Memory Protection Generative Adversarial Network (MPGAN): A Framework to Overcome the Forgetting of GANs Using Parameter Regularization Methods

YIFAN CHANG, WENBO LI, JIAN PENG, HAIFENG LI, YU KANG, (Senior Member, IEEE), AND YINGLIANG HUANG

1Department of Automation, University of Science and Technology of China, Hefei 230026, China
2Institute of Technology Innovation, Hefei Institute of Physical Science, Chinese Academy of Sciences, Hefei 230088, China
3School of Geosciences and Info-Physics, Central South University, Changsha 410083, China
4Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031, China

Corresponding author: Yu Kang (kangduyu@ustc.edu.cn)

This work was supported in part by the Development Program of China under Grant 2018YFB1004600 and Grant 2017YFC1503000, and in part by the National Nature Science Foundation of China under Grant 41701594 and Grant 41871302.

ABSTRACT Generative adversarial networks (GANs) suffer from catastrophic forgetting when learning multiple consecutive tasks. Parameter regularization methods that constrain the parameters of the new model in order to be close to the previous model through parameter importance are effective in overcoming forgetting. Many parameter regularization methods have been tried, but each of them is only suitable for limited types of neural networks. Aimed at GANs, this paper proposes a unified framework called Memory Protection GAN (MPGAN), in which many parametrization methods can be used to overcome forgetting. The proposed framework includes two modules: Protecting Weights in Generator and Controller. In order to incorporate parameter regularization methods into MPGAN, the Protecting Weights in Generator module encapsulates different parameter regularization methods into a “container”, and consolidates the most important parameters in the generator through a parameter regularization method selected from the container. In order to differentiate tasks, the Controller module creates unique tags for the tasks. Another problem with existing parameter regularization methods is their low accuracy in measuring parameter importance. These methods always rely on the first derivative of the output function, and ignore the second derivative. To assess parameter importance more accurately, a new parameter regularization method called Second Derivative Preserver (SDP), which takes advantage of the second derivative of the output function, is designed into MPGAN. Experiments demonstrate that MPGAN is applicable to multiple parameter regularization methods and that SDP achieves high accuracy in parameter importance.

INDEX TERMS Catastrophic forgetting, generative adversarial network, parameter regularization methods.

I. INTRODUCTION

Generative adversarial networks (GANs) [1] have led to significant improvements in image generation [1], [2], image deblur [3], super-resolution [4], [5], and other domains. Unfortunately, when trained on multiple tasks arriving sequentially, GANs always forget previously learned knowledge, which is known as catastrophic forgetting [6]. Catastrophic forgetting severely restricts the applications for GANs in real-world scenarios [7]–[9]. For example, it is impractical for hospitals to retain patient data permanently due to privacy regulations or expensive costs. When suffering from catastrophic forgetting, GANs would encounter the challenge of reusing learned knowledge from old patient data without storing them.

Many methods have been proposed to alleviate catastrophic forgetting. The mainstream methods for overcoming forgetting of convolutional neural networks (CNNs) [8]–[19] can generally be divided into three categories: transfer learning approaches, rehearsal mechanisms, and
parameter regularization methods. Transfer learning approaches [8]–[10] attempt to solve the problem of catastrophic forgetting by relaying previously learned information to the current model. The trouble with transfer learning approaches is that they generally require the preservation of all of the previously learned model parameters and therefore do not scale with a large number of tasks. In the rehearsal mechanisms [11]–[13], the past knowledge is regularly replayed with real samples drawn to jointly optimize the network parameters that can be used for the new task. The problem with rehearsal mechanisms is that the joint training of past data and new data is both time-consuming and working memory-intensive, especially when the number of tasks is very large. Parameter regularization strategies [14]–[19] constrain learning efficiency on certain weights of the new task based on how important they are to previous tasks in order to keep close to the previous model. Compared with transfer learning approaches and rehearsal mechanisms, parameter regularization methods do not have to preserve a past model, retrain on old data, or add an explicit memory, which is cost-effective in real-world environments. For this reason, parameter regularization strategies are proposed to solve the problem of catastrophic forgetting in GANs in this work.

Although several proposed parameter regularization methods such as Elastic Weight Consolidation (EWC) [14] and Memory Aware Synapses (MAS) [15], have significantly counteracted forgetting in certain CNNs, no unified framework exists that can overcome forgetting by various different parameter regularization methods. Aimed at GANs, to bridge this gap, this work proposes the Memory Protection GAN (MPGAN). The advantage of MPGAN is that as long as a strategy refers to parameter regularization methods, it can be used in the MPGAN framework to overcome the problem of GANs’ forgetting. To construct MPGAN, two issues must be resolved. The first is that although tasks always continuously come one after another, they should be discrete. Thus, how to differentiate each task coming in sequentially should be addressed first. The second issue is how to extract the commonality of and preserve the difference between different parameter regularization methods when merging them into one framework. To address these two issues, the Controller module and the Protecting Weights in the Generator (PWG) module are developed. By organically combining the two modules with traditional GANs, the proposed MPGAN framework is created, which can utilize various parameter regularization methods to address the problem of GANs’ forgetting.

Different parameter regularization methods [14], [15] calculate the parameter importance through different methods. Traditional parameter regularization methods may underestimate the parameter importance because they only rely on the first derivative of the output function to compute the importance. The first derivative will be close to zero when CNNs converge, but sometimes the second derivative will still exist even if the first derivative is minimal. Therefore, traditional parameter regularization methods that only use the first derivative of output function would underestimate the importance for some parameters. To compute the parameter importance more accurately, this work creates a parameter regularization approach called Second Derivative Preserver (SDP), which provides the additional second derivative on the original first derivative of output function. SDP is also incorporated into MPGAN and achieves satisfactory performance in mitigating the forgetting of GANs.

Specific contributions of this work include:

1) This work proposes a unified framework, called MPGAN, which consists of a traditional GAN model, a Controller module and a PWG model to address the problem of catastrophic forgetting in GANs. The framework can use different parameter regularization methods.

2) This work proposes a more accurate parameter regularization method, called SDP, which takes advantage of the second derivative as an amendment to the first derivative of output function. SDP is applied in MPGAN.

II. RELATED WORKS

There are three categories of methods of investigating catastrophic forgetting: transfer learning approaches, rehearsal mechanisms, and parameter regularization methods.

Transfer learning approaches [8]–[10] mitigate catastrophic forgetting by reusing previously acquired knowledge in the current model. Progressive neural networks [8] are immune to forgetting through their ability to leverage prior knowledge via lateral connections to previously learned features. Deep block-modular neural networks [9] explore a block-modular architecture for CNNs, which allows parts of the existing network to be re-used for the purpose of solving a new task without a decrease in performance when solving the original task. Transfer learning has been applied in GANs. For example, Lifelong GAN [10] employs knowledge distillation to transfer learned knowledge from previous networks to the new network. Lifelong GAN distills information from the old model to the new model by encouraging the two networks to produce similar output values or patterns given the auxiliary data as inputs. The major drawback of transfer learning is that an external working memory is needed to preserve all of the previously learned model parameters.

In rehearsal mechanisms [11]–[13], the data from old tasks are sampled from generated pseudodata and interleaved with the data of new tasks to be jointly trained in order to consolidate the old knowledge. For example, many rehearsal strategies are designed to replay previous memories in CNNs, including choosing recent memories, and choosing memories most similar, or least similar, to new ones [20]. GANs are a type of generative model and the generator has an active role in generating images. As a result, the rehearsal mechanisms have been widely used to overcome GANs’ forgetting. For example, in a Deep Generative Replay network [11], the generator is paired with a solver and they are selected to allow sequential learning on multiple tasks by generating and rehearsing fake data that mimics former training examples. When it comes to Memory Replay GANs (MeRGANs) [12],
two models that refer to the joint training with replay and replay alignment are proposed and are used to prevent forgetting by leveraging replays. Besides GANs, rehearsal mechanisms also successfully address forgetting of other generative models such as Variational Autoencoder (VAE) [21]. In lifelong generative modelling [13], a student-teacher architecture is introduced to learn and preserve all the distributions seen so far to mitigate forgetting of VAE. Most rehearsal strategies require regenerating samples from previous categories and jointly retraining new samples with these additional replayed samples. However, there are two shortcomings when employing the rehearsal strategy to overcome the GANs’ catastrophic forgetting in real-world environments [11], [12]. One is that regenerating old pseudodata from old models is highly time-consuming. The other is that training old data from the previous task together with new data is even more time-consuming, and also uses a great deal of storage space.

The parameter regularization methods [14], [15] also show good performance in overcoming forgetting in CNNs. They attempt to reduce representational overlap among tasks by regularisation such as freezing the weight of a convolutional layer and weight consolidation. The classic parameter regularization method of EWC [14] estimates how essential each parameter is at the conclusion of a task through diagonal precision provided by the diagonal of the Fisher information matrix [22]. It then prevents the model from straying too far from the parameters that are important for the task. Similar to EWC [14], MAS [15] also computes the importance of the parameters and penalizes changed important parameters, but the method of calculating the importance of parameters is based on how sensitive the predicted output function is to a change in this parameter. The intelligent synapses (IS) model [19] protects the parameters according to their importance along the task’s entire training trajectory. In IS, each parameter of the CNNs is awarded an importance measure based on how much it reduces the loss while learning tasks. These parameter regularization methods have been utilized in GANs. For example, the Continual Learning GAN [20] employs EWC to protect parameters in the generator critical for previous tasks from being replaced by those of latter tasks to solve GAN’s forgetting. EWC-GAN and IS-GAN [21] each protect important parameters in the discriminator and achieves satisfactory results in both forgetting and mode-collapse. Similarly, the self-supervised GAN [22] focuses on the discriminator and class information plays an important role in identifying real images. It encourages the discriminator to maintain useful representations as class information by adding a rotation-based loss to the original loss function of the discriminator. Although these reconstructed GANs can minimize forgetting, they only focus on one parameter regularization method, which suggests that other parameter regularization methods may fail when applied to these models. This paper attempts to create a framework that can utilize various parameter regularization methods to prevent forgetting in GANs.

### III. PRELIMINARY KNOWLEDGE

In this section, some preliminary knowledge including vanilla GANs and traditional parameter regularization methods are presented.

#### A. VANILLA GANs

Two neural networks — a generator $G$ and a discriminator $D$ are simultaneously trained in the traditional GAN framework, as shown in Figure 1. For image generation, $G$ captures the data distribution and generates a fake image as output. $D$ estimates the probability that a sample came from the training data (true image) rather than $G$. The training procedure for $G$ is to maximize the probability of $D$ making a mistake [1]. In other words, $D$ and $G$ play the following two-player minimax game with loss function (1):

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim P_z}[\log(1 - D(G(z)))].$$

where $x$ represents the true data, $P_{\text{data}}(x)$ is the distribution of $x$, $z$ is input noise variables, $P_z(z)$ is a prior on $z$, and $D(x)$ represents the probability that $x$ came from the data rather than $G$.

#### B. TRADITIONAL PARAMETER REGULARIZATION METHODS

Classic parameter regularization methods such as EWC [14] and MAS [15] have been shown to be effective in calculating parameter importance.

The EWC regularization method assumes that the parameters associated with the higher Fisher diagonal elements are considered more important [8], so it uses the Fisher information matrix to calculate parameter importance in CNNs as shown in (2):

$$I_{\text{EWC}}(\theta) = E \left[ \left( \frac{\partial}{\partial \theta} \log F(\theta) \right)^2 \right],$$

where $I_{\text{EWC}}(\theta)$ is the parameter importance calculated by EWC, $F$ is the output function, $\theta$ is the set of all parameters in CNNs.
When it comes to computing the parameter importance in the generator, it can be expressed as the following (3):

\[ I_{EWC}(\theta) = E \left[ \left( \frac{\partial}{\partial \theta} \log G(z) \right)^2 \right], \tag{3} \]

where \( z \) is a random noise and \( G \) is the generator.

MAS [10] measures the parameter importance calculated by the squared \( l_2 \) norm of the output function as shown in (4) [10]:

\[ I_{MAS}(\theta) = \frac{\partial [\| F(\theta) \|_2^2]}{\partial \theta}, \tag{4} \]

where \( I_{MAS}(\theta) \) is the parameter importance calculated by MAS, \( F \) is the output function, and \( \| F(\theta) \|_2 \) is the \( l_2 \) norm of the output function.

Accordingly, when applying MAS into the generator, the importance can be estimated through (5):

\[ I_{MAS}(\theta) = \frac{\partial [\| \log G(z) \|_2^2]}{\partial \theta}, \tag{5} \]

where \( z \) is a random noise and \( G \) is the generator.

**IV. MPGAN FRAMEWORK AND SDP**

In this section, on the basis of the above preliminary knowledge, the proposed MPGAN is described in greater detail by first analyzing the reason forgetting exists in GANs and then introducing strategies to address it, and the SDP method is discussed.

**A. CAUSE OF FORGETTING IN TRADITIONAL GANS**

As mentioned above, the generator generates images and the discriminator provides the optimization information to guide the generator’s generation through gradient descent. If the parameters in the generator are updated completely according to standard gradient descent when multiple tasks are coming sequentially, the parameters achieved on previous tasks would be erased and thus their performance would decrease abruptly, which will lead to catastrophic forgetting [6], [23]. Consider two tasks coming in consecutive. The first task (T1) is to generate dogs; the second task (T2) is to generate cats. The models of the generator and the discriminator in traditional GANs have already been trained to generate dogs (T1). When T2 arrives, the discriminator aims to distinguish the true cats’ images from the faked ones. Thus, the discriminator guides the generator to update the parameters to cater to T2. If the generator is absolutely obedient to the discriminator’s gradient descent direction, it would lose the information of the old task (T1). Thus, one way to mitigate forgetting is to prevent parameters critical for previously learned tasks from undergoing drastic changes in value. Based on this idea, MPGAN framework is created.

**B. MPGAN FRAMEWORK**

In this part, the structure, operation process and principle of the proposed MPGAN is examined.

Two issues must be addressed to solve catastrophic forgetting of GANs. The first is how to differentiate tasks coming sequentially and continuously, which is a problem that all neural networks face when attempting to overcome forgetting. The second issue, as analyzed in Section IV A, is how to prevent previously learned important information in the generator from being changed by new ones through a series of different parameter regularization methods. To solve the two issues, the Controller and the Protecting Weights in Generator (PWG) are introduced and MPGAN is created. The framework of the MPGAN is shown in Figure 2.

1) CONTROLLER

Tasks coming consecutively should be discrete and independent, thus the Controller is proposed to differentiate tasks. Specifically, the Controller receives a task as input and creates
a unique tag for this task, as shown in (6):

\[ tag_i = Controller(T_i), \]  

(6)

where \( T_i \) is the i-th task and \( tag_i \) is the tag that the controller creates for the \( T_i \).

After the tag is created, it is combined with the data of the task and sent into GAN for training. The tags have varying modalities. In the simplest case, the tags can be one-hot encoding or feature maps. More generally, they can be structured objects, such as a paragraph of natural language explaining how to solve the task. The tags are important because they are the only identity of the past task when the data of the task is inaccessible for the reason of storage cost, or privacy. The old task related to the tag will be evoked when the MPGAN receives the tag that it has encountered before. The workflow of the Controller is shown in Figure 3.

![FIGURE 3. The workflow of the controller.](image)

2) PWG

Per the analysis in Section IV A, if the parameters in the generator are protected from being completely changed by the new ones, the problem of forgetting in GANs may be alleviated. PWG is proposed to protect learned critical parameters in the generator through various parameter regularization methods.

Utilizing the idea of encapsulation service, PWG first establishes a container that encapsulates many different parameter regularization methods, including EWC, MAS and so on. Then, when the training of \( T_i \) is finished, PWG assesses the parameter importance of \( T_i \) through a selected parameter regularization method. Different parameter regularization methods have their own way in computing parameter importance. Note that no matter which method is chosen, the value of importance is set to zero if it is negative in order to reduce the resistance of learning new tasks as shown in (7):

\[ PWG(\theta_i) = \max(0, I_i), \]  

(7)

where \( \theta_i \) represents the parameters of \( T_i \) in the generator, \( I_i \) is the parameter importance of \( \theta_i \) calculated by the parameter regularization methods. \( PWG(\theta_i) \) is the final parameter importance after the negative importance is set to zero for \( T_i \).

Finally, PWG constrains the drastic movement of important parameters while allowing unimportant parameters to be updated for \( T_{i+1} \) according to their importance. PWG achieves this procedure by adding a penalty term on the original loss function of the \( T_{i+1} \) task as shown in (8):

\[
\theta_{i+1} = \arg \min_{\theta_{i+1}} L_{new} = \arg \min_{\theta_{i+1}} \left( L_{i+1} + \frac{\lambda}{2} PWG(\theta_i) \odot (\theta_i - \theta_{i+1}) \odot (\theta_i - \theta_{i+1}) \right),
\]  

(8)

where \( \theta_{i+1} \) is the parameters of \( T_{i+1} \) in the generator. \( \lambda \) represents how important the previous task is to the latter task. \( L_{i+1} \) is the original GAN loss function of \( T_{i+1} \). \( L_{new} \) is the new MPGAN loss function of \( T_{i+1} \).

Reasonably, parameters with large value in importance are salient in preserving the information of the learned task and the big perturbation in its value would cause drastic decrease on its performance, so the large change in important parameters would be punished significantly.

### C. SDP METHOD

The accuracy of calculation in parameter importance in PWG affects the results of overcoming forgetting of MPGAN.

Although both EWC and MAS shown in Section III B can estimate parameter importance, the accuracy of importance may not be high. To be specific, EWC and MAS rely only on the first derivative of the output function to approximate the true parameter importance. However, at some times the second derivative may still be high even when the first derivative of the output function is small, and thus EWC and MAS which only depend on the first derivative of output function would cause substantial underestimation of the importance for certain parameters. This work creates a new parameter importance calculation method, called SDP. Unlike EWC and MAS, the SDP method estimates parameter importance by adding the second derivative of the output function to the original first derivative.

In SDP, given a well-trained model, parameter importance is measured by how much the performance of the model degrades when a parameter \( \theta_k \) is removed, as shown in (9):

\[
\delta F = F(\theta \odot [1, \ldots, 1, 0, 1, \ldots, 1]) - F(\theta), \quad (9)
\]

where \( F \) is the output function of the model, \( \delta F \) represents how much degradation of performance the loss of parameter \( \theta_k \) results in, and \( \theta \) is the set of all parameters in the model.

Parameter \( \theta_k \) is more important when its \( \delta F \) is larger. Calculating the parameter importance \( \theta_k \) in a CNN one by one is tedious and sometimes may be computationally complex, as an alternative, \( \delta F \) in (9) can be expressed in the method of Taylor expansion for all parameters together, as shown in (10):

\[
\delta F = \left( \frac{\partial F}{\partial \theta} \right)^T \delta \theta + \frac{1}{2} \delta \theta^T \left( \frac{\partial^2 F}{\partial \theta^2} \right) \delta \theta + O(||\delta \theta||^3), \quad (10)
\]

where \( \delta F / \partial \theta \) represents the gradient on parameters, and \( \partial^2 F / \partial \theta^2 \) is the Hessian matrix on parameters.

The gradient will be close to zero when CNNs converge. As a result, the first derivative of \( \delta F \) will be too small to
obtain a precise calculation of parameter importance, and the second derivative of $\delta F$ is employed as an amendment. Hence, parameter importance in SDP is calculated by (11):

$$I_{SDP}(\theta) = \left( \frac{\partial F}{\partial \theta} \right)^T \delta \theta + \frac{1}{2} \delta \theta^T \left( \frac{\partial^2 F}{\partial \theta^2} \right) \delta \theta,$$

(11)

Fisher information is closely related to the curvature of the log likelihood function, as measured by its Hessian matrix $[22]$. Due to the complexity of the calculation of the Hessian matrix, in practice, the Fisher information is introduced to approximate the Hessian matrix by (12):

$$I_{SDP}(\theta) = \left( \frac{\partial F}{\partial \theta} \right)^T \delta \theta + \frac{1}{2} \delta \theta^T E \left[ \left( \frac{\partial F}{\partial \theta} \right)^2 \right] \delta \theta,$$

(12)

where $E[(\frac{\partial F}{\partial \theta})^2]$ represents the Fisher information.

**D. THE WORKFLOW OF MPGAN**

In this section, the workflow of MPGAN is summarized. On the first task, MPGAN is trained and tested as the vanilla GAN is, except for the task tag supplied to the generator. The importance of parameters in the generator is evaluated according to the chosen regularization method. On further tasks, MPGAN is trained with different task tags and with the regularization of parameter importance with respect to the previous tasks. After each new stage, the performance on the tasks up to this stage can be tested by supplying the correct tags.

**Algorithm 1 Training the Model at the i-Th Task**

**Input:** The added data of new task $T_i$.

**Require:** The model of previous task $T_{i-1}$

1: if $i = 0$ then
2: Create a tag for $T_i$ according to equation (6)
3: Train MPGAN to perform $T_i$ according to equation (1)
4: else
5: Create a tag for $T_i$ according to equation (6)
6: Select a parameter regularization method from EWC, MAS or SDP
7: Calculate parameter importance of $T_{i-1}$ according to equation (3), equation (5) or equation (12)
8: Update the MPGAN to perform $T_i$ according to equation (8)

The detailed training processes of MPGAN is shown in Algorithm 1 and the task of image generation is used as an example. Sequential multi-tasking image generation can be expressed as $\{T_0, \ldots, T_i\}$. For $T_{i-1}$, the Controller creates a tag $tag_{i-1}$ for it. Then, $T_{i-1}$ is sent to the MPGAN for training. After $T_i$ comes, the controller creates a tag $tag_i$ for it, and PWG begins its operation. It first chooses a regularization method to calculate the parameter importance $I_{i-1}$ of the generator. When training $T_i$, PWG protects the parameters in the generator to retain old values proportional to their $I_{i-1}$.

For testing, if previous $\{T_1, \ldots, T_{i-1}\}$ are required to be performed again, the controller sends their tags to MPGAN to evoke its corresponding memories.

**V. EXPERIMENTS AND RESULTS**

**A. DETAILS OF EXPERIMENTS**

GAN models used in MPGAN: Three different GAN models were selected to demonstrate the effectiveness of MPGAN. These GAN models can all generate high quality images. But they suffer from catastrophic forgetting when comes to continual image generation. The three GAN models were therefore used to check the effectiveness of the MPGAN framework in overcoming catastrophic forgetting.

- WGAN-GP [24]: It is known for its ability to stabilize training.
- InfoGAN [25]: It can disentangle representation which is helpful for learning salient features of data.
- SAGAN [26]: It can generate high-resolution details by using cues from all feature locations.

Baseline: MPGAN was compared with four baselines.
- Joint learning (JL) [27]: all tasks are trained jointly in a non-sequential setting. JL is the upper boundary because no forgetting exists in it.
- Stochastic Gradient Descent (SGD) [27]: tasks are learned in a sequential setting without any strategy to prevent forgetting. SGD is the lower boundary.
- Deep Generative Replay network (DGR) [11]: Images generated by a generator trained on previous tasks are combined with the training images for the current task to form a hybrid training set. This belongs to the rehearsal mechanisms.
- Lifelong GAN (LLG) [10]: It prevents forgetting by utilizing knowledge distillation to transfer learned knowledge from previous networks to the new network. This belongs to the transfer learning methods.

**TABLE 1. The usage of the four datasets.**

| Dataset | CelebA+Anime-Face | MNIST | SVHN |
|---------|------------------|-------|------|
| Task 1  | CelebA-face      | Digits 0,1,2,3,4 | Digits 0,1,2,3,4 |
| Task 2  | Anime face       | Digits 5,6,7,8,9 | Digits 5,6,7,8,9 |

Dataset: Four image generation datasets were selected to verify the effectiveness of MPGAN. The usage of the four datasets is shown in Table 1.

- MNIST [28]: It contains hand-written digits 0 to 9.
- SVHN [29]: It contains digits 0 to 9 obtained from house numbers in Google Street View images.
- CelebA [30]: It is a large-scale facial attributes dataset with more than 200,000 images of celebrities.
- Anime-Face-Dataset $^1$: It is a collection of high-quality anime faces.

In the following experiments, MNIST and SVHN are divided into two parts for the datasets of two continual tasks respectively.

$^1$ Anime-Face Dataset are collected from the website www.getchu.com.
Evaluation: The performance of MPGAN in overcoming forgetting was measured through four metrics.

- Average classification accuracy (ACC) [31]: ACC pertains to the accuracy of the classifier network trained on real images and evaluated on generated images. In our experiments, ACC is used to measure the quality of samples in previous tasks. Intuitively, if forgetting occurs, MPGAN would not regenerate previous samples clearly and properly, and the ACC would be low, and vice versa. Thus, the ACC can be used as a metric to measure the forgetting of MPGAN. Higher ACC indicates better quality of samples in previous tasks and less forgetting.
- Frechet Inception Distance (FID) [32]: FID is a commonly used metric for evaluating generative models. It compares the statistical properties of generated samples to real samples. A lower FID score indicates more realistic images that match the statistical properties of real images [33]. FID is sensitive to both quality and diversity.
- Backward Transfer (BWT) [31]: BWT measures the influence that learning a latter task has on the performance on a former task. It describes how much the new task weakens the previous tasks.
- Computational overhead: It includes training time and memory size. They measures how much time and memory size different models require during the training.

MPGAN can overcome forgetting more significantly when ACC and BWT are higher and FID and computational overhead are lower.

The formulas for ACC, BWT and FID are (13), (14), and (15), respectively:

$$\text{ACC} = \frac{1}{T} \sum_{i=1}^{T} R_{T,i},$$

$$\text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} R_{T,i} - R_{T,i},$$

$$\text{FID} = \| \mu_T - \mu_G \|^2 - Tr(\Sigma_r - \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}),$$

where $T$ is the number of tasks, and $R_{i,j}$ is the test classification accuracy of the model on task $t_j$ after observing the last sample from task $t_i$.

As shown in Figure 4, when training MPGAN\textsubscript{WGAN-GP} for the second task, the regenerated samples of the celebrity faces in the first task remained outlines and textures even in the 30-th epoch in all of the MPGAN\textsubscript{WGAN-GP} models, while the celebrity faces trained with SGD vanished in the 3rd epoch. Among all MPGAN\textsubscript{WGAN-GP} models, MPGAN\textsubscript{EWC\_WGAN-GP} produced less distinct samples than MPGAN\textsubscript{MAS\_WGAN-GP} and MPGAN\textsubscript{SDP\_WGAN-GP}. In addition, ACC and FID were employed to quantitatively analyze the regenerated samples of the celebrity faces, as shown in Tables 2 and 3. The ACC of regenerated celebrity faces in all MPGAN\textsubscript{WGAN-GP} models ranged from 65% to 92%, but the ACC of SGD declined sharply to only 1.73% and 0.05% in the 10-th and 30-th epochs, respectively. Similarly, the marked contrast in FID between all
of the MPGANWGAN-GP models and SGD illustrate that MPGANWGAN-GP models could significantly maintain their original data distributions of previously learned samples.

In conclusion, the presented regenerated samples in the first task indicate that MPGANWGAN-GP can, to a certain extent, hinder important old parameters from drifting to new values. In addition, MPGANSDPWGAN-GP can best memorize the previous samples for its most accurate calculation of importance of parameters in all of the MPGANWGAN-GP models.

C. CONTINUOUS IMAGE GENERATION RESULTS

If catastrophic forgetting is prevented, MPGAN would not completely forget previous knowledge. Therefore, MPGAN may learn a continual sequence of tasks. In this section,
the ability of MPGAN to learn continual image generation tasks was evaluated.

InfoGAN and SAGAN were selected as the GAN models in MPGAN and the MPGAN was called MPGAN\textsubscript{InfoGAN} and MPGAN\textsubscript{SAGAN}, respectively. MPGAN\textsubscript{InfoGAN} (MPGAN\textsubscript{SAGAN}) could be specified into MPGAN\textsubscript{EWC}, MPGAN\textsubscript{MAS} and MPGAN\textsubscript{SDP} (MPGAN\textsubscript{EWC}, MPGAN\textsubscript{MAS} and MPGAN\textsubscript{SDP}), respectively. MNIST and SVHN were used as the dataset for MPGAN\textsubscript{InfoGAN} and MPGAN\textsubscript{SAGAN}, respectively. The experimental process was as follows. The dataset was divided into two parts: one was to generate digits 0, 1, 2, 3, and 4 as the first task and one was to generate digits 5, 6, 7, 8, and 9 as the second task. MPGAN trained the two tasks coming in sequentially as shown in Algorithm 1. Regarding testing, the tags of the two tasks were fed into MPGAN, in order to check whether the samples of the old and new task can be regenerated. If the regenerated samples for both tasks were satisfactory, MPGAN’s ability to learn continual image generation tasks could be validated.

The regenerated results of MNIST and SVHN are shown in Figure 5 and Figure 6, respectively. The ten blocks in Figure 5 and Figure 6 represent the regenerated digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9, respectively. In each block, the first column is the results of JT; the second, the third and the
fourth column, which are framed in the red box, are the results of proposed MPGAN models that include MPGAN\textsubscript{InfoGAN} (MPGAN\textsubscript{SDP}), MPGAN\textsubscript{MAS} (MPGAN\textsubscript{SDP} SAGAN) and MPGAN\textsubscript{EWC} (MPGAN\textsubscript{SDP} SAGAN) from left to right, respectively; the fifth is the results of LLG; the sixth is the results of DGR and the last is the results of SGD. The results show that the previously learned digits 0 to 4 generated by SGD are hardly recognizable because SGD completely forgets acquired knowledge. In contrast, as shown in Figure 5 and Figure 6, all MPGAN\textsubscript{InfoGAN} and MPGAN\textsubscript{SAGAN} models generate proper and clear digits in the first task, which demonstrates that MPGAN would not forget acquired digits. In addition, the satisfactorily regenerated digits 5 to 9 demonstrate that MPGAN could correctly perform the second image generation task again.

Table 4 and Table 5 display the ACC, FID, BWT, and computational cost of MNIST and SVHN, respectively. Reasonably, SGD completely forgets previous digits while all MPGAN\textsubscript{InfoGAN} and MPGAN\textsubscript{SAGAN} models show little forgetting and thus their ACC are all above 91% and 83%, respectively.

Among all MPGAN\textsubscript{InfoGAN} models, MPGAN\textsubscript{SDP} achieves the best results. It outperforms LLG and DGR by 3.84% and 2.75% and outperforms the other two MPGAN\textsubscript{InfoGAN} models MPGAN\textsubscript{MAS} and MPGAN\textsubscript{EWC} 2.26% and 5.36% on the first task, respectively. Similarly, MPGAN\textsubscript{SDP} SAGAN achieves the higher ACC than other baselines except the JL method.

As expected, the FID value shows that all MPGAN\textsubscript{InfoGAN} and MPGAN\textsubscript{SAGAN} models experience marginal forgetting and can generate previous digits that are recognizable and diverse. In particular, MPGAN\textsubscript{SDP} InfoGAN performs 1.28 and 2.98 better than two strong baselines LLG and DGR respectively. MPGAN\textsubscript{SDP} SAGAN performs 10.96 and 15 better than LLG and DGR respectively. BWT indicates that all MPGAN\textsubscript{InfoGAN} models and MPGAN\textsubscript{SAGAN} models can achieve better trade-off between memorizing previous tasks and learning new ones. MPGAN\textsubscript{SDP} InfoGAN only forgets the smallest amount of past knowledge of previous tasks (−0.77%) compared with LLG (−4.81%) and DGR (−2.69%) while SGD loses the most (−18.97%). MPGAN\textsubscript{SDP} SAGAN also forgets the least past knowledge of previous tasks (−1.34%).

In addition, the computational overhead including the training time per epoch in the second task and parameters of different methods were measured. JL consumes large memory because it requires that all data are available and trained together, which is impractical in real scenarios. DGR also occupies a number of memory and time because it regenerates some portions of previous data and trains them with new data together. LLG does not need as much memory as JL and DGR, but the distillation loss also requires the preservation of old data as auxiliary data and the distillation loss itself has to store the old output results. Compared with the JL, rehearsal mechanisms and transfer learning approaches, MPGAN which uses parameter regularization methods consumes just a half or a quarter of time and memory. The most memory and time-consuming part lies in the calculation of parameters importance and their storage.
FIGURE 7. The first task’s ACC of MPGAN models with different λ on MNIST and SVHN.

were tested on the first task, and red boxes were placed around the best values of λ as shown in Figure 7. The best value of λ is the one that can assist MPGAN in regenerating previous samples with the highest ACC. It is determined that λ = 50, λ = 30 and λ = 30 could best counteract forgetting on MNIST for MPGAN\textsuperscript{EWC}, MPGAN\textsuperscript{MAS} and MPGAN\textsuperscript{SDP} models, respectively because the regenerated samples have the highest ACC when these values of λ are used. In addition, when λ = 100, λ = 150, and λ = 150, the regenerated digits in the first task on SVHN have the highest ACC for MPGAN\textsuperscript{EWC}, MPGAN\textsuperscript{MAS} and MPGAN\textsuperscript{SDP} models, respectively. From Figure 7, it is also indicated that small values of λ would cause a rapid deviation from the models’ previous task values when starting to train the next task. Large values of λ would interfere with new tasks and have no obvious improvement on the quality of images.

In summary, the qualitative and quantitative experimental results of MPGAN\textsuperscript{InfoGAN} and MPGAN\textsuperscript{SAGAN} on MNIST and SVHN illustrate that MPGAN can recollect the past knowledge and achieve continual image generation. But the overall performance of MPGAN\textsuperscript{InfoGAN} is better than that of MPGAN\textsuperscript{SAGAN}. Specifically, the ACC on the first task of MPGAN\textsuperscript{InfoGAN} are 8%-10% higher than that of MPGAN\textsuperscript{SAGAN}. The FID of MPGAN\textsuperscript{InfoGAN} is 11-20 lower than that of MPGAN\textsuperscript{SAGAN}. MPGAN\textsuperscript{InfoGAN} forgets less 0.2%-1.6% knowledge of the past than MPGAN\textsuperscript{SAGAN} from the values of BWT. The main reason for inferior performance on SVHN is that the images in SVHN are more complicated in color, texture and background, so preserving the detailed information is more difficult.

In this experiment, the successful regeneration of former and latter tasks demonstrates that MPGAN is capable of learning new knowledge while remembering old information, which means that MPGAN can achieve continual image generation on MNIST and SVHN. It also indicates that SDP can assist MPGAN in producing samples with higher visual fidelity.

VI. DISCUSSION

This paper introduces MPGAN which combines GANs with the Controller and PWG modules to address catastrophic forgetting in GANs by parameter regularization methods. Throughout experiments firstly it is demonstrated that MPGAN is applicable to many different parameter regularization methods such as EWC, MAS and the proposed parameter regularization method SDP. The previously regenerated samples are clear and diverse in MPGAN. ACCs of them are always more than 80%. FIDs are all below 81. BWTs are almost approximately -3 to 0. Then experimental results also indicate that SDP can assess parameter importance in a more accurate manner and thus can overcome forgetting more significantly. MPGAN augmented with SDP always outperforms MPGAN enhanced with EWC and MAS by improving approximately 2%-5% on ACC, 3%-13% on FID and 31%-74% on BWT respectively, and it even outperforms some state-of-the-art methods e.g., DGR and LLG. In addition, MPGAN using parameter regularization methods occupies less time and memory than rehearsal mechanisms such as DGR and transfer learning such as LLG.

Although MPGAN is able to overcome the problem of catastrophic forgetting in the experiments above, it may not succeed in doing so with other datasets. Because parameter regularization methods assume that the data distribution of images in different tasks can be united into one joint distribution, and they attempt to reduce the representational overlap among tasks by use of a weight freezing or weight consolidation strategy. However, there are three problems in that assumption. First, when the data distributions between tasks are largely different, their distributions may have little overlap. This indicates that there exists no one set of parameters configuration that can fit the data distributions of both the old task and the new task. Second, when the model is updated by the new data, it would be restricted to finding a local optimum point only around the parameter space already optimized by old tasks, and it would cause other points with better joint distribution for both the new task and old...
task to be missed or ignored. Third, parameter regularization methods approximate the unified joint distribution by various approaches, such as Fisher information matrix in EWC and information entropy in the proposed SDP, but the results of these approximations are not very accurate. This means that the joint distribution obtained by various parameter regularization methods is not the true distribution and may contain egregious errors.

VII. CONCLUSION

In this paper, a framework called MPGAN to encapsulate various parameter regularization methods to address the problem of GANs’ catastrophic forgetting when learning multiple consecutive tasks is introduced. In addition, a new parameter regularization method, called SDP, which estimates parameter importance in calculating parameter importance in MPGAN. However, further improvements can still be made on making MPGAN capable of more complex continual tasks such as continual style transferring [34] and continual face recognition [35].

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2014, pp. 2672–2680.

[2] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier GANs,” in Proc. 34th Int. Conf. Mach. Learn., vol. 70, Aug. 2017, pp. 2642–2651.

[3] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and I. Matas, “DeblurGAN: Blind motion deblurring using conditional adversarial networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 8183–8192.

[4] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a Laplacian pyramid of adversarial networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2015, pp. 1486–1494.

[5] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, “Photo-realistic single image super-resolution using a generative adversarial network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4681–4690.

[6] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in Psychology of Learning and Motivation, vol. 24, G. H. Bower, Ed. New York, NY, USA: Academic, 1989, pp. 109–165.

[7] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” Neural Netw., vol. 113, pp. 54–71, May 2019, doi: 10.1016/j.neunet.2019.01.012.

[8] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” 2016, arXiv:1606.04671. [Online]. Available: https://arxiv.org/abs/1606.04671.

[9] A. V. Terekhov, G. Montone, and J. K. O’Regan, “Knowledge transfer in deep block-modular neural networks,” in Proc. 14th Int. Conf. Biomimetic and Biohybrid Syst., vol. 9222, S. Wilson, P. Verschure, A. Mura, and T. Prescott, Eds. Newton County, MO, USA: Springer, Jul. 2015, pp. 268–279.

[10] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, “Lifelong GAN: Continual learning for conditional image generation,” in Proc. IEEE Int. Conf. Comput. Vis., Oct. 2019, pp. 2759–2768.
Y. Chang et al.: MPGAN: A Framework to Overcome the Forgetting of GANs Using Parameter Regularization Methods

[34] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis., Oct. 2017, pp. 2223–2232.

[35] M. Turk and A. Pentland, “Face recognition using eigenfaces,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Mar. 1991, pp. 586–591.

YIFAN CHANG is currently pursuing the doctor’s degree with the College of Information Science and Technology, University of Science and Technology of China. Her research interests include generative adversarial networks, continual learning, and lifelong learning.

WENBO LI received the B.S. degree in water resource from the North China University of Water Conservancy and Electric Power, Zhengzhou, China, in 2003, and the Ph.D. degree in spatial information science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2008. He is currently an Associate Professor with the Hefei Institute of Technology Innovation, Chinese Academy of Sciences, Hefei, China. He has authored over 30 journal articles. He is also a reviewer of many journals. His current research interests include remote sensing, deep learning, generative adversarial networks, and natural disasters.

JIAN PENG is currently pursuing the Ph.D. degree in cartography and geographical information system with Central South University, Changsha, China. His research interest includes the image intelligent understanding inspired by brain memory mechanism. His research interests include neural computation, continual learning, and remote sensing.

HAIFENG LI (Member, IEEE) received the master’s degree in transportation engineering from the South China University of Technology, Guangzhou, China, in 2005, and the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, in 2009. He was a Research Associate with the Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong, in 2011, and a Visiting Scholar with the University of Illinois at Urbana-Champaign, Urbana, IL, USA, from 2013 to 2014. He is currently a Professor with the School of Geosciences and Info-Physics, Central South University, Changsha, China. He has authored over 30 journal articles. He is a reviewer of many journals. His current research interests include geo/remote sensing big data, machine/deep learning, and artificial/brain-inspired intelligence.

YU KANG (Senior Member, IEEE) received the Dr.Eng. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2005. From 2005 to 2007, he was a Postdoctoral Fellow with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. He is currently a Professor with the State Key Laboratory of Fire Science, Department of Automation and Institute of Advanced Technology, University of Science and Technology of China.

YINGLIANG HUANG received the bachelor’s degree in economics and the master’s degree in western economics from Central China Normal University, Wuhan, China, in 2004 and 2007, respectively. She is currently an Assistant Researcher with the Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, China. Her current research interests include emergency management, artificial intelligence, and public policy.

* * *