On GANs and GMMs

B

by Yumin Cho

Introduction

E. Richardson and Y. Weiss, On GANs and GMMs. In Advances in Neural Information Processing Systems, pages 5847–5858, 2018.

On GANs and GMMs

Eitan Richardson
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel
eitanrich@cs.huji.ac.il

Yair Weiss
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel
yweiss@cs.huji.ac.il

Abstract

A longstanding problem in machine learning is to find unsupervised methods that can learn the statistical structure of high dimensional signals. In recent years, GANs have gained much attention as a possible solution to the problem, and in particular have shown the ability to generate remarkably realistic high resolution sampled images. At the same time, many authors have pointed out that GANs may fail to model the full distribution ("mode collapse") and that using the learned models for anything other than generating samples may be very difficult.

In this paper, we examine the utility of GANs in learning statistical models of images by comparing them to perhaps the simplest statistical model, the Gaussian Mixture Model. First, we present a simple method to evaluate generative models based on relative proportions of samples that fall into predetermined bins. Unlike previous automatic methods for evaluating models, our method does not rely on an additional neural network nor does it require approximating intractable computations. Second, we compare the performance of GANs to GMMs trained on the same datasets. While GMMs have previously been shown to be successful in modeling small patches of images, we show how to train them on full sized images despite the high dimensionality. Our results show that GMMs can generate realistic samples (although less sharp than those of GANs) but also capture the full distribution, which GANs fail to do. Furthermore, GMMs allow efficient inference and explicit representation of the underlying statistical structure. Finally, we discuss how GMMs can be used to generate sharp images.[1]

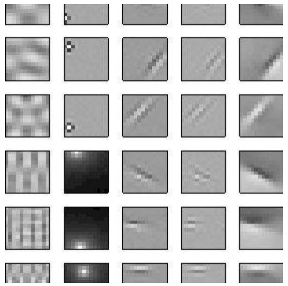
1 Introduction

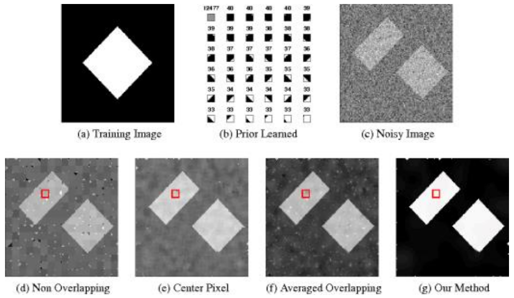
Natural images take up only a tiny fraction of the space of possible images. Finding a way to explicitly model the statistical structure of such images is a longstanding problem with applications to engineering and to computational neuroscience. Given the abundance of training data, this would also seem a natural problem for unsupervised learning methods and indeed many papers apply unsupervised learning to small patches of images [42, 4, 32]. Recent advances in deep learning, have also enabled unsupervised learning of full sized images using various models: Variational Auto Encoders [21, 17], PixelCNN [40, 39, 23, 38], Normalizing Flow [9, 8] and Flow GAN [14].[2]

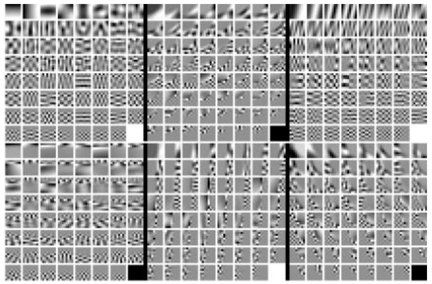
Perhaps the most dramatic success in modeling full images has been achieved by Generative Adversarial Networks (GANs) [13], which can learn to generate remarkably realistic samples at high resolution [34, 26], (Fig. 1). A recurring criticism of GANs, at the same time, is that while they are excellent at generating pretty pictures, they often fail to model the entire data distribution, a phenomenon usually referred to as *mode collapse*: "Because of the mode collapse problem, applications

How to model the statistical structure of images explicitly?

Unsupervised learning for small patches of image

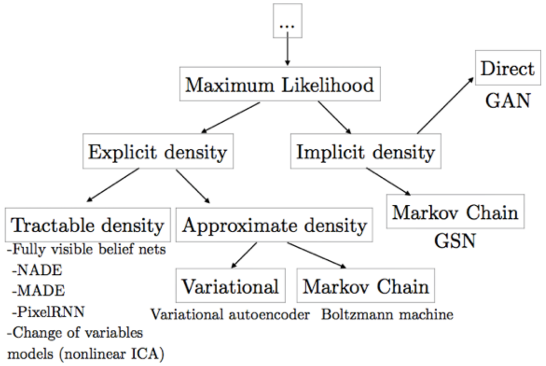


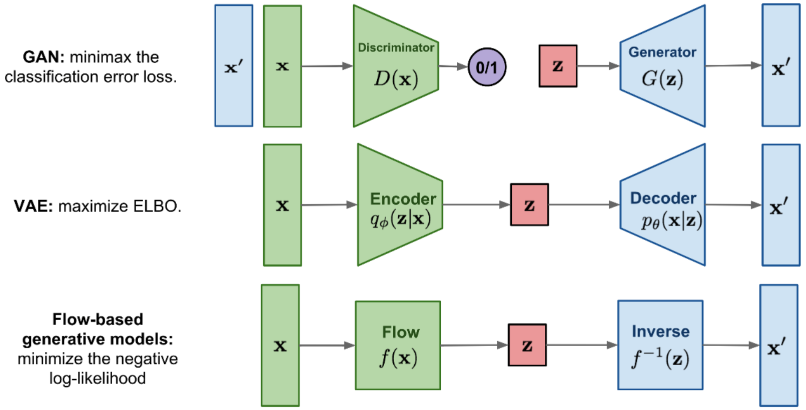




Unsupervised learning of full-sized images

- **GAN**
- VAE
- Normalizing Flow
- PixelCNN



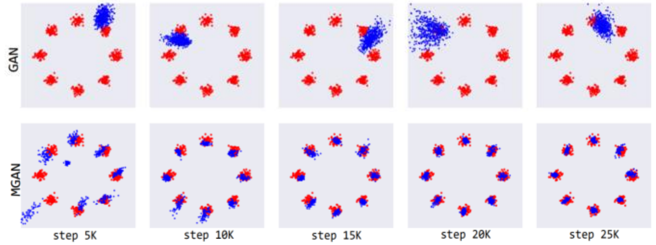


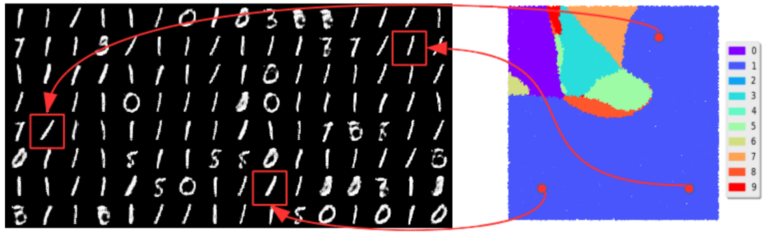
Introduction

Is GAN the best solution to the problem of learning models of full images?

Mode collapse

GAN often fails to model the entire data distribution





Lack of an objective evaluation method

"we feel the quality of the generated images is at least comparable to the best published results so far."

IS(Inception Score), **FID**(Fréchet Inception Distance)

→ Insensitivity to image properties and artifacts

→ Discrepancy between evaluated domain and trained dataset

MS-SSIM(Multi Scale Structural Similarity Index)

→ Difficulty in measuring whether it captures the true distribution

Log-likelihood based methods, **SWD**(Sliced Wasserstein Distance)

→ Intractability in high dimensions

(a) IS = 6.45 (b) IS = 6.31

A new Evaluation Method - NDB

Two set of samples from same distribution

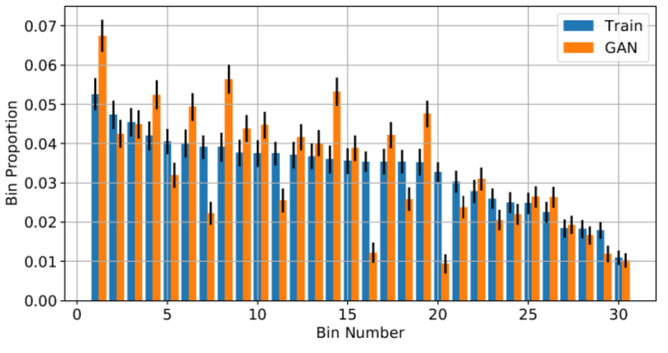
the number of samples that **fall into a given bin** should be same

$I_B(s)$: indicator function for bin B

$\{s_i^p\}$: N_p samples from distribution p

if $p = q$,

$$\text{then } \frac{1}{N_p}\sum_{i}^{N_p} I_B\left(s_i^p\right) \approx \frac{1}{N_q}\sum_{j}^{N_q} I_B\left(s_j^q\right)$$



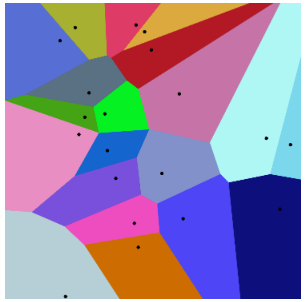
Are they statistically different?

2-sample proportion test

$$Z = \frac{(\hat{p}_1 - \hat{p}_2) - 0}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

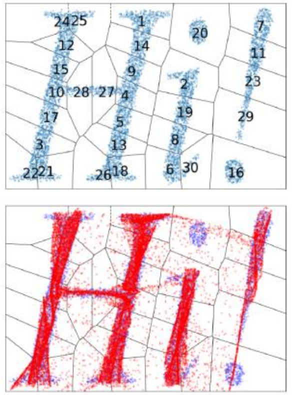
Which bin to use to compare two distributions?

using **Voronoi cell** as bins guarantees that every bin has some samples



How to calculate NDB score

1. perform K-means clustering
2. assign each samples to the nearest of the centroids
3. perform 2-sample test on each cell separately
4. report NDB(number of statistically-different bins)



Pros and Cons

☺ Agnostic and sensitive to different image properties

☹ What if L2 distance is meaningless?

Full Image GMM

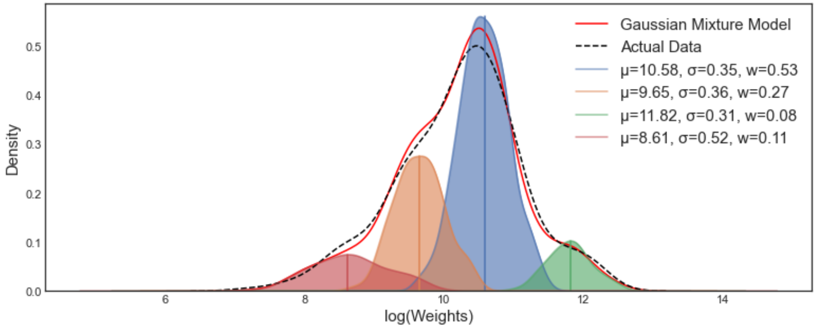
Problem

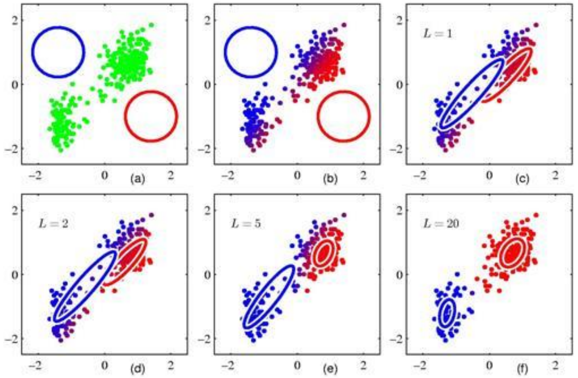
Dimensionality

64x64 color images need a single covariance matrix with 7.5x10⁷ parameters

Complexity

The number of Gaussians required grows exponentially with the dimension





Solution

Mixture of Factor Analyzers(MFA)

Single Factor Analyzer

$$\mathrm{x} \sim N(\mu, \hat{\Sigma}) \qquad \hat{\Sigma} = AA^T + Dx$$

$$\mathrm{x} = Az + \varepsilon \qquad \mathrm{x} \in \mathbb{R}^d, z \sim N(0, I), \varepsilon \sim N(0, D)$$

$$\mathrm{x}_i = \sum_{j=1}^{l} a_{ij} z_j + \varepsilon_i$$

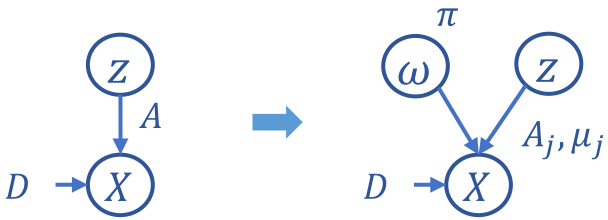
$$\operatorname{argmax}_{A,D} L(\hat{\Sigma}, S) \qquad S\text{: Covariance matrix for observations}$$

Assume a mixture of K factor analyzers indexed by ω_j

$$P(\mathrm{x}) = \sum_{j=1}^{K} \int P(\mathrm{x}|z, \omega_j) P(z|\omega_j) P(\omega_j) dz$$

$$\mathrm{x}|z, \omega_j \sim N(\mu_j + A_j z, D), \; z \sim N(0, I), \; P(\omega_j) = \pi_j$$

<FA> <MFA>



Full Image GMM

```python
def gmm_initial_guess(samples, num_components, latent_dim, clustering_method='km', component_model='fa',
                      default_noise_std=0.5, dataset_std=1.0):
    N, d = samples.shape
    components = {}
    if clustering_method == 'rnd':
        # In random mode, l+1 samples are randomly selected per component, a plane is fitted through them and the
        # noise variance is set to the default value.
        print('Performing random-selection and FA/PPCA initialization...')
        for i in range(num_components):
            fa = FactorAnalysis(latent_dim)
            used_samples = np.random.choice(N, latent_dim + 1, replace=False)
            fa.fit(samples[used_samples])
            components[i] = {'A': fa.components_.T, 'mu': fa.mean_, 'D': np.ones([d])*np.power(default_noise_std, 2.0),
                             'pi': 1.0/num_components}
    elif clustering_method == 'km':
        # In k-means mode, the samples are clustered using k-means and a PPCA or FA model is then fitted for each cluster.
        labels = kmeans_clustering(samples/dataset_std, num_components)
        print("Estimating Factor Analyzer parameters for each cluster")
        components = {}
        for i in range(num_components):
            print('.', end='', flush=True)
            if component_model == 'fa':
                model = FactorAnalysis(latent_dim)
                model.fit(samples[labels == i])
                components[i] = {'A': model.components_.T, 'mu': model.mean_, 'D': model.noise_variance_,
                                 'pi': np.count_nonzero(labels == i)/float(N)}
            elif component_model == 'ppca':
                model = PCA(latent_dim)
                model.fit(samples[labels == i])
                components[i] = {'A': model.components_.T, 'mu': model.mean_, 'D': np.ones([d])*model.noise_variance_/d,
                                 'pi': np.count_nonzero(labels == i)/float(N)}
            else:
                print('Unknown component model -', component_model)
        print()
    else:
        print('Unknown clustering method -', clustering_method)
    return mfa.MFA(components)
```

Experiments

Check visual quality of samples and compare using NDB/K scores

MFA model generates realistic and diverse images although they are not as sharp as the GAN samples

Figure 1: Samples from three datasets (first two rows) and samples generated by GANs (last two rows): CelebA - WGAN-GP, MNIST - DCGAN, SVHN - WGAN

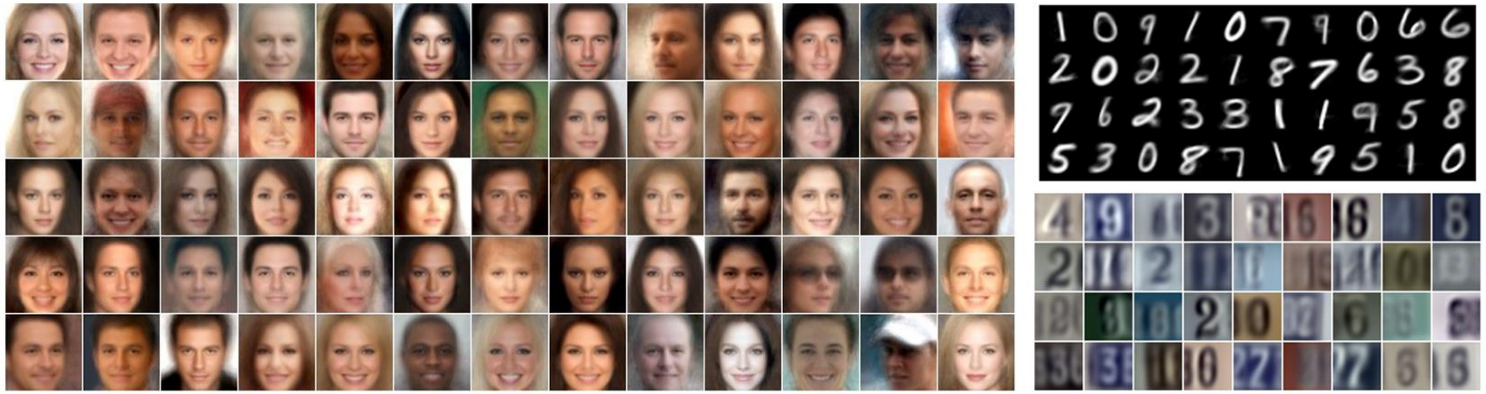


Figure 4: Random samples generated by our MFA model trained on CelebA, MNIST and SVHN

<CelebA>

MODEL	K=100	K=200	K=300
TRAIN	0.01	0.03	0.03
TEST	0.12	0.07	0.08
MFA	**0.21**	**0.12**	**0.16**
VAE	0.78	0.73	0.72
VAE-DFC	0.77	0.65	0.62
DCGAN	0.68	0.69	0.65
BEGAN	0.94	0.85	0.82
WGAN	0.76	0.66	0.62
WGAN-GP	0.42	0.32	0.27

<MNIST>

MODEL	K=100	K=200	K=300
TRAIN	0.06	0.04	0.05
MFA	**0.14**	**0.13**	**0.14**
DCGAN	0.41	0.38	0.46
WGAN	0.16	0.20	0.21

<SVHN>

MODEL	K=100	K=200	K=300
TRAIN	0.03	0.03	0.03
MFA	**0.32**	**0.23**	**0.24**
DCGAN	0.78	0.74	0.76
WGAN	0.87	0.83	0.82

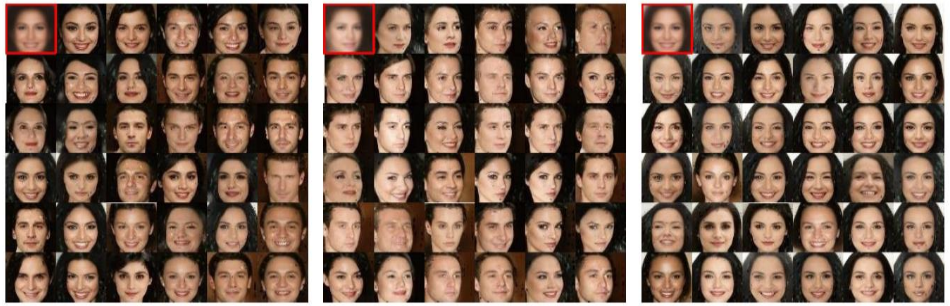
Experiments

Advantages of NDB score

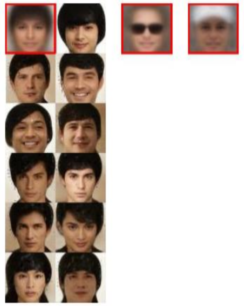
1. Visual insight into the mode collapse problem

Ex) BEGAN - failed to generate samples belonging to some of bins

<Over-allocated bins>



<Under-allocated bins>



2. Ability to perform inference

It provides a closed-form expression for inpainting and calculation of log likelihood

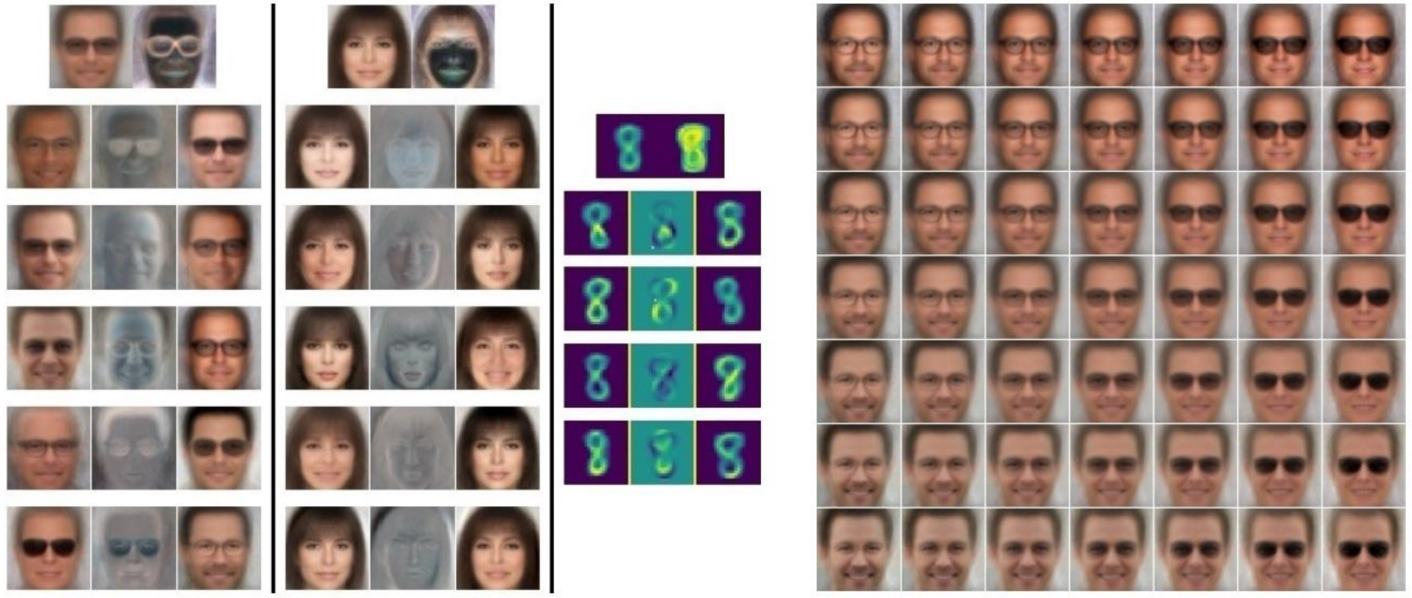
(a) (b)

Figure 7: Inference using the explicit MFA model: (a) Samples from the 100 images in CelebA with the lowest likelihood given our MFA model (outliers) (b) Image reconstruction – in-painting: In each row, the original image is shown first and then pairs of partially-visible image and reconstruction of the missing (black) part conditioned on the observed part.

Experiments

3. Disentangle the manifold

Even though the manifold of images is very nonlinear, GMM successfully models it as a combination of local linear manifolds

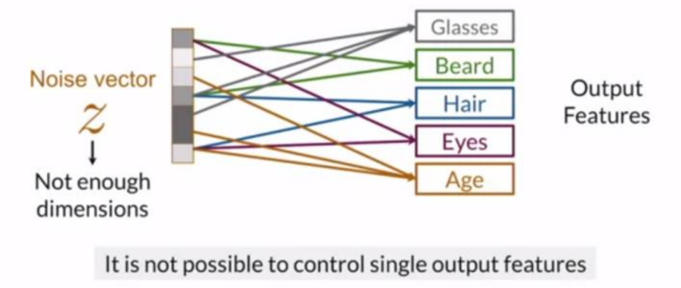


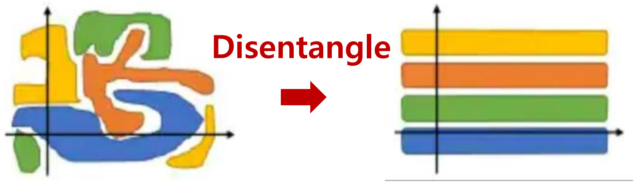
(a) (b)

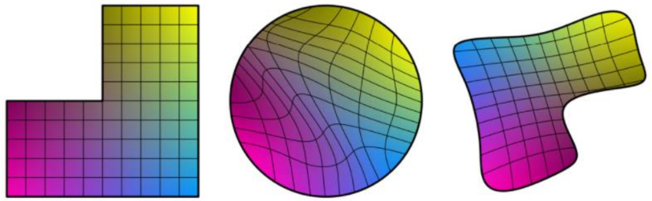
Figure 6: (a) Examples of learned MFA components trained on CelebA and MNIST: Mean image (μ) and noise variance (D) are shown on top. Each row represents a column-vector of the rectangular scale matrix A – the learned changes from the mean (showing vectors 1-5 of 10). The three images shown in row i are: $\mu + A^{(i)}$, $0.5 + A^{(i)}$, $\mu - A^{(i)}$. (b) Combinations of two column-vectors $(A^{(i)}, A^{(j)})$: z_i changes with the horizontal axis and z_j with the vertical axis, controlling the combination. Both variables are zero in the central image, showing the component mean.

Cf.) Entanglement

When trying to control one feature, others that are correlated change and Z-space entanglement makes controllability difficult.







(a) Distribution of features in training set (b) Mapping from $\mathcal{Z}$ to features (c) Mapping from $\mathcal{W}$ to features

Generating Sharp Images with GMMs

Sharpen Measure

Measures the relative energy of high-pass filtered versions of a set of images compared to the original image

<CelebA>

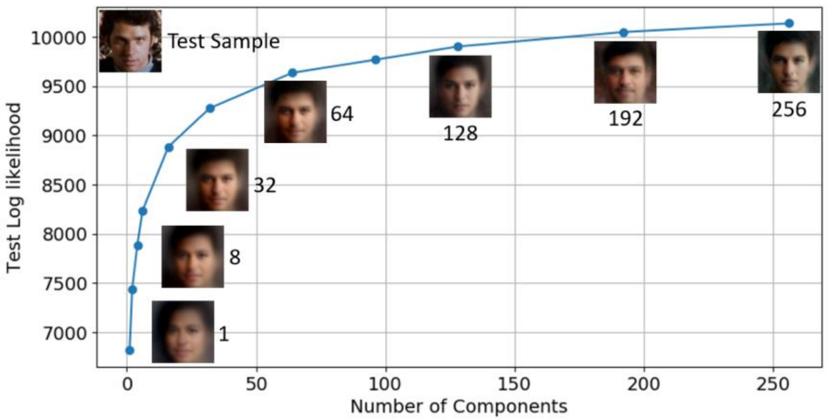
Source	NBD/K(100)	NBD/K(200)	NBD/K(300)	sharpness
TRAIN	0.01	0.03	0.03	-3.4
TEST	0.12	0.07	0.08	
MFA	**0.21**	**0.12**	**0.16**	-5.4
VAE	0.78	0.73	0.72	-
VAE-DFC	0.77	0.65	0.62	-
DCGAN	0.68	0.69	0.65	-
BEGAN	0.94	0.85	0.82	-
WGAN	0.76	0.66	0.62	-
WGAN-GP	0.42	0.32	0.27	**−3.9**

➡ GANs are better than GMMs in generating sharp images while GMMs are better at actually **capturing the statistical structure** and **enabling efficient inference**

Problem

Trade off between sharpness and overfitting

By increasing the number of components, sharpness increases up to that of GANs but this clearly overfits to the training data



Solution

1. Pairing GMM with a Conditional GAN

Add fine details by using pix2pix Conditional GAN

2. Adversarial GMM Training

Replace the WGAN-GP Generator network with a GMM Generator

$$x = \sum_{i=1}^{K} c_i(A_i z_1 + \mu_i + D_i z_2)$$

Generating Sharp Images with GMMs

Pairing GMM with a Conditional GAN

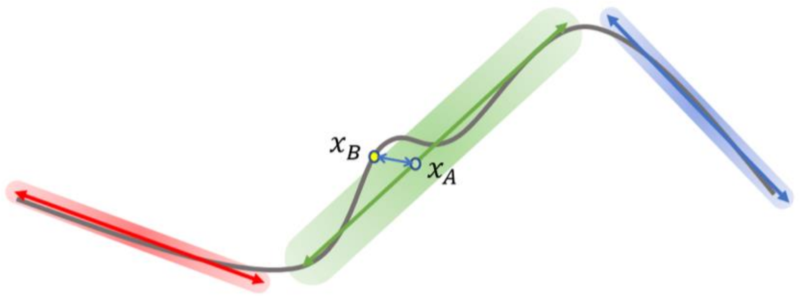
MFA+pix2pix

1. Generate for each training sample a matching image from MFA model
2. Find the most likely **component** $\mathbf{c}$ and a **latent variable** z that **maximizes the posterior** probability and generate the sample $\hat{x}$

$$(\widehat{c}, \widehat{z}) = \arg\max_{c,z} P(z|x, \mu_c, \Sigma_c)$$

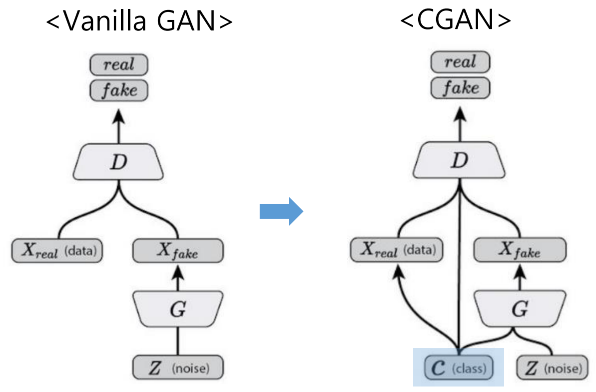
$$\widehat{x} = A_{\widehat{c}}\widehat{z} + \mu_{\widehat{c}}$$

3. Train pix2pix model on pairs for converting $\hat{x}$ to x
4. Apply learned pix2pix model to new images sampled from GMM to generate fine-detailed images



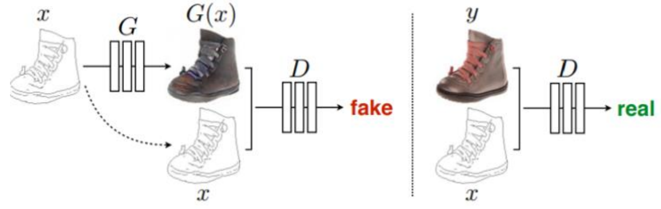
CGAN(Conditional GAN)

Put **class labels** as auxiliary information and condition on to both the Generator and Discriminator



pix2pix

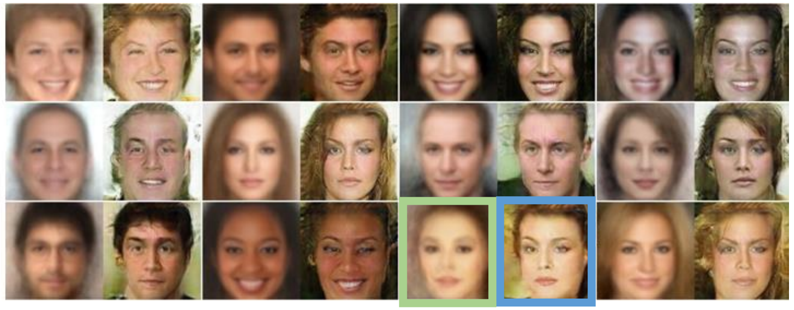
Use CGAN as a general-purpose solution to **image-to-image translation** problems



Generating Sharp Images with GMMs

Result

1. Pairing GMM with a Conditional GAN



random samples from MFA

matching samples generated by the conditional pix2pix

<CelebA>

Source	NBD/K(100)	NBD/K(200)	NBD/K(300)	sharpness
TRAIN	0.01	0.03	0.03	-3.4
TEST	0.12	0.07	0.08	
MFA	**0.21**	**0.12**	**0.16**	-5.4
MFA+pix2pix	0.34	0.34	0.33	**−3.5**
ADVERSARIAL MFA	0.33	0.30	0.22	-3.8
VAE	0.78	0.73	0.72	-
VAE-DFC	0.77	0.65	0.62	-
DCGAN	0.68	0.69	0.65	-
BEGAN	0.94	0.85	0.82	-
WGAN	0.76	0.66	0.62	-
WGAN-GP	0.42	0.32	0.27	-3.9

2. Adversarial GMM Training

Conclusion

Contributions

1. NDB

We present a new metric to evaluate generative models and it reveals GAN mode collapse

2. Full-image GMM

We show that it is possible to efficiently train GMMs on the same datasets that are usually used with GANs and the model can generates realistic samples, captures the underlying distribution, provides explicit representation of the statistical structure and allows inference unlike GANs

∴ We should focus on **efficient inference** and **accurate representation of statistical structure**, even at the expense of not generating the prettiest pictures

References

- **On GANs and GMMs**

https://arxiv.org/abs/1805.12462

https://idea-stat.snu.ac.kr/seminar/20200516/GAN_and_GMM.pdf

- **The EM Algorithm for Mixtures of Factor Analyzers**

http://www.cs.utoronto.ca/~hinton/absps/tr-96-1.pdf

- **Build Basic Generative Adversarial Networks (GANs)**

https://www.coursera.org/lecture/build-basic-generative-adversarial-networks-gans/challenges-with-controllable-generation-7hcuL