Multi-Modal Generative Adversarial Networks

By

MATAN BEN-YOSEF

Under the supervision of

PROF. DAPHNA WEINSHALL

THE HEBREW
UNIVERSITY
OF JERUSALEM

Faculty of Computer Science and Engineering
THE HEBREW UNIVERSITY OF JERUSALEM

A thesis submitted to the Hebrew University of Jerusalem as a partial fulfillment of the requirements of the degree of MASTER OF SCIENCE in the Faculty of Computer Science and Engineering.

AUGUST 2018

Abstract

Generative Adversarial Networks [11] (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, called Multi-Modal-GAN (MM-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 empirical experiments, using both synthetic and real-world datasets, we quantitatively show that MM-GANs outperform baselines, both when evaluated using the commonly used Inception Score [32], and when evaluated using our own alternative scoring method. In addition, we qualitatively demonstrate how the *unsupervised* variant of MM-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.

Dedication and Acknowledgements

There are a number of people without whom this thesis might not have been written, and to whom I am wish to dedicate this work.
To my advisor, Professor Daphna Weinshall, who has given me the opportunity to pursue my ideas through this research. I was very fortunate to be a student of yours.
To my friends and family, and especially to my parents, who have endlessly encouraged and supported me throughout this journey.
And most of all, to my life partner, Iris, who has lit my way in times when all other lights seemed to go out.

TABLE OF CONTENTS

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

Generative models have long been an important and active field of research in machine-learning. Such models take as input a training set of data points from an unknown data distribution, and return an estimate of that distribution. By learning to capture the statistical distribution of the training data, this family of models allows one to generate additional data points by sampling from the learned distribution. Well-known families of generative methods include the Naïve Bayes model, Hidden Markov models, Deep Belief Networks, Variational Auto-Encoders [19] (VAEs) and Generative Adversarial Networks (GANs) [11]; this thesis focuses on the latter family.

Generative Adversarial Networks 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. If G is able to perfectly match the real data distribution $p_{\mathcal{X}}$, then D will be maximally confused, predicting 0.5 for all input samples. Such a state is known as a *Nash equilibrium*, and has been shown in [11] to be the optimal solution for this learning framework. Both D and G are implemented by deep differentiable networks, typically consisting

of multiple convolutional and fully-connected layers. They are alternately trained using the Stochastic Gradient Descent algorithm. Figure 1.1 illustrates the structure of a GAN.

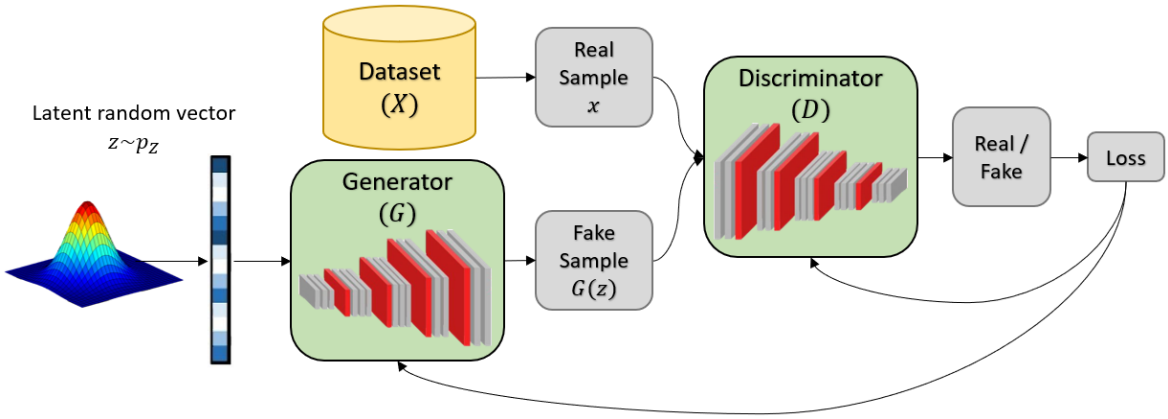


Figure 1.1: Illustration of a GAN's structure.

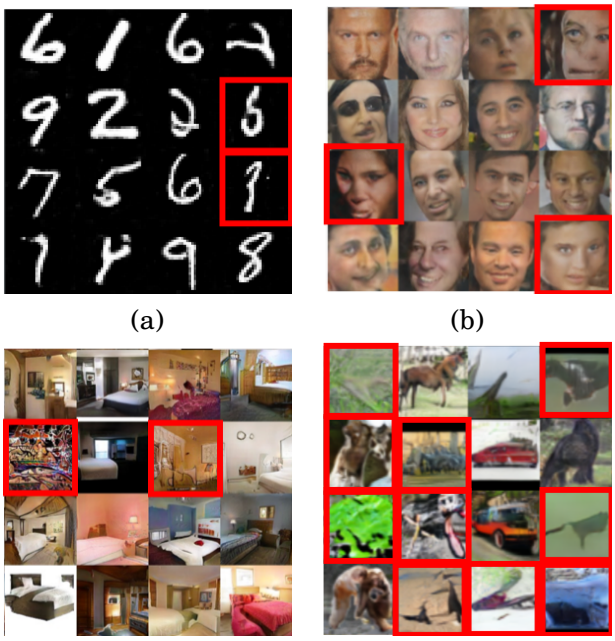
GANs have been extensively used in the domain of computer-vision, where their applications include super resolution from a single image [22], text-to-image translation [31], image-to-image translation [15, 17, 43], image in-painting [41] and video completion [25]. Aside from their usages in the computer-vision domain, GANs have been used for other tasks such as semi-supervised learning [18, 33], music generation [10, 40], text generation [42] and speech enhancement [29].

1.1 Motivation For This Work

In the short period of time since their introduction, many different enhancement methods and training variants have been suggested to improve their performance (see brief review in Section 1.2 below). Despite these efforts, often a large proportion of the generated sample 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 1.2).

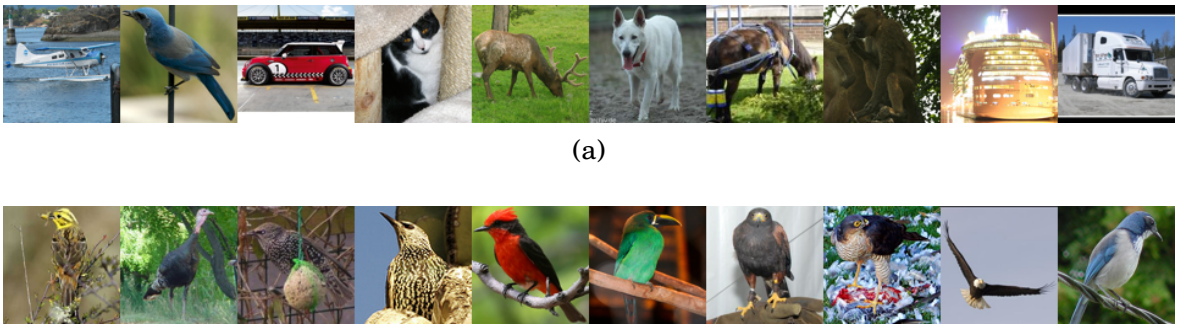
The problem described above worsens with the increased complexity of the training set, and specifically when the training set is characterized by large *inter-class* and *intra-class* diversity. The *inter-class* and the *intra-class* diversity of a dataset can, informally, be defined as the variability among samples belonging to *different* classes, and the variability among samples belonging to the *same* class, respectively. Figure 1.3 illustrates these terms.

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.



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

Figure 1.2: Images generated by different GANs trained on (a) MNIST, (b) CelebA, (c) LSUN Bedrooms, (d) STL-10. Images marked with a red square are, arguably, of low quality.



(a) (b)

Figure 1.3: Samples from the STL-10 dataset which demonstrate the complexity of this dataset. (a) Different samples belonging to 10 different classes which demonstrate a large *inter-class* diversity. (b) Different samples belonging to the same class (birds) which demonstrate a large *intra-class* diversity.

1.2 Related Work

In an attempt to improve the performance of the original GAN model [11], many variants and extensions have been proposed in the past few years. Much effort was directed at improving GANs through architectural changes to G and D. Thus [30] proposed a family of GAN architectures called Deep Convolutional GANs (DCGANs), which replaced the traditional fully-connected layers with strided and fractionally-strided convolutional and transposed convolutional layers. This allows the spatial down-sampling and up-sampling operators to be learned during training. DCGANs, and other variants of these models, are widely used for many applications involving computer vision and images. LAPGAN [8] offers higher-quality image synthesis by generating images in a multi-scale coarse-to-fine fashion: training images are converted to a Laplacian pyramid, and a cascade of convolutional GANs is tasked with the generation of each layer of the pyramid, conditioned on the one above it.

The loss function used in GANs' training process was also an important point of focus in previous works: [24] found that the loss function used in regular GANs may lead to the vanishing gradients problem during the learning process. To overcome this problem, they adopted the least squares loss function for the discriminator, which resulted in higher image quality, and stabler training of this model. [2] Argued that the divergences which GANs typically minimize are potentially not continuous with respect to the generator's parameters, leading to training difficulty. Instead of minimizing the Jensen-Shannon divergence between the generated data and the real data distributions, as proposed in [11], they proposed to minimize the Earth-Mover distance between these two distributions. Further building upon this direction, [12] proposed to penalize the norm of the discriminator's gradient with respect to its input, instead of clipping its weights, as performed in [2].

Other improvements to the original model were achieved by introducing supervision into the training setting: [26] suggested a variant of GANs called conditional GANs [26] (CGANs), where the Generator and the Discriminator are both conditioned on some side information, e.g. a class label, which is fed to them in addition to a random latent vector. [28] took this idea further and proposed a variant of the conditional GAN where the discriminator acts as a multi-class classifier and outputs a probability distribution over class labels, in addition to the probability that a given input sample is real. Conditional variants of GANs have proved to enhance the sample quality, while also improving the stability of the notorious training process of these models.

Another branch of related works, which perhaps more closely relates to our work, involves the learning of a meaningfully structured latent space: Info-GAN [7] decomposes the input noise into an incompressible source and a "latent code", attempting to discover latent factors 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 [23] 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. [5] 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.

1.3 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.

Our approach is 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. 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. Supervision can be incorporated into this model by adding a correspondence (not necessarily injective) between labels and mixture components. This family of models is described in Section 2.

1.4 Contributions

The main contributions of this thesis are:

- In Chapter 2, we propose a novel family of GANs which we call Multi-Modal GANs (MM-GANs). We further extend this family and provide a supervised variant of MM-GANs which is capable of conditional sample synthesis.
- In Chapter 3, we discuss the shortcomings of the popular Inception Score [32], and further show that GANs offer a trade-off between sample quality and diversity. We propose an alternative evaluation score which is, arguably, better suited to the task of image synthesis using GANs, and which can quantify the quality-diversity trade-off.
- In Chapter 4, we empirically evaluate our proposed model on the task of sample synthesis, when trained with various diverse datasets. We show that MM-GANs outperform baselines and achieve better scores.
- In Chapter 5, we describe a method for clustering datasets using MM-GANs, and provide qualitative and quantitative evaluation using various datasets of real images.

CHAPTER 2

MULTI-MODAL GAN

We next describe our proposed model, and describe its training schema. We then describe an extension to this model, and discuss the possible benefits introduced by this extension.

2.1 Unsupervised Multi-Modal GAN

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

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\mathcal{X}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \tag{2.1}$$

Above $p_{\mathcal{X}}$ denotes the distribution of real training samples, and p_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:

$$L(G) = -\mathop{\mathbb{E}}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log D(G(\mathbf{z}))] \tag{2.2}$$

$$L(D) = -\mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\mathcal{X}}(\mathbf{x})}[\log D(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{z}\sim p_Z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \tag{2.3}$$

Usually, a multivariate uniform distribution (e.g. $U[-1,1]^d$), or a multivariate normal distribution (e.g. $N(0,I_{d\times d})$) is used as p_Z when training GANs. In our proposed model, we optimize for the same target function as in 2.1, but instead of using a unimodal random distribution for the prior p_Z, we propose to use a multi-modal distribution which can better suit the inherent

multi-modality of the real training data distribution, $p_{\mathcal{X}}$. In this work, we propose to use a mixture of Gaussians as a multi-modal prior distribution. Formally, we have:

$$p_Z(\mathbf{z}) = \sum_{k=1}^{K} \alpha_k * p_k(\mathbf{z}) \tag{2.4}$$

where K denotes the number of Gaussians in the mixture, $\{\alpha_k\}_{k=1}^{K}$ denotes the elements of 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 μ_k, Σ_k of each Gaussian in the mixture 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:

2.1.1 Static MM-GAN

In the basic MM-GAN model, which we call *Static Multi-Modal GAN (Static MM-GAN)*, we assume that the parameters of the mixture of Gaussians distribution are fixed before training the model, and cannot change during the model's 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.

2.1.2 Dynamic MM-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* MM-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 [19]: 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 (2.2) and (2.3). 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 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 $2 * K * d$.

2.2 Supervised MM-GAN

It has been previously shown ([26, 28]) that training GANs with class labels supervision has several benefits which include better sample quality, as well as improved training stability. We therefore investigated a further extension to both variants of our proposed model, in order to support training with label supervision when labels are available.

In the supervised setting, we change the MM-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 the range of $[0, 1]$, and can be interpreted as the probability that the given sample is a real sample of class i. Informally, this modification can be thought of as having N binary discriminators, where each discriminator i is trained to separate real samples of class i from fake samples of class i and from real samples of classes other than class i.

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 of 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. This can be useful in cases where the training set is characterized by high intra-class diversity and when single classes can be broken down to multiple, visually distinct, sub-classes. When $K < N$ f is injective, and multiple classes can be mapped to the same Gaussian achieving the clustering of class labels.

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

$$L(G) = -\mathop{\mathbb{E}}_{\mathbf{z}\sim p_Z(\mathbf{z})}\left[\log D(G(\mathbf{z}))_{f(y(\mathbf{z}))} + \sum_{m=1, m\neq f(y(\mathbf{z}))}^{N} \log(1 - D(G(\mathbf{z}))_m)\right] \tag{2.5}$$

$$L(D) = -\mathop{\mathbb{E}}_{\mathbf{z}\sim p_Z(\mathbf{z})}\left[\sum_{m=1}^{N} \log(1 - D(G(\mathbf{z}))_m)\right] - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\mathcal{X}}(\mathbf{x})}\left[\log D(\mathbf{x})_{y(x)} + \sum_{m=1, m\neq y(x)}^{N} \log(1 - D(\mathbf{x})_m)\right] \tag{2.6}$$

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 $\mathbf{z}$ was sampled.

The training procedure for MM-GANs is fully described in algorithm 1.

Algorithm 1 Training procedure of the **MM-GAN** model. Most of our experiments are conducted with the default values $d = 100$, $c = 0.1$, $\sigma = 0.15$, $b_D = 64$, $b_G = 128$, $\gamma = 0.0002$. K and $iters$ vary with different experiments.

Require:

K - the number of Gaussians in the mixture.
d - the dimension of the latent space (Z).
c - defines 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 $k \sim Categ(\frac{1}{K}, ..., \frac{1}{K})$ ▷ uniformly 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 $k \sim Categ(\frac{1}{K}, ..., \frac{1}{K})$ ▷ uniformly 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.

CHAPTER 3

GAN EVALUATION SCORE

We describe in this chapter a new scoring method for GANs, proposed as an alternative to existing scoring methods. In particular, we argue that this new scoring method is better suited for the task than the commonly used Inception Score [32].

3.1 Inception Score, Definition and Shortcomings

[32] 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 [35] pre-trained on the ImageNet dataset, and $p(y)$ denotes the marginal distribution over all images sampled from the Generator.

The *Inception Score* [32] is therefore defined as:

$$\exp\left(\mathbb{E}_{\mathbf{x}\sim p_G}\left[D_{KL}(p(y|\mathbf{x})||p(y))\right]\right) \tag{3.1}$$

This score has been used extensively over the last few years. However, it has a number of drawbacks which we found to be rather limiting::

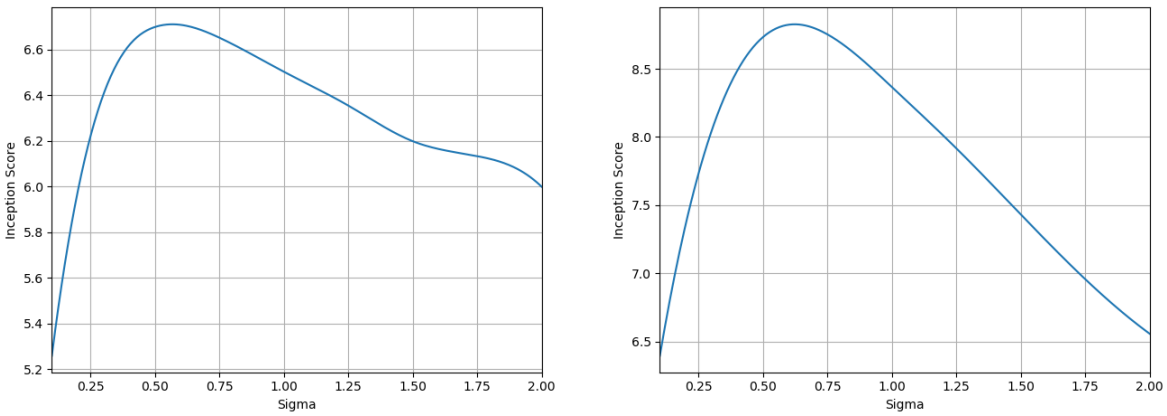
1. The Inception Score is based on the Inception network [35], which was pre-trained on the ImageNet dataset. This dataset contains ~ 1.2 million natural images belonging to 1,000 different classes. As a result the use of the Inception Score is limited to cases where the dataset consists of natural images. 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 3.1.



(a) (b)

Figure 3.1: Inception Scores of Static MM-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 $\sigma = 1$.

3.2 Alternative Score: Measuring the Quality-Diversity Trade-off

A GAN's Generator can be thought of as a model which maps a probability distribution of a source domain Z (the latent space), to a probability distribution of a target domain $\mathcal{X}$ (the data space). The source probability distribution is known, and can be easily sampled, while the target probability distribution is the one we are interested in estimating by training a GAN. In cases where the source probability distribution's PDF is not constant (i.e. it is not uniform), it would be reasonable to expect that a well trained Generator will map samples of high probability in the source domain to samples of high probability in the target domain, and vice versa. In other words, we expect that such a Generator, G, will support the following: $\forall \mathbf{z} \in Z, p_Z(\mathbf{z}) \approx p_{\mathcal{X}}(G(\mathbf{z}))$, where $p_Z(\mathbf{z})$ is the probability of a sample $\mathbf{z}$ in the source domain, and $p_{\mathcal{X}}(G(\mathbf{z}))$ is the probability of the sample $G(\mathbf{z})$ in the target domain.

Following this intuition, If we measure the quality of a sample $\mathbf{x} \in \mathcal{X}$ in the target domain by its probability $p_{\mathcal{X}}(\mathbf{x})$, then we can expect samples drawn from dense areas in the source domain (i.e. close to the modals of the distribution) to be 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 different GAN models.

We therefore propose a new scoring system which can be used to measure the trade-off between quality and diversity offered by GAN models. The details of this method are provided in Section 3.3 below. We use this scoring method to quantitatively demonstrate the benefits of our proposed model over other baselines, as presented in Section 4.3.

3.3 Quality-Diversity Trade-off Score

Next we propose a new scoring method for GANs, which allows one to evaluate the trade-off between samples' quality and diversity. This scoring method also relies on a pre-trained classifier, but unlike the Inception Score, this classifier is trained on the *same* training set on which the GAN is trained on. This classifier is used to measure both the quality and the diversity of generated samples, as explained below.

[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 Figs. 4.2, 4.4). We further elaborate on this matter in Section 4.3.

3.3.1 Quality Score

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

$$q(\mathbf{x}) = 1 - \frac{\exp(||c_l(\mathbf{x}) - c_l(NN(\mathbf{x}))||_2)}{\exp(||c_l(\mathbf{x}) - c_l(NN(\mathbf{x}))||_2) + a} \tag{3.2}$$

Above a is a constant greater than zero, and $NN(\mathbf{x})$ is the nearest-neighbor of $\mathbf{x}$ in the training set, with regards to the pre-trained classifier's intermediate representation c_l, and is defined as $NN(\mathbf{x}) = \arg\min_{\mathbf{x}' \in X} ||c_l(\mathbf{x}) - c_l(\mathbf{x}')||_2$. We also define the quality score for a set of samples X as follows:

$$q(X) = \sum_{\mathbf{x} \in X} \frac{1}{|X|} q(\mathbf{x}) \tag{3.3}$$

3.3.2 Diversity Score

To measure the diversity of generated samples, we take into account both the inter-class, and the intra-class diversity. For **intra-class** diversity we measure the average (negative) MS-SSIM metric [36] between all pairs of generated images in a given set of generated images X:

$$d_{intra}(X) = 1 - \frac{1}{|X|^2} \sum_{(\mathbf{x},\mathbf{x}') \in X \times X} MS-SSIM(\mathbf{x},\mathbf{x}') \tag{3.4}$$

For **intra-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:

$$d_{inter}(X) = \frac{1}{\log(N)} H\left(\frac{1}{|X|} \sum_{\mathbf{x} \in X} c(\mathbf{x})\right) \tag{3.5}$$

We combine both the **intra-class** and the **inter-class** diversity scores into a single **diversity score** as follows:

$$d(X) = \sqrt{d_{intra}(X) * d_{inter}(X)} \tag{3.6}$$

3.3.3 Combined Score

While it is important to look at the *quality* and *diversity* scores separately, since they measure two complementary properties of a model, it is sometimes necessary to obtain a single score per model. We therefore define the following combined measure:

$$s(X) = \sqrt{q(X) * d(X)} \tag{3.7}$$

The range of the proposed *quality*, *diversity* and *combined* scores is $[0,1]$, where 0 marks the lowest score, and 1 marks the highest score. This property makes them easy to comprehend, and convenient to use when comparing the performance of different models.

CHAPTER 4

EXPERIMENTAL EVALUATION

In this chapter we empirically evaluate the benefits of our proposed approach, comparing the performance of MM-GAN with alternative baselines. Specifically, we compare the performance of the unsupervised MM-GAN model to that of the originally proposed GAN [11], and the performance of our proposed supervised MM-GAN model to that of AC-GAN [28]. 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 MM-GAN models are similar to those used for training the baseline models. In the following experiments we evaluated the different models on the 6 datasets listed in Table 4.1. Further details about these datasets are provided in Table 4.1. 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	Number of Classes	Samples Dimension	Train Samples	Test Samples
Toy-Dataset	Points sampled from different Gaussians in the 2-D Euclidean space.	9	2	5,000	-
MNIST [21]	Images of handwritten digits.	10	28x28x1	60,000	10,000
Fashion-MNIST [38]	Images of clothing articles.	10	28x28x1	60,000	10,000
CIFAR-10 [20]	Natural images.	10	32x32x3	50,000	10,000
STL-10 [1]	Natural images.	10	96x96x3	5,000	8,000
Synthetic Traffic Signs [27]	Synthetic images of street traffic signs.	43	40x40x3	100,000	-

Table 4.1: Details of the different datasets used in the empirical evaluation: a Toy-Dataset which we have created (see details in Section 4.1), MNIST [21], Fashion-MNIST [38], CIFAR-10 [20], STL-10 [1] and the Synthetic Traffic Signs Dataset [27].

4.1 Toy-Dataset

We first compare the performance of our proposed MM-GAN models to the aforementioned baseline models using a toy dataset, which was created in order to gain more intuition regarding the properties of the MM-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 M Gaussians, i.e., $\forall \mathbf{x}$ $p(\mathbf{x}) = \sum_{m=1}^{M} \frac{1}{M} p_m(\mathbf{x})$ where $p_m(\mathbf{x}) \sim N(\mu_m, \Sigma_m)$. In our experiments we used $M = 9$ Gaussians, $\forall m \in [M]$ $\Sigma_m = 0.1 * I$ 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 MM-GAN model, one supervised, using the labels of the samples, and one unsupervised, which was not given access to these labels. In both cases, we used $K = 9$ Gaussians in the mixture from which latent vectors are sampled. Figure 4.1 presents samples generated by the baseline models (GAN, AC-GAN) and samples generated by our proposed MM-GAN models (both unsupervised and supervised variants). It is clear that both variants of the MM-GAN generate samples with a higher likelihood, which matches the original distribution more closely as compared to the baseline methods. It is also evident that in this configuration, the diversity of samples generated by the MM-GAN model is lower than that of the classic GAN model. This illustrates the trade-off between quality and diversity, which we explore more thoroughly in Section 4.3. Figure 4.2 demonstrates the superiority of MM-GAN as compared to classic GAN, when measuring the trade-off between quality and diversity offered by

these models (see Section 4.3 for further elaboration on this matter).

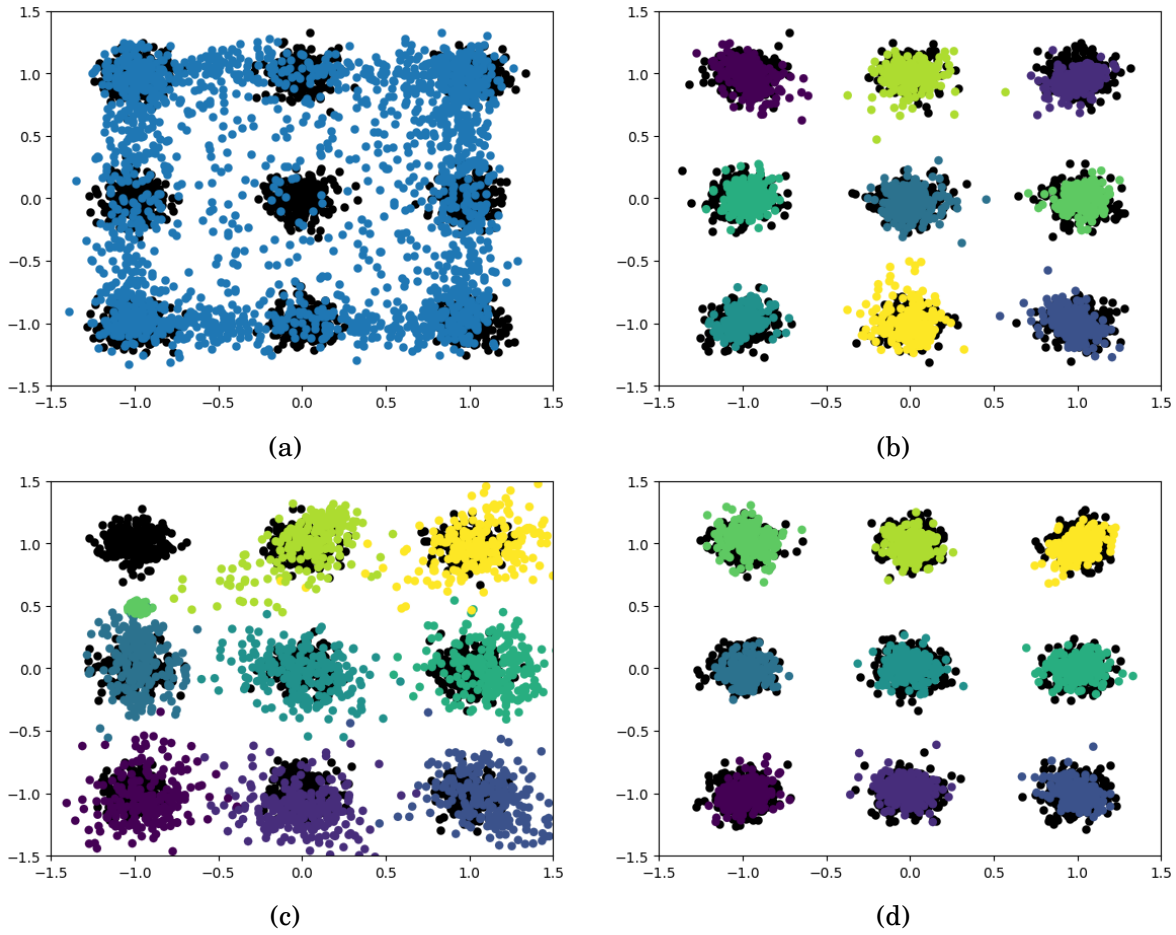


Figure 4.1: Samples from the toy-dataset along with samples generated from: (a) GAN, (b) unsupervised MM-GAN, (c) AC-GAN, (d) supervised MM-GAN. Samples from the training set are drawn in black, and samples generated by the trained Generators are drawn in color. In (b) and (d), the color of each sample represents the Gaussian from which the corresponding latent vector was sampled.

An intriguing observation is that the MM-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 when training unsupervised MM-GAN on the MNIST and Fashion-MNIST datasets. In Chapter 5 we exploit this phenomenon by training unsupervised clustering models.

Finally, we note that the MM-GAN models converge considerably faster than the baseline models. Figure 4.3 shows the (negative) log-likelihood of samples generated from the different models, as a function of the training epoch.

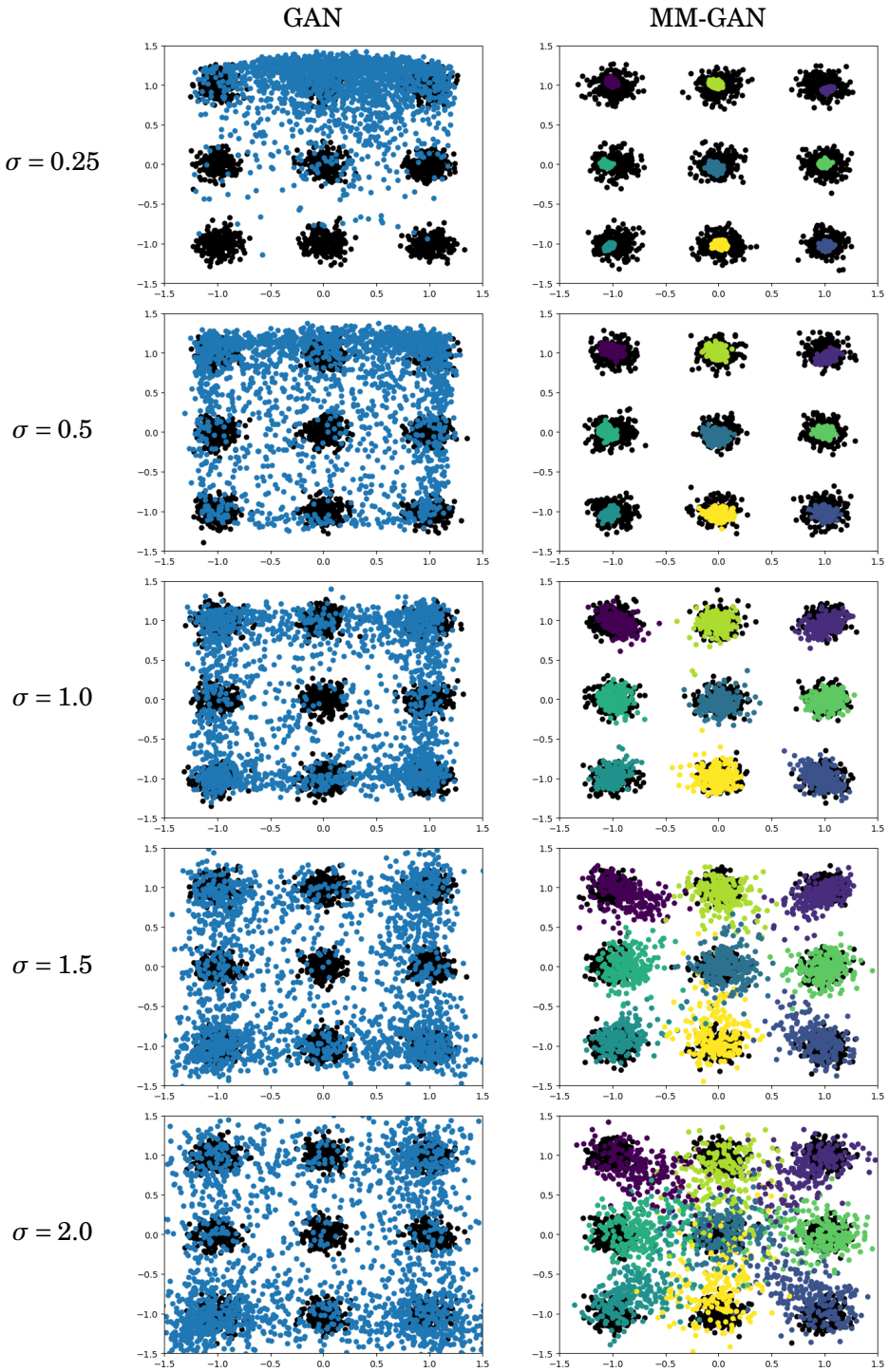


Figure 4.2: Samples from the toy-dataset along with samples generated from GAN (left column) and unsupervised MM-GAN (right column), using different σ values for sampling latent vectors from the latent space Z. During the training process of both models, latent vectors were sampled with $\sigma = 1.0$. Samples from the training set are drawn in black, and samples generated by the trained Generators are drawn in color. In samples generated by the MM-GAN, the color of each sample represents the Gaussian from which the corresponding latent vector was sampled. MM-GAN clearly offers a better trade-off between quality and diversity as compared to the baseline.

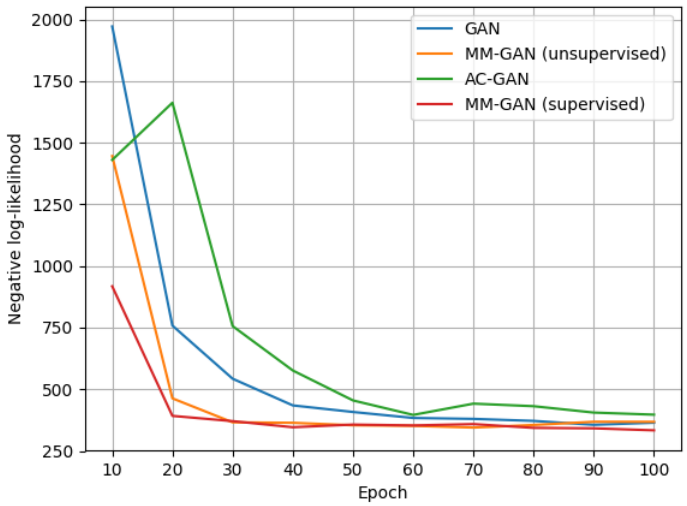


Figure 4.3: Convergence rate of our proposed models vs. baselines. The plot shows the negative log-likelihood of generated samples, as a function of the training epoch of each model. Both variants of the MM-GAN model converge much faster as compared to the baseline models.

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 [32] to evaluate and compare the performance of the difference models, the two MM-GAN models and the baseline models (GAN and AC-GAN). We trained the models on two real datasets with 10 classes each, the CIFAR-10 [20] and STL-10 [1] datasets. Each variant of the MM-GAN model is 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 4.2. In all cases, the two MM-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 MM-GAN model outperforms AC-GAN by a large margin. We also found that the number of Gaussians used in the MM-GAN's latent space probability distribution can improve or impair the performance of the corresponding model, depending on the dataset.

CIFAR-10

Model (unsupervised)	**Score**
GAN	5.71 (±0.06)
MM-GAN (k=10)	5.92 (±0.07)
MM-GAN (k=20)	5.91 (±0.05)
MM-GAN (k=30)	**5.98** (±**0.05**)

Model (supervised)	**Score**
AC-GAN	6.23 (±0.07)
MM-GAN (k=10)	**6.84** (±**0.03**)
MM-GAN (k=20)	6.81 (±0.04)
MM-GAN (k=30)	6.83 (±0.02)

STL-10

Model (unsupervised)	**Score**
GAN	6.80 (±0.07)
MM-GAN (k=10)	**7.06** (±**0.11**)
MM-GAN (k=20)	6.58 (±0.16)
MM-GAN (k=30)	7.03 (±0.10)

Model (supervised)	**Score**
AC-GAN	7.45 (±0.10)
MM-GAN (k=10)	**8.32** (±**0.06**)
MM-GAN (k=20)	8.16 (±0.05)
MM-GAN (k=30)	8.08 (±0.07)

Table 4.2: Inception Scores for different MM-GAN models vs. baselines trained on the CIFAR-10 and STL-10 datasets.

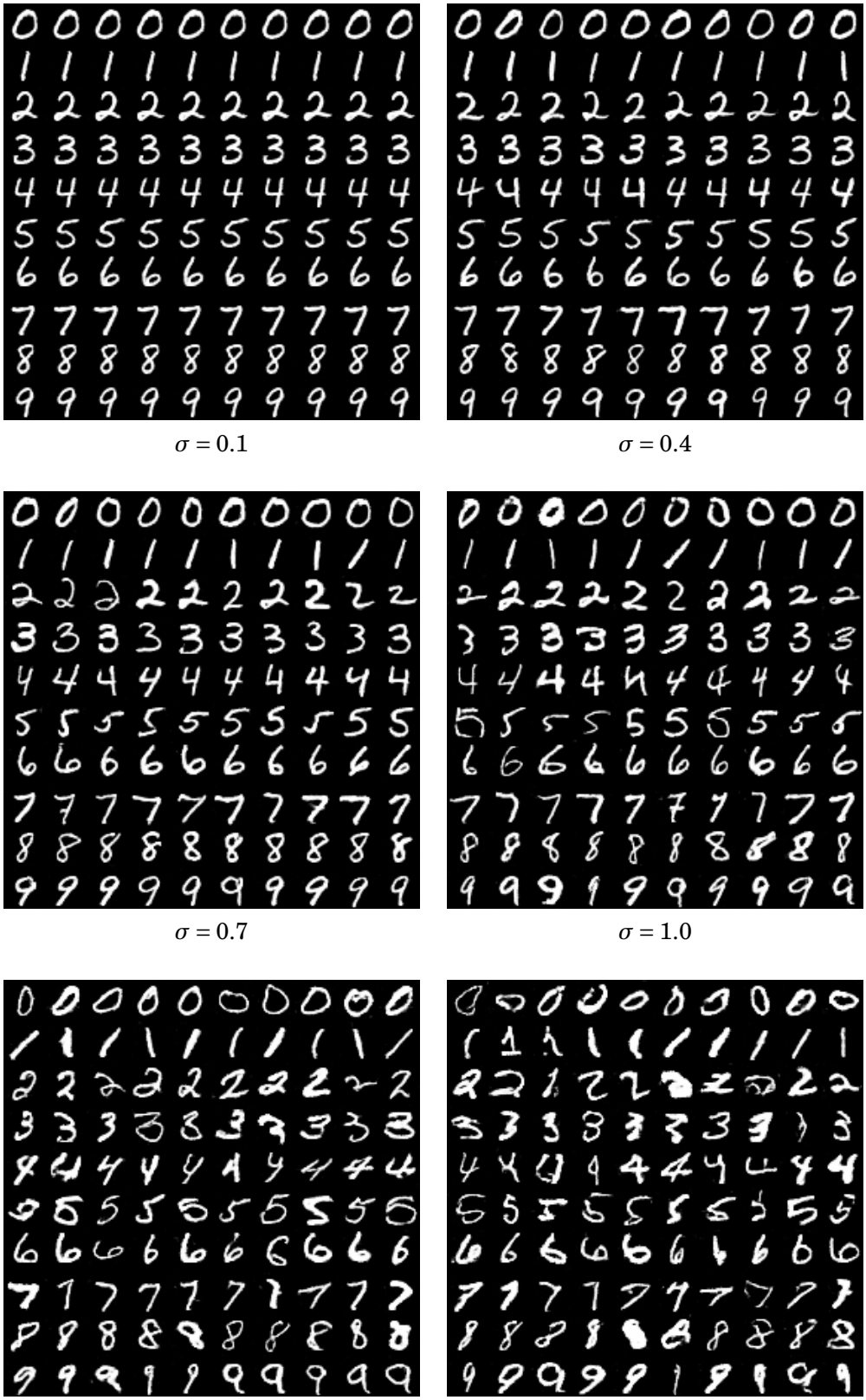
4.3 Trade-off between Quality and Diversity

As discussed in Chapter 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 MM-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 4.2, 4.4 demonstrate qualitatively the quality-diversity trade-off offered by MM-GANs when trained on the Toy and MNIST datasets.

We evaluated each model by calculating our proposed Quality Score from Eq. (3.2), and the Combined Diversity Score from Eq. (3.6), 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 MM-GAN and baseline models, when trained on the CIFAR-10, STL-10, Fashion-MNIST and MNIST datasets, are presented in Figure 4.5.

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 MM-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 offers higher quality, but dramatically lower diversity samples, 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 between different classes. In all cases, the charts predictably show an ascending Quality Score, and a descending Combined Diversity Score, as σ is increased. This correlates well with qualitative results which we have examined during our experiments, and thus shows that our proposed scoring method fits its task.



$\sigma = 0.1$ $\sigma = 0.4$

$\sigma = 0.7$ $\sigma = 1.0$

$\sigma = 1.5$ $\sigma = 2.0$

Figure 4.4: Samples taken from an MM-GAN trained on the MNIST dataset. In each panel, samples are taken with a different value of σ. The quality of samples decreases, and the diversity increases, as σ grows.

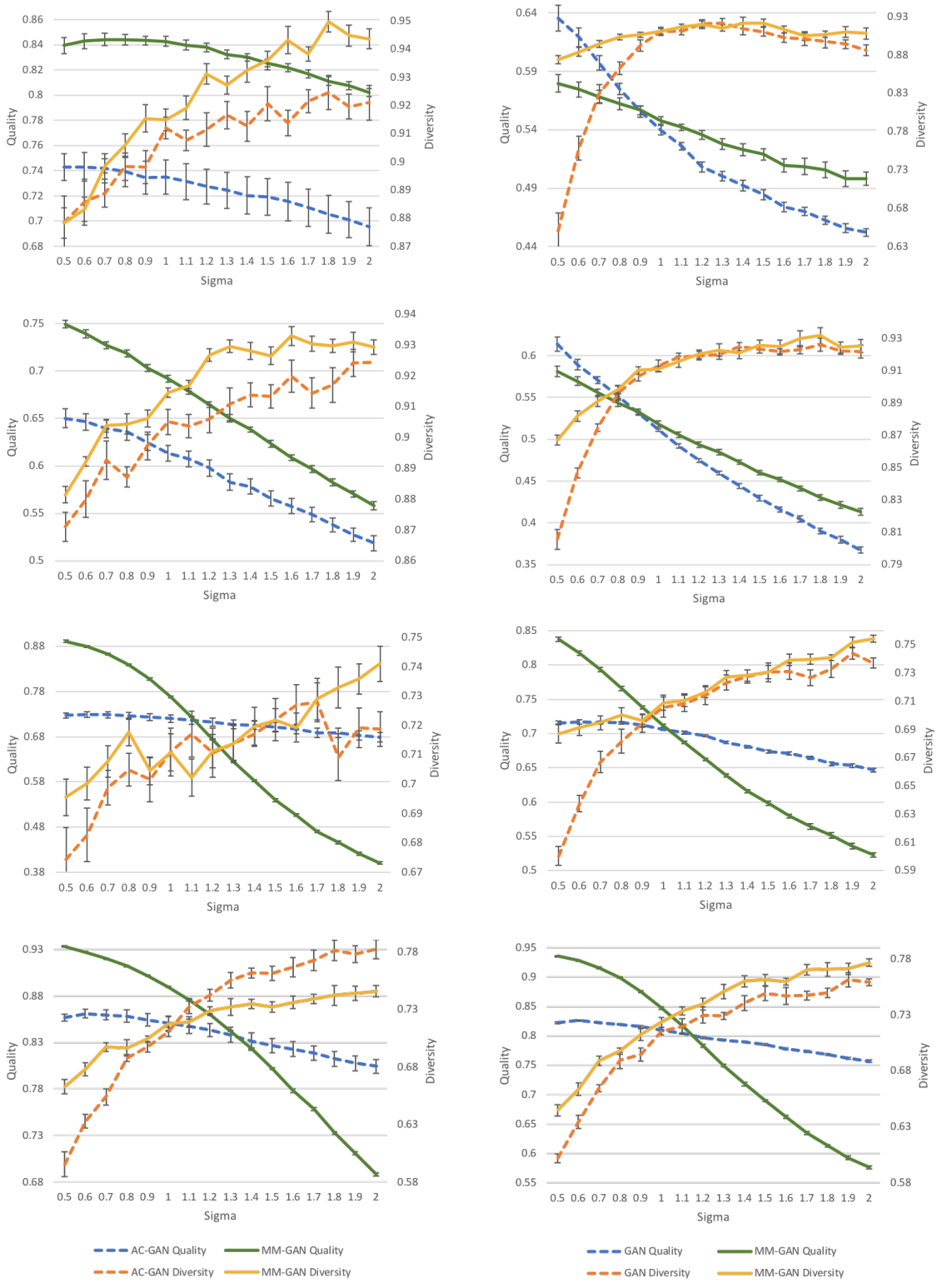


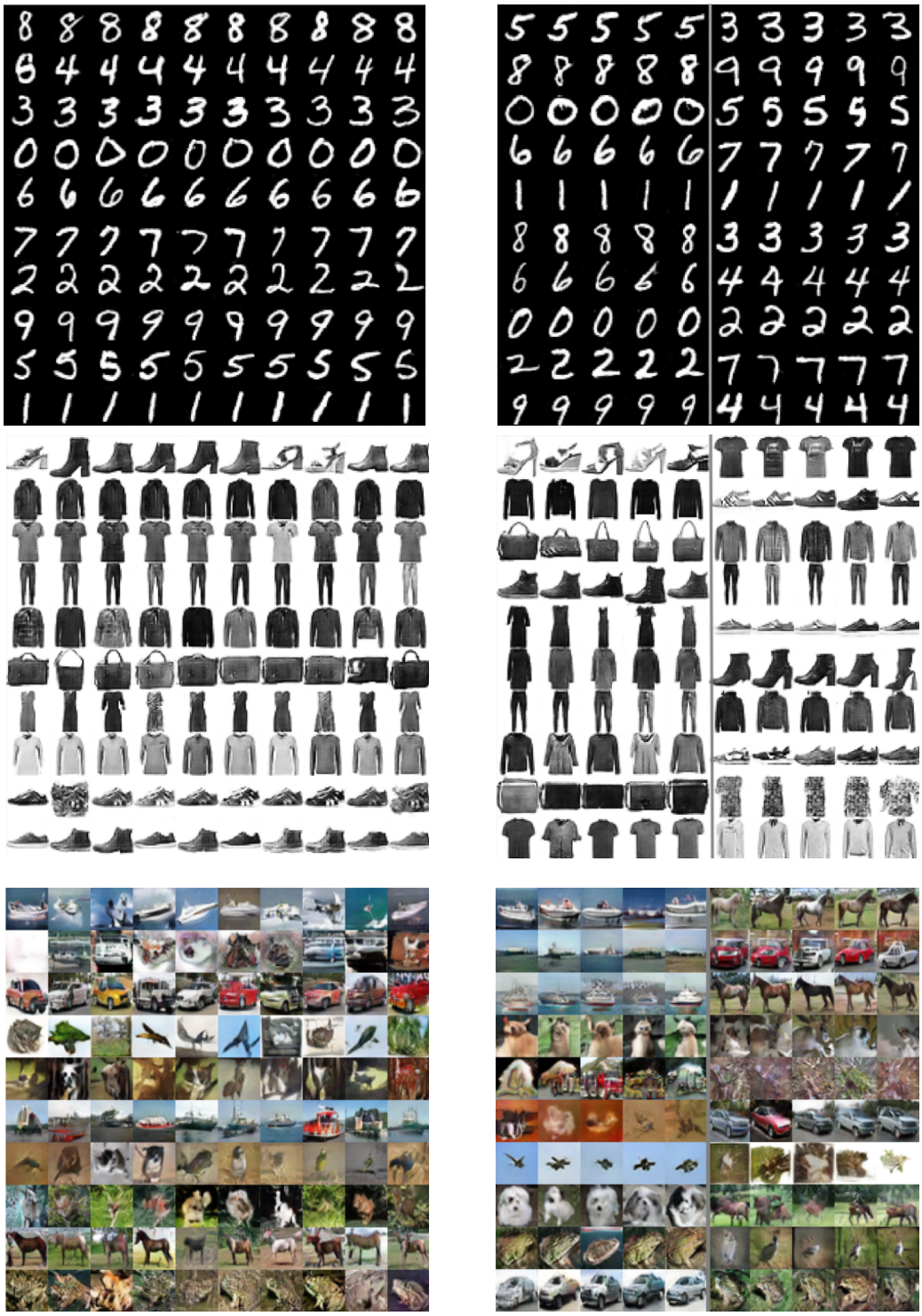
Figure 4.5: Quality and Diversity scores of MM-GANs vs. baselines trained on 4 datasets, each corresponding to a different row, shown from top to bottom as follows: **CIFAR-10**, **STL-10**, **Fashion-MNIST** and **MNIST-10**. Left column: AC-GANs vs. supervised MM-GANs. Right column: GANs vs. unsupervised MM-GANs. Error bars show the standard error of the mean.

CHAPTER 5

UNSUPERVISED CLUSTERING USING MM-GANS

Throughout our experiments, we noticed an intriguing phenomenon where the *unsupervised* variant of MM-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 MM-GAN's Generator, to a single class in the data space. Figures 4.1, 5.1 demonstrate this phenomenon on 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 chapter, we exploit this observation to develop a new clustering algorithm, and provide quantitative evaluation of the proposed method when applied on different datasets.



(a) (b)

Figure 5.1: Samples taken from two unsupervised MM-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.

5.1 Clustering Method

Our clustering method works as follows: we first train an unsupervised MM-GAN on the dataset, where K, the number of Gaussians forming the latent space, is set to equal the number of clusters in the intended partition. Using the trained MM-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. A hard-assignment to clusters can be trivially calculated from the soft-assignment vector by selecting the cluster k with which the sample is mostly associated, i.e. $\arg\max_{k\in[K]} c(\mathbf{x})_k$. This clustering procedure is formally described in Algorithm 2.

Algorithm 2 Unsupervised clustering procedure using MM-GANs.

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

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

5.2 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 4.1). For every dataset, we run our method with the number of clusters set to be the number of classes in the dataset. 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 $[0,1]$ where a larger value indicates more precise clustering results. Both metrics are formally defined as follows:

$$ACC(c|X,y) = \max_{\pi \in S_N} \frac{1}{|X|} \Sigma_{\mathbf{x} \in X} \mathbb{1}_{y(\mathbf{x})=\pi(c(\mathbf{x}))} \tag{5.1}$$

$$NMI(c|X,y) = \frac{1}{|X|} \Sigma_{\mathbf{x} \in X} \frac{I(y(\mathbf{x}), c(\mathbf{x}))}{\sqrt{H(y(\mathbf{x}))\, H(c(\mathbf{x}))}} \tag{5.2}$$

Above X denotes the dataset on which clustering is performed, $y(\mathbf{x})$ denotes the ground-truth label of sample $\mathbf{x}$, $c(\mathbf{x})$ denotes the cluster assignment of sample $\mathbf{x}$, H denotes entropy, I denotes mutual-information, and S_N denotes the set of all permutations on N elements (the number of classes in the dataset).

The unsupervised clustering scores of our method are presented in Table 5.1.

Dataset	**Method**	**ACC**	**NMI**
MNIST	K-Means [39]	0.5349	0.500
	AE + K-Means [39]	0.8184	-
	DEC [39]	0.8430	-
	DCEC [13]	0.8897	0.8849
	InfoGAN [7]	0.9500	-
	CAE-l_2 + K-Means [3]	0.9511	-
	CatGAN [34]	0.9573	-
	DEPICT [9]	0.9650	0.9170
	DAC [6]	0.9775	0.9351
	GAR [16]	0.9832	-
	IMSAT [14]	0.9840	-
	MM-GAN (Ours)	**0.9924**	**0.9618**
Synthetic Traffic Signs	K-Means*	0.2447	0.1977
	AE + K-Means*	0.2932	0.2738
	MM-GAN (Ours)	**0.8974**	**0.9274**
Fashion-MNIST	K-Means*	0.4714	0.5115
	AE + K-Means*	0.5353	0.5261
	MM-GAN (Ours)	**0.5816**	**0.5690**

Table 5.1: 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.

When evaluated on the MNIST dataset, our method outperforms other recent alternative methods, and, to the best of our knowledge, achieves state-of-the-art performance. Less impressive performance is achieved on the Fashion-MNIST dataset. The fact that this dataset is characterized by small inter-class diversity may be the underlying reason for this. In such a case, an MM-GAN

with merely $K = 10$ Gaussians may struggle to model this dataset in such a way where different Gaussians in the latent space are mapped to different classes in the data space. Thus, some Gaussians in the latent space are mapped to multiple classes in the data-space and therefore the resulting performance of our method deteriorates. In such case, improved performance can potentially be achieved by increasing the number of Gaussians forming the latent space; however, in this configuration it would not be possible to quantitatively measure the performance of the resulting dataset partitioning, thus we skip this test.

CHAPTER 6

SUMMARY AND DISCUSSION

This chapter summarizes this work and discusses the benefits and shortcomings of the methods we have proposed.

6.1 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. Additionally, we propose a supervised variant of this model which is capable of conditional sample synthesis. We note that these modifications can be applied to any GAN model, regardless of the specifics of the loss function and architecture.

In order to compare the different models, we note that the performance of GANs, and perhaps other families of generative models, exhibits a certain trade-off between the quality of their generated samples and the diversity of those samples. Therefore arguably the performance of such models must be evaluated by *separately* measuring the quality and the diversity of the generated samples, unlike common practice. For this purpose we propose a scoring method which separately takes into account these two factors. The proposed score can be modified, based on the application's requirement, by adjusting the proportion of each factor when employing the trained model.

In our empirical study, using both synthetic and real-world datasets, we quantitatively showed that MM-GANs outperform baselines, both when evaluated using the commonly used Inception

Score [32], 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 MM-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 which has demonstrated the superior performance of our model over other competitors when evaluated on the MNIST dataset.

6.2 Discussion

It is important to emphasize that the architectural modifications we proposed in this work are orthogonal to, and can be used in conjunction with, other architectural improvements suggested in prior art, such as those reviewed in Section 1.2. Thus, other variants of GANs can also benefit from adopting the proposed method. For example, one may use a multi-modal prior in conjunction with the popular WGAN-GP model [12] in order to achieve better training stability as well as higher quality sample generation, or the InfoGAN model [7] in order to improve the modeling of multi-modal attributes.

The MMGAN model, along with the proposed scoring method, allow one to control the quality-diversity trade-off and directly choose between drawing higher-quality or higher-diversity samples. This can be useful in cases where these factors have an influence on the application for which the GAN is employed. For example, when a GAN is used to boost the performance of a classifier trained in a semi-supervised learning settings, e.g. [32, 33], both the quality and the diversity of the synthetic samples can influence the performance of the target classifier. Thus one may want to carefully choose the right proportions of these two factors when employing the model. Another example is Curriculum Learning [4, 37], a setting in which training samples are gradually revealed from the easiest to the most difficult. Here one can employ our method in order to initially generate high quality and low diversity samples, which are arguably easier, followed by samples of higher diversity and lower quality.

Although lots of room for improvements in GANs still exists, we feel that this work brings these family of models a step closer to achieving the desired capability of true photo-realistic image synthesis, and hope that it will serve as a basis for future work which will continue the march towards this goal.

BIBLIOGRAPHY

[1] A. Y. N. ADAM COATES, HONGLAK LEE, *An analysis of single layer networks in unsupervised feature learning*, AISTATS, (2011).

[2] M. ARJOVSKY, S. CHINTALA, AND L. BOTTOU, *Wasserstein GAN*, ArXiv e-prints, (2017).

[3] C. AYTEKIN, X. NI, F. CRICRI, AND E. AKSU, *Clustering and unsupervised anomaly detection with l2 normalized deep auto-encoder representations*, arXiv preprint arXiv:1802.00187, (2018).

[4] Y. BENGIO, J. LOURADOUR, R. COLLOBERT, AND J. WESTON, *Curriculum learning*, in Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, 2009, ACM, pp. 41–48.

[5] A. BOESEN LINDBO LARSEN, S. KAAE SØNDERBY, H. LAROCHELLE, AND O. WINTHER, *Autoencoding beyond pixels using a learned similarity metric*, ArXiv e-prints, (2015).

[6] J. CHANG, L. WANG, G. MENG, S. XIANG, AND C. PAN, *Deep adaptive image clustering*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5879–5887.

[7] 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, (2016).

[8] E. DENTON, S. CHINTALA, A. SZLAM, AND R. FERGUS, *Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks*, ArXiv e-prints, (2015).

[9] K. G. DIZAJI, A. HERANDI, C. DENG, W. CAI, AND H. HUANG, *Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization*, in Computer Vision (ICCV), 2017 IEEE International Conference on, IEEE, 2017, pp. 5747–5756.

[10] 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, (2017).

[11] 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, (2014).

[12] I. GULRAJANI, F. AHMED, M. ARJOVSKY, V. DUMOULIN, AND A. COURVILLE, *Improved Training of Wasserstein GANs*, ArXiv e-prints, (2017).

[13] X. GUO, X. LIU, E. ZHU, AND J. YIN, *Deep clustering with convolutional autoencoders*, in International Conference on Neural Information Processing, Springer, 2017, pp. 373–382.

[14] W. HU, T. MIYATO, S. TOKUI, E. MATSUMOTO, AND M. SUGIYAMA, *Learning discrete representations via information maximizing self-augmented training*, arXiv preprint arXiv:1702.08720, (2017).

[15] P. ISOLA, J.-Y. ZHU, T. ZHOU, AND A. A. EFROS, *Image-to-Image Translation with Conditional Adversarial Networks*, ArXiv e-prints, (2016).

[16] O. KILINC AND I. UYSAL, *Learning latent representations in neural networks for clustering through pseudo supervision and graph-based activity regularization*, arXiv preprint arXiv:1802.03063, (2018).

[17] 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, (2017).

[18] D. P. KINGMA, D. J. REZENDE, S. MOHAMED, AND M. WELLING, *Semi-Supervised Learning with Deep Generative Models*, ArXiv e-prints, (2014).

[19] D. P. KINGMA AND M. WELLING, *Auto-Encoding Variational Bayes*, ArXiv e-prints, (2013).

[20] A. KRIZHEVSKY, V. NAIR, AND G. HINTON, *Cifar-10 (canadian institute for advanced research)*.

[21] Y. LECUN AND C. CORTES, *MNIST handwritten digit database*, (2010).

[22] 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, (2016).

[23] A. MAKHZANI, J. SHLENS, N. JAITLY, I. GOODFELLOW, AND B. FREY, *Adversarial Autoencoders*, ArXiv e-prints, (2015).

[24] X. MAO, Q. LI, H. XIE, R. Y. K. LAU, Z. WANG, AND S. P. SMOLLEY, *Least Squares Generative Adversarial Networks*, ArXiv e-prints, (2016).

[25] M. MATHIEU, C. COUPRIE, AND Y. LECUN, *Deep multi-scale video prediction beyond mean square error*, ArXiv e-prints, (2015).

[26] M. MIRZA AND S. OSINDERO, *Conditional Generative Adversarial Nets*, ArXiv e-prints, (2014).

[27] B. MOISEEV, A. KONEV, A. CHIGORIN, AND A. KONUSHIN, *Evaluation of traffic sign recognition methods trained on synthetically generated data*, in Advanced Concepts for Intelligent Vision Systems, J. Blanc-Talon, A. Kasinski, W. Philips, D. Popescu, and P. Scheunders, eds., Cham, 2013, Springer International Publishing, pp. 576–583.

[28] A. ODENA, C. OLAH, AND J. SHLENS, *Conditional Image Synthesis With Auxiliary Classifier GANs*, ArXiv e-prints, (2016).

[29] S. PASCUAL, A. BONAFONTE, AND J. SERRÀ, *SEGAN: Speech Enhancement Generative Adversarial Network*, ArXiv e-prints, (2017).

[30] A. RADFORD, L. METZ, AND S. CHINTALA, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, ArXiv e-prints, (2015).

[31] S. REED, Z. AKATA, X. YAN, L. LOGESWARAN, B. SCHIELE, AND H. LEE, *Generative Adversarial Text to Image Synthesis*, ArXiv e-prints, (2016).

[32] T. SALIMANS, I. GOODFELLOW, W. ZAREMBA, V. CHEUNG, A. RADFORD, AND X. CHEN, *Improved Techniques for Training GANs*, ArXiv e-prints, (2016).

[33] J. T. SPRINGENBERG, *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks*, ArXiv e-prints, (2015).

[34] J. T. SPRINGENBERG, *Unsupervised and semi-supervised learning with categorical generative adversarial networks*, arXiv preprint arXiv:1511.06390, (2015).

[35] C. SZEGEDY, V. VANHOUCKE, S. IOFFE, J. SHLENS, AND Z. WOJNA, *Rethinking the Inception Architecture for Computer Vision*, ArXiv e-prints, (2015).

[36] Z. WANG, E. SIMONCELLI, A. BOVIK, ET AL., *Multi-scale structural similarity for image quality assessment*, in ASILOMAR CONFERENCE ON SIGNALS SYSTEMS AND COMPUTERS, vol. 2, Citeseer, 2003, pp. 1398–1402.

[37] D. WEINSHALL, G. COHEN, AND D. AMIR, *Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks*, ArXiv e-prints, (2018).

[38] H. XIAO, K. RASUL, AND R. VOLLGRAF, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, 2017.

[39] J. XIE, R. GIRSHICK, AND A. FARHADI, *Unsupervised deep embedding for clustering analysis*, in International conference on machine learning, 2016, pp. 478–487.

[40] L.-C. YANG, S.-Y. CHOU, AND Y.-H. YANG, *MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation*, ArXiv e-prints, (2017).

[41] 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, (2016).

[42] L. YU, W. ZHANG, J. WANG, AND Y. YU, *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*, ArXiv e-prints, (2016).

[43] J.-Y. ZHU, T. PARK, P. ISOLA, AND A. A. EFROS, *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, ArXiv e-prints, (2017).