MEMORIZATION PRECEDES GENERATION: LEARNING UNSUPERVISED GANS WITH MEMORY NETWORKS

Anonymous authors
Paper under double-blind review

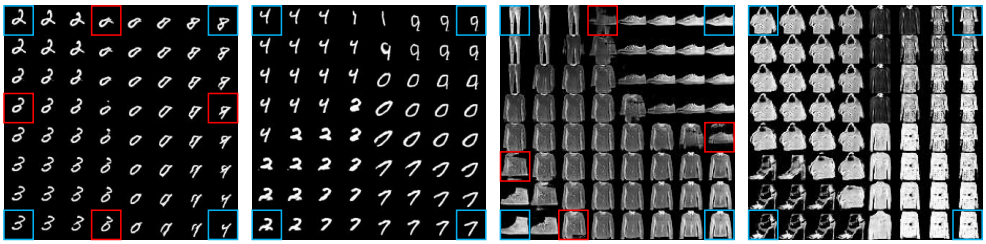
ABSTRACT

We propose an approach to address two undesired properties of unsupervised GANs. First, since GANs use only a continuous latent distribution to embed multiple classes or clusters of a dataset, GANs often do not correctly handle the structural discontinuity between disparate classes in a latent space. Second, discriminators of GANs easily forget about past generated samples by generators, incurring instability during adversarial training. We argue that these two infamous problems of unsupervised GANs can be largely alleviated by a *memory structure* to which both generators and discriminators can access. Generators can effectively store a large amount of training samples that are needed to understand the underlying cluster distribution, which eases the structure discontinuity problem. At the same time, discriminators can memorize previously generated samples, which mitigate the forgetting problem. We propose a novel end-to-end GAN model named *memoryGAN*, that involves a memory network that can be trained in an unsupervised manner, and integrable to many existing models of GANs. With evaluations on multiple datasets including Fashion-MNIST, CelebA, CIFAR10, and Chairs, we show that our model is probabilistically interpretable, and generates image samples of high visual fidelity. We also show that our memoryGAN also achieves the state-of-the-art inception scores among unsupervised GAN models on the CIFAR10 dataset, without additional tricks or weaker divergences.

1 INTRODUCTION

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are one of emerging branches of unsupervised models for deep neural networks. It consists of two neural networks named *generator* and *discriminator* that compete each other in a zero-sum game framework. GANS have been successfully applied to multiple generation tasks, including image syntheses (*e.g.* (Reed et al., 2016b; Radford et al., 2016; Zhang et al., 2016a)), image super-resolution (*e.g.* (Ledig et al., 2017; Snderby et al., 2017)), image colorization (*e.g.* (Zhang et al., 2016b)), to name a few. Despite such remarkable progress, GANs are notoriously difficult to train. Currently, such training instability problems have mostly tackled by finding better distance measures (*e.g.* (Li et al., 2015; Nowozin et al., 2016; Arjovsky & Bottou, 2017; Arjovsky et al., 2017; Gulrajani et al., 2017; Warde-Farley & Bengio, 2017; Mroueh et al., 2017; Mroueh & Sercu, 2017)) or regularizers (*e.g.* (Salimans et al., 2016; Metz et al., 2017; Che et al., 2017; Berthelot et al., 2017)).

In this paper, we address two undesired properties of unsupervised GANs that cause instability during training. The first one is that GANs use a unimodal continuous latent space (*e.g.* Gaussian distribution), and thus fail to handle structural discontinuity between different classes or clusters. It partly attributes to the infamous mode collapsing problem. For example, GANs embed both *building* and *cats* into a common continuous latent distribution, even though there is no intermediate structure between them. Hence, even a perfect generator would produce unrealistic images for some latent codes that reside in intermediate regions of two disparate classes. Fig.1 (a,c) visualize this problem with examples of affine-transformed MNIST and Fashion-MNIST datasets. There always exist latent space regions that cause unrealistic samples (red boxes) between different classes (blue boxes at the corners). Another problem is the forgetting behavior of the discriminator about past synthesized samples by the generator, during adversarial training of GANs. The catastrophic forgetting has explored in deep neural network research, such as (Kirkpatrick et al., 2016; Kemker et al., 2017). In the context of GANs, Shrivastava et al. (2017) address the problem that the discriminator often



(a) DCGAN (b) MemoryGAN (c) DCGAN (d) MemoryGAN

Figure 1: Evidential visualization of structural discontinuity in a latent space with examples of affine-MNIST in (a)–(b) and Fashion-MNIST in (c)–(d). In each set, we first generate four images by randomly sampling four latent variables (z's for DCGAN and (z, c)'s for MemoryGAN), which are shown at the four corners in blue boxes. We then generate 64 images by interpolating latent variables between them. Note that c is a memory index, and thus it is meaningless to directly interpolate over c. Therefore, for the four sampled c's, we take their key values K_c and interpolate over K_c in the query space. For each interpolated vector $\hat{K}_c$, we then find the memory slot $c = \mathrm{argmax}_{c \in \{1,\dots,N\}} p(c|\hat{K}_c)$, *i.e.* the memory index whose posterior is the highest with respect to $\hat{K}_c$. (a,c) In unsupervised DCGAN, unrealistic samples (depicted in red boxes) always exist in some interpolated regions, because different classes are embedded in a unimodal continuous latent space. (b,d) In our MemoryGAN, the samples gradually change, but no structural discontinuity occurs.

focuses only on the latest input images. Since the loss functions of two networks depend on each other's performance, such forgetting behavior results in serious instability, including causing the divergence of adversarial training, or making the generator re-introduce artifacts that the discriminator has forgotten.

In this work, we claim that a simple memorization module can effectively mitigate both infamous instability issues of unsupervised GANs. First, to ease the structure discontinuity problem, the memory can store a large amount of training samples that help the generator better understand the underlying class or cluster distribution of a dataset. Fig.1 (b,d) illustrate some examples that key values of our memory network successfully learn implicit image clusters. Thus, we can separate discrete cluster sampling with continuous latent variable sampling in the generative process, which helps reduce the structural discontinuity within a continuous latent space. Second, to mitigate the forgetting problem, the memory can memorize previously generated samples by the generator, including even rare ones. It makes the discriminator train robust to temporal proximity of specific batch inputs.

Based on these intuitions, we propose a novel end-to-end GAN model named *memoryGAN*, that involves a life-long memory network to which both generator and discriminator can access. Our model can learn a highly multi-modal latent space of data in an unsupervised way, without additional tricks or weaker divergences. Moreover, the proposed memory structure is orthogonal to generator and discriminator design, and thus is integrable to many existing variants of GANs. In our experiments, we show that our model can generate image samples of competitive visual fidelity on several benchmark datasets, including Fashion-MNIST (Xiao et al., 2017), CelebA (Liu et al., 2015), CIFAR10 (Krizhevsky, 2009), and Chairs (Aubry et al., 2014). Our memoryGAN also achieves the state-of-the-art inception scores among unsupervised GAN models on the CIFAR10 dataset.

We summarize the contributions of this paper as follows.

- We propose *MemoryGAN* as a novel unsupervised framework to resolve the two key instability issues of existing GAN training, the structural discontinuity in a latent space and the forgetting problem of GANs. To the best of our knowledge, our model is a first attempt to incorporate a memory network module with an unsupervised GAN model.
- In our experiments, we show that our model is probabilistically interpretable by visualizing data likelihoods, learned categorical priors, and posterior distributions of memory slots. We qualitatively show that our model generates realistic image samples of high visual fidelity on CIFAR10, CelebA, and Chairs dataset. We also outperform existing unsupervised GAN models in terms of inception scores on the CIFAR10 dataset. We plan to make public source codes and pretrained model parameters.

2 RELATED WORK

Memory networks. Augmenting neural networks with memory has been heavily studied recently (Bahdanau et al., 2014; Graves et al., 2014; Sukhbaatar et al., 2015). In these early memory models, computational requirement necessitates the memory size to be small. Some networks such as (Weston et al., 2014; Xu et al., 2016) use large-scale memory but its size is fixed prior to training. Recently, Kaiser et al. (2017) extend (Santoro et al., 2016; Rae et al., 2016) and propose a large-scale life-long memory network, which does not need to be reset during training. It exploits nearest-neighbor algorithms for efficient memory lookup, and thus scales to a large memory size. Unlike previous approaches, our memory network in the discriminator is designed based on the mixture model of Von Mises-Fisher distribution (vMF) and incremental Expectation-Maximization (EM) algorithm. We will further specify the uniqueness of read, write and sampling mechanism of our memory network in section 3.1.

There have been a few models that strengthen the memorization capability of generative models. Li et al. (2016) use additional trainable parameter matrices as a form of memory for deep generative models, but they do not consider the GAN framework. Arici & Celikyilmaz (2016) use RBMs as an associative memory, to transfer some knowledge of the discriminator's feature distribution to the generator. However, they do not address the two problems of our interest, the structural discontinuity and the forgetting problem; as a result, their memory structure is completely different with ours.

Structural discontinuity in a latent space. Previous works have addressed this problem by splitting the latent vector into multiple subsets, and allocating a separate distribution for each data cluster. That is, many conditional GANs concatenate random noises with vectorized external information like class labels or text embedding, which serve as cluster identifiers, (Mirza & Osindero, 2014; Gauthier, 2015; Reed et al., 2016b; Zhang et al., 2016a; Reed et al., 2016a; Odena et al., 2017; Dash et al., 2017). In an unsupervised setting, however, such networks are not applicable because they require supervision of conditional information. Some image editing GANs and cross-domain transfer GANs extract the content information from an input data using an auxiliary encoder network (Yan et al., 2016; Perarnau et al., 2016; Antipov et al., 2017; Denton et al., 2016; Lu et al., 2017; Zhang et al., 2017; Isola et al., 2017; Taigman et al., 2017; Kim et al., 2017; Zhu et al., 2017). However, they require additional auxiliary encoder network, and transform only a given image rather than generating from the entire sample space. On the other hand, MemoryGAN learns clusters of data without supervision, additional encoder network, or a source image to edit from.

InfoGAN (Chen et al., 2016) is another important unsupervised GAN framework, although there are two key differences between InfoGAN and our MemoryGAN. First, InfoGAN implicitly learns the latent cluster information of data into model parameters, while MemoryGAN explicitly maintains the information about the whole training set using a life-long memory network. Thus, MemoryGAN keeps track of current cluster information stably and flexibly without suffering from forgetting old samples. Second, MemoryGAN explicitly offers various distributions like priors, conditional likelihoods, and marginal likelihoods, unlike InfoGAN. Such interpretability is useful for designing or training the models.

Fogetting problem. It is a well-known problem that the discriminator of GANs is prone to forgetting past samples that the generator synthetizes. Multiple studies such as (Salimans et al., 2016; Shrivastava et al., 2017) address the forgetting problem in GANs and make a consensus on the need for memorization mechanism. In order for GANs to be less sensitive to temporal proximity of specific batch inputs, Salimans et al. (2016) add a regularization term of the ℓ_2-distance between current network parameters and the running average of previous parameters, to prevent the network from revisiting previous parameters during training. Shrivastava et al. (2017) modify the adversarial training algorithm to involve a buffer of synthetic images generated by the previous generator network. Instead of adding a regularization term or modifying GAN algorithm, we explicitly increase the model's memorization capacity by introducing a life-long memory into the discriminator.

3 THE MEMORYGAN

Fig.2 shows the proposed *MemoryGAN* architecture, which includes a novel discriminator network named *Discriminative Memory Network* (DMN) and a generator network as *Memory Conditional Generative Network* (MCGN). We describe DMN and MCGN in section 3.1–3.3, and then discuss how our MemoryGAN can resolve the two instability issues in section 3.4.

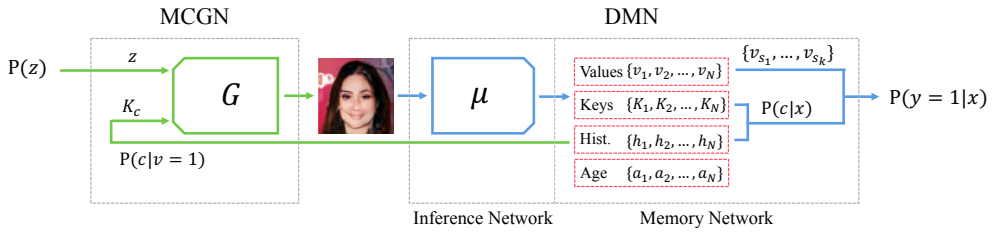


Figure 2: The memoryGAN architecture. The Discriminative Memory Network (DMN) consists of an inference network and a memory module. The Memory Conditional Generative Network (MCGN) samples a memory slot index c from a categorical prior distribution over memory slots, and use the mean direction of corresponding mixture component K_c as conditional cluster information.

3.1 The Discriminative Memory Network

The DMN consists of an inference network and a memory network. The inference network μ is a convolutional neural network (CNN), whose input is a datapoint $x \in \mathbb{R}^D$ and output is a normalized query vector $q = \mu(x) \in \mathbb{R}^M$ with $\|q\| = 1$. Then the memory module takes the query vector as input and decides whether x is real or fake: $y \in \{0, 1\}$.

We denote the memory network by four tuples: $\mathcal{M} = (K, v, a, h)$. $K \in \mathbb{R}^{N \times M}$ is a memory key matrix, where N is the memory size (*i.e.* the number of memory slots) and M is the dimension. $v \in \{0, 1\}^N$ is a memory value vector, $a \in \mathbb{R}^N$ is a vector that tracks the age of items stored in each memory slot, and $h \in \mathbb{R}^N$ is the slot histogram, where each h_i can be interpreted as the effective number of data points that belong to i-th memory slot. We initialize them as follows $K = 0$, $a = 0$, $h = 10^{-5}$, and v from a flat categorical distribution.

Our memory network partly borrows some mechanisms of the *life-long memory network* (Kaiser et al., 2017), that is free to increase a memory size, and has no need to be reset during training. Our memory also uses the k-nearest neighbor indexing for efficient memory lookup, and adopts the Least Recently Used (LRU) scheme for memory update. Nonetheless, there are several novel features of our memory structure as follows. First, our method is probabilistically interpretable; we can easily compute the data likelihood, categorical prior and posterior distribution of memory indices. Second, our memory learns the approximate distribution of a query by maximizing likelihood using an incremental EM algorithm. Third, our memory is optimized from the GAN loss, instead of the memory loss proposed in (Kaiser et al., 2017). Fifth, our method tracks the slot histogram to determine the degree of contributions of each sample to the slot and sampling purpose.

Below we first discuss how to compute the discriminative probability $p(y|x)$ for a given real or fake sample x at inference, using the memory and the learned inference network (section 3.1.1). We then explain how to update the memory during training (section 3.1.2).

3.1.1 The Discriminative Output

For a given sample x, we first find out which memory slots should be referred for computing the discriminative probability. We use $c \in \{1, 2, \ldots, N\}$ to denote a memory slot index. We represent the posterior distribution over memory indices using a Von Mises-Fisher (vMF) mixture model as

$$p(c = i|x) = \frac{p(x|c = i)p(c = i)}{\sum_{j=1}^{N} p(x|c = j)p(c = j)} = \frac{\exp(\kappa K_i^T x)p(c = i)}{\sum_{j=1}^{N} \exp(\kappa K_j^T x)p(c = j)} \tag{1}$$

where the likelihood $p(x|c = i) = C(\kappa)\exp(\kappa K_i^T x)$ with $\kappa = 1$ of a constant concentration parameter. The normalizing constant $C(\kappa)$ is eliminated since it is a common denominator. Note that the vMF is, in effect, equivalent to Gaussian distribution that is defined on a unit sphere and properly normalized.

The categorical prior of the memory indices, $p(c)$, is directly obtained by nomalizing the slot histogram: $p(c = i) = \frac{h_i + \beta}{\sum_{j=1}^{N}(h_j + \beta)}$, where $\beta (= 10^{-8})$ is a small smoothing constant for numerical stability. By using $p(y = 1|c = i, x) = v_i$ and Eq.(1), we can estimate the discriminative probability

$p(y=1|x)$ by marginalizing the joint probability $p(y=1,c|x)$ with respect to c:

$$p(y=1|x) = \sum_{i=1}^{N} p(y=1|c=i,x)p(c=i|x) = \sum_{i=1}^{N} v_i p(c=i|x) = \mathbb{E}_{i\sim p(c|x)}[v_i]. \tag{2}$$

However, in practice, it is not scalable to exhaustively sum over the whole memory of size N for each x; we approximate the predictive probability by using a subset of keys. We choose indices of k slots $S = \{s_1, ..., s_k\}$ that have k-largest posterior probability (*e.g.* we use $k = 128 \sim 256$):

$$S = \underset{c_1,\dots,c_k}{\operatorname{argmax}}\, p(c|x) = \underset{c_1,\dots,c_k}{\operatorname{argmax}}\, p(c,x) = \underset{c_1,\dots,c_k}{\operatorname{argmax}} \exp(\kappa K_c^T)(h_c + \beta) \tag{3}$$

For efficiency, we use approximated discriminative output $p(y|x) \approx \sum_{i\in S} v_i \frac{p(x|c=i)p(c=i)}{\sum_{j\in S} p(x|c=j)p(c=j)}$.

We clip $p(y|x)$ into $[\epsilon, 1-\epsilon]$ with a small constant $\epsilon = 0.001$ for numerical stability.

3.1.2 Memory Update

The memory keys and values are updated during the training. We adopt both a conventional memory update mechanism and an incremental EM algorithm. We denote a training sample x and its label y that is 1 for real and 0 for fake. For each x, we first find k-nearest slots S_y as done in Eq.(3), except that we here use the conditional posterior $p(c|x, v_c = y)$ instead of $p(c|x)$, in order to perform the EM algorithm using only the slots that belong to the same class with y. Then we update the memory in two different ways, according to whether $\{v_{s_1}, \dots, v_{s_k}\}$ contains the correct label y or not. If there is no correct label in the slots of S, we find the oldest memory slot by $n_a = \operatorname{argmax}_{i\in\{1,\dots,N\}} a_i$, and set to $K_{n_a} \leftarrow q = \mu(x)$, $v_{n_a} \leftarrow y$, $a_{n_a} \leftarrow 0$, and $h_{n_a} \leftarrow \frac{1}{N}\sum_{i=1}^{N} h_i$. If S contains a correct value, the memory keys are updated using the following modified incremental EM algorithm for T iterations. In the expectation step, we compute posterior $\gamma_i^t = p(c_i|x)$ for $i \in S_y$, by applying previous keys $\hat{K}_i^{t-1}$ and $\hat{h}_i^{t-1}$ to Eq.(1). In the maximization step, we update the required sufficient statistics.

$$\hat{h}_i^t \leftarrow \hat{h}_i^{t-1} + \gamma^t - \gamma^{t-1}, \quad \hat{K}_i^t \leftarrow \hat{K}_i^{t-1} + \frac{\gamma^t - \gamma^{t-1}}{\hat{h}_i^t}(q_i - \hat{K}_i^t) \tag{4}$$

where $t \in 1, ..., T$, $\gamma^0 = 0$, $\hat{K}_i^1 = K_i$, $\hat{h}_i^1 = \alpha h_i$ and $\alpha = 0.5$. After T iterations, we update the slots of S_y by $K_i \leftarrow \hat{K}_i^t$ and $h_i \leftarrow \hat{h}_i^t$. The decay rate α controls how much it exponentially reduces the contribution of old queries to the slot position of the mean directions of mixture components. The α value is important for the performance, give than old queries that were used to update a key previously are unlikely to be fit to the current mixture distribution, since the inference network μ itself updates during the training. Finally, it is worth noting that this memory updating mechanism is orthogonal to any adversarial training algorithm, because it is performed separately while the discriminator is updated. Moreover, adding our memory module does not affects the running speed of the model at test time, since the memory is updated only at training time.

3.2 The Memory-Conditional Generative Network

Our generative network MCGN is largely based on the conditional GANs of InfoGAN (Chen et al., 2016), However, one key difference is that the MCGN synthesizes a sample not only conditioned on a random noise vector $z \in \mathbb{R}^{D_z}$, but also on conditional memory information. That is, in addition to sample z from a Gaussian, the MCGN samples a memory index i from $P(c=i|v_c=1) = \frac{h_i v_i}{\sum_j^N h_j v_j}$, which reflects the exact appearance frequency of the memory cell i within real data. Finally, the generator synthesizes a fake sample from the concatenated representation $[K_i, z]$, where K_i is a key value vector of the memory index i.

Unlike other conditional GANs, MCGN requires neither additional annotation nor any external encoder network. Instead, MCGN makes use of conditional memory information that DMN learns in an unsupervised way. Recall that *DMN* learns the vMF mixture memory with only the query representation $q = \mu(x)$ of each sample x and its indicator y.

Finally, we summarize the training algorithm of MemoryGAN in Algorithm 1.

Algorithm 1 Training algorithm of *MemoryGAN*. ϕ is parameters of discriminator, θ is parameters of generator. $\alpha = 0.5, \eta = 2 \cdot 10^{-4}$.

for number of training iterations **do**
  Sample a minibatch of examples x from training data.
  Sample a minibatch of noises $z \sim p(z)$ and memory indices $c \sim p(c|v = 1)$.
  Update by the gradient descent $\phi \leftarrow \phi - \eta \nabla_\phi L_D$
  Find S_y for each data in the minibatch
  Initialize $\gamma_s^0 \leftarrow 0, \hat{h}_s^0 \leftarrow \alpha h_s$ and $\hat{K}_s^0 \leftarrow K_s$ for $s \in S_y$
  for number of EM iterations **do**
    Estimate posterior γ_s^t for $s \in S_y$
    $\hat{h}_s^t \leftarrow \hat{h}_s^{t-1} + \gamma_s^t - \gamma_s^{t-1}$ for $s \in S_y$
    $\hat{K}_s^t \leftarrow \hat{K}_s^{t-1} + \frac{\gamma_s^t - \gamma_s^{t-1}}{\hat{h}_s^t}(q_i - \hat{K}_s^t)$ for $s \in S_y$
  Update vMF mixture model $h_{s_y} \leftarrow \hat{h}_{s_y}^T, K_{s_y} \leftarrow \hat{K}_{s_y}^T$ for $s_y \in S_y$
  Sample a minibatch of noises $z \sim p(z)$ and memory indices $c \sim p(c|v = 1)$.
  Update by gradient descent $\theta \leftarrow \theta - \eta \nabla_\theta L_G$

3.3 The Objective Function

Our objective is based on that of InfoGAN (Chen et al., 2016), which maximizes the mutual information between a subset of latent variables and observations. We add another mutual information loss term between K_i and $G(z, K_i)$, to ensure that the structural information is consistent between a sampled memory information K_i and a generated sample $G(z, K_i)$ from it:

$$I(\mathsf{K_i}; \mathsf{G(z, K_i)}) \geq \mathsf{H(K_i)} - \hat{I} - \log \mathsf{C}(\kappa) \tag{5}$$

where $\hat{I}$ is the expectation of negative cosine similarity $\hat{I} = -E_{x \sim G(z, K_i)}[\kappa K_i^T \mu(x)]$. We defer the derivation of Eq.(5) to Appendix A.1. Finally, the modified GAN objective can be written with the lower bound of mutual information, with a hyperparameter λ (*e.g.* we use $\lambda = 2 \cdot 10^{-6}$):

$$\mathcal{L}_D = -E_{x \sim p(x)}[\log D(x)] - E_{(z,c) \sim p(z,c)}[\log(1 - D(G(z, K_i)))] + \lambda \hat{I}. \tag{6}$$

$$\mathcal{L}_G = E_{(z,c) \sim p(z,c)}[\log(1 - D(G(z, K_i)))] + \lambda \hat{I}. \tag{7}$$

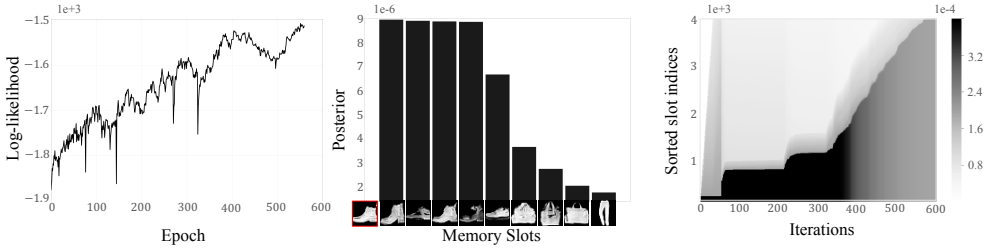
3.4 How does *memoryGAN* Mitigate the Two Instability Issues?

Our memoryGAN implicitly learns the joint distribution $p(x, z, c) = p(x|z, c)p(z)p(c)$, for which we assume the continuous latent variable z and the discrete memory latent variable c are independent. This assumption reflects our intuition that when we synthesize a new sample (*e.g.* an image), we separate the modeling of its class or cluster (*e.g.* an object category) from the representation of other image properties, attributes, or styles (*e.g.* the rotation and translation of the object). Such separation of modeling duties can largely alleviate the structural discontinuity problem in our model. For data synthesis, we sample K_i and z as an input to the generator. K_i represents one of underlying clusters of training data that the DMN learns in a form of key values. Thus, z does not need to care about the class discontinuity but focus on the attributes or styles of synthesized samples.

Our model suffers less from the forgetting problem, intuitively thanks to the explicit memory. The memory network memorizes high-level representation of clusters of real and fake samples in a form of key value vectors. Moreover, the DMN allocates memory slots even for infrequent real samples and maintains the learned slot histogram h nonzero for them (*i.e.* $p(c|v_c = 1) \neq 0$). It opens a chance of sampling from rare ones for a synthesized sample, although their chance could be low.

4 Experiments

We present examples of probabilistic interpretation of MemoryGAN to understand how it works in section 4.1. We show both qualitative and quantitative results of image generation in section 4.2 on



(a) Test Log-Likelihood (b) Posterior (c) Prior

Figure 3: Some probabilistic interpretation of MemoryGAN. (a) The biased log-likelihood $\tilde{p}(X|y = 1)$ of $10,000$ real test images (*i.e.* unseen for training) according to training epochs. (b) An example of posteriors $p(c|x)$ of nine randomly selected memory slots for a given input image shown at the left-most column in the red box. (c) The categorical prior distribution $p(c)$ along training iterations. We sort N slots and remove slots of probabilities within $[0, 0.0004]$ for readability.

Table 1: Comparison of CIFAR10 inception scores between state-of-the-art unsupervised GAN models. Our memoryGAN achieves the highest score.

Method	Score	Objective function	Auxiliary net.
ALI (Dumoulin et al., 2017)	5.34 ± 0.05	GAN	Inference Net.
BEGAN (Berthelot et al., 2017)	5.62	Energy based	Decoder Net.
DCGAN (Radford et al., 2016)	6.54 ± 0.067	GAN	-
Improved GAN (Salimans et al., 2016)	6.86 ± 0.06	GAN+ M.D. + H.A.	-
EGAN-Ent-VI (Dai et al., 2017)	7.07 ± 0.10	Energy based	Decoder Net.
DFM (Warde-Farley & Bengio, 2017)	7.72 ± 0.13	Energy based	Decoder Net.
WGAN-GP (Gulrajani et al., 2017)	7.86 ± 0.07	Wasserstein+GP	-
Fisher GAN (Mroueh et al., 2017)	7.90 ± 0.05	Fisher IPM	-
MemoryGAN	$\mathbf{8.04 \pm 0.13}$	GAN + M.I.	-

Fashion-MNIST (Xiao et al., 2017), CelebA (Liu et al., 2015), CIFAR10 (Krizhevsky, 2009), and Chairs (Aubry et al., 2014). We perform ablation experiments to demonstrate the usefulness of key components of our model in section 4.3. We present more experimental results in Appendix B.

We use DCGAN (Radford et al., 2016) for our inference network and generator, except the experiments on CIFAR10 and CelebA, for which we use the one similar to WGAN-GP ResNet (Gulrajani et al., 2017). We use minibatches of size 64, a learning rate of $2 \cdot 10^{-4}$, and Adam (Kingma & Ba, 2014) optimizer for all experiments.

4.1 Probabilistic Interpretation

Fig.3 presents some probabilistic interpretation of MemoryGAN to gain intuitions of how it works. We train MemoryGAN using Fashion-MNIST (Xiao et al., 2017), with a hyperparameter setting of $z \in \mathbb{R}^2$, the memory size $N = 4,096$, the key dimension $M = 256$, the number of nearest neighbors $k = 128$, and $\lambda = 0.01$. Fig.3(a) shows the biased log-likelihood $\tilde{p}(X|y = 1)$ of $10,000$ real test images (*i.e.* unseen for training), while learning the model up to reaching its highest inception score. Although the inference network keeps updated during adversarial training, the Von Mises-Fisher mixture model based memory successfully tracks its approximate distribution; as a result, the likelihood keeps increasing. Fig.3(b) show posteriors $p(c|x)$ of nine randomly selected memory slots for a a random input image (shown at the left-most column in the red box). We observe that the memory slots are properly activated, in that the memory slots with highest posteriors indeed include the information of the same class and similar styles like(*e.g.* shapes or colors) to the input image. The slot images below the bars are drawn by the generator using its slot key value V_i and a random noise z. Finally, Fig.3(c) shows categorical prior distribution $p(c)$ of memory slots. At the initial state, the inference network is not well trained, so it uses only a small number of slots even for different examples. As training proceeds, the prior spreads along N memory slots, which means the Von Mises-Fisher mixture model based memory fully utilizes the memory slots to distribute training samples according to the clusters of the data distribution.

(a) CIFAR10 (32 × 32) (b) CelebA (64 × 64) (c) Chair (64 × 64)

Figure 4: Image samples generated from (a) CIFAR10, (b) CelebA, and (c) Chairs dataset. Top four rows show successful examples, while the bottom two show near-miss or failure cases.

Table 2: Variation of inception scores when removing one of key components of our memoryGAN on CIFAR10, affine-MNIST and Fashion-MNIST datasets.

Variants	CIFAR10	affine-MNIST	Fashion-MNIST
MemoryGAN	**8.04 ± 0.13**	**8.60 ± 0.02**	**6.39 ± 0.29**
(– EM)	6.67 ± 0.17	8.17 ± 0.04	6.05 ± 0.28
(– MCGN)	6.39 ± 0.02	8.11 ± 0.03	6.02 ± 0.27
(– Memory)	5.35 ± 0.17	8.00 ± 0.01	5.82 ± 0.19

4.2 Image Generation Performance

We perform both quantitative and qualitative evaluations on our model's ability to generate realistic images. Table 1 shows that our model outperforms state-of-the-art unsupervised GAN models in terms of inception scores with no additional divergence measure or auxiliary network on the CIFAR10 dataset. We carry out no other hyperparameter tuning for our method, except for using Layer Normalization(Ba et al., 2016) instead of Batch Normalization and using ELU activation function instead of ReLU and Leaky ReLU. Unfortunately, except CIFAR10, there are no reported inception scores that shares a standard classifier for other datasets, on which we do not compare quantitatively.

Fig.4 shows generated samples on CIFAR10, CelebA, and Chairs dataset, on which our model achieves competitive visual fidelity. We observe that more regular the shapes of classes are, the more realistic generated images are. For example, *car* or *chair* images are more realistic, while the faces of *dogs* and *cats*, *hairs*, and *sofas* are less. MemoryGAN also has failure samples in the whole latent space as shown in 4. Since our approach is completely unsupervised, sometimes a single memory slot may include similar images from different classes. It causes failure cases. Nevertheless, significant proportion of memory slots of MemoryGAN contain similar shaped single class, which leads much better performance than existing unsupervised GAN models. We set $N = 16,384$, $M = 512$, $k = 256$, and $\lambda = 1e-6$ for all datasets. We use $z \in \mathbb{R}^{128}$ for DCGAN and $z \in \mathbb{R}^{16}$ for MemoryGAN.

4.3 Ablation Study

We perform a series of ablation experiments to demonstrate that key components of MemoryGAN indeed improve the inception scores. The variants are as follows. (i) (– EM) is our MemoryGAN but adopts the memory updating rule of Kaiser et al. (2017) (*i.e.* $K^t \leftarrow \frac{K^{t-1}+q}{\|K^{t-1}+q\|}$). (ii) (– MCGN) removes the slot-sampling process from the generative network, which equals to the DCGAN that uses DMN as discriminator. (iii) (– Memory) is equivalent to the original DCGAN. Results in 2 show that each of proposed components of our MemoryGAN makes significant contribution to its outstanding image generation performance. As expected, without the memory network, the performance is the worst over all datasets.

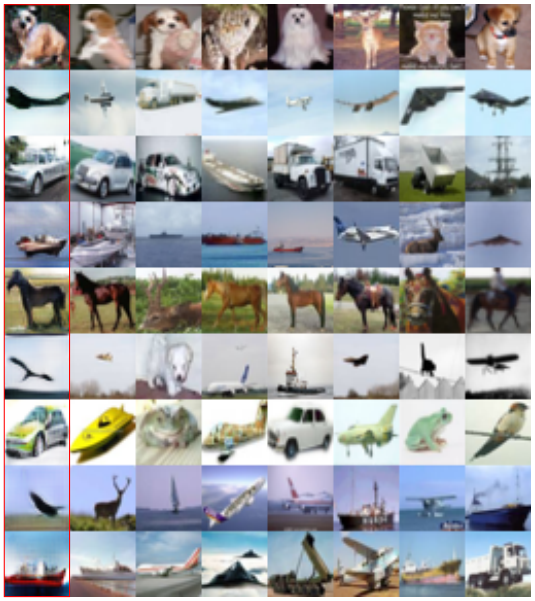


Figure 5: Examples of generalization ability of MemoryGAN. For each sample produced by MemoryGAN (in the left-most column), the seven nearest images in the training set are shown in the following columns. The MemoryGAN indeed creates novel images rather than memorizing and retrieving the training images.

4.4 Generalization Examples

We present examples of generalization ability of MemoryGAN in Fig.5, where for each sample produced by MemoryGAN (in the left-most column), the seven nearest images in the training set are shown in the following columns. We represent each image x by a normalized $q = \mu(x)$, and compute the cosine similarity between images. Apparently, our MemoryGAN generates novel images rather than merely memorizing and retrieving the images in the training set.

4.5 Computational Overhead

Our memoryGAN can resolve the two problems of unsupervised GANs with a mild increase of training time. For example of CIFAR10, we measure the training time per epoch for MemoryGAN with 4,128K parameters and DCGAN with 2,522K parameters, which are 135 sec and 124 sec, respectively. It means MemoryGAN is only 8.9% slower than DCGAN for training, even with a scalable memory module. At test time, since only generator is used, there is no time difference between MemoryGAN and DCGAN.

5 Conclusion

We proposed a novel end-to-end unsupervised GAN model named *memoryGAN*, that effectively learns a highly multi-modal latent space and does not suffer from forgetting. Empirically, we showed that our model achieved the state-of-the-art inception scores among unsupervised GAN models on the CIFAR10 dataset. We also demonstrated that our model generates realistic image samples of high visual fidelity on Fashion-MNIST, CIFAR10, CelebA, and Chairs datasets. As an interesting future work, we can extend our model for few-shot generation that synthesizes rare samples.

References

Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. *arXiv:1702.01983*, 2017.

Tarik Arici and Asli Celikyilmaz. Associative Adversarial Networks. *NIPS*, 2016.

Martin Arjovsky and Lon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *ICLR*, 2017.

Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein GAN. *ICLR*, 2017.

Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignment Using a Large Dataset of CAD Models. *CVPR*, 2014.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473*, 2014.

David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. *arXiv:1703.10717*, 2017.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode Regularized Generative Adversarial Networks. *ICLR*, 2017.

Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *NIPS*, 2016.

Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating Energy-based Generative Adversarial Networks. *ICLR*, 2017.

Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Marcus Liwicki, and Muhammad Zeshan Afzal. TAC-GAN - text conditioned auxiliary classifier generative adversarial network. *arXiv:1703.06412*, 2017.

Emily L. Denton, Sam Gross, and Rob Fergus. Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks. *arXiv:1611.06430*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially Learned Inference. *ICLR*, 2017.

Jon Gauthier. Conditional generative adversarial networks for convolutional face generation. *Stanford University*, 2015.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative Adversarial Nets. *NIPS*, 2014.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. *arXiv:1410.5401*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv:1704.00028*, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR*, 2017.

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to Remember Rare Events. *ICLR*, 2017.

Ronald Kemker, Angelina Abitino, Marc McClure, and Christopher Kanan. Measuring Catastrophic Forgetting in Neural Networks. *arXiv:1708.02072*, 2017.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *arXiv:1703.05192*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv:1612.00796*, 2016.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *arXiv:1412.6980*, 2009.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *CVPR*, 2017.

Chongxuan Li, Jun Zhu, and Bo Zhang. Learning to Generate with Memory. *ICML*, 2016.

Yujia Li, Kevin Swersky, and Richard Zemel. Generative Moment Matching Networks. *ICML*, 2015.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. *ICCV*, 2015.

Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Conditional CycleGAN for Attribute Guided Face Image Generation. *arXiv:1705.09966*, 2017.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. *ICLR*, 2017.

Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784*, 2014.

Youssef Mroueh and Tom Sercu. Fisher GAN. *arXiv:1705.09675*, 2017.

Youssef Mroueh, Tom Sercu, and Vaibhava Goel. McGan: Mean and Covariance Feature Matching GAN. *arXiv:1702.08398*, 2017.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *NIPS*, 2016.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. *ICML*, 2017.

Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible conditional gans for image editing. *NIPS*, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ICLR*, 2016.

Jack W. Rae, Jonathan J. Hunt, Tim Harley, Ivo Danihelka, Andrew W. Senior, Greg Wayne, Alex Graves, and Timothy P. Lillicrap. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes. *NIPS*, 2016.

Scott E. Reed, Zeynep Akata, Bernt Schiele, and Honglak Lee. Learning Deep Representations of Fine-grained Visual Descriptions. *CVPR*, 2016a.

Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative Adversarial Text to Image Synthesis. *ICML*, 2016b.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *NIPS*, 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. One-shot Learning with Memory-Augmented Neural Networks. *arXiv:1605.06065*, 2016.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *CVPR*, 2017.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly Supervised Memory Networks. *arXiv:1503.08895*, 2015.

Casper Kaae Snderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszr. Amortised MAP Inference for Image Super-Resolution. *ICLR*, 2017.

Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised Cross-Domain Image Generation. *ICLR*, 2017.

David Warde-Farley and Yoshua Bengio. Improving Generative Adversarial Networks With Denoising Feature Matching. *ICLR*, 2017.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory Networks. *arXiv:1410.3916*, 2014.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747*, 2017.

Jiaming Xu, Jing Shi, Yiqun Yao, Suncong Zheng, Bo Xu, and Bo Xu. Hierarchical Memory Networks for Answer Selection on Unknown Words. *arXiv:1609.08843*, 2016.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2Image: Conditional Image Generation from Visual Attributes. *ECCV*, 2016.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. *arXiv:1612.03242*, 2016a.

He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image De-raining Using a Conditional Generative Adversarial Network. *arXiv:1701.05957*, 2017.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful Image Colorization. *ECCV*, 2016b.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv:1703.10593*, 2017.

Table 3: Summary of experimental settings.

	affine-MNIST	Fashion-MNIST	CIFAR10	Chair	CelebA
Test sample size	$640,000$	$64,000$	$640,000$	-	-
Test classifier (Acc.)	AlexNet (0.9897)	ResNet (0.9407)	Inception Net.	-	-
Noise dimension	2	2	16	16	16
Key dimension M	256	256	512	512	512
Memory size N	$4,096$	$4,096$	$16,384$	$16,384$	$16,384$
Image size	40×40	28×28	32×32	64×64	64×64
Learning-rate decay	None	None	Linear	Linear	Linear

A Appendix A: More Technical Details

A.1 Derivation of The Objective Function

Our objective is based on that of InfoGAN (Chen et al., 2016), which maximizes the mutual information between a subset of latent variables and observations. We add another mutual information loss term between K_i and $G(z, K_i)$, to ensure that the structural information is consistent between a sampled memory information K_i and a generated sample $G(z, K_i)$ from it:

$$\begin{aligned} I(K_i; G(z, K_i)) &= H(K_i) - H(K_i|G(z, K_i)) \\ &\geq H(K_i) + E_{x\sim G(z,K_i)}[E_{q\sim P(K_i|x)}[-\log Q(q|x)]] \\ &= H(K_i) + E_{x\sim G(z,K_i)}[E_{q\sim P(K_i|x)}[\kappa K_i^T q - \log C(\kappa)]] \end{aligned} \tag{8}$$

where Q is a Von Mises-Fisher distribution $C(\kappa)exp(\kappa K_i^T q)$. Maximizing the lower bound is equivalent to minimizing the expectation term $\hat{I} = -E_{x\sim G(z,K_i)}[E_{q\sim P(K_i|x)}[\kappa K_i^T q]]$. The modified GAN objective can be written with the lower bound of mutual information, with a hyperparameter λ (we use $\lambda = 2 \cdot 10^{-6}$).

$$\begin{aligned} \mathcal{L}_D &= -E_{x\sim p(x)}\left[\log D(x)\right] - E_{(z,c)\sim p(z,c)}[\log(1 - D(G(z, K_i)))] + \lambda \hat{I}. \\ \mathcal{L}_G &= E_{(z,c)\sim p(z,c)}[\log(1 - D(G(z, K_i)))] + \lambda \hat{I}. \end{aligned} \tag{9}$$

B Appendix B: More Experimental Results

B.1 Experimental Settings

We summarize the experimental setting used for each dataset in Table 3. The test sample size indicates the number of samples that we use to evaluate inception scores. For Chair and CelebA dataaset, we do not compute the inception scores. The test classifier indicates which image classifier is used for the inception scores. For affine-MNIST and Fashion-MNIST, we use a simplified AlexNet and ResNet, respectively. The noise dimension is the dimension of z, and the image size indicates the height and width of training images. For CIFAR10, Chair, and CelebA, we linearly decay the learning rate: $\alpha^t = \alpha^0 \cdot (1 - \gamma^{-1} \cdot t/1,000)$, where t is the iteration, and γ is the number of minibatches per epoch.

B.2 More interpolation Examples for Visualizing Structure in a Latent Space

We present more interpolation examples similar to Fig.1. We use Fashion-MNIST (Xiao et al., 2017) and affine transformed MNIST, in which we randomly rotate MNIST images by 20° and randomly place it in a square image of 40 pixels. First, we randomly fix a memory slot c, and randomly choose four noise vectors $\{z_1, \ldots, z_4\}$. Then, we interpolate the four noise vectors to extract intermediate noise vectors $\hat{z} = (1 - e)z_i + ez_j$, where $i, j \in \{1, 2, 3, 4\}$ and $e \in (0, 1)$, and visualize the generated samples $G(\hat{z}, c)$. Fig.6 shows the results. We set memory size $N = 4,096$, the key dimension $M = 256$, the number of nearest neighbors $k = 128$, and $\lambda = 0.01$ for both datasets. We use $z \in \mathbb{R}^{32}$ for *DCGAN* and $z \in \mathbb{R}^2$ for *MemoryGAN*. In both datasets, the *DMN* successfully learns to separate a latent space into multiple modes, using Von Mises-Fisher mixture memory module. Fig.7 presents results of the same experiments on the Chair dataset. We here show

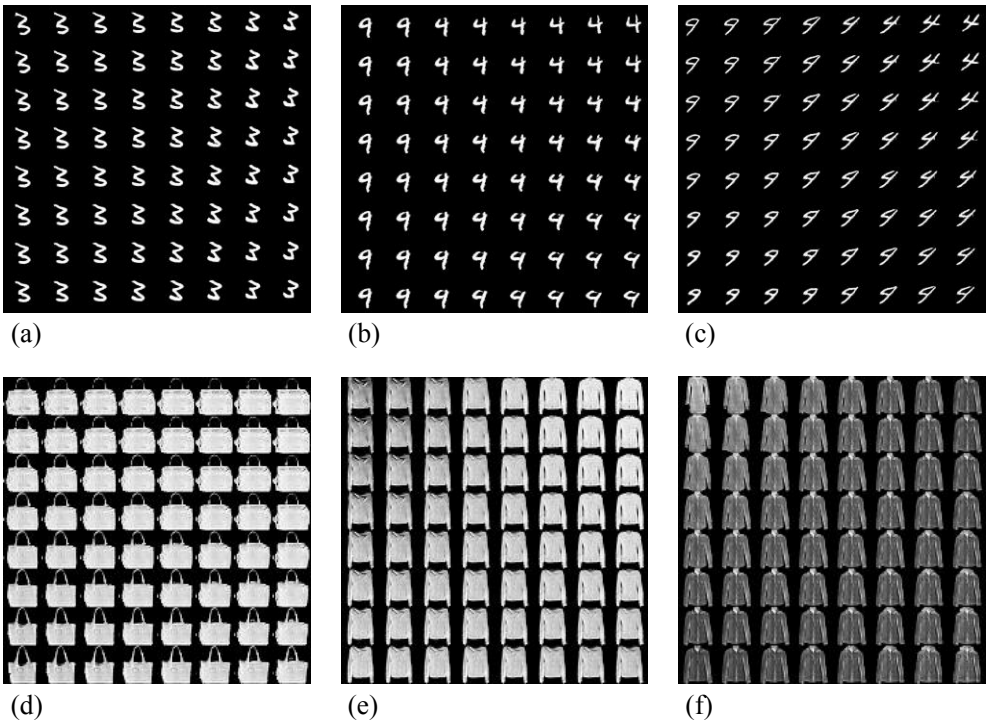


Figure 6: More interpolation examples generated by MemoryGAN. Samples are generated by fixing a discrete latent variable c and sampling continuous latent variable z. Some memory slots contain single stationary structure as shown in (a) and (d), while other slots contain images of the same class, but variation of styles as shown in (b)-(c) and (e)-(f).

an example that includes failure cases in (d), where interestingly some unrealistic chair images are observed as the back of chair rotates from left to right.

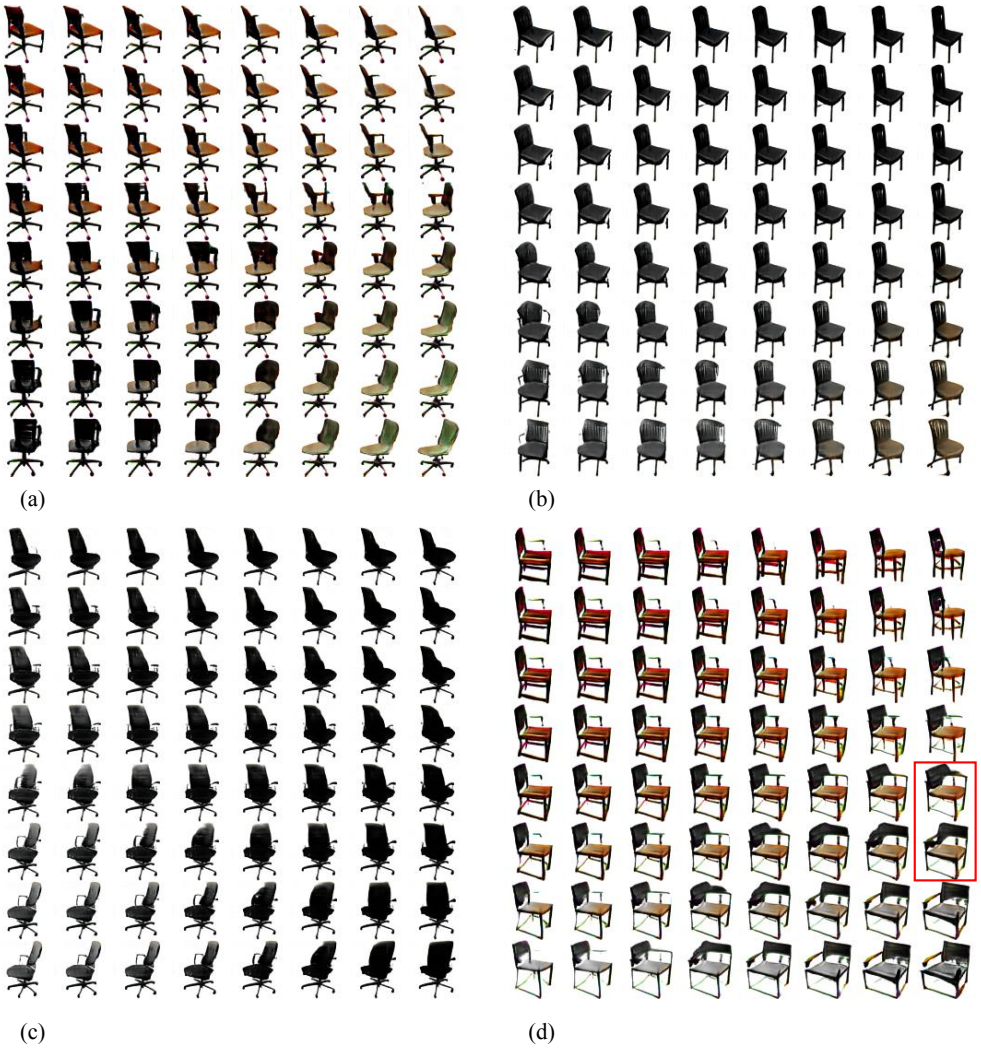


Figure 7: Interpolation examples generated using *MemoryGAN* and Chair dataset. Memory slots contain shape, color and angle of chair (a)-(c). In failure case (d), there are some unrealistic chair images as the back of chair rotates from left to right.