Improving Generative Moment Matching Networks with Distribution Partition

Yong Ren,[1] **Yucen Luo,** [1] **Jun Zhu** [1*]

[1] Department of Computer Science and Technology, Institute for AI, BNRist Center, Tsinghua University
reny11@foxmail.com, luoyc15@mails.tsinghua.edu.cn, dcszj@mail.tsinghua.edu.cn

Abstract

Generative moment matching networks (GMMN) present a theoretically sound approach to learning deep generative models. However, such methods are typically limited by the high sample complexity, thereby impractical in generating complex data. In this paper, we present a new strategy to train GMMN with a low sample complexity while retaining the theoretical soundness. Our method introduces some auxiliary variables, whose values are provided by a pre-trained model such as an encoder network in practice. Conditioned on these variables, we partition the distribution into a set of conditional distributions, which can be effectively matched with a low sample complexity. We instantiate this strategy by presenting an amortized network called GMMN-DP with shared auxiliary variable information for the data generation task, as well as developing an efficient stochastic training algorithm. The experimental results show that GMMN-DP can generate complex samples on datasets such as CelebA and CIFAR-10, where the vanilla GMMN fails.

Introduction

Deep generative models have achieved great success on tasks with uncertainty modeling, such as image generation (Ledig et al. 2017), missing data imputation (Li et al. 2018b) and transfer learning (Isola et al. 2017). Among various models, generative moment matching networks (GMMN) (Li, Swersky, and Zemel 2015) present an attractive choice. GMMN adopts the maximum mean criterion (MMD) (Gretton et al. 2008) as the objective, which is simple and theoretically sound — when the underlying kernel is characteristic (Sriperumbudur, Fukumizu, and Lanckriet 2011), MMD is capable of distinguishing any two different distributions. This approach has been extended for conditional generation and classification tasks (Ren et al. 2016).

One challenge that limits the wide application of GMMN is that the involved kernel embedding of distributions in MMD are global statistics requiring a high sample complexity in general. In practice, due to the computational cost and memory constraints, stochastic gradient descent (SGD) with reasonably small-sized mini-batches is the most effective choice to train GMMN, which however is likely to have high variance because of the high sample complexity. Therefore, the vanilla GMMN can only generate data with low-dimensional latent representations, such as the samples from the MNIST dataset. For those complex datasets such as CelebA and CIFAR-10, a small batch size (i.e. 64 or 128) in practice cannot meet the sample complexity requirements to converge (Li, Swersky, and Zemel 2015).

There are some attempts towards lowering the sample complexity required to perform stochastic training for GMMN. These methods usually construct new kernels with a stronger statistic testing power, involving a function that extracts meaningful features from data and then combines it with some other kernels. For example, MMD-GAN (Li et al. 2017) uses adversarial training to learn a target-specific feature extractor z, and to keep the characteristic property, z is required to be injective and is learned approximately by special constraints. Sutherland et al. (2017) get z from a pre-defined set of injective functions by directly maximizing the test power objective.

In this paper, we present a new approach to improving the performance of GMMN. The most striking difference is that instead of constructing data-specific kernels, we use a fixed one, preventing the notorious problem of instability in training several networks alternately (Arjovsky, Chintala, and Bottou 2017), i.e., the generator and the kernel networks. We process the data distribution to adapt the fixed kernel such that stochastic training algorithms with a small mini-batch size can give an accurate estimator involved in the objective function.When the kernel is characteristic, our method enjoys the theoretical advantages of moment matching. Specifically, the main contributions of this work lie in three folds:

- We analyze the difficulty of the vanilla GMMN and propose an alternative training strategy which requires a lower sample complexity intrinsically. Such method is based on the distribution partition, where we introduce additional auxiliary random variables Y and a pre-training step to transform the data generating process $X \sim P_d(X)$ into a two-stage one (i.e. $Y \sim P(Y), X|Y \sim P(X|Y)$) with additional requirements that the marginal $P(X) = P_d(X)$ and the randomness in $P(X|Y)$ can be covered by small data sizes. Instead of matching $P_d(X)$ directly, we match $P(X|Y = \boldsymbol{y})$ for each $\boldsymbol{y}$ separately via moment based criterion and finally recover the target distribution with guarantees.

[*]Corresponding author

- Based on the proposed strategy, we present several distribution partition methods, and an amortized network structure called GMMN-DP as well as an efficient stochastic training algorithm. GMMN-DP can be regarded as matching a series of conditional embedding operators in an amortized manner.
- We present empirical results showing that compared with the original GMMN, our proposed methods can produce samples of much better quality on benchmark datasets MNIST, CelebA and CIFAR-10 with a small mini-batch size (e.g., 64), while GMMN fails on the later two. Empirical stability and computational complexity are also analyzed.

Problem Settings and Preliminary

We consider the following data generation problem. Given a dataset $\mathcal{D}$ consisting of samples $\{\boldsymbol{x}_i\}_{i=1}^N$ drawn from an unknown distribution P_d, our target is to generate samples from it. We here briefly review some preliminary knowledge, including kernel embedding of (conditional) distributions and their applications in generative modeling.

Kernel Embedding of Distributions

We start with a brief overview of Hilbert space embedding for distributions, where we embed distributions as elements in a *reproducing kernel Hilbert space* (RKHS). Fixing an RKHS $\mathcal{F}$ on $\mathcal{X}$ with kernel k, for a distribution P over $\mathcal{X}$, the embedding takes expectation on its feature map $\phi(x) := k(x, \cdot)$:

$$\mu_X := \mathbb{E}_X[\phi(X)] = \int_{\mathcal{X}} \phi(\boldsymbol{x}) P(d\boldsymbol{x}).$$

Under some regularity conditions, μ_X is guaranteed to be an element in the RKHS (Gretton et al. 2008).

For RKHS $\mathcal{G}$ on $\mathcal{Y}$ with kernel k' and $P(X,Y)$ over $\mathcal{X} \times \mathcal{Y}$. The embedding can be extended to conditional distributions $P(X|Y)$, which is defined point-wise as:

$$\mu_{X|y} := \mathbb{E}_{X|y}[\phi(X)] = \int_{\mathcal{X}} \phi(\boldsymbol{x}) P(d\boldsymbol{x}|y).$$

The above embedding can be represented as an operator $C_{X|Y} : \mathcal{G} \to \mathcal{F}$, which satisfies the following properties:

1. $\mu_{X|y} = C_{X|Y}\phi(y)$; 2. $\mathbb{E}_{X|y}[g(X)|y] = \langle g, \mu_{X|y} \rangle_{\mathcal{F}}$.

Under some conditions, the operator does exist (Song et al. 2009) and has the representation $C_{X|Y} = C_{XY} C_{YY}^{-1}$, where the $C_{XY} : \mathcal{G} \to \mathcal{F}$ is the cross-covariance operator:

$$C_{XY} := \mathbb{E}_{XY}[\phi(X) \otimes \phi'(Y)] - \mu_X \otimes \mu_Y.$$

where $\otimes$ is the tensor product and C_{XY} can also be viewed as an element in the tensor product space $\mathcal{F} \otimes \mathcal{G}$.

In practice, finite sample estimations for μ_X and $C_{X|Y}$ can be obtained from N *i.i.d.* samples $\mathcal{D}_{XY} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ drawn from the joint distribution $P(X,Y)$:

$$\widehat{\mu}_X = \frac{1}{N}\sum_{i=1}^N \phi(\boldsymbol{x}_i), \quad \widehat{C}_{X|Y} = \Phi(K + \lambda I)^{-1}\Upsilon^\top,$$

where $\Upsilon = (\phi'(\boldsymbol{y}_1), ..., \phi'(\boldsymbol{y}_N))$, $\Phi = (\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N))$, $K = \Upsilon^\top\Upsilon$ and λ is a regularization term.

Generative Modelling with Kernel Embedding

Generative modelling aims to learn a generator G, usually parameterized by deep networks, from which we can draw samples $X' \sim P_g$. The objective is to minimize the distance between P_g and P_d. One strategy is to directly compare their embeddings $L_{\text{MMD}} = \|\mu_{X'} - \mu_X\|_{\mathcal{F}}$, which is called Maximum Mean Discrepancy (MMD) (Gretton et al. 2008). The most remarkable property is that when the underlying kernel is characteristic (e.g., the commonly used RBF kernels), MMD is able to distinguish any two different distributions because the embedding is an injective mapping. With samples $D'_X = \{\boldsymbol{x}'_i\}_{i=1}^M$ from P_g, we can estimate L_{MMD} with its finite sample version:

$$\widehat{L}^2_{\text{MMD}} = \left\| \frac{1}{N}\sum_{i=1}^N \phi(\boldsymbol{x}_i) - \frac{1}{M}\sum_{j=1}^M \phi(\boldsymbol{x}'_j) \right\|^2_{\mathcal{F}} \tag{1}$$

Using the kernel trick, Eq.1 can be computed in closed form (Gretton et al. 2008).

In the conditional generating case $P(X|Y)$, both the dataset and generated samples are conditioned on some auxiliary variables Y. We model the joint distribution $P(X,Y)$ and the generator can provide samples $(\boldsymbol{x}', \boldsymbol{y}') : \boldsymbol{y}' \sim P(Y), \boldsymbol{x}'|\boldsymbol{y}' \sim P_g$. The MMD is then replaced by its conditional version– Conditional Maximum Mean Discrepancy (CMMD) (Ren et al. 2016), where the conditional embedding operators are compared using $L_{\text{CMMD}} = \|C_{X|Y} - C_{X'|Y'}\|_{\mathcal{F}\otimes\mathcal{G}}$. With samples $D'_{XY} = \{(\boldsymbol{x}'_i, \boldsymbol{y}')\}_{i=1}^M$ from P_g, the finite sample estimator is:

$$\widehat{L}^2_{\text{CMMD}} = \left\| \Phi(K{+}\lambda I)^{-1}\Upsilon^\top - \Phi'(K'{+}\lambda I)^{-1}\Upsilon'^\top \right\|^2_{\mathcal{F}\otimes\mathcal{G}},$$

where Φ', K', Υ' are defined similarly for D'_{XY}.

Our Method

We now present our method. We first propose the general idea of matching with distribution partition, followed by discussions on practical distribution partition method. Finally we present our network GMMN-DP and its corresponding stochastic training algorithm. Suppose we have RKHS $\mathcal{F}$ on $\mathcal{X}$ with kernel k, RKHS $\mathcal{G}$ on $\mathcal{Y}$ with kernel k'.

Matching with Distribution Partition

When using MMD as the objective for generative modeling as in GMMN (Li, Swersky, and Zemel 2015), the computation cost of the kernel gram matrix is $O(n^2 k)$, where n is the sample size and k is the feature dimension. This prevents us from using a large sample size (e.g., the whole dataset), hence we have to resort to stochastic mini-batch training. However, the main difficulty arises from obtaining accurate estimations for the embedding μ_X, which in general converges in a sublinear $O(1/n^2)$ rate. In practice, the mini-batch size is usually set to be less than 128, which fails to estimate μ_X well on datasets with complex and diverse samples, e.g. CelabA and CIFAR-10.

Instead of matching the whole distribution P_d directly, we propose to match it in a *divide and conquer* manner. The overall randomness in P_d is strong [1], which requires a large mini-batch size to capture. However, if we partition the whole space into small ones with weak randomness, we can expect to match each one with a small mini-batch size.

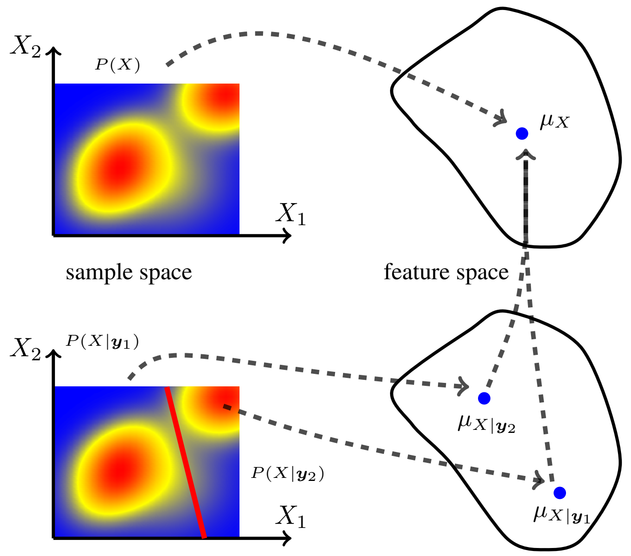


Figure 1: Illustration of the matching process. The sample space is partitioned into small sub-spaces. Instead of matching μ_X, we match $\mu_{X|\boldsymbol{y}}$ for each sub-space and eventually recover the whole distribution.

The overall illustration is shown in Fig.1. Specifically, suppose for now we want to match P_d with P_g, where the later one can be parameterized by a network as discussed later.We first introduce some auxiliary variables Y to partition the whole sample space $\mathcal{X}$ into sub-spaces $\{\mathcal{X}_i\}_{i\in\mathcal{I}}$ with $\mathcal{X} = \bigcup_{i\in\mathcal{I}} \mathcal{X}_i$, where each sub-space is indicated by $\boldsymbol{y}_i \sim P(Y)$. Note that we do not require the sub-spaces $\mathcal{X}_i$ are disjoint with each other. Here we slightly abuse the term "partition" since we allow overlapping. Under this partition, we actually introduce a two-step sample generating procedure for (X, Y), where we first draw the indicator $Y \sim P(Y)$ and then draw samples X from the conditional distribution $P(X|Y)$, where the sample space for $P(X|Y = \boldsymbol{y}_i)$ is $\mathcal{X}_i$. We assume $P(Y)$ is fixed and known. There are two necessary requirements for the above process:

- The marginal distribution $P(X) = \int_{\mathcal{Y}} P(X, d\boldsymbol{y})$ is the same as the data distribution $P_d(X)$, where $\mathcal{Y}$ is the corresponding sample space for Y.
- The randomness in each $P(X|\boldsymbol{y})$ is weak, where the term "weak" means we can use a small batch-size to obtain a relatively accurate estimation of its kernel embedding.

A detailed discussion about the partition step shall be shown below. After the partition step, we match each $P(X|\boldsymbol{y})$ with $P_g(X|\boldsymbol{y})$. There are two cases. When the domain of $\boldsymbol{y}$ is discrete and finite, we can use MMD to match $P(X|\boldsymbol{y})$ with our generators $P_g(X|\boldsymbol{y})$ separately.

In general the domain of $\boldsymbol{y}$ is continuous, we cannot exhaust every $\boldsymbol{y}_i$. In this case, we further group a series of $\boldsymbol{y}$ and use CMMD to match the operator $C_{X|Y'}$ for $P(X|Y')$ with $C_{X'|Y'}$ for our generator $P_g(X'|Y')$, where the sample space of Y' is the grouped values. It is worth pointing out that here we do not estimate $C_{X|Y}$ for the whole conditional distribution $P(X|Y)$, which in general is a global statistic that needs large sample size (Song et al. 2009). Instead, we only match a sub-set of the domain of (X, Y). For example, suppose the generating process is $Y \sim \frac{1}{K}\sum_{i=1}^{K}\mathcal{U}(2i, 2i + 1)$, $X|Y \sim \mathcal{N}(Y, 1)$, where $\mathcal{U}$ is a uniform distribution and $\mathcal{N}$ is a Gaussian. If we want to know the conditional embedding $\mu_{X|Y=2}$, instead of estimating $C_{X|Y}$, we can only estimate $C_{X|Y_1'}$, where $Y_1' \sim U(2 * 1, 2 * 1 + 1)$.

After the matching step, we can draw samples (X, Y) via $Y \sim P(Y), X|Y \sim P_g(X|Y)$ and we have $X \sim P_d(X)$, which means that $P_d(X)$ is recovered. We summarize the results in theorem 1 and leave the details in appendix A.

Theorem 1. *Suppose that $\mathbb{E}_{\mathcal{X}}\sqrt{k(\boldsymbol{x},\boldsymbol{x})} < \infty$ and for each Y' and $f \in \mathcal{F}, \mathbb{E}[f(X)|Y'] \in \mathcal{G}$ in the continuous case. When the kernel k is characteristic, the true distribution $P_d(X)$ can be matched with $P_g(X)$ by the above procedure.*

Distribution Partition with Samples

One remaining important problem is that how we partition the distribution P_d to satisfy the two requirements. When the data generating process $X \sim P_d(X)$ is inherently hierarchical and known to us, we can partition the distribution according to the structure. Unfortunately, in general, we do not know it and hence resort to a reverse process, where we have N samples $\mathcal{D} = \{x_i\}_{i=1}^{N}$ from the true distributions $P_d(X)$ and we partition the samples to approximate the true distribution. When the sample size N approaches to infinity, we can approximate $P_d(X)$ arbitrarily accurate.

The simplest method is to randomly partition samples into small sub-sets, each with an indicator such as the index. The size of the sub-sets is designed so that the small mini-batch size can give accurate approximation to the embeddings of the sub-sets. This method naturally satisfies the two requirements asymptotically. For datasets of high diversity, such as CIFAR-10, due to the constraint of memory and computation cost, random partition leads to very small sub-sets (i.e., < 256 in practice) and similar mini-batch size in the training process. In this case, it leads to several drawbacks. Since there are no meaningful correlations between the indicators, we only get isolated generators without utilizing the inherent similarity between samples. That is to say, the model is trying to memorize every single sample by rote. Consequently, we need larger models and the convergence rate is slow, as shown in the experiments section.

We discuss an alternative here. The overall idea is to introduce a *pre-trained* model to provide $\boldsymbol{x}$ with meaningful latent code $\boldsymbol{y}$ with the property that similar $\boldsymbol{y}$ corresponds to similar $\boldsymbol{x}$. Specifically, we build an inverse mapping $\mathcal{M} : \boldsymbol{x} \to P(Y|\boldsymbol{x})$ and randomly allocate each $\boldsymbol{x}_i$ with some $\boldsymbol{y}_i$ (e.g. each $\boldsymbol{x}_i$ has one $\boldsymbol{y}_i$ in practice). Those $\boldsymbol{x}_i$'s with the

[1]Typically, the randomness in a distribution is measured by entropy. Here the term "strong" means the embedding cannot be estimated well with small sample size such as 64.

same $\boldsymbol{y}_i$ are grouped as a sub-set of $\mathcal{D}$ and all the sub-sets form an approximate partition (perhaps with overlaps) of the data distribution. The randomness in each subset of the partition is controlled by the size and the similarity of the samples (i.e., small size and high similarity enjoy weak randomness). The partition depends on the quality of pre-trained model, diversity in samples and randomness in the mapping $\mathcal{M}$. For example, if mapping $\mathcal{M}$ have no randomness (e.g. $\mathcal{M} : \boldsymbol{x} \to 0$), all samples form a single set; if the precision of $\mathcal{M}$ is one decimal instead of two, samples are inclined to group together.

The choice of the pre-trained mapping $\mathcal{M}$ is flexible and important, which involves how to learn the latent representations of special interest. Here we list two common choices:

- *Encoder-decoder based models:* The encoders of this type of models can usually capture the latent structure in the data. Thus they naturally provide the mapping $\mathcal{M}$, such as denoising auto-encoders (DAEs) (Vincent et al. 2008), variational auto-encoders (VAEs) (Kingma and Welling 2013) and their variants.
- *Clustering based methods:* Clustering discovers the similarity between data. There are many clustering methods that yield meaningful latent codes (Xie, Girshick, and Farhadi 2016; Law, Urtasun, and Zemel 2017). For example, recently proposed clustering method with local aggregation (Zhuang, Zhai, and Yamins 2019) is able to find nearest neighbors of visual sense, which is a good choice for our pre-trained model.

When some auxiliary information is provided (e.g., supervision signals such as the label information), one can take them into account for better representation. How to make use of auxiliary information efficiently is beyond the scope of this paper, and here we only focus on the unsupervised case.

GMMN-DP Nets and Training Algorithm

From the above discussions, suppose now we have the partition for the samples $\mathcal{D} = \bigcup_{i=1}^{L} \mathcal{D}_i$ with a pre-trained mapping $\mathcal{M}$ and each data $\boldsymbol{x}_i$ with one corresponding latent code $\boldsymbol{y}_i$.

We start from the general case, where the domain of $\boldsymbol{y}$ is continuous, to define the generative moment matching networks with distribution partition (GMMN-DP). Similar with conditional generative moment matching network (CGMMN) (Ren et al. 2016), GMMN-DP depicts a conditional generating process, where the implicit generator G is parameterized by a neural network g_θ with parameters θ. The input of g_θ consists of two parts: the first part is the latent code $\boldsymbol{y}$, serving as an indicator of which part of the distribution the generated samples belong to and latent space information; and the other one is some randomly sampled data $\boldsymbol{z}$ to further give randomness in that part. $\boldsymbol{z}$ can be easily sampled from uniform distribution $U[0, 1]$ with pre-determined dimension.

We use stochastic training method. We first divide the whole dataset into small mini-batches. At each iteration, we sample a mini-batch $\mathcal{B}$ from the training set and generate a mini-batch samples $\mathcal{B}'$ of the same size from g_θ. We optimize the CMMD objective with proper regularizations by SGD. The algorithm is summarized in Alg. 1. Note that unlike GCMMN, which models a single operator of $C_{X|Y}$, GMMN-DP can be interpreted as modelling a series of conditional embedding operator $C_{X|Y_i'}$ in an amortized manner, where the union of sample spaces of Y_i''s equals to the Y's.

```
Input: Pre-trained mapping $\mathcal{M} : \boldsymbol{x} \to P(Y|\boldsymbol{x})$,
dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, mini-batch size $B$, optimizer
ADAM.
Output: Generator $g_\theta$ parameterized by $\theta$ with
distribution $P_g(X|Y)$.
Use $\mathcal{M}$ to augment each $\boldsymbol{x}_i$ with a $\boldsymbol{y}_i$, $\widetilde{\mathcal{D}} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}$
Divide data $\widetilde{\mathcal{D}}$ into mini batches $\Gamma$ with batch size $B$.
repeat
  1. Draw a mini batch $\mathcal{B}$ from $\Gamma$.
  2. For every $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ from $\mathcal{B}$, generate a sample
  $\boldsymbol{x}_i'|\boldsymbol{y}_i$ from $g_\theta$, which forms $\mathcal{B}' = \{\boldsymbol{x}_i', \boldsymbol{y}_i\}$.
  3. Computing $l = \widehat{\text{CMMD}}(\mathcal{B}, \mathcal{B}')$.
  4. Update $\theta$ using ADAM with gradient $\frac{\partial l}{\partial \theta}$.
until Convergence
```

Algorithm 1: Stochastic training for GMMN-DP

Recall that the generating process for GMMN-DP is to sample $Y \sim P(Y)$ first and then $X|Y \sim P_g(X|Y)$. The mapping process gives $P(Y) = \frac{1}{N}\sum_{i=1}^{N} \delta(Y - \boldsymbol{y}_i)$ when we allocate each $\boldsymbol{x}_i$ with one $\boldsymbol{y}_i$. Notice that with finite samples D, what we recover is the empirical distribution in theory with $\boldsymbol{y}_i$ regarded as "anchors". To capture the whore latent space, we can use kernel density estimator (KDE) with Gaussian kernels K_h concentrated on each $\boldsymbol{y}_i$: $P(Y) = \sum_{i=1}^{N} K_h(Y - \boldsymbol{y}_i)$ and regard $\boldsymbol{y}_i$ as samples from it, where h is kernel bandwidth. To generate new samples, we can first get $\boldsymbol{y}$ from the KDE and then sample from $P_g(X|\boldsymbol{y})$.

The case that the domain of Y is discrete and finite is not common, since it requires that the dataset is relatively small and has low-dimensional latent space. In this case, we can construct the same network structure as GMMN-DP and replace CMMD with MMD to match each $P(X|\boldsymbol{y}_i)$ separately, which is also an amortized network modelling a series of distributions.

Related Work

There are a diverse range of deep generative models using stochastic training algorithms with small batch size. We review some related work here, pointing out how they solve the sample complexity problem and their relations to ours.

Localization: Most generative models can be classified into to some categories. The moment matching based methods are "global" ones, where by global we mean that the objective is a statistic involving all samples theoretically. Consequently, the stochastic training algorithm usually introduces huge variance, failing to capture the global information. However, "local" methods bypass global statistics, among which the most representative and typical ones are VAE (Kingma and Welling 2013) and GAN (Goodfellow et al. 2014). For VAE, the lower bound is defined separately for each sample. This essentially introduces local parameters (i.e. the encoder) to depict the generating process for each data point. In terms of GAN, the discriminator takes one $\boldsymbol{x}$ each time, also capturing local information instead of the global statistic. Though generative models based on the above two precursors (Arjovsky, Chintala, and Bottou 2017; Johnson et al. 2016; Li et al. 2018a) has been gaining traction in recent years, the encoder or the discriminator keeps localization. The localization treatment gathers information of the data incrementally, naturally adapted to the stochastic training algorithm. Our proposed method can be regarded as a localization treatment of the GMMN, where we match the target distribution by parts.

Maximize testing power of MMD: There are many attempts trying to make GMMN applicable on datasets with high diversity. Given a fixed dataset, most of them focus on maximizing the statistic testing power of the underlying kernels. Sutherland et al. (2017) takes the testing power as the objective directly and finds a kernel with maximum value from a pre-determined family. Li et al. (2017); Binkowski et al. (2018) construct the kernel in an adversarial learning way, by extracting powerful features from discriminators. To keep the kernel characteristic, Li et al. (2017) achieved the injection property by using an encoder-decoder structure to reconstruct the real and generated samples, while Binkowski et al. (2018) proposed repulsive loss with explicitly maximizing the pair-wise distances among the real samples. Our proposed method works in an opposite direction. Instead of learning a data-specific kernels, we keep kernel fixed and process data to take better advantage of the testing power.

Experiments

We present experimental results of the generative task on the commonly used datasets MNIST (LeCun et al. 1998), CelebA (Liu et al. 2015) and CIFAR-10. First we list some settings shared across all the experiments if not further pointing out. The code can be found HERE[2].

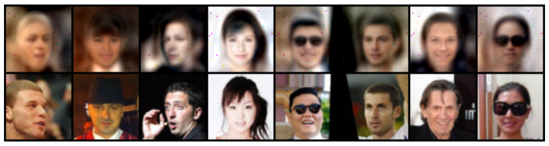


Figure 2: Samples of the reconstruction results of the pre-trained VAE on dataset CelebA. The first row is the vague reconstructions of the corresponding original data in the second row.

Pre-trained data partition model: We use the vanilla VAE (Kingma and Welling 2013) as our distribution partitioner, i.e., the encoder as mapping $\mathcal{M} : \boldsymbol{x} \to P(Y|\boldsymbol{x})$. Both the encoder and decoder are several (i.e. 3 to 4 based on different datasets) MLP layers. The dimension of the final output (i.e. latent code) n_y is set according to the complexity of datasets. We use $n_y = 16$ for MNIST, $n_y = 64$ for CelebA and CIFAR-10. The pre-trained model is relatively small and not able to reconstruct the dataset, as shown in Fig.2. After giving each $\boldsymbol{x}_i$ a latent code $\boldsymbol{y}_i$, we do not further group similar latent codes so that each sub-set of the partition actually contains 1 sample with a large probability. We find this setting accelerates the convergence speed.

[2]https://github.com/McGrady00H/Improving-MMD-with-Distribution-Partition

Network architecture: For our GMMN-DP generator, we follow the architecture of the generator of DCGAN (Radford et al. 2016), which is composed of several layers (i.e., $3-5$ based on different datasets), of transposed convolution operator followed by batch normalization and ReLU. The model size is adapted to the complexity of the datasets.

Hyper-parameters: We use a mixture of 7 RBF kernels $K(x, x') = \sum_{i=1}^{7} K_{\sigma_i}(x, x')$ with σ_i to be $\{1, 4, 8, 16, 24, 32, 64\}$ for the sample space and $K(y, y')$ to be RBF kernel with $\sigma = 1$. The model is optimized using Adam with learning rate 0.001 and $\beta = (0.9, 0.999)$. The batch size B is set to be 64 for all the datasets. The regularization parameter λ for $\widehat{C}_{X|Y}$ is set to be 0.01. The dimension for the additional randomness $\boldsymbol{z}$ is set to be 2.

The Generative Quality

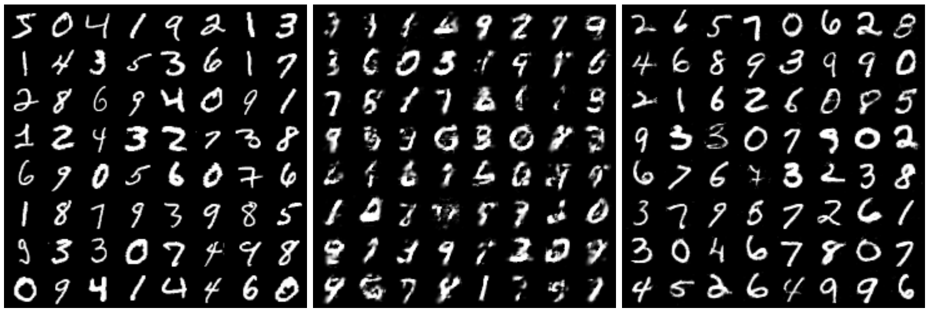
We start with the generating results on two standard benchmark datasets MNIST and CelebA. The output value of the pre-trained VAE ranges from -3 to 3 and we set the kernel bandwidth $h = 0.3$. As mentioned before, GMMNP-DP models a series of local conditional embedding operator $C_{X|Y}$ and is expected to generate samples for unseen $\boldsymbol{y}$s. We compare our GMMN-DP with the original GMMN. For fair comparison, the structure of GMMN and model size is the same as our GMMN-DP and the hyper-parameters are set to be the same as well.

The results are shown in Fig.3. We can easily find that our GMMN-DP can generate clear samples with sharp boundaries for the MNIST, which are indistinguishable from the original dataset. On the CelebA dataset, our GMMN-DP is able to capture details of the faces. On the contrary, GMMN can only generate samples with general graphic outlines.

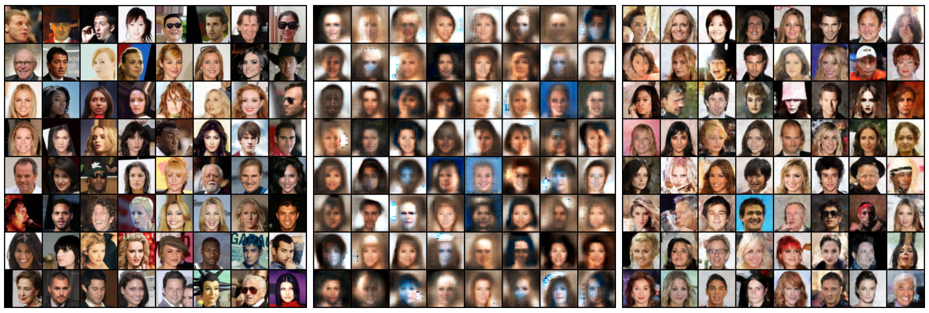
To verify that our GMMN-DP can make efficient use of the similarity between samples by depicting a smooth conditional generative process instead of merely copying the dataset. We show generating results for the linear interpolations of the latent code $\boldsymbol{y}$. We first fix a latent code $\boldsymbol{y}$ generated by the VAE mapping. Then we vary $\boldsymbol{y}$ by making linear interpolation between $[-3, 3]$ just for 1 dimension while keep the other dimensions fixed. The results are shown in Fig.4. We can find that the transformations are continuous and the images are of good quality, both on the dataset MNIST and CelebA. This verifies that the conditional embedding operators $C_{X|Y'}$ can generalize well for the unseen data.

We then report the generating results for the more challenging dateset CIFAR-10. We set the kernel bandwidth $h = 0.15$ or 0.08 for our GMMN-DP, which are balanced values between the sample quality and the generalization.

We first provide quantitative analysis. We use the inception score (IS) on CIFAR-10 images to measure the quality and diversity of generated samples. The implementation of IS follows Salimans et al. (2016) which is consistent with the



(a) MNIST samples (b) GMMN samples (c) GMMN-DP samples



(d) CelebA samples (e) GMMN samples (f) GMMN-DP samples

Figure 3: Original data samples and generated samples from GMMN and GMMN-DP (ours) on datasets MNIST and CelebA.

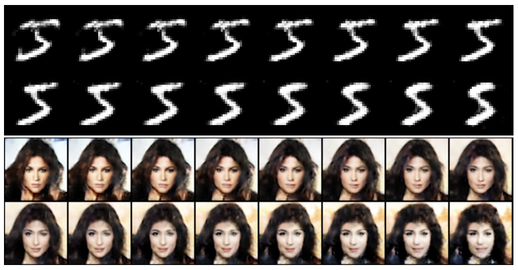
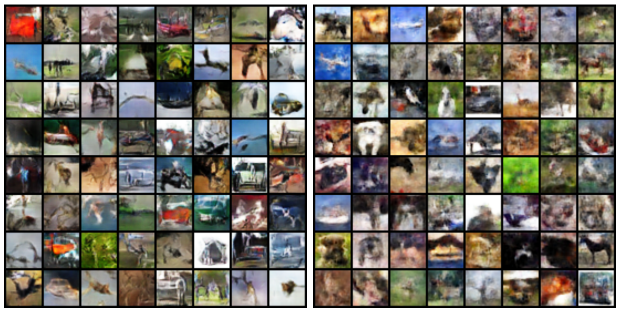


Figure 4: Linear interpolation results on MNIST and CelebA, where GMMN-DP can depict a smooth generative process.

METHODS	INCEPTION SCORES ± STD.
CIFAR-10 DATASET	11.95 ± 0.20
MMD-GAN	6.17 ± 0.07
WGAN	5.88 ± 0.07
GMMN	3.47 ± 0.03
GMMN-DP ($h = 0$)	$\mathbf{7.49} \pm 0.06$
GMMN-DP ($h = 0.08$)	$\mathbf{6.21} \pm 0.06$
GMMN-DP ($h = 0.15$)	5.08 ± 0.06

Table 1: Inception scores for GMMN-DP and some other related generative models.

results reported in other related work. The results are summarized in Table??. When the kernel bandwidth $h = 0$, which corresponds to the case that Y samples from the empirical distribution $P_0(Y) = \frac{1}{N}\sum_{i=1}^{N}\delta(Y - \boldsymbol{y}_i)$. In this case, the samples are inclined to be identical to the dataset, which matches the theoretical property recovering the data distribution. The corresponding IS 7.49 is highest as well. As we increase the kernel bandwidth, the IS decreases monotonously. This is an expected issue since that as we varying $\boldsymbol{y}$, the sample quality depends on the generalization ability of the estimated local conditional embedding operator $C_{X|Y}$. We conjure that the rate of decay is largely depends on the quality of the mapping $\mathcal{M}$, which is proved to some extent in further experiments (see Fig.8).



(a) MMD-GAN samples (b) GMMN-DP samples ($h = 0.15$)

Figure 5: Generated samples from MMD-GAN and GMMN-DP (ours) on dataset CIFAR-10.

Now we give some qualitative analysis. We compare our GMMN-DP with MMD-GAN (Li et al. 2017). The generator of MMD-GAN is set to be the same as our GMMN-DP. Fig.5 shows that both the MMD-GAN and our GMMN-DP can generate meaningful and diverse samples. The difference is

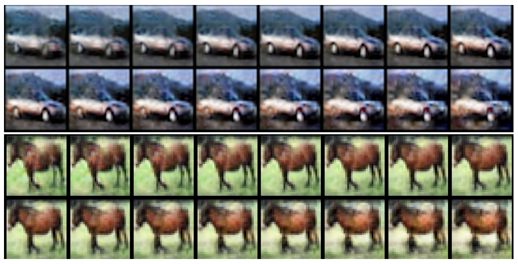


Figure 6: Linear interpolation results on dataset CIFAR-10. The global structure is maintained when only one dimension of the latent code varies.

that MMD-GAN samples have sharper boundaries while less global structure information. On the contrary, our GMMN-DP is trying to keep the global information as the randomness increasing while changing some details, which is further verified in Fig.6. It shows the linear interpolation results, where we vary only one dimension of the latent code with the others fixed. We can find the global structure changes while keeps meaningful information.

Stability of GMMN-DP

We experimentally illustrate the relation between the CMMD loss in GMMN-DP with the generated sample quality. We take the CelebA dataset as example. The result is shown in Fig.7. The samples are generated by a fixed $\boldsymbol{y}_i$ given by the VAE mapping and the loss is smoothed by a moving average. We can easily find that as the training processes, the loss decreases globally and the sample quality increases. The converge rate is rather fast as we can observe that the sample quality becomes high after tens of thousands of iterations.

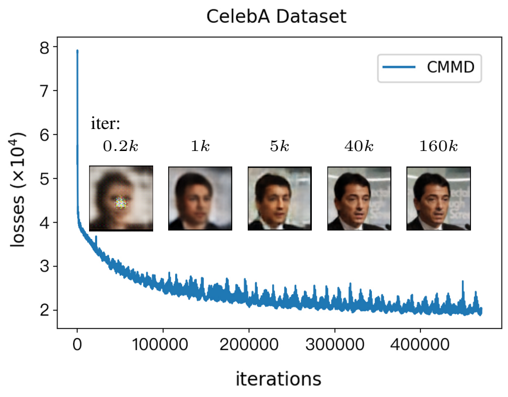


Figure 7: Illustration of the relation between the sample quality and the loss on dataset CelebA. There is a clear correlation between the sample quality and the loss.

Necessity of the Pre-trained Model

We investigate the importance of the pre-trained mapping $\mathcal{M}$ that meaningful latent code can accelerate the convergence speed. We use the dataset CIFAR-10 with 300×64 samples. The competitor is a random mapping $\mathcal{M}$ as mentioned before, which has no information about the structure of the samples. We keep other settings the same and the result is shown in Fig.8. Obviously, GMMN-DP with VAE pre-trained mapping convergences faster and has lower nadir. Based on this evidence, it is reasonable to conjure that the more informative the mapping $\mathcal{M}$ is, the better convergence rate and sample quality we can obtain.

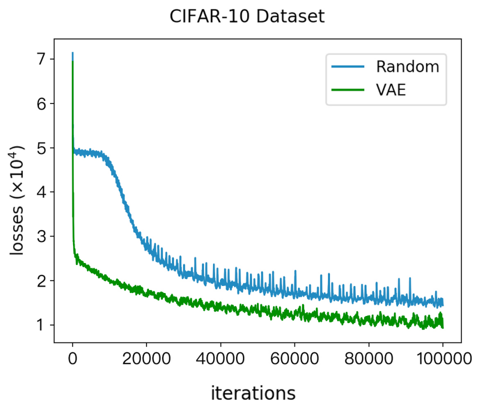


Figure 8: Illustration of the mapping quality and the convergence speed on CIFAR-10. A better result is obtained with more meaningful latent codes.

Computation Cost Analysis

We provide computational complexity analysis for GMMN-DP. For training with mini-batch size B, data dimension D and latent code dimension Z, each iteration is $O(B^3 + B^2(D + Z))$ with fixed kernels. The term $O(B^3)$ is the computation cost of the inverse manipulation on $(K + \lambda I)^{-1}$. In our experimental settings with a single RTX 2080ti GPU, the average time per iteration on CIFAR10 with $B = 64$ and model size $4,300$MB is 0.25s. When the dataset is partitioned into fixed mini-batches, the inverse matrix can be calculated and cached in advance to further reduce the computational complexity to $O(B^2 D)$ per iteration, which is the cost of the kernel gram matrix.

Conclusions and Discussions

We propose a new training method for MMD based generative models. Our method partitions the whole sample space into small sub-spaces, which break the strong randomness into weak ones that a small mini-batch size is sufficient to depict it. By matching each small sub-space, we can recover the whole distribution, without breaking the underlying universal property for kernels. We present the distribution partition methods, a GMMN-DP network and its supporting stochastic training algorithm for practical use. Experimental results show that our proposed methods can give promising results on high-dimensional latent space datasets such as CelebA and CIFAR-10, which have much better quality than the original GMMN and are comparable with some GAN based models.

Acknowledgements

This work was supported by NSFC Projects (Nos. 62061136001, 61620106010, U19B2034, U1811461), Beijing NSF Project (No. L172037), Beijing Academy of Artificial Intelligence (BAAI), Tsinghua-Huawei Joint Research Program, Huawei Hisilicon Kirin solution, the MindSpore team, and the NVIDIA NVAIL Program with GPU/DGX Acceleration.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. *ICML* .

Binkowski, M.; Sutherland, D. J.; Arbel, M.; and Gretton, A. 2018. Demystifying MMD GANS. *ICLR* .

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. *NeurIPS* .

Gretton, A.; Borgwardt, K.; Rasch, M.; Scholkopf, B.; and Smola, A. 2008. A kernel two-sample test. *JMLR* .

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* .

Johnson, M.; Duvenaud, D. K.; Wiltschko, A.; Adams, R. P.; and Datta, S. R. 2016. Composing graphical models with neural networks for structured representations and fast inference. *NeurIPS* .

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *ICLR* .

Law, M. T.; Urtasun, R.; and Zemel, R. S. 2017. Deep spectral clustering learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1985–1994. JMLR. org.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* .

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; and et al. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *CVPR* .

Li, C.; Welling, M.; Zhu, J.; and Zhang, B. 2018a. Graphical Generative Adversarial Networks. *NeurIPS* .

Li, C.; Zhu, J.; ; and Zhang, B. 2018b. Max-Margin Deep Generative Models for (Semi-)Supervised Learning. *TPAMI* .

Li, C.-L.; Chang, W.-C.; Cheng, Y.; Yang, Y.; and Póczos, B. 2017. MMD GAN: Towards Deeper Understanding of Moment Matching Network. *NeurIPS* .

Li, Y.; Swersky, K.; and Zemel, R. 2015. Generative moment matching networks. *ICML* .

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. *CVPR* .

Muandet, K.; Fukumizu, K.; Sriperumbudur, B.; and Schölkopf, B. 2017. Kernel Mean Embedding of Distributions: A Review and Beyond. *Foundations and Trends in Machine Learning* .

Radford, A.; Metz, L.; ; and Chintala, S. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR* .

Ren, Y.; Li, J.; Luo, Y.; and Zhu, J. 2016. Conditional Generative Moment-Matching Networks. *NeurIPS* .

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. *NeurIPS* .

Song, L.; Huang, J.; Smola, A.; and Fukumizu, K. 2009. Hilbert space embeddings of conditional distributions with applications to dynamical systems. *ICML* .

Sriperumbudur, B. K.; Fukumizu, K.; and Lanckriet, G. R. G. 2011. Universality, Characteristic Kernels and RKHS Embedding of Measures. *JMLR* .

Sutherland, D. J.; Tung, H.-Y.; Strathmann, H.; De, S.; Ramdas, A.; Smola, A.; and Gretton, A. 2017. Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy. *ICLR* .

Vincent, P.; Larochelle; Bengio, H.; Y.; and Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. *ICML* .

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487.

Zhuang, C.; Zhai, A. L.; and Yamins, D. 2019. Local Aggregation for Unsupervised Learning of Visual Embeddings. *ICCV* .

Çınlar Erhan, ed. 2011. *Probability and stochastics*. Springer.