Neurocomputing 361 (2019) 126–136

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Label-removed generative adversarial networks incorporating with K-Means

Ce Wang[a], Zhangling Chen[b,*], Kun Shang[c], Huaming Wu[b]

[a] Center for Combinatorics, Nankai University, Tianjin 300071, PR China
[b] Center for Applied Mathematics, Tianjin University, Tianjin 300072, PR China
[c] College of Mathematics and Econometrics, Hunan University, Changsha, Hunan 410082, PR China

ARTICLE INFO

Article history:
Received 23 January 2019
Revised 10 April 2019
Accepted 2 June 2019
Available online 13 July 2019

Communicated by Dr. Yongsheng Dong

Keywords:
Un-conditional generative adversarial networks
K-Means
Metric learning

ABSTRACT

Generative Adversarial Networks (GANs) have achieved great success in generating realistic images. Most of these are conditional models, although acquisition of class labels is expensive and time-consuming in practice. To reduce the dependence on labeled data, we propose an un-conditional generative adversarial model, called K-Means-GAN (KM-GAN), which incorporates the idea of updating centers in K-Means into GANs. Specifically, we redesign the framework of GANs by applying K-Means on the features extracted from the discriminator. With obtained labels from K-Means, we propose new objective functions from the perspective of deep metric learning (DML). Distinct from previous works, the discriminator is treated as a feature extractor rather than a classifier in KM-GAN. Meanwhile, the utilization of K-Means makes features of the discriminator more representative. Experiments are conducted on various datasets, such as MNIST, Fashion-10, CIFAR-10 and CelebA, and show that the quality of samples generated by KM-GAN is comparable to some conditional generative adversarial models.

1. Introduction

Generative modeling has been an active but challenging research field in traditional machine learning because of the intractability of many probabilistic computations arising in approximating maximum likelihood estimation (MLE). To avoid these computations, Generative Adversarial Network (GAN) [1] greatly improves the quality of generated images by implicitly modeling the target distribution via neural networks instead of approximation of intractable likelihood functions in capturing data distribution. To better utilize the information of data structure in labeled data, Conditional GAN (CGAN) [2] feeds real labels along with images and generates more realistic images. Unfortunately, CGAN and subsequent extensions [3–7] suffer from a challenge that they require large amounts of labeled data which is expensive or even impossible to acquire in practice.

To decrease the dependence of GANs on labeled data, it would be nicer to find a substitution to replace the role of real labels. It is well known that representation learning enables machine learning models to get more information about data structure and class distribution. A commonly and widely used technique in representation learning is employing K-Means. Recent works [8–11] have improved clustering results via jointly training K-Means and deep neural networks. By fusing K-Means with the powerful nonlinear expressiveness of neural networks, they get “K-Means-friendly” [9] representations, i.e., features that are more representative for clustering tasks. But most of these neural networks are realized by a pre-trained auto-encoder on large-scale datasets, such as ImageNet, which means they still utilize prior knowledge (real-label) as supervision.

Inspired by the success of jointly training neural networks and K-Means on clustering tasks, Variational deep embedding (VaDE) [12] and Joint Generative MomentMatching Network (JGMMN) [13] instead combine generative models with clustering methods, and achieve competitive results not only on clustering, but also on generating tasks. More specifically, VaDE proposes continuous clustering objectives for Variational Autoencoder (VAE) [14] and JGMMN augments original loss functions of Generative Moment Matching Networks (GMMN) [15] with regularization terms to constrain latent variables. On the other hand, authors of [16] perform K-Means on features of the top layer of discriminators in GAN and Info-GAN [17], respectively, and show that features of Info-GAN are obviously more “K-Means-friendly” than those of regular GAN. This implies that constraints on the latent space of GANs induce more representative features. Furthermore, extensions of GANs [18,19] achieve state-of-the-art results on clustering by fusing GANs with clustering methods. Although these works have

[*] Corresponding author.
E-mail address: zhanglingchen0214@tju.edu.cn (Z. Chen).

https://doi.org/10.1016/j.neucom.2019.06.041
0925-2312/© 2019 Published by Elsevier B.V.

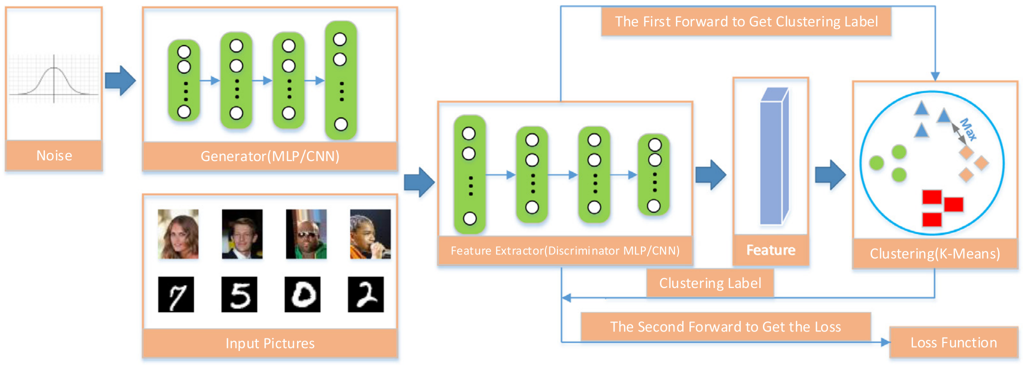

Noise
Generator(MLP/CNN)
Input Pictures
Feature Extractor(Discriminator MLP/CNN)
Feature
Clustering(K-Means)
Max
The First Forward to Get Clustering Label
Clustering Label
The Second Forward to Get the Loss
Loss Function

Fig. 1. The framework of KM-GAN. Notice that the first forward pass is to get the clustering labels. In the second forward pass, the clustering labels and data are fed to back-propagate the obtained loss functions.

achieved exciting results on clustering by combining advantages of GANs and clustering method, utilizing clustering methods to improve the quality of generating images of GANs still deserves more attentions. This brings the main motivation of our work: *can we re-design the framework of GANs in an un-conditional manner and utilize the capability of K-Means on representation learning to replace the role of real labels?*

In order to make use of clustering labels of K-Means to direct the generating process as real labels in GANs, we consider operating K-Means on the top layer of the discriminator. But the main difficulty is how to deal with the un-differentiable objective of K-Means using Stochastic Gradient Decent (SGD) [20]. Deep Embedded Clustering (DEC) [8] straightforwardly separates the optimization into updating centers and network parameters successively. Another CNN-based method [21] also adopts this technique and further proposes a feature drift compensation scheme to mitigate the drift error caused by different optimization directions of K-Means and regular loss functions. Then Deep Clustering Network (DCN) [9] introduces a defined “pretext” objective, a mathematical combination of reconstruction loss and K-Means clustering objective, and optimizes K-Means with back-propagation. Quite recently, Deep K-Means [22] proposes a continuous reparametrization of the objective of K-Means clustering to optimizes it with SGD.

Motivated by these works, we propose an un-conditional generative adversarial model, named K-Means-GAN (KM-GAN), which embeds the idea of updating centroids of K-Means into the framework of GANs. As the framework illustrated in Fig. 1, our model conducts the discriminator as a nonlinear feature extractor and utilizes K-Means clustering algorithm for getting more representative features. Further, we employ obtained results of K-Means instead of one-hot real labels to direct the generator in the generating process. Then we propose objectives containing clustering labels from the perspective of deep metric learning (DML) to let the optimization direction of K-Means agree with the generating process. The specific optimization process includes three terms to alternately optimize, of which the “center-loss” term tries to pull the corresponding centers of real and generated images closer. Furthermore, the objective of the discriminator is proposed to minimize the distance between real samples and their corresponding centers and maximize the distance between fake samples and their corresponding real centers. Meanwhile, the loss function of the generator, which is interpreted as an adversarial term, attempts to approximate the target distribution by decreasing the distance between generated samples and their corresponding real centers.

Contribution. To the best of our knowledge, our work is the first attempt to combine the training of unsupervised K-Means algorithm with GAN simultaneously through SGD for generating tasks. Our main contributions are summarized as follows:

- We propose an un-conditional implementation of GANs, called K-Means-GAN (KM-GAN), and equip it with new objective functions from the perspective of DML.
- We incorporate GANs with the idea of traditional K-Means and utilize obtained labels, replacing the role of real labels, to direct the generating process and get more representative features.
- We empirically show that KM-GAN is capable of generating diverse samples and the quality of generated images on several real-world datasets is competitive with that of conditional GANs.

2. Background

In this section, we introduce notations and briefly review preliminary knowledge, including the framework of GANs and K-Means. The notations provided in this section will also be used in subsequent sections.

2.1. Notations

Throughout the paper, we use b for the batch size, D for the discriminator, G for the generator and k for the pre-defined number of classes.

2.2. Framework of GANs

GAN [1] consists of two components: a discriminator D and a generator G which are both realized by the neural networks. The main idea is actually an adversarial training procedure between them. Throughout the adversarial training, the generator G maps samples from a prior noise distribution, such as gaussian distribution, to the data space, while the discriminator D estimates the probability that its inputs come from real data distribution rather than the generated distribution.

More specifically, given a noise distribution $P_{\mathbf{z}}$ and training samples $\mathbf{x} \sim P_{\mathbf{x}}$, the adversarial training contains two steps. Firstly, we fix parameters of the generator, generate $G(\mathbf{z})$ from samples $\mathbf{z}$ of the noise distribution $P_{\mathbf{z}}$, and update parameters of the discriminator by optimizing the objective of D as follows:

$$\min_{D} \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \tag{1}$$

Then we fix parameters of D and update parameters of G to approximate target distribution by optimizing the loss function of G as follows:

$$\max_{G} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \tag{2}$$

In order to generate more realistic images, CGANs [2] implements GANs with one-hot real labels, which provides supplementary information of class distribution for the generating process.

This method qualitatively and quantitatively improves the performance of GANs in generating tasks. Recent works [23,24] then extend VAE and GMMN based on this technique for more realistic images. Furthermore, Deep Convolutional GAN (DCGAN) [4] designs a stable architecture utilizing convolutional neural networks and raises several tricks to stabilize the adversarial training process. On the other hand, lots of works [3,5,25–28] propose objectives for GANs to improve stability and image quality.

2.3. K-Means

K-Means [29] is a traditional clustering method used to group a set of given data points $\{\mathbf{x}_i\}_{i=1,2,\ldots,N} \in \mathbb{R}^m$ into k clusters, where k is a pre-defined number. After randomly choosing k points of data samples as initialized center, the main algorithm is composed of two steps. The first is to assign clustering labels to each point according to the Euclidean distance between the point and the current k centers. Then we compute new centers as the weighted average of points in each class. The algorithm stops when each center does not change. Formally, the cost function is as follows:

$$\min_{\mathbf{M}\in\mathbb{R}^{m*k},\mathbf{s}_i\in\mathbb{R}^k} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{M}\mathbf{s}_i\|_2^2$$
$$s.t. \quad s_{ij} \in \{0,1\}, \mathbf{1}^T\mathbf{s}_i = 1, \forall i, j, \tag{3}$$

where $\mathbf{s}_i$ is the one-hot clustering label of data point $\mathbf{x}_i$, s_{ij} denotes the jth element of vector $\mathbf{s}_i$, and $\mathbf{M}$ is a matrix whose k columns correspond to the k centers.

Although K-Means is widely employed as a part of other models [8,9,21], the performance of K-Means depends heavily on the initialized centers. To remedy this issue, K-Means++ [30] proposes a better procedure to initialize centers. Later extensions [16,21,22] adopt the procedure and achieve improvements on other applications, such as deep clustering and representation learning. Moreover, Minibatch K-Means[31] instead updates centers using a batch of samples in each iteration to generalize original K-Means for dealing with large-scale datasets and online scenarios.

3. Proposed method

As mentioned before, we consider re-designing the framework of GANs and utilize results of K-means to replace the role of one-hot real labels in an un-conditional manner. So we treat the discriminator as a feature extractor rather than a classifier and operate K-Means on extracted features to produce clustering labels which are viewed as the substitution of real labels. With obtained features and labels, we propose our objectives from the perspective of DML to carry out adversarial learning. More importantly, we come up with a "center-loss" term to connect the optimization of adversarial learning and centers updating in K-Means. In the following subsections, we first introduce our proposed objectives and the optimization procedure of regular KM-GAN. Then we generalize it with regularization terms in order to deal with more general datasets.

3.1. Regular KM-GAN

We first introduce the "center-loss" term since it fills the gap of two different optimization directions between adversarial learning and K-Means, and it is fairly important for the whole algorithm to work effectively. The term is interpreted as a role to decrease the distance between corresponding centers of real and generated images. Formally, the formula is as follows:

$$\min_{D,G} \quad L_{center} = \left\| \sum_{m=1}^{k} \frac{\mathbf{c}_m + \sum_{j=1}^{j_{c_m}} D(\mathbf{x}_{n_{j,c_m}})}{1 + j_{c_m}} - \sum_{m=1}^{k} \frac{\widehat{\mathbf{c}}_m + \sum_{j=1}^{j_{\widehat{c}_m}} D(G(\mathbf{z}_{n_{j,\widehat{c}_m}}))}{1 + j_{\widehat{c}_m}} \right\|_1$$
$$s.t. \quad L_{center} \geq d_{round}, \tag{4}$$

where k is the pre-defined number of classes, $\mathbf{c}_m$ ($\widehat{\mathbf{c}}_m$) is the mth center of features of real data (generated data) updated after last iteration, j_{c_m} ($j_{\widehat{c}_m}$) is the number of features belonging to the center $\mathbf{c}_m$ ($\widehat{\mathbf{c}}_m$), n_{j,c_m} ($n_{j,\widehat{c}_m}$) denotes the position of corresponding feature of real data (generated data) that is in class m according to results of K-Means in the first forward pass and d_{round} is a hyper-parameter needed to tune according to different datasets to avoid degeneration.

Indeed, L_{center} calculates the difference of a second order statistical magnitude, i.e., the average of k centers, between features of real and generated images. The intension is to keep centers of synthesized data not far away from that of real data and accelerate distribution approximation. The exploration of minimizing statistical magnitudes is motivated by the success of recent works [5,15,32] on classification and generation tasks. Especially, GMMN successfully approximates data distribution through minimizing all orders of statistics, which is realized by the Gaussian kernel. So we intuitively utilize the second order statistics and reuse the results of K-Means to propose the continuous term. Experimental results further show that KM-GAN fails to generate meaningful images even on MNIST without "center-loss" term.

Although the "center-loss" term is proposed to approximate the target distribution, we still need objective functions for the discriminator and the generator to finish the regular adversarial training. Firstly, we define the objective function of discriminator as follows:

$$\min_{D} \quad L_D = \|D(\mathbf{x}) - C_{real}\|_2 - \|D(G(\mathbf{z})) - C_{gen}\|_2, \tag{5}$$

where C_{real} (C_{gen}), computed based on real centers, consists of b center pieces for the pre-defined batch size b. Each of these center pieces is the centroid of the real class where the feature piece in the corresponding position of this batch belongs. It is natural to see that L_D penalizes the distance between each class of real data and their corresponding k centers. The interpretation is to minimize intra-class distance of each class in the feature space of real data from the viewpoint of DML. On the contrary, L_D maximizes the distance between generated data and centers of their corresponding real classes to discriminate the counterfeit from real data.

On the other hand, the corresponding objective function of the generator is defined as follows:

$$\min_{G} \quad L_G = \|D(G(\mathbf{z})) - C_{gen}\|_2. \tag{6}$$

Obviously, the effect of the objective is to compete with the discriminator to approximate the target distribution. When decreasing the distance between synthesized data and centers of their corresponding k real classes, the features of generated images are distributed around each real center like features of real data. Then with the impact of "center-loss" to pull centers of real and fake data close, fake data distribution would approximate the target distribution finally. The term also plays a role as an adversarial term in the framework of KM-GAN.

3.2. Three-Step alternating optimization

It is straightforward to optimize network parameters of GANs and update centers step by step as in DEC [8]. But the different directions of these two steps make the optimization more difficult. To deal with this issue, we utilize "center-loss" term to bridge the gap. Especially, the "center-loss" term reuses results of K-Means and obtained features from the discriminator, which builds a connection between these two steps. In the specific optimization, we first solve the subproblem of adversarial learning, i.e., updating parameters of the discriminator and generator, respectively. Then inspired by alternating optimization in [9], we utilize "center-loss" to re-update parameters of D and G via SGD. With current parameters, we obtain centers in feature space at last by computing Eq. (3). The concretely three-step alternating optimization procedure is shown in Algorithm 1.

Algorithm 1 Training algorithm for regular KM-GAN.

```
Input: Real images $\mathbf{X}$, noise distribution $P_{\mathbf{Z}}$, pre-defined number of classes $k$, number of iterations $T$, batch size $b$=64, learning rate $r = 0.0002$ and the hyper-parameter of Adam $\beta_1 = 0.5$
Output: Generated samples $G(\mathbf{z})$
  Initialize parameters of $D$ and $G$ networks
  Initialize $k$ real centers of mapped features $D(\mathbf{X})$ by K-Means++
  for $t = 1 : T$ do
    Sample a batch $\{\mathbf{x}_i\}_{i=1}^b$ from real data $\mathbf{X}$
    Sample a batch $\{\mathbf{z}_i\}_{i=1}^b$ from noise distribution $P_{\mathbf{Z}}$
    Obtain features $\{D(\mathbf{x}_i)\}_{i=1}^b$ and $\{D(G(\mathbf{z}_i))\}_{i=1}^b$
    Obtain clustering labels according to Euclidean distance with current $k$ centers
    $grad_{\theta_d} = \nabla_{\theta_d} L_D$
    $\theta_d$ =Adam($grad_{\theta_d}, \theta_d, \alpha, \beta_1, \beta_2$)
    $grad_{\theta_g} = \nabla_{\theta_g} L_G$
    $\theta_g$ =Adam($grad_{\theta_g}, \theta_g, \alpha, \beta_1, \beta_2$)
    $grad_{\theta_d,\theta_g} = \nabla_{\theta_d,\theta_g} L_{center}$
    $\theta_d, \theta_g$ =Adam($grad_{\theta_d,\theta_g}, \theta_d, \theta_g, \alpha, \beta_1, \beta_2$)
    Update centers via K-Means objective in Eq. (3)
  end for
```

In the described algorithm, we conduct K-Means++ technique to better initialize centers of features. In addition, since the optimization of network parameters employs Adam [20] and depends on the pre-defined batch size, it is natural to come up with Minibatch K-Means. With this procedure, the "center-loss" further plays a role to mitigate the error caused by different optimization directions of K-Means and regular loss functions in each iteration, which is similar to Hsu and Lin [21].

3.3. Generalized KM-GAN

Although commonly used datasets have obvious criterions to cluster, such as MNIST [33] and CIFAR-10 [34], there exist datasets that do not have these obvious criterions. For example, CelebA [35] and LFW [36] contains too many personalities and images for each personality are not enough for generation tasks. It is even harder to find a suitable number for the pre-defined k. In this case, operating K-Means to cluster features is too difficult. To handle with such problem, we generalize regular KM-GAN with two regularization terms. They act as constraints [5] on the whole class of real and fake images. Before explaining the constraints, we define two necessary terms L_{intra} and L_{inter} used to generalize KM-GAN as follows:

$$L_{intra} = \sum_{\mathbf{x}_i,\mathbf{x}_j \in B_d} \|D(\mathbf{x}_i) - D(\mathbf{x}_j)\|_1 + \sum_{G(\mathbf{z}_i),G(\mathbf{z}_j)\in B_g} \|D(G(\mathbf{z}_i)) - D(G(\mathbf{z}_j))\|_1,$$

$$L_{inter} = \sum_{\mathbf{x}_i \in B_d, G(\mathbf{z}_j)\in B_g} \|D(\mathbf{x}_i) - D(G(\mathbf{z}_j))\|_1,$$

where B_d and B_g denote the corresponding batch of real samples $\{\mathbf{x}_i\}_{i=1}^b$ and generated samples $\{G(\mathbf{z}_i)\}_{i=1}^b$, respectively.

Table 1
Details of synthetic data and real-world datasets.

Dataset	Numbers of Images	Feature dimensions	Classes
Synthesis	10,000	100	4
MNIST	70,000	28×28	10
Fashion-10	70,000	28×28	10
CIFAR-10	70,000	$32 \times 32 \times 3$	10
CelebA	202,599	$64 \times 64 \times 3$	No

Then the objective functions of the discriminator and the generator become:

$$L_D = \min_{\theta_D} \|D(X) - C_{real}\|_2 - \|D(G(z)) - C_{gen}\|_2 + \lambda * (L_{intra} - L_{inter}),$$
$$L_G = \min_{\theta_G} \|D(G(z)) - C_{gen}\|_2 + \lambda * L_{inter}. \quad (7)$$

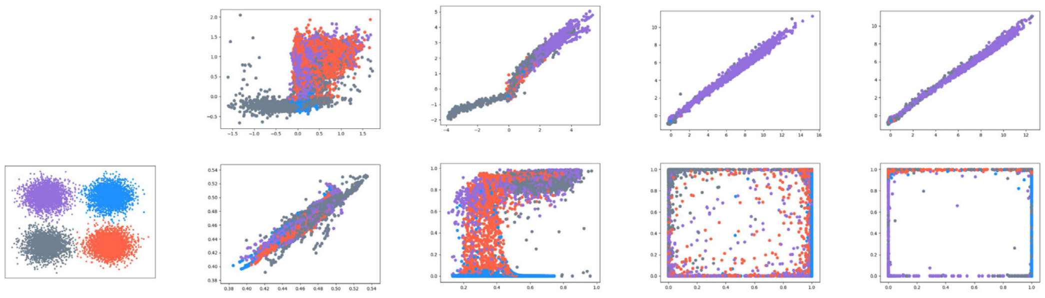
In the case described above, the objective functions of regular KM-GAN are not effective enough since they depends heavily on k. However, these two terms, one decreasing intra-class distances of the whole real and fake data in feature space while the other minimizing inter-class distance to approximate data distribution, help to approximate the data distribution as a whole class. With above regularization terms, experimental results also show that the final centers reduce to the same one whatever the pre-defined k is (such as $k = 10$ or $k = 20$), which coincides with the goal of these regularization terms. This implies that KM-GAN could be applied to more general scenarios with them. We use the hyperparameter λ in experiments to balance the regular loss functions and these two regularization terms.

4. Experiments

Datasets: In this section, we first conduct experiments on a synthetic dataset to show the capability of the discriminator of KM-GAN to represent features. Then we qualitatively and quantitatively show that KM-GAN is able to generate realistic and diverse images on real-world datasets including MNIST, Fashion-10, CIFAR-10 and CelebA. Details about these datasets are shown in Table 1. Note that the hyperparameter λ is only used on CelebA dataset.

Model Architecture and Notations: Experimental results [28] show that deeper and more complex architectures, such as architectures of ResNet [37], could improve the performance of GANs. To keep a fair comparison, we use architectures similar to DCGAN [1] for all tested models in the following experiments. This means we do not consider models based on ResNet or other complex architectures. The detailed architectures of the discriminator and the generator are shown in later experiments. We now explain the symbols which will be used to show architectures. We use FC, Conv, Upconv, ReLU, LReLU and BN to represent fully connected layer, convolutional layer, de-convolutional layer, ReLU activation layer, Leaky-ReLU activation layer and batch normalization layer. Specifically, "FC $4 \times 4 \times 512$ BN LReLU" implies that the inputs are fully connected to $4 \times 4 \times 512$ outputs and then are connected with BN and LReLU in sequence. Similarly, "5×5 Conv 128 stride 2 BN LReLU" shows that the inputs are transformed first by a 2-stride convolutional layer with 5×5 kernels and 128 output channels, and then connected with BN and LReLU in sequence.

Evaluations: The evaluation of generative models is also an important issue that helps us compare the performance of models in different aspects, including image quality, diversity and even overfitting problems [38,39]. Although different architectures and different datasets make it hard to measure these aspects with a certain metric, researchers have also proposed several methods to bridge the gap, such as Inception Score [40], FID [41], GILBO [42], and the criterion in [43]. To qualitatively and quantitatively show



(a) Real data (b) Epoch 0 (c) Epoch 50 (d) Epoch 150 (e) Epoch 200

Fig. 2. Subfigure (a) is the visualization of the intrinsic 2-dimensional structure of synthetic data. Subfigures (b)–(e) are visualizations of features on the top layer of corresponding discriminators of DCGAN and KM-GAN in the training process on synthetic dataset. The first line belongs to DCGAN while the second line is of KM-GAN. Besides, the four kinds of colored points represent different categories. Obviously, the features of KM-GAN could separate most of them while that of DCGAN is ineffective.

Table 2
Architectures of $\mathcal{M}$, generator and discriminator.

Mapping function $\mathcal{M}$	Generator	Discriminator
Input $\{\mathbf{h}_i\}_{i=1}^n \in \mathbb{R}^2$	Input $\{\mathbf{z}_i\}_{i=1}^n \in \mathbb{R}^{100}$	Input $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{100}$
FC 10 Sigmoid	FC 10 BN ReLU	FC 100 BN ReLU
FC 100 Sigmoid	FC 50 BN ReLU	FC 50 BN ReLU
$\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{100}$	FC 100 BN ReLU	FC 10 BN ReLU
	$\{G(\mathbf{z}_i)\}_{i=1}^n \in \mathbb{R}^{100}$	FC 2 Sigmoid
		$\{D(\mathbf{x}_i), D(G(\mathbf{z}_i))\}_{i=1}^n \in \mathbb{R}^2$

the performance of KM-GAN, we choose different ways to measure our model on above real-world datasets. For MNIST and Fashion-10, we both compare generated images of KM-GAN with samples generated by DCGAN. Besides, we classify generated images of KM-GAN and DCGAN via a well-trained classifier (realized by a multi-layer perceptron) on Fashion-10 to show the diversity of generated samples. Furthermore, we apply Inception Score and FID on CIFAR-10 and CelebA, respectively. Inception Score is the first proposed to evaluate generated images on CIFAR-10 and ImageNet. Then researchers generalize this metric to FID for evaluations on more datasets. We apply different metrics to test generated images of KM-GAN from different perspectives and avoid overfitting problems [38].

Except for the choice of metrics, we also need to consider hyperparameters in KM-GAN. In the following experiments, we first experiment with different values of some hyperparameter while keeping others fixed, and run KM-GAN for a certain number of epochs. Then we choose the best values for these hyperparameters via validating techniques. With these carefully chosen hyperparameters, we rerun KM-GAN several times and compare the final results with other conditional and un-conditional models.

4.1. *KM-GAN on synthetic data*

The synthetic dataset consists of 10,000 points that belong to $\mathbb{R}^{100}$ and has K-Means-friendly [9] structure in a two-dimensional domain which we could not observe from the original data. In fact, we first choose four two-dimensional Gaussian distributions with different means and covariance matrices as in Fig. 2(a). Then we sample 2500 points from each distribution and map them into $\mathbb{R}^{100}$ via a mapping function $\mathcal{M}$, which is realized by a non-linear neural network shown in Table 2. On this toy dataset, we simply set d_{round} to be 0. The network structures of DCGAN [1] and KM-GAN are the same and shown in Table 2. The obtained features of KM-GAN and DCGAN on the synthetic dataset in the training process are shown in Fig. 2.

As we can see from Eqs. (5) and (6) of KM-GAN, extracted features of the discriminator play an important role not only on the objective function of the discriminator itself, but also on that of the generator. To demonstrate that extracted features of KM-GAN are representative enough to do representation learning, we compare with those of DCGAN in different epochs. From the visualization of the features produced by discriminators of these two models in the training process as shown in Fig. 2, features of different classes of DCGAN degenerate to similar points. Obviously, those features of our proposed KM-GAN are more representative to show the intrinsic structure although they are both capable of generating high-quality images on real-world datasets.

4.2. *KM-GAN on MNIST*

MNIST [33] dataset has 70,000 gray images of handwritten digits of size 28×28. We first conduct experiments to compare KM-GAN with its reduced version which operates K-Means in pixel space as introduced in Algorithm. Then we improve KM-GAN with weight-clipping which stabilizes the training process. The network structures of KM-GAN for training MNIST are the same as that of DCGAN, the hyperparameter d_{round} is set to be 10,000, and all tested models on MNIST is trained for 24 epochs.

4.2.1. *Feature space vs. original space*

To demonstrate the effect of carrying out K-Means in feature space rather than pixel space, we compare KM-GAN with reduced KM-GAN, in which we operate K-Means to cluster data in pixel space. In fact, computations of K-Means appear to increase quickly as the dimensionality of data increases when experimenting with reduced KM-GAN. However, the capability of dimensionality reduction of KM-GAN avoids such computational difficulties. In the following, we further qualitatively show the advantage of operating K-Means in latent space as exhibited images in Fig. 3.

From images in Fig. 3(b) and (c), it is obvious that the quality of generated digits is significantly better when clustering is operated in the feature space. Regular KM-GAN successfully generates realistic handwritten digits in both different classes and different angles while reduced KM-GAN even suffers mode collapse, i.e., most of these generated images are similar or identical. Besides, synthesized images seem to be different from original classes since we feed no real labels to direct generating process, which gives a chance to generate more diverse images.

4.2.2. *Improvement on KM-GAN*

Although KM-GAN is proven to be capable of generating realistic and diverse images, it still fails to generate images sometimes. So we utilize a common technique called weight clipping to constrain parameters of the discriminator (feature extractor) in a smaller bounding box. Specifically, we clamp the weights of D to a fixed box so that it could only output values in a certain range.

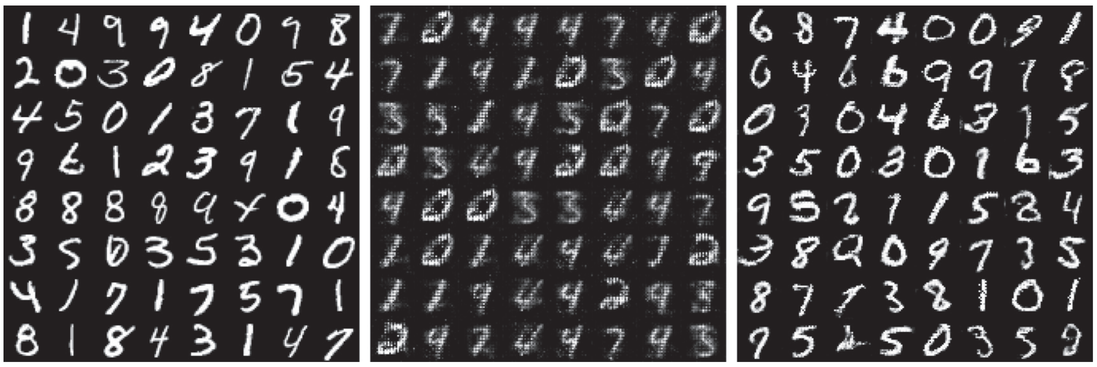


Fig. 3. Comparison of generated samples of KM-GAN and reduced version of KM-GAN on MNIST dataset.

Algorithm 2 Training algorithm for reduced KM-GAN.

```
Input: Real Images $\mathbf{X}$, noise distribution$P_{\mathbf{Z}}$,pre-defined number of classes $k$, number of iterations $T$, batch size $b = 64$, learning rate $r = 0.0002$and the hyper-parameter of Adam $\beta_1 = 0.5$
Output: Generated samples $G(\mathbf{z})$
  Initialize $k$ real centers $\widetilde{C}$ of data $\mathbf{X}$ by K-Means++
  Initialize parameters of $D$ and $G$ networks
  for $t = 1 : T$ do
      Sample a batch $\{\mathbf{x}_i\}_{i=1}^{b}$ from real data $\mathbf{X}$
      Sample a batch $\{\mathbf{z}_i\}_{i=1}^{b}$ from noise distribution $P_{\mathbf{Z}}$
      Obtain features $\{D(\mathbf{x}_i)\}_{i=1}^{b}$ and $\{D(G(\mathbf{z}_i))\}_{i=1}^{b}$
      Obtain clustering labels according to Euclidean distance with current $k$ centers
      $\widetilde{L}_{center} = \|D(\widetilde{C}_{real}) - D(\widetilde{C}_{gen})\|_1$
      $grad_{\theta_d,\theta_g} = \triangledown_{\theta_d,\theta_g}\widetilde{L}_{center}$
      $\theta_d, \theta_g =$Adam($grad_{\theta_d,\theta_g}, \theta_d, \theta_g, \alpha, \beta_1, \beta_2$)
      $\widetilde{L}_D = \|D(\mathbf{x}) - D(\widetilde{C}_{real})\|_2$
      $grad_{\theta_d} = \triangledown_{\theta_d}\widetilde{L}_D$
      $\theta_d =$Adam($grad_{\theta_d}, \theta_d, \alpha, \beta_1, \beta_2$)
      $\widetilde{L}_G = \|D(G(\mathbf{z})) - D(\widetilde{C}_{gen})\|_2$
      $grad_{\theta_g} = \triangledown_{\theta_g}\widetilde{L}_G$
      $\theta_g =$Adam($grad_{\theta_g}, \theta_g, \alpha, \beta_1, \beta_2$)
      Update centers in original pixel space via K-Means objective in Eq. (3)
  end for
```

The technique further guarantees the property that points close in pixel space are not far away from each other after being mapped into feature space.

As synthesized images shown in Fig. 4(a) and (b), the performance of KM-GAN without weight clipping is already competitive with DCGAN on MNIST dataset. This demonstrates that the utilization of clustering labels successfully replaces the role of real labels to direct generating process and encourages us to pay more attention to un-conditional generative models. What is more, to stabilize the three-step alternating optimization process, we equip KM-GAN with weight clipping and the bounding box is set to be $[-1, 1]$. The synthesized images shown in Fig. 4(c) are competitive or even better than KM-GAN without weight clipping. This technique will be applied in later experiments, and we will introduce how to choose the best bounding box for weight clipping and d_{round} on CIFAR-10 dataset.

4.3. KM-GAN on Fashion-10

Fashion-10 dataset, consisting of various types of more complicated fashion products rather than handwritten digits, has the same number of images as MNIST and the size of each image is also 28×28. So we train with the same architectures as used on MNIST to examine KM-GAN on Fashion-10. The number of epochs and hyperparameter d_{round} are also the same as on MNIST. From the experimental results shown in Fig. 5(a) and (b), the quality of synthesized images of KM-GAN is comparable to that of DCGAN.

To further quantitatively show that our proposed method is also capable of generating diverse images without the help of one-hot real labels, we train a three-layer convolutional classifier on Fashion-10 separately (97% accuracy on training set and 91% on test set) and use the network to classify 5000 synthesized images of KM-GAN and DCGAN. The result of the frequency of each class is shown in Fig. 5(c). Since Fashion-10 equally contains images of each class, conditional models easily generate images equally for each class with the help of real labels. So we compare with results of DCGAN to further show that KM-GAN is also capable of achieving this. Specifically, in the frequency chart of generated images, numbers 0–9 denote 10 classes of the dataset and two colors, "blue" and "gray", represent results of KM-GAN and DCGAN, respectively. From the resulted class distributions, most classes are generated with probability close to 10% by KM-GAN except the class "shirts", which is under-represented with 7.0%. We infer that this is because the class "shirts" is very similar to "T-shirts" and "pullovers".

4.4. KM-GAN on CIFAR-10

CIFAR-10 [34] is a dataset with 60,000 RGB images of size 32×32 in 10 classes. There are 6000 images in each class with 5000 for training and 1000 for testing. All these images are used here to train KM-GAN. The network structures are shown in Table 3 and we set d_{round} to be 20,000 and bounding box to be $[-0.1, 0.1]$, which we will explain how to choose in later experiments.

We first qualitatively evaluate the generated images of KM-GAN on CIFAR-10 dataset and show the experimental results in Fig. 6. To demonstrate the capability of our proposed objective functions, we compare with MBGAN which also proposes substituted objective functions from the perspective of DML. Results show that synthesized images of KM-GAN are obviously more realistic and meaningful. We further compare with DCGAN and there is no visual difference between the quality of synthesized images of these two models, which also demonstrates the effectiveness of KM-GAN.

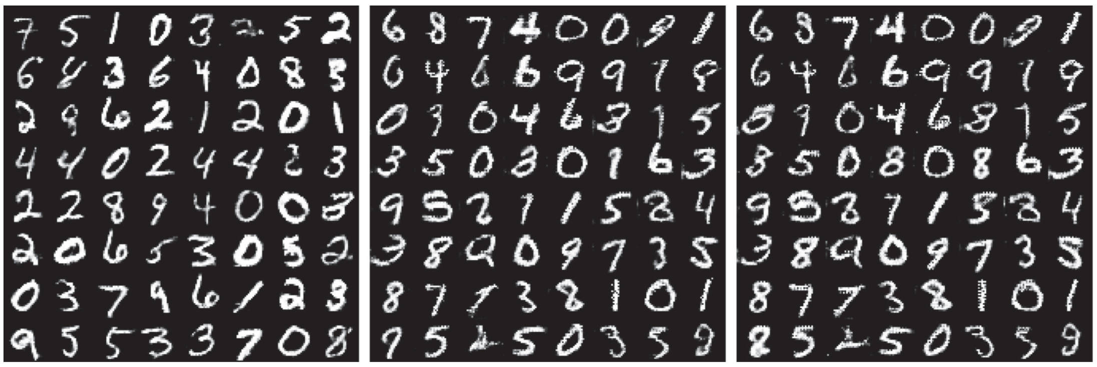
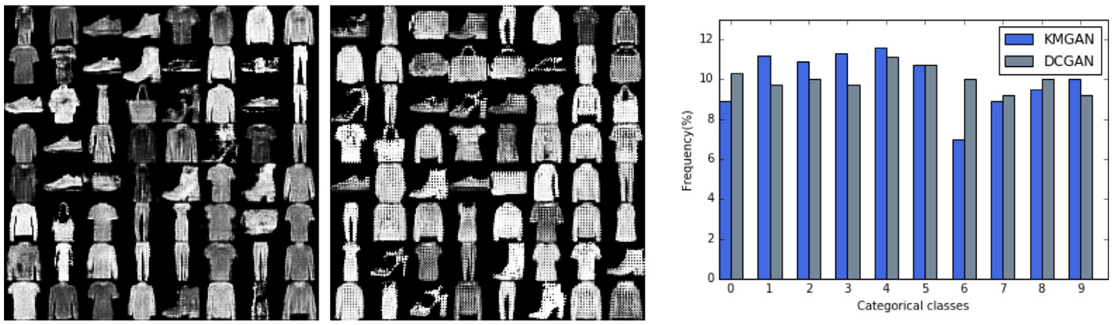

(a) DCGAN (b) KM-GAN (c) KM-GAN with weight clipping

Fig. 4. Subfigures (a) and (b) compare generated images of DCGAN and KM-GAN on MNIST dataset, and subfigure (c) shows generated images of KM-GAN improved with weight clipping.



(a) DCGAN (b) KM-GAN (c) Frequency chart

Fig. 5. Evaluation of synthesized images of KM-GAN on Fashion-10 dataset. Subfigures (a) and (b) exhibit a random batch of generated images of DCGAN and KM-GAN, and subfigure (c) shows the distributions of generated images of DCGAN and KM-GAN with gray and blue bars, respectively.

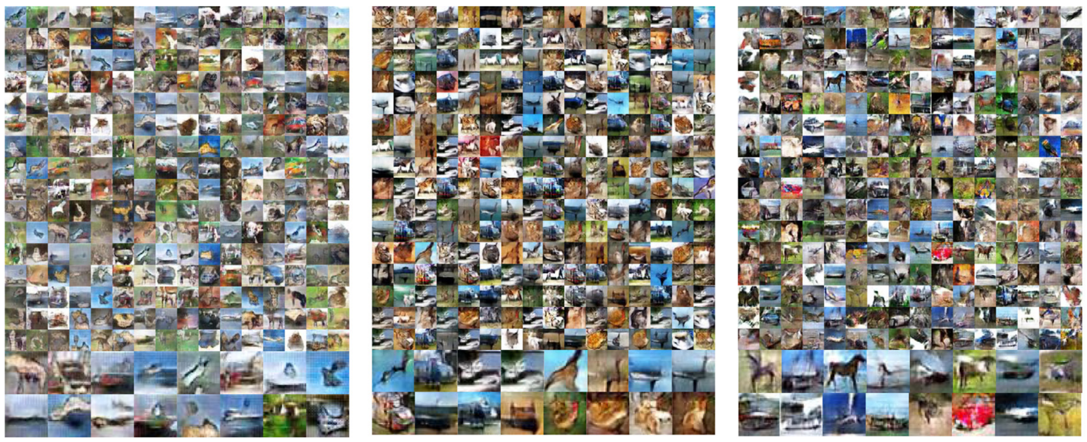

(a) MBGAN (b) DCGAN (c) KM-GAN

Fig. 6. Comparison of generated samples of MBGAN, DCGAN and KM-GAN on CIFAR-10. Note that the last two lines in the bottom of each subfigure are chosen carefully from above generated samples for an intuitive comparison.

Since we use clustering labels of K-Means to replace one-hot real labels in KM-GAN, i.e., a purely un-supervised training, we quantitatively evaluate the diversity of images synthesized by our model via another index called Inception Score [40] on CIFAR-10 dataset. The index applies Inception model [44] to every generated image and computes the final score with the following formula:

$$\mathbf{IS}(G(\mathbf{z})) = \exp(\mathbb{E}_{\mathbf{z}}\mathbf{KL}(p(y|G(\mathbf{z}))\|p(y))). \quad (8)$$

Indeed, the main idea of Eq. (8) is that diverse images which contain meaningful objects are supposed to have a conditional label distribution $p(y|G(\mathbf{z}))$ with low entropy and a marginal distribution $\int p(y|G(\mathbf{z}))d\mathbf{z}$ with high entropy.

As we explain in the beginning of this section, GANs are sensitive to hyperparameters, so we carefully choose the best bounding box for weight clipping and d_{round} from a set of candidates before the quantitatively evaluation. Specifically, we first run KM-GAN with each bounding box for 24 epochs three times and keep

Table 3
Architectures of generator and discriminator on CIFAR-10.

Generator	Discriminator
Input $\{\mathbf{z}_i\}_{i=1}^n \in \mathbb{R}^{100}$	Input $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{64\times64\times3}$
FC $4\times4\times512$ BN LReLU	5×5 Conv
	64 stride 2 LReLU
5×5 Upconv	5×5 Conv
256 stride 2 BN LReLU	128 stride 2 BN LReLU
5×5 Upconv	5×5 Conv
128 stride 2 BN LReLU	256 stride 2 BN LReLU
5×5 Upconv	5×5 Conv
64 stride 2 BN LReLU	512 stride 2 BN LReLU
5×5 Upconv	FC 4096 BN LReLU
3 Stride 2 Sigmoid	
	FC 100 Sigmoid

Table 4
Inception Scores of different bounding boxes on CIFAR-10.

bounding boxes	None	[−1, 1]	[−0.1, 0.1]	[−0.01, 0.01]
Inception Score	Failed	3.15 ± 0.28	5.56 ± 0.12	4.02 ± 0.61

Table 5
Inception Scores of different values for d_{round} on CIFAR-10.

d_{round}	None	5000	10,000	20,000	30,000
Inception Score	Failed	4.04 ± 0.32	4.46 ± 0.28	5.56 ± 0.12	Failed

Table 6
Inception Score on CIFAR-10 dataset.

	Model	Inception score
Conditional Models	MBGAN	4.27 ± 0.07
	MLGAN-clipping [5]	5.23 ± 0.29
	DCGAN	5.92 ± 0.17
	WGAN [15] (with labels)	5.88 ± 0.07
	WGAN-GP	6.46 ± 0.03
Un-conditional Models	MIX + WGAN [45]	4.04 ± 0.07
	Improved GANs [40]	4.36 ± 0.04
	ALI [46] (from [47])	4.79
	Wasserstein GANs [25] (from [45])	3.82 ± 0.06
	KM − GAN	$\mathbf{5.61\pm0.09}$

other hyperparameters fixed (d_{round} is fixed to be 20,000). Then we apply Inception Score to generated images and choose the candidate with the best score (i.e., the highest score). The experimental results are shown in Table 4.

It is natural to see that KM-GAN performs the best when the bounding box is set to be $[-0.1, 0.1]$. After choosing the best bounding box for weight clipping, we fix it and seek out the best d_{round} is at 20,000 with the same procedure. The results are shown in Table 5.

With the chosen bounding box and d_{round}, we rerun the model for 24 epochs five times and compare the mean inception score with both conditional and un-conditional models to characterize the performance of KM-GAN as in Table 6. Specifically, WGAN, Improved GANs, and MIX+WGAN are trained without feeding real labels, while ALI itself is an un-conditional model utilizing an auto-encoder to assist the generator to approximate target distribution. Obviously, KM-GAN out-performs these models with a large margin, which demonstrates the effectiveness of KM-GAN. Then we compare with two conditional methods based on DML, MLGAN and MBGAN. KM-GAN also works better than them. Furthermore, we compare with DCGAN and WGAN, which are very stable and commonly used in the research field of GANs. Results show that KM-GAN is a little lower but competitive with them, which coincides with the qualitative evaluation in Fig. 6. We infer that this is because synthesized images of KM-GAN shown in Fig. 6(c) are more meaningful while the backgrounds of generated images of DCGAN shown in Fig. 6(b) are clearer. However, the result is lower than the state-of-the-art result achieved by WGAN-GP, which improves WGAN with gradient penalty. This issue is indeed out of the scope of our exploration in this paper. But the issue is also compatible with KM-GAN, which means we could explore to constrain the gradients of our model with such a penalty in our future works.

Table 7
Architectures of generator and discriminator on CelebA.

Generator	Discriminator
Input $\{\mathbf{z}_i\}_{i=1}^n \in \mathbb{R}^{100}$	Input $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{64\times64\times3}$
FC $4\times4\times512$ BN ReLU	5×5 Conv
	64 stride 2 BN LReLU
5×5 Upconv	5×5 Conv
256 stride 2 BN ReLU	128 stride 2 BN LReLU
5×5 Upconv	5×5 Conv
128 stride 2 BN ReLU	256 stride 2 BN LReLU
5×5 Upconv	5×5 Conv
128 stride 2 BN ReLU	512 stride 2 BN LReLU
5×5 Upconv	FC 100 Sigmoid
3 Stride 2 Tanh	

Table 8
FID with different λ on CelebA.

λ	0	1	5	10	20
FID	Failed	32.90 ± 1.85	34.25 ± 0.89	31.71 ± 0.37	$42.93\pm13, 57$

4.5. KM-GAN on CelebA

CelebA [35], as a large-scale face dataset, contains more than 200,000 RGB face images from 10,177 celebrity identities, and there are 40 binary attributes and 5 landmarks for each image. We crop these images into 64×64 for the following experiments.

4.5.1. Qualitative and quantitative evaluation

We first conduct experiments to qualitatively show the performance of KM-GAN. We set hyperparameters d_{round} to be 20,000 and the bounding box for weight clipping to be $[-0.01, 0.01]$ with the same procedure as on CIFAR-10 dataset. The network structures are shown in Table 7. Besides, we set λ to be 10 which is chosen carefully by later quantitative evaluations. Generated samples of DCGAN and KM-GAN are exhibited in Fig. 7.

From samples shown in Fig. 7, KM-GAN also performs well on CelebA dataset. To further quantitatively evaluate the quality of generated images of KM-GAN, we compare it with several models via FID [41] metric, which is a generalized version of Inception Score. FID metric assumes that each data distribution follows a multidimensional Gaussian distribution and measures the distance between any two such distributions via Wasserstein-2 metric. With such an assumption, FID could be applied to more datasets, such as CelebA and MNIST. The formula used for FID metric is as follows:

$$\mathbf{FID}((\mathbf{m},\mathbf{C}),(\tilde{\mathbf{m}},\tilde{\mathbf{C}})) = \|\mathbf{m}-\tilde{\mathbf{m}}\|_2^2 + \mathrm{Tr}(\mathbf{C}+\tilde{\mathbf{C}}-2(\mathbf{C}\tilde{\mathbf{C}})^{1/2}), \tag{9}$$

where $(\mathbf{m}, \mathbf{C})$ and $(\tilde{\mathbf{m}},\tilde{\mathbf{C}})$ denote two distributions which follow multidimensional Gaussians. Note that lower FID is better.

To give a better result, we also search the best λ for KM-GAN with other hyperparameters fixed as on CIFAR-10. We first run KM-GAN with each candidate for 24 epochs three times. Then we compute the mean FID between $10k$ samples generated by the model and $10k$ samples chosen randomly from the training set, and choose the best λ. The results are shown in Table 8.

When λ is set to be 10, KM-GAN is obviously more stable and performs the best. So we set λ to be 10, rerun the model for 24 epochs five times, and compute the mean FID as the final result. Then we compare with several models, including WGAN, WGAN-GP, LS-GAN and BEGAN. They are recently proposed to extend the

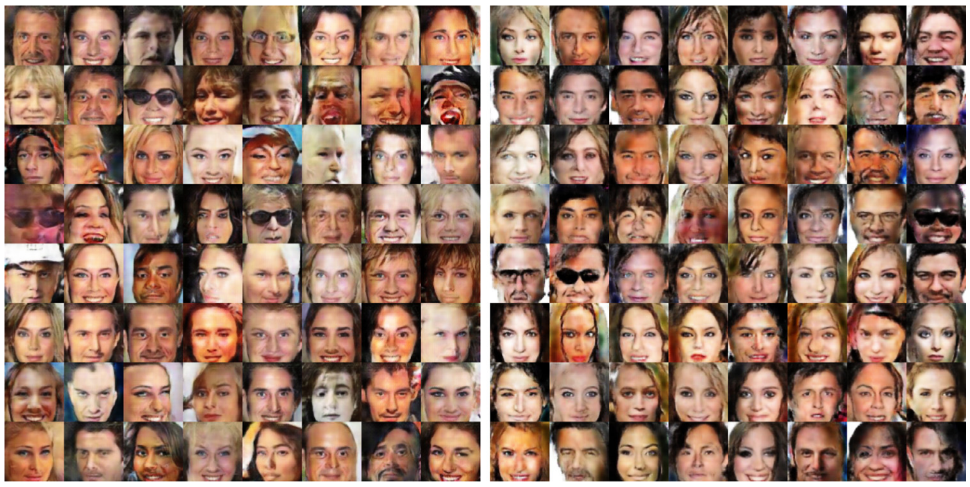


Fig. 7. Comparison of generated samples of DCGAN and KM-GAN on CelebA.

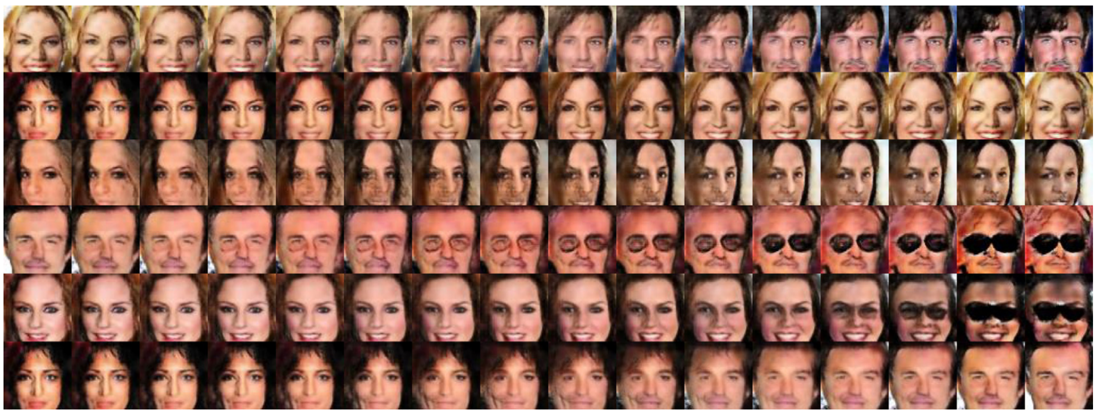


Fig. 8. Interpolations of generated images on CelebA dataset.

Table 9
FID of different models on CelebA.

Models	FID
WGAN [48]	41.3 ± 2.0
WGAN-GP [48]	30.0 ± 1.0
LSGAN [48]	53.9 ± 2.8
BEGAN [48]	38.9 ± 0.9
DCGAN	28.9 ± 1.9
KMGAN	$\mathbf{31.5 \pm 0.5}$

objectives of GANs from different perspectives. The results of them are reported in the large-scale study [48] with similar network architectures and validating techniques to us. These models are trained in an un-conditional manner since CelebA is of no real labels. The final comparison with them is shown in Table 9.

From Table 9, our model out-performs all other models by a large margin except WGAN-GP and DCGAN. Among these two models, DCGAN is designed with a suitable convolutional architectures while WGAN-GP extends WGAN with a proposed gradient penalty to stabilize the optimization process. Although our result is a little lower than theirs, we could explore these two directions for KM-GAN since the two directions are compatible with our framework. Besides, the difference between results of KM-GAN and WGAN-GP on CelebA is obviously smaller than on CIFAR-10, which also demonstrates the potential of KM-GAN on un-labeled data.

4.5.2. Linear interpolation

Then we interpolate synthesized images to demonstrate the generalization capability of KM-GAN rather than only generate the training face images. We first interpolate $\mathbf{z} \in \mathbb{R}^{100}$ and then map interpolated $\mathbf{z}$ with the generator. The results are as shown in Fig. 8. The leftmost and rightmost images are mapped from $\mathbf{z}_0$ and $\mathbf{z}_1$, respectively. The other images are generated from $\mathbf{z}_\beta = \beta\mathbf{z}_0 + (1-\beta)\mathbf{z}_1 (\beta \in [0, 1])$, i.e., interpolations of corresponding noise vectors. As shown in Fig. 8, generated images change smoothly from leftmost to rightmost. Indeed, we choose features of faces, including hair color, angles of faces, with or without eyeglasses and some other special features, to exhibit the continuous change clearly. Especially, on the first row, the face of a smiling woman with golden hair transits to the face of a seriously man with dark hair slowly. In addition, on the second row, the face of a woman with dark hair and close mouth changes to the face of a smiling woman with golden hair. These interpolations indicate that our proposed KM-GAN is able to generate images continuously instead of only memorizing training data.

5. Conclusion

In this paper, we propose an un-conditional extension of GANs, called KM-GAN, by fusing GANs with the idea of K-Means and utilizing the clustering results to propose objective functions that direct the generating process. The purpose is to replace the role of one-hot real labels with the clustering results, which generalizes GANs to applications where real labels are expensive or impossible to obtain. In addition, we conduct experiments on several real-world datasets to demonstrate that KM-GAN is really capable of generating realistic and diverse images without mode collapse. In the future, we would further pay attention to proving the positive correlation between high-quality synthesized images and high clustering accuracy and utilize the relationship to improve performance of both tasks.

Declarations of interest

None.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant number: 61801325), the Natural Science Foundation of Tianjin City (Grant number: 18JCQNJC00600), Huawei Innovation Research Program (Grant number: HO2018085347), and the authors would like to thank Jianshi Yan, Jiaxiang Guo, Tianli Liao, Yifang Xu, Bowen Wu, Mengya Zhang, Chengdong Zhao and Dong Wang for their helpful advices.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[2] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784(2014).
[3] G. Dai, J. Xie, Y. Fang, Metric-based generative adversarial network, in: Proceedings of the 2017 ACM on Multimedia Conference, ACM, 2017, pp. 672–680.
[4] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434(2015).
[5] Z.-Y. Dou, Metric learning-based generative adversarial network, arXiv preprint arXiv:1711.02792(2017).
[6] X. Mao, Q. Li, H. Xie, R.Y. Lau, Z. Wang, S.P. Smolley, Least squares generative adversarial networks, in: Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 2813–2821.
[7] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, S. Belongie, Stacked generative adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2, 2017.
[8] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: Proceedings of the International conference on machine learning, 2016, pp. 478–487.
[9] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 3861–3870.
[10] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5147–5156.
[11] E. Aljalbout, V. Golkov, Y. Siddiqui, D. Cremers, Clustering with deep learning: taxonomy and new methods, arXiv preprint arXiv:1801.07648(2018).
[12] Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: an unsupervised and generative approach to clustering, in: Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 1965–1972.
[13] H. Gao, H. Huang, Joint generative moment-matching network for learning structural latent code., in: Proceedings of the IJCAI, 2018, pp. 2121–2127.
[14] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv preprint arXiv:1312.6114(2013).
[15] Y. Li, K. Swersky, R. Zemel, Generative moment matching networks, in: Proceedings of the International Conference on Machine Learning, 2015, pp. 1718–1727.
[16] V. Premachandran, A.L. Yuille, Unsupervised learning using generative adversarial training and clustering, 2016.
[17] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: interpretable representation learning by information maximizing generative adversarial nets, in: Proceedings of the Advances in Neural Information Processing systems, 2016, pp. 2172–2180.
[18] M. Ben-Yosef, D. Weinshall, Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images, arXiv preprint arXiv:1808.10356(2018).
[19] S. Mukherjee, H. Asnani, E. Lin, S. Kannan, Clustergan: latent space clustering in generative adversarial networks, arXiv preprint arXiv:1809.03627(2018).
[20] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980(2014).
[21] C.-C. Hsu, C.-W. Lin, Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data, IEEE Trans. Multimed. 20 (2) (2018) 421–429.
[22] M.M. Fard, T. Thonet, E. Gaussier, Deep *k*-means: jointly clustering with *k*-means and learning representations, arXiv preprint arXiv:1806.10069(2018).
[23] C. Doersch, Tutorial on variational autoencoders, arXiv preprint arXiv:1606.05908(2016).
[24] Y. Ren, J. Zhu, J. Li, Y. Luo, Conditional generative moment-matching networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 2928–2936.
[25] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 214–223.
[26] S. Nowozin, B. Cseke, R. Tomioka, F-GAN: Training generative neural samplers using variational divergence minimization, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 271–279.
[27] G.-J. Qi, Loss-sensitive generative adversarial networks on lipschitz densities, arXiv preprint arXiv:1701.06264(2017).
[28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of wasserstein gans, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 5767–5777.
[29] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, Oakland, CA, USA, 1967, pp. 281–297.
[30] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
[31] D. Sculley, Web-scale k-means clustering, in: Proceedings of the Nineteenth international conference on World wide web, ACM, 2010, pp. 1177–1178.
[32] Y. Wen, K. Zhang, Z. Li, Y. Qiao, A discriminative feature learning approach for deep face recognition, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 499–515.
[33] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011, 2011, p. 5.
[35] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3730–3738.
[36] E. Learned-Miller, G.B. Huang, A. RoyChowdhury, H. Li, G. Hua, Labeled faces in the wild: a survey, in: Proceedings of the Advances in Face Detection and Facial Image Analysis, Springer, 2016, pp. 189–248.
[37] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[38] S. Barratt, R. Sharma, A note on the inception score, arXiv preprint arXiv:1801.01973(2018).
[39] L. Theis, A.v. d. Oord, M. Bethge, A note on the evaluation of generative models, arXiv preprint arXiv:1511.01844 (2015).
[40] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 2234–2242.
[41] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local Nash equilibrium, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 6626–6637.
[42] A.A. Alemi, I. Fischer, Gilbo: One metric to measure them all, in: Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 7037–7046.
[43] K. Shmelkov, C. Schmid, K. Alahari, How good is my GAN? in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 213–229.
[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
[45] S. Arora, R. Ge, Y. Liang, T. Ma, Y. Zhang, Generalization and equilibrium in generative adversarial nets (GANs), arXiv preprint arXiv:1703.00573(2017).
[46] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, A. Courville, Adversarially learned inference, arXiv preprint arXiv:1606.00704(2016).
[47] Y. Pu, W. Wang, R. Henao, L. Chen, Z. Gan, C. Li, L. Carin, Adversarial symmetric variational autoencoder, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 4330–4339.
[48] M. Lucic, K. Kurach, M. Michalski, S. Gelly, O. Bousquet, Are GANs created equal? A large-scale study, in: Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 698–707.

Ce Wang received the B.S. degree from the Department of Mathematics, Jilin University, in 2015. He is currently pursuing the Ph.D. degree with the Center for Combinatorics, Nankai University. His main interests include generative models, image processing, machine learning, and deep learning.

Zhangling Chen is currently a Ph.D. candidate who majors in applied mathematics Tianjin University. Her research interests include deep learning, face recognition, object recognition and generative models.

Kun Shang received the B.S. degree from the Faculty of mathematics and statistics, Hubei University, in 2011 and Ph.D. degree in the Center for Applied Mathematics of Tianjin University, in 2018. He is currently an assistant professor in the College of Mathematics and Econometrics of Hunan University in China. His main interests include pattern recognition, image processing, low-rank tensor minimization and optimization theory and algorithm.

Huaming Wu received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universitäat Berlin, Germany in 2015. He is currently an associate professor in the Center for Applied Mathematics, Tianjin University. His research interests include model-based evaluation, wireless and mobile network systems, mobile cloud computing and deep learning.