Kernel Mean Matching for Content Addressability of GANs

Wittawat Jitkrittum [*1] **Patsorn Sangkloy** [*2] **Muhammad Waleed Gondal** [1] **Amit Raj** [2]
James Hays [2] **Bernhard Schölkopf** [1]

Abstract

We propose a novel procedure which adds "content-addressability" to any given unconditional implicit model e.g., a generative adversarial network (GAN). The procedure allows users to control the generative process by specifying a set (arbitrary size) of desired examples based on which similar samples are generated from the model. The proposed approach, based on kernel mean matching, is applicable to any generative models which transform latent vectors to samples, and does not require retraining of the model. Experiments on various high-dimensional image generation problems (CelebA-HQ, LSUN bedroom, bridge, tower) show that our approach is able to generate images which are consistent with the input set, while retaining the image quality of the original model. To our knowledge, this is the first work that attempts to construct, *at test time*, a content-addressable generative model from a trained marginal model.

1. Introduction

Modern high-dimensional, complex generative models take the form of implicit models: these are models which generate a sample by transforming a latent random vector (also known as code) with a function given by a deep neural network (Mohamed and Lakshminarayanan, 2016; Nowozin et al., 2016). A state-of-the-art class of implicit models has been the generative adversarial networks (GANs, Goodfellow et al., 2014), which have been shown to learn to generate high-resolution realistic natural images (Arjovsky et al., 2017; Karras et al., 2017; Mescheder et al., 2018).

[*]Equal contribution [1]Empirical Inference Department, Max Planck Institute for Intelligent Systems, Germany [2]School of Interactive Computing, Georgia Institute of Technology, USA. Correspondence to: Wittawat Jitkrittum <wittawat@tuebingen.mpg.de>, Patsorn Sangkloy <patsorn.sangkloy@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

In image generative modeling, it is desirable to have *control* over how images are generated. An issue with typical implicit models is that it is unclear how the latent code can be manipulated to generate images which satisfy a given description (e.g., outdoor scene containing a red bridge). In general, without explicitly imposing structure into the latent space, there is no obvious relationship between the latent code and the generated images. This problem was the basis for InfoGAN (Chen et al., 2016) which augments the GAN loss function with an information-theoretic regularization term to encourage disentangled representation of the latent code. Different forms of explicit control signals for image generation were considered in the literature, including class labels of images (Mirza and Osindero, 2014; Nguyen et al., 2017), text description (Reed et al., 2016; Xu et al., 2018), visual attributes (Yan et al., 2016), and context variables (Ren et al., 2016).

Among others, a less explored form of control signal has been visual content (an image, or a set of images in general). Given an input set of images and a similarity measure, content-based image generation seeks to generate a diverse set of images that are perceptually or semantically similar to the given input set. Each output image is expected to contain features of some or all input images. To take a concrete example, the input set might contain two face images: one with [light hair, dark skin], and another with [dark hair, light skin]. If two faces are considered similar when at least one of these attributes match, then valid generated faces might be with [light hair, light skin] or [dark hair, dark skin]. To solve this task, it is crucial to be able to construct set-level representation by aggregating features in each input image. We note that we distinguish this problem from image-to-image translation where certain aspects of one input image are changed in a controlled manner while keeping other aspects the same (Zhu et al., 2017). Instances of the image-to-image translation problem include image colorization (Isola et al., 2017), artistic style transfer (Gatys et al., 2016; Huang and Belongie, 2017; Zhu et al., 2017), sketch-to-image conversion (Sangkloy et al., 2017), and texture completion (Xian et al., 2018).

Only a small number of existing works address content-based image generation, and the focus has been primarily on the case when the input set contains only one image i.e.,

no need to aggregate features in different images. In this case, one may consider the variational autoencoder (VAE, Kingma and Welling, 2014; Rezende et al., 2014) and related formulations (Makhzani et al., 2016; Tolstikhin et al., 2018) which encode the input image to a latent space and stochastically decode the code to generate images. Two issues remain to be addressed. Firstly, reconstructing the input image is part of the formulation in these approaches; thus, the output images have low variability, and appear to almost reproduce the input image. Secondly, and more importantly, it is unclear how to represent aggregated, set-level information of the input set when it contains more than one image. It is tempting to use recurrent neural networks (RNNs) to model sets since they allow dependency among items in the collection. However, an RNN explicitly imposes an order, and it is not clear how it can be used to model exchangeable data such as sets (Korshunova et al., 2018, p. 2). This observation was the motivation of BRUNO (Korshunova et al., 2018), a latent variable model defining an exchangeable joint distribution, meaning that it is invariant under permutation of observations and is suitable for modeling input images as a set. Content-based image generation can be realized with the posterior predictive distribution, conditioned on the input set. While promising, there is opportunity for improvement. Since BRUNO models images as a whole, there is no control over what aspects of the conditioned input images should be captured when performing content-based generation. That is, image features captured by the model are largely determined by the training data, and model architecture. Once the model is trained, it is highly challenging to change those features at run time, without retraining. This statement holds true for many approaches which construct a purpose-built (conditional) generative model. The issue is one of the key challenges we tackle in the present work.

In this work, we take a different approach and address content-based image generation by leveraging available pre-trained implicit generative models without constructing a bespoke model. We propose a general procedure which enables any unconditional implicit model to perform content-based generation. Briefly, the procedure is based on kernel mean matching (Chen et al., 2010; Gretton et al., 2012a): it generates images from the model so that their mean feature, in a reproducing kernel Hilbert space (RKHS), matches that of the input images. Importantly, the procedure does not require any training or retraining of the implicit model. Aspects of the input images to capture can be specified at test time with an image feature extractor of choice, and the allowed number of input images is arbitrary. To our knowledge, this is the first work that proposes a generic scheme to construct, *at test time*, a content-addressable generative model from a trained implicit model. Experiments on various problems (CelebA-HQ, LSUN bedroom, bridge, tower) show that our approach is able to combine features from multiple input images and generate consistent, realistic images, while retaining the image quality of the original model.

2. Background

We first briefly review the kernel mean embedding (Smola et al., 2007), and the kernel mean matching problem (Chen et al., 2010; Gretton et al., 2012a). Our proposed method (Section 3) will be based on the kernel mean matching.

Kernel Mean Embedding Let $\mathcal{X} \subset \mathbb{R}^d$ be the data domain (e.g., domain of images with d pixels), and $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric, positive definite kernel associated with Hilbert space $\mathcal{H}$. It is known that there exists a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathcal{H}}$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the inner product on $\mathcal{H}$. The space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), and k is its reproducing kernel (Berlinet and Thomas-Agnan, 2004). We interchangeably write $\phi(\boldsymbol{x})$ and $K(\boldsymbol{x}, \cdot)$, and write $\langle \cdot, \cdot \rangle$ for $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Let P be a Borel probability distribution defined on $\mathcal{X}$. The kernel mean embedding of P is defined as $\mu_P := \mathbb{E}_{\boldsymbol{x} \sim P}[\phi(\boldsymbol{x})]$ which is an element in $\mathcal{H}$ (assumed to exist under some regularity conditions; see Lemma 3 in Gretton et al. 2012a, for instance). One can see μ_P as a representation of the distribution P in the form of a single point in $\mathcal{H}$.

As an illustrative example, consider $\mathcal{X} = \mathbb{R}$, $\phi(x) := (x, x^2)^\top$, and $\mathcal{H} = \mathbb{R}^2$ so that the kernel $K(x, y) = \phi^\top(x)\phi(y) = xy + x^2y^2$. Given a distribution P, in this case, its mean embedding is $\mu_P = \mathbb{E}_{x \sim P}(x, x^2)^\top = \left(\mathbb{E}_{x \sim P}[x], \mathbb{E}_{x \sim P}[x^2]\right)^\top \in \mathcal{H}$, which is a two-dimensional vector consisting of the first two moments of P. One may also think of μ_p as a vector of summary statistics of P.

In general, if $\mathcal{H}$ is infinite-dimensional, then the mean embedding μ_P is an infinitely long vector. An important question is: when is μ_P unique to only P? Equivalently, given two distributions P, Q, under what conditions does $P \neq Q$ imply $\mu_P \neq \mu_Q$? The answer to this question will be crucial in the next section when we define a distance between two distributions based on kernel mean embedding. It turns out that the conditions are on the kernel K. If K is characteristic (Fukumizu et al., 2008; Sriperumbudur et al., 2011), then the kernel mean map $P \mapsto \mathbb{E}_{\boldsymbol{x} \sim P} K(\boldsymbol{x}, \cdot)$ is injective, meaning that mean embeddings uniquely identify distributions. A Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\sigma^2}\right)$ for some bandwidth $\sigma > 0$, and an inverse multi-quadric (IMQ) kernel $K(\boldsymbol{x}, \boldsymbol{y}) = (c^2 + \|\boldsymbol{x}-\boldsymbol{y}\|_2^2)^{-1/2}$ for some $c > 0$ are characteristic (Sriperumbudur et al., 2011; Gorham and Mackey, 2017). The kernel $K(x, y) = xy + x^2y^2$ considered previously is not characteristic, since there are distributions which share the first two moments and differ in

higher-order moments; these different distributions would lead to the same mean embedding under this kernel.

Maximum Mean Discrepancy (MMD) The kernel mean embedding technique allows us to measure distance in the Hilbert space $\mathcal{H}$ between two distributions. Given two distributions P and Q, it is known that if K is characteristic, then $\|\mu_P - \mu_Q\|_{\mathcal{H}} = 0$ if and only $P = Q$ (Gretton et al., 2012a). This distance is known as maximum mean discrepancy (MMD) and we write $\mathrm{MMD}(P, Q) = \|\mu_P - \mu_Q\|_{\mathcal{H}}$. In our proposed approach (Section 3), we will use MMD to measure the distance between the input images and the generated images. Note that if the kernel K is not characteristic, then MMD is only a pseudometric; in particular, $\|\mu_P - \mu_Q\|_{\mathcal{H}} = 0$ does not imply that $P = Q$. For brevity, we shorten $\mathbb{E}_{\boldsymbol{x}\sim P}$ to $\mathbb{E}_{\boldsymbol{x}}$, and $\mathbb{E}_{\boldsymbol{y}\sim Q}$ to $\mathbb{E}_{\boldsymbol{y}}$. It can be shown that $\mathrm{MMD}^2(P, Q) = \mathrm{MMD}^2$ can be written as

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'} K(\boldsymbol{x}, \boldsymbol{x}') + \mathbb{E}_{\boldsymbol{y},\boldsymbol{y}'} K(\boldsymbol{y}, \boldsymbol{y}') - 2\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}} K(\boldsymbol{x}, \boldsymbol{y}),$$

where $\boldsymbol{x}, \boldsymbol{x}'$ are independently drawn from P, and similarly for $\boldsymbol{y}, \boldsymbol{y}'$ (see Lemma 6 of Gretton et al. 2012a). Given samples $X_m := \{\boldsymbol{x}_i\}_{i=1}^m \overset{i.i.d.}{\sim} P$ and $Y_n := \{\boldsymbol{y}_i\}_{i=1}^n \overset{i.i.d.}{\sim} Q$, a plug-in estimator of MMD^2 is given by

$$\frac{1}{m^2}\sum_{i,j=1}^{m} K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} K(\boldsymbol{y}_i, \boldsymbol{y}_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{y}_j), \tag{1}$$

which does not require the knowledge of the underlying (possibly infinite-dimensional) feature map ϕ.

The MMD estimator in (1) is equivalent to $\|\hat{\mu}_P - \hat{\mu}_Q\|_{\mathcal{H}}^2$, where $\hat{\mu}_P := \frac{1}{m}\sum_{i=1}^m K(\boldsymbol{x}_i, \cdot)$ and $\hat{\mu}_Q := \frac{1}{n}\sum_{i=1}^n K(\boldsymbol{y}_i, \cdot)$ are empirically estimated mean embeddings. In general, the empirical mean embeddings may take the form of a weighted average i.e., $\hat{\mu}_{P,\boldsymbol{w}} = \sum_{i=1}^m w_i K(\boldsymbol{x}_i, \cdot)$ for some weights $\boldsymbol{w} := (w_1, \ldots, w_m)$ which are specified, or learned (Fukumizu et al., 2013; Huang et al., 2007; Song et al., 2008). We write

$$\widehat{\mathrm{MMD}}^2(X_m, Y_n, \boldsymbol{w}) := \|\hat{\mu}_{P,\boldsymbol{w}} - \hat{\mu}_Q\|_{\mathcal{H}}^2$$
$$= \sum_{i,j=1}^{m} w_i w_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} K(\boldsymbol{y}_i, \boldsymbol{y}_j) - \frac{2}{n}\sum_{i=1}^{m} w_i \sum_{j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{y}_j). \tag{2}$$

The weighted form in (2) will be useful in our task for controlling the amount of contribution from each input image to the generated images. Notice that if $w_i = \frac{1}{m}$ for all $i = 1, \ldots, m$, then (1) is recovered. In this work, the weight vector $\boldsymbol{w}$ is manually specified.

Kernel Mean Matching Given an input (weighted) mean embedding $\hat{\mu}_{P,\boldsymbol{w}} = \sum_{i=1}^m w_i K(\boldsymbol{x}_i, \cdot)$, kernel mean matching (Chen et al., 2010; Bach et al., 2012; Lacoste-Julien et al., 2015; Chen et al., 2018) aims to find a set of points $Y_n := \{\boldsymbol{y}_i\}_{i=1}^n \subset \mathcal{X}$ such that the mean embedding estimated from Y_n is as close as possible to $\hat{\mu}_{P,\boldsymbol{w}}$. Mathematically,

$$Y_n^* = \arg\min_{\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n\}} \widehat{\mathrm{MMD}}^2(X_m, Y_n, \boldsymbol{w}). \tag{3}$$

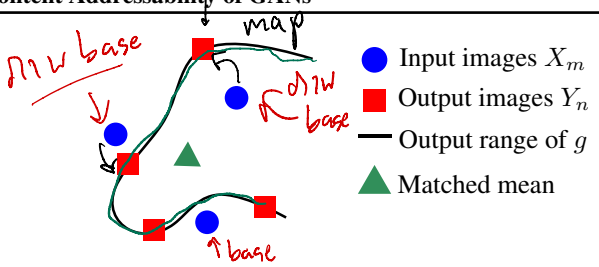


Figure 1: An illustration of the proposed method for content-addressable image generation (see (5)). Given input images (blue circles), our approach generates images (red squares) from the model g so as to match the mean feature (green triangle) of the input images represented in a reproducing kernel Hilbert space. The input images do not need to be in the range of g.

By interpreting K as a similarity function on images, one can see the third term in (2) as capturing similarity between the input and the output points. The second term encourages the output points $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ to be diverse. This formulation thus yields diverse output points which are similar to the input samples (in the sense that the two underlying distributions are close). Note that the first term is independent of the output points.

MMD is theoretically-grounded and has several practical advantages: it defines a differentiable (given that K is differentiable) distance on a large class of distributions; and its estimator can be easily computed on the basis of two sets of samples. Unlike many existing divergence measures, MMD estimator in (2) does not require estimates of the underlying probability densities. These properties make it a natural candidate as a test statistic for nonparametric two-sample testing (Gretton et al., 2012a;b) i.e., determining whether two independent collections of samples are from the same distribution.

3. Content-Addressable Image Generation

In this section, we detail our proposed procedure that enables any implicit generative models to perform content-based image generation. Let $\boldsymbol{z}$ be a latent random vector (code) of an implicit generative model g such that $\boldsymbol{y} = g(\boldsymbol{z})$ is a sample drawn from the model, where $\boldsymbol{z} \sim p_z$ and p_z is a fixed prior distribution defined on a domain $\mathcal{Z}$. Given a trained model $g\colon \boldsymbol{z} \mapsto \boldsymbol{x}$, a kernel K (discussed in Section 2), and a set of input points $X_m = \{\boldsymbol{x}_i\}_{i=1}^m$ (content) represented as a weighted mean embedding $\hat{\mu}_{P,\boldsymbol{w}}$, we propose to generate new samples Y_n, conditioned on X_m, by

charactoric ~~ti~~ c kernel

$$\begin{bmatrix} x^2 \\ x^3 \\ \vdots \\ x \end{bmatrix} \Rightarrow$$

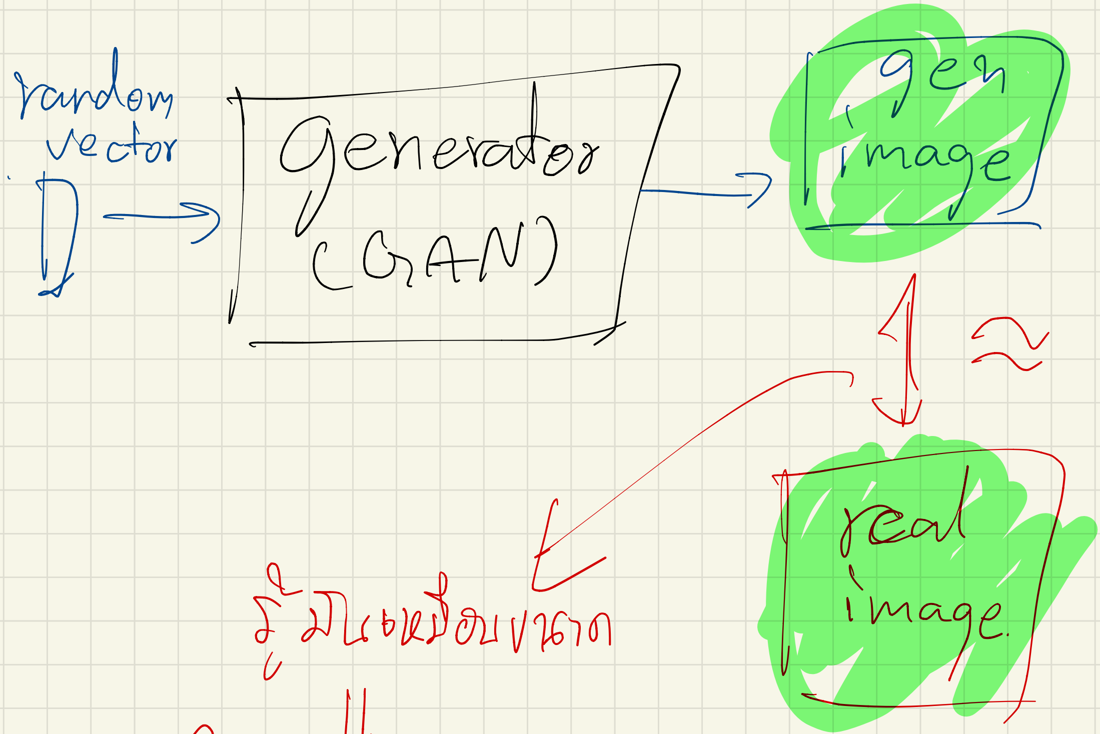
mean → สามารถ → ~~caption~~ ~~ได้~~ information ได้สมบูรณ์

Guss - kernel (เหมือนใน ch - kernel)

↘ ค่าแรก distance = 0

⇓

~~ทั้งสอง~~ distribution เหมือนกัน



รู้มันเหมือนขนาด

⇓

ใช้ map kernel

⇓

วัด distance.

solving the following optimization problem:

$$\min_{\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n\}} \widehat{\mathrm{MMD}}^2(X_m, Y_n, \boldsymbol{w}) \text{ s.t. } \forall i, \boldsymbol{y}_i \in \mathcal{R}(g), \quad (4)$$

where $\mathcal{R}(\cdot)$ denotes range of a function. This formulation is a constrained version of (3) where the output images are required to be on the output manifold of g. The output images are thus guaranteed to be generated by g, leveraging the information about natural images contained in the trained model. Without the constraint, the search space would be the full pixel space, and the optimized images would be less likely natural. We note that (4) is equivalent to the following more convenient form:

$$\min_{\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}} \widehat{\mathrm{MMD}}^2(X_m, \{g(\boldsymbol{z}_i)\}_{i=1}^n, \boldsymbol{w}) \text{ s.t. } \forall i, \boldsymbol{z}_i \in \mathcal{Z}, \quad (5)$$

where we use the fact that if $\boldsymbol{y} \in \mathcal{R}(g)$, then there exists a latent vector $\boldsymbol{z}$ such that $\boldsymbol{y} = g(\boldsymbol{z})$. This equivalence allows us to optimize the latent vectors $Z_n := (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$ instead of pixels. In practice, the latent space is typically of much lower dimension compared to the image space. Optimizing the latent codes directly thus provides a more tractable way to find relevant output images given the input. An illustration of our approach is presented in Figure 1. Since the MMD estimator is differentiable, any gradient-based optimization algorithms can be used to solve (5).

Kernel Design Our approach relies on a positive definite kernel K to specify similarity between two images. It characterizes features of the input images that determine the output images. We propose using a kernel K which takes the form:

$$K(\boldsymbol{x}, \boldsymbol{y}) := k(E(\boldsymbol{x}), E(\boldsymbol{y})), \quad (6)$$

where $E\colon \mathcal{X} \to \mathbb{R}^{d_e}$ is a pre-trained image feature extractor e.g., VGG net (Simonyan and Zisserman, 2014), and k is a simple, nonlinear kernel (e.g., an IMQ kernel) on top of the extracted features. Combining structural properties encoded in the deep network E and nonlinear features implicitly defined by k has shown great successes in many learning tasks (Wilson et al., 2016; van der Wilk et al., 2017; Wenliang et al., 2018). We note that if $k(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^\top \boldsymbol{b}$ (i.e., linear kernel), then the objective in (5) becomes $\left\| \sum_{i=1}^m w_i E(\boldsymbol{x}_i) - \frac{1}{n} \sum_{j=1}^n E(g(\boldsymbol{z}_j)) \right\|^2_{\mathbb{R}^{d_e}}$ which matches the first moment of the features extracted from the input set and the generated images. In experiments, we argue that using a nonlinear kernel k improves the representation of the input images. We observe that an IMQ kernel as k (see Section 2) yields realistic output images relevant to the input. With the kernel taking the form in (6), the minimization objective becomes

$$\frac{1}{n^2} \sum_{i,j=1}^n k(E(g(\boldsymbol{z}_i)), E(g(\boldsymbol{z}_j))) - \frac{2}{n} \sum_{i=1}^m w_i \sum_{j=1}^n k(E(\boldsymbol{x}_i), E(g(\boldsymbol{z}_j))),$$

where the first term (constant) in (2) is dropped.

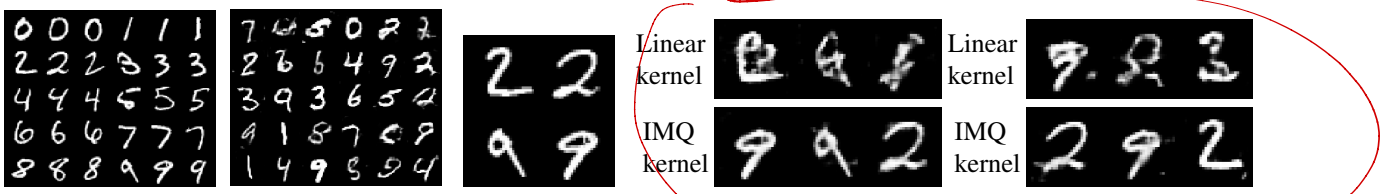
Optimization To solve (5), we use Adam (Kingma and Ba, 2015) which relies on the gradient $\nabla_{Z_n} \widehat{\mathrm{MMD}}^2(X_m, \{g(\boldsymbol{z}_i)\}_{i=1}^n, \boldsymbol{w})$ to update Z_n and find a local minimum. After each update, we clamp the values of Z_n so that the absolute value of each value is no larger than $c > 0$ chosen appropriately depending on the prior p_z. This is equivalent to projecting onto an ℓ_∞-ball with radius c centered at the origin. For instance, if the prior $p_z = \mathrm{Uniform}([-1, 1]^{d_z})$, then we set $c = 1$. If $p_z = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, then we set $c := 3.5$. This value is motivated by the fact that more than 99.9% of the probability mass of the standard normal is in the interval $(-3.5, 3.5)$. Clamping all coordinates of Z_n in this way helps prevent Z_n from going outside the region where g can decode to get natural images.

4. Related Works

Our proposed method can be seen from different angles: conditional image generation, sets as inputs, latent space optimization, and mean matching with MMD. Here, we briefly describe works related to each of these aspects.

Conditional Image Generation Unconditional generative adversarial networks proposed by Goodfellow et al. (2014) have been extended for conditional image generation in numerous contexts such as image to image translation (Isola et al., 2017; Zhu et al., 2017; Liu et al., 2017; Huang et al., 2018), image in painting (Pathak et al., 2016; Iizuka et al., 2017), class based image generation (Mirza and Osindero, 2014), and text based image generation (Reed et al., 2016). Plug and play networks (Nguyen et al., 2017) perform image generation through iterative sampling of latent vectors conditioned on a single class or caption signal. Image-to-image translation networks rely on paired or unpaired training data to translate images between domains. In all of these cases, the generation process is conditioned on a single input of a particular modality (i.e., class, text, or image). StyleGAN (Karras et al., 2019) is a contemporary work that modifies the generator architecture to explicitly condition on two image sources (style and content). Our work proposes a general framework that allows conditioning on sets of images, rather than fixed number of inputs, without additional changes to the generator architecture.

Sets as Input Previous works have demonstrated using sets as inputs for classification or segmentation tasks. PointNet and PointNet++ models (Qi et al., 2017a;b) take a set of points as input and use max pooling to aggregate the features from the point set. Alternatively, another way to incorporate pooling of features from a set is through RNNs as demonstrated for attribute prediction by Wang et al. (2016). However, the final pooled features obtained from an RNN are sensitive to the ordering of the input images. This issue was the motivation in Korshunova et al. (2018) who ex-



(a) Real images (b) Samples from DCGAN (c) Input X_m (d) Output Y_n. No extractor. (e) Output Y_n. CNN extractor.

Figure 2: Content-based image generation on MNIST with a DCGAN model. **(a)**: Real images from the dataset. **(b)**: Unconditional samples from the model. **(c)**: Input set of $m = 4$ images for content-based generation. **(d)**: Output set of $n = 3$ images generated by our proposed approach with the linear kernel k (top), and with the IMQ kernel (bottom). Here, no feature extractor E is used i.e., identity map for E in (6). **(e)**: Same as in (d) with the extractor set to the output of the first two convolutional layers of a convolutional neural network (CNN) classifier (10-digit classification). In both cases, the IMQ kernel gives output images which are more consistent with the input than does the linear kernel.

tended RNNs and proposed a latent variable model defining an exchangeable joint distribution over the input items. Zaheer et al. (2017) (Deep sets) shows that under some conditions, any set function can be written as a function of the sum of transformation of each item in the set. This result coincides with the representation used by the kernel mean embedding (Smola et al., 2007) which forms the basis of our procedure.

Optimizing Latent Space Another line of work controls the image generation process by directly optimizing the latent variables. Brock et al. (2016) use an introspective adversarial network to allow for direct control over generated image by editing in the latent space. Zhu et al. (2016) design a set of editing operations by first projecting the image to a latent space, and editing the image generated from the latent vector. Similarly, Yeh et al. (2017) address the image inpainting task by iteratively sampling latent vectors to find an image in the natural image manifold closest to the input partial image. Xiao et al. (2018) propose a framework to exchange attribute information between two images by exchanging parts of the latent codes. In a similar stride, our proposed objective optimizes for a set of latent vectors whose corresponding images match the mean feature of the input set. Importantly, unlike most previous methods which handle single or pairs of images, our method is capable of finding a set of latent vectors from a arbitrary-sized set of input images, without retraining the specified marginal model.

MMD In the context of generative modeling, MMD was shown to be a promising objective function for training GANs where the kernel is also learned (MMD-GAN, Li et al., 2015; Sutherland et al., 2016; Li et al., 2017; Bińkowski et al., 2018; Wang et al., 2019). MMD has been applied for generating $\{\boldsymbol{y}_i\}_{i=1}^n$ from a mean embedding representation of an empirical distribution defined by $\{\boldsymbol{x}_i\}_{i=1}^m$. This task is known as kernel mean matching, or kernel herding (Chen et al., 2010; Bach et al., 2012; Lacoste-Julien et al., 2015; Chen et al., 2018) where the output set $\{\boldsymbol{y}_i\}_{i=1}^n$ is directly optimized in the data domain. By contrast, we parametrize $\boldsymbol{y}_i$ with $g(\boldsymbol{z}_i)$ for each i, and optimize the latent vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$.

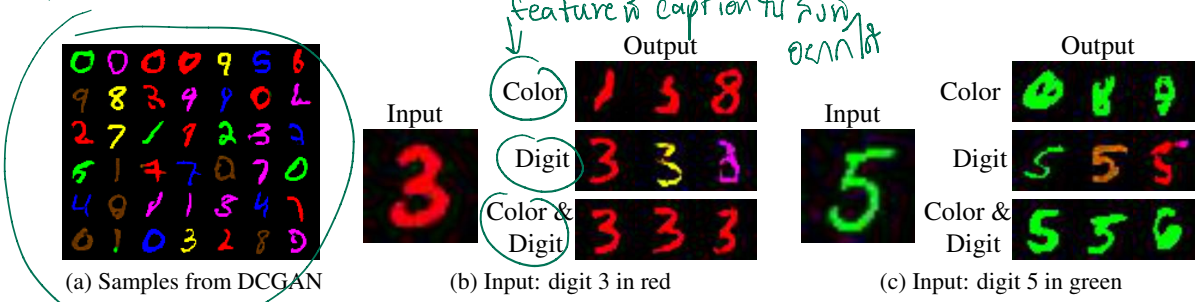
5. Experiments

In this section, we show that our approach is able to perform content-based image generation on many image datasets and GAN models. We first demonstrate the approach on a simple problem (MNIST) in Section 5.1, and verify that using a nonlinear kernel k (see (6)) helps improve the representation of the input set. We then demonstrate (in Section 5.2) that aspects of input images that should be captured can be easily controlled by changing the feature extractor (i.e., E in (6)). In the following sections, we consider generative modeling problems on real images (CelebA-HQ, LSUN-bedroom, LSUN-bridge, LSUN tower), and show that our approach is able to generate high-quality images that are relevant to the input, without retraining the GAN models. Python code is available at https://github.com/wittawatj/cadgan.

5.1. Better Representation with Nonlinear Kernels

To show the importance of a nonlinear kernel k in (6), we consider a DCGAN (Radford et al., 2015) model trained on MNIST.[1] The task in this case is to generate images of handwritten digits from the DCGAN model that are similar to the input. For reference, real and sampled (unconditionally) images are shown in Figures 2a, 2b, respectively. We compare two different kernels (k in (6)): 1) linear kernel, and 2) the IMQ kernel with kernel parameter c set to 10. To

[1] Pytorch code for the DCGAN model on MNIST: https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/dcgan/dcgan.py



(a) Samples from DCGAN (b) Input: digit 3 in red (c) Input: digit 5 in green

Figure 3: Content-based image generation on Colored MNIST with a DCGAN model. **(a)**: Unconditional samples from the model. **(b)**, **(c)**: $n = 3$ output images from the use of three different feature extractors. "Color" indicates a feature extractor which only captures image colors. "Digit" indicates an extractor given by a CNN-based digit classifier (10 classes). "Color & Digit" stacks features from both extractors. The output images are consistent with the input in the sense as specified by the extractors used.

isolate the effect of nonlinearity from the kernel, and nonlinear transformation in the extractor E, we first consider no feature extractor E i.e., the kernel is directly applied on the pixel values. Figure 2d shows the output from our approach with input images X_m given in Figure 2c and input weights $(w_1, \ldots, w_m) := (1/m, \ldots, 1/m)$.

We see in Figure 2d that images generated with the IMQ kernel faithfully capture the input images. The failure of the linear kernel can be explained by noting that the input mean embedding in this case is given by $\hat{\mu}_P = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i$ and is simply a superimposition of all the input images. It is clear that this does not represent set-level information contained in the input set e.g., that there two 2's, and two 9's. On the other hand, the use of the IMQ kernel results in $\hat{\mu}_P = \frac{1}{m}\sum_{i=1}^{m} \psi(\boldsymbol{x}_i)$ where $\psi(\cdot)$ is an infinite-dimensional map induced by the kernel, and provably provides a more powerful representation e.g., IMQ kernels are C_0-universal (Sriperumbudur et al., 2011, p. 2397).

In general, one would require a feature extractor E which specifies relevant aspects of the input images to capture. We show that even with the presence of a nonlinear feature extractor, it is still beneficial to put a nonlinear kernel on top of extracted features. We consider the same MNIST problem where the extractor E is set to the output of the first two convolutional layers of a convolutional neural network (CNN) classifier trained to classify the ten digits of real MNIST images.[2] It has been observed that the first few convolutional layers roughly capture low-level image features. The generated images are shown in Figure 2e. We observe that the generated images with the linear kernel appear to be closer to being handwritten digits than in the previous case where no extractor is used (top figure of Figure 2d); they are, however, still far from the input images.

[2]Pytorch code for the CNN classifier on MNIST: https://github.com/pytorch/examples/blob/master/mnist/main.py.

We see that using the IMQ kernel on top of the features gives good results since the extracted features are further expanded by the nonlinear kernel.

5.2. Flexible Control over the Similarity Criterion

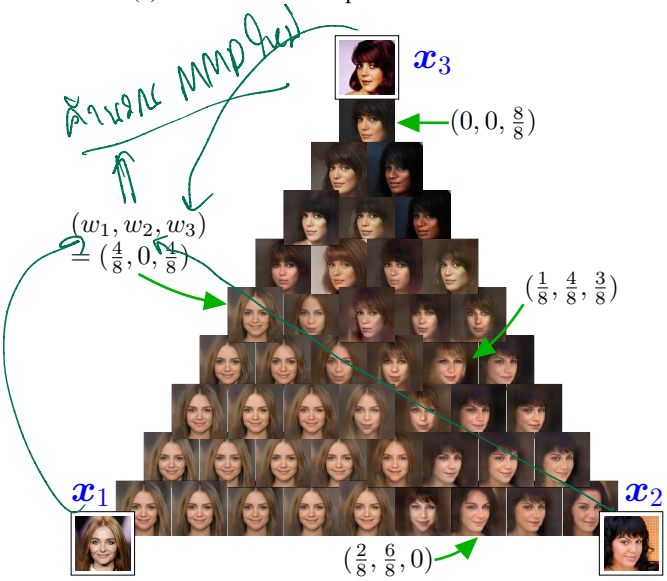
In this section, we demonstrate that aspects of the input images that will be captured in the output images can easily be controlled by changing the extractor E. We construct a colored version of the MNIST problem where each image in the original dataset is colored with six colors: red, green, blue, yellow, pink, and brown, thereby creating six new data points for each image. A DCGAN model (not class-conditional) is trained on the new dataset (details in Section B) and is used for content-based generation with the same IMQ kernel as in Section 5.1. We consider three different choices for the extractor:

1. **Color**: For each input image, perform channel-wise max pooling such that the result is a 2×2-pixel image (three color channels). Treat the $3 \times 2 \times 2 = 12$ output values as extracted features. The operation roughly captures the overall color of the digit. Note that the background color is black i.e., RGB pixel values are (0, 0, 0), and does not influence the features.

2. **Digit**: Extracted features are the ten outputs of the final layer of a CNN-based classifier trained to classify the ten digits in MNIST. The classifier is the same network as used in Figure 2e. We convert images to grayscale by averaging across the three color channels before feeding to the classifier. This extractor is expected to capture only the digit identity of the handwritten digit, ignoring the color.

3. **Color & Digit**: Extracted features are concatenation of outputs from the two extractors above. This extractor is designed to capture both the color and the digit identity.

(a) Unconditional samples from the GAN model



(b) Generated images given three input images and their weights

Figure 4: Compression (Section 5.3): generate one image so as to match the (weighted) mean feature of $m = 3$ input images. **(a)**: Unconditional samples from the GAN model studied in Mescheder et al. (2018) (trained on the CelebA-HQ dataset). **(b)**: Generated images from the proposed procedure given three input images $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ (bordered images in the corners), and input weights w_1, w_2, w_3. For a higher resolution image, see Figure 10 in the appendix.

Unconditional samples from the model, and generated results are shown in Figure 3. In both test cases (Figures 3b and 3c), the output images are consistent with the input in the sense as specified by the extractor being used. Specifically, when the Color extractor is used, the generated images have the same color as the input image, but with a variety of digit types. When the Digit extractor is used, the output images contain digits of the same digit type, but with diverse colors. We emphasize that the extractor can be changed at run time, without retraining the marginal generative model.

5.3. Compression by Matching the Mean

A noteworthy special case of our formulation is when $m > n$ (more input images than output images). In this case, the output mean embedding $\hat{\mu}_Q$ has fewer degrees of freedom than the input mean embedding $\hat{\mu}_{P,\boldsymbol{w}}$ in the sense that there are fewer summands. As a result, for the two mean embeddings to match, each output image is forced to combine features from multiple input images. For this reason, we refer to this task as the *compression* task. An interesting instance of this task is when $m = 3$ and $n = 1$. With $m = 3$ input images, the (weighted) input mean embedding can be written as $\hat{\mu}_{P,\boldsymbol{w}} = \sum_{i=1}^{2} w_i k(E(\boldsymbol{x}_i), \cdot) + (1 - w_1 - w_2) k(E(\boldsymbol{x}_3), \cdot)$, where $w_1, w_2 \in [0, 1]$ specifies the relative importance of the first two input images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively. The weight for the third input $\boldsymbol{x}_3$ is given by $w_3 = 1 - w_1 - w_2$. These weights give an extra freedom to control how much each of the input images contributes to the mean feature that should be matched by the output mean embedding.

To illustrate the compression, we use a GAN model from Mescheder et al. (2018) pretrained on the CelebA-HQ problem (Karras et al., 2017). Sample images from the model are shown in Figure 4a (more in Figure 22 in the appendix). We use the same IMQ kernel as used previously, and set the extractor E to be the output of layer Relu3-3 of the VGG-Face network (Parkhi et al., 2015).[3] The images generated from our procedure are shown in Figure 4b for various settings of the input weights $(w_1, w_2, w_3) =: \boldsymbol{w}$. Each of the output images is positioned such that the closeness to a corner (an input image) indicates the importance (weight) of the corresponding input image. See Figure 9 for a precise weight vector specification at each position. We observe that when one of the weights is exactly one (i.e., equivalent to the problem of having only $m = 1$ input image), the output image almost reproduces the input image (see the output images in the corners). When only one of the weights is 0 (i.e., equivalent to having $m = 2$ input images), the output image interpolates between the two input images (see the output images along the edges of the triangle). Beyond these two special cases, varying the weights so that $w_1 > 0, w_2 > 0$ and $w_3 > 0$ appears to smoothly blend key visual features of the three input faces, giving output images which are consistent with all the input images and weights (see the images in the interior of the triangle). More compression results can be found in Section E (appendix).

We emphasize that changing the weight between two input images is not equivalent to a commonly used approach of linearly interpolating between the latent vector that generates $\boldsymbol{x}_1$ and the latent vector that generates $\boldsymbol{x}_2$. In our procedure, for each $\boldsymbol{w}$, the obtained latent vector $\boldsymbol{z}_{\boldsymbol{w}}$ satisfies $\boldsymbol{z}_{\boldsymbol{w}} = \arg\min_{\boldsymbol{z}} \|\hat{\mu}_{P,\boldsymbol{w}} - k(E(g(\boldsymbol{z}), \cdot))\|^2_{\mathcal{H}}$ and is such that $g(\boldsymbol{z}_{\boldsymbol{w}})$ is an image whose feature vector is close to the mean feature defined by $\hat{\mu}_{P,\boldsymbol{w}}$. Simply interpolating between two latent vectors may not give output images with this property.

[3] Pretrained VGG-Face models are available at http://www.robots.ox.ac.uk/~vgg/software/vgg_face/.

5.4. Content-Based Generation of Complex Scenes

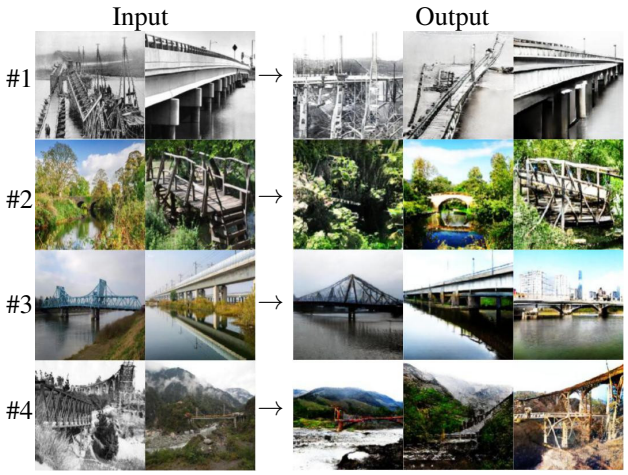
In the final experiment, we demonstrate our content-based generation method in its full generality (i.e., $m > 1$ and $n > 1$) on images of complex scenes. We consider three categories of the LSUN dataset (Yu et al., 2015): bedroom, bridge, tower, and use pretrained GAN models from Mescheder et al. (2018) which were trained separately on training samples from each category. The models are based on DCGAN architecture with additional residual connections (He et al., 2016). Unconditional samples from these models can be found in Figures 19, 20 and 21, respectively in the appendix. For content-based generation, we use the IMQ kernel with parameter $c = 100$ and set the extractor E to be the output of the layer before the last fully connected layer of a pretrained Places365-ResNet classification model (Zhou et al., 2017).[4] This network was trained to classify 365 unique scenes (training set comprising ten million images), and is expected to be able to capture high-level visual features of complex scenes.

Our results in Figure 5 show that in each test case, the three generated images are highly consistent with the two input images (from the LSUN's test set). For instance, in bridge#1 (test case #1 of the LSUN-bridge category in Figure 5), not only is the tone black-and-white but the bridge structure is also well captured. In other cases such as tower#1, our procedure appears to generate similar buildings as present in the input images, but with a different viewing angle. This feat demonstrates that the proposed procedure can generate images that are *semantically similar* to the input. Our procedure does not degrade the quality of the generated images (compare the image quality to that of unconditional samples in Figures 19, 20 and 21).

6. Discussion and Outlook

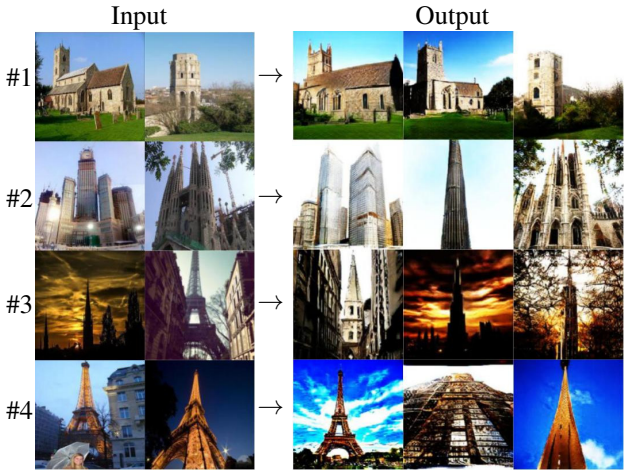
We have presented a procedure for constructing a content-based generator by leveraging existing pretrained unconditional generative models. To our knowledge, this is the first work that addresses this setting, at test time, and without retraining the underlying models. There are opportunities for improvement. One topic of current research is on theoretically grounded, quantitative measure of the coherence between input and output sets of images, which are relatively small, compared to model evaluation of GANs in general (Heusel et al., 2017; Bińkowski et al., 2018; Jitkrittum et al., 2018). Preliminary results on quantitative evaluation of our approach are presented in Section A (appendix). More experimental results can be found in the appendix.

[4]Pretrained Places365 networks are available at: https://github.com/CSAILVision/places365.



(a) LSUN-bridge

(b) LSUN-bedroom



(c) LSUN-tower

Figure 5: Generated output images from our approach. In each of the three LSUN categories, there are four test cases (denoted by #1, . . ., #4), each containing two input images from the LSUN test set.

Acknowledgements

We thank the reviewers for their thoughtful comments. We thank Tom Wallis for a fruitful discussion. Patsorn Sangkloy is supported by the Royal Thai Government Scholarship Program.

References

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 1

Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1355–1362, 2012. 2, 4

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer, 2004. 2

Mikolaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018. 4, 6

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 4

Wilson Ye Chen, Lester Mackey, Jackson Gorham, Francois-Xavier Briol, and Chris Oates. Stein points. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 844–853. PMLR, 10–15 Jul 2018. 2, 4

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016. 1

Yutian Chen, Max Welling, and Alexander J. Smola. Supersamples from kernel herding. In *UAI*, 2010. 1, 2, 2, 4

Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. Kernel measures of conditional dependence. In *Advances in Neural Information Processing Systems*, pages 489–496, 2008. 2

Kenji Fukumizu, Le Song, and Arthur Gretton. Kernel bayes' rule: Bayesian inference with positive definite kernels. *Journal of Machine Learning Research*, 14(1): 3753–3783, 2013. 2

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016. 1

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 1, 4

Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. In *ICML*, pages 1292–1301. PMLR, 2017. 2

A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012a. 1, 2, 2

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems 25*, pages 1205–1213. 2012b. 2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5.4

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*. 2017. 6, 2

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007. 2

Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1510–1519, 2017. 1

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *arXiv preprint arXiv:1804.04732*, 2018. 4

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017. 4

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 4

Wittawat Jitkrittum, Heishiro Kanagawa, Patsorn Sangkloy, James Hays, Bernhard Schölkopf, and Arthur Gretton. Informative features for model comparison. In *Advances in Neural Information Processing Systems 31*. 2018. 6

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2017. 1, 5.3

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 4

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 3

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 1

Iryna Korshunova, Jonas Degrave, Ferenc Huszar, Yarin Gal, Arthur Gretton, and Joni Dambre. Bruno: A deep recurrent model for exchangeable data. In *Advances in Neural Information Processing Systems 31*, pages 7190–7198. 2018. 1, 4

Simon Lacoste-Julien, Fredrik Lindsten, and Francis Bach. Sequential Kernel Herding: Frank-Wolfe Optimization for Particle Filtering. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 544–552. PMLR, 09–12 May 2015. 2, 4

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017. 4

Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. In *ICML*, pages 1718–1727, 2015. 4

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017. 4

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016. 1

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *ICML*, 2018. 1, 4, 5.3, 5.4, 19, 20, 21, 22

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv e-prints*, art. arXiv:1411.1784, November 2014. 1, 4

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv e-prints*, art. arXiv:1610.03483, October 2016. 1

Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, volume 2, page 7, 2017. 1, 4

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016. 1

O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015. 5.3

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 4

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017a. 4

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017b. 4

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 5.1

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR. 1, 4

Yong Ren, Jun Zhu, Jialian Li, and Yucen Luo. Conditional generative moment-matching networks. In *Advances in Neural Information Processing Systems*, pages 2928–2936, 2016. 1

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. 1

Patsorn Sangkloy, Jingwan Lu, Chen Fang, FIsher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. *Computer Vision and Pattern Recognition, CVPR*, 2017. 1

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3

Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 13–31, 2007. 2, 4

Le Song, Xinhua Zhang, Alex Smola, Arthur Gretton, and Bernhard Schölkopf. Tailoring density estimation via reproducing kernel moment matching. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 992–999. ACM, 2008. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390281. 2

Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12:2389–2410, 2011. 2, 5.1

Dougal J. Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *ICLR*. 2016. 4

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. 1

Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional gaussian processes. In *Advances in Neural Information Processing Systems 30*, pages 2849–2858. 2017. 3

Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016. 4

Wei Wang, Yuan Sun, and Saman Halgamuge. Improving MMD-GAN training with repulsive loss function. In *International Conference on Learning Representations*, 2019. 4

Li Wenliang, Dougal Sutherland, Heiko Strathmann, and Arthur Gretton. Learning deep kernels for exponential family densities. *arXiv e-prints*, art. arXiv:1811.08357, 2018. 3

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016. 3

W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays. Texturegan: Controlling deep image synthesis with texture patches. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, June 2018. doi: 10.1109/CVPR.2018.00882. 1

Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with GAN for transferring multiple face attributes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–187, September 2018. 4

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. 2018. 1

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016. 1

Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493, 2017. 4

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5.4

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017. 4

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5.4

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. 4

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 1, 4

Kernel Mean Matching for Content Addressability of GANs

Supplementary

A. Quantitative Evaluation

Quantitative evaluation of our proposed procedure is a topic of ongoing research. As a preliminary result, we consider two ways to measure the distance between the input and output sets of images:

1. Learned Perceptual Image Patch Similarity (**LPIPS**, Zhang et al. (2018)) is a similarity measure between two images which has been shown to correlate well with human perceptual similarity. We use a VGG network pre-trained on ImageNet as the feature extractor. Given an input set $X_m = \{\boldsymbol{x}_i\}_{i=1}^m$ and an output set $Y_n = \{\boldsymbol{y}_j\}_{j=1}^n$, we use the mean of LPIPS computed on all input-output pairs as the score for measuring the coherence between the input and output sets:

$$\text{mean-LPIPS}(X_m, Y_n) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\text{LPIPS}(\boldsymbol{x}_i, \boldsymbol{y}_j).$$

   Lower mean-LPIPS means higher coherence between the input and the output sets.

2. Fréchet Inception Distance (**FID**, Heusel et al. (2017)) which has recently become a commonly used approach for measuring the distance between real and generated images for evaluating a GAN model. Here we compute the FID between the input images $X_m = \{\boldsymbol{x}_i\}_{i=1}^m$ and the generated output images $Y_n = \{\boldsymbol{y}_j\}_{j=1}^n$. We use the pool-3 layer of the Inception network as the feature extractor.

We randomly sample $m = 3$ images from the respective LSUN held-out set (LSUN Bridge, LSUN Bedroom, LSUN Tower) as input to generate $n = 3$ images (repeat for 100 trials). All parameter settings are the same as used to produce Figure 5. As a baseline, we consider a procedure which simply generates $n = 3$ images from the GAN model (independently of the input X_m). This procedure is referred to as "Prior". The results are shown in Table 1.

Table 1: Quantitative evaluation of our proposed procedure using LPIPS and FID. We report means and standard deviations of mean-LPIPS and FID computed from 100 trials (lower is better).

	mean-LPIPS	**FID**
LSUN Bridge		
Ours	$\mathbf{0.698} \pm 0.033$	$\mathbf{248.78} \pm 66.61$
Prior	0.731 ± 0.025	345.38 ± 44.76
LSUN Bedroom		
Ours	$\mathbf{0.703} \pm 0.033$	$\mathbf{194.76} \pm 55.06$
Prior	0.732 ± 0.026	214.36 ± 47.30
LSUN Tower		
Ours	$\mathbf{0.689} \pm 0.029$	$\mathbf{267.49} \pm 72.68$
Prior	0.692 ± 0.025	298.25 ± 48.68

The results confirm that the generated images from our procedure have higher coherence (lower LPIPS, and lower FID) to the input than do images unconditionally sampled from the model.

B. Colored MNIST Experiment

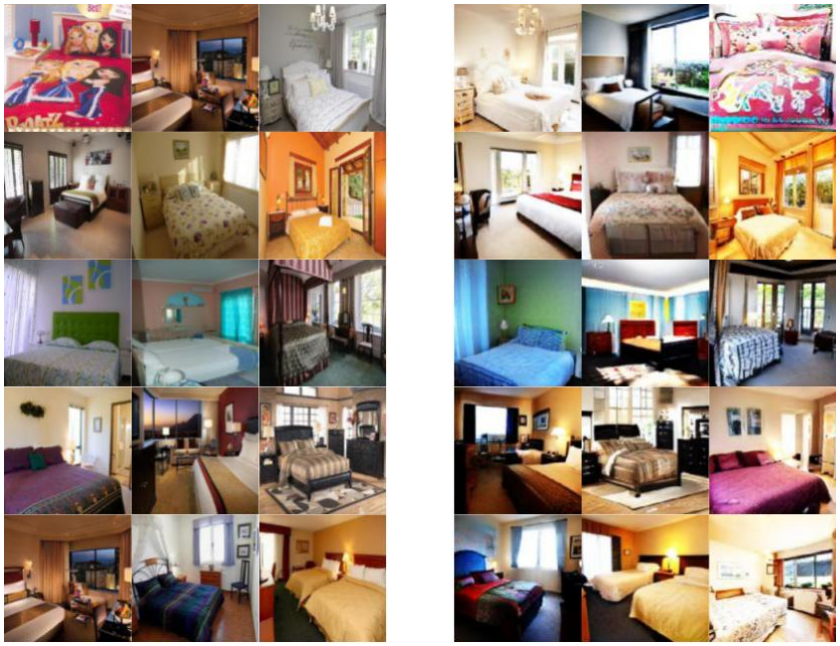
In this section, we give more details of the experiment described Section 5.2. For each image in the original MNIST dataset, six images are created by coloring it. The RGB colors are Red: (1, 0, 0), Green: (0, 1, 0), Blue: (0, 0, 1), Yellow: (1, 1, 0), Pink (magenta): (1, 0, 1), and Brown: (0.4, 0.2, 1). Pytorch code for the DCGAN generator is given below.

```
class generator(nn.Module):

    def __init__(self):
        super(generator, self).__init__()
        depth = 64
        self.deconv1 = nn.ConvTranspose2d(100, depth*8, 4)
        self.bn1 = nn.BatchNorm2d(depth*8)
        self.deconv2 = nn.ConvTranspose2d(depth*8, depth*4, 4, stride=2, padding=1)
        self.bn2 = nn.BatchNorm2d(depth*4)
        self.deconv3 = nn.ConvTranspose2d(depth*4, depth*2, 4, stride=2, padding=2)
        self.bn3 = nn.BatchNorm2d(depth*2)
        self.deconv4 = nn.ConvTranspose2d(depth*2, 3, 4, stride=2, padding=1)
        self.relu = nn.ReLU(inplace=True)
        self.sigmoid = nn.Sigmoid()

    def forward(self, input):
        out = self.relu(self.bn1(self.deconv1(input)))
        out = self.relu(self.bn2(self.deconv2(out)))
        out = self.relu(self.bn3(self.deconv3(out)))
        out = self.sigmoid(self.deconv4(out))
        return out
```

C. More Results (LSUN-Bedroom)

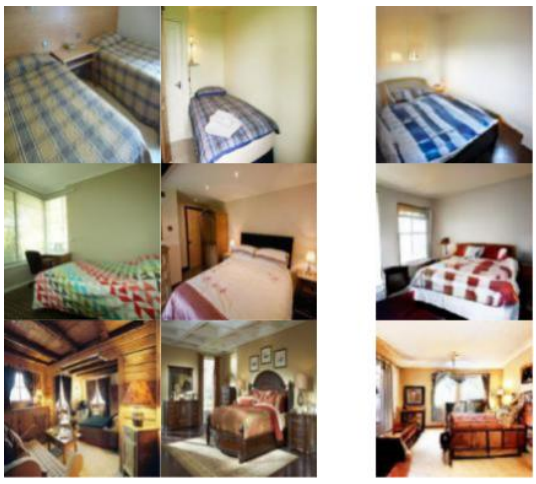
LSUN-bedroom In the following figure, each row shows one input-output pair ($m = 3$ input images $\rightarrow n = 3$ output images). The left column contains input images, and the right column contain output images generated by our proposed method.

Input Output



LSUN-bedroom, compression $m = 2$ input images $\rightarrow n = 1$ output image.

Input Output



D. Failure Cases

In this section, we present some failure cases from our proposed method.

Far Outside the Range of g We observe that our procedure can fail when the input images are too different from the

images used to train the chosen implicit model. To illustrate, we consider the same model and kernel settings as in Figure 4b (CelebA-HQ problem). Figure 6 shows examples of such failure case.

(a) Case 1 (b) Case 2

Figure 6: Failure cases of our approach due to large discrepancy of the input images and the images used to train the model. Here, the GAN model used is the same one used in Figure 4b (CelebA-HQ model trained on images of celebrities). Inspection of unconditional samples shown in Figure 22 indicates that visual features of the input images are underrepresented by the model. Presumably these input images may be far from the output manifold of the model.

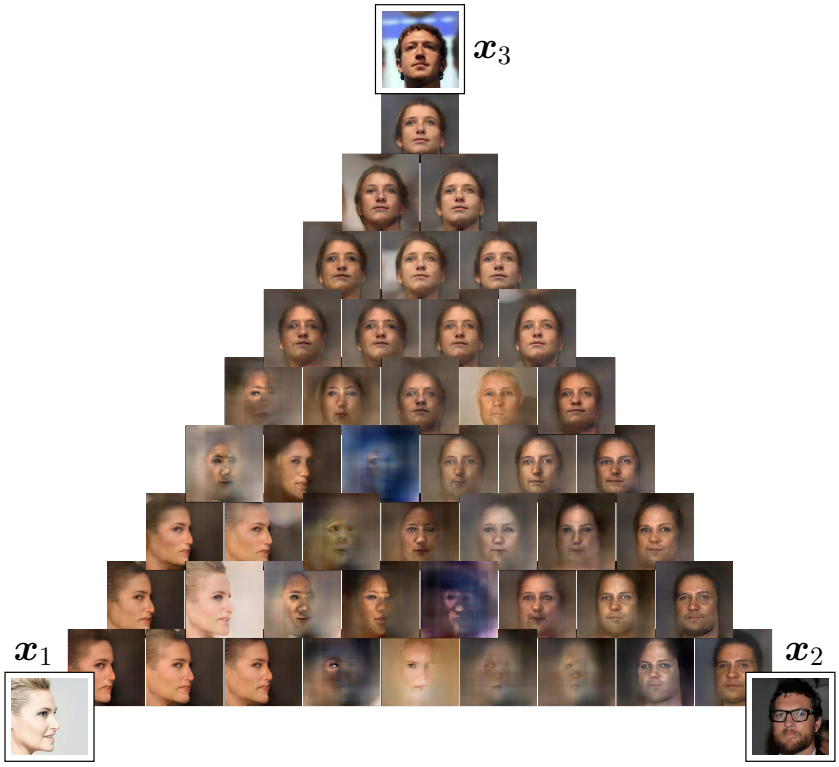


Figure 7: A failure case of our approach due to large discrepancy of one input image and the images used to train the model. The GAN model is the same one used in Figure 4b. Here, $\boldsymbol{x}_1$ is an image showing only the right side of a face. Presumably, non-frontal faces may be underrepresented by the model. While our procedure can generate an output image which is consistent to $\boldsymbol{x}_1$ when it is the only input image (see the output image closest to $\boldsymbol{x}_1$), when feature combination is enforced by using weights $w_1 > 0, w_2 > 0$ and $w_3 > 0$, the procedure fails to generate coherent output images (see the images in the interior of the triangle). We suspect that the model has been trained with relatively few images of non-frontal faces; so generating non-frontal faces with the required variations (as specified by $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$) may be challenging.

Repeated Outputs When the model can generate the specified ($m = 1$) input image well, the output set — in the case of $n > 1$ — may contain almost identical output images. This is illustrated in Figure 8.

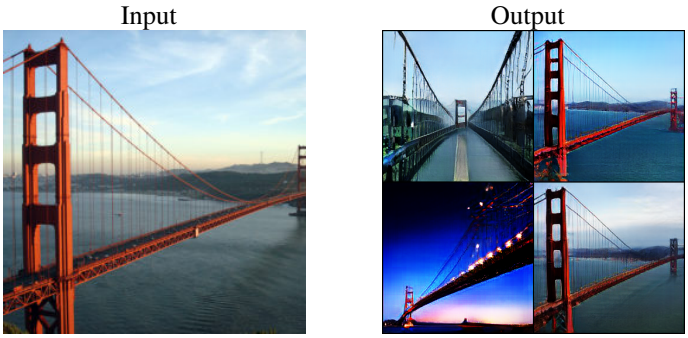


Figure 8: The procedure can give almost identical output images when the underlying g can model the given input image $\boldsymbol{x}$ well. Here, we consider the same model and other hyperparameters as used in Figure 5a (i.e., LSUN-bridge model). Mathematically, this means that there exists a latent vector $\boldsymbol{z}$ such that $g(\boldsymbol{z}) \approx \boldsymbol{x}$. As a result, the mean feature can be matched if such $\boldsymbol{z}$ (or very small perturbation of such $\boldsymbol{z}$) is repeatedly produced.

E. Compression from $m = 3$ Input Images to $n = 1$ Output Image

In this section, we present more results from the compression experiment presented in Section 5.3. The generative model, the feature extractor used, and other hyperparameters are the same as used to produce the result in Figure 4b. The results are shown in Figures 10 and 11 where the output images (given $m = 3$ input images) are arranged in a simplex (an equilateral triangle with the three input images at the three corners). Each of the output images is positioned such that the closeness to a corner (an input image) indicates the importance (weight) of the corresponding input image. A precise weight vector specification at each position is shown in Figure 9. As a special case of compression with $m = 3$, we present results from compression with $m = 2$ input images in Figure 12.

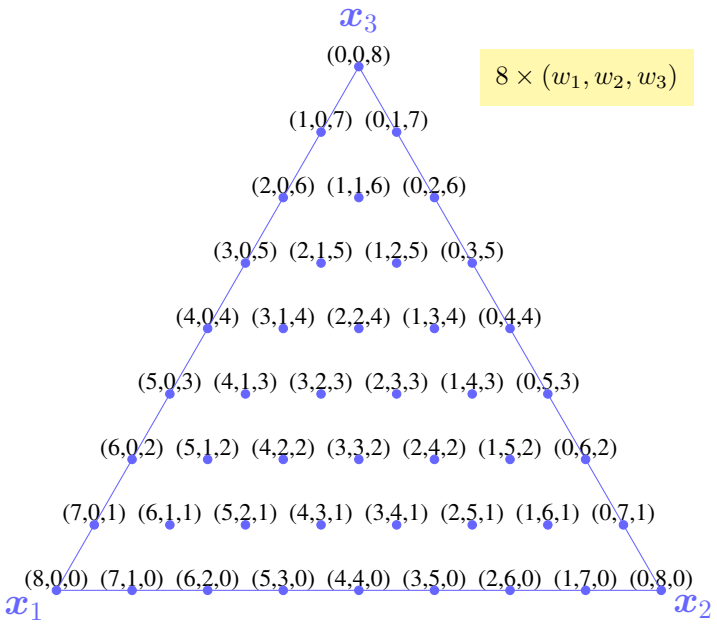


Figure 9: Weight vector specification in the compression experiment (see Section 5.3) with $m = 3$ input images: $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$. Each position in the triangle corresponds to one value of $\boldsymbol{w} = (w_1, w_2, w_3)$ such that $w_1, w_2, w_3 \in [0, 1]$ and $\sum_{i=1}^{3} w_i = 1$. To avoid cluttering the figure, we show $8 \times (w_1, w_2, w_3)$ instead of (w_1, w_2, w_3).

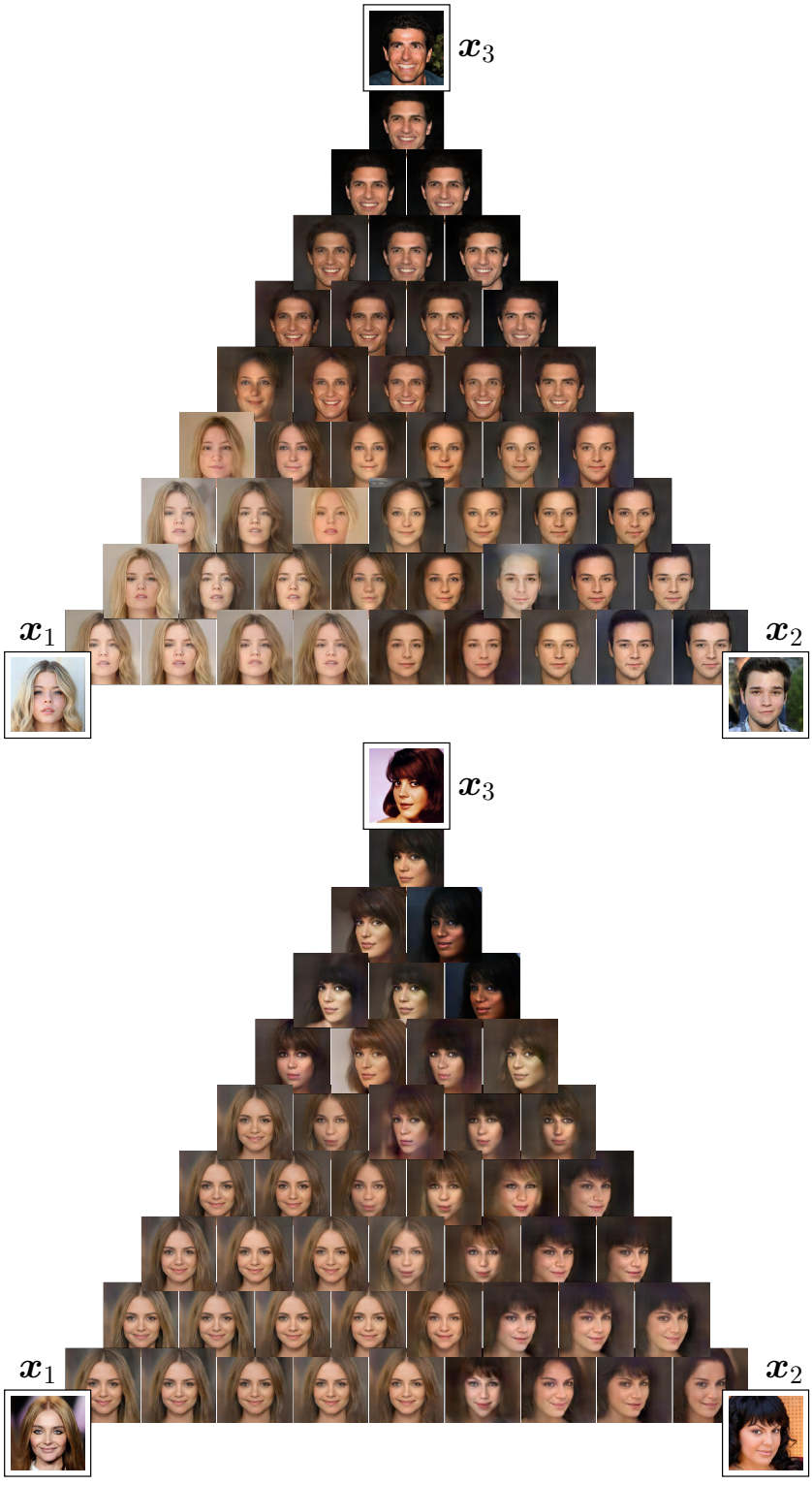


Figure 10: Compression with $m = 3$: generate one image so as to match the (weighted) mean feature of $m = 3$ input images. All hyperparameters are the same as used to produce Figure 4b. Each of the output images is positioned such that the closeness to a corner (an input image) indicates the importance (weight) of the corresponding input image. See Figure 9 for a precise weight vector specification at each position.

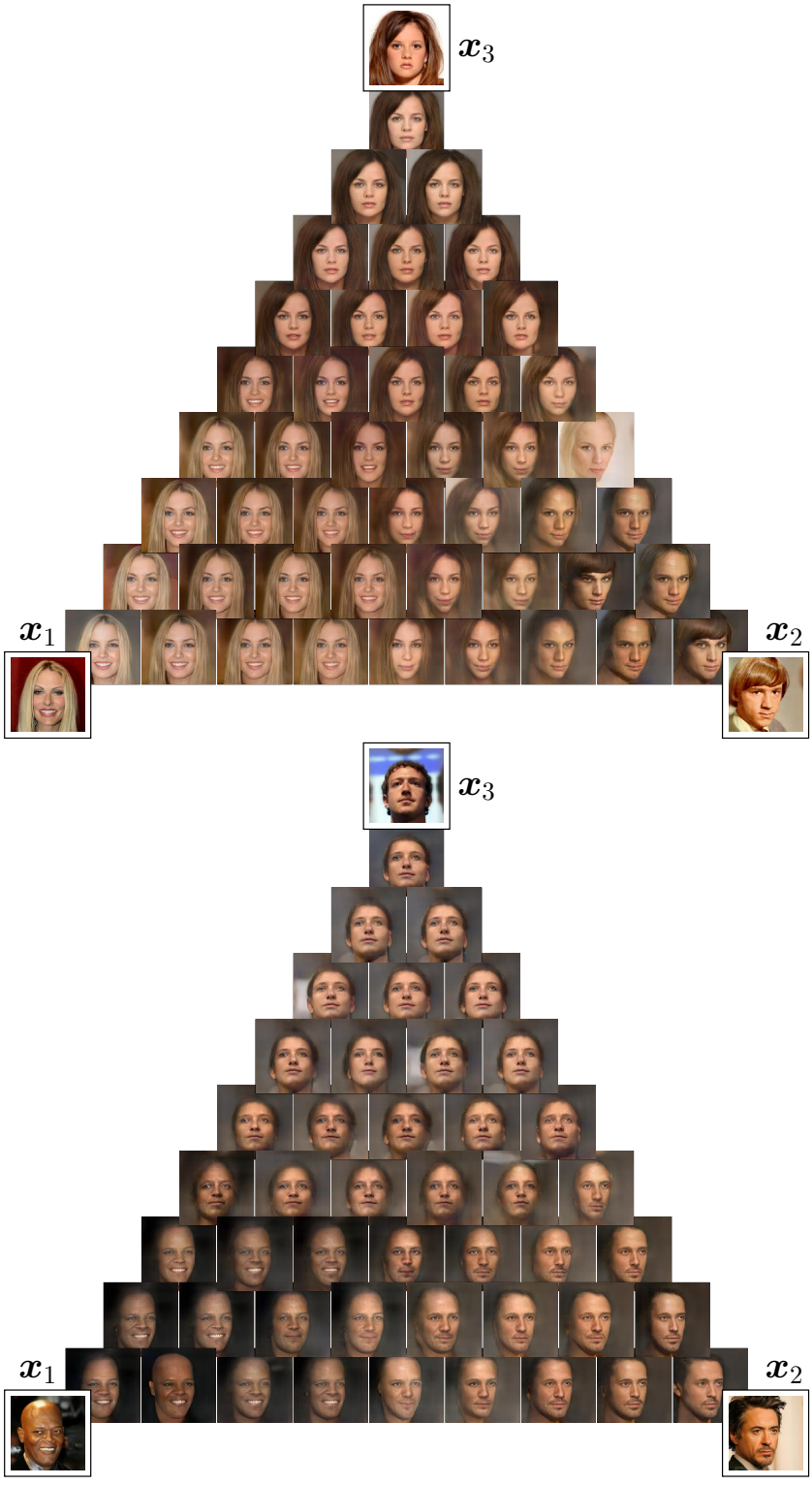


Figure 11: Compression with $m = 3$: generate one image so as to match the (weighted) mean feature of $m = 3$ input images. All hyperparameters are the same as used to produce Figure 4b. Each of the output images is positioned such that the closeness to a corner (an input image) indicates the importance (weight) of the corresponding input image. See Figure 9 for a precise weight vector specification at each position.

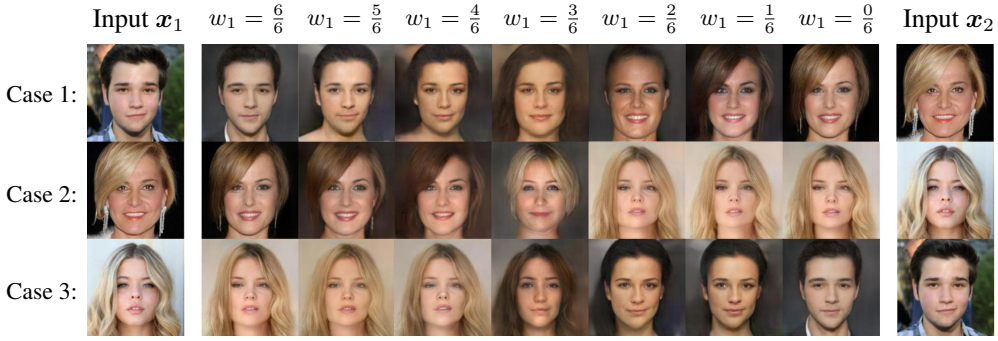


Figure 12: Compression with $m = 2$: generate one image so as to match the (weighted) mean feature of $m = 2$ input images. Generated images from the proposed procedure given two inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from three independent test cases. The weight w_1 specifies the emphasis on the input $\boldsymbol{x}_1$. The weight on $\boldsymbol{x}_2$ is given by $w_2 = 1 - w_1$.

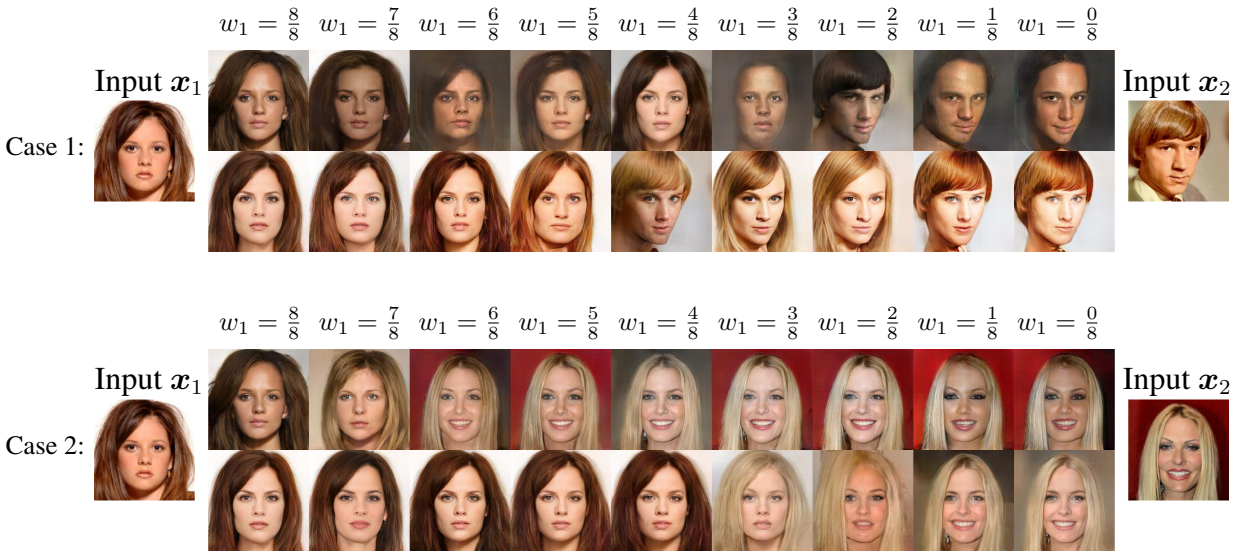


Figure 13: Generate $n = 2$ output images from $m = 2$ input images. See details in Section F.

F. From $m = 2$ Input Images to $n = 2$ Output Images

Compression (combining features of the input images) arises when $m > n$ i.e., more input than output images. The case of $m < n$ is considered in Figure 5 (LSUN) which can be seen as a set expansion procedure or "more-like-this" generation i.e., generate a set of diverse output images which are consistent with the specified input set. In this section, we consider the case where $m = n$. All hyperparameter settings are the same as in Section 5.3. The results are shown in Figures 13 and 14. When $w_1 = 0$ or $w_1 = 1$, the problem reduces to the case where $m = 1$ and $n = 2$ (i.e., set expansion). In these cases, we observe that the two output images are different and both bear some similarity to the one input image. When $0 < w_1 < 1$, the procedure appears to create variations of the two input images.

G. Runtime Vs n (Number of Output Images)

The complexity of the proposed procedure in Eq. (5) is $O(n^2 + mn)$ per optimization iteration. In each iteration, the gradient vector of size $d_z n$ needs to be computed, where d_z is the latent dimension of the GAN model. Figures 15a and 15b show average runtimes per iteration (in milliseconds) for Colored MNIST and LSUN bedroom. Standard deviation is in the order of 10^{-5}. The problem setting follows that considered in Figure 3 and Figure 5, respectively.

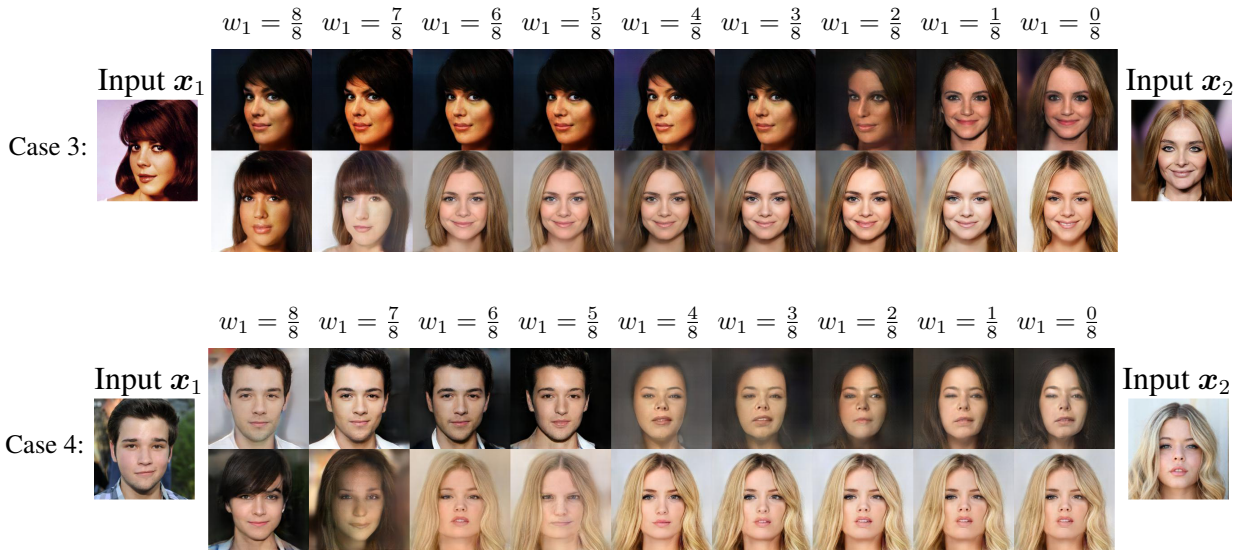
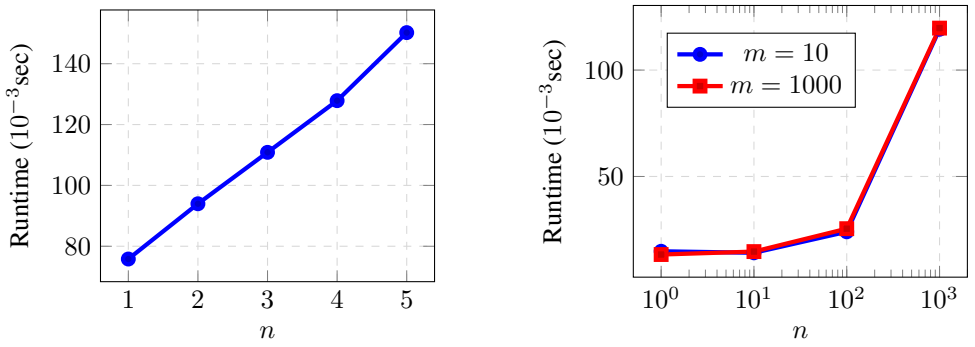


Figure 14: Generate $n = 2$ output images from $m = 2$ input images. See details in Section F.



(a) LSUN Bedroom. $m = 1$.

(b) Colored MNIST

Figure 15: Average runtime per optimization iteration. We observe that runtimes largely depend on n. See details in Section G.

H. Objective Value vs n

We observe that increasing n decreases the objective value (see Figure 16). That is, more output images allow better matching of the input mean features.

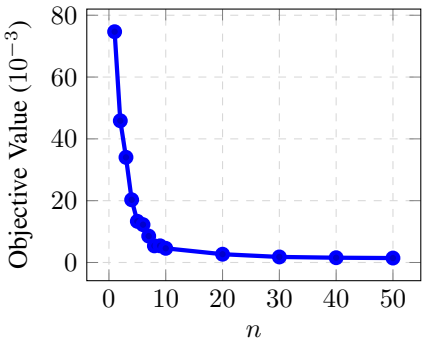


Figure 16: Objective value across n (lower is better). Here we consider the MNIST problem (Figure 2) with input set containing $m = 50$ images: five images from each of the ten classes. Since the input contains ten types of digits, we expect at least $n = 10$ images to be able to well capture the input set. This explains why the objective decreases sharply from $n = 1$ to $n = 10$. At $n = 10$, the optimized output images contain the ten digits.

I. Large m, n

The aim of this section is to illustrate the results of our procedure when m, n are large. We consider the MNIST problem as in Section 5.1 using the IMQ kernel and the CNN feature extractor. We set $m = 14$ and $n = 24$. The results are shown in Figure 17. A key point is that the class proportions of the input images are respected in the generated images. For example, in the top test case in Figure 17, the 0 digit forms the majority in both the input and the output sets.

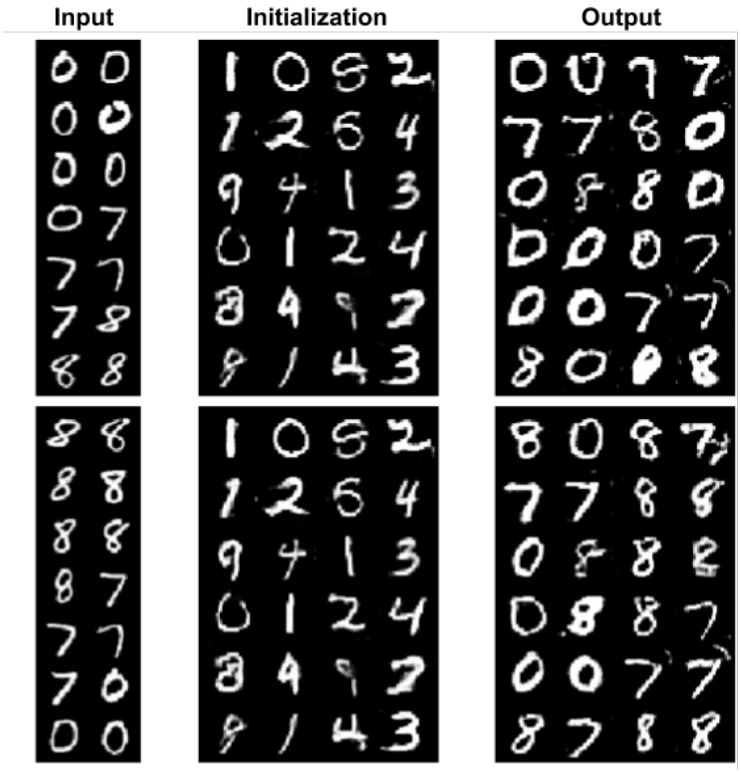
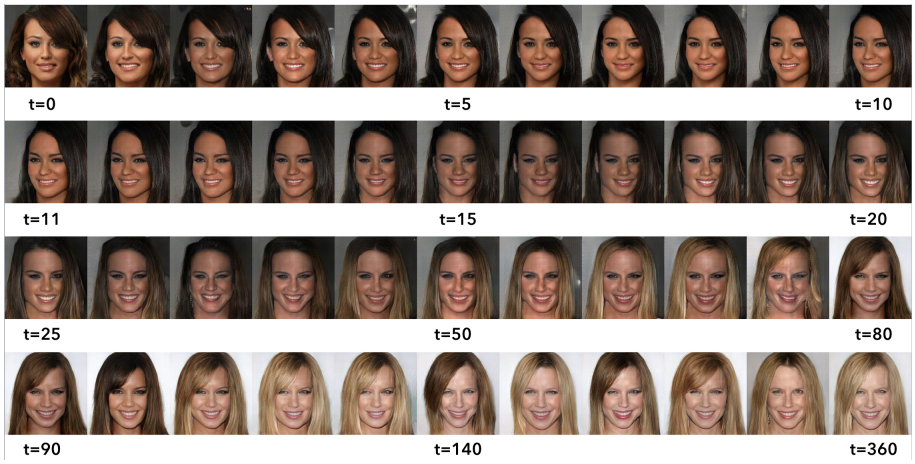


Figure 17: Content addressable image generation on MNIST with $m = 14$ and $n = 24$. There are two independent test cases: top and bottom. **Input**: X_m containing 14 input images. **Initialization**: 24 generated images from the latent vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{24}$ randomly drawn from the prior distribution i.e., initial points for the optimization. **Output**: Output images Y_n.

J. Optimization Trajectory

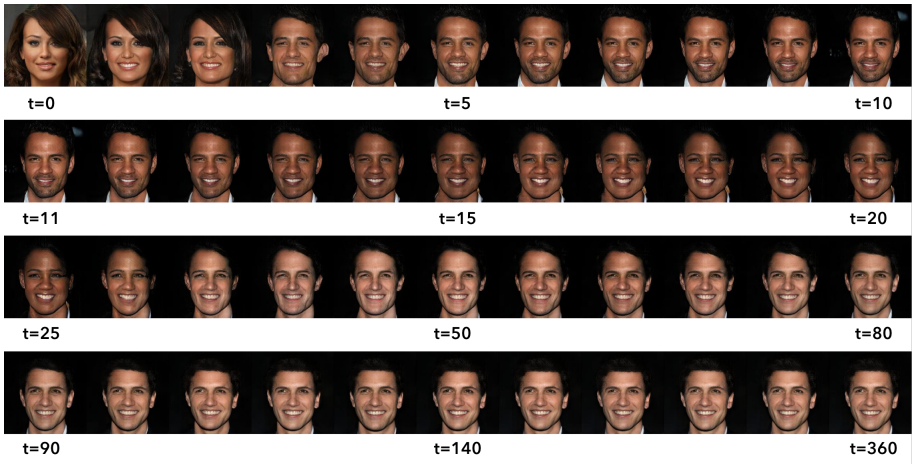
The aim of this section is to show how the output image during the optimization for solving (5) looks like. We consider the CelebA-HQ problem with $m = 1, n = 1$ and all other hyperparameters are the same as used to produce the result in Figure 4b. With $m = 1, n = 1$, the problem is to find one latent vector $\boldsymbol{z} = \arg\min_{\boldsymbol{z}} \|k(E(\boldsymbol{x}_1, \cdot)) - k(E(g(\boldsymbol{z}), \cdot))\|^2_{\mathcal{H}}$ given $n = 1$ input image $\boldsymbol{x}_1$. Iteratively solving this optimization with Adam creates a sequence of latent vectors $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(T)}$, where $\boldsymbol{z}^{(t)}$ denotes the latent vector from the t^{th} iteration. The output image from the t^{th} iteration is given by $g(\boldsymbol{z}^{(t)})$ where g is the pre-trained GAN model. The output images from selected iterations until $t = 360$ are shown in Figure 18. The changes after this iteration are barely visible.

(a) Input $\boldsymbol{x}_1$



(b) Output images during the optimization with input image in Figure 18a

(c) Input $\boldsymbol{x}_1$



(d) Output images during the optimization with input image in Figure 18c

Figure 18: Output images from selected iterations during the optimization of (5) until iteration $t = 360$. The changes after this iteration are barely visible. The image at $t = 0$ corresponds to the image generated from a randomly drawn latent vector from the prior distribution (initial point). In both cases, the initial point is set to be the same.

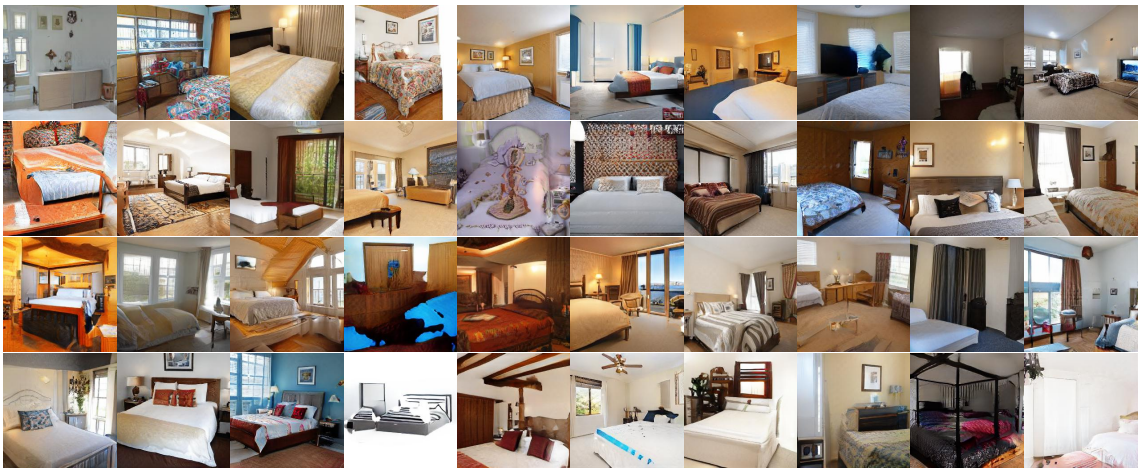


Figure 19: Samples from Mescheder et al. 2018's unconditional GAN model trained on LSUN-bedroom.

Figure 20: Samples from Mescheder et al. 2018's unconditional GAN model trained on LSUN-bridge dataset.

Figure 21: Samples from Mescheder et al. 2018's unconditional GAN model trained on LSUN-tower dataset.

Figure 22: Samples from Mescheder et al. 2018's unconditional GAN model trained on CelebA-HQ dataset.