GAN Memory with No Forgetting

Yulai Cong* **Miaoyun Zhao*** **Jianqiao Li** **Sijia Wang** **Lawrence Carin**
Department of Electrical and Computer Engineering, Duke University

Abstract

Seeking to address the fundamental issue of memory in lifelong learning, we propose a GAN memory that is capable of realistically remembering a stream of generative processes with *no* forgetting. Our GAN memory is based on recognizing that one can modulate the "style" of a GAN model to form perceptually-distant targeted generation. Accordingly, we propose to do sequential style modulations atop a well-behaved base GAN model, to form sequential targeted generative models, while simultaneously benefiting from the transferred base knowledge. Experiments demonstrate the superiority of our method over existing approaches and its effectiveness in alleviating catastrophic forgetting for lifelong classification problems.

1 Introduction

Concerning the ability to continually learn new knowledge without forgetting previously learned experiences, lifelong learning (or continual learning) is a long-standing challenge for machine learning and artificial intelligence systems [74, 28, 71, 11, 14, 58]. The main issue associated with lifelong learning is the notorious catastrophic forgetting of deep neural networks [48, 36, 85], *i.e.,* training a model with new information severely interferes with previously learned knowledge.

To alleviate the catastrophic forgetting issue, many methods have been proposed, with most focusing on discriminative/classification tasks [36, 63, 92, 54, 91]. Reviewing existing methods, [75] revealed generative replay (or pseudo-rehearsal) [70, 67, 84, 64, 86] is an effective and general strategy for lifelong learning, with this further supported by [40, 76]. That revelation is anticipated, for if the characteristics of previous data are remembered perfectly (*e.g.,* via realistic generative replay), no forgetting should be expected for lifelong learning. Compared with the coreset idea, that saves representative samples of previous data [54, 63, 11], generative replay has advantages in addressing privacy concerns and remembering potentially more complete data information (via the generative process). However, most existing generative replay methods either deliver blurry generated samples [10, 40] or only work well on simple datasets [40, 76, 40] like MNIST; besides, they often don't scale well to practical situations with high resolution [58] or a long sequence [84], sometimes even with negative backward transfer [93, 80]. Therefore, it's challenging to continually learn a well-behaved generative replay model [40], even for moderately complex datasets like CIFAR10.

We seek a realistic generative replay framework to alleviate catastrophic forgetting; going further, we consider developing a realistic generative memory with growing (expressive) power, believed to be a fundamental building block toward general lifelong learning systems. We leverage the popular GAN [25] setup as the key component of that generative memory, which we term GAN memory, because (i) GANs have shown remarkable power in synthesizing realistic high-dimensional samples [9, 51, 33, 34]; (ii) by modeling the generative process of training data, GANs summarize the data statistical information in the model parameters, consequently also protecting privacy (the original data need not be saved); and (iii) a GAN often generates realistic samples not observed in training data,

*Equal Contribution. Correspondence to: Yulai Cong <yulaicong@gmail.com> and Miaoyun Zhao <miaoyun9zhao@gmail.com>.

Preprint. Under review.

delivering a synthetic data augmentation that potentially benefits better performance of downstream tasks [78, 8, 9, 21, 33, 26, 27]. Distinct from existing methods, our GAN memory leverages transfer learning [6, 16, 46, 89, 57] and (image) style transfer [18, 30, 41]. Its key foundation is a discovery that one can leverage style-transfer techniques [62, 95] to modulate a source generator/discriminator into a powerful generator/discriminator for perceptually-distant target domains (see Section 4.1), with a limited amount of style parameters. Exploiting that discovery, our GAN memory sequentially modulates (and also transfers knowledge from) a well-behaved base/source GAN model to realistically remember a sequence of (target) generative processes with *no* forgetting. The base model could be flexibly selected as an existing GAN model trained on a large-scale public dataset; our experiments will show that flexibility roughly means source and target data should have the same data *type* ($e.g.$, images).

Our GAN memory serves as a solution to the fundamental memory issue of general lifelong learning. In practice, it can be used, for example, as a realistic generative replay to alleviate catastrophic forgetting for challenging downstream tasks with high-dimensional data and a long (and varying) task sequence. Our contributions are as follows.

- Based on FiLM [62] and AdaFM [95], we develop modified variants, termed mFiLM and mAdaFM, to better adapt/transfer the source fully connected (FC) and convolutional (Conv) layers to target domains, respectively. We demonstrate that mFiLM and mAdaFM can be leveraged to modulate the "style" of a source GAN model (including both the generator and discriminator) to form a generative/discriminative model capable of addressing a perceptually-distant target domain.
- Based on the above discovery, we propose our GAN memory, that is endowed with growing (expressive) generative power, yet with *no* forgetting of existing capabilities, by leveraging a limited amount of task-specific style parameters. We analyze the roles played by those style parameters and reveal their further compressibility.
- We generalize our GAN memory to its conditional variant, followed by empirically verifying its effectiveness in delivering realistic synthesized samples to alleviate catastrophic forgetting for challenging lifelong classification tasks.

2 Related work

Lifelong learning Exiting lifelong learning methods can be roughly grouped into three categories, *i.e.,* regularization-based [36, 67, 92, 54, 66], dynamic-model-based [47, 66, 47], and generative-replay-based methods [70, 42, 84, 64, 86, 76]. Among these methods, generative replay is believed an effective and general strategy for general lifelong learning problems [64, 86, 40, 76], as discussed above. However, most existing methods of this type often have blurry generation (for images) or scalability issues [84, 58, 93, 80]. MeRGAN [84] leverages a copy of the current generator to replay previous data information, showing increasingly blurry historical generations with reinforced generation artifacts [93] as training proceeds. Lifelong GAN [93] employs cycle consistency (via an auxiliary encoder) and knowledge distillation (via copies of that encoder and the generator) to remember image-conditioned generation. However, it still shows decreased performance on historical tasks. By comparison, our GAN memory delivers realistic synthesis with *no* forgetting and scales well to high-dimensional situations with a long task-sequence, capable of serving as a realistic generative memory for general lifelong learning systems.

Transfer learning Being general and effective, transfer learning has attracted increasing attention recently in various research fields, with a focus on discriminative tasks like classification [6, 68, 88, 90, 16, 24, 44, 46, 72, 89] and those in natural language processing [3, 61, 53, 52, 2]. However for generative tasks, only a few efforts have been made [83, 57, 95]. For example, a GAN pretrained on a large-scale source dataset is used in [83] to initialize the GAN model in a target domain, for efficient training or even better performance; alternatively, [57] freezes the source GAN generator only to modulate its hidden-layer statistics to "add" new generation power with $L1$/perceptual loss; observing the general applicability of low-level filters in GAN generator/discriminator, [95] transfers-then-freezes them to facilitate generation with limited data in a target domain. Those methods either only concern synthesis in the target domain (completely forgetting source generation) [83, 95] or deliver blurry target generation [57]. Our GAN memory provides both realistic target generation and no forgetting on source generation.

3 Preliminary

We briefly review two building blocks on which our method is constructed: generative adversarial networks (GANs) [25, 33] and style-transfer techniques [62, 95].

Generative adversarial networks (GANs) GANs continue to show increasing power in synthesizing highly realistic observations [32, 45, 51, 9, 33, 34], and have found wide applicability in various fields [39, 1, 19, 79, 81, 82, 12, 87, 37]. A GAN often consists of a generator G and a discriminator D, with both trained adversarially with objective

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim q_{\text{data}}(\boldsymbol{x})}\big[\log D(\boldsymbol{x})\big] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\big[\log(1 - D(G(\boldsymbol{z})))\big], \tag{1}$$

where $p(\boldsymbol{z})$ is a simple distribution ($e.g.$, Gaussian) and $q_{\text{data}}(\boldsymbol{x})$ is the underlying (unknown) data distribution from which we observe samples.

Style-transfer techniques An extensive literature [23, 69, 77, 13, 60, 38] has explored how one can manipulate the style of an image (*e.g.,* the texture [18, 30, 41] or attributes [33, 34]) by modulating the statistics of its latent features. These methods use style-transfer techniques like conditional instance normalization [18] or adaptive instance normalization [30], most of which are related to Feature-wise Linear Modulation (FiLM) [62]. FiLM imposes simple element-wise affine transformations to latent features of a neural network, showing remarkable effectiveness in various domains [30, 17, 33, 57]. Given a d-dimensional feature $\boldsymbol{h} \in \mathbb{R}^d$ from a layer of a neural network,[2] FiLM yields

$$\hat{\boldsymbol{h}} = \boldsymbol{\gamma} \odot \boldsymbol{h} + \boldsymbol{\beta}, \tag{2}$$

where $\hat{\boldsymbol{h}}$ is forwarded to the next layer, $\odot$ denotes the Hadamard product, and the scale $\boldsymbol{\gamma} \in \mathbb{R}^d$ and shift $\boldsymbol{\beta} \in \mathbb{R}^d$ may be conditioned on other information [18, 62, 17]. Different from FiLM modulating latent features, another technique named adaptive filter modulation (AdaFM) modulates source convolutional (Conv) filters to manipulate its "style" to deliver a boosted transfer performance [95]. Specifically, given a Conv filter $\mathbf{W} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times K_1\times K_2}$, where $C_{\text{in}}/C_{\text{out}}$ denotes the number of input/output channels and $K_1 \times K_2$ is the kernel size, AdaFM yields

$$\hat{\mathbf{W}} = \boldsymbol{\Gamma} \odot \mathbf{W} + \mathbf{B}, \tag{3}$$

where the scale matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$, the shift matrix $\mathbf{B} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}}$, and the modulated $\hat{\mathbf{W}}$ is used to convolve with input feature maps for output ones. Note $\odot$ is implemented with broadcasting.

4 Proposed method

Targeting the fundamental memory issue of lifelong learning, we propose to exploit popular GANs to design a realistic *generative* memory (named GAN memory) to sequentially remember data-generating processes. Specifically, we consider a lifelong generation problem: the GAN memory sequentially accesses a stream of datasets/tasks $\{\mathcal{D}_1, \mathcal{D}_2, \cdots\}$[3] (during task t, only $\mathcal{D}_t$ is accessible); after task t, the GAN memory should be able to synthesize realistic samples resembling $\{\mathcal{D}_1, \cdots, \mathcal{D}_t\}$. Below we first reveal a surprising discovery that lays the foundation of the paper; we then build on top of it our GAN memory followed by a detailed analysis; and finally we present compression techniques to facilitate our GAN memory for lifelong problems with a long task sequence.

4.1 A surprising discovery

Moving well beyond the style-transfer literature modulating image features to manipulate its style [18, 30, 17], we discover that one can even modulate the "style" of a source generative/discriminative process (*e.g.,* a GAN generator/discriminator trained on a source dataset $\mathcal{D}_0$) to form synthesis power for a perceptually-distant target domain (*e.g.,* a generative/discriminative power on $\mathcal{D}_1$), via manipulating its FC and Conv layers with style-transfer techniques (or their variants).

Before introducing the technical details, we emphasize our basic assumption of well-behaved source FC and Conv parameters; often parameters from a GAN model trained on large-scale datasets satisfy that assumption. To highlight our discovery, we choose a moderately sophisticated GAN model [49] trained on the CelebA [43] (containing only faces) as the source, and select perceptually-distant target datasets including Flowers, Cathedrals, Cats, Brain-tumor images, Chest X-rays, and Anime images (see Figure 5 and Section 5.1). With the style-transfer techniques detailed below, we observe realistic

[2] For simplicity, we omit layer-index notation throughout the paper.

[3] This setup is not limited, as it's often convenient to use a physical memory buffer to form the data stream.

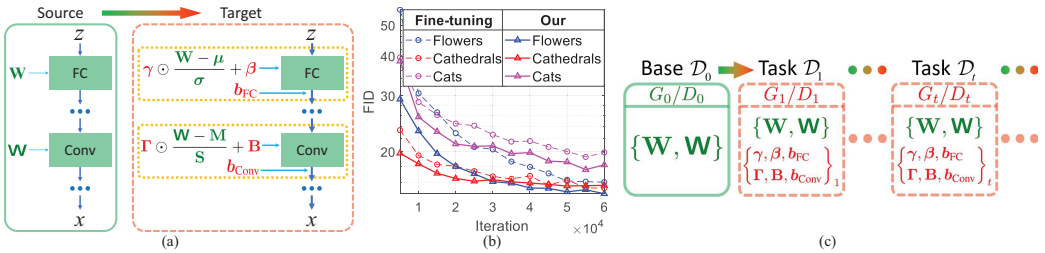


Figure 1: (a) Style modulation of a source GAN model (demonstrated with the generator, but it is also applied to the discriminator). Source parameters (green $\{\mathbf{W}, \mathbf{W}\}$) are frozen, with limited trainable style parameters (*i.e.,* red $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}$) introduced to form the augmentation to the target domain. (b) Comparing our style modulation to the strong fine-tuning baseline (see Appendix B for details). (c) The architecture of our GAN memory for a stream of target generation tasks.

generations in all target domains (see Figure 5), even though the generation power is modulated from an entirely different source domain. Alternatively given a specific target domain, that observation also implies the flexibility in choosing a source model, *i.e.,* the source (with well-behaved parameters) should have the same target data type, but it need not be related to the target domain. In the context of image-based data, this implies a certain universal structure to images, that may be captured within a GAN by one (relatively large) image dataset. Via appropriate style transformation of this model, it can be adapted to new and very different image classes, using potentially limited observations from those target domains.

We next present the style-transfer techniques employed here, modified FiLM (mFiLM) and modified AdaFM (mAdaFM), for modulating FC and Conv layers, respectively.

FC layers Given a source FC layer $\boldsymbol{h}^{\text{source}} = \mathbf{W}\boldsymbol{z} + \boldsymbol{b}$ with weight $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, bias $\boldsymbol{b} \in \mathbb{R}^{d_{\text{out}}}$, and input $\boldsymbol{z} \in \mathbb{R}^{d_{\text{in}}}$, mFiLM modulates its *parameters* to form a target function $\boldsymbol{h}^{\text{target}} = \hat{\mathbf{W}}\boldsymbol{z} + \hat{\boldsymbol{b}}$ with

$$\hat{\mathbf{W}} = \boldsymbol{\gamma} \odot \frac{\mathbf{W} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \boldsymbol{\beta}, \qquad \hat{\boldsymbol{b}} = \boldsymbol{b} + \boldsymbol{b}_{\text{FC}}, \tag{4}$$

where $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{d_{\text{out}}}$, with the elements $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i$ denoting the mean and standard derivation of the vector $\mathbf{W}_{i,:}$, respectively; $\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}} \in \mathbb{R}^{d_{\text{out}}}$ are target-specific scale, shift, and bias style parameters trained with target data ($\mathbf{W}$ and $\boldsymbol{b}$ are frozen – from learning on the original source domain – during target training). One may interpret mFiLM as applying FiLM [62] (or batch normalization [31]) to a source FC *weight* to modulate its (row) statistics/style (encoded in $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) to adapt to a target domain.

Conv layers Given a source Conv layer $\mathbf{H}^{\text{source}} = \mathbf{W} * \mathbf{H}' + \boldsymbol{b}$ with input feature maps $\mathbf{H}'$, Conv filters $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K_1 \times K_2}$, and bias $\boldsymbol{b} \in \mathbb{R}^{C_{\text{out}}}$, we leverage mAdaFM to modulate its parameters to form a target Conv layer as $\mathbf{H}^{\text{target}} = \hat{\mathbf{W}} * \mathbf{H}' + \hat{\boldsymbol{b}}$, where

$$\hat{\mathbf{W}} = \boldsymbol{\Gamma} \odot \frac{\mathbf{W} - \mathbf{M}}{\mathbf{S}} + \mathbf{B}, \qquad \hat{\boldsymbol{b}} = \boldsymbol{b} + \boldsymbol{b}_{\text{Conv}}, \tag{5}$$

where $\mathbf{M}, \mathbf{S} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$ with the elements $\mathbf{M}_{i,j}, \mathbf{S}_{i,j}$ denoting the mean and standard derivation of the vector $\text{vec}(\mathbf{W}_{i,j,:,:})$, respectively. The trainable target-specific style parameters are $\boldsymbol{\Gamma}, \mathbf{B} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$ and $\boldsymbol{b}_{\text{Conv}} \in \mathbb{R}^{C_{\text{out}}}$, with $\mathbf{W}$ and $\boldsymbol{b}$ frozen. Similar to mFiLM, mAdaFM first removes the source style (encoded in $\mathbf{M}$ and $\mathbf{S}$), followed by leveraging $\boldsymbol{\Gamma}$ and $\mathbf{B}$ to learn the target style.

The adopted style modulation process (shown with the generator) is illustrated in Figure 1(a). Given a source GAN model, we transfer and freeze all its parameters to a target domain, followed by using mFiLM/mAdaFM in (4)/(5) to modulate its FC/Conv layers (with style parameters $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}$) to yield the target generation model. With that style modulation, we transfer the source knowledge (within the frozen parameters) to the (potentially) perceptually-distant target domain to facilitate a realistic generation therein (see Figure 5 for generated samples). For quantitative evaluations, comparisons are made with a strong baseline, *i.e.,* fine-tuning the whole source model (including both generator and discriminator) on target data. The FID scores (lower is better) [29] from both methods are summarized in Figure 1(b), where our method consistently outperforms that strong baseline by a large margin, on both training efficiency and performance,[4] highlighting the

[4] The newly-introduced style parameters $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}$ of our method are only about 20% of those of the fine-tuning baseline, yet they deliver better efficiency and performance. This is likely because the target data are not sufficient enough to train well-behaved parameters like the source ones, when performing fine-tuning

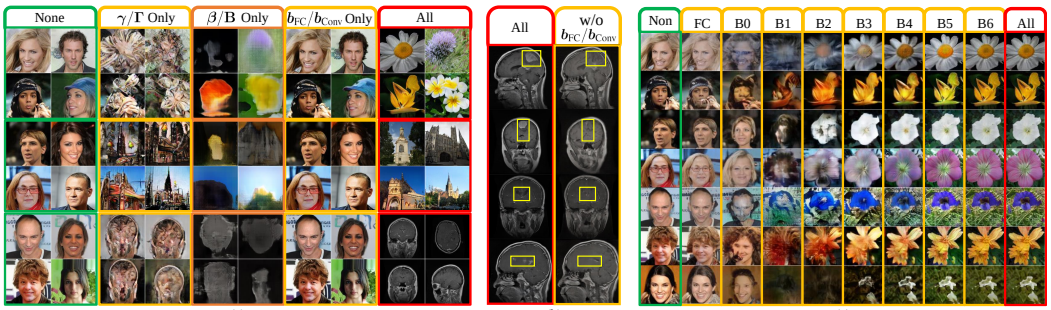


Figure 2: (a) Style parameters modulate different generation perspectives. (b) The biases model sparse objects. (c) Modulations in different blocks have different strength/focus over the generation. Bm is the mth residual block. B0/B6 is closest to the noise/observation. See Appendix C for details.

valuable knowledge within frozen source parameters (apparently a type of universal information about images) and showing a potentially better way for transfer learning.

Complementing the common knowledge of transfer learning, *i.e.,* low-level filters (those close to observations) are generally applicable, while high-level ones are task-specific [88, 90, 44, 4, 5, 20, 57, 95], the above discovery reveals an orthogonal dimensionality for transfer learning. Specifically, the shape of kernels (*i.e.,* relative relationship among kernel elements $\{\mathbf{W}_{i,j,1,1}, \mathbf{W}_{i,j,1,2}, \cdots\}$) may be generally transferable whereas the *statistics* of kernels (the mean $\mathbf{M}_{i,j}$ or standard derivation $\mathbf{S}_{i,j}$) or among-kernel correlations (*e.g.,* relative relationship among kernel statistics) are task-specific. A similar conjecture on low-level Conv filters was discussed in [95]; we reveal such patterns even hold for the whole GAN model (for both low-level and high-level kernels of the generator/discriminator), which is unanticipated because common experience associates high-level kernels with task-specific information. This insight might reveal a new avenue for transfer learning.

4.2 GAN memory to sequentially remember a stream of generative processes

Based on the above observations, we propose a GAN memory that has the power to realistically remember a stream of generative processes with *no* forgetting. The key observation here is when modulating a source GAN model for a target domain, the source model is frozen (thus no forgetting of the source) with limited target-specific style parameters (*i.e.,* no influence among tasks) introduced to form a realistic target generation. Accordingly, we can use a well-behaved source GAN model as a base, followed by sequentially modulating its "style" to deliver realistic generation power on a stream of target datasets,[5] as illustrated in Figure 1(c). We use the same settings as in Section 4.1. As style parameters are often limited (and can be further compressed as in Section 4.3), one could expect from our GAN memory a substantial compression of streaming datasets, while not forgetting realistic generative replay (see Figure 5). To help better understand how/why our GAN memory works, we next reveal five of its properties.

Each group of style parameters modulates a different generation perspective. Style parameters consist of three groups, *i.e.,* scales $\{\boldsymbol{\gamma}, \boldsymbol{\Gamma}\}$, shifts $\{\boldsymbol{\beta}, \mathbf{B}\}$, and biases $\{\boldsymbol{b}_{\text{FC}}, \boldsymbol{b}_{\text{Conv}}\}$. Taking as examples the style parameters trained on the Flowers, Cathedrals, and Brain-tumor images, Figure 2(a) demonstrates the generation perspective modulated by each group: (i) when none/all groups are applied, GAN memory generates realistic source/target images (see the first/last column, respectively); (ii) when only modulating via the scales (denoted as $\boldsymbol{\gamma}/\boldsymbol{\Gamma}$ Only, the second column), the generated samples show textural/structural information from target domains (like the textures on petals or the contours of buildings/skulls); (iii) as shown in the third column, the shifts if solely applied principally control the low-frequency color information from target domains; (iv) finally the biases (see the forth column) control the illumination and sparse objects (not obvious here). To clearly reveal the role played by the biases, we keep both scales and shifts fixed, and compare the generated samples with/without biases on the Brain-tumor dataset; Figure 2(b) shows the biases are important in modeling sparse objects like the tumor or tissue details, which may be valuable in pathological analysis. Also the following Figure 3(a) show that the biases help with a better training efficiency.

alone. Note that with the techniques from Section 4.3, one can use much less style parameters ($e.g.$, 7.3% of those of the fine-tuning baseline) to yield a comparable performance.

[5] Our GAN memory is amenable to streaming training, parallel training, and even their flexible combinations, thanks to the frozen base model and task-specific style parameters.

Figure 3: (a) Ablation study on Flowers. (b) Smooth interpolations between flower and cat generative processes. (c) Realistic replay from our conditional-GAN memory. See Appendix D for details and more demonstrations.

Style parameters within different blocks show different strength/focus over the generation. Figure 2(c) shows the generated samples on the Flowers dataset when gradually and accumulatively adding modulations to each block (from FC to B6). To begin with, the FC modulation changes the overall contrast and illumination of the generation; then the style parameters in B0-B3 make the most effort to modulate the face manifold into a flower, followed by modulations in B4-B6 refining the generation details. Such patterns are somewhat consistent with existing practice, in the sense that high-level/low-level *kernel statistics* are more task-specific/generally-applicable. It's therefore interesting to consider combining the two orthogonal dimensions, *i.e.,* the existing low-layer/high-layer split and the revealed kernel-shape/kernel-statistics split, for potentially better transfer learning.

Normalization contributes significantly to better training efficiency and performance. To investigate how the weight normalization and the biases in (4)/(5) contribute, ablation studies are conducted, with the results shown in Figure 3(a). With the weight normalization to remove the source "style," our method shows both an improved training efficiency and a boosted performance; the biases contribute to a better efficiency but with minimal influence on the performance.

GAN memory enables smooth interpolations among generative processes, by dynamically combining two (or more) sets of style parameters. Figure 3(b) demonstrates smooth interpolations between flower and cat generative processes. Such a property can be used to deliver versatile data augmentation among different domains, which may, for example, benefit a downstream robust classification [59, 7].

GAN memory readily generalizes to label-conditioned generation problems, where each task dataset $\mathcal{D}_t$ contains both observations $\{\boldsymbol{x}\}$ and the paired labels $\{y\}$ with $y \in \{1, \cdots, C_t\}$. Mimicking the conditional GAN [50], we specify one FC bias per class to model the label information; accordingly, the style parameters for the tth task are $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \{\boldsymbol{b}_{\text{FC}}\}_{i=1}^{C_t}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}$. Figure 3(c) shows the realistic replay from our conditional-GAN memory after sequential training on five tasks (exampled with bird, dog, and butterfly; see Section 5.2 for details).

4.3 GAN memory with further compression

Delivering realistic sequentially learned generations with no forgetting, the GAN memory presented above is expected to be sufficient for many practical applications with a moderate number of tasks. However, for challenging situations with many tasks, to save a set of style parameters for each task might become prohibitive.[6] For that problem, we reveal below (i) style parameters (*i.e.,* the expensive matrices $\boldsymbol{\Gamma}$ and $\mathbf{B}$) can be compressed to lower the memory cost for each task; (ii) one can exploit sharing parameters among tasks to further enhance memory savings.[7]

We first investigate the singular values of $\boldsymbol{\Gamma}$s and $\mathbf{B}$s learned at different blocks/layers, and summarize them in Figure 4(a). It's clear the $\boldsymbol{\Gamma}$ and $\mathbf{B}$ parameters are in general low-rank; moreover, the closer a $\boldsymbol{\Gamma}$ or $\mathbf{B}$ is to the noise (often with a larger matrix size and thus more expensive), the stronger its low-rank property (yielding better compressibility). Accordingly, we truncate/zero-out small singular values at each block/layer to test the corresponding performance decline. Figure 4(b) summarizes the results, where keeping 80% matrix energy [55] ($\approx 35\%$ top singular values) of $\boldsymbol{\Gamma}$ and $\mathbf{B}$ in B0-B4 almost has the same performance, verifying the compressibility of $\boldsymbol{\Gamma}$ and $\mathbf{B}$.

Based on the compressibility of $\boldsymbol{\Gamma}$ and $\mathbf{B}$, we next reveal parameter sharing among tasks can be exploited for further memory saving. Specifically, we propose to leverage matrix factorization and low-rank regularization to form a lifelong knowledge base mimicking [66]. Taking the tth task as an

[6] A compromise may save limited task-specific style parameters to hard disks and only load them when used.

[7] The cheap vector parameters $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{b}_{\text{Conv}}\}$ can be similarly processed in a dictionary learning manner. As they are often quite inexpensive to retain, we consider them being task-specific for simplicity.

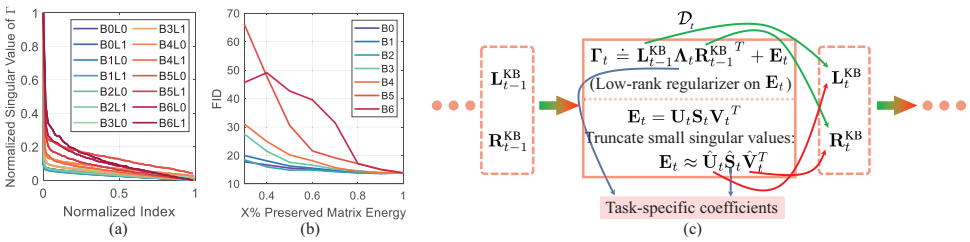


Figure 4: (a) Normalized singular values of $\mathbf{\Gamma}$ at all blocks/layers on Flowers (see Appendix Figure 14 for $\mathbf{B}$). Maximum normalization is applied to both axes. BmLn stands for the nth Conv layer of the mth block. (b) Influence of truncation (via preserving $X\%$ matrix energy [55]) on generation. (c) GAN memory with further compression and knowledge sharing. See Appendix G for details.

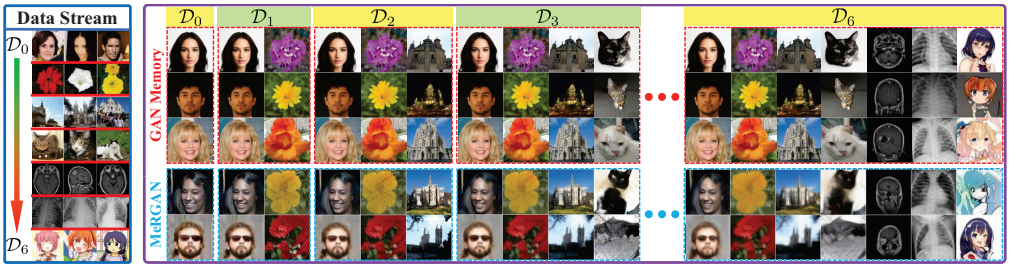


Figure 5: The task/data stream (left) and generated samples after training on each task/dataset (right).

example, instead of optimizing over a task-specific $\mathbf{\Gamma}_t$ (similarly for $\mathbf{B}_t$), we alternatively optimize over its parameterization $\mathbf{\Gamma}_t \doteq \mathbf{L}^{\text{KB}}_{t-1}\mathbf{\Lambda}_t(\mathbf{R}^{\text{KB}}_{t-1})^T + \mathbf{E}_t$, where $\mathbf{L}^{\text{KB}}_{t-1}$ and $\mathbf{R}^{\text{KB}}_{t-1}$ are respectively the existing left/right knowledge base, $\mathbf{\Lambda}_t = \text{Diag}(\boldsymbol{\lambda}_t)$, and $\boldsymbol{\lambda}_t$ and $\mathbf{E}_t$ are task-specific trainable parameters. The nuclear norm $\|\mathbf{E}_t\|_*$ is added to the loss to encourage a low-rank property. After training on the tth task, we apply singular value decomposition to $\mathbf{E}_t$, keep the top singular values to preserved $X\%$ matrix energy, and use the corresponding left/right singular vectors to update the left/right knowledge base to $\mathbf{L}^{\text{KB}}_t$ and $\mathbf{R}^{\text{KB}}_t$. The overall procedure is demonstrated in Figure 4(c).

5 Experiments

Experiments on high-dimensional image datasets from diverse fields are conducted to demonstrate the effectiveness of the proposed techniques. Specifically, we first test our GAN memory to realistically remember a stream of generative processes; we then show that our (conditional) GAN memory can be used to form realistic pseudo rehearsal (synthesis of data from prior tasks) to alleviate catastrophic forgetting for challenging lifelong classification tasks; and finally for long task sequences, we reveal the techniques from Section 4.3 enable significant memory savings but with comparable performance. Detailed experimental settings are given in Appendix A.

5.1 GAN memory on a stream of generation tasks

To demonstrate the superiority of our GAN memory over existing replay-based methods, we design a challenging lifelong generation problem consisting of 6 perceptually-distant tasks/datasets (see Figure 5): Flowers [56], Cathedrals [96], Cats [94], Brain-tumor images [15], Chest X-rays [35], and Anime faces.[8] The GAN model [49] trained on the CelebA [43] ($\mathcal{D}_0$) is selected as the base; other such GAN models may readily be considered. We compare our GAN memory with the memory replay GAN (MeRGAN) [84], which keeps another copy of the generator in memory to replay historical generations, to mitigate catastrophic forgetting. Qualitative comparisons between both methods along the sequential training are shown in Figure 5. It's clear our GAN memory delivers realistic generations with no forgetting on historical tasks, whereas MeRGAN shows increasingly blurry historical generations with reinforced generation artifacts [83] as training proceeds. For quantitative comparisons, the FID scores [29] along the training are summarized in Figure 6 (left), highlighting the advantages of our GAN memory, *i.e.,* realistic generations with no forgetting. Also revealed is that our method even performs better with a better efficiency for the current task, likely thanks to the transferred common knowledge within frozen base parameters.

[8] `https://github.com/jayleicn/animeGAN`

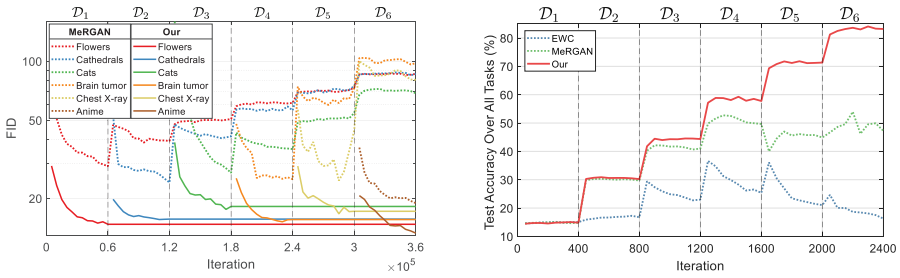


Figure 6: (Left) FID curves on the lifelong generation problem of Section 5.1. (Right) Classification accuracy on the lifelong classification problem of Section 5.2.

5.2 Conditional-GAN memory as pseudo rehearsal for lifelong classifications

Witnessing the success of our (conditional) GAN memory in continually remembering realistic generations, we next utilize it as a pseudo rehearsal to assist downstream lifelong classifications. Specifically, we design 6 streaming tasks by selecting fish, bird, snake, dog, butterfly, and insect images from the ImageNet [65]; each task is then formalized as a 6-classification problem (*e.g.,* for task bird, the goal is to classify 6 categories of birds). We employ the challenging class-incremental learning setup [76], *i.e.,* the classifier (after task t) is expected to accurately classify all observed (first $6t$) categories. For comparisons, we employ the regularization-based EWC [36] and the generative-replay-based MeRGAN [84]. Detailed settings are given in Appendix F.

Testing classification accuracy on all 36 categories from the compared methods along training are summarized in Figure 6 (right). It's clear that EWC barely works in the class-incremental learning scenario [75, 40, 76]. MeRGAN doesn't work well when the task sequence is long, because of its increasingly blurry rehearsal as shown in Figure 5. By comparison, our conditional-GAN memory with no forgetting succeeds in stably maintaining an increasing accuracy as new tasks come, highlighting its practical value in alleviating catastrophic forgetting for general lifelong learning problems. See Appendix F for evolution of the performance on each task along the training process.

5.3 GAN memory with parameter compression and sharing

Table 1: Comparisons of GAN memory with (Compr) or without (Naive) compression techniques. #Params denotes the number of newly introduced parameters for each task.

Task	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
$\#\text{Params}_{\text{Compr}}$	3.8M	1.7M	0.9M	0.8M	0.3M	0.3M
$\#\text{Params}_{\text{Compr}}/\#\text{Params}_{\text{Naive}}$	36.4%	16.0%	9.0%	7.6%	2.7%	2.9%
FID (Compr)	**27.67**	23.49	**28.90**	**31.07**	32.19	49.28
FID (Naive)	28.89	**22.80**	34.36	35.72	**29.50**	**40.03**

To verify the effectiveness of the compression techniques presented in Section 4.3, which are believed valuable for lifelong learning with many tasks, we design another lifelong generation problem based on the ImageNet for better demonstration.[9] Specifically, we select 6 categories of butterfly images to form a stream of 6 generation tasks/datasets (one category per task), among which similarity/knowledge-sharability is expected. The procedure shown in Figure 4(c) is employed for our method. See Appendix G for details. We compare our GAN memory with compression techniques (denoted as Compr) to its naive implementation with task-specific style parameters (Naive), with the results summarized in Table 1. It's clear that (i) even for task $\mathcal{D}_1$ (with an empty knowledge base), the low-rank property of $\mathbf{\Gamma}/\mathbf{B}$ enables a significant parameter compression; (ii) based on the existing knowledge base, much less new parameters are necessary to form a new generation model (*e.g.,* for task $\mathcal{D}_2$ or $\mathcal{D}_3$), confirming the reusability/sharability of existing knowledge; and (iii) though it has significant parameter compression, Compr delivers comparable performance to Naive, confirming the effectiveness of the presented compression techniques.

6 Conclusions

We reveal that one can modulate the "style" of a GAN model to accurately synthesize the statistics of data from perceptually-distant targets. Based on that recognition, we propose our GAN memory

[9] The datasets from Section 5.1 are too perceptually-distant to illustrate parameter sharability among tasks.

with growing generation power, but with no forgetting. We then analyze our GAN memory in detail, reveal for it new compression techniques, and empirically verify its advantages over existing methods. Concerning a better base model, one may leverage the generative replay ability of the GAN memory to form a long-period update, mimicking human behavior during rapid-eye-movement sleep [73, 22].

Broader impact

Capable of remembering a stream of data generative processes with no forgetting, our GAN memory has the following potential positive impact in the society: (i) it may serve as a powerful generative replay for challenging lifelong applications such as self-driving; (ii) as no original data are saved, the concerns on data privacy may be well addressed; (iii) GAN memory enables flexible control over the replayed contents, which is of great value to practical applications, where training data are unbalanced, or where one needs to flexibly select which model capability to maintain/forget during training. Since our GAN memory is built on top of GANs, it may inherit their ethical and societal impact. Despite being versatility, GANs may be improperly used to synthesize fake images/news/videos, resulting in negative consequences. Furthermore, we should be cautious of the failure of adversarial training due to mode collapse, which may compromise the generative capability on the current task. Note that training failure, if happens, will not hurt the performance on other tasks, showing certain robustness of our GAN memory.

Acknowledgments and Disclosure of Funding

The research was supported by part by DARPA, DOE, NIH, NSF and ONR. The Titan Xp GPU used was donated by the NVIDIA Corporation.

References

[1] K. Ak, J. Lim, J. Tham, and A. Kassim. Attribute manipulation generative adversarial networks for fashion images. In *ICCV*, pages 10541–10550, 2019.

[2] Y. Arase and J. Tsujii. Transfer fine-tuning: A BERT case study. *arXiv preprint arXiv:1909.00931*, 2019.

[3] X. Bao and Q. Qiao. Transfer learning from pre-trained bert for pronoun resolution. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 82–88, 2019.

[4] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, pages 6541–6549, 2017.

[5] D. Bau, J. Zhu, H. Strobelt, B. Zhou, J. Tenenbaum, W. Freeman, and A. Torralba. GAN dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

[6] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.

[7] D. Bertsimas, J. Dunn, C. Pawlowski, and Y. Zhuo. Robust classification. *INFORMS Journal on Optimization*, 1(1):2–34, 2019.

[8] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. Dickie, M. Hernández, J. Wardlaw, and D. Rueckert. GAN augmentation: augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[9] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[10] H. Caselles-Dupré, M. Garcia-Ortiz, and D. Filliat. Continual state representation learning for reinforcement learning using generative replay. *arXiv preprint arXiv:1810.03880*, 2018.

[11] F. Castro, M. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *ECCV*, pages 233–248, 2018.

[12] C. Chan, S. Ginosar, T. Zhou, and A. Efros. Everybody dance now. In *CVPR*, pages 5933–5942, 2019.

[13] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, pages 1897–1906, 2017.

[14] Z. Chen and B. Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[15] J. Cheng, W. Yang, M. Huang, W. Huang, J. Jiang, Y. Zhou, R. Yang, J. Zhao, Y. Feng, and Q. Feng. Retrieval of brain tumors by adaptive spatial pooling and fisher vector representation. *PloS one*, 11(6), 2016.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[17] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 3(7):e11, 2018.

[18] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[19] W. Fedus, I. Goodfellow, and A. Dai. MaskGAN: better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*, 2018.

[20] Y. Frégier and J. Gouray. Mind2mind: transfer learning for GANs. *arXiv preprint arXiv:1906.11613*, 2019.

[21] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

[22] S. Gais, G. Albouy, M. Boly, T. Dang-Vu, A. Darsaud, M. Desseilles, G. Rauchs, M. Schabus, V. Sterpenich, G. Vandewalle, et al. Sleep transforms the cerebral trace of declarative memories. *PNAS*, 104(47):18778–18783, 2007.

[23] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[26] C. Han, K. Murao, T. Noguchi, Y. Kawata, F. Uchiyama, L. Rundo, H. Nakayama, and S. Satoh. Learning more with less: conditional PGGAN-based data augmentation for brain metastases detection using highly-rough annotation on MR images. *arXiv preprint arXiv:1902.09856*, 2019.

[27] C. Han, L. Rundo, R. Araki, Y. Furukawa, G. Mauri, H. Nakayama, and H. Hayashi. Infinite brain MR images: PGGAN-based data augmentation for tumor detection. In *Neural Approaches to Dynamics of Signal Exchanges*, pages 291–303. Springer, 2020.

[28] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[29] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS*, pages 6626–6637, 2017.

[30] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510, 2017.

[31] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[32] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[33] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, June 2019.

[34] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*, 2019.

[35] D. S Kermany, M. Goldbaum, W. Cai, C. CS Valentim, H. Liang, S. L Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[36] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, page 201611835, 2017.

[37] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville. MelGAN: Generative adversarial networks for conditional waveform synthesis. *arXiv preprint arXiv:1910.06711*, 2019.

[38] L. Kurzman, D. Vazquez, and I. Laradji. Class-based styling: Real-time localized style transfer with semantic segmentation. In *ICCV*, pages 0–0, 2019.

[39] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 4681–4690, 2017.

[40] T. Lesort, H. Caselles-Dupré, M. Garcia-Ortiz, A. Stoian, and D. Filliat. Generative models from the perspective of continual learning. In *IJCNN*, pages 1–8. IEEE, 2019.

[41] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. Yang. Universal style transfer via feature transforms. In *NIPS*, pages 386–396, 2017.

[42] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[43] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015.

[44] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[45] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? a large-scale study. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *NeurIPS*, pages 700–709. Curran Associates, Inc., 2018.

[46] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *NIPS*, pages 165–177, 2017.

[47] N. Masse, G. Grant, and D. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

[48] M. McCloskey and N. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[49] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? In *ICML*, pages 3478–3487, 2018.

[50] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[51] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[52] M. Moradshahi, H. Palangi, M. Lam, P. Smolensky, and J. Gao. HUBERT untangles BERT to improve transfer across NLP tasks. *arXiv preprint arXiv:1910.12647*, 2019.

[53] M. Mozafari, R. Farahbakhsh, and N. Crespi. A BERT-based transfer learning approach for hate speech detection in online social media. *arXiv preprint arXiv:1910.12574*, 2019.

[54] C. Nguyen, Y. Li, T. Bui, and R. Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[55] V. Nikiforov. The energy of graphs and matrices. *JMAA*, 326(2):1472–1475, 2007.

[56] M. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICCV, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[57] A. Noguchi and T. Harada. Image generation from small datasets via batch statistics adaptation. *arXiv preprint arXiv:1904.01774*, 2019.

[58] G. Parisi, R. Kemker, J. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[59] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.

[60] T. Park, M. Liu, T. Wang, and J. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346, 2019.

[61] Y. Peng, S. Yan, and Z. Lu. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. *arXiv preprint arXiv:1906.05474*, 2019.

[62] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[63] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017.

[64] A. Rios and L. Itti. Closed-loop memory gan for continual learning. *arXiv preprint arXiv:1811.01146*, 2018.

[65] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[66] J. Schwarz, J. Luketina, W. Czarnecki, A. Grabska-Barwinska, Y. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

[67] A. Seff, A. Beatson, D. Suo, and H. Liu. Continual learning in generative adversarial nets. *NeurIPS*, 2017.

[68] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[69] F. Shen, S. Yan, and G. Zeng. Neural style transfer via meta networks. In *CVPR*, pages 8061–8069, 2018.

[70] H. Shin, J. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *NIPS*, pages 2990–2999, 2017.

[71] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, pages 3400–3409, 2017.

[72] Q. Sun, B. Schiele, and M. Fritz. A domain based approach to social relation recognition. In *CVPR*, pages 3481–3490, 2017.

[73] P. Taupin and F. Gage. Adult neurogenesis and neural stem cells of the central nervous system in mammals. *Journal of neuroscience research*, 69(6):745–749, 2002.

[74] S. Thrun and T. Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995.

[75] G. van de Ven and A. Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[76] G. van de Ven and A. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[77] H. Wang, X. Liang, H. Zhang, D. Yeung, and E. Xing. Zm-net: Real-time zero-shot image manipulation network. *arXiv preprint arXiv:1703.07255*, 2017.

[78] J. Wang and L. Perez. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, 2017.

[79] K. Wang and X. Wan. SentiGAN: Generating sentimental texts via mixture adversarial networks. In *IJCAI*, pages 4446–4452, 2018.

[80] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, C. Cao, D. Jiang, and M. Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*, 2020.

[81] R. Wang, D. Zhou, and Y. He. Open event extraction from online text using a generative adversarial network. *arXiv preprint arXiv:1908.09246*, 2019.

[82] T. Wang, M. Liu, J. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

[83] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu. Transferring GANs: generating images from limited data. In *ECCV*, pages 218–234, 2018.

[84] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al. Memory replay GANs: Learning to generate new categories without forgetting. In *NeurIPS*, pages 5962–5972, 2018.

[85] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019.

[86] Y. Xiang, Y. Fu, P. Ji, and H. Huang. Incremental learning using conditional adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6619–6628, 2019.

[87] R. Yamamoto, E. Song, and J. Kim. Probability density distillation with generative adversarial networks for high-quality parallel waveform generation. *arXiv preprint arXiv:1904.04472*, 2019.

[88] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[89] A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 3712–3722, 2018.

[90] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.

[91] G. Zeng, Y. Chen, B. Cui, and S. Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[92] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995. JMLR. org, 2017.

[93] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori. Lifelong GAN: Continual learning for conditional image generation. In *ICCV*, pages 2759–2768, 2019.

[94] W. Zhang, J. Sun, and X. Tang. Cat head detection-how to effectively exploit shape and texture features. In *ECCV*, pages 802–816. Springer, 2008.

[95] M. Zhao, Y. Cong, and L. Carin. On leveraging pretrained GANs for limited-data generation, 2020.

[96] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, pages 487–495, 2014.

Appendix of GAN Memory with No Forgetting

Yulai Cong*, Miaoyun Zhao*, Jianqiao Li, Sijia Wang, Lawrence Carin
Department of ECE, Duke University

A Experimental settings

For all experiments about GAN memory and MeRGAN, we inherit the architecture and experimental settings from GP-GAN [49]. Note, for both GAN memory and MeRGAN, we use the GP-GAN model pretrained on CelebA. In the implementation of GAN memory, we apply style modulation on all layers except the last Conv layer for generator, and apply the proposed style modulation on all layers except the last FC layer for discriminator. Adam is used as the optimizer with learning rate 1×10^{-4} and coefficients $(\beta_1, \beta_2) = (0.0, 0.99)$. For the discriminator, gradient penalty on real samples (R_1-regularizer) is applied with $\gamma = 10.0$. For MeRGAN, the Replay alignment parameter is set as $\lambda_{\text{RA}} = 1 \times 10^{-3}$, which is the same as their publication.

All images are resized to 256×256 for consistency. The dimension of the input noise is set to 256 for all tasks. The FID scores are calculated using N real and generated images, for the dataset with less than 10,000 images, we set N as the number of data samples; for the dataset with larger than 10,000 samples, we set N as 10,000.

Code will be available upon publication.

B On Figure 1(b)

To demonstrate that the introduced trainable parameters in our GAN memory is enough to manage a decent performance on new task. We test its performance via FID and compare with a strong baseline Fine-tuning.

On the Fine-tuning method. It inherits the architecture from GP-GAN which is the same as the green/frozen part of our GAN memory (see Figure 1(a)). Given a target task/data (*e.g.,* Flowers, Cathedrals, or Cats), all the parameters are trainable and fine-tuned to fit the target data. We consider Fine-tuning as a strong baseline because it utilizes the whole model power on the target data.

C On Figure 2

For all illustrations, we train GAN memory on the target data and record the well trained style parameters as $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=1}$. According to equation (4) and (5), we can represent the style parameters for the source data CelebA as $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=0} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{0}, \mathbf{M}, \mathbf{S}, \mathbf{0}\}$. Then, we can get the generation by selectively replacing $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=0}$ with $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=1}$. Note, for Figure 2(a)(b) the operations are explained within one specific FC/Conv layer, one need to apply it to all layers in real implementation. The detailed techniques are as follows.

On Figure 2(a). Take the target data Flowers as an example,

- "None" means no modulation from target data is applied, we only use the modulation from the source data $\{\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{0}, \mathbf{M}, \mathbf{S}, \mathbf{0}\}$ and get face images;
- "$\boldsymbol{\gamma}/\boldsymbol{\Gamma}$ Only" means that we replace the $\boldsymbol{\gamma}/\boldsymbol{\Gamma}$ parameters from source data with the one from target data, namely using the modulation $\{\{\boldsymbol{\gamma}\}_{t=1}, \boldsymbol{\sigma}, \mathbf{0}, \{\boldsymbol{\Gamma}\}_{t=1}, \mathbf{S}, \mathbf{0}\}$ for generation;
- "$\boldsymbol{\beta}/\mathbf{B}$ Only" means that we replace the $\boldsymbol{\beta}/\mathbf{B}$ parameters from source data with the one from target data, namely using the modulation $\{\boldsymbol{\mu}, \{\boldsymbol{\beta}\}_{t=1}, \mathbf{0}, \mathbf{M}, \{\mathbf{B}\}_{t=1}, \mathbf{0}\}$ for generation;
- "$\boldsymbol{b}_{\text{FC}}/\boldsymbol{b}_{\text{Conv}}$ Only" means that we replace the $\boldsymbol{b}_{\text{FC}}/\boldsymbol{b}_{\text{Conv}}$ parameters from source data with the one from target data, namely using the modulation $\{\boldsymbol{\mu}, \boldsymbol{\sigma}, \{\boldsymbol{b}_{\text{FC}}\}_{t=1}, \mathbf{M}, \mathbf{S}, \{\boldsymbol{b}_{\text{Conv}}\}_{t=1}\}$ for generation;
- "All" means all style parameters from target data $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=1}$ are applied and the model generates flowers;

On Figure 2(b).

- "All" is obtained via a similar way to that of Figure 2(a);
- "w/o $\boldsymbol{b}_{\text{FC}}/\boldsymbol{b}_{\text{Conv}}$" means using the style parameters from target data without $\boldsymbol{b}_{\text{FC}}/\boldsymbol{b}_{\text{Conv}}$, namely using the modulation $\{\{\boldsymbol{\gamma}\}_{t=1}, \{\boldsymbol{\beta}\}_{t=1}, \mathbf{0}, \{\boldsymbol{\Gamma}\}_{t=1}, \{\mathbf{B}\}_{t=1}, \mathbf{0}\}$ for generation;

On Figure 2(c).

- "None" and "All" are obtained via a similar way to that of Figure 2(a);
- "FC" is obtained by applying a newly designed style parameters which copies the style parameters for souce data and replaces these style parameters within FC layer with the style parameters within the FC layer for target data;
- "B0" is obtained by copying the designed style parameters under the "FC" setting and replacing these style parameters within B0 block with those from target data;
- "B1" is obtained by copying the designed style parameters under the "B0" setting and replacing these style parameters within B1 block with those from target data;
- and so forth for the later blocks.

D On Figure 3

D.1 On Figure 3(a)

The ablation study is conducted to test the effect the normalization operation and the bias term on GAN memory.

- "Our" is our GAN memory;
- "NoNorm" is a modified version of our GAN memory which removes the normalization on $\mathbf{W}/\mathbf{W}$ in Equation (4)/(5) and results in,

$$\hat{\mathbf{W}} = \boldsymbol{\gamma} \odot \mathbf{W} + \boldsymbol{\beta}$$

and

$$\hat{\mathbf{W}} = \boldsymbol{\Gamma} \odot \mathbf{W} + \mathbf{B}$$

- "NoBias" is a modified version of our GAN memory which removes the bias term $\boldsymbol{b}_{\text{FC}}/\boldsymbol{b}_{\text{Conv}}$ in Equation (4)/(5) and results in,

$$\hat{\boldsymbol{b}} = \boldsymbol{b}$$

D.2 On Figure 3(b)

Here we discuss the detailed techniques for interpolation among different generative processes with our GAN memory and show more examples in Figure 7, 8, 9, and 10. Taking the smooth interpolation between flowers and cats generative processes as an example, we do the following procedures to get the results.

- Train GAN memory on Flowers and Cats independently and get the well trained style parameters for Flowers as $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=1}$ and for Cats as $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=2}$;
- Sample a bunch of random noise $\boldsymbol{z}$ (*e.g.,* 8 random noise vectors) and fix it;
- Get the interpolated generation by dynamically combining the two sets of style parameters via

$$(1-\lambda_{\text{Interp}})\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=1} + \lambda_{\text{Interp}}\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=2},$$

with λ_{Interp} varying from 0 to 1.

Note, the task CelebA has been well pretrained and we can obtain the style parameters directly as $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{b}_{\text{FC}}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{b}_{\text{Conv}}\}_{t=0} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{0}, \mathbf{M}, \mathbf{S}, \mathbf{0}\}$.

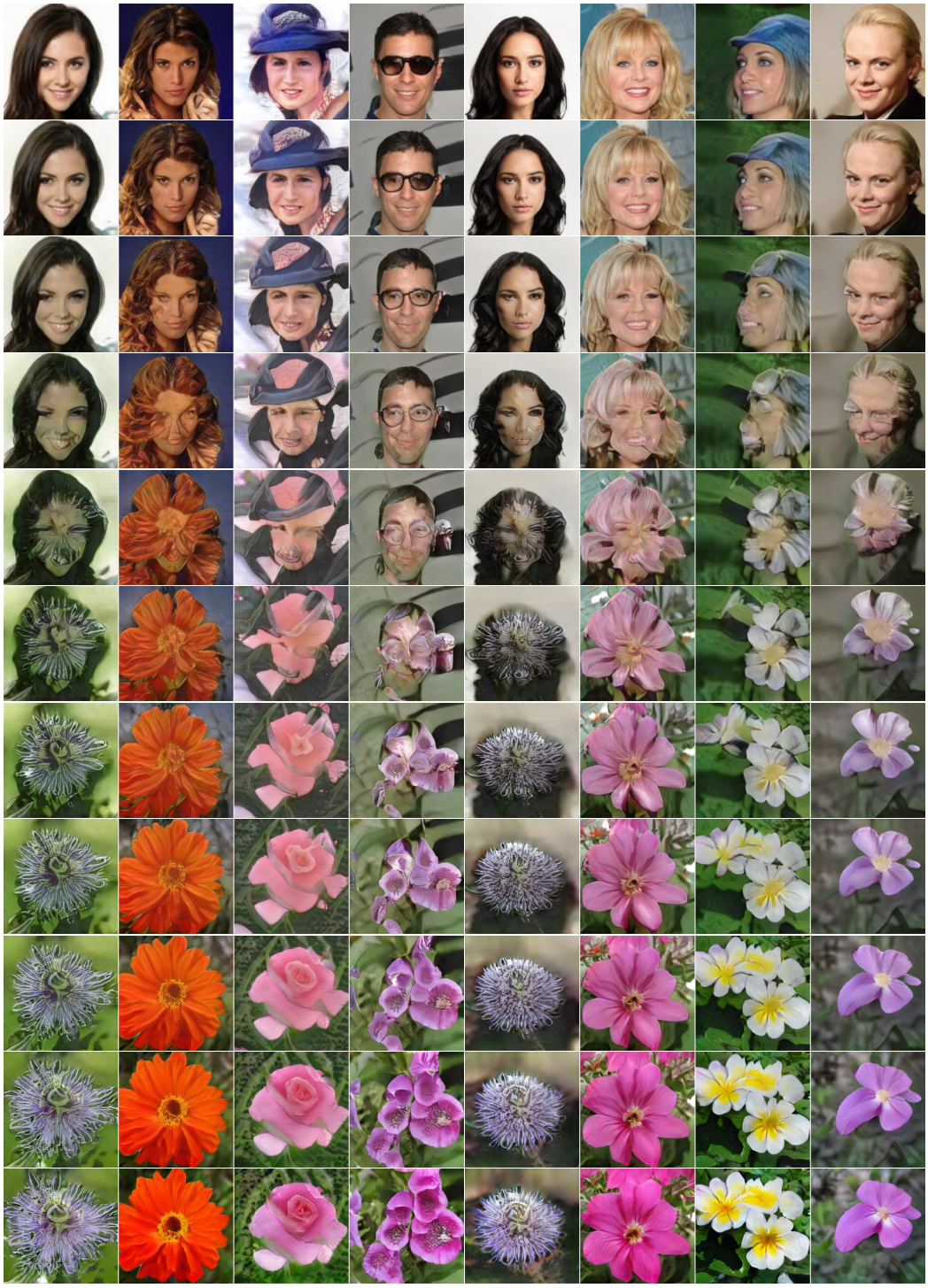


Figure 7: Smooth interpolations between faces and flowers generative processes via GAN memory.

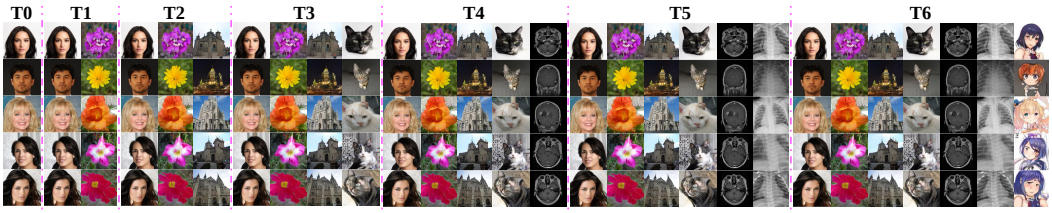
Figure 8: Smooth interpolations between flowers and cats generative processes via GAN memory.

Figure 9: Smooth interpolations between cats and cathedrals generative processes via GAN memory.

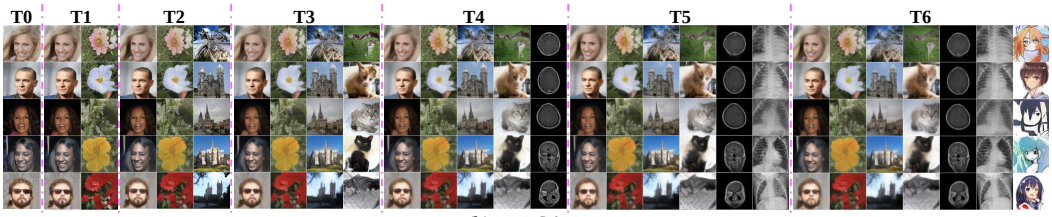
Figure 10: Smooth interpolations between cathedrals and Brain-tumor images generative processes via GAN memory.

E More results for Section 5.1

Given the same model pretrained for CelebA, we train GAN memory and MeRGAN on a sequence of tasks: Flowers (8,189 images), Cathedrals (7,350 images), Cats (9,993 images), Brain-tumor images (3,064 images), Chest X-rays (5,216 images), and Anime images (115,085 images). When task t presents, only the data from $\mathcal{D}_t$ is available, and GAN memory/MeRGAN is expected to generate samples for all learned tasks $\mathcal{D}_1, \cdots, \mathcal{D}_t$. Due to the limited space, we only show part of the results in Figure 5. To make it clearer, Figure 11 shows a complete results with the generations for all tasks along the training process listed.



(a) GAN Memory



(b) MeRGAN

Figure 11: Comparing the generated samples from GAN memory (top) and MeRGAN (bottom) on lifelong learning of a sequential generation tasks: Flowers, Cathedrals, Cats, Brain-tumor images, Chest X-rays, and Anime images. MeRGAN shows an increasing blurriness along the task sequence, while GAN memory can learn to perform the current task and remembering the old tasks realistically.

F Experimental settings for lifelong classification

Given a sequence of 6 tasks: fish, bird, snake, dog, butterfly, and insect selected from ImageNet. Each task is a classification problem over six categories/sub-classes. We employ the challenging class-incremental learning setup, when task t presents, the classifier (have been trained on previous $t-1$ tasks) is expected to accurately classify all observed $6t$ categories (namely, the previous $6(t-1)$ categories plus the current 6 categories). There are $6 \times 6 = 36$ categories/sub-classes in total, for each category/sub-class, we randomly select 1200 images for training and 100 for testing.

As for the classification model, we select the pretrained ResNet18 model for ImageNet. The optimizer for classification is selected as Adam with learning rate 1×10^{-4} and coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$.

For EWC, we adopt the code from [76], and set the parameters as $\lambda_{\text{EWC}} = 10^4$. We also tried the setting $\lambda_{\text{EWC}} = 10^9$ used in [84], however, the results shows that $\lambda_{\text{EWC}} = 10^9$ is too strong in our case, *e.g.,* when the learning proceeds to task $5/6$, the parameters are strongly pinned to the one learned for previous tasks, which makes it difficult to learn the new/current task, and the accuracy for the cuttent task kept almost 0. For GAN memory and MeRGAN, we adopt the two step framework from [70], every time a new task t presents, we (i) train GAN memory/MeRGAN (which has already remembered all the previous tasks $1 \sim t-1$) to remember the generation for both the previous tasks and the current task; (ii) at the same time, train the classifier/solver on a combined dataset: real samples for current task with their labels provided and generated samples for previous tasks replayed by GAN memory/MeRGAN. Since we apply label conditioned generation, where the category is an input, the replay process can be readily controlled with the reliable sampling of $(\boldsymbol{x}, \boldsymbol{y})$ pairs (*e.g.,* sample the label $\boldsymbol{y}$ and noise $\boldsymbol{z}$ first and then sample data $\boldsymbol{x}$ via the generator $\boldsymbol{x} = G(\boldsymbol{z}, \boldsymbol{y})$), which is considered effective in avoiding potential classification errors and biased sampling towards recent categories [84]. Note, for both GAN memory and MeRGAN, we used the GP-GAN model pretrained on CelebA.

The batch size for EWC is set as $n = 36$. For replay based method, the batch size for classification is set as follows. When task $t = 1, 2, \cdots, T$ presents, the batch size is $n \times t$: (i) n samples from the current task; (ii) $n \times (t-1)$ generated samples replayed by GAN memory/MeRGAN for previous $t-1$ tasks (each task with n replayed samples).

The classification accuracy shown in Figure 6 (right) is obtained by testing the classifier on the whole test dataset for all tasks with $36 \times 100 = 3600$ images. We also show in Figure 12 the evolution of the classification accuracy for each task during the whole training process. Take Figure 12(a) as an

example, the shown classification accuracy for task 1 on $\mathcal{D}_1$ is obtained by testing the classifier on the test images belonging to $\mathcal{D}_1$ with $6 \times 100 = 600$ images.

We observe from Figure 12 that, (i) EWC forgets the previous task much quickly than the others, *e.g.,* when the learning proceeds to task $\mathcal{D}_6$, the knowledge/capability learned from task $\mathcal{D}_1, \mathcal{D}_2$, and $\mathcal{D}_3$ are totally forgotten and the performances are seriously decreased; (ii) MeRGAN shows clear performance decline on historical classifications due to its blurry rehearsal on complex datasets, which is especially obvious when the task sequence becomes long (see Figure 12(a)(b)(c)). (iii) By comparison, our (conditional) GAN memory succeeds in stably maintaining historical accuracy even when the sequence becomes long thanks to its realistic pseudo rehearsal, highlighting its practical values in serving as a generative memory for general lifelong learning problems.

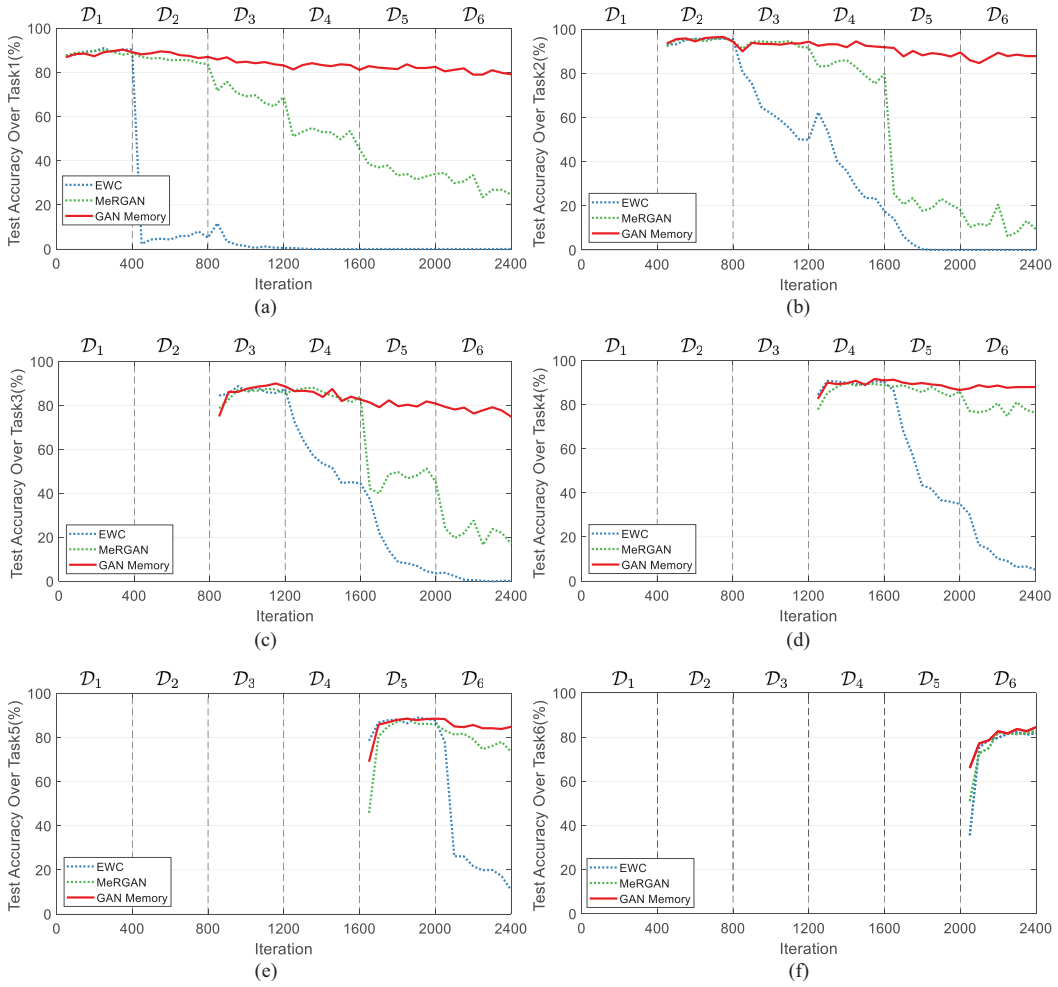
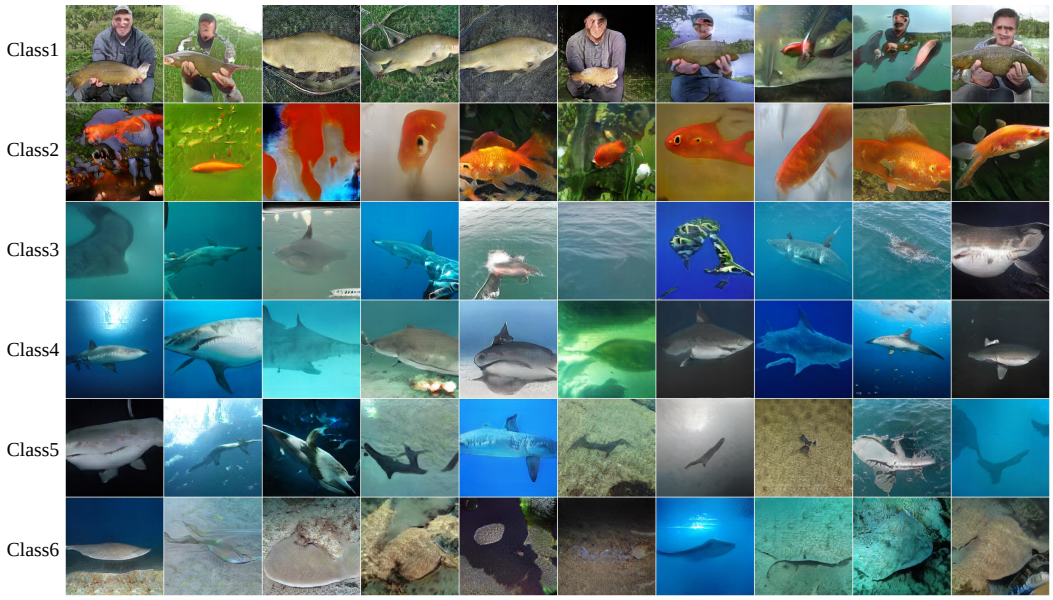


Figure 12: The evolution of the classification accuracy for each task during the whole training process: (a) Task $\mathcal{D}_1$; (b) Task $\mathcal{D}_2$; (c) Task $\mathcal{D}_3$; (d) Task $\mathcal{D}_4$; (e) Task $\mathcal{D}_5$; (f) Task $\mathcal{D}_6$.

Figure 13 shows the realistic replay from our conditional-GAN memory after sequential training on five tasks.[10]

[10]The last task have its real data available, thus there is no need to learn to replay



(a) fish

(c) snake

(d) dog

(e) butterfly

Figure 13: Realistic replay from our conditional-GAN memory. (a) fish; (b) bird; (c) snake; (d) dog; (e) butterfly.

G GAN memory with further compression

G.1 The compression method

Our proposed GAN memory is practical when the sequence of tasks is mediate. However when the number of tasks is extremely large, *e.g.,* 1000, the introduced parameters for GAN memory will be expensive too. To make the GAN memory scalable to the number of tasks, we propose a naive method to compress the scale and bias $\mathbf{\Gamma}, \mathbf{B}$ by taking advantage of the redundancy (Low-rank) within it, which is potential to remember extremely long sequence of tasks with finite parameters.

Similar to Figure 4(a), the singular values of scale $\mathbf{\Gamma}$ and bias $\mathbf{B}$ learned at different blocks/layers are summarized in Figure 14. Obviously, those bias $\mathbf{B}$ are also generally low-rank and the closer a bias $\mathbf{B}$ to the noise, the stronger low-rank property it shows.

Maximum normalization is applied for both Figure 4(a) and Figure 14 to provide a clear illustration. Take curve "B0L0" as an example, given the singular values vector as $\boldsymbol{y}$ and the index[11] of each element as $\boldsymbol{x}$, we do the Maximum normalization as $\hat{\boldsymbol{x}} = \boldsymbol{x}/\max(\boldsymbol{x})$, $\hat{\boldsymbol{y}} = \boldsymbol{y}/\max(\boldsymbol{y})$, and then plot $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}\}$.

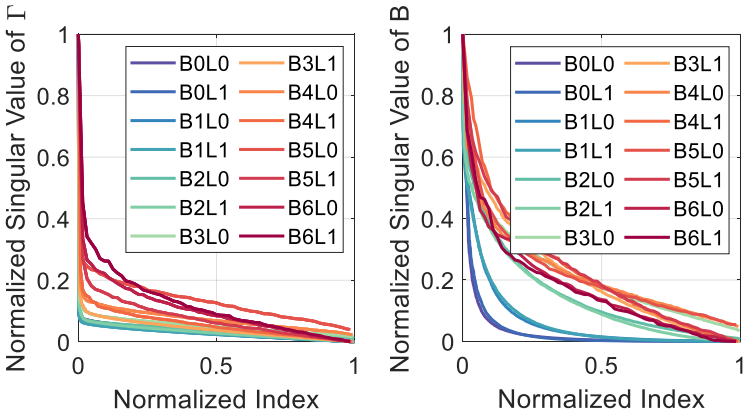


Figure 14: The singular value of the scale $\mathbf{\Gamma}$ (left) and bias $\mathbf{B}$ (right) within each Conv layer on Flowers.

Based on the discovered low-rank property, we test the compressibility of the learned parameters by truncating/zero-out small singular values. The results in Figure 4(b) are obtained by only keeping $30\% \sim 100\%$ matrix energy [55] at each block alternatively and evaluating their performance (FID). Take B0 as an example, we keep $30\% \sim 100\%$ matrix energy for all the scale $\mathbf{\Gamma}$ and bias $\mathbf{B}$ matrices within block B0 with the other blocks/layers unchanged and evaluate the FID.

To have a straight forward understanding of the proposed compression method for our GAN memory (see Figure 4(c)), we summarize the specific steps in Algorithm 1, where $r = 0.01$ works well in the experiments. Given $\mathbf{E}_t = \mathbf{U}_t\mathbf{S}_t{\mathbf{V}_t}^T$ with $\mathbf{S}_t = \text{Diag}(\boldsymbol{s}_t)$, the low-rank regularization for the first task is simply implemented via $\min \|\mathbf{E}_1\|_* = \min \|\boldsymbol{s}_1\|_1$; the low-rank regularization for the later tasks are implemented via $\min \|\mathbf{E}_t\|_* = \min \|\boldsymbol{\eta} \odot \boldsymbol{s}_t\|_1$, where $\boldsymbol{\eta}$ is a vector of same size as $\boldsymbol{s}_t$ with non-negative values and is designed as $\boldsymbol{\eta} = 0.1 + \sigma(\frac{10 \cdot j}{J})$ with $\sigma(\cdot)$ as a sigmoid function, J as the length of $\boldsymbol{s}_t$, j as the index of $[\boldsymbol{s}_t]_j$. This kind of designation forms a prior which imposes weak sparse constraint on part of the elements in $\boldsymbol{s}_t$ and imposes strong sparse constraint on the rest. This prior is found effective in keeping a good performance.

Algorithm 1 GAN memory with further compression (exampled by $\mathbf{\Gamma}_t$ within a Conv layer)

Input: A sequence of T tasks, $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$, the knowledge base $\mathbf{L} = \emptyset$ and $\mathbf{R} = \emptyset$, a scale r to balance between the low-rank regularization and the generator loss.

Output: The knowledge base $\mathbf{L}$ and $\mathbf{R}$, the task-specific coefficients, $\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_T$

1: **for** t in task 1 to task T **do**
2: Train GAN memory on the current task t with the following settings:
(i) $\mathbf{\Gamma}_t \doteq \mathbf{L}_{t-1}^{\text{KB}}\mathbf{\Lambda}_t{\mathbf{R}_{t-1}^{\text{KB}}}^T + \mathbf{E}_t$ with $\mathbf{\Lambda}_t = \text{Diag}(\boldsymbol{\lambda}_t)$;
(ii) Generator loss with low-rank regularization $r\|\mathbf{E}_t\|_*$ considered;
3: Do SVD on $\mathbf{E}_t$ as $\mathbf{E}_t = \mathbf{U}_t\mathbf{S}_t{\mathbf{V}_t}^T$ with $\mathbf{S}_t = \text{Diag}(\boldsymbol{s}_t)$
4: Zero-out the small singular values to keep $X\%$ matrix energy for $\mathbf{\Gamma}_t$ and obtains $\mathbf{E}_t \approx \hat{\mathbf{U}}_t\hat{\mathbf{S}}_t\hat{\mathbf{V}}_t^T$
5: Collect the task-specific coeffecients for task t as $\boldsymbol{c}_t = [\boldsymbol{\lambda}_t, \boldsymbol{s}_t]$
6: Collect knowledge base $\mathbf{L} = [\mathbf{L}, \hat{\mathbf{U}}_t]$, $\mathbf{V} = [\mathbf{V}, \hat{\mathbf{V}}_t]$;
7: **end for**

[11] The index here begins from 0.

G.2 Experiments on the compressibility of the GAN memory

From Figure 4(b), we observe that $\mathbf{\Gamma}/\mathbf{B}$ from different blocks/layers have different compressibility: the larger the matrix size for $\mathbf{\Gamma}/\mathbf{B}$ (the closer to the noise), the larger the compression ratio is. Thus, it is proper to keep different percent of matrix energy for different blocks/layers to minimize the decline in final performance. The results shown in Table 1 are obtained by keeping $\{80\%, 80\%, 90\%, 95\%\}$ matrix energy for $\mathbf{\Gamma}/\mathbf{B}$ within B0, B1, B2, B3 respectively. The compression techniques are not considered for the other blocks/layers (*i.e.,* B4, B5, B6 and FC) because these layers have a relatively small amount of parameters.

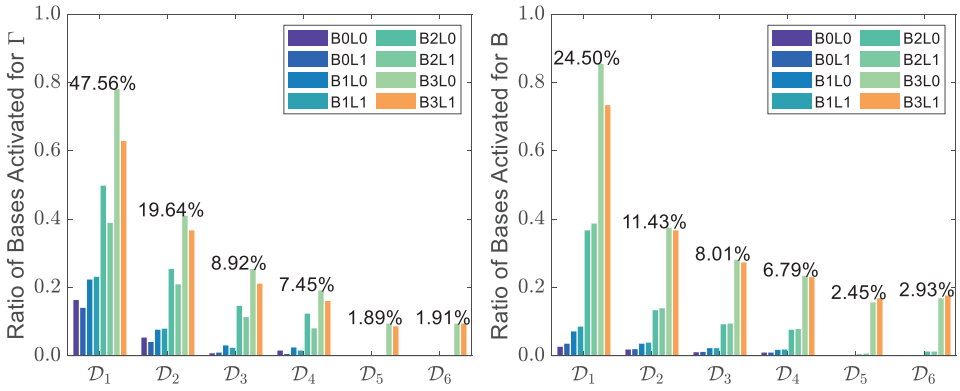


Figure 15: The percentage of newly added bases for scale $\mathbf{\Gamma}$(left) and bias $\mathbf{B}$(right) after the sequential training on each of the 6 butterfly tasks. Each bar value is a ratio between the number of newly added bases and the maximum rank of $\mathbf{\Gamma}/\mathbf{B}$. The percentages are the ratio between the amount of newly added parameters with and without using compression for each task.

By comparing to a naive implementation with task-specific style parameters (see Section 4.2), Figure 15 shows the the overall (and block/layer wise) compression ratios delivered by the compression techniques as training proceeds. Each bar shows the ratio between the number of newly added bases and the maximum rank of $\mathbf{\Gamma}/\mathbf{B}$. The shown percentages above each group of bars are the ratio between the amount of newly added parameters with and without using compression for each task. It's clear that (i) even for task $\mathcal{D}_1$ (with an empty knowledge base), the low-rank property of $\mathbf{\Gamma}/\mathbf{B}$ enables a significant parameter compression; (ii) based on the existing knowledge base, much less new parameters are necessary to form a new generation power (*e.g.,* for task $\mathcal{D}_2$, $\mathcal{D}_3$, $\mathcal{D}_4$, and $\mathcal{D}_5$), confirming the reusability/sharability of existing knowledge; and (iii) the lower the block (the closer to the noise with a larger matrix size for $\mathbf{\Gamma}/\mathbf{B}$), the larger the compression. All these three factors together lead to the significant overall compression ratios.

Intuitively, when more tasks are learned and the knowledge base is rich enough, no new bases would be necessary for future tasks (despite we still need to save tiny task-specific coefficients). For the proposed compression method, the selection of a proper threshold for keeping matrix energy is very important in balancing between saving parameters and keeping performance (*e.g.,* when the task number becomes large, the negative effect of the selected threshold on FID performance emerges, see Table 1), we leave this as future research direction.