Manifold Matching via Deep Metric Learning for Generative Modeling

Mengyu Dai*
Microsoft
mendai@microsoft.com

Haibin Hang*
University of Delaware
haibin@udel.edu

Abstract

We propose a manifold matching approach to generative models which includes a distribution generator (or data generator) and a metric generator. In our framework, we view the real data set as some manifold embedded in a high-dimensional Euclidean space. The distribution generator aims at generating samples that follow some distribution condensed around the real data manifold. It is achieved by matching two sets of points using their geometric shape descriptors, such as centroid and \( p \)-diameter, with learned distance metric; the metric generator utilizes both real data and generated samples to learn a distance metric which is close to some intrinsic geodesic distance on the real data manifold. The produced distance metric is further used for manifold matching. The two networks learn simultaneously during the training process. We apply the approach on both unsupervised and supervised learning tasks: in unconditional image generation task, the proposed method obtains competitive results compared with existing generative models; in super-resolution task, we incorporate the framework in perception-based models and improve visual qualities by producing samples with more natural textures. Experiments and analysis demonstrate the feasibility and effectiveness of the proposed framework.

1. Introduction

Deep generative models including Variational Autoencoder (VAE) [21], Generative Adversarial Networks (GAN) [13] and their variants [3, 26, 24, 42, 10, 55, 41] have achieved great success in generative tasks such as image and video synthesis, super-resolution (SR), image-to-image translation, text generation, neural rendering, etc. The above approaches try to generate samples which mimic real data by minimizing various discrepancies between their corresponding statistical distributions, such as using KL divergence [21], Jensen-Shannon divergence [13], Wasserstein distance [3], Maximum Mean Discrepancy [26] and so on. These approaches focused on the data distribution aspect and did not pay enough attention to the underlying metrics of these distributions. The interplay between distribution measure and its underlying metric is a central topic in optimal transport (cf. [47]). Despite that researchers have successfully employed optimal transport theory in generative models [3, 50, 9], simply assuming the underlying metric to be Euclidean metric may neglect some rich information lying in the data [1]. In addition, although the above approaches are validated to be effective, successful training setups are mostly based on empirical observations and lack of physical interpretations.

In this paper we bring up a geometric perspective which serves as an important parallel view of generative models as GANs. Table 1 summarizes the main differences between classic GANs and our proposed (so-called MvM) framework. Instead of directly matching statistical discrepancies under Euclidean distance, we provide a more flexible framework which is built upon learning the intrinsic distances among data points. Specifically, we treat the real data set as some manifold embedded in high-dimensional Euclidean space, and generate a fake distribution measure condensed around the real data manifold by optimizing a Manifold Matching (MM) objective. The MM objective is built on shape descriptors, such as centroid and \( p \)-diameter with respect to some proper metric learnt by a metric generator using Metric Learning (ML) approaches. During training process, the (fake data) distribution generator and the metric generator work interchangeably and produces better distribution (metric) that facilitates the efficient training of metric (distribution) generator. The learned distances can not only be used to formulate energy-based loss functions [22] for MM, but can also reveal meaningful geometric structures of real data manifold.

Table 1. Main differences between GANs and MvM.

<table>
<thead>
<tr>
<th>Differences</th>
<th>GANs</th>
<th>MvM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main point of view</td>
<td>statistics</td>
<td>geometry</td>
</tr>
<tr>
<td>Matching terms</td>
<td>means, moments, etc.</td>
<td>centroids, ( p )-diameters</td>
</tr>
<tr>
<td>Matching criteria</td>
<td>statistical discrepancy</td>
<td>learned distances</td>
</tr>
<tr>
<td>Underlying metric</td>
<td>default Euclidean</td>
<td>learned intrinsic</td>
</tr>
<tr>
<td>Objective functions</td>
<td>one min-max value function</td>
<td>two distinct objectives</td>
</tr>
</tbody>
</table>

*Equal contributions.
Manifold Matching: Shen et al. [45] proposed a nonlinear manifold matching algorithm using shortest-path distance and joint neighborhood selection, and illustrated its usage in medical imaging applications. Priebe et al. [39] investigated in manifold matching task from the perspective of jointly optimizing the fidelity and commensurability, with an application in document matching. Lim and Ye [28] decomposed GAN training into three geometric steps and used SVM separating hyperplane that has the maximal margins between classes. Lei et al. [25] showed the intrinsic relations between optimal transportation and convex geometry, and further used it to analyze generative models. Genevay et al. [12] introduced the Sinkhorn loss in generative models, based on regularized optimal transport with an entropy penalty. Shao et al. [44] introduced ways of exploring the Riemannian geometry of manifolds learned by generative models, and showed that the manifolds learned by deep generative models are close to zero curvature. Park et al. [38] added a manifold matching loss in GAN objectives which tried to match distributions using kernel tricks. However, the learning process highly relies on optimizing objectives in the original GAN framework. In addition, without proper metrics, using pre-defined kernels may fail to match the true shapes of data manifolds. In our work, the manifold matching is implemented using geometric descriptors under proper metrics learned by a metric generator.

Deep Metric Learning: Among rich sources of literature on deep metric learning, we mainly focus on a few that are related to our work. Xing et al. [51] first proposed distance metric learning with applications to improve clustering performance. Hoffer and Ailon implemented deep metric learning with Triplet network [18] which aimed to learn useful representations through distance comparisons. Duan et al. [11] proposed a deep adversarial metric learning framework to generate synthetic hard negatives from negative samples. The hard negative generator and feature embedding were trained simultaneously to learn more precise distance metrics. The metrics were learned in a supervised fashion and then used in classification tasks. Unlike [11], our approach utilizes geometric descriptors for matching data manifolds to generate data without using any labelled information. Mohan et al. [37] proposed a direction regularization method which tried to improve the representation space being learnt by guiding the pairs move towards right directions in the metric space. In this work, we utilize the approach in [37] for metric learning implementation.

Perception-Based SISR: SISR aims to recover a high-resolution (HR) image from a low-resolution (LR) one. Ledig et al. [23] first incorporated adversarial component in their objective and achieved high perceptual quality. However, SRGAN can generate observable artifacts such as undesirable noise and whitening effect. Similar issues were also mentioned in Sajjadi et al. [43]. Wang et al. [48] proposed ESRGAN which improved perceptual quality by improving SRGAN network architecture and combining PSNR-oriented network and a GAN-based network to balance perceptual quality and fidelity. Ma et al. [31] proposed a gradient branch which provides additional structure priors for the SR process. Utilization of the gradient branch need corresponding network architectures to be equipped with. Soh et al. [46] introduced natural manifold discriminator which tried to distinguish real and generated noisy and blurry samples. Since the natural manifold discriminator focuses on classifying certain types of manually generated fake data, one following question is: can we find a more robust way to learn useful information from real data? Thus in this case we view one usage of our work in SR task as an extension of the natural manifold discriminator. Some other recent methods [27, 8, 40, 29, 16, 6] mainly work on improving network architectures which are not directly comparable to our approach. In this paper we focus on objectives regardless of generator architectures, while the method can be incorporated into existing works.

3. Methodology

3.1. Proposed Framework

We propose a metric measure framework for generative modeling which contains a distribution generator \( f_\theta : \mathbb{R}^m \rightarrow \mathbb{R}^D \) and a metric generator \( g_w : \mathbb{R}^D \rightarrow \mathbb{R}^n \).

\[
\mathbb{R}^m \xrightarrow{f_\theta} \mathbb{R}^D \xrightarrow{g_w} \mathbb{R}^n
\]

The metric generator \( g_w \) would produce some metric \( d \) on \( \mathbb{R}^D \) to be the pullback of the Euclidean metric \( d_E \) on \( \mathbb{R}^n \) (see Definition 3.2). The distribution generator \( f_\theta \) would produce some measure \( \mu \) on \( \mathbb{R}^D \) to be the pushforward of some prior distribution \( \nu \) on \( \mathbb{R}^m \) (see Definition 3.1). In implementations, \( m, D, n \) represent dimensions of the input variable, target image, and image embedding respectively.
Now we have a metric measure space \((\mathbb{R}^D, d, \mu)\). The space of real data is viewed as some manifold \(M \subseteq \mathbb{R}^D\) embedded in Euclidean space. The measure \(\mu\) is said to be condensed around manifold \(M\) if the majority of the measure \(\mu\) is distributed nearby \(M\) (see Fig. 1). The manifold \(M\) is called totally geodesic (or “straight”) with respect to metric \(d\) if for any two points \(a, b \in M\), the shortest path measured by metric \(d\) stays on \(M\) (see Fig. 2). Using the generators \(f_\theta, g_w\) modeled as neural networks, we aim to find proper parameters \(\theta\) and \(w\) such that the induced measure \(\mu\) and metric \(d\) satisfies:

(1) \(\mu\) is as condensed as possible around \(M\);

(2) \(M\) is as “straight” as possible under \(d\).

![Figure 1](image1.png)

(i) The probability density function of a distribution \(\mu\) which condensed around a circle (manifold) \(M \subseteq \mathbb{R}^2\); (ii) The orange dots represents random samples of \(\mu\) and the green circle represents the real data manifold \(M \subseteq \mathbb{R}^2\).

![Figure 2](image2.png)

(ii) The blue segment represents the shortest path between two points \(a, b\) with respect to Euclidean distance \(d_E\). (i) A non-geodesic sub-manifold of \(\mathbb{R}^3\) under \(d_E\); (ii) A geodesic sub-manifold of \(\mathbb{R}^3\) under \(d_E\).

Generally speaking, the two networks would produce a sequence of metrics \(\{d_{2i}\}_{i \geq 0}\) and a sequence of measures \(\{\mu_{2i+1}\}_{i \geq 0}\) inductively and alternatively as follows: (i) Let \(d_0 = d_E\); Then for any \(i > 0\), (ii) derive measure \(\mu_{2i-1}\) using manifold matching based on metric \(d_{2i-2}\); (iii) derive metric \(d_{2i}\) using metric learning based on measure \(\mu_{2i-1}\).

\[d_0 = d_E \leadsto \mu_1 \leadsto d_2 \leadsto \mu_3 \leadsto d_4 \leadsto \cdots\]

In the following we introduce how to implement manifold matching and metric learning in details.

### 3.2. Manifold Matching

Let \(\mathcal{P}(X)\) be the set of all probability measures on space \(X\). Let \(\mathcal{D}(X)\) be the set of all metrics on space \(X\).

**Definition 3.1** (Pushforward measure). Given a map \(f : X \to Y\) and a probability measure \(\mu \in \mathcal{P}(X)\), the pushforward measure \(f_*\mu \in \mathcal{P}(Y)\) is defined as: for any measurable set \(A \subseteq Y\),

\[(f_*\mu)(A) := \mu(f^{-1}(A)).\]

**Definition 3.2** (Pullback metric). Given a map \(g : Y \to Z\) and a metric \(d \in \mathcal{D}(Z)\), the pull back metric \(g^*d \in \mathcal{D}(Y)\) is defined as: for any \(y_1, y_2 \in Y\),

\[(g^*d)(y_1, y_2) := d(g(y_1), g(y_2)).\]

Manifold matching in our work refers to finding parameter \(\theta_0\) of a specific generative network \(f_\theta : \mathbb{R}^m \to \mathbb{R}^D\) such that the pushforward \((f_\theta)_*\nu\) of some prior distribution \(\nu\) via \(f_\theta\) is condensed around a manifold \(M \subseteq \mathbb{R}^D\). In our case, the real data manifold \(M \subseteq \mathbb{R}^D\) generally has no explicit expression. In other words, given a point \(a \in \mathbb{R}^D\), there is no way to explicitly tell whether \(a \in M\) or how far away \(a\) is from \(M\). For this reason, we attempt to estimate the shape of \(M\) via a set of sample points from \(M\).

The centroid of a space is an important descriptor of its shape. For a metric measure space, the Fréchet mean (cf. [14, 7, 4]) is a natural generalization of the centroid:

**Definition 3.3** (Fréchet mean). The Fréchet mean set \(\sigma(X)\) of a metric measure space \(X = (X, d, \mu)\) is defined as

\[\arg \min_{x \in X} \int_X d^2(x, y)d\mu(y).\]

The Fréchet mean roughly informs the center of \(X\), but to reach the goal of manifold matching, we also need a shape descriptor indicating the size of \(X\). Hence we introduce the notion of \(p\)-diameter [34]:

**Definition 3.4** (\(p\)-diameter). For any \(p \geq 1\), the \(p\)-diameter of metric measure space \(X = (X, d, \mu)\) is defined as

\[\text{diam}_p(X) := \left(\int_X \int_X d(x, x')^pd\mu(x)d\mu(x')\right)^{1/p}.\]

The above definitions of Fréchet mean and \(p\)-diameter are for metric measure spaces, but it also applies to any manifold assuming a uniform volume measure on it. Let \(S = \{x_1, x_2, \cdots, x_k\}\) be a sequence of independent identically distributed points sampled from \(\mu\). Let \(\mu_{k} = \frac{1}{k} \sum_{i=1}^{k} \delta_{x_i}\) denote the empirical measure. We can estimate the shape of \((X, d, \mu)\) by the shape of \((S, d|_S, \mu_k)\). In the following we simply denote \(\sigma(S) := \sigma(S, d|_S, \mu_k)\) and \(\text{diam}_p(S) := \text{diam}_p(S, d|_S, \mu_k)\).
When to estimate the p-diameter, if p is relatively large, \( \text{diam}_p(S) \) is very sensitive to the outliers. To ensure a trustworthy estimation, we choose \( p = 2 \) and use \( \text{diam}_2 \) as a size indicator.

Let \( S_R \) be a set of random real data samples of \( M \) and \( S_F \) be a set of random fake data samples of \( \mu = (f_{\theta})_*\nu \). Then the objective function we propose for manifold matching is:

\[
L_{MM} := d(\sigma(S_R), \sigma(S_F)) + \lambda [\text{diam}_2(S_R) - \text{diam}_2(S_F)],
\]

where \( \lambda \) is a weight parameter.

### 3.3. Metric Learning

Shape descriptors for manifold matching greatly rely on a proper choice of metric \( d \). Although in most cases Euclidean metric \( d_E \) is easy to access, it may not be an intrinsic choice and barely reveals the actual shape of a data set. The intrinsic metric on a Riemannian manifold \( M \) is specified by geodesic distance. Specifically, the geodesic distance between two points on the manifold equals the length of shortest path on \( M \) which connects them. From this point of view, a better choice of metric on the ambient space \( \mathbb{R}^D \supseteq M \) should make the shortest path connecting \( a, b \in M \) stay as close as possible to \( M \), or in other words, make \( M \) as “straight” as possible. Here we apply Triplet metric learning to learn a proper metric on \( \mathbb{R}^D \).

**Definition 3.5.** Given a triple \((x_a, x_p, x_n)\) with \( x_a, x_p \in M \) and \( x_n \notin M \), the Triplet loss is defined as

\[
L_{tri} := \max \{0, d^2(x_a, x_p) - d^2(x_a, x_n) + \alpha\}.
\]

Here \( d = (g_{\theta})^*d_E \) and \( \alpha \) is a margin parameter. People usually call \( x_a \) an anchor sample, \( x_p \) a positive sample and \( x_n \) a negative sample. By minimizing \( L_{tri} \), we attempt to pull back the positive sample to anchor and push out the negative sample, only when \( d(x_a, x_p) \) is relatively larger than \( d(x_a, x_n) \). Fig. 3 illustrates how this would “straighten” the manifold.

Among numerous methods for metric learning, in our implementation we choose one recent approach [37] which adapted Triplet loss by adding a direction regularizer to make the metric learned towards right direction. Hence in our paper the Triplet loss becomes:

\[
L_{apn} = \max \{0, d^2(x_a, x_p) - d^2(x_a, x_n) + \alpha - \gamma \cos(g_{\theta}(x_n) - g_{\theta}(x_a), g_{\theta}(x_p) - g_{\theta}(x_a))\},
\]

where \( \gamma \) is the direction guidance parameter which controls the magnitude of regularization applied to the original Triplet loss \( L_{tri} \). In practice one can also employ other methods to learn proper distance metrics.

### 3.4. Objective Functions

In the metric learning community, the metric generator \( g_{\theta} \) is usually viewed as a metric embedding. From this point of view we have (see supplement for the proof):

**Proposition 3.6.** Given two measure \( \mu_1 \) and \( \mu_2 \) on the same metric space \((X, d)\), where \( d = g_{\theta}^*d_E \). Then \( d(\sigma(X, d, \mu_1), \sigma(X, d, \mu_2)) = d_E(g_{\theta}\mu_1, g_{\theta}\mu_2) \).

Let \( \|\cdot\| \) denote \( L^2 \) norm and \( d = (g_{\theta})^*d_E \), then we have \( d(x, x') = \|g_{\theta}(x) - g_{\theta}(x')\| \) for \( x, x' \in \mathbb{R}^D \). By above proposition, the explicit formula of terms in our objective functions (1) are as follows:

\[
d(\sigma(S_R), \sigma(S_F)) = \|g_{\theta}(S_R) - g_{\theta}(S_F)\|,
\]

\[
diam_2(S) = \frac{1}{\text{card}(S)} \left( \sum_{x, x' \in S} \|g_{\theta}(x) - g_{\theta}(x')\|^2 \right)^{1/2}.
\]

For unconditional generation task, we take Eqn (1) as manifold matching objective, and Eqn (2) as metric learning objective. During training we minimize both (1) and (2). We display our implementation pipeline for unconditional image generation task in Fig. 4 and summarize the training procedure in Algorithm 1. The convergence of training can be addressed using results from [17]. Particularly, the setting in [17] not only applies to min-max GANs, but is also valid for more general GANs where the discriminator’s objective is not necessarily related to the generator’s objective.

In our framework, using Adam optimizer with different decays for the two networks fits this setting.

As for super-resolution task, one can utilize information obtained from LR-HR pairs for more effective matching, thus we include an additional pair matching loss in this case:

\[
L_{pair} = \|g_{\theta}(x_R) - g_{\theta}(x_F)\|,
\]

where \( x_F \) is super-resolved image from LR \( x_L \), \( x_L \) is down-sampled image from HR \( x_R \). We use \( L^1 \) norm for pixel-wise image loss, where \( L_{img} = \|x_R - x_F\|_1 \). Together, in SISR task our total data generator loss becomes

\[
L_{gen} = L_{img} + \lambda_2 L_{pair} + \lambda_3 L_{MM}.
\]
We conduct experiments on two tasks: unconditional image generation and single image super-resolution. To better understand the process of manifold matching, in both tasks we track various distances in batches during training. We represent distance between centroids of two sets as $d_c$, distance between 2-diameters of two sets as $d_g$, and distance between paired SR-HR samples in super-resolution task as $d_p$, respectively. We also use Hausdorff distance as a measurement of distance between the two sets. Hausdorff distance is defined as $d_H(A,B) = \max \{ \sup_{a \in A} \inf_{b \in B} d(a,b), \sup_{b \in B} \inf_{a \in A} d(a,b) \}$, where $A$ and $B$ are two non-empty subsets of a metric space $(M, d)$. Here we adopt Euclidean metric to calculate Hausdorff distance between two sets of Triplet network embedding, which is equivalent to measure Hausdorff distance between two sets of corresponding images in the image space under learned metric. All experiments are implemented under PyTorch framework using a Tesla V100 GPU.

4. Experiments

We use the matching criteria in Eqn 1 to validate the feasibility of the proposed framework. Note that no paired information or GAN loss is used for the task.

**Implementation Details:** We employed a ResNet data generator and a deep convolutional net metric generator with $\gamma = 0.01$, $\lambda = 1$, dimension of input latent vector $m = 128$, and output embedding $n = 10$ as our default setting. Adam optimizer with learning rate $1e-4$, $\beta_1 = 0$ and $\beta_2 = 0.9$ for data generator, and $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for metric generator was used during training. Details of network architectures are provided in supplementary material.

**Dataset and Evaluation Metrics:** We implemented our method on CelebA [30] and LSUN bedroom [52] datasets. For training we used around 200K images in CelebA and 3M images in LSUN. All images were center-cropped and resized to $32 \times 32$ or $64 \times 64$. For each dataset we randomly generated 50K samples and used Fréchet Inception Distance (FID) [17] for quantitative evaluation. Smaller FID indi-
cates better result.

**Effect of Metric Learning on Distorting Real Data Manifold:** During training we randomly choose 1024 real samples and detect its shape by looking at the eigenvalues of the corresponding (normalized) distance matrix. Particularly, we plot the first 10 largest eigenvalues which correspond to the size of the first 10 principle components of real samples. As shown in Fig. 5, the shape of the samples becomes more uniform with training going. This is an empirical evidence that \( g_w \) distorts real data manifold to be uniformly curved. In this situation, matching centers and diameters should be enough for manifold matching.

**Effects of Matching Different Geometric Descriptors:**

We study the effect of matching different geometric descriptors. Examples of generated \( 32 \times 32 \) samples on CelebA using different matching criteria are shown in Fig. 6. (i) Centroid matching learns some common shallow patterns from real data set; (ii) Matching 2-diameters could capture more complicated intrinsic structures of the data manifold, while misalignment between the two sets can result in low-quality samples (e.g. image on the right side in the second last row); (iii) Combining the two descriptors together leads to more stable sample quality. We also visualize these manifold matching status for illustration purpose. We project output of \( g_w \) to 2-dim plane using UMAP [33], and display the projected points in Fig. 6 (a)(b)(c). (a) and (b) intuitively show two typical matching status if one only matches centroids or 2-diameters between the two sets, respectively.

![Figure 6. Randomly generated 32 × 32 images on CelebA with different manifold matching criteria during training.](image)

Figure 6. Randomly generated \( 32 \times 32 \) images on CelebA with different manifold matching criteria during training. (i) centroid only; (ii) 2-diameter only; (iii) both centroid and 2-diameter. (a)(b)(c) are corresponding UMAP plots of real (green) and generated (orange) samples with different manifold matching criteria. (a) centroid only; (b) 2-diameter only; (c) both.

<table>
<thead>
<tr>
<th>Batch size</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective training</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Time(s) / epoch</td>
<td>178</td>
<td>155</td>
<td>139</td>
<td>127</td>
<td>127</td>
<td>-</td>
</tr>
</tbody>
</table>

**Effects of Batch Size:**

We further study the influence of batch size on training time and stability. Table 2 reports average training time per epoch on resized \( 32 \times 32 \) CelebA images. One can see batch size does not have a significant influence on average training time when matching 2-diameters. In addition, we observe stable and efficient training sessions with different batch sizes. For batch size as large as 1024, we did not observe satisfying sample quality in reasonable training time.

**Quantitative Results:**

We track \( d_c, d_g \) and \( d_H \) during training and display them in Fig. 7. With training going forward, the distances keep decreasing and gradually converge. The observation aligns with our manifold matching assumption even with no labelled information involved. For quantitative evaluation we present FID scores in Table 4. Here we also display results from some classic GAN frameworks using the same generator architecture. As shown in the table our method obtains competitive results. Examples of randomly generated samples are displayed in Fig. 8.

![Figure 7. Distances versus training epochs in unconditional image generation task.](image)

Figure 7. Distances versus training epochs in unconditional image generation task.

**Effects of Batch Size:**

Table 2. Comparisons of results trained on different batch sizes.

4.2. Single Image Super-Resolution

In a typical perception-based SR model [23, 46], the generator loss function is usually made up by three compo-
Table 3. Evaluation scores of different training settings in ×4 SISR task. Demonstration of settings is displayed in Table 6. For each pair of settings with the same generator backbone, the one with better performance is highlighted.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Set5</th>
<th>Set14</th>
<th>BSD100</th>
<th>Urban100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>SSIM</td>
<td>LPIPS</td>
<td>NIQE</td>
</tr>
<tr>
<td>ResNet-GAN</td>
<td>29.03</td>
<td>0.8468</td>
<td>0.1885</td>
<td>5.2143</td>
</tr>
<tr>
<td>ResNet-MvM</td>
<td>29.76</td>
<td>0.8606</td>
<td>0.1941</td>
<td>6.1478</td>
</tr>
<tr>
<td>RDN-GAN</td>
<td>30.06</td>
<td>0.8658</td>
<td>0.1850</td>
<td>6.1283</td>
</tr>
<tr>
<td>RDN-MvM</td>
<td>30.06</td>
<td>0.8658</td>
<td>0.1850</td>
<td>6.1283</td>
</tr>
<tr>
<td>NSRNet-GAN</td>
<td>29.46</td>
<td>0.8544</td>
<td>0.1852</td>
<td>5.7818</td>
</tr>
<tr>
<td>NSRNet-MvM</td>
<td>29.79</td>
<td>0.8641</td>
<td>0.1845</td>
<td>6.0846</td>
</tr>
</tbody>
</table>

Table 4. FID evaluation on 64 × 64 experiments with a ResNet generator.

<table>
<thead>
<tr>
<th>Method</th>
<th>CelebA (FID)</th>
<th>LSUN bedroom (FID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WGAN [3]</td>
<td>37.1 (1.9)</td>
<td>73.3 (2.5)</td>
</tr>
<tr>
<td>WGAN-GP [15]</td>
<td>18.0 (0.7)</td>
<td>26.9 (1.1)</td>
</tr>
<tr>
<td>SNGAN [36]</td>
<td>21.7 (1.5)</td>
<td>31.3 (2.1)</td>
</tr>
<tr>
<td>SWGAN [50]</td>
<td>13.2 (0.7)</td>
<td>14.9 (1.0)</td>
</tr>
<tr>
<td>MvM</td>
<td>11.1 (0.1)</td>
<td>13.7 (0.3)</td>
</tr>
</tbody>
</table>

Figure 9. Generated ×4 samples with GAN loss and MvM loss using the same generator backbone. (Zoom in for better view.)

Figure 10. Various distances versus training epochs in super-resolution experiment with our method. Distances are displayed every 40 epochs (1000 iterations).

(A) MvM As A Substitute of GAN Loss: We display comparison of evaluation results between GAN and MvM in...
Table 5. Evaluation scores of different training settings in ×4 SISR task. Demonstration of settings is displayed in Table 6. The best performance is highlighted in red and second best in blue.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Set5 PSNR</th>
<th>Set5 SSIM</th>
<th>Set5 LPIPS</th>
<th>Set5 NIQE</th>
<th>Set14 PSNR</th>
<th>Set14 SSIM</th>
<th>Set14 LPIPS</th>
<th>Set14 NIQE</th>
<th>BSD100 PSNR</th>
<th>BSD100 SSIM</th>
<th>BSD100 LPIPS</th>
<th>BSD100 NIQE</th>
<th>Urban100 PSNR</th>
<th>Urban100 SSIM</th>
<th>Urban100 LPIPS</th>
<th>Urban100 NIQE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>28.42</td>
<td>0.8104</td>
<td>0.2977</td>
<td>7.3647</td>
<td>26.00</td>
<td>0.7027</td>
<td>0.3757</td>
<td>6.3393</td>
<td>25.96</td>
<td>0.6675</td>
<td>0.3577</td>
<td>6.2967</td>
<td>23.14</td>
<td>0.6577</td>
<td>0.3182</td>
<td>6.4299</td>
</tr>
<tr>
<td>B</td>
<td>29.66</td>
<td>0.8586</td>
<td>0.1401</td>
<td>6.6582</td>
<td>26.08</td>
<td>0.7542</td>
<td>0.2265</td>
<td>5.3638</td>
<td>26.05</td>
<td>0.7145</td>
<td>0.3257</td>
<td>5.5376</td>
<td>24.00</td>
<td>0.7464</td>
<td>0.2156</td>
<td>4.7883</td>
</tr>
<tr>
<td>C</td>
<td>29.30</td>
<td>0.8445</td>
<td>0.1324</td>
<td>6.6590</td>
<td>25.81</td>
<td>0.7394</td>
<td>0.2132</td>
<td>5.5318</td>
<td>25.81</td>
<td>0.7010</td>
<td>0.3006</td>
<td>5.8399</td>
<td>23.83</td>
<td>0.7346</td>
<td>0.2047</td>
<td>4.6377</td>
</tr>
<tr>
<td>D</td>
<td>29.75</td>
<td>0.8593</td>
<td>0.1357</td>
<td>6.5475</td>
<td>26.11</td>
<td>0.7542</td>
<td>0.2193</td>
<td>5.3559</td>
<td>26.06</td>
<td>0.7145</td>
<td>0.3160</td>
<td>5.6108</td>
<td>24.15</td>
<td>0.7506</td>
<td>0.2028</td>
<td>4.7447</td>
</tr>
<tr>
<td>E</td>
<td>29.63</td>
<td>0.8547</td>
<td>0.1227</td>
<td>6.2838</td>
<td>26.07</td>
<td>0.7512</td>
<td>0.2051</td>
<td>5.0657</td>
<td>26.05</td>
<td>0.7119</td>
<td>0.2957</td>
<td>5.2257</td>
<td>24.12</td>
<td>0.7484</td>
<td>0.1862</td>
<td>4.2814</td>
</tr>
<tr>
<td>F</td>
<td>29.70</td>
<td>0.8603</td>
<td>0.1409</td>
<td>6.2673</td>
<td>26.06</td>
<td>0.7556</td>
<td>0.2239</td>
<td>5.3777</td>
<td>26.05</td>
<td>0.7146</td>
<td>0.3218</td>
<td>5.6095</td>
<td>23.98</td>
<td>0.7477</td>
<td>0.2126</td>
<td>4.7880</td>
</tr>
<tr>
<td>G</td>
<td>29.87</td>
<td>0.8615</td>
<td>0.1281</td>
<td>5.8904</td>
<td>26.17</td>
<td>0.7564</td>
<td>0.2048</td>
<td>4.9612</td>
<td>26.10</td>
<td>0.7134</td>
<td>0.2930</td>
<td>4.9864</td>
<td>24.25</td>
<td>0.7567</td>
<td>0.1866</td>
<td>4.3432</td>
</tr>
</tbody>
</table>

Table 6. Training settings for GAN-based SISR methods with fixed generator architecture and GAN component.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Method</th>
<th>Perceptual component</th>
<th>Pretrained</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Bicubic</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>GAN-ResNet</td>
<td>VGG16-Pooling5</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>SRGAN</td>
<td>VGG19-Pooling2,5</td>
<td>Yes</td>
</tr>
<tr>
<td>D</td>
<td>EnhanceNet</td>
<td>VGG19-Pooling2,5</td>
<td>Yes</td>
</tr>
<tr>
<td>E</td>
<td>ESRGAN</td>
<td>VGG19-Conv5-4</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>NatSR</td>
<td>NMD</td>
<td>Yes</td>
</tr>
<tr>
<td>G(ours)</td>
<td>MvM</td>
<td>MM</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3. Under different backbones, MvM performs better than GAN in similarity-based metrics in all cases. It also obtains better scores in perception-based metrics in majority of the cases. Examples of generated samples using the same generator backbone are displayed in Fig. 9. We see MvM resulted in samples with more natural textures.

(B) MvM As A Substitute of Naturalness Loss: We display a few different setups in Table 6. Final results with both distortion-based and perception-based evaluation scores on benchmark datasets are presented in Table 5, where GAN-ResNet represents SRGAN without VGG component in loss function. With the same common setup, MvM obtained better results in most of the cases for both types of metrics. Examples of generated samples from various settings are shown in Fig. 11. As we see MvM generates samples with more natural textures and less artifacts under the same setup. We notice that although both GAN and MvM utilize adversarial learning for training, the two approaches behave differently in super-resolution task. GAN tends to generate more details but with artifacts, while MvM tends to generate more natural textures in images. The two approaches do not conflict with each other. Instead, one serves as a complement for the other to result in better sample quality.

5. Discussion and Conclusion

In this paper, we have proposed a manifold matching approach for generative modeling, which matches geometric descriptors of real and generated data sets using learned distance metrics. Experiments on two tasks validated its feasibility and effectiveness. Moreover, the proposed framework is robust and flexible in that each network has its own designated objective. Despite that our method has led to some promising results, there is yet much room for improvements. For example, the currently used geometric descriptors may not fully recover the information of the underlying manifolds, thus matching towards other descriptors could potentially benefit the learning process. As to metric learning, in our paper we employ the method in [37] fed with random samples to learn a metric, while in practice one could investigate ways for better approximation of metrics, such as utilizing sampling methods or other metric learning methods. In addition, although empirical evidences and intuitions agree with the current metric measure space setting, further theoretical analysis using optimal transport is worth exploring. In the last few years we have witnessed great success of probability-based generative modeling approaches, and we believe joining geometry with statistics would lead to stronger expression ability for generative models, which is a promising direction for researchers to explore.

6. Acknowledgment

MD and HH would like to thank the four anonymous reviewers especially Reviewer 1 and Reviewer 4 for their valuable comments. HH acknowledges partial support by NSF grant DMS-1854683.
References


