATASETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Adversarial Networks (GANs) have been shown to produce realistically looking synthetic images with remarkable success, yet their performance seems less impressive when the training set is highly diverse. In order to provide a better fit to the target data distribution when the dataset includes many different classes, we propose a variant of the basic GAN model, a Multi-Modal Gaussian-Mixture GAN (GM-GAN), where the probability distribution over the latent space is a mixture of Gaussians. We also propose a supervised variant which is capable of conditional sample synthesis. In order to evaluate the model's performance, we propose a new scoring method which separately takes into account two (typically conflicting) measures - diversity vs. quality of the generated data. Through a series of experiments, using both synthetic and real-world datasets, we quantitatively show that GM-GANs outperform baselines, both when evaluated using the commonly used Inception Score (Salimans et al., 2016), and when evaluated using our own alternative scoring method. In addition, we qualitatively demonstrate how the *unsupervised* variant of GM-GAN tends to map latent vectors sampled from different Gaussians in the latent space to samples of different classes in the data space. We show how this phenomenon can be exploited for the task of unsupervised clustering, and provide quantitative evaluation showing the superiority of our method for the unsupervised clustering of image datasets. Finally, we demonstrate a feature which further sets our model apart from other GAN models: the option to control the quality-diversity trade-off by altering, post-training, the probability distribution of the latent space. This allows one to sample higher quality and lower diversity samples, or vice versa, according to one's needs.

1 INTRODUCTION

Generative models have long been an important and active field of research in machine-learning. Generative Adversarial Networks (Goodfellow et al., 2014) include a family of methods for learning generative models where the computational approach is based on game theory. The goal of a GAN is to learn a Generator (G) capable of generating samples from the data distribution ($p_{\mathcal{X}}$), by converting latent vectors from a lower-dimension latent space (Z) to samples in a higher-dimension data space ($\mathcal{X}$). Usually, latent vectors are sampled from Z using the uniform or the normal distribution.

In order to train G, a Discriminator (D) is trained to distinguish real training samples from fake samples generated by G. Thus D returns a value $D(\mathbf{x}) \in [0, 1]$ which can be interpreted as the probability that the input sample ($\mathbf{x}$) is a real sample from the data distribution. In this configuration, G is trained to obstruct D by generating samples which better resemble the real training samples, while D is continuously trained to tell apart real from fake samples. Crucially, G has no direct access to real samples from the training set, as it learns solely through its interaction with D. Both D and G are implemented by deep differentiable networks, typically consisting of multiple convolutional and fully-connected layers. They may be alternately trained using Stochastic Gradient Descent.

In the short period of time since the introduction of the GAN model, many different enhancement methods and training variants have been suggested to improve their performance (see brief review below). Despite these efforts, often a large proportion of the generated samples is, arguably, not satisfactorily realistic. In some cases the generated sample does not resemble any of the real samples

from the training set, and human observers find it difficult to classify synthetically generated samples to one of the classes which compose the training set (see illustration in Figure 1).

Figure 1: Images generated by different GANs trained on MNIST (top row), CelebA (middle row) and STL-10 (bottom row). Red square mark images of, arguably, low quality (best seen in color).

This problem worsens with the increased complexity of the training set, and specifically when the training set is characterized by large *inter-class* and *intra-class* diversity. In this work we focus on this problem, aiming to improve the performance of GANs when the training dataset has large *inter-class* and *intra-class* diversity.

Related Work. In an attempt to improve the performance of the original GAN model, many variants and extensions have been proposed in the past few years. These include architectural changes to G and D as in Radford et al. (2015), modifications to the loss function as in Mao et al. (2016); Gulrajani et al. (2017), or the introduction of supervision into the training setting as in Mirza & Osindero (2014); Odena et al. (2016). Another branch of related work, which is perhaps more closely related to our work, involves the learning of a meaningfully structured latent space. Thus Info-GAN (Chen et al., 2016) decomposes the input noise into an incompressible source and a "latent code", Adversarial Auto-Encoders (Makhzani et al., 2015) employ GANs to perform variational inference, and Larsen et al. (2015) combine a Variational Auto-Encoder with a Generative Adversarial Network (see Appendix A for a more comprehensive description).

Our Approach. Although modifications to the structure of the latent space have been investigated before as described above, the significance of the probability distribution used for sampling latent vectors was rarely investigated. A common practice today is to use a standard normal (e.g. $N(0, I)$) or uniform (e.g. $U[0, 1]$) probability distribution when sampling latent vectors from the latent space. We wish to challenge this common practice, and investigate the beneficial effects of modifying the distribution used to sample latent vectors in accordance with properties of the target dataset.

Specifically, many datasets, especially those of natural images, are quite diverse, with high inter-class and intra-class variability. At the same time, the representations of these datasets usually span high dimensional spaces, which naturally makes them very sparse. Intuitively, this implies that the underlying data distribution, which we try to learn using a GAN, is also sparse, i.e. it mostly consists of low-density areas with relatively few areas of high-density.

We propose to incorporate this prior-knowledge into the model, by sampling latent vectors using a multi-modal probability distribution which better matches these characteristics of the data space distribution. It is important to emphasize that this architectural modification is orthogonal to, and can be used in conjunction with, other architectural improvements such as those reviewed above (see for instance Figure 9 in Appendix D.) Supervision can be incorporated into this model by adding correspondence (not necessarily injective) between labels and mixture components.

The rest of this paper is organized as follows: In Section 2 we describe the family of GM-GAN models. In Section 3 we offer an alternative method which focuses on measuring the trade-off between sample quality and diversity of generative models. In Section 4 we empirically evaluate our proposed model using various diverse datasets, showing that GM-GANs outperform the corresponding baseline methods with uni-modal distribution in the latent space. In Section 5 we describe a method for clustering datasets using GM-GANs, and provide qualitative and quantitative evaluation using various datasets of real images.

2 MULTI-MODAL GAUSSIAN-MIXTURE GAN (GM-GAN)

Unsupervised GM-GAN. The target function which we usually optimize for, when training a GAN composed of Generator G and Discriminator D, can be written as follows:

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\mathcal{X}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_{\mathcal{Z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{1}$$

Above $p_{\mathcal{X}}$ denotes the distribution of real training samples, and $p_{\mathcal{Z}}$ denotes some d-dimensional prior distribution which is used as a source of stochasticity for the Generator. The corresponding loss functions of G and D can be written as follows:

$$L(G) = -\mathop{\mathbb{E}}_{\mathbf{z}\sim p_{\mathcal{Z}}(\mathbf{z})}[\log D(G(\mathbf{z}))] \tag{2}$$

$$L(D) = -\mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\mathcal{X}}(\mathbf{x})}[\log D(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{z}\sim p_{Z}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{3}$$

Usually, a multivariate uniform distribution (e.g. $U[-1,1]^d$), or a multivariate normal distribution (e.g. $N(0, I_{d\times d})$), are used as substitute for $p_{\mathcal{Z}}$ when training GANs. In our proposed model, we optimize the same target function as in (1), but instead of using a unimodal random distribution for the prior $p_{\mathcal{Z}}$, we propose to use a multi-modal distribution which can better suit the inherent multi-modality of the real training data distribution, $p_{\mathcal{X}}$.

Specifically, we propose to use a mixture of Gaussians as a multi-modal prior distribution where $p_{\mathcal{Z}}(\mathbf{z}) = \sum_{k=1}^{K} \alpha_k * p_k(\mathbf{z})$. Here K denotes the number of Gaussians in the mixture, $\{\alpha_k\}_{k=1}^{K}$ denotes a categorical random variable, and $p_k(\mathbf{z})$ denotes the multivariate Normal distribution $N(\mu_k, \Sigma_k)$, defined by the mean vector μ_k, and the covariance matrix Σ_k. In the absence of prior knowledge we assume a uniform mixture of Gaussians, that is, $\forall k \in [K]\ \alpha_k = \frac{1}{K}$.

The parameters $[\mu_k, \Sigma_k]$ of each Gaussian in the mixture can be fixed, or learned along with the parameters of the GAN in an "end-to-end" fashion to allow for a more flexible model. We investigated two corresponding variants of the new model - one (*Static*) where the the parameters of the Gaussians mixture are fixed throughout the model's training process, and one (*Dynamic*) where these parameters are allowed to change during the training process in order to potentially converge to a better solution. The details of these two variants are given in Appendix B.

Supervised GM-GAN. In the supervised setting, we change the GM-GAN's discriminator so that instead of returning a single scalar, it returns a vector $\mathbf{o} \in \mathbb{R}^N$ where N is the number of classes in the dataset. Each element o_i in this vector lies in $[0, 1]$. The Generator's purpose in this setting is, given a latent vector $\mathbf{z}$ sampled from the k'th Gaussian in the mixture, to generate a sample which will be classified by the discriminator as a real sample from class $f(k)$, where $f : [K] \to [N]$ is a discrete function mapping identity of Gaussians to class labels. When $K = N$, f is bijective and the model is trained to map each Gaussian to a unique class in the data space. When $K > N$ f is surjective, and multiple Gaussians can be mapped to the same class in order to model high diversity classes. When $K < N$ f is injective, and multiple classes can be grouped together by mapping to the same Gaussian.

We modify both loss functions of G and D to accommodate the class labels. The modified loss functions become the following:

$$L(G) = \mathop{-\mathbb{E}}_{\mathbf{z}\sim p_{Z}(\mathbf{z})}\left[\log D(G(\mathbf{z}))_{f(y(\mathbf{z}))} + \sum_{\substack{m=1 \\ m\neq f(y(\mathbf{z}))}}^{N} \log(1 - D(G(\mathbf{z}))_m)\right]$$

$$L(D) = \mathop{-\mathbb{E}}_{\mathbf{z}\sim p_{Z}(\mathbf{z})}\left[\sum_{m=1}^{N} \log(1 - D(G(\mathbf{z}))_m)\right] - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\mathcal{X}}(\mathbf{x})}\left[\log D(\mathbf{x})_{y(\mathbf{x})} + \sum_{\substack{m=1 \\ m\neq f(y(\mathbf{z}))}}^{N} \log(1 - D(\mathbf{x})_m)\right]$$

where $y(\mathbf{x})$ denotes the class label of sample $\mathbf{x}$, and $y(\mathbf{z})$ denotes the index of the Gaussian from which the latent vector $\mathbf{z}$ has been sampled. The training procedure for GM-GANs is fully described in Algorithm 1.

Algorithm 1 Training the **GM-GAN** model.

Require:
K - the number of Gaussians in the mixture.
d - the dimension of the latent space (Z).
c - the range from which the Gaussians' means are sampled.
σ - scaling factor for the covariance matrices.
$iters$ - the number of training iterations.
b_D - the batch size for training the discriminator.
b_G - the batch size for training the Generator.
γ - the learning rate.
f - a mapping from Gaussian indices to class indices (in a supervised setting only).

for $k = 1...K$ **do**
  Sample $\mu_k \sim U[-c, c]^d$ ▷ init the mean vector of Gaussian k
  $\Sigma_k \leftarrow \sigma * I_{dxd}$ ▷ init the covariance matrix of Gaussian k
for $i = 1...iters$ **do**
  for $j = 1...b_D$ **do**
    Sample $\mathbf{x_j} \sim p_{\mathcal{X}}$ ▷ get a real sample from the training-set.
    Sample $k \sim Categ(\frac{1}{K}, ..., \frac{1}{K})$ ▷ sample a Gaussian index.
    Sample $\mathbf{z_j} \sim N(\mu_k, \Sigma_k)$ ▷ sample from the k'th Gaussian
    $\hat{\mathbf{x}}_\mathbf{j} \leftarrow G(\mathbf{z_j})$ ▷ generate a fake sample using the Generator
    if $supervised$ **then** ▷ compute the loss of D
      $L_{real}(D)^{(j)} \leftarrow -\log D(\mathbf{x_j})_{y(\mathbf{x_j})} - \sum_{m=1, m\neq y(\mathbf{x_j})}^{N} \log(1 - D(\mathbf{x_j})_m)$
      $L_{fake}(D)^{(j)} \leftarrow -\sum_{m=1}^{N} \log(1 - D(\hat{\mathbf{x}}_\mathbf{j})_m)$
    else
      $L_{real}(D)^{(j)} \leftarrow -\log D(\mathbf{x_j})$
      $L_{fake}(D)^{(j)} \leftarrow -\log(1 - D(\hat{\mathbf{x}}_\mathbf{j}))$
  $L(D) \leftarrow \frac{1}{2*b_D} \sum_{j=1}^{b_D} L_{real}(D)^{(j)} + L_{fake}(D)^{(j)}$
  $\theta_D \leftarrow Adam(\nabla_{\theta_D}, L(D), \theta_D, \gamma)$ ▷ update the weights of D by a single GD step.
  for $j = 1...b_G$ **do**
    Sample $k \sim Categ(\frac{1}{K}, ..., \frac{1}{K})$ ▷ sample a Gaussian index.
    Sample $\mathbf{z_j} \sim N(\mu_k, \Sigma_k)$ ▷ sample from the k'th Gaussian
    $\hat{\mathbf{x}}_\mathbf{j} \leftarrow G(\mathbf{z_j})$ ▷ generate a fake sample using the Generator
    if $supervised$ **then** ▷ compute the loss of G
      $L(G)^{(j)} \leftarrow -\log D(\hat{\mathbf{x}}_\mathbf{j})_{f(y(\mathbf{z}_j))} - \sum_{m=1, m\neq f(y(\mathbf{z_j}))}^{N} \log(1 - D(\hat{\mathbf{x}}_\mathbf{j})_m)$
    else
      $L(G)^{(j)} \leftarrow -\log D(\hat{\mathbf{x}}_\mathbf{j})$
  $L(G) \leftarrow \frac{1}{b_G} \sum_{j=1}^{b_G} L_(G)^{(j)}$
  $\theta_G \leftarrow Adam(\nabla_{\theta_G}, L(G), \theta_G, \gamma)$ ▷ update the weights of G by a single GD step.

3 GAN Evaluation Score: the Quality-Diversity Trade-off

We describe next a new scoring method for GANs, which is arguably better suited for the task than the commonly used Inception Score proposed in Salimans et al. (2016) and described in Appendix C. The Inception score has been used extensively over the last few years, but it has a number of drawbacks: (i) It is limited to the evaluation of GANS which are trained to generate natural images. (ii) It only measures the samples' *inter-class* diversity, ignoring the *intra-class* diversity of samples. (iii) It combines together a measure of quality and a measure of diversity into a single score. (iv) Different scores can be achieved by the same GAN, when sampling latent vectors with different parameters of the source probability distribution (e.g. σ, see Figure 6 in Appendix C).

The Quality-Diversity trade-off: The quality of sample $\mathbf{x} \in \mathcal{X}$ may be measured by its probability $p_{\mathcal{X}}(\mathbf{x})$, which implies that samples drawn from dense areas in the source domain (i.e. close to the modes of the distribution) are mapped to high quality samples in the target domain, and vice

versa. Therefore, we can increase the expected quality of generated samples in the target domain by sampling with high probability from dense areas of the source domain, and with low probability from sparse areas of the source domain. While increasing the expected quality of generated samples, this procedure also reduces the sample diversity[1]. This fundamental trade-off between quality and diversity must be quantified if we want to compare the performance of different GAN models.

Next we propose a new scoring method for either supervised or unsupervised GAN models, which is useful for multi-class image datasets, evaluating the trade-off between samples' quality and diversity. This scoring method also relies on a pre-trained classifier $\mathbf{C}$, but unlike the Inception Score, this classifier is trained on the *same* training set on which the GAN is trained on. Classifier $\mathbf{C}$ is used to measure both the quality and the diversity of generated samples, as explained below.

Quality Score To measure the quality of generated sample $\mathbf{x}$, we propose to use the intermediate representation of $\mathbf{x}$ in the pre-trained classifier $\mathbf{C}$, and to measure the Euclidean distance from this representation to its nearest-neighbor in the training set. More specifically, if $\mathbf{C}_l(\mathbf{x})$ denotes the activation levels in the pre-trained classifier's layer l given sample $\mathbf{x}$, then the quality score $q(\mathbf{x})$, for sample $\mathbf{x}$ and a set of samples X, is defined as follows:

$$q(\mathbf{x}) = 1 - \frac{\exp\left(||\mathbf{C}_l(\mathbf{x}) - \mathbf{C}_l(NN(\mathbf{x}))||_2\right)}{\exp\left(||\mathbf{C}_l(\mathbf{x}) - \mathbf{C}_l(NN(\mathbf{x}))||_2\right) + a}, \qquad q(X) = \sum_{\mathbf{x}\in X} \frac{1}{|X|} q(\mathbf{x}) \tag{4}$$

Above $NN(\mathbf{x})$ denotes the nearest-neighbor of $\mathbf{x}$ in the training set, defined as $NN(\mathbf{x}) = \arg\min_{\mathbf{x}'\in X} ||\mathbf{C}_l(\mathbf{x}) - \mathbf{C}_l(\mathbf{x}')||_2$, and a denotes a positive constant.

Diversity Score To measure the diversity of generated samples, we take into account both the inter-class, and the intra-class diversity. We measure *intra-class* diversity by the average (negative) MS-SSIM metric (Wang et al., 2003) between all pairs of generated images in a given set of generated images X:

$$d_{intra}(X) = 1 - \frac{1}{|X|^2} \sum_{(\mathbf{x},\mathbf{x}')\in X\times X} \text{MS-SSIM}(\mathbf{x}, \mathbf{x}') \tag{5}$$

For *inter-class* diversity, we use the pre-trained classifier to classify the set of generated images, such that for each sampled image $\mathbf{x}$, we have a classification prediction in the form of a one-hot vector $c(\mathbf{x})$. We then measure the entropy of the average one-hot classification prediction vector to evaluate the diversity between classes in the samples set:

$$d_{inter}(X) = \frac{1}{\log(N)} H\left(\frac{1}{|X|}\sum_{\mathbf{x}\in X} c(\mathbf{x})\right) \tag{6}$$

Finally, the *diversity score* is defined as the geometric mean of (5) and (6):

$$d(X) = \sqrt{d_{intra}(X) * d_{inter}(X)} \tag{7}$$

4 Empirical Evaluation

In this section we empirically evaluate the benefits of our proposed approach, comparing the performance of GM-GAN with the corresponding baselines. Thus we compare the performance of the unsupervised GM-GAN model to that of the originally proposed GAN (Goodfellow et al., 2014), and the performance of our proposed supervised GM-GAN model to that of AC-GAN (Odena et al., 2016). In both cases, the baseline models' latent space probability distribution is standard normal, i.e. $\mathbf{z} \sim N(0, I)$. The network architectures and hyper-parameters used for training the GM-GAN models are similar to those used for training the baseline models. For the most part we used the *Static* GM-GAN with default values $d = 100$, $c = 0.1$, $\sigma = 0.15$, $B_D = 64$, $b_G = 128$, $\gamma = 0.0002$; K

[1] In our experiments, we were able to control this quality-diversity trade-off by modifying the probability distribution which is used for sampling latent vectors from the latent space Z (see Figures 3, 4). We further elaborate on this matter in Section 4.3.

and $iters$ varied in the different experiments. The *Dynamic* GM-GAN model was only used in the experiments summarized in Figure 5.

In the following experiments we evaluate the different models on the 6 datasets listed in Table 1. In all cases, the only pre-processing made on the training images is a transformation of pixel-values to the range of $[-1, 1]$.

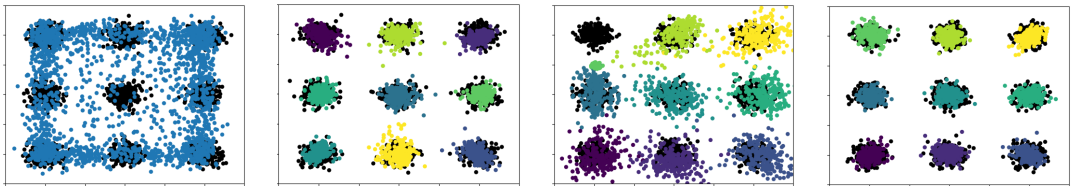
Dataset Name	**Description**	**Classes Number**	**Samples Dimension**	**Train Samples**	**Test Samples**
Toy-Dataset	Points sampled from different Gaussians in the 2-D Euclidean space.	9	2	5,000	-
MNIST	Images of handwritten digits.	10	28x28x1	60,000	10,000
Fashion-MNIST	Images of clothing articles.	10	28x28x1	60,000	10,000
CIFAR-10	Natural images.	10	32x32x3	50,000	10,000
STL-10	Natural images.	10	96x96x3	5,000	8,000
Synthetic Traffic Signs	Synthetic images of street traffic signs.	43	40x40x3	100,000	-

Table 1: Details of the different datasets used in the empirical evaluation: a Toy-Dataset specially created (see details in Section 4.1), MNIST (LeCun & Cortes, 2010), Fashion-MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al.), STL-10 (Adam Coates, 2011) and the Synthetic Traffic Signs Dataset (Moiseev et al., 2013).

4.1 TOY-DATASET

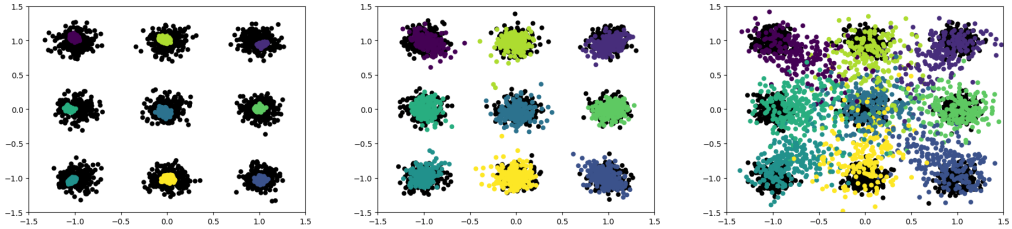
We first compare the performance of our proposed GM-GAN models to the aforementioned baseline models using a toy dataset, which has been created in order to gain more intuition regarding the properties of the GM-GAN model. The dataset consists of 5,000 training samples, where each training sample $\mathbf{x}$ is a point in $\mathbb{R}^2$ drawn from a homogeneous mixture of K Gaussians, i.e., $\forall \mathbf{x}$ $p(\mathbf{x}) = \sum_{k=1}^{K} \frac{1}{K} p_k(\mathbf{x})$ where $p_k(\mathbf{x}) \sim N(\mu_k, \Sigma_k)$. In our experiments we used $K = 9$ Gaussians, $\forall k \in [K]$ $\Sigma_k = 0.1 * I$ and $\mu = \{-1, 0, 1\} \times \{-1, 0, 1\}$. We labeled each sample with the identity of the Gaussian from which it was sampled.

We trained two instances of the GM-GAN model, one supervised using the labels of the samples, and one unsupervised oblivious of these labels. In both cases, we used $K = 9$ Gaussians in the mixture from which latent vectors were sampled. Figure 2 presents samples generated by the baseline models (GAN, AC-GAN) and samples generated by our proposed GM-GAN models (both unsupervised and supervised variants). It is clear that both variants of the GM-GAN generate samples with a higher likelihood, which matches the original distribution more closely as compared to the baseline methods. The trade-off between quality and diversity is illustrated in Figure 3, showing high quality and low diversity for $\sigma = 0.25$, and vice versa for $\sigma = 2.0$ (see Section 4.3).



(a) (b) (c) (d)

Figure 2: Samples from the toy-dataset along with samples generated from: (a) GAN, (b) unsupervised GM-GAN, (c) AC-GAN, (d) supervised GM-GAN. Samples from the training set are drawn in black, and samples generated by the trained Generators are drawn in color. In (b) and (d), the color of each sample represents the Gaussian from which the corresponding latent vector is sampled.



$\sigma = 0.25$ $\sigma = 1.0$ $\sigma = 2.0$

Figure 3: Samples from the toy-dataset along with samples generated by an unsupervised GM-GAN using different σ values for sampling latent vectors from the latent space Z ($\sigma = 1.0$ was used during training). The color code is the same as in Figure 2b,d.

An intriguing observation is that the GM-GAN's Generator is capable, without any supervision, of mapping each Gaussian in the latent space to samples in the data-space which are almost perfectly aligned with a single Gaussian. We also observe this phenomenon when training unsupervised GM-GAN on the MNIST and Fashion-MNIST datasets. In Section 5 we exploit this phenomenon to achieve a clustering algorithm. Finally, we note that the GM-GAN models converge considerably faster than the classical GAN model, see Figure 7 in Appendix D.

4.2 Real Datasets, Inception Scores

We next turn to evaluate our proposed models when trained on more complex datasets. We start by using the customary Inception Score (Salimans et al., 2016) to evaluate and compare the performance of the difference models, the two GM-GAN models and the baseline models (GAN and AC-GAN). We trained the models on two real datasets with 10 classes each, CIFAR-10 and STL-10 (see Table 1). Each variant of the GM-GAN model was trained multiple times, each time using a different number (K) of Gaussians in the latent space probability distribution. In addition, each model was trained 10 times using different initial parameter values. We then computed for each model its mean Inception Score and the corresponding standard error. The results for the two unsupervised and two supervised models are presented in Table 2. In all cases, the two GM-GAN models achieve higher scores when compared to the respective baseline model. The biggest improvement is achieved in the supervised case, where the supervised variant of the GM-GAN model outperforms AC-GAN by a large margin.

CIFAR-10

Model (unsupervised)	**Score**
GAN	5.71 ($\pm$0.06)
GM-GAN ($K = 10$)	5.92 ($\pm$0.07)
GM-GAN ($K = 20$)	5.91 ($\pm$0.05)
GM-GAN ($K = 30$)	**5.98** ($\pm$**0.05**)

Model (supervised)	**Score**
AC-GAN	6.23 ($\pm$0.07)
GM-GAN ($K = 10$)	**6.84** ($\pm$**0.03**)
GM-GAN ($K = 20$)	6.81 ($\pm$0.04)
GM-GAN ($K = 30$)	6.83 ($\pm$0.02)

STL-10

Model (unsupervised)	**Score**
GAN	6.80 ($\pm$0.07)
GM-GAN ($K = 10$)	**7.06** ($\pm$**0.11**)
GM-GAN ($K = 20$)	6.58 ($\pm$0.16)
GM-GAN ($K = 30$)	7.03 ($\pm$0.10)

Model (supervised)	**Score**
AC-GAN	7.45 ($\pm$0.10)
GM-GAN ($K = 10$)	**8.32** ($\pm$**0.06**)
GM-GAN ($K = 20$)	8.16 ($\pm$0.05)
GM-GAN ($K = 30$)	8.08 ($\pm$0.07)

Table 2: Inception Scores for different GM-GAN models vs. baselines trained on 2 datasets.

4.3 Trade-off between Quality and Diversity

As discussed in Section 3, the Inception Score is not sufficient, on its own, to illustrate the trade-off between the quality and the diversity of samples which a certain GAN is capable of generating. In our experiments, we control the quality-diversity trade-off by varying, after the model's training,

the probability distribution which is used to sample latent vectors from the latent space. We do so by multiplying the covariance matrix of each Gaussian by a scaling factor σ. Specifically, when using the baseline models we sample $\mathbf{z} \sim N(0, \sigma * I)$, and when using the GM-GAN models we sample $\mathbf{z}|k \sim N(\mu_k, \sigma * \Sigma_k)$, $k \sim Categ(\frac{1}{K}, ..., \frac{1}{K})$. Thus, when $\sigma < 1$, latent vectors are sampled with lower variance around the modes of the latent space probability distribution, and therefore the respective samples generated by the Generator are of higher expected quality, but lower expected diversity. The opposite happens when $\sigma > 1$, where the respective samples generated by the Generator are of lower expected quality, but higher expected diversity. Figures 3, 4 demonstrate qualitatively the quality-diversity trade-off offered by GM-GANs when trained on the Toy and MNIST datasets.

$\sigma = 0.1$ $\sigma = 1.0$ $\sigma = 2.0$

Figure 4: Samples taken from a GM-GAN trained on the MNIST dataset. In each panel, latent vectors samples are drawn using different σ values ($\sigma = 1.0$ was used during training). Clearly The quality of samples decreases, and the diversity increases, as σ grows.

Next, we evaluated each model by calculating our proposed Quality Score from Eq. (4), and the Combined Diversity Score from Eq. (7), for each $\sigma \in \{0.5, 0.6, ..., 1.9, 2.0\}$. Each model was trained 10 times using different initial parameter values. We computed for each model its mean Quality and mean Combined Diversity scores and the corresponding standard errors. The Quality and Diversity Scores of the GM-GAN and baseline models, when trained on the CIFAR-10 and STL-10 datasets, are presented in Figure 5 (see additional datasets in Figure 8 in Appendix D).

In some cases (e.g. supervised training on CIFAR-10 and STL-10) the results show a clear advantage for our proposed model as compared to the baseline, as both the quality and the diversity scores of GM-GAN surpass those of AC-GAN, for *all* values of σ. In other cases (e.g. unsupervised training on CIFAR-10 and STL-10), the results show that for the lower-end range of σ, the baseline model generates images of higher quality but dramatically lower diversity, as compared to our proposed model. In accordance, when visually examining the samples generated by the two models, we notice that most samples generated by the baseline model belong to a single class, while samples generated by our model are much more diverse and are scattered uniformly among the different classes. In all cases, the charts predictably show an ascending Quality Score, and a descending Combined Diversity Score, as σ is increased.

5 Unsupervised Clustering using GM-GANs

Throughout our experiments, we noticed an intriguing phenomenon where the *unsupervised* variant of GM-GAN tends to map latent vectors sampled from different Gaussians in the latent space to samples of different classes in the data space. Specifically, each Gaussian in the latent space is usually mapped, by the GM-GAN's Generator, to a single class in the data space. Figure 10 in Appendix D demonstrates this phenomenon using different datasets. The fact that the latent space in our proposed model is sparse, while being composed of multiple Gaussians with little overlap, may be the underlying reason for this phenomenon. In this section we exploit this observation to develop a new clustering algorithm, and provide quantitative evaluation of the proposed method.

Clustering Method In the proposed method we first train an unsupervised GM-GAN which receives as input the data points without corresponding labels. K, the number of Gaussians forming

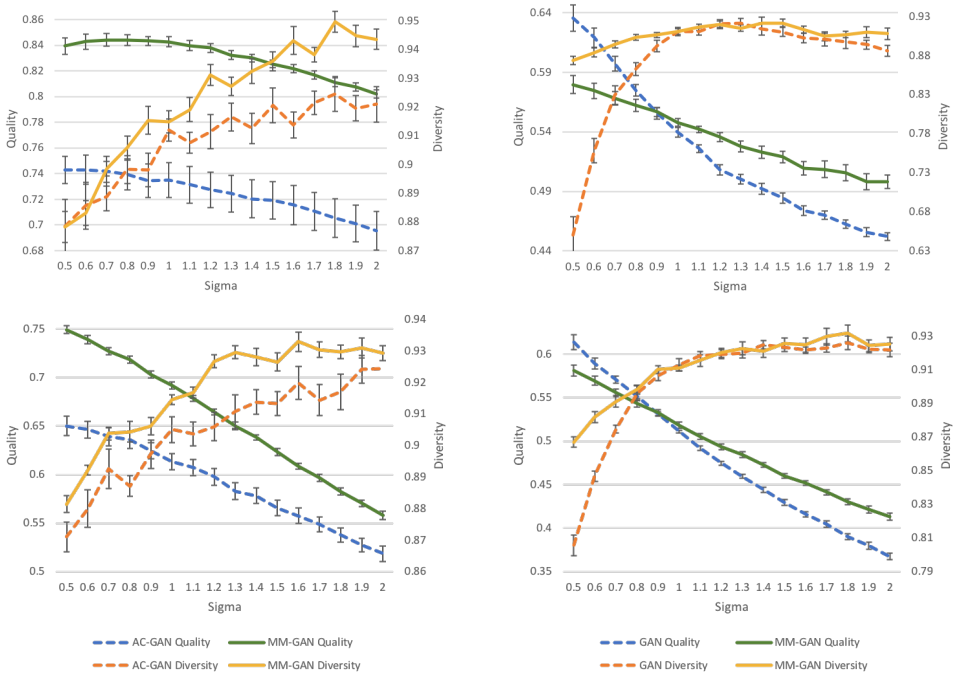


Figure 5: Quality and Diversity scores of GM-GANs vs. baselines trained on 2 datasets: **CIFAR-10** (top) and **STL-10** (bottom). Left column: AC-GANs vs. supervised GM-GANs. Right column: GANs vs. unsupervised GM-GANs. Error bars show the standard error of the mean.

the latent space, is set to equal the number of clusters in the intended partition. Using the trained GM-GAN model, we sample from each Gaussian $k \in [K]$ a set of M latent vectors, from which we generate a set of M synthetic samples $\widetilde{X}_k = \left\{\widetilde{\mathbf{x}}_k^{(i)}\right\}_{i\in[M]}$. We then train a K-way multi-class classifier on the unified set of samples from all Gaussians $\bigcup_{k\in[K]} \widetilde{X}_k$, where the label of sample $\widetilde{\mathbf{x}} \in \widetilde{X}_k$ is set to k, i.e. the index of the Gaussian from which the corresponding latent vector has been sampled. Finally, we obtain the soft-assignment to clusters of each sample $\mathbf{x}$ in the original dataset by using the output of this classifier $c(\mathbf{x}) \in [0,1]^K$ when given $\mathbf{x}$ as input. Each element $c(\mathbf{x})_k$ ($k \in [K]$) of this output vector marks the association level of the sample $\mathbf{x}$ to the cluster k. Hard-assignment to clusters can be trivially obtained from the soft-assignment vector by selecting the most likely cluster $\bar{k} = \arg\max_{k\in[K]} c(\mathbf{x})_k$. This procedure is formally described in Algorithm 2.

Algorithm 2 Unsupervised clustering procedure using GM-GANs.

Require:
X - a set of samples to cluster.
K - number of clusters.
M - number of samples to draw from each Gaussian.

$(G, D) \leftarrow$ GM-GAN(X, K) ▷ Train an unsupervised GM-GAN on X using K Gaussians.
for $k = 1...K$ **do**
  Sample $Z_k \sim N(\mu_k, \Sigma_k)^M$ ▷ Sample M latent vectors from the k'th latent Gaussian.
  $\widetilde{X}_k \leftarrow G(Z_k)$ ▷ Generate M samples using the set of latent vectors Z_k.
  $\forall \widetilde{\mathbf{x}} \in \widetilde{X}_k \; y(\widetilde{\mathbf{x}}) \leftarrow k$ ▷ Label every sample by the Gaussian from which it was generated.
$\widetilde{X} \leftarrow \bigcup_k \widetilde{X}_k$ ▷ Unite all samples into the set $\widetilde{X}$.
$c \leftarrow$ classifier$(\widetilde{X}, y)$ ▷ Train a classifier on samples $\widetilde{X}$ and labels y.
$\forall \mathbf{x} \in X$ cluster$(\mathbf{x}) \leftarrow \arg\max_{k\in[K]} c(\mathbf{x})_k$ ▷ Cluster X using classifier c.

Dataset	Method	ACC	NMI
MNIST	K-Means (Xie et al., 2016)	0.5349	0.500
	AE + K-Means (Xie et al., 2016)	0.8184	-
	DEC (Xie et al., 2016)	0.8430	-
	DCEC (Guo et al., 2017)	0.8897	0.8849
	InfoGAN (Chen et al., 2016)	0.9500	-
	CAE-l_2 + K-Means (Aytekin et al., 2018)	0.9511	-
	CatGAN (Springenberg, 2015)	0.9573	-
	DEPICT (Dizaji et al., 2017)	0.9650	0.9170
	DAC (Chang et al., 2017)	0.9775	0.9351
	GAR (Kilinc & Uysal, 2018)	0.9832	-
	IMSAT (Hu et al., 2017)	0.9840	-
	GM-GAN (Ours)	**0.9924**	**0.9618**
Traffic Signs	K-Means*	0.2447	0.1977
	AE + K-Means*	0.2932	0.2738
	GM-GAN (Ours)	**0.8974**	**0.9274**
Fashion-MNIST	K-Means*	0.4714	0.5115
	AE + K-Means*	0.5353	0.5261
	GM-GAN (Ours)	**0.5816**	**0.5690**

Table 3: Clustering performance of our method on different datasets. Scores are based on clustering accuracy (ACC) and normalized mutual information (NMI). Results of a broad range of recent existing solutions are also presented for comparison. The results of alternative methods are the ones reported by the authors in the original papers. Methods marked with (*) are based on our own implementation, as we didn't find any published scores to compare to.

Empirical Evaluation We evaluated the proposed clustering method on three different datasets: MNIST, Fashion-MNIST, and a subset of the Synthetic Traffic Signs Dataset containing 10 selected classes (see Table 1). To evaluate clustering performance we adopt two commonly used metrics: *Normalized Mutual Information* (NMI), and *Clustering Accuracy* (ACC). *Clustering accuracy* measures the accuracy of the hard-assignment to clusters, with respect to the best permutation of the dataset's ground-truth labels. *Normalized Mutual Information* measures the mutual information between the ground-truth labels and the predicted labels based on the clustering method. The range of both metrics is $[0, 1]$. The unsupervised clustering scores of our method are presented in Table 3. We noet in passing that Algorithm 2 can be implemented with other GAN variants which are augments with a GM distribution of the latent space with similar beneficial results, see Figure 9 in Appendix D.

6 Summary

This work is motivated by the observation that the commonly used GAN architecture may be ill suited to model data in such cases where the training set is characterized by large inter-class and intra-class diversity, a common case with real-world datasets these days. To address this problem we propose a variant of the basic GAN model where the probability distribution over the latent space is a mixture of Gaussians, a multi-modal distribution much like the target data distribution which the GAN is trained to model. This model can be used with or without label supervision. In addition, the proposed modifications can be applied to any GAN model, regardless of the specifics of the loss function and architecture (see, for example, Figure 9 in Appendix D.).

In our empirical study, using both synthetic and real-world datasets, we quantitatively showed that GM-GANs outperform baselines, both when evaluated using the commonly used Inception Score (Salimans et al., 2016), and when evaluated using our own alternative scoring method. We also demonstrated how the quality-diversity trade-off offered by our models can be controlled by altering, post-training, the probability distribution of the latent space. This allows one to sample higher-quality, lower-diversity samples or vice versa, according to one's needs. Finally, we qualitatively demonstrated how the *unsupervised* variant of GM-GAN tends to map latent vectors sampled from different Gaussians in the latent space to samples of different classes in the data space. We further showed how this phenomenon can be exploited for the task of unsupervised clustering, and backed our method with quantitative evaluation.

REFERENCES

Andrew Y. Ng Adam Coates, Honglak Lee. An analysis of single layer networks in unsupervised feature learning. *AISTATS*, 2011. URL http://cs.stanford.edu/~acoates/stl10.

Caglar Aytekin, Xingyang Ni, Francesco Cricri, and Emre Aksu. Clustering and unsupervised anomaly detection with l2 normalized deep auto-encoder representations. *arXiv preprint arXiv:1802.00187*, 2018.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5879–5887, 2017.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *ArXiv e-prints*, June 2016.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 5747–5756. IEEE, 2017.

H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. *ArXiv e-prints*, September 2017.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. *ArXiv e-prints*, March 2017.

Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International Conference on Neural Information Processing*, pp. 373–382. Springer, 2017.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. *arXiv preprint arXiv:1702.08720*, 2017.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *ArXiv e-prints*, November 2016.

Ozsel Kilinc and Ismail Uysal. Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization. *arXiv preprint arXiv:1802.03063*, 2018.

T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *ArXiv e-prints*, March 2017.

D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. *ArXiv e-prints*, June 2014.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

A. B. L. Larsen, S. Kaae Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *ArXiv e-prints*, December 2015.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *ArXiv e-prints*, September 2016.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial Autoencoders. *ArXiv e-prints*, November 2015.

X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. Least Squares Generative Adversarial Networks. *ArXiv e-prints*, November 2016.

M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *ArXiv e-prints*, November 2015.

M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *ArXiv e-prints*, November 2014.

Boris Moiseev, Artem Konev, Alexander Chigorin, and Anton Konushin. Evaluation of traffic sign recognition methods trained on synthetically generated data. In Jacques Blanc-Talon, Andrzej Kasinski, Wilfried Philips, Dan Popescu, and Paul Scheunders (eds.), *Advanced Concepts for Intelligent Vision Systems*, pp. 576–583, Cham, 2013. Springer International Publishing. ISBN 978-3-319-02895-8.

A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *ArXiv e-prints*, October 2016.

S. Pascual, A. Bonafonte, and J. Serrà. SEGAN: Speech Enhancement Generative Adversarial Network. *ArXiv e-prints*, March 2017.

A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv e-prints*, November 2015.

S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative Adversarial Text to Image Synthesis. *ArXiv e-prints*, May 2016.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. *ArXiv e-prints*, June 2016.

J. T. Springenberg. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. *ArXiv e-prints*, November 2015.

Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Zhou Wang, Eero Simoncelli, Alan Bovik, et al. Multi-scale structural similarity for image quality assessment. In *ASILOMAR CONFERENCE ON SIGNALS SYSTEMS AND COMPUTERS*, volume 2, pp. 1398–1402. Citeseer, 2003.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487, 2016.

L.-C. Yang, S.-Y. Chou, and Y.-H. Yang. MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation. *ArXiv e-prints*, March 2017.

R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic Image Inpainting with Deep Generative Models. *ArXiv e-prints*, July 2016.

L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *ArXiv e-prints*, September 2016.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *ArXiv e-prints*, March 2017.

A RELATED WORK

GANs have been extensively used in the domain of computer-vision, where their applications include super resolution from a single image (Ledig et al., 2016), text-to-image translation (Reed et al., 2016), image-to-image translation (Zhu et al., 2017; Isola et al., 2016; Kim et al., 2017), image in-painting (Yeh et al., 2016) and video completion (Mathieu et al., 2015). Aside from their use in the computer-vision domain, GANs have been used for other tasks such as semi-supervised learning (Kingma et al., 2014; Springenberg, 2015), music generation (Yang et al., 2017; Dong et al., 2017), text generation (Yu et al., 2016) and speech enhancement (Pascual et al., 2017).

Subsequently much effort was directed at improving GANs through architectural changes to G and D, as in the DCGANs described in (Radford et al., 2015). Improved performance was reported in Mao et al. (2016); Gulrajani et al. (2017), among others, by modifying the loss function used to train the GAN model. Additional improvement was achieved by introducing supervision into the training setting, as in conditional GANs (Mirza & Osindero, 2014; Odena et al., 2016). These conditional variants were shown to enhance the quality of the generated sample, while also improving the stability of the notorious training process of these models.

In an effort to impose a meaningfully structure on the latent space, Info-GAN (Chen et al., 2016) decomposes the input noise into an incompressible source and a "latent code", attempting to discover latent sources of variation by maximizing the mutual information between the latent code and the Generator's output. This latent code can be used to discover object classes in a purely unsupervised fashion, although it is not strictly necessary that the latent code be categorical. Adversarial Auto-Encoders (Makhzani et al., 2015) employ GANs to perform variational inference by matching the aggregated posterior of the auto-encoder's hidden latent vector with an arbitrary prior distribution. As a result, the decoder of the adversarial auto-encoder learns a deep generative model that maps the imposed prior to the data distribution. Larsen et al. (2015) combined a Variational Auto-Encoder with a Generative Adversarial Network in order to use the learned feature representations in the GAN's discriminator as basis for the VAE reconstruction objective. As a result, this hybrid model is capable of learning a latent space in which high-level abstract visual features (e.g. wearing glasses) can be modified using simple arithmetic of latent vectors.

B DYNAMIC VS STATIC GM-GAN MODEL

The parameters of the mixture of Gauusian distribution used to sample the latent vector can be fixed or learned. One may be able to choose these parameters by using prior knowledge, or pick them randomly. Perhaps a more robust solution is to learn the parameters of the Gaussian Mixture along with the parameters of the GAN in an "end-to-end" fashion. This should, intuitively, allow for a more flexible, and perhaps better performing model. We therefore investigated two variants of the new model - one (static) where the the parameters of the Gaussians mixture are fixed throughout the model's training process, and one (dynamic) where these parameters are allowed to change during the training process in order to potentially converge to a better a solution. These variants are described in detail next:

Static GM-GAN. In the basic GM-GAN model, which we call *Static Multi-Modal GAN (Static GM-GAN)*, we assume that the parameters of the mixture of Gaussians distribution are fixed before training the model, and cannot change during the training process. More specifically, each of the mean vectors μ_k is uniformly sampled from the multivariate uniform distribution $U[-c, c]^d$, and each of the covariance matrices Σ_k has the form of $\sigma * I_{d\times d}$, where $c \in \mathbb{R}$ and $\sigma \in \mathbb{R}$ are hyper-parameters left to be determined by the user.

Dynamic GM-GAN. We extend our basic model in order to allow for the dynamic tuning of parameters for each of the Gaussians in the mixture. We start by initializing the mean vectors and covariance matrices as in the static case, but we include them in the set of learnable parameters that are optimized during the GAN's training process. This modification allows the Gaussians' means to wander to new locations, and lets each Gaussian have a unique covariance matrix. This potentially allows the model to converge to a better local optimum, and achieve better performance.

The architecture of the *Dynamic* GM-GAN is modified so that G receives as input a categorical random variable $\mathbf{k}$, which determines from which Gaussian the latent vector should be sampled. This vector is fed into a stochastic node used for sampling latent vectors given the Gaussian's index, i.e. $\mathbf{z}|k \sim N(\mu_k, \Sigma_k)$. In order to optimize the parameters of each Gaussian in the training phase, back-propagation would have to be performed through this stochastic node, which is not possible. To overcome this obstacle, we use the re-parameterization trick as suggested by Kingma & Welling (2013): instead of sampling $\mathbf{z} \sim N(\mu_k, \Sigma_k)$ we sample $\epsilon \sim N(0, I)$ and define $\mathbf{z} = A_k\epsilon + \mu_k$, where $A \in \mathbb{R}^{d\times d}$ and $\mu_k \in \mathbb{R}^d$ are parameters of the model, and d is the dimension of the latent space. We thus get $\mu(\mathbf{z}) = \mu_k$ and $\Sigma(\mathbf{z}) = A_k A_k^T$.

We note that when training either the static or dynamic variants of our model, we optimize for the same loss functions as in (2) and (3). Clearly other loss functions can be used in conjunction with the suggested architectural modifications, as those changes are independent.

We also note that the dynamic variant of our model includes additional $K * (d^2 + d)$ trainable parameters, as compared to the static model. In cases where K and d are sufficiently large, this can introduce significant computational overhead to the optimization procedure. To mitigate this issue, one can reduce the number of degrees of freedom in Σ_k, e.g. by assuming a diagonal matrix, in which case the number of additional trainable parameters is reduced to $2 * K * d$.

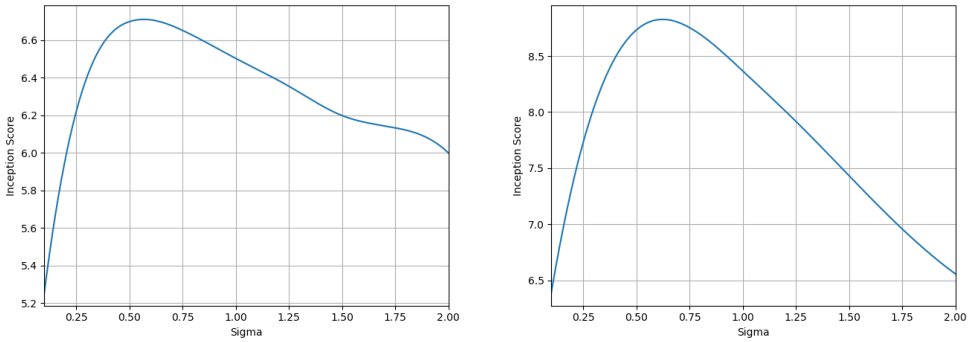
C THE *Inception Score*

Salimans et al. (2016) proposed a method to evaluate generative models for natural image synthesis, such as VAEs and GANs, using a pre-trained classifier. It is based on the fact that good samples, i.e. images that look like images from the true data distribution, are expected to yield: (i) low entropy $p(y|\mathbf{x})$, implying high prediction confidence; (ii) high entropy $p(y)$, implying highly varied predictions. Here $\mathbf{x}$ denotes an image sampled from the Generator, $p(y|\mathbf{x})$ denotes the inferred class label probability given $\mathbf{x}$ by the Inception network (Szegedy et al., 2016) pre-trained on the ImageNet dataset, and $p(y)$ denotes the marginal distribution over all images sampled from the Generator. Thus the *Inception Score* is defined as:

$$\exp\left(\mathbb{E}_{\mathbf{x}\sim p_G}\left[D_{KL}(p(y|\mathbf{x})||p(y))\right]\right) \tag{8}$$

This score has a number of drawbacks which we found to be rather limiting:

1. The Inception Score is based on the Inception network (Szegedy et al., 2016), which was pre-trained on the ImageNet dataset. This dataset contains $\sim$ 1.2 million natural images belonging to 1,000 different classes. As a result the use of the Inception Score is limited to cases where the dataset consists of natural images. For example, we cannot use the Inception Score to evaluate the performance of a GAN trained on the MNIST dataset, which contains gray-scale images of hand-written digits.
2. Even in cases where the dataset on which we train a GAN consists of natural images, the distribution of these images is likely to be very different from that of ImageNet. In which case, the confidence of the Inception network's prediction on such images may not correlate well with their actual quality.
3. The Inception Score only measures the samples' *inter-class* diversity, namely, the distribution of these samples across different classes $p(y)$. Another equally important measure, which must be taken into account, is the *intra-class* diversity of samples, namely, the variance of different samples which all belong to the same class.
4. The Inception Score combines together a measure of quality and a measure of diversity into a single score. When evaluating the qualities of a GAN using solely this combined score, one cannot asses the true trade-off between the quality and the diversity of generated images. Thus a given Inception Score can be achieved by a GAN which generates very diverse but poor quality images, and also by a GAN which generates similarly looking but high quality images. Different Inception Scores can also be achieved by the same GAN, when sampling latent vectors with different parameters of the source probability distribution (e.g. σ), as illustrated in Figure 6.



(a) (b)

Figure 6: Inception Scores of Static GM-GAN models trained on (a) CIFAR-10 and (b) STL-10, when latent vectors are sampled using different values of σ. In both cases, the same model achieves very different Inception Scores when different values of σ are used. Both models were trained using $\sigma = 1$. Note that the best score is obtained for $\sigma < 1$, far from the training value $\sigma = 1$.

D EXPERIMENTAL EVALUATION, ADDITIONAL FIGURES

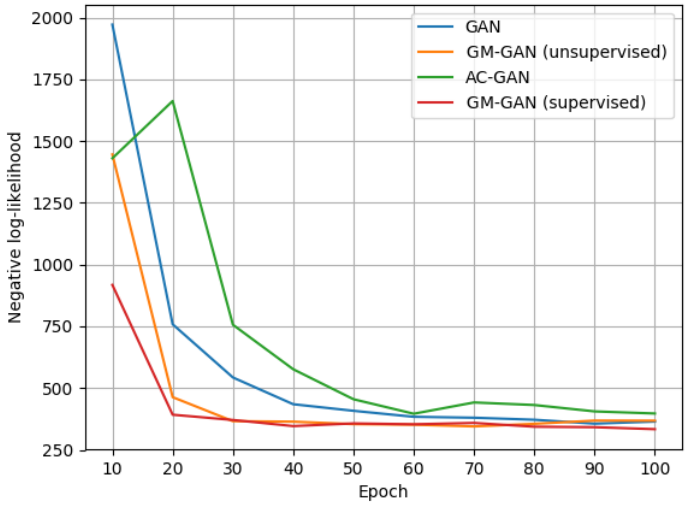


Figure 7: Convergence rate of our proposed models vs. baselines. The plot shows the negative log-likelihood of generated samples, as a function of the training epoch of each model. Both variants of the GM-GAN model converge much faster as compared to the baseline models.

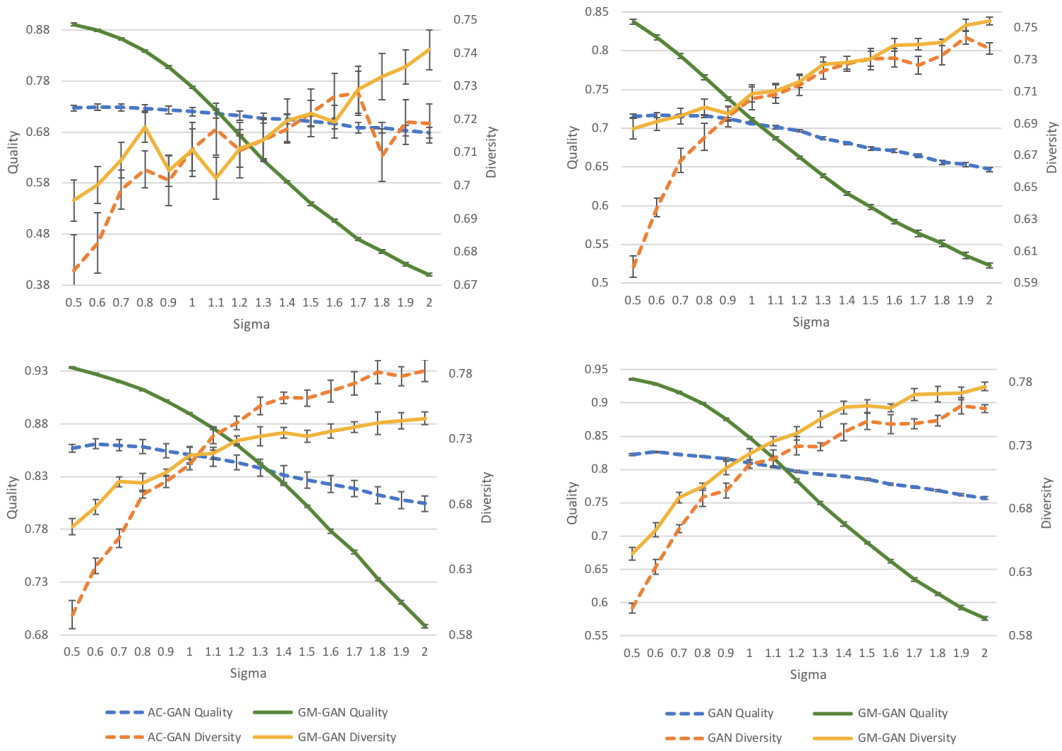


Figure 8: Quality and Diversity scores of GM-GANs vs. baselines trained on 2 datasets: **Fashion-MNIST** (top) and **MNIST** (bottom). Left column: AC-GANs vs. supervised GM-GANs. Right column: GANs vs. unsupervised GM-GANs. Error bars show the standard error of the mean.

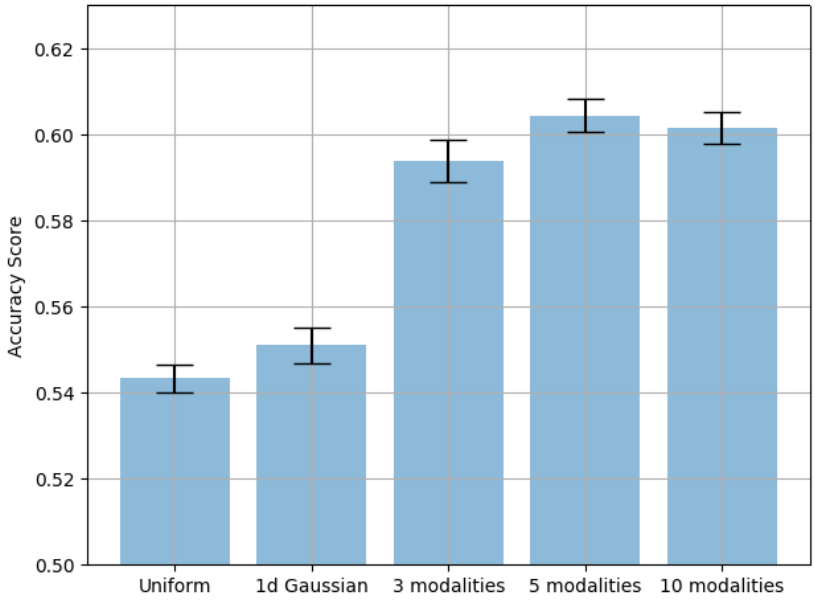
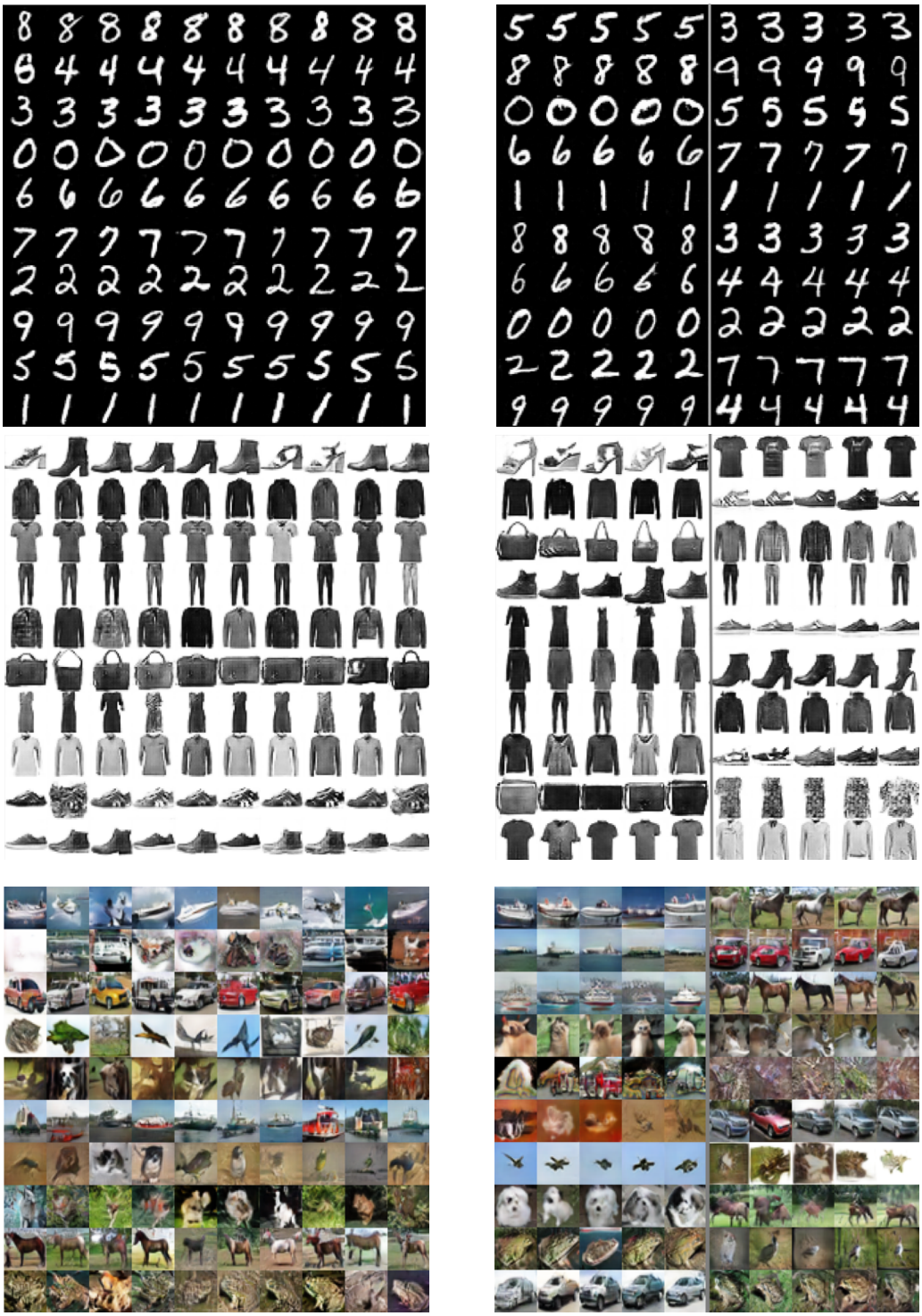


Figure 9: Unsupervised clustering with the info-GAN model (Chen et al., 2016) trained on the fashion-MNIST dataset. We evaluated the original model with uniform and Gaussian latent space distribution. In addition, we incorporated multi-modal Gaussian distribution of the latent space into the model with 3, 5 and 10 mixture components. The results of the GM-info-GAN variants are clearly better than vanilla, with best results obtained for 5 Gaussian components.



(a) (b)

Figure 10: Samples taken from two unsupervised GM-GAN models trained on the MNIST (top panels), Fashion-MNIST (middle panels) and CIFAR-10 (bottom panels) datasets. In (a) the Gaussian mixture contains $K = 10$ Gaussians; in each panel, each row contains images sampled from a different Gaussian. In (b) the Gaussian mixture contains $K = 20$ Gaussians; in each panel, each half row contains images sampled from a different Gaussian.