GopGAN: Gradients Orthogonal Projection Generative Adversarial Network With Continual Learning

Xiaobin Li, *Graduate Student Member, IEEE*, and Weiqiang Wang, *Member, IEEE*

***Abstract*— The generative adversarial networks (GANs) in continual learning suffer from catastrophic forgetting. In continual learning, GANs tend to forget about previous generation tasks and only remember the tasks they just learned. In this article, we present a novel conditional GAN, called the gradients orthogonal projection GAN (GopGAN), which updates the weights in the orthogonal subspace of the space spanned by the representations of training examples, and we also mathematically demonstrate its ability to retain the old knowledge about learned tasks in learning a new task. Furthermore, the orthogonal projection matrix for modulating gradients is mathematically derived and its iterative calculation algorithm for continual learning is given so that training examples for learned tasks do not need to be stored when learning a new task. In addition, a task-dependent latent vector construction is presented and the constructed conditional latent vectors are used as the inputs of generator in GopGAN to avoid the disappearance of orthogonal subspace of learned tasks. Extensive experiments on MNIST, EMNIST, SVHN, CIFAR10, and ImageNet-200 generation tasks show that the proposed GopGAN can effectively cope with the issue of catastrophic forgetting and stably retain learned knowledge.**

***Index Terms*— Catastrophic forgetting, continual learning, generative adversarial networks (GANs), orthogonal projection matrix.**

I. Introduction

GENERATIVE adversarial networks (GANs) [6] are the game of two players, i.e., generator and discriminator, which are widely applied in synthesizing high-quality data. For image synthesis, the generator takes latent vectors as input and is expected to generate high-quality synthesized images, whereas the discriminator is expected to effectively discriminate them from real images. With more effective losses [1], [7], [22] and new architectures [16], [27], [28], GANs have shown the remarkable generative ability.

Conditional GANs [24] are able to generate semantic images by embedding the conditions into the input of generator and discriminator. The various conditions in CGANs include label [27], text [41] and images [14], [42], and labels are the most common one [27].

It is unrealistic for agents to obtain all training data and learn once in practice, and we expect agents to acquire knowledge such as human beings, i.e., constantly learn in their lives and effectively learn new tasks quickly without forgetting the knowledge learned in the past. To achieve general artificial intelligence, it is necessary to endow agents with continual learning ability. However, the experiments show that it is still challenging now for deep neural networks to reserve the knowledge obtained in previously learned tasks when learning a new task, known as catastrophic forgetting [4], [23]. GANs also seriously suffer from catastrophic forgetting. For example, in sequential tasks training, a generator can only generate satisfactory images for the last task and forgets the ability ever owned for other previous tasks.

Recent years have witnessed some efforts of researchers to mitigate catastrophic forgetting. For example, the conceptor-aided backprop (CAB) method [10] and the orthogonal weight modification (OWM) method [39] have been proposed to cope with the issue in a novel way for continual learning. Concretely, an orthogonal projection matrix in each layer of networks is constructed to preserve the learned knowledge, and during learning a new task, the gradients in neural networks are projected to the orthogonal direction of all previously learned features by the orthogonal projection matrix. The updated weights can not only fit the new task but also retain the validity of encoding the old knowledge. However, for CGANs, the inputs of a generator are random noises independently sampled from the same distribution except label condition, and it brings about the practical disappearance of free memory space for unlearned tasks (we will explain the point in Section III-C).

Inspired by the OWM algorithms [10], [39], we present a new conditional GAN, called gradients orthogonal projection GAN (GopGAN), to generate synthetic data in the way of continual learning, and apply it for the sequential learning task of image synthesis. Fig. 1 shows the schematic of GopGAN. Here, we summarize the main contributions of this article as follows. First, we give a rigorous mathematical proof to show updating weights in the orthogonal subspace of feature space can guarantee that the learning of a new task does not result in forgetting the knowledge acquired in old tasks. Then, by defining and solving an optimization problem, we derive a new optimal solution of projection matrix. Furthermore,

Manuscript received 14 July 2020; revised 3 January 2021 and 31 May 2021; accepted 25 June 2021. Date of publication 16 July 2021; date of current version 5 January 2023. This work was supported in part by the National Key Research and Development Program of China under Contract 2017YFB1002203, in part by NSFC under Grant 61976201, in part by the NSFC Key Projects of International (Regional) Cooperation and Exchanges under Grant 61860206004, and in part by the Ningbo 2025 Key Project of Science and Technology Innovation under Grant 2018B10071. *(Corresponding author: Weiqiang Wang.)*

The authors are with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: lixiaobin161@mails.ucas.ac.cn; wqwang@ucas.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3093319.

Digital Object Identifier 10.1109/TNNLS.2021.3093319

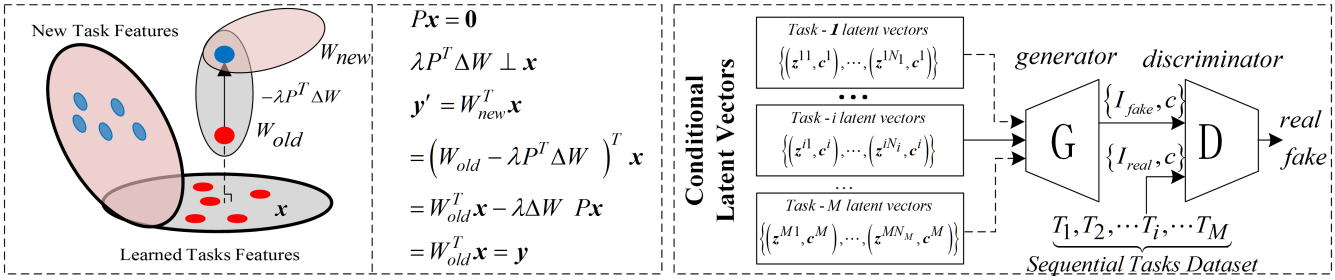


Fig. 1. GopGAN. Left: weights updating schematic. Red points in big ellipse are the input features from the learned tasks and the red point in small ellipse is the corresponding weights. Blue points are the input features from the new task to be learned. When learning new task, old weights will be updated in the orthogonal direction of learned features so that the network will keep the learned features unchanged even after learning new task completely. The formula on the right is easy to understand the updating mechanism of the network, where $\boldsymbol{y}$ is the response of the network to the feature $\boldsymbol{x}$ of the learned tasks(detailed derivation can be seen in Section III-B). Right: GopGAN architecture with conditional latent vectors(detailed in III-C).

we present an iterative calculation algorithm for continual learning so that features of training examples in the learning of past tasks do not need to be stored for learning a new task. Second, for GANs, we find that the noise input of generator comes from the same distribution and is shared by all the categories, which makes the calculated projection matrix prone to approach zero matrix, whose proof can be seen in Section III-B and the following supplementary proof in Appendix A, and then, little memory space is available for new tasks. To address the problem, we propose a new task-dependent latent vector to make each task have a different latent vector so as to obtain enough memory space for new tasks.

The remaining parts of this article are organized as follows. Section II introduces the related works for GANs with continual learning. In Section III, we first introduce the concept of gradients orthogonal projection and give the mathematical explanation of the property in learning a new task, which can effectively overcome the forgetting of the knowledge acquired in old tasks. Then, by defining an objective function with respect to projection matrix P, we derive a new solution of orthogonal projection matrix P. Furthermore, the corresponding iterative formula is mathematically derived and an efficient computing algorithm is given. To better combine the proposed technique of weight updates with CGANs, we finally describe the proposed task-dependent latent vectors. In Section IV, we discuss the relationship between the proposed method GopGAN and other methods including sequential fine-tuning (SFT), EWC [18], OWM [39], and LifelongGAN [40]. In Section V, we utilize FID [11], average accuracy (A) [2], forgetting measure (F) [2], intransigence measure (I) [2], and training data occupation (D) to evaluate the different aspects of the generation and extensive experiments have been carried out to evaluate the proposed GopGAN by comparing it with the state-of-the-art GANs on MNIST, SVHN, EMNIST, CIFAR10, and ImageNet-200 generation tasks. Section VI concludes this article.

II. Related Works

In recent years, generative adversarial networks (GANs) [6] have shown their effectiveness in images generation. However, without supervisory conditions, the categories of generated images are unpredictable. Mirza and Osindero [24] proposed a conditional GAN (CGAN) to generate semantic images with conditions. We adopt CGAN with label condition as our basic network architecture.

Continual learning aims to endow agents with the capacity of learning sequential tasks without forgetting the knowledge acquired in previous tasks. Recent years have witnessed some efforts, and most of them focus on the modification of weights' gradients in network learning. Kirkpatrick *et al.* [18] proposed a method called elastic weight consolidation (EWC), which introduced weights constraint between new task and old tasks. However, in the sequential tasks training, the network with EWC will retain a Fisher matrix for each learning task, and the accumulation of Fisher regularizers will overly constrain the network parameters, resulting in the learning impairment of new tasks. Schwarz *et al.* [31] proposed an improved EWC called Online EWC that is scalable to a large number of tasks. Similar to Online EWC, Chaudhry and Dokania [2] presented EWC++, which is slightly different from Online EWC in updating Fisher matrix. The pioneering works, CAB [10] and OWM [39], are the variants of the backpropagation algorithm and have been successfully applied in classification tasks, but we find that there is no work to apply these two methods to GANs. CAB and OWM are two similar methods of addressing catastrophic forgetting by updating networks weights along the orthogonal direction of feature space of learned tasks. The difference between CAB and OWM is the design of the projection matrix. CAB constructs the projection matrix based on the Conceptor [15], and OWM is based on the recursive least square (RLS) algorithm [5], [36]. Different from CAB and OWM, we redefine the form of orthogonal projection matrix and apply it to GANs.

GANs also suffer from catastrophic forgetting in continual learning [32], [35]. Seff *et al.* [32] applied the EWC to GANs, which to some extent improved the quality of generated images. Wu *et al.* [37] proposed memory replay GANs (MeRGANs) based on the memory replay mechanisms [33] to mitigate the forgetting of previous categories. In MeRGANs, two methods, joint training with replay (MeRGAN-JTR) and replay alignment (MeRGAN-RA), are proposed. However, both of them require a lot of memory space to replay the old tasks, which will occupy lots of space and time in network training. Based on MeRGANs, Rios and Itti [29] proposed a cumulative closed-loop memory replay GAN (CloGAN),

which decreases the memory space by setting the clusters of each old task. Compared to MeRGAN, CloGAN saves a lot of memory space but slightly reduces the quality of generated images. Essentially, MeRGANs and CloGAN are the methods of retraining the old tasks not retaining the learned knowledge, which is the biggest difference from our method. The proposed GopGAN can preserve the learned knowledge and does not require the replays of old tasks. Zhai *et al.* [40] proposed LifelongGAN using the method of knowledge distillation [13] to transfer the knowledge learned from the previous network to the new network. Although knowledge distillation can make the new network inherit the knowledge learned from the previous network to a certain extent, it will damage the response of the current network to previous tasks. According to the constraint rules of continual learning, this method cannot use the previous task data for knowledge distillation, which leads to the deviation of the response of the network to the previous task data after learning the new task. Ye and Bors [38] proposed a lifelong VAEGAN to induce a powerful generative replay network to learn meaningful latent representations, benefiting representation learning. They also applied their generative replay network in unsupervised, semisupervised, and supervised learning and achieved a good result. We give a strict theoretical proof that our method can keep the input response of the new network to the learned task unchanged, thus eliminating the catastrophic forgetting problem, which is also proved by extensive experiments. In Section IV, we discuss the other lifelong GANs and our method in details.

III. Gradients Orthogonal Projection GANs

A. Continual Learning Settings for GANs

The proposed GopGAN uses CGAN [24] as the base network architecture to generate semantic images. A CGAN consists of generator G and discriminator D. The generator G takes a latent vector $\boldsymbol{z}$ and label condition $\boldsymbol{c}$ as input to generate fake images $\tilde{I}$, i.e., $\tilde{I} = G(\boldsymbol{z}|\boldsymbol{c})$, where $\boldsymbol{z}$ is sampled from random Gaussian noise distribution, and $\boldsymbol{c}$ is a fixed one-hot vector for each specific task in continual learning. The discriminator D receives conditional vector $\boldsymbol{c}$ and real training image I or fake image $\tilde{I}$ generated by G and outputs a probability scalar to show that the input image is real or fake. The optimization objective function of CGAN network can be formulated as

$$\min_G \max_D V(D,G) = \mathbb{E}_{I\sim p_{\text{data}}}[\log D(I|\boldsymbol{c})] + \mathbb{E}_{\boldsymbol{z}\sim p_z}\left[\log\left(1 - D(G(\boldsymbol{z}|\boldsymbol{c}))\right)\right] \quad (1)$$

where p_{data} and p_z denote the distributions of real data and latent vectors, respectively.

When we apply GANs to generate multiple-category fake data, it can be modeled as a learning problem of sequential tasks or continual learning. In continual learning, the learning order can be arbitrary. For the sake of simplification, suppose that we have M learning tasks and they are denoted by T_i, $i = 1, 2, \ldots, M$, where i just corresponds to the learning order. Each task corresponds to a single category, so its training data D_i have the same label $\boldsymbol{c}_i$. Only when task T_i has been finished completely, can task T_{i+1} start to train, and meanwhile, we expect that the training data $\cup_{k=1}^{i} D_k$ in learned tasks are unavailable to task T_{i+1}.

Algorithm 1 GopGAN Training Algorithm. Train the Generator G and Discriminator D With Sequential Tasks $T_1, T_2, \ldots, T_M$

Input: M-the number of tasks;
λ_i-the learning rate for each task;
L-the number of layers in generator;
B-training batch size;
N-training steps;
K-the number of examples to update P;
α-the hyper-parameter in updating P.

Initialize weights W_0^D in D.
For each layer in G, initialize weight matrix $W_{l,0}^G$ and orthogonal projection matrix $P_{l,0} = \boldsymbol{I}$.
for $i = 1, 2, \ldots, M$ **do**
  Get a batch of label conditions $\{\boldsymbol{c}^j\}_{j=1}^B$ with label i;
  $\hat{W}_{l,i}^G(0) = W_{l,i-1}^G$, for $l = 1, \ldots, L$;
  $\hat{W}_i^D(0) = W_{i-1}^D$;
  for $t = 1, 2, \ldots, N$ **do**
    # *update discriminator*
    Sample $\{I^j\}_{j=1}^B$ a batch from task T_i;
    Sample $\{\boldsymbol{z}^j\}_{j=1}^B$ a batch of task-dependent latent vectors;
    Calculate gradients $\Delta W_i^D(t) = \nabla_{W^D} - V(D,G)$ with inputs $\{< \boldsymbol{c}^j, I^j, \boldsymbol{z}^j >\}_{j=1}^B$;
    $\hat{W}_i^D(t) = \hat{W}_i^D(t-1) - \lambda_i \Delta W_i^D(t)$;
    # *update generator*
    Sample $\{\boldsymbol{z}^j\}_{j=1}^B$ a batch of conditional latent vectors;
    For $l = 1, 2, \ldots, L$, calculate gradients $\Delta W_{l,i}^G(t) = \nabla_{W_l^G} V(D,G)$ with inputs $\{< \boldsymbol{c}^j, \boldsymbol{z}^j >\}_{j=1}^B$;
    $\Delta \hat{W}_{l,i}^G(t) = P_{l,i-1}^T \Delta W_{l,i}^G(t)$, for $l = 1, \ldots, L$;
    $\hat{W}_{l,i}^G(t) = \hat{W}_{l,i}^G(t-1) - \lambda_i \Delta \hat{W}_{l,i}^G(t)$
  end for
  # *recursively update orthogonal projection matrix* P
  $\hat{P}_{l,i}(0) = P_{l,i-1}$;
  for $k = 1, 2, \ldots, K$ **do**
    Sample task-dependent latent vector $\boldsymbol{z}^k$ and label condition vector $\boldsymbol{c}^k$ with label i;
    Calculate feature $\boldsymbol{x}_l^k$ for each layer in G;
    $\hat{P}_{l,i}(k) = \hat{P}_{l,i}(k-1) - \frac{\hat{P}_{l,i}(k-1)\boldsymbol{x}_l^k \boldsymbol{x}_l^{kT} \hat{P}_{l,i}(k-1)}{\alpha + \boldsymbol{x}_l^{kT} \hat{P}_{l,i}(k-1)\boldsymbol{x}_l^k}$
  end for
  Let $W_{l,i}^G = \hat{W}_{l,i}^G(N)$, $W_i^D = \hat{W}_i^D(N)$ and $P_{l,i} = \hat{P}_{l,i}(K)$
end for

B. Gradients Orthogonal Projection

In sequential tasks learning, we suppose that the GAN has finished learning tasks $T_1, T_2, \ldots, T_{i-1}$ and just start to learn task T_i now. Its generator consists of L layers. The lth layer's input features of examples from datasets of learned tasks $\mathcal{D}_{i-1} = \cup_{k=1}^{i-1} D_k$ are collected into a matrix $A_l =$

$[\boldsymbol{x}_l^1, \boldsymbol{x}_l^2, \ldots, \boldsymbol{x}_l^{N_{i-1}}]$, where column vector $\boldsymbol{x}_l^j$ with length d_l denotes the jth example and N_{i-1} denotes the number of examples in $\mathcal{D}_{i-1}$. Correspondingly, for example, $\boldsymbol{x}_l^j$, the output feature of the lth layer can be computed by $\boldsymbol{x}_{l+1}^j = \sigma(W_{l,i-1}^T \boldsymbol{x}_l^j)$, where $W_{l,i-1}^T$ is the transpose of weight matrix $W_{l,i-1}$ of the lth layer after finishing task T_{i-1} and $\sigma(\cdot)$ denotes a nonlinear activation function. In GANs, the nonlinear activation function $\sigma(\cdot)$ usually adopts LeakyRelu [21] or other ReLu-based functions [8]. It should be noted that $\boldsymbol{x}_{l+1}^j$ is also the input feature of layer $l+1$. Specially, when $l = 1$, the input of the first layer is obtained by concatenating latent vector $\boldsymbol{z}^j$ with label condition vector $\boldsymbol{c}^j$, i.e., $\boldsymbol{x}_1^j = \boldsymbol{z}^j \oplus \boldsymbol{c}^j$, where $\oplus$ denotes the concatenation operation.

In conventional training, we have $A_{l+1} = \sigma(W_{l,i-1}^T A_l)$. Here, weight matrix $W_{l,i-1}$ serves as the initial weights when training task T_i, i.e., $W_{l,i}(0) \triangleq W_{l,i-1}$, where $W_{l,i-1}$ denotes the learning result of last task T_{i-1}. Thus, $A_{l+1} = \sigma(W_{l,i}(0)^T A_l)$. The weights at step t are updated by

$$\begin{aligned} W_{l,i}(t) &= W_{l,i}(t-1) - \lambda_i \triangle W_{l,i}(t) \\ &= W_{l,i}(0) - \lambda_i \sum_{k=1}^{t} \triangle W_{l,i}(k) \end{aligned} \tag{2}$$

where λ_i is the learning rate for task T_i and $\triangle W_{l,i}(t)$ denotes the weights gradient calculated at step t by some method such as stochastic gradient descent method. After t steps learning for task T_i, we have weights matrix $W_{l,i}(t)$ and new features A'_{l+1} can be computed by

$$\begin{aligned} A'_{l+1} &= \sigma\left(W_{l,i}^T(t) A_l\right) \\ &= \sigma\left(W_{l,i}^T(0) A_l - \lambda_i \sum_{k=1}^{t} \triangle W_{l,i}^T(k) A_l\right). \end{aligned} \tag{3}$$

Unpredictable inputs of the new task will make the update of weights uncontrollable. It is almost impossible that the equation $\sum_{k=1}^{t} \triangle W_{l,i}^T(k) A_l = \boldsymbol{O}$ happens to be satisfied, where $\boldsymbol{O}$ denotes the zero matrix. Thus, A'_{l+1} is hardly equal to A_{l+1}. $A'_{l+1} \neq A_{l+1}$ means that the network has forgotten the representation learned from old tasks and its response to examples input from old tasks is unpredictable and uncontrollable.

In the proposed GopGAN, to avoid generator G forgetting previous tasks' features $A_l, l = 1, 2, \ldots, L$, we construct a gradients orthogonal projection matrix P_l with size $d_l \times d_l$ for each layer, and let nonzero matrix P_l satisfy $P_l A_l = \boldsymbol{O}$. During learning a new task, the update of weights is carried out by a modified gradient $P_l^T \triangle W_{l,i}(t)$ instead of gradient $\triangle W_{l,i}(t)$, which can guarantee that $A'_{l+1} = A_{l+1}$ always holds mathematically, i.e., the representation learned from old tasks will not be forgotten during learning a new task. The mathematical proof is given as follows:

$$\begin{aligned} A'_{l+1} &= \sigma(W_{l,i}^T(t) A_l) \\ &= \sigma\left(\left(W_{l,i}(0) - \lambda_i \sum_{k=1}^{t} P_l^T \triangle W_{l,i}(k)\right)^T A_l\right) \\ &= \sigma\left(W_{l,i}^T(0) A_l - \lambda_i \sum_{k=1}^{t} \triangle W_{l,i}^T(k) P_l A_l\right) \\ &= \sigma(W_{l,i}(0)^T A_l) = A_{l+1}. \end{aligned} \tag{4}$$

Equation (4) shows that at any time t of learning the new task, no matter what weight updates are performed, the network response of each layer to examples of old tasks keeps unchanged if we can find a nonzero matrix P_l, which satisfies $P_l A_l = \boldsymbol{O}$.

To find such a matrix P_l, we formulate it into an optimization problem, and the objective function is defined as

$$P_l = \arg\min_P \left(||P A_l||_F^2 + \alpha ||P - I||_F^2\right) \tag{5}$$

where $||\cdot||_F$ denotes the Frobenius norm of matrix, I is the identity matrix, and α is the hyperparameter reflecting the tolerance to noises of A_l. When $\alpha = 0$, the row space of the solution P_l is orthogonal to the space spanned by column vectors of A_l, which means that for each column vector $\boldsymbol{x}_l^j$ in A_l, we have $P_l \boldsymbol{x}_l^j = \boldsymbol{0}$. In practical training, when the number of learned tasks increases, features matrix A_l will become full rank, and then, there is no nonzero solution P_l for (5). When α gets bigger, the solution P_l will pay more attention to the most important directions of $\{\boldsymbol{x}_l^j, j = 1, 2, \ldots\}$ and will ignore the noises in $\{\boldsymbol{x}_l^j\}$. A few outliers occupying nonmainstream directions will reduce the memory space available for new tasks. Theoretically, reducing the value of α will improve the quality of generated images, which is also consistent with the experimental results (shown in sectionV).

To solve the above optimization problem, we compute the derivative of the objective function with respect to P and get

$$2 A_l A_l^T P + 2\alpha (P - I) = \boldsymbol{O}. \tag{6}$$

Then, we get the solution of (5), i.e.,

$$P_l = \alpha (A_l A_l^T + \alpha I)^{-1}. \tag{7}$$

We apply the RLS algorithm [36] to recursively compute P_l. The detailed derivation is given in Appendix B. Finally, we get the iterative formula of P_l, i.e.,

$$P_l(k+1) = P_l(k) - \frac{P_l(k) \boldsymbol{x}_l^{k+1} \boldsymbol{x}_l^{k+1^T} P_l(k)}{\alpha + \boldsymbol{x}_l^{k+1^T} P_l(k) \boldsymbol{x}_l^{k+1}}. \tag{8}$$

We have summarized the proposed GopGANs in Algorithm 1. Before all the task learning, the orthogonal projection matrix P_l in every layer is initialized to an identity matrix I. It should be noted that the gradient orthogonal projectors are updated only after a new task has finished learning. Theoretically, the rank of projectors Rank(P) denotes the available memory space for unlearned tasks. When Rank(P) = 0, the weights will not be updated, and it means that a network loses the ability to learn new tasks.

C. Task-Dependent Latent Vector for GANs

In conventional CGANs, all the categories share the same latent vectors $\boldsymbol{z}^j$ except the label condition $\boldsymbol{c}^j$. The latent vectors are sampled from standard normal distribution, that is, $z_k^j \sim N(0, 1)$, where $\boldsymbol{z}^j = [z_1^j, z_2^j, \ldots, z_k^j, \ldots, z_{d_1}^j]$. If we ignore the label condition, the input of generator's first layer $\boldsymbol{x}_1^j$ equals to $\boldsymbol{z}^j$ with length d_1. The mathematical expectation of correlation matrix of $\boldsymbol{x}_1^j$ is $\mathbb{E}[\boldsymbol{x}_1^j \boldsymbol{x}_1^{j^T}] = \boldsymbol{I}$ and $\mathbb{E}[\boldsymbol{x}_1^{j^T} \boldsymbol{x}_1^j] = d_1$. According to the updating equation (8),

considering the first layer of generator ($l = 1$), we assume that (8) still approximately hold when α is ignored, since α has a very small value, and then, we have

$$\begin{aligned}\mathbb{E}[P_1(k+1)] &\approx \mathbb{E}[P_1(k)] - \frac{\mathbb{E}[P_1(k)]\mathbb{E}\left[\boldsymbol{x}_1^{k+1}\boldsymbol{x}_1^{k+1^T}\right]\mathbb{E}[P_1(k)]}{\mathbb{E}\left[\boldsymbol{x}_1^{k+1^T}P_1(k)\boldsymbol{x}_1^{k+1}\right]} \\ &\approx \mathbb{E}[P_1(k)] - \frac{\mathbb{E}[P_1(k)]}{d_1} \\ &= \left(1-\frac{1}{d_1}\right)^{k+1}\mathbb{E}[P_1(0)] = \left(1-\frac{1}{d_1}\right)^{k+1}\boldsymbol{I}.\end{aligned} \tag{9}$$

The detailed proof is available in Appendix A. In the training of GopGAN, the length of latent vector is usually set to 100 and the updating steps of P_1 are more than 1000, which will cause that $P_l = \boldsymbol{O}$. If matrix P_l becomes a zero matrix after learning a task, the weights in the first layer will not be updated anymore when GopGAN starts to train a new task, which not only seriously impedes network training but also reduces the quality of the generated images. Even with label condition $\boldsymbol{c}^j$ (i.e., $\boldsymbol{x}_1^j = \boldsymbol{z}^j \oplus \boldsymbol{c}^j$), since the label condition $\boldsymbol{c}^j$ is a one-hot vector, the valid memory space for unlearned tasks is just one dimension in their own label index, which still causes great interference to the training of the network.

To expand the available memory space for unlearned tasks, we proposed a task-dependent latent vector construction method. The latent vector is divided M slices and the number of slices is equal to the number of tasks. The latent subvector for the task T_i is denoted as $\boldsymbol{z}_i^j$ with length n_i. When training task T_i with label i, the slice $\boldsymbol{z}_i^j$ is activated and the other tasks' slices will be deactivated, which can provide unlearned tasks with sufficient memory space. In this article, we adopt the one-hot form to construct the latent vector similar to the construction of label condition $\boldsymbol{c}^j$, where the deactivated latent subvectors are replaced by zero vectors. When training task T_i, the conditional latent vector is

$$\boldsymbol{z}^j = \left[\boldsymbol{0}_1^T, \ldots, \boldsymbol{0}_{i-1}^T, \boldsymbol{z}_i^j, \boldsymbol{0}_{i+1}^T, \ldots, \boldsymbol{0}_M^T\right]^T. \tag{10}$$

Using the task-dependent latent vector, the available memory space in the first layer for unlearned tasks will increase from 0 to $d_1 - \sum_{i=1}^{Q} n_i$, where Q denotes the number of learned tasks. Experiments in Section V show that GopGANs with task-dependent latent vectors can greatly improve the quality of generated images.

IV. Discussion

In this section, we discuss the relationship between the proposed method GopGAN and other methods, including SFT, EWC [18], OWM [39], and LifelongGAN [40].

A. Relationship of GopGAN and SFT

For SFT, when its generator starts to learn a new task T_i, its weights are initialized using the weights learned after the last task T_{i-1} and the weight $W_{l,i}^{\mathrm{SFT}}(t)$ of layer l is updated at step t by

$$W_{l,i}^{\mathrm{SFT}}(t) = W_{l,i}^{\mathrm{SFT}}(t-1) - \lambda_i \Delta W_{l,i}(t) \tag{11}$$

where $\Delta W_{l,i}(t)$ denotes the corresponding gradient. Different from SFT, the GopGAN updates the weight $W_{l,i}^{\mathrm{Gop}}(t)$ at step t by

$$W_{l,i}^{\mathrm{Gop}}(t) = W_{l,i}^{\mathrm{Gop}}(t-1) - \lambda_i P_l^T \Delta W_{l,i}(t) \tag{12}$$

where gradients $\Delta W_{l,i}(t)$ are modified by matrix P_l, which is obtained by seeking the optimal solution, i.e.,

$$P_l = \arg\min_P \left(||PA_l||_F^2 + \alpha||P - I||_F^2\right) \tag{13}$$

where matrix $A_l = [\boldsymbol{x}_l^1, \boldsymbol{x}_l^2, \ldots, \boldsymbol{x}_l^{N_{i-1}}]$ is constructed by collecting the inputs of all the learned tasks at the lth layer and $\boldsymbol{x}_l^k$ denotes the inputs of layer l for the kth example from tasks $T_1, T_2, \ldots, T_{i-1}$.

In practice, before the converge of the generator when training a new task, the gradient $\Delta W_{l,i}$ is usually not a zero matrix, i.e., $\Delta W_{l,i}(t) \neq \boldsymbol{O}$, so that only when $P_l = \boldsymbol{I}$, (11) equals to (12). For positive real number α and $A_l \neq \boldsymbol{O}$, when α approaches positive infinity, the limit value of P_l is obtained by

$$\begin{aligned}\lim_{\alpha\to+\infty} P_l &= \lim_{\alpha\to+\infty} \arg\min_P \left(||PA_l||_F^2 + \alpha||P - I||_F^2\right) \\ &= \lim_{\alpha\to+\infty} \arg\min_P \left(\alpha||P - I||_F^2\right) = \boldsymbol{I}.\end{aligned} \tag{14}$$

The above derivation shows that the SFT is a special case of GopGAN when α approaches positive infinity. For the method GopGAN, as α gets bigger, the generator will forget more and more old knowledge learned from the previous tasks, and finally, it will degrade into SFT. Our experiments also show that a smaller value of α can help the generator to retain more old knowledge.

B. Comparison of GopGAN and EWC

In GAN with EWC [18], [32], the loss of the generator can be represented as

$$L_{\mathrm{EWC}} = L_{\mathrm{adv}} + \frac{\lambda_{\mathrm{EWC}}}{2} L_{\mathrm{ewc}} \tag{15}$$

where L_{adv} is the common adversarial loss in GAN and L_{ewc} is defined as

$$L_{\mathrm{ewc}} = \sum_{l=1}^{L}\sum_{j=1} F_l^j \left(W_{l,i}^j - W_{l,i-1}^j\right)^2 \tag{16}$$

where F_l^j is the jth diagonal element of the Fisher information matrix F_l for layer lth and $W_{l,i}^j$ is the jth element in matrix $W_{l,i}$. The total loss of EWC consists of two parts: the new task generation loss L_{adv} and EWC loss L_{ewc} between the new task and old tasks. L_{adv} enables the model to learn new tasks, whereas L_{ewc} hopes that the new model will retain the knowledge learned from the old tasks as much as possible. The total loss is a tradeoff between the new task and old tasks.

The gradients with respect to weights in the lth layer $\Delta W_{l,i} = \partial L_{\mathrm{EWC}}/\partial W_{l,i}$ of the total loss in (15) also consist of two parts

$$\begin{aligned}\Delta W_{l,i} &= \Delta W_{l,i}^{\mathrm{adv}} + \frac{\lambda_{\mathrm{EWC}}}{2} \cdot \Delta W_{l,i}^{\mathrm{ewc}} \\ &= \frac{\partial L_{\mathrm{adv}}}{\partial W_{l,i}} + \frac{\lambda_{\mathrm{EWC}}}{2} \cdot \frac{\partial L_{\mathrm{ewc}}}{\partial W_{l,i}}.\end{aligned} \tag{17}$$

For inputs $A_{l,k}(k = 1, \ldots, i-1)$ of old tasks, the update of weights $W_{l,i}$ based on $\Delta W_{l,i}^{\text{ewc}}$, $W_{l,i}^{\text{ewc}} = W_{l,i} - \lambda_i \Delta W_{l,i}^{\text{ewc}}$, tries not to forget old tasks, i.e., ${W_{l,i}^{\text{ewc}}}^T A_{l,k} \approx {W_{l,i-1}}^T A_{l,k}$ for $k = 1, \ldots, i-1$. However, the gradient $\Delta W_{l,i}^{\text{adv}}$ do not have this constraint. It cannot be guaranteed that after updating $W_{l,i}$ based on $\Delta W_{l,i}^{\text{adv}}$, i.e., $W_{l,i}^{\text{adv}} = W_{l,i} - \lambda_i \Delta W_{l,i}^{\text{adv}}$, ${W_{l,i}^{\text{adv}}}^T A_{l,k} \approx {W_{l,i-1}}^T A_{l,k}$ for $k = 1, \ldots, i-1$. Because the loss L_{adv} is completely irrelevant to the old tasks, with maximum probability, ${W_{l,i}^{\text{adv}}}^T A_{l,k} \neq {W_{l,i-1}}^T A_{l,k}$ for $k = 1, \ldots, i-1$. In other words, the new task destroys the model's knowledge of the old tasks, which is why the total loss is a tradeoff between the new task and old tasks.

EWC has another problem. For a CNN, it has many layers. In each layer, the weights of the new task model has a small fluctuation in response to $A_{l,k}$ for $k = 1, \ldots, i-1$. With the forward propagation of the network, these fluctuations are likely to be greatly amplified after various nonlinear operations, such as ReLu and MaxPooling, making the new model's response to old tasks almost far away from the original responses in previous models. This is also why EWC sometimes does not perform well in continual learning, especially in models with many network layers or tasks with a large number of tasks.

It should be noted that the GopGAN has the ability to keep the responses to all the previous tasks from T_1 to T_{i-1} unchanged by the gradients orthogonal projection. For GopGAN, we have $W_{l,i}^T A_{l,k} = W_{l,i-1}^T A_{l,k}$ for $k = 1, \ldots, i-1$, and the detailed derivation has been given in this article. It is notable that, in GopGAN, the gradients of the total loss are modified by the orthogonal projection matrix P instead of partial gradient, which leaves the response of the updated model to the old task unchanged. However, EWC only reflects the goal of keeping the model's response to the old task unchanged in the partial loss EWC, but the adversarial loss of the new task will still break the relationship.

C. Comparison of GopGAN and OWM

The method OWM [39] and GopGAN were separately proposed to serve the same purpose, that is, to address catastrophic forgetting by training the network along the direction orthogonal to space spanned by the inputs of learned tasks. Comparatively speaking, the method OWM is used for classification tasks and GopGAN is for GAN in continual learning. Compared with the OWM, our GopGAN has three main improvements.

First, in the paper of OWM [39], the matrix P_{owm} is given directly without any derivation to explain why the matrix should be expressed as follows:

$$P_{\text{owm}} = \boldsymbol{I} - A(A^T A + \alpha \boldsymbol{I})^{-1} A^T. \tag{18}$$

In Section III-B, we give that a rigorous mathematical proof to show updating weights in the orthogonal subspace of feature space can guarantee that the learning of a new task does not result in forgetting the knowledge acquired in old tasks. Then, by defining and solving an optimization problem, we derive a new optimal solution of projection matrix denoted as P_{gop}, i.e.,

$$P_{\text{gop}} = \alpha(AA^T + \alpha \boldsymbol{I})^{-1}. \tag{19}$$

According to the Woodbury matrix identity [12], it can be derived that

$$\begin{aligned} P_{\text{gop}} &= \alpha(AA^T + \alpha \boldsymbol{I})^{-1} \\ &= \boldsymbol{I} - A(A^T A + \alpha \boldsymbol{I})^{-1} A^T = P_{\text{owm}} \end{aligned} \tag{20}$$

which shows that our gradients orthogonal projection matrix is equivalent to the one in the OWM.

Second, the expression in (19) is convenient for applying the RLS algorithm [36]. We also provide the detailed derivation of how to recursively compute P_{gop} in (19) so that there is no need to store features of the training examples used to learn the previous tasks when learning a new task.

Third, in the continual learning of GANs, we find that the noise input of generator comes from the same distribution and is shared by all the categories, which makes the calculated projection matrix approach to the zero matrix, whose proof can be seen in Section III-B and the following supplementary proof in Appendix A, and then, little memory space is available for new tasks. To address the problem, we propose a new task-dependent latent vector to make each task have a different latent vector so as to obtain enough memory space for new tasks. Extensive experiments show that with task-dependent latent vectors, GANs can generate better quality of images compared with the GAN only based on the OWM.

D. Comparison of GopGAN and Lifelong GAN

We define that $\boldsymbol{A}$ denotes the inputs of learned tasks and $\boldsymbol{B}$ denotes the inputs of the ith task that will be learned by the network. During training the ith task, the generator is G_i and the generator for the first $i-1$ tasks is G_{i-1}.

To avoid catastrophic forgetting, our method GopGAN keeps the response of the new generator G_i to the learned tasks' data $\boldsymbol{A}$ unchanged while learning the new task, i.e.,

$$\text{GopGAN:} \begin{cases} G_i(\boldsymbol{A}) = G_{i-1}(\boldsymbol{A}) = GT(\boldsymbol{A}) \\ G_i(\boldsymbol{B}) = GT(\boldsymbol{B}) \end{cases} \tag{21}$$

where $\text{GT}(\boldsymbol{A})$ denotes the ground truth of input $\boldsymbol{A}$ and $G_i(\boldsymbol{B}) = \text{GT}(\boldsymbol{B})$ means that the generator G_i can generate real images consistent with the ground truth. $G_i(\boldsymbol{A}) = G_{i-1}(\boldsymbol{A})$ means that GopGAN can keep the learned knowledge unchanged. More strictly, in GopGAN, the response of each layer in the new network to the features of the learned task also remains the same, not just the final output.

LifelongGAN [40] used distillation loss [13] to avoid forgetting learned knowledge, which can be expressed in the following formula:

$$\text{LifelongGAN:} \begin{cases} G_i(\boldsymbol{B}) = G_{i-1}(\boldsymbol{B}) \\ G_i(\boldsymbol{B}^{\text{aux}}) = G_{i-1}(\boldsymbol{B}^{\text{aux}}) \\ G_i(\boldsymbol{B}) = \text{GT}(\boldsymbol{B}) \end{cases} \tag{22}$$

where $\boldsymbol{B}^{\text{aux}}$ is the auxiliary data in LifelongGAN generated from training data $\boldsymbol{B}$ by montage or swap operation. LifelongGAN tries to retain the knowledge of the learned task by distilling the response of the generator G_{i-1} to the new task

data $\boldsymbol{B}$ and $\boldsymbol{B}^{\text{aux}}$. Because the previous generator G_{i-1} has not been exposed to the data of the new task, distilling the knowledge of the previous generator to the new task cannot make the new generator G_i completely retain the knowledge that has been learned. Due to the different distribution of learned tasks' data $\boldsymbol{A}$ and the data $\boldsymbol{B}$ of the task to be leaned, $G_i(\boldsymbol{B}) = G_{i-1}(\boldsymbol{B})$ and $G_i(\boldsymbol{B}^{\text{aux}}) = G_{i-1}(\boldsymbol{B}^{\text{aux}})$ are not equivalent to $G_i(\boldsymbol{A}) = G_{i-1}(\boldsymbol{A})$. Moreover, when training a new task, LifelongGAN's knowledge of the learned task will be interfered by the new task, which means that the learning of the new task will also be accompanied by forgetting. Our method overcomes this problem very well. The detailed theoretical derivation is shown in Section III and experiments are shown in Section V.

V. Experiments

A. Experiments Setting

1) Datasets: In this article, we carry out the experiments on several benchmark datasets MNIST [20], SVHN [26], EMNIST [3], CIFAR10 [19], and ImageNet-200 [34]. MNIST is a handwriting digit dataset with ten categories from 0 to 9. There are 60 000 examples in the training set with image size $28 \times 28 \times 1$. In our experiments, all examples are resized to $32 \times 32 \times 1$. The testing set is not used and the training set is not augmented. Similar to MNIST, SVHN incorporates an order of magnitude more labeled data and comes from a significantly harder, unsolved, real-world problem. The generation task on SVHN is more challenging than MNIST and EMNIST. The train set with 73 257 digits is used in our experiments and all digits are of fixed resolution of $32 \times 32 \times 3$ pixels. EMNIST contains more categories than MNIST and SVHN. We adopt balanced EMNIST dataset containing 47 categories. CIFAR10 contains ten categories with 32×32 color images. We use the training set that contains total 50 000 images to train the generator. ImageNet-200 (tiny ImageNet-200) is an image classification dataset provided by Stanford University, which contains 200 categories. Each category contains 500 training images, 50 verification images, and 50 test images.

2) Architectures: When we compare the performances of different approaches on a dataset, the same CGAN architecture is adopted for the sake of fair comparisons. However, for different datasets, the different CGAN architectures are adopted since they have different degrees of data complexity to be modeled. We also implement SFT, EWC [18], OWM [39], MeRGANs [37], and LifelongGAN [40] with the same architecture for fair comparison and all the experiments are carried out using PyTorch with Adam optimizer [17]. In the SFT method, when we start to train a new task, the model parameters are initialized from the last learned task model and the loss of the model is only the common adversarial loss on the new task.

3) Hyperparameters: The learning rates of generator λ_g and discriminator λ_d are the same 2×10^{-4}. The batch size is set to 64. γ in Online EWC is 0.99. λ_{RA} in MeRGAN is 0.001 for MNIST and EMNIST and 0.01 for SVHN, CIFAR10, and ImageNet-200. α in OWM and GopGAN is 1×10^{-4}.

4) Quantitative Metrics: FID [11], average accuracy (A) [2], forgetting measure (F) [2], intransigence measure(I) [2], and training data occupation (D) are utilized to evaluate the different aspects of the generation in our experiments. FID is proposed to compute the Frechet distance between the real images distribution and generated images distribution and a smaller FID means that the images with better quality are generated. In our experiments, we use a pretrained Inception V3 network provided by PyTorch to calculate FID. A_k is the average accuracy of all k learned tasks on the final model. Forgetting measure reflects the catastrophically forgetting knowledge of previous tasks and intransigence measure reflects the inability to update the knowledge to learn the new task.

The metrics A, F, and I are proposed in continual classification tasks and our work focuses on continual generation tasks. Thus, to evaluate the quality of generated images, we use improved metrics $A, r\text{-}A, F, r\text{-}F, I$. We define that a_j^k is the accuracy of the classifier network trained on real images and evaluated on generated images of the jth task after training the generator incrementally from tasks 1 to k. $r\text{-}a_j^k$ is the accuracy of the classifier network trained on generated images of k tasks and evaluated on real images of the jth task. Then, average accuracies are $A_k = \sum_{j=1}^{k} a_j^k/k$ and $r\text{-}A_k = \sum_{j=1}^{k} r\text{-}a_j^k/k$. Forgetting measures are $F_k = (1/(k-1)) \sum_{j=1}^{k-1} f_j^k$ and $r\text{-}F_k = (1/(k-1)) \sum_{j=1}^{k-1} r\text{-}f_j^k$, where

$$f_j^k = \max_{o \in \{j,\ldots,k-1\}} a_j^o - a_j^k \quad \forall j < k$$
$$r\text{-}f_j^k = \max_{o \in \{j,\ldots,k-1\}} r\text{-}a_j^o - r\text{-}a_j^k \quad \forall j < k. \tag{23}$$

For intransigence measure, we first train a classifier network and a generator with all k tasks dataset $\bigcup_{j=1}^{k} \mathcal{D}_j$. Then, we generate dataset $\mathcal{D}_k'$ of the kth task and measure its accuracy on classifier network, denoted as a_k^*. Intransigence measure is $I_k = a_k^* - a_k^k$. Lower F_k, $r\text{-}F_k$, or I_k, the better the model. We use D_k to denote the amount of training data used for learning task k.

B. Experiments on MNIST

The architecture of CGANs on MNIST is shown in Table II. For each task on MNIST, the iterations are set to 80 000. The length of latent vector is set to 100 in all methods. In the following, MeRGANs denotes both methods MeRGAN-JTR and MeRGAN-RA.

1) Ordered MNIST: Fig. 2 and Table I show the comparison between GopGAN and other methods on ordered MNIST, where the generation tasks are ordered from category 0 to 9. The results of SFT and EWC show that with the category number increases, sequential tasks fine-tuning learning suffers from catastrophic forgetting in GANs. In Fig. 2(b), GANs with SFT and EWC can only remember the last task and forget previous tasks absolutely. GopGAN can generate the images with better quality for all the categories. Although OWM has some improvements compared to SFT, it still has the problem of serious forgetting. Compared with OWM, GopGAN with task-dependent latent vectors can effectively improve the quality of the generated images.

TABLE I

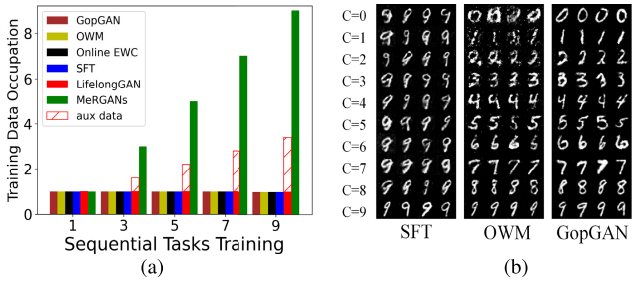
RESULTS ON ORDERED AND SHUFFLED MNIST. AUX DATA ARE THE AUXILIARY DATA IN LIFELONGGAN, AND IN EXPERIMENTS, WE SET ∼10 000 IMAGE PAIRS TO DISTILL LEARNED KNOWLEDGE. THE SHUFFLED MNIST IS RANDOMLY SHUFFLED BY TASKS ORDER (TEN TIMES), AND THEN, THE AVERAGE ACCURACY IS CALCULATED

Methods	$A_{10}(\%)$	F_{10}	I_{10}	r-$A_{10}(\%)$	r-F_{10}	D_{10}
Ordered MNIST						
SFT	13.83	0.95	5.70×10^{-3}	12.53	0.96	4.49M
Online EWC	59.61	0.47	0.33	54.38	0.58	4.49M
OWM	87.39	0.17	0.15	82.93	0.25	4.49M
MeRGANs	99.63	**0.09**	5.86×10^{-2}	99.52	0.12	**45.3**M
LifelongGAN	99.51	0.13	9.65×10^{-2}	99.50	0.17	4.49M+aux data
GopGAN(ours)	**99.65**	0.11	3.27×10^{-2}	**99.60**	0.12	4.49M
Shuffled MNIST						
SFT	13.07±0.63	0.98±0.09	$(5.70 \pm 0.01) \times 10^{-3}$	12.13±0.15	0.98±0.004	4.49M
Online EWC	57.82±0.50	0.51±0.015	0.35±0.004	53.19±2.47	0.64±0.004	4.49M
OWM	84.57±7.74	0.25±0.063	0.18±0.004	79.87±**9.48**	0.28±0.009	4.49M
MeRGANs	99.28±0.125	**0.11**±0.0003	$(5.77\pm0.01)\times10^{-2}$	99.50±0.13	0.16±0.009	**45.3**M
LifelongGAN	99.07±0.23	0.14±0.0001	$(9.74\pm0.01)\times10^{-2}$	89.80±0.28	0.18±0.0001	4.49M+aux data
GopGAN(ours)	**99.45**±0.04	**0.11**±0.0001	$(3.34\pm0.01)\times10^{-2}$	**99.46±0.06**	0.12±0.001	**4.49**M

TABLE II

ARCHITECTURE OF CGAN ON MNIST. MLP 256 DENOTES FULLY CONNECTED LAYER WITH 256 HIDDEN NOTES. LR 0.2 DENOTES THE ACTIVATION FUNCTION LEAKYRELU [21] WITH PARAMETER 0.2

Layers	1	2	3	4	5
G	Input z,c	MLP 256 LR 0.2	MLP 512 LR 0.2	MLP 1024 LR 0.2	MLP 1024 LR 0.2
D	Input I,c	MLP 512 LR 0.2	MLP 512 LR 0.2	MLP 512 LR 0.2	MLP 1 LR 0.2



(a) (b)

Fig. 2. Comparison between GopGAN and other methods on ordered MNIST. (a) Scale drawing of training data occupation in sequential tasks training. Because LifelongGAN uses auxiliary data that are indeterminate, we use unfilled boxes to represent the auxiliary data. We show the training data occupation in Table I. (b) Visualization of generated images.

Fig. 2(a) shows a rough statistics to feature space occupation in each task T_i. Table I also provides the quantitative analysis that how much amount of data will be used for learning the last task (digit 9 generation). In Fig. 2(a), a unit on the vertical axis refers to the space occupation of method SFT in one task learning. We can see that other methods except for MeRGANs and LifelongGAN occupy the same feature space. However, because of the replay mechanism, when training task T_i, MeRGANs will replay additional $i-1$ tasks data, which means that MeRGANs occupy more and more feature space with the number of tasks increasing. LifelongGAN needs to generate lots of auxiliary data to distill knowledge from the old generator to new generator. In experiments, we set ∼10 000 image pairs to distill learned knowledge. Lots of training time consuming and memory space occupation are the weaknesses of MeRGANs and LifelongGAN, though it can achieve better results in accuracy than the other methods such as SFT, EWC, and OWM. Our method GopGAN does not need additional training data and can also achieve the state of the art.

2) Shuffled MNIST: Furthermore, we carry out the experiments on MNIST with shuffled categories. The sequential tasks $T_{\text{mnist}} = \{T_1, T_2, \ldots, T_{10}\}$ are shuffled randomly. On each randomly shuffled MNIST dataset, all the methods are carried out once. We randomly shuffle the MNIST ten times and then calculate the mean and variance of each method. Table I shows the results of experiments on shuffled MNIST datasets. SFT and EWC almost forget previous tasks so that the accuracies are very low. Compared with SFT, OWM has significant improvement, increasing average accuracy from 13.07% to 84.57%. OWM method has a higher variance than other methods since it is sensitive to the order of the sequential tasks. Our method GopGAN adopts the task-dependent latent vectors and decreases the sensitivity to tasks order. MeRGANs and LifelongGAN are also affected by the task order, though with comparable averaged accuracy, but have larger variance than our method.

3) Effects of α and Task-Dependent Latent Vector: We design the experiments to evaluate the effects of α and task-dependent latent vectors length on ordered MNIST dataset. The experimental results are shown in Tables III and IV. In experiments, the length of latent subvector of each category is equal.

When α goes down from 0.1 to around 0.001, the average accuracy is slightly improved. Then, reduce α, and the average

TABLE III

EFFECTS OF α TO ACCURACY AND FORGETTING SCORE. THE LENGTH OF TASK-DEPENDENT LATENT VECTORS IS FIXED TO 500

α	0.1	0.01	0.001	0.0001	0.00001
$A_{10}(\%)$	99.51	99.57	**99.65**	99.64	99.65

TABLE IV

EFFECTS OF LATENT VECTOR LENGTH d_1 TO ACCURACY AND FORGETTING SCORE. HYPERPARAMETER α IS FIXED TO 0.001

d_1	100	300	500	1000
$A_{10}(\%)$	99.53	99.58	99.65	99.65

TABLE V

EFFECT OF TASK-DEPENDENT LATENT VECTOR. "W/O" DENOTES GOPGAN WITHOUT TASK-DEPENDENT LATENT VECTOR. "W" DENOTES GOPGAN WITH TASK-DEPENDENT LATENT VECTOR

	Average accuracy $A_{10}(\%)$		
	MNIST	SVHN	CIFAR10
w/o	87.39	53.20	64.82
w	99.65	83.02	91.05

accuracy does not improve significantly, or even slightly reduced, which is expected, because with the decrease of α, the updating of neural network model becomes more demanding. In Table IV, it can be concluded that increasing the length of task-dependent latent vectors can slightly improve the accuracy. Compared with the OWM method, we can conclude that the main improvement on accuracy of GopGAN comes from the method of task-dependent latent vectors construction we proposed in GopGAN rather than its length.

To evaluate the effect of task-dependent latent vector, we carry out experiments on the MNIST, SVHN, and CIFAR10 datasets. Table V shows the result. We can see that GopGAN with task-dependent latent vector has a great improvement. With the task-dependent latent vector, on MNIST, the accuracy is improved by about 12% and about 30% and 26% on SVHN and CIFAR10, respectively, which shows the effectiveness of task-dependent latent vector.

C. Multitask Experiments

We evaluate the GopGAN on multicategory experiments on the EMNIST dataset. The balanced EMNIST dataset consists of 47 categories. In experiments, we set 47 generation tasks, each category for one task. Fig. 3 shows the generated samples after sequentially learning all tasks. The SFT method only remembers the last task, the letter '*t*' generation, and forgets all the previous 46 generation tasks. Without task-dependent latent vector, the OWM method cannot retain all the learned tasks and the generated images are confused. In the multitask generation, the images generated by our method are more accurate and diverse than MeRGANs and LifelongGAN. Table VI shows the changes in the accuracy rate and the amount of data required for the last task of training during the

Fig. 3. Generated samples on balanced Eminist. Raw images are sampled from the original training dataset.

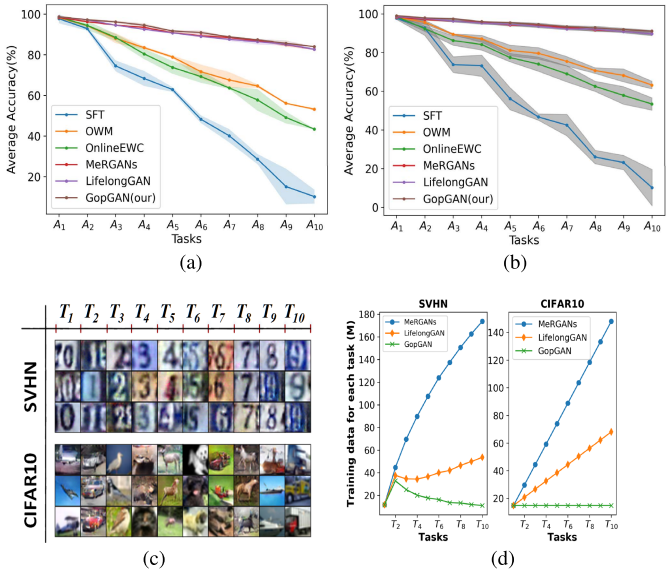


Fig. 4. Comparison between GopGAN and other methods on the SVHN and CIFAR10 datasets. (a) Average accuracy on the SVHN dataset in sequential tasks training. (b) Average accuracy on the CIFAR10 dataset in sequential tasks training. (c) Visualization of generated images on the SVHN and CIFAR10 datasets. (d) Memory occupation of training data on each task. The memory occupation $D_k(GopGAN) = D_k(SFT) = D_k(OWM) = D_k(OnlineEWC)$, so SFT, OWM, and Online EWC do not appear in (d).

learning process. For SFT, Online EWC, and OWM methods, as the number of tasks increases, the accuracy rate decreases rapidly. Our method can maintain considerable accuracy while saving lots of training data.

D. Color Images Generation

1) Experiments on SVHN and CIFAR10: We evaluate our method GopGAN on more challenging datasets consisting of real-world color images. Both SVHN and CIFAR10 datasets consist of ten categories. In experiments, we set ten generation tasks both on SVHN and CIFAR10, each category for one task. The backbone of the generator is Resnet18 [9] and Resnet18 for discriminator. For the LifelongGAN method, when learning new task, we generate $2000 * N\_$ auxiliary pair images to distill learned knowledge, where $N\_$ denotes the number of learned tasks. For each method, we evaluate ten times to compute the mean and variance.

The average accuracy, generated samples, training data size, and FID are shown in Fig. 4 and Table VII. Catastrophic forgetting occurs in SFT, which has the lowest average accuracy

TABLE VI
RESULTS ON EMNIST. FOR LIFELONGGAN, WE SET ∼10 000 IMAGE PAIRS TO DISTILL LEARNED KNOWLEDGE

Methods	$T_1 \rightarrow T_{16}$		$T_1 \rightarrow T_{32}$		$T_1 \rightarrow T_{47}$	
	$A_{16}(\%)$	D_{16}	$A_{32}(\%)$	D_{32}	$A_{47}(\%)$	D_{47}
SFT	9.38±4.13	1.81M	7.90±3.97	1.81M	5.47±4.05	1.81M
Online EWC	63.02±1.06	1.81M	37.25±1.57	1.81M	29.78±1.46	1.81M
OWM	75.92±0.84	1.81M	46.88±1.04	1.81M	38.30±2.88	1.81M
MeRGANs	98.22±1.19	**29.01M**	**96.59**±1.12	**58.01M**	92.89±0.16	**85.2M**
LifelongGAN	96.90±1.84	7.55M	93.75±0.98	7.55M	92.17±0.55	7.55M
GopGAN(ours)	**98.80**±0.36	1.81M	95.83±0.41	1.81M	**93.21±0.09**	1.81M

TABLE VII
RESULTS ON SVHN AND CIFAR10. FOR LIFELONGGAN, WE SET ∼2000 IMAGE PAIRS FOR EACH LEARNED TASK TO DISTILL LEARNED KNOWLEDGE. ↑ DENOTES THE HIGHER THE BETTER. ↓ DENOTES THE LOWER THE BETTER

Methods	SVHN			CIFAR10		
	$A_{10}(\%)\uparrow$	FID↓	D_{10}	$A_{30}(\%)\uparrow$	FID↓	D_{10}
SFT	10.20±4.13	100.14±10.13	11.04M	10.59±6.26	110.94±12.12	14.8M
Online EWC	43.60±1.06	97.83±6.31	11.04M	54.17±4.16	93.07±8.46	14.8M
OWM	53.20±0.84	127.04±14.13	11.04M	64.82±1.28	136.06±16.31	14.8M
MeRGANs	82.68±0.19	73.87±4.13	**173.60M**	90.71±0.13	72.89±6.16	**148M**
LifelongGAN	82.17±0.84	75.38±6.48	53.70M	89.51±0.33	82.17±4.55	68.08M
GopGAN(ours)	**83.02**±0.36	51.84±3.87	11.04M	**91.05**±0.16	47.03±6.09	14.8M

TABLE VIII
RESULTS ON IMAGENET-200. ↑ DENOTES THE HIGHER THE BETTER. ↓ DENOTES THE LOWER THE BETTER. FOR LIFELONGGAN, WE SET ∼2000 IMAGE PAIRS FOR EACH LEARNED TASK TO DISTILL LEARNED KNOWLEDGE

Methods	$A_{10}(\%)\uparrow$	$F_{10}\downarrow$	$I_{10}\downarrow$	FID↓	D_{10}
SFT	8.90±4.03	0.75±0.14	$(4.03 \pm 0.01) \times 10^{-3}$	110.47±13.20	18.8M
Online EWC	38.38±6.04	0.51±0.16	0.09 ± 0.001	100.92±9.06	18.8M
OWM	40.08±4.38	0.43±0.36	0.11 ± 0.04	92.93±6.81	18.8M
MeRGANs	57.01±2.16	0.29±0.12	$(5.86\pm0.01)\times10^{-2}$	45.13±8.74	**188.00**M
LifelongGAN	56.48±1.08	0.13±0.09	$(8.65\pm0.03)\times10^{-2}$	48.78±6.19	**120.32**M
GopGAN(ours)	**60.65**±3.18	0.11±0.09	$(3.27\pm0.01)\times10^{-2}$	**40.05**	18.8M

and the highest FID. With the same training data occupation, our method GopGAN is far superior to SFT, OWM, and Online EWC. With the increase in the number of learning tasks, OWM and Online EWC suffer from serious catastrophic forgetting. From Table VII, we can see that both OWM and Online EWC not only have low average accuracy and large FID but also have large variance on average accuracy and FID. Although the average accuracies of MeRGANs and LifelongGAN are comparable to that of our method, MeRGANs and LifelongGAN not only need more training data for learning new tasks but also generate samples of lower quality (performance in higher FID) than our method. Additional training data will increase the training time for MeRGANs and LifelongGAN. Especially, MeRGANs need to generate samples for all the learned tasks, which increases sharply with the increase of the number of learning tasks and the training time also increases.

2) Experiments on ImageNet-200: We apply our method on more complex dataset ImageNet-200, which is selected from Imagenet-1000 [30]. ImageNet-200 consists of 200 categories. In experiments, we divide ImageNet-200 into ten sequential generation task and each task consists of 20 categories. The backbone of the generator is Resnet18 [9] and Resnet18 for discriminator. Spectral normalization [25] is also adopted in experiments. To compute average accuracy, a resnet-32 model is pretrained on ImageNet-200, which achieves 61.02% accuracy on test dataset. For the LifelongGAN method, when learning new task, we generate $6000 * N\_$ auxiliary pair images to distill learned knowledge, where $N\_$ denotes the number of learned tasks.

The generated samples are shown in Fig. 5. The average accuracy, forgetting measure, intransigence measure, FID, and training data occupation are shown in Table VIII. Even with large categories (up to 200) generation, our method can still continually generate images with high quality. SFT, OWM, and Online EWC still suffer from catastrophic forgetting, e.g., low average accuracy, large forgetting measure, and FID. With so many categories, although these 200 categories are divided into ten generation tasks, MeRGANs and LifelongGAN still perform worse than ours (e.g., 3.6% lower and 4.2% lower on average accuracy). The reason for this phenomenon is

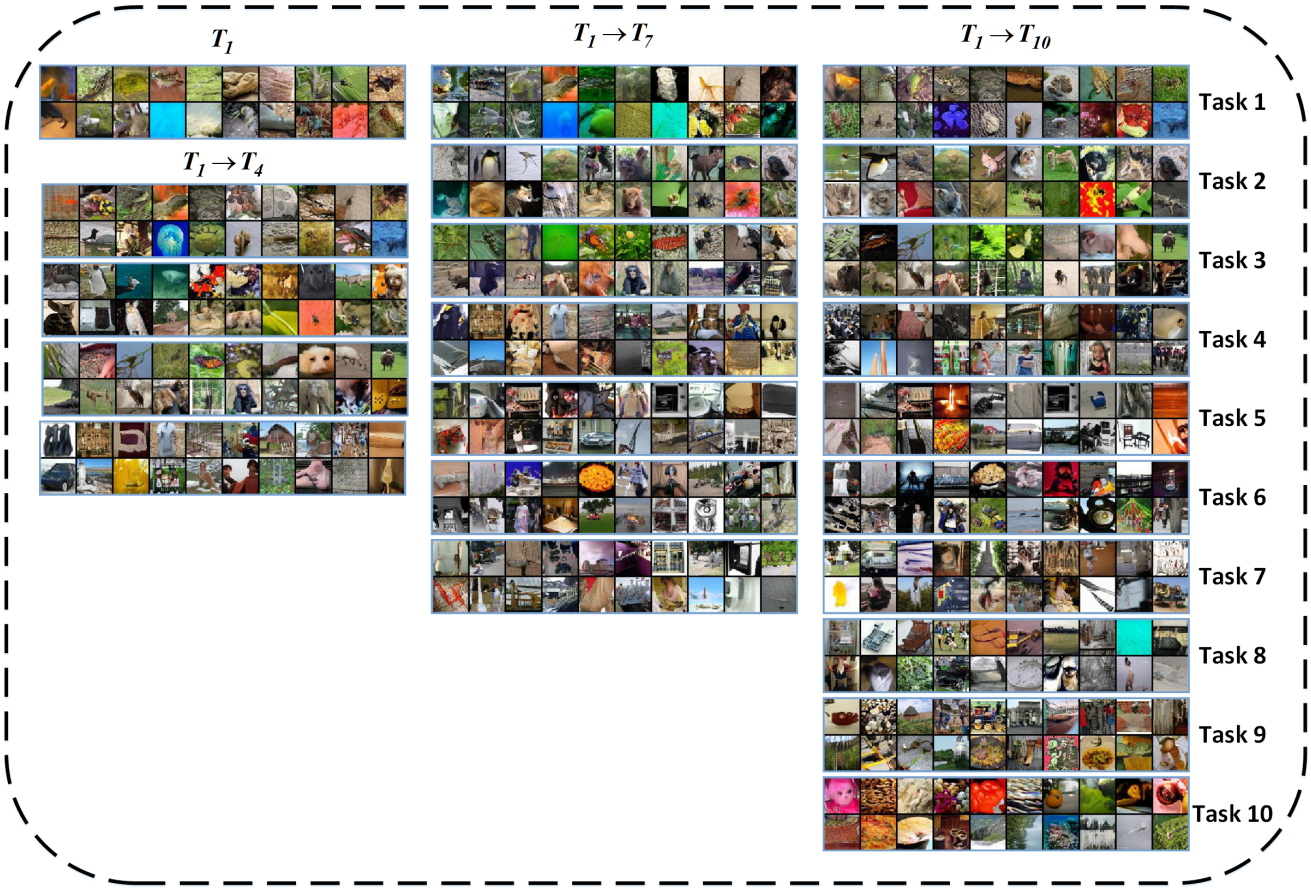


Fig. 5. ImageNet-200 generated samples with ten sequential tasks by our method GopGAN.

obvious. MeRGANs need to generate massive data of learned categories. With the increase of tasks, the images quality generated by the generator cannot be guaranteed (worse and worse). These biases will continue to accumulate, resulting in poor performance. Knowledge distillation has the same problem. LifelongGAN uses fake images from irrelevant categories to distill knowledge from learned categories, which makes LifelongGAN cannot achieve good performance.

VI. Conclusion

In this article, we propose a novel GopGAN to effectively address the issue of catastrophic forgetting in continual learning. The proposed GopGAN has not only a solid theory foundation but also obtain the support of experimental results. GopGAN can not only generate high-quality images but also overcome catastrophic forgetting in GAN with continual learning. The proposed task-dependent latent vectors can effectively increase the performance of GopGANs and make it achieve the state-of-the-art performance. Extensive experiments on MNIST, EMNIST, SVHN, CIFAR10, and ImageNet-200 show the superiority of GopGAN in learning sequential tasks.

Appendix A
Proof of (9) in Section III-C.

Here, we provide the proof of equation 9 in Section III-C. Before we start to prove it, we simplify the problem as follows. In the first layer of the generator, the input is random noises sampled from the normal distribution $\mathcal{N}(0,1)$, which is denoted as $\boldsymbol{x}$ with length d. The recursively solving of the gradients orthogonal projection matrix is formulated as

$$P(k+1) = P(k) - \frac{P(k)\boldsymbol{x}^{k+1}\boldsymbol{x}^{k+1^T}P(k)}{\alpha + \boldsymbol{x}^{k+1^T}P(k)\boldsymbol{x}^{k+1}} \tag{24}$$

where $\boldsymbol{x}^{k+1}$ denotes the $(k+1)$th input and the initial $P(0)$ is the identity matrix, that is, $P(0) = \boldsymbol{I}$.

Then, we will prove that if $\boldsymbol{x}^k \sim \mathcal{N}(0,1)$ and α is close to zero, $P(k) = \boldsymbol{O}$, a zero matrix, when k approaches positive infinity.

Proof: For each element in $\boldsymbol{x}^k = [x_1^k, x_2^k, \ldots, x_d^k]^T$, $x_j^k \sim \mathcal{N}(0,1)$, $j = 1, 2, \ldots, d$. We first prove that $\mathbb{E}[\boldsymbol{x}^k \boldsymbol{x}^{k^T}] = \boldsymbol{I}$, where $\mathbb{E}[\cdot]$ denotes the mathematical expectation. Definition $\mathbb{D}[\cdot]$ represents the mathematics variance.

For any two elements in $\boldsymbol{x}^k$, x_i^k and $x_j^k (i \neq j)$ are mutually independent. Then, there exist $\mathbb{E}[x_i^k x_j^k] = \mathbb{E}[x_i^k]\mathbb{E}[x_j^k] = 0$, when $i \neq j$ and $\mathbb{E}[x_i^k x_i^k] = \mathbb{D}[x_i^k] - \mathbb{E}^2[x_i^k] = 1$, which can be written as

$$\mathbb{E}[x_i^k x_j^k] = \begin{cases} 1, & i = j \\ 0, & i \neq j. \end{cases} \tag{25}$$

According to (25), we can get that

$$\mathbb{E}[\boldsymbol{x}^k \boldsymbol{x}^{k^T}] = \begin{bmatrix} \mathbb{E}\left[x_1^k x_1^k\right] & \cdots & \mathbb{E}\left[x_1^k x_d^k\right] \\ \vdots & \ddots & \vdots \\ \mathbb{E}\left[x_d^k x_1^k\right] & \cdots & \mathbb{E}\left[x_d^k x_d^k\right] \end{bmatrix}_{d\times d} = \boldsymbol{I}. \quad (26)$$

Because $\boldsymbol{x}^{k+1}$ is sampled from a fixed distribution that is uncorrelated to P, there exist

$$\begin{aligned} \mathbb{E}\left[P(k)\boldsymbol{x}^{k+1}\boldsymbol{x}^{k+1^T}P(k)\right] &= \mathbb{E}[P(k)]\mathbb{E}\left[\boldsymbol{x}^{k+1}\boldsymbol{x}^{k+1^T}\right]\mathbb{E}[P(k)] \\ &= \mathbb{E}[P(k)]\boldsymbol{I}\mathbb{E}[P(k)] \\ &= \mathbb{E}[P(k)]\mathbb{E}[P(k)] \end{aligned} \quad (27)$$

and according to (25)

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}^{k+1^T}P(k)\boldsymbol{x}^{k+1}] &= \mathbb{E}\left[\sum_{i=1}^{d}\sum_{j=1}^{d} P_{ij}(k)x_i^{k+1}x_j^{k+1}\right] \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d}\mathbb{E}\left[P_{ij}(k)x_i^{k+1}x_j^{k+1}\right] \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d}\mathbb{E}\left[P_{ij}(k)\right]\mathbb{E}\left[x_i^{k+1}x_j^{k+1}\right] \\ &= \sum_{i=1}^{d} P_{ii}(k) = \text{trace}(P(k)) \end{aligned} \quad (28)$$

where trace$(\cdot)$ is the trace of the matrix and $P_{ij}(k)$ is the element in the ith row and the jth column in $P(k)$ that is denoted as

$$P(k) = \begin{bmatrix} P_{11}(k) & \cdots & P_{1d}(k) \\ \vdots & \ddots & \vdots \\ P_{d1}(k) & \cdots & P_{dd}(k) \end{bmatrix}_{d\times d}. \quad (29)$$

Because α is close to zero, we ignore it in (24). Then, according to (27) and (28), the mathematical expectation of $P(k+1)$ can be written as

$$\begin{aligned} \mathbb{E}[P(k+1)] &\approx \mathbb{E}[P(k)] - \frac{\mathbb{E}\left[P(k)\boldsymbol{x}^{k+1}\boldsymbol{x}^{k+1^T}P(k)\right]}{\mathbb{E}\left[\boldsymbol{x}^{k+1^T}P(k)\boldsymbol{x}^{k+1}\right]} \\ &= \mathbb{E}[P(k)] - \frac{\mathbb{E}[P(k)]\mathbb{E}[P(k)]}{\text{trace}(P(k))}. \end{aligned} \quad (30)$$

Because $P(0) = \boldsymbol{I}$, we can get the mathematical expectation of $P(k+1)$ in (30)

$$\begin{aligned} \mathbb{E}[P(1)] &= \boldsymbol{I} - \frac{\boldsymbol{I}}{d} = \frac{d-1}{d}\boldsymbol{I} \\ \mathbb{E}[P(2)] &= \frac{d-1}{d}\boldsymbol{I} - \frac{\left(\frac{d-1}{d}\right)^2\boldsymbol{I}}{\frac{d-1}{d} * d} = \left(\frac{d-1}{d}\right)^2\boldsymbol{I} \\ &\vdots \\ \mathbb{E}[P(k)] &= \left(\frac{d-1}{d}\right)^k\boldsymbol{I} \\ \mathbb{E}[P(k+1)] &= \left(\frac{d-1}{d}\right)^{k+1}\boldsymbol{I}. \end{aligned} \quad (31)$$

In practice, the length of latent vector is usually set to $d = 100$ and the number of updating steps $k = 2000$, and then, we have $\mathbb{E}[P(2000)] = 1.86 \times 10^{-9}\boldsymbol{I} \approx \boldsymbol{O}$. Even when $k = 1000$, $\mathbb{E}[P(1000)] = 4.32 \times 10^{-5}\boldsymbol{I} \approx \boldsymbol{O}$. Finally, the proof is completed. ■

APPENDIX B
SOLVING THE P_l IN (7)

To make the computation of matrix P_l more efficient, we can apply the RLS algorithm [36] to recursively compute P_l example by example rather than solve matrix P_l after collecting all the examples.

Before we formally derive the iterative computation formula of P_l, first review an important proposition in [36] as follows:

Proposition 2.1: Let A and B be two positive-definite $M \times M$ matrices related to

$$A = B^{-1} + CD^{-1}C^T \quad (32)$$

where D is a positive-definite $N \times N$ matrix and C is an $M \times N$ matrix. According to the matrix inversion lemma, we may express the inverse of the matrix A as

$$A^{-1} = B - BC(D + C^T BC)^{-1}C^T B. \quad (33)$$

Now, we start the derivation of the iterative computation formula about P_l. Let $A_l(k)$ denote the matrix consisting of the first k features in A_l, i.e., $A_l(k) = [\boldsymbol{x}_l^1, \boldsymbol{x}_l^2, \ldots, \boldsymbol{x}_l^k]$. After importing the $(k+1)$th feature $\boldsymbol{x}_l^{k+1}$, we have $A_l(k+1)A_l^T(k+1) = A_l(k)A_l^T(k) + \boldsymbol{x}_l^{k+1}\boldsymbol{x}_l^{k+1^T}$. Let $P_l(k)$ denote the orthogonal projection matrix corresponding to $A_l(k)$. Then, according to (7), we have

$$\begin{aligned} P_l(k+1) &= \alpha\left(\hat{A}_l(k+1)\hat{A}_l(k+1)^T + \alpha I\right)^{-1} \\ &= \alpha\left(\hat{A}_l(k)\hat{A}_l^T(k) + \alpha I + \boldsymbol{x}_l^{k+1}\boldsymbol{x}_l^{k+1^T}\right)^{-1}. \end{aligned} \quad (34)$$

In (34), let $B^{-1} = A_l(k)A_l^T(k) + \alpha I$, and then, according to (7), we have $B^{-1} = \alpha P_l^{-1}(k)$. Furthermore, let $C = \boldsymbol{x}_l^{k+1}$ and $D = 1$. According to (32) and (33), we can easily obtain the computation formula about $P_l(k+1)$, i.e.,

$$P_l(k+1) = P_l(k) - \frac{P_l(k)\boldsymbol{x}_l^{k+1}\boldsymbol{x}_l^{k+1^T}P_l(k)}{\alpha + \boldsymbol{x}_l^{k+1^T}P_l(k)\boldsymbol{x}_l^{k+1}}. \quad (35)$$

The advantages of computing matrix P_l according to (8) lie in two aspects. First, it makes the computation very efficient since the computation of the inverse of big matrix is not required. Second, when learning a new task, there is no need to store all the features for old tasks in A_l and only ones for the current task, which saves a lot of storage space.

REFERENCES

[1] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, abs/1701.07875, pp. 1–19, Jun. 2017.

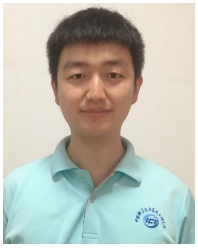
[2] A. Chaudhry, P. K. Dokania, T. Ajanthan, and H. S. P. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proc. 15th Eur. Conf.*, Munich, Germany, Sep. 2018, pp. 556–572.

[3] G. Cohen, S. Afshar, J. Tapson, and A. V. Schaik, "EMNIST: An extension of MNIST to handwritten letters," *CoRR*, abs/1702.05373, pp. 1–10, May 2017.

[4] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognit. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.

[5] H. Gene Golub and F. Charles Van Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[6] J. Ian Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2014, pp. 2672–2680.
[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Jun. 2017, pp. 5767–5777.
[8] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1026–1034.
[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.
[10] X. He and H. Jaeger, "Overcoming catastrophic interference using conceptor-aided backpropagation," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–11.
[11] 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 *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6626–6637.
[12] J. Nicholas Higham, *Accuracy and Stability of Numerical Algorithms* (Society for Industrial and Applied Mathematics). Philadelphia, PA, USA: SIAM, 2002.
[13] E. Geoffrey Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, pp. 1–9, Mar. 2015.
[14] P. Isola, J. Zhu, T. Zhou, and A. Alexei Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 21-26, pp. 5967–5976.
[15] H. Jaeger, "Controlling recurrent neural networks by conceptors," *CoRR*, abs/1403.3369, pp. 1–4, Mar. 2014.
[16] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *Proc. 6th Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–26.
[17] P. Diederik Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–15.
[18] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *CoRR*, abs/1612.00796, pp. 1–25, Mar. 2016.
[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Comput. Sci. Dept., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 1, Jan. 2009.
[20] Y. LeCun. (1998). *The Mnist Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, p. 3.
[22] X. Mao, Q. Li, H. Xie, Y. K. Raymond Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 2813–2821, 2017.
[23] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motivat.*, vol. 24, pp. 109–165, Dec. 1989.
[24] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, pp. 1–5, Apr. 2014.
[25] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. 6th Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–26.
[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.
[27] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, Aug. 2017, pp. 2642–2651.
[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–16.
[29] A. Rios and L. Itti, "Closed-loop memory GAN for continual learning," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, Aug. 2019, pp. 3332–3338.
[30] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[31] J. Schwarz *et al.*, "Progress & compress: A scalable framework for continual learning," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 4535–4544.
[32] A. Seff, A. Beatson, D. Suo, and H. Liu, "Continual learning in generative adversarial nets," *CoRR*, vol. abs/1705.08395, pp. 1–9, May 2017.
[33] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 2990–2999.
[34] Stanford. (2017). *Tiny Imagenet*. [Online]. Available: https://www.kaggle.com/c/tiny-imagenet/
[35] Y. Wang, C. Wu, L. Herranz, J. V. D. Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring GANs: Generating images from limited data," in *Proc. 15th Eur. Conf.*, Munich, Germany, Sep. 2018, pp. 220–236.
[36] B. Wittenmark, "Adaptive filter theory: Simon haykin," *Automatica*, vol. 29, no. 2, pp. 567–568, 1993.
[37] C. Wu, L. Herranz, X. Liu, Y. wang, J. V. D. Weijer, and B. Raducanu, "Memory replay GANs: Learning to generate new categories without forgetting," in *Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 5962–5972.
[38] F. Ye and G. Adrian Bors, "Learning latent representations across multiple data domains using lifelong VAEGAN," in *Proc. 16th Eur. Conf.*, Glasgow, U.K., Aug. 2020, pp. 777–795.
[39] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Mach. Intell.*, vol. 1, no. 8, pp. 364–372, Aug. 2019.
[40] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, "Lifelong GAN: Continual learning for conditional image generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Seoul, South Korea, Nov. 2019, pp. 2759–2768.
[41] H. Zhang, T. Xu, and H. Li, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 5908–5916.
[42] J. Y. Zhu *et al.*, "Toward multimodal image-to-image translation," in *Proc. Adv. Neural Inf. Process Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 465–476.



Xiaobin Li (Graduate Student Member, IEEE) received the B.E. degree from Nankai University, Tianjin, China, in 2016. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China.

His research interests include machine learning, pattern recognition, image processing, and computer vision.

Weiqiang Wang (Member, IEEE) received the B.E. and M.E. degrees in computer science from Harbin Engineering University, Harbin, China, in 1995 and 1998, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing, China, in 2001.

He is currently a Professor with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing. His research interests include multimedia content analysis and computer vision.