GAN-EM: GAN Based EM Learning Framework

Wentian Zhao[1*], **Shaojie Wang**[1*], **Zhihuai Xie**[2], **Jing Shi**[1] and **Chenliang Xu**[1]
[1]University of Rochester
[2]Tsinghua University
{wentian.zhao, shaojie.wang}@rochester.edu, xiezh16@mails.tsinghua.edu.cn
{jing.shi, chenliang.xu}@rochester.edu

Abstract

Expectation maximization (EM) algorithm is to find maximum likelihood solution for models having latent variables. A typical example is Gaussian mixture model (GMM) which requires Gaussian assumption, however, natural images are highly non-Gaussian so that GMM cannot be applied to perform image clustering task on pixel space. To overcome such limitation, we propose a GAN based EM learning framework that can maximize the likelihood of images and estimate the latent variables. We call this model GAN-EM, which is a framework for image clustering, semi-supervised classification and dimensionality reduction. In M-step, we design a novel loss function for discriminator of GAN to perform maximum likelihood estimation (MLE) on data with soft class label assignments. Specifically, a conditional generator captures data distribution for K classes, and a discriminator tells whether a sample is real or fake for each class. Since our model is unsupervised, the class label of real data is regarded as latent variable, which is estimated by an additional network (E-net) in E-step. The proposed GAN-EM achieves state-of-the-art clustering and semi-supervised classification results on MNIST, SVHN and CelebA, as well as comparable quality of generated images to other recently developed generative models.[1]

1 Introduction

Expectation maximization (EM) [Dempster *et al.*, 1977] is a traditional learning framework, which has various applications in unsupervised learning. A typical example is Gaussian mixture model (GMM), where data distribution is estimated by maximum likelihood estimate (MLE) under the Gaussian assumption in M-step, and soft class labels are assigned using Bayes rule in E-step. Although GMM has the nice property that likelihood increases monotonically, many previous studies [Dagher and Nachar, 2006] [Wainwright and Simoncelli, 1999] [Srivastava *et al.*, 2003] have shown that natural image intensities exhibit highly non-Gaussian behaviors so that GMM cannot be applied to image clustering on pixel space directly. The motivation of this work is to find an alternative way to achieve EM mechanism without Gaussian assumption.

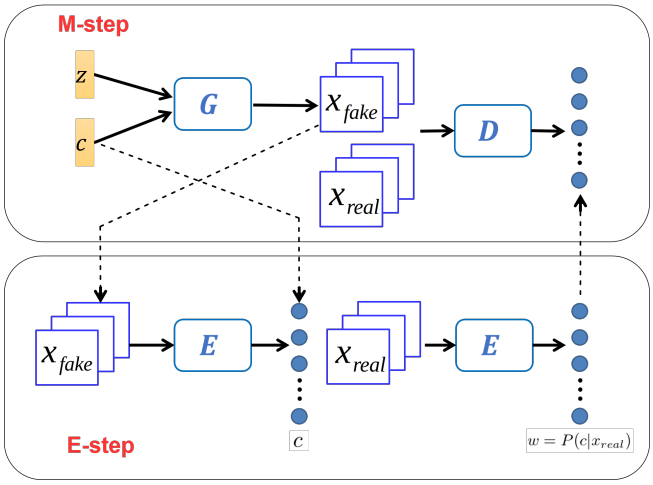


Figure 1: GAN-EM architecture. G: generator, D: discriminator, E: E-net, z: random noise, c: specified class, x_{real}: real images, and x_{fake}: generated images. G takes z and c as input and generates x_{fake}. The inverse function of G is approximated by E, which is trained with input x_{fake} and label c. D takes in both x_{real} and x_{fake} and outputs the probability of an input to be real for each class. E is tasked to make soft class assignments to all x_{real}.

Generative adversarial network (GAN) [Goodfellow *et al.*, 2014] has been proved to be powerful on learning data distribution. We propose to apply it in M-step to maximize the likelihood of data with soft class label assignments passed from E-step. It is easy for GAN to perform MLE as in [Goodfellow, 2017; Nowozin *et al.*, 2016], but to incorporate the soft class assignments into the GAN model in the meantime is rather difficult. To address this problem, we design a weighted binary cross entropy loss function for discriminator where the weights are the soft label assignments. In Sec. 4, we prove that such design enables GAN to optimize the Q function of EM algorithm. Since neural networks are not reversible in most cases, we could not use Bayes rule to compute the expectation analytically in E-step like that for

*The two authors contribute equally in this work.

[1]Supplementary material can be found at http://www.cs.rochester.edu/~cxu22/p/.

GMM. To solve this, we use generated samples to train another network, named E-net, then predict the soft class labels for real samples in E-step.

To evaluate our model, we perform the clustering task based on MNIST and achieve lowest error rate with all 3 different numbers of clusters: 10, 20 and 30, which are common settings in previous works. We also test the semi-supervised classification performance on MNIST and SVHN with partially labeled data, both results being rather competitive compared to recently proposed generative models. Especially, on SVHN dataset, GAN-EM outperforms all other models. Apart from the two commonly used datasets, we test our model on an additional dataset, CelebA, under both unsupervised and semi-supervised settings, which is a more challenging task because attributes of human faces are rather abstract. It turns out that our model still achieves the best results.

We make the following contributions: (1) We are the first to achieve general EM process using GAN by introducing a novel GAN based EM learning framework (GAN-EM) that is able to perform clustering, semi-supervised classification and dimensionality reduction; (2) We conduct thoughtful experiments and show that our GAN-EM achieves state-of-the-art clustering results on MNIST and CelebA datasets, and semi-supervised classification results on SVHN and CelebA.

2 Related Work

Image Clustering: Image classification has been well developed due to the advances of Convolutional Neural Network (CNN) [Krizhevsky *et al.*, 2012] in recent years. However, the excellent classification performance relies on large amounts of image labels. Deep models are far from satisfactory in scenarios where the annotations are insufficient. Therefore, image clustering is an essential problem in computer vision studies. Deep Embedded Clustering (DEC) [Xie *et al.*, 2016] is proposed to learn feature representations and cluster assignments using deep neural networks. Another study on deep clustering is [Peng *et al.*, 2016], which aims to cluster data into multiple categories by implicitly finding a subspace to fit each class.

Deep EM: A successful combination of neural networks and EM is the neural expectation maximization (N-EM) [Greff *et al.*, 2017]. N-EM trains the parameters of EM using a neural network, which derives a differentiable clustering model, and is used for unsupervised segmentation, where N-EM can cluster constituent objects. Neural-EM aims to learn the parameters of the mixture models using neural networks. However, in our GAN-EM, the goal is to learn the weights of neural networks, i.e., the generator, whose output is the generated samples based on the currently learned mixture models. Therefore, the neural network plays a different role between these two models. Banijamali et al. [Banijamali *et al.*, 2017] use generative mixture of networks (GMN) to simulate the GMM. They first use K-means to obtain prior knowledge of the dataset, and then treat each network as a cluster. Variational deep embedding (VaDE) [Jiang *et al.*, 2017] combines GMM with variational autoencoder (VAE), which keeps the Gaussian assumption. In M-step, VaDE maximizes the lower bound on the log-likelihood given by Jensen inequality. In E-step, a neural network is used to model the mapping from data to class assignment.

GAN Clustering: Generative adversarial networks evolute through the years. At the outset, the vanilla GAN [Goodfellow *et al.*, 2014] could not perform clustering or semi-supervised classification. Springenberg [Springenberg, 2015] proposed categorical generative adversarial networks (CatGAN), which can perform unsupervised learning and semi-supervised learning. They try to make the discriminator be certain about the classification of real samples, and be uncertain about that of generated samples, and apply the opposite for the generator. Moreover, their model is based on the assumption that all classes are uniformly distributed, while we relax such assumption in our model. InfoGAN [Spurr *et al.*, 2017], adversarial autoencoder [Makhzani *et al.*, 2015], feature matching GAN [Salimans *et al.*, 2016] and pixelGAN autoencoder [Makhzani and Frey, 2017] are all GAN variants that can do clustering tasks in unsupervised manner. Our proposed GAN-EM is quite different from the previous GAN variants, in which we fit GAN to the EM framework which has been proved that the likelihood of data increases monotonically. A similar work to ours is [Yang and Zhou, 2018]. Similar to GMM, they fit GANs into the GMM (GANMM). In GANMM, hard label assignment strategy limits the model to K-means, which is an extreme case of EM for mixture model [Bishop, 2006]. We have two main differences from their work. First, we use soft label assignment, rather than the hard assignment in GANMM. To the best of our knowledge, our work is the first to achieve general EM process using GAN. Second, we use only one GAN, rather than K GANs where K is the number of clusters. Our model is scalable with respect of K. The drawback of using multiple GANs will be discussed in Sec. 4.3. Experimental results show that our GAN-EM outperforms GANMM by a big margin.

3 GAN-EM

The overall architecture of our model is shown in Fig. 1. We first clarify some denotations. ψ is the parameters for GAN, which includes ψ_G for generator and ψ_D for discriminator, and ϕ is the parameters for multinomial prior distribution, which is composed of $\phi_i = P(c = i; \phi)$. $\theta = \psi, \phi$ stands for all the parameters for the EM process. We denote the number of clusters by K, and the number of training samples by N.

3.1 M-step

The goal of M-step is to update the parameters θ that maximizes the lower bound of log-likelihood $\log P(x; \theta)$ given the soft class assignment $w = P(c|x; \theta^{old})$ provided by E-step (where w is a $N \times K$ matrix). θ consists of ϕ and ψ. Updating ϕ is simple, since we can compute the analytical solutions for each ϕ_i: $\phi_i^* = N_i/N$, where N_i is the number of samples for the i-th cluster. Details are in Sec. 4.

To update ψ, we extend GAN to a multi-class version to learn the mixture models $P(x|c = i; \psi)\ for\ i = 1 \dots K$. The vanilla GAN [Goodfellow *et al.*, 2014] is modified in several ways as follows.

Generator: Similar to conditional GAN [Mirza and Osindero, 2014], apart from the random noise z, class label infor-

mation c is also added to the input of the generator. With c as input, we specify the generator to generate the c-th class images. This makes our generator act as K generators.

Discriminator: Different from the vanilla GAN that uses only one output unit, we set K units in the output layer.

Loss Function: Here, we only give the form of the loss functions, the derivation of which will be discussed in Sec. 4.1 in detail. L_G and L_D are loss functions of the generator and the discriminator respectively. We have:

$$L_G = -\frac{1}{2} \mathop{\mathbb{E}}_{c\sim U} \mathop{\mathbb{E}}_{z\sim\mathcal{N}} \exp\{\sigma^{-1}[D_c(G(z,c))]\} \ , \quad (1)$$

$$L_D = \mathop{\mathbb{E}}_{c\sim U} \mathop{\mathbb{E}}_{z\sim\mathcal{N}} \sum_{i=1}^{K} \log(1 - D_i(G(z,c))) + \mathop{\mathbb{E}}_{x\sim P_r} \sum_{i=1}^{K} w_i \log(D_i(x)) \ , \quad (2)$$

where z is random noise, c is the specified class of images to be generated, U and $\mathcal{N}$ are uniform distribution and Gaussian distribution respectively, $x \sim P_r$ is to sample images from the real data distribution. $G(z,c)$ stands for generated images, σ for sigmoid activation function, and $w_i = P(c = i|x;\theta^{old})$ for the i-th class label assignments, which comes from the previous E-step. Here, $D_i(\cdot)$ denotes the i-th output unit's value of the discriminator. Notice that L_G is a modified form for the loss of the generator so that GAN can perform likelihood maximization [Goodfellow, 2017]. The first term of L_D means that all output units are expected to give a low probability to fake images. Conversely, the second term guides the discriminator to output a high probability for all real ones, while the loss for each output unit is weighted by the soft label assignment w passed from E-step.

3.2 E-step

In the unsupervised manner, class label c for real data cannot be observed and is regarded as latent variable. Thus the goal of E-step is to estimate such latent variable, which is normally obtained using Bayes rule given the parameters learned from the previous M-step. Therefore, we have:

$$P(c = i|x;\theta) = \frac{P(x|c = i;\psi)P(c = i;\phi)}{\sum_j P(x|c = j;\psi)P(c = j;\phi)} \ , \quad (3)$$

where $P(x|c = i;\psi)$ represents the distribution of data in class i given by the generator network, and the denominator is the normalization term. However, Eq. 3 is hard to calculate since neural network is not invertible.

To circumvent such problem, we introduce another neural network, called E-net, to fit the distribution expressed on the left hand side of Eq. 3. To train the E-net, we first generate samples from the generator, where the number of generated samples for cluster i is subject to ϕ_i because $P(c = i|x;\theta)$ is proportional to ϕ_i according to Eq. 3 (remind that $\phi_i = P(c = i;\phi)$). After the E-net is well trained, it approximates the parameters θ and act as an inverse generator. Similar approach is also used in BiGAN [Donahue *et al.*, 2016], where they also prove that $E = G^{-1}$ almost everywhere. However, the goal of BiGAN is feature learning, which is different from ours.

Algorithm 1 GAN-EM

Initialization: $w_i = 1/K \ for \ i = 1 \dots K$
for $iteration = 1 \dots n$ **do**
  update ϕ: ▷ M-step
    $\phi_i = N_i/N \ for \ i = 1 \dots K$
  update ψ:
    for $epoch = 1 \dots p$ **do**
      train GAN: $\min_{\psi_G} L_G$, $\max_{\psi_D} L_D$
    end for
  for $epoch = 1 \dots q$ **do** ▷ E-step
    Sample a batch of $c \sim \phi$, and obtain $G(z,c)$
    train E-net: $\min_{\eta} L_E$ (η: weights of E-net[2])
  end for
  update label assignment: $w = E(x_{real})$
end for
Predict: $w = E(x_{real})$

Specifically, as shown in Fig. 1 we take the output of the generator, i.e., $x_{fake} = G(z,c)$, as the input of the E-net, and take the corresponding class c as the output label. Therefore, we can learn the approximate distribution of the left hand side of Eq. 3, and thus obtain soft class assignments:

$$w = P(c|x_{real};\theta) \ , \quad (4)$$

then feed them back to M-step. The E-net takes the following loss:

$$L_E = \mathop{\mathbb{E}}_{z\sim\mathcal{N}} \mathop{\mathbb{E}}_{c\sim\phi} \mathrm{CE}\{E(G(z,c)), u\} \ , \quad (5)$$

where $\mathrm{CE}\{\cdot,\cdot\}$ stands for cross-entropy function, u is a one-hot vector that encodes the class information c, and $E(\cdot)$ is the output of E-net. The trained E-net is then responsible for giving the soft class assignment w for real images.

3.3 EM Algorithm

The bridge between M-step and E-step is the generated fake images with their conditional class labels and the output w produced by E-net. So far, the whole training loop has been built up. Then we can train M-step and E-step alternatively to achieve the EM mechanism without Gaussian assumption. In the ideal case where neural network reaches global minima, the convergence can be guaranteed. The solution of neural network optimization is local minima instead of global minima, which cannot guarantee the likelihood to increase monotonically. However, the experiment results show that the error rate for clustering decreases monotonically and converges within 20 epochs in most cases, which will be shown in Sec. 5.2. We start the training with M-step, where w is initialized with uniform distribution. The pseudo code is in Algorithm 1.

4 Theoretical Analysis

This section mainly focuses on the theoretical analysis of GAN in M-step. We first show how our model works with K GANs by deriving the objective functions. Then, we simplify the model by using only one GAN.

[2]Because E-net aims to learn an inverse function of generator, η is not independent with ψ. Thus, we could not say that η is the

4.1 Background

In M-step, we aim to maximize the Q function [Bishop, 2006] expressed as:

$$Q(\theta;\theta^{old}) = \mathop{\mathbb{E}}_{x\sim P_r} \sum_{i=1}^{K} P(c=i|x;\theta^{old}) \log P(x, c=i;\theta)$$

$$= \mathop{\mathbb{E}}_{x\sim P_r} \sum_{i=1}^{K} w_i \log[P(x|c=i;\psi)P(c=i;\phi)] \ . \tag{6}$$

Furthermore, we can write Eq. 6 as the sum of two terms $Q^{(1)}$ and $Q^{(2)}$:

$$Q^{(1)} = \mathop{\mathbb{E}}_{x\sim P_r} \sum_{i=1}^{K} w_i \log P(x|c=i;\psi) \ , \tag{7}$$

$$Q^{(2)} = \mathop{\mathbb{E}}_{x\sim P_r} \sum_{i=1}^{K} w_i \log \phi_i \ . \tag{8}$$

Remind that $\phi_i = P(c=i;\phi)$ is explained in the previous section. These two terms are independent with each other, so they can be optimized separately.

We first optimize Eq. 8. Since ϕ_i is irrelevant to x, we can ignore the expectation because it only introduces a constant factor. With the constraint $\sum_i \phi_i = 1$, the optimal solution for $Q^{(2)}$ is $\phi_i^* = N_i/N \ (i = 1 \dots K)$, where N_i is the sample number of the i-th cluster (N_i is not an integer necessarily because the class label assignment is in a soft version), and N is the sample number of the whole dataset. Derivation can be seen in Appendix A.2.

Then we consider Eq. 7. We are expecting to employ GANs to optimize $Q^{(1)}$. Each term in the summation in $Q^{(1)}$ is independent with each other because we are currently considering K separate GANs. Therefore, we can optimize each term in $Q^{(1)}$, rewrite it as

$$Q_i^{(1)} = \mathop{\mathbb{E}}_{x\sim P_r} w_i \log P_{f_i}(x|c=i;\psi_i) \ , \tag{9}$$

to fit the single GAN model, and sum them up in the end. Here, ψ_i is the parameters of the i-th GAN, P_r stands for the distribution of all real data, and P_{f_i} stands for the fake data distribution for each cluster $c = i$.

For convenience, we introduce a new distribution $P_{r_i} = \frac{1}{Z} w_i P_r$, where $\frac{1}{Z}$ is the normalization factor, $Z = \int_x w_i P_r dx$. Substitute P_{r_i} into Eq. 9 and we have:

$$Q_i^{(1)} = \mathop{\mathbb{E}}_{x\sim P_{r_i}} \log P_{f_i}(x|c=i;\psi_i) \ . \tag{10}$$

The constant coefficient Z is dropped for convenience.

4.2 Objectives

In this subsection, we aim to show how our design for M-step (i.e. Eq. 1 and Eq. 2) is capable of maximizing Eq. 10 which takes exactly the form of likelihood. This is feasible due to the following two facts [Goodfellow, 2017]:

1. MLE is equivalent to minimizing the KL-divergence between the real distribution and generated distribution;
2. When discriminator is optimal, we can modify the loss function for generator (as will be shown in Eq. 14) so that the optimization goal of GAN is to minimize KL divergence.

Maximizing Eq. 10 is equivalent to minimizing $KL(P_{r_i} \parallel P_{f_i})$ according to fact 1. Then we show that how GANs can be tasked to minimize such KL-divergence by introducing minor changes.

According to fact 2, only when discriminator is optimized can we modify GAN to minimize KL divergence. Therefore, we consider the optimum of discriminators. With $P_{r_i} \sim w_i P_r$, the loss function of the i-th discriminator given any generator (denoted by fake data) is:

$$L_i = \mathop{\mathbb{E}}_{x\sim P_r} w_i \log D_i(x) + \mathop{\mathbb{E}}_{x\sim P_{f_i}} \log(1 - D_i(x))$$

$$= \mathop{\mathbb{E}}_{x\sim P_{r_i}} \log D_i(x) + \mathop{\mathbb{E}}_{x\sim P_{f_i}} \log(1 - D_i(x)) \ , \tag{11}$$

where $D_i(\cdot)$ is the i-th discriminator, similar to vanilla GAN [Goodfellow *et al.*, 2014]. We show that when the discriminators reach optimum, L is equivalent to the sum of JS divergence. The following corollary is derived from Propositional 1 and Theorem 1 in [Goodfellow *et al.*, 2014].

Corollary 1. *Equation 11 is equivalent to the JS divergence between real distribution and generated distribution for each cluster when discriminators are optimal, i.e.*

$$L_i = -(w_i + 1)\log 2 + 2JSD(P_{r_i} \parallel P_{f_i}) \ , \tag{12}$$

and the optimal D_i^* for each cluster is:

$$D_i^*(x) = \frac{P_{r_i}(x)}{P_{r_i}(x) + P_{f_i}(x)} \ . \tag{13}$$

Proof. See Appendix A.1. □

If we use the same loss function for generator as the vanilla GAN, the JS divergence in Eq. 12 will be minimized. However, we aim to make GAN to minimize the KL divergence, $KL(P_{r_i} \parallel P_{f_i})$, for each cluster so as to achieve the goal of maximizing Eq. 10. In Corollary. 1, we already have the optimal discriminator given fixed generator. Therefore, according to fact 2, we need to modify the loss function for the generator as:

$$-\frac{1}{2}\mathbb{E}_z \exp \sigma^{-1}[D_i(G_i(z))] \ , \tag{14}$$

where σ is the sigmoid activation function in the last layer of the discriminator, and $G_i(\cdot)$ is the output of i-th generator.

Now we have derived the objectives of single generator and discriminator, and we need to ensemble them up as a whole model. Since we are currently using K GANs, we only need to sum up Eq.14 for the loss of generators:

$$-\frac{1}{2}\sum_{i=1}^{K} \mathbb{E}_z \exp \sigma^{-1}[D_i(G_i(z)] \ , \tag{15}$$

parameter of EM process. In other words, η is not part of θ.

and sum up Eq. 11 for the loss of discriminators:

$$\sum_{i=1}^{K} \mathop{\mathbb{E}}_{x\sim P_r} w_i \log D_i(x) + \mathop{\mathbb{E}}_{x\sim P_{f_i}} \log(1 - D_i(x)) \ . \qquad (16)$$

Here, Eq. 16 is equivalent to Eq. 2 since $x \sim P_{f_i}$ is generated by generator G. The derivation from Eq. 15 to Eq. 1 will be introduced in Sec. 4.3.

4.3 Single GAN v.s. Multiple GANs

We have shown that K GANs can be tasked to perform MLE in M-step of EM. In fact, using such many GANs is intractable since the complexity of the model grows along with cluster numbers proportionally. Moreover, data is separated per cluster and distributed to different GANs, which could not make the most use of data for individual GAN efficiently.

Single Generator

For the generator part, we employ a conditional variable c [Mirza and Osindero, 2014] to make a single generator act as K generators. Then the final loss function for generator is exactly Eq. 1.

Single Discriminator

In our work, instead of applying K discriminators, we use a single discriminator with K output units. Each output has individual weights in the last fully connected layer. The preceding layers of the discriminator is shared since the convolutional layers play a role in extracting features that are often in common among different clusters of data.

To this end, we denote the last fully connected layer of the discriminator by function $\tilde{D}$, and all other layers by function f, then we have: $D_i(x) = \widetilde{D}_i(f(x)) = \widetilde{D}_i(\tilde{x}) \ , \ \forall i \in 1, \ldots, K$, where $\tilde{x} = f(x)$ stands for the features learned by f. The objective still holds the form of Eq. 16, but the meaning of D_i has changed from the i-th discriminator to the i-th output unit of $D(x)$ that stands for the probability of x belonging to i-th cluster. Till now, we have finished deriving the loss functions for our proposed model.

In practice, to speed up the learning, we add an extra output unit for the discriminator using a binary cross entropy function, which is the same as vanilla GAN, only used for distinguishing all the real data and all the fake data regardless of class labels.

5 Experiments

We perform unsupervised clustering on MNIST [LeCun *et al.*, 1989] and CelebA [Liu *et al.*, 2015] datasets, and semi-supervised classification on MNIST, SVHN [Netzer *et al.*, 2011] and CelebA datasets. We also evaluate the capability of dimensionality reduction by adding an additional hidden layer to the E-net. The results show that our model achieve state-of-the-art results on various tasks. Meanwhile, the quality of generated images are also comparable to many other generative models. The training details and network structures are illustrated in Appendix B.

(a) MNIST (unsupervised) (b) SVHN (1000 labels)

Figure 2: Clustering and semi-supervised classification results by GAN-EM.

5.1 Implementation Details

We apply RMSprop optimizer to all 3 networks G, D and E with learning rate 0.0002 (decay rate: 0.98). The random noise of generator is in uniform distribution. In each M-step, there are 5 epoches with a minibatch size of 64 for both the generated batch and the real samples batch. We use a same update frequency for generator and discriminator. For E-step, we generate samples using well trained generator with batch size of 256, then we apply 1000 iterations to update E-net.

5.2 Unsupervised Clustering

GAN-EM achieves state-of-the-art results on MNIST clustering task with 10, 20 and 30 clusters. We evaluate the error rate based on the following metric which has been used in most other clustering models in Tab. 1:

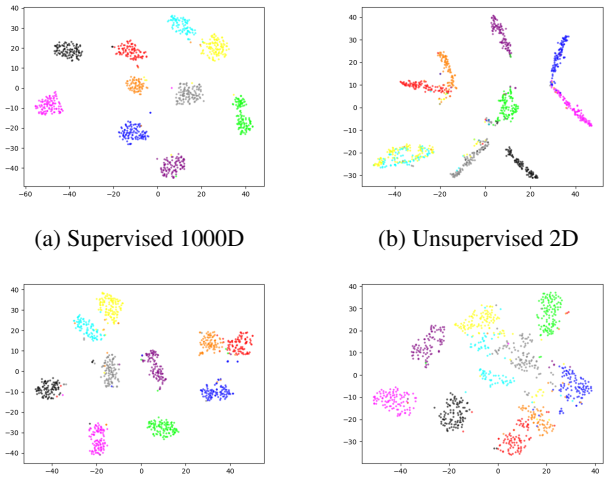
$$Err = 1 - \max_{m\in\mathcal{M}} \frac{\sum_{i=1}^{N} \mathbb{1}\{l_i = m(c_i)\}}{N} \ , \qquad (17)$$

where c_i is for the predicted label of the i-th sample, l_i for ground truth label, $\mathcal{M}$ for all one-to-one mapping from ground truth labels to predicted labels of all samples in the cluster, and N for number of all samples. The experimental results are shown in column 1 of Tab. 1.

Since different models have different experiment setups, there is no uniform standard for the clustering numbers under the unsupervised setting. MNIST dataset has 10 different digits, so naturally we should set the number of clusters $K = 10$. However, some models such as CatGAN, AAE, and PixelGAN use $K = 20$ or 30 to achieve better performance, since the models might be confused by different handwriting styles of digits. In other words, the more clusters we use, the better performance we can expect. To make fair comparisons, we conduct experiments with $K = 10, 20, 30$ respectively. Also, all models in Tab. 1 take input on the 784-dimension raw pixel space. Note that both K-means and GMM have high error rates (46.51 and 32.61) on raw pixel space, since MNIST is highly non-Gaussian, which is an approximate Bernoulli distribution with high peaks at 0 and 1. The huge margin achieved by GAN-EM demonstrates that the relaxation of Gaussian assumption is effective on clustering problem.

	MNIST (Unsupervised)	MNIST (100 labels)	MNIST (1000 labels)	SVHN (1000 labels)	CelebA (Unsupervised)	CelebA (100 labels)
K-means	46.51[1]	-	-	-	-	-
GMM	32.61(±0.06)[1]	-	-	-	-	-
DEC	15.7[1]	-	-	-	-	-
VAE	-	3.33(±0.14)	2.40(±0.02)	36.02(±0.10)	-	45.38
AAE	4.10(± 1.13)[3]	1.90(±0.10)	1.60(±0.08)	17.70(±0.30)	42.88	31.03
CatGAN	4.27[2]	1.91(±0.10)	1.73(±0.28)	-	44.57	34.78
InfoGAN	5.00[4]	-	-	-	-	-
Improved GAN	-	**0.93(±0.06)**	-	8.11(±1.30)	-	-
VaDE	5.54[1]	-	-	-	43.64	-
PixelGAN	5.27(±1.81)[3]	1.08(±0.15)	-	6.96(±0.55)	44.27	32.54
GANMM	35.70(±0.45)[1]	-	-	-	49.32	-
GAN-EM	4.20(± 0.51)[1]	1.09(± 0.18)	**1.03(± 0.15)**	**6.05(±0.26)**	**42.09**	**28.82**
	4.04(± 0.42)[2]	-	-	-	-	-
	3.97(± 0.37)[3]	-	-	-	-	-

Table 1: Experiment results of different models. [1]Clustering with $K = 10$. [2]$K = 20$. [3]$K = 30$. [4]Cluster number not specified.



(a) Supervised 1000D (b) Unsupervised 2D

(c) Unsupervised 100D (d) Unsupervised 1000D

Figure 3: Representation of unsupervised dimensionality reduction on MNIST. Each color denotes one class of digit.

Our proposed GAN-EM has the lowest error rate 3.97 with $K = 30$. Moreover, When $K = 20$, GAN-EM still has better results than other models. With 10 clusters, GAN-EM is only outperformed by AAE, but AAE achieves the error rate of 4.10 using 30 clusters. VaDE also has a low error rate under the setting of $K = 10$, yet still higher than that of our model under the same setting. GANMM has a rather high error rate[3], while GAN-EM achieves the state-of-the-art clustering results on MNIST, which shows that the clustering capability of EM is much superior to that of K-means. We also plot the test error rate curve of the training process as shown in Fig. 4, where each curve stands for one separate training

[3]When the feature space is reduced to 10 dimensions using SAE, GANMM achieves an error rate of 10.92 (±0.15) with $K = 10$ [Yang and Zhou, 2018].

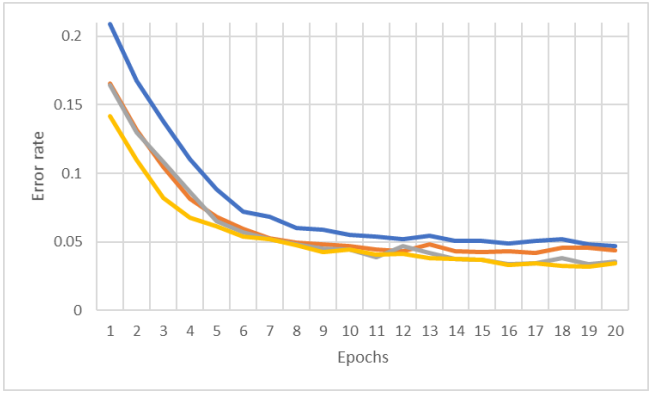


Figure 4: Convergence of test error rate of GAN-EM on MNIST.

($K = 10$), to show the convergence of the training.

Then we test our model on CelebA dataset using the same strategy as stated above. CelebA is a large-scale human face dataset that labels faces by 40 binary attributes. Totally unsupervised clustering on such tasks is rather challenging because these face attributes are so abstract that it is difficult for CNN to figure out what features it should extract to cluster the samples. Tab. 1 (column 5) illustrates the average error rate of different models on all 40 CelebA attributes. We also list detailed results of GAN-EM on all the 40 attributes in Appendix A.3. We achieve the best overall result for unsupervised clustering, and we demonstrate two representative attributes on which we achieve lowest error rates, i.e. hat (29.41) and glasses (29.89), in Figs. 5, where the two sets of images are generated by the generator given two different conditional labels respectively. The details of strategies for selecting samples is illustrated in supplementary material Appendix B.3.

5.3 Semi-supervised Classification

It is easy to extend our model to semi-supervised classification tasks where only a small part of samples' labels are

Figure 5: Unsupervised feature learning on CelebA: (left) 'hat'; (right) 'glasses'.

known, while the remainders are unknown. We use almost the same training strategies as clustering task except that we add the supervision to the E-net in every E-step. The method is that at the end of the E-net training using generated fake samples, we train it by labeled real samples. Then the loss function takes the form $L_E = \text{CE}\{E(x_{real}), u\}$, where u is the one-hot vector that encodes the class label c. Once the E-net has an error rate below ϵ on the labeled data, we stop the training, where ϵ is a number that is close to zero (e.g. 5% or 10%) and can be tuned in the training process. The reason why ϵ is greater than zero is to avoid over-fitting.

We evaluate the performance of semi-supervised GAN-EM on MNIST, SVHN and CelebA datasets. As shown in Tab. 1 (column 2, 3, 4, 6), our GAN-EM achieves rather competitive results on semi-supervised learning on all three datasets (state-of-the-art on SVHN and CelebA). The images generated by GAN-EM on SVHN are shown in Fig. 2b. On MNIST, when we use 100 ground truth labels for the semi-supervised classification, the error rate is 1.09, which is only 0.16 higher than the top-ranking result by improved GAN, and when 1000 ground truth labels are used, GAN-EM achieves the lowest error rate 1.03. On SVHN dataset, 1000 labels are applied as other models do, and we achieve state-of-the-art result with an error rate of 6.05. For CelebA, the number of ground truth labels is set to 100, and our model outperforms all other models with respect to average error rate on all 40 attributes.

5.4 Dimensionality Reduction

We can easily modify our GAN-EM model to perform dimensionality reduction by inserting a new layer r with k hidden units to the E-net (k is the number of dimension that we want to transform to). Layer r lays right before the output layer. Then, we can use exactly the same training strategy as the unsupervised clustering task. Once the training converges, we consider the E-net as a feature extractor by removing the output layer. Then we feed the real samples to the E-net and take the output on layer r as the extracted features after dimensionality reduction. Three different dimensions, i.e. 1000, 100 and 2, are selected for test on the MNIST dataset. We also apply t-SNE [Hinton, 2008] technique to project the dimensionality reduced data to 2D for visualization purpose.

Fig. 3a shows the supervised feature learning result. Figs. 3b, 3c and 3d are unsupervised data dimensionality reduction results with three different dimensions. We can see that our model can deal with all cases very well. Most different digits have large gap with each other and the same digits are clustered together compactly.

6 Conclusion

In this paper, we propose a novel GAN-EM learning framework that embeds GAN into EM algorithm to do clustering, semi-supervised classification and dimensionality reduction. We achieve state-of-the-art results on MNIST, SVHN and CelebA datasets with competitive fidelity of generated images. Although all our experiments are performed based on vanilla GAN, GAN-EM framework can also be embedded by many other GAN variants and better results are expected.

Acknowledgements

This work was supported in part by NSF IIS 1741472, IIS 1813709, and CHE 1764415. This article solely reflects the opinions and conclusions of its authors and not the funding agents.

References

[Banijamali *et al.*, 2017] Ershad Banijamali, Ali Ghodsi, and Pascal Poupart. Generative mixture of networks. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 3753–3760, 2017.

[Bishop, 2006] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[Dagher and Nachar, 2006] Issam Dagher and Rabih Nachar. Face recognition using IPCA-ICA algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(6):996–1000, 2006.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[Donahue *et al.*, 2016] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[Goodfellow, 2017] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[Greff *et al.*, 2017] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6694–6704, 2017.

[Hinton, 2008] G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9(2):2579–2605, 2008.

[Jiang *et al.*, 2017] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI, Melbourne, Australia*, pages 1965–1972, 2017.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[LeCun *et al.*, 1989] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[Liu *et al.*, 2015] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[Makhzani and Frey, 2017] Alireza Makhzani and Brendan J. Frey. Pixelgan autoencoders. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1972–1982, 2017.

[Makhzani *et al.*, 2015] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[Nowozin *et al.*, 2016] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 271–279, 2016.

[Peng *et al.*, 2016] Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In *IJCAI*, pages 1925–1931. IJCAI/AAAI Press, 2016.

[Salimans *et al.*, 2016] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234, 2016.

[Springenberg, 2015] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *CoRR*, abs/1511.06390, 2015.

[Spurr *et al.*, 2017] Adrian Spurr, Emre Aksan, and Otmar Hilliges. Guiding infogan with semi-supervision. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part I*, pages 119–134, 2017.

[Srivastava *et al.*, 2003] A. Srivastava, A. B. Lee, Eero P. Simoncelli, and S.-C. Zhu. On advances in statistical modeling of natural images. *Journal of Mathematical Imaging and Vision*, 18(1):17–33, 2003.

[Wainwright and Simoncelli, 1999] Martin J. Wainwright and Eero P. Simoncelli. Scale mixtures of gaussians and the statistics of natural images. In *NIPS*, pages 855–861. The MIT Press, 1999.

[Xie *et al.*, 2016] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 478–487, 2016.

[Yang and Zhou, 2018] Yu Yang and Wen-Ji Zhou. Mixture of gans for clustering. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI, Stockholm, Sweden*, 2018.