Pattern Recognition 151 (2024) 110383

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Introspective GAN: Learning to grow a GAN for incremental generation and classification

Chen He, Ruiping Wang *, Shiguang Shan, Xilin Chen

Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
University of Chinese Academy of Sciences, Beijing, 100049, China

ARTICLE INFO

Keywords:
Incremental learning
Catastrophic forgetting
Generative Adversarial Networks

ABSTRACT

Lifelong learning, the ability to continually learn new concepts throughout our life, is a hallmark of human intelligence. Generally, humans learn a new concept by knowing **what it looks like** and **what makes it different from the others**, which are correlated. Those two ways can be characterized by generation and classification in machine learning respectively. In this paper, we carefully design a dynamically growing GAN called **Introspective GAN (IntroGAN)** that can perform incremental generation and classification simultaneously with the guidance of prototypes, inspired by their roles of efficient information organization in human visual learning and excellent performance in other fields like zero-shot/few-shot/incremental learning. Specifically, we incorporate prototype-based classification which is robust to feature change in incremental learning and GAN as a generative memory to alleviate forgetting into a unified end-to-end framework. A comprehensive benchmark on the joint incremental generation and classification task is proposed and our method demonstrates promising results. Additionally, we conduct comprehensive analyses over the properties of IntroGAN and verify that generation and classification can be mutually beneficial in incremental scenarios, which is an inspiring area to be further exploited. The code is available at https://github.com/TonyPod/IntroGAN.

1. Introduction

Lifelong learning is one of the most essential abilities for humans to live in the ever-changing environment. Generally, humans learn new visual concepts from two aspects: *what it looks like* and *what makes it different from the others*. For example, when children first learn the visual concept *dog* after seeing a bunch of dog images, they can summarize what a typical dog looks like. By changing their color or texture, children can envisage many different dogs in their minds. On the other hand, by correlating *dog* with the previously learned *cat*, they can find their differences and learn to distinguish between those two similar concepts. The aforementioned mental processes – imagination and recognition – seem correlated and the inability of either one impairs human visual learning (Fig. 1). In machine learning, those two mental processes can be characterized by generation and classification task respectively.

Lifelong machine learning [1], which studies the ongoing learning ability of machines, can similarly extend to two branches: incremental generation and incremental classification. While previous works pay much attention to incremental classification, those two branches are correlated and can be mutually beneficial in that (Fig. 2):

1. **Classification needs generation to replay old memory and boost generalization ability**. Classification learns discriminative features and decision boundaries, making it highly dependent on which classes the algorithm has seen so far. However, in lifelong learning we cannot foresee the incoming classes, so the discriminative features and decision boundaries need to be adjusted over time, which is problematic if we cannot replay the old memory. For example, if we can only leverage a single binary feature to classify *horse* and *dog*, the learned feature might be *size* of the object. Assuming that we need to learn *bull* next, to distinguish it from the previously learned *horse*, one must recall that *horse* does not have horns while *bull* does, which is difficult for most discriminative models since they do not care about features that are unimportant for the current task. To address this problem, a trivial solution is to store all the previously seen samples, which is memory inefficient. While storing a few typical samples (or to say *episodic memory* as in [2]) is a feasible solution, we think that it can be augmented with a *generative memory*, which can better depict the class distribution and offer

* Corresponding author at: Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China.
E-mail address: wangruiping@ict.ac.cn (R. Wang).

https://doi.org/10.1016/j.patcog.2024.110383
Received 10 November 2020; Received in revised form 15 December 2023; Accepted 26 February 2024
Available online 27 February 2024

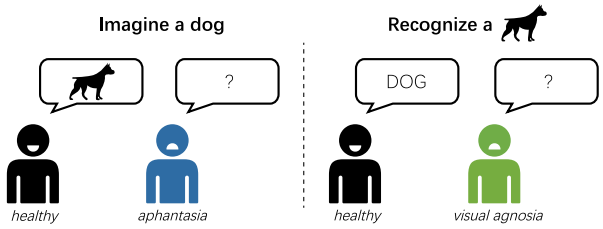

Fig. 1. Illustrations of the inabilities of imagination and visual recognition (i.e. *aphantasia* where one cannot voluntarily visualize imagery [4] and *visual agnosia* where one cannot recognize visually presented objects [5]). People that suffer from *aphantasia* have unusually low Vividness of Visual Imagery Questionnaire (VVIQ) scores, which is similar to the characteristics of *prosopagnosia* (a variant of *visual agnosia* in face recognition) [6], indicating possible underlying relationship between those two inabilities [4]. In each sub-figure above, the person on the left is a healthy person, whereas the person on the right has the inability.

much more sample diversity, leading to a more generalizable classifier [3].

2. **An additional classifier can generate discriminative images and stabilize the generation process**. Instead of trying to mimic every detail of the original image, the added classifier may guide the generator to emphasize more the discriminative features and generate samples that really look like the given category. In addition, experimental results in Section 4.4 show that an unsupervised generator may be dominated by the easy category and gradually forget to generate samples of other categories in incremental scenarios. However, adding an additional classifier can eliminate this effect and help stabilize the generation process.

By now we have shown the importance of generation and classification in both human learning and machine learning, which inspires us to design a unified approach that brings generation and classification together to tackle the incremental learning problem. As for image generation, Generative Adversarial Networks (GANs) have proved their superior performance and can be adopted as a generative memory in incremental learning to mitigate catastrophic forgetting. As for image classification, inspired by the role of prototypes for their excellent performance in zero-shot [7]/few-shot [8]/incremental [9] learning and image classification [10], we convert conventional classification to prototype-based classification where the introduced prototypes can also be used in generation. Those ideas are further unified in an end-to-end GAN framework named **Introspective GAN (IntroGAN)**[1] which can continually learn new visual concepts and perform incremental generation and classification simultaneously. As far as we are concerned, we are among the first to propose the joint incremental generation and classification task along with a comprehensive benchmark and evaluation metrics. Comprehensive analyses show that IntroGAN yields promising results on benchmark datasets and challenging settings like low-shot or feature classification. More analyses are performed and we demonstrate that incremental generation and classification can be mutually beneficial.

Our contributions are as follows:

1. We propose a new benchmark to evaluate the joint incremental generation and classification task. We demonstrate that generation and classification can be mutually beneficial in incremental scenarios with experimental results and in-depth analysis.

2. We propose a dynamically growing GAN called Introspective GAN (IntroGAN) to address the joint incremental and classification task with the help of prototypes, which demonstrate competitive performance on proposed benchmarks and more challenging scenarios like low-shot and large-scale datasets.

3. We introduce a novel GAN fine-tuning technique that can accelerate GAN training in incremental scenarios.

2. Related works

Lifelong Learning. Lifelong (machine) learning is an advanced machine learning paradigm that learns continuously, accumulates knowledge learned in previous tasks, and leverages it to facilitate future learning [1]. It has other names with subtle differences such as continual learning [11,12], incremental learning [13,14] and continuous learning [15] etc., which are used interchangeably for most researchers. A notorious phenomenon that often co-occurs with lifelong learning is catastrophic forgetting [16], which indicates that the newly learned patterns may completely erase the previous acquired knowledge. According to stability-plasticity dilemma theory [17], forgetting is essentially an expression of the excessive plasticity. Based on the ways of reviewing historical data, existing methods that mitigate catastrophic forgetting can roughly fall into three categories [18]: *rehearsal* which explicitly reviews old data (e.g. exemplars) [9,18,19], *pseudo-rehearsal* which implicitly reviews old data (e.g. pseudo-patterns [20–22], GANs [18,23,24], VAEs [24] or autoencoders [25]), and *non-rehearsal* which isolates the old parameters and does not review old data [26–28]. Since our method leverages both exemplars and generated data by GANs for memory replay, it falls between *rehearsal* and *pseudo-rehearsal*.

Leveraging Prototypes. The prototype, which has its origins in Eleanor Rosch's early study [29], means the most central member of a category. In computer vision it has been overloaded with diverse meanings: a large spectrum of prototype-based classification methods rely on merely one prototype for each class and predict the label of a test sample by finding its nearest prototype. Since those methods often use the class mean as the prototype for each class, they can all fall into the NCM-like methods (NCM is short for Nearest Class Mean [30]), which have been successfully applied in Zero-Shot Learning [7,31], Few-Shot Learning [8,32], Incremental Learning [9] etc. Apart from assigning one prototype for each class, there are approaches that leverage multiple prototypes [10,32–34], which are quintessential observations that best represent the dataset (or better say exemplars). Inspired by [8,9], our framework leverages multiple exemplars to generate a robust prototype for each class in the feature space and adopts NCM-like strategy for classification (more details in Section 3.2). The differences with [8,9] are further elaborated in Section 3.5. Since the exemplars are selected from real samples instead of synthesized ones, they can be added to the training set to train the network as well in IntroGAN.

Generative Adversarial Networks. The last few years have witnessed a huge success of Generative Adversarial Networks (GANs) in image generation [35–40]. Apart from generating photorealistic images, GANs have also been employed in other fields such as Zero-Shot Learning [41], Image Translation [42], Image Classification [43] etc., and researchers show increasing interest in applying GANs in lifelong learning scenarios [18,23,44–48]: [44–48] mainly focus on adapting GANs to the problem of incremental generation, whereas [18,23] focus on incremental classification and use GANs for memory replay. Instead of focusing on generation or classification alone via GANs in previous works, we focus on the joint incremental generation and classification task, motivated by the aforementioned human visual learning mechanism (Section 1). To tackle the joint task, we propose a novel GAN framework which leverages prototypes and other useful techniques (Section 3). More discussions with some related works including the

[1] "Introspective" in our paper simply means that compared to usual incremental classification methods that focuses on knowing "what makes it different from the others", IntroGAN further reflects more on "what it (a certain class) looks like". (Vanilla) GANs treat all classes as a whole and it is hard for them to reflect on a certain class. That is why we attach the word "Introspective" to "GAN" to emphasize the ability to reflect on a certain class.

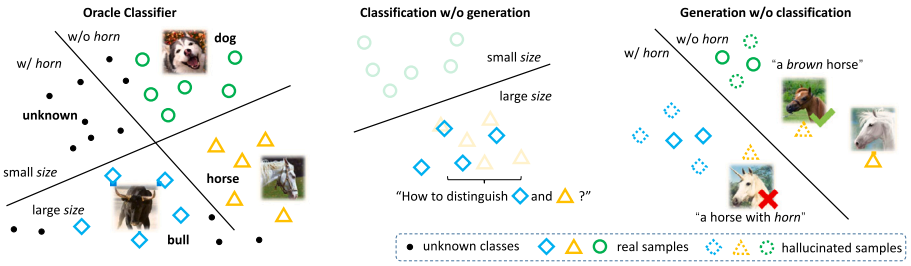

Fig. 2. A illustrative diagram of why incremental generation and classification are mutually beneficial in lifelong learning. The figure is at the time when we have seen *horse*, *dog*, and first encounter *bull*. ***Left***: underlying samples and decision boundaries of binary classifiers in the feature space (whether seen or unseen). ***Middle***: due to the inability to replay old memory, we cannot remember whether *horse* has *horn* or not; by merely using the previously learned feature *size*, *bull* and *horse* are overlapped and cannot be distinguished (light-colored circles or triangles mean that the samples do not exist anymore). ***Right***: without the guidance of the classifier, the generated sample may not look like the given category and is actually in the region of other classes.

comparisons between GANs and the popular diffusion models [49] in recent years are shown in Sec. S2.[2]

3. Introspective GAN (IntroGAN)

3.1. Problem formulation

While conventional incremental learning assumes that data continually arrive and a new sample may come from an arbitrary class (whether old or new), we adopt a simpler but more frequently used setting called Class Incremental Learning [9] which assumes that samples of a class or a batch of classes arrive at a time. We assume that k classes are added at a time ($k \geq 1$), then at time t the indices of the new classes are $(t-1)k+1, \ldots, tk$. $X^{(c)}_{train}$ and $X^{(c)}_{test}$ are the training and test samples of class c respectively. For each time step t, the objective of our task is to achieve good generation and classification results on the test set of the seen tk classes, i.e. $\{X^{(1)}_{test}, X^{(2)}_{test}, \ldots, X^{(tk)}_{test}\}$. Supposing that there are T class increments, then there will be eventually $K = Tk$ seen classes.

3.2. General idea of IntroGAN

The idea of IntroGAN is to make good use of prototypes in both generation and classification. For clarity, the prototype means the representative point for each class in the feature space, which can be either fixed or obtained on-the-fly. For disambiguation, we assume that each class can have only one prototype but multiple exemplars. As for classification, we convert the conventional inner-product based classifier[3] into a prototype-based classifier. The reason is that prototype-based classification methods learn a shared embedding space where samples cluster around their corresponding prototypes, which can generalize to novel classes at zero cost [33] and exhibit excellent performance in Zero/Few-Shot Learning [7,8]. That is the initial motivation why the prototype-based classifier is adopted in IntroGAN.

The next problem is whether to obtain prototypes as fixed points or dynamic points in the feature space. In usual incremental learning scenarios, the feature extractor should be continually updated (as illustrated in Fig. 2, more features should be introduced when we want to classify more classes), therefore absolute coordinates in the feature space will become useless once the feature extractor updates (*absolute prototype* in Fig. 3a). Thus, the prototypes should change dynamically with the update of the feature representation. One simple way to achieve this goal is to always use the feature of a fixed image sample as a prototype, but the problem is obvious since it may move far from the class centroid in certain feature space, making it less representative of the corresponding class (*fixed prototype* in Fig. 3b). Another solution is to treat the closest exemplar to the test sample as a prototype, but it may incur noise when some exemplars deviate too far from the class distribution and becomes less reliable (*closest prototype* in Fig. 3c). A better alternative is to keep several exemplar images and use the average of their features as the prototypes [8,9], which is more robust to feature change problem (*relative prototype* in Fig. 3d). According to the Law of Large Numbers, the more exemplar images, the more accurate for the *relative prototype* to estimate the real class centroid in the feature space. The experimental results of different choices of prototypes mentioned above are further visited in Sec. S6.4.

As for generation, the stored exemplars can be used as extra training samples and we add them to the training set. Therefore, the replayed samples compose of both generative and episodic memory, which can benefit from both the diversity offered by generated samples and the authenticity provided by real exemplars. The schematic illustration of IntroGAN is shown in Fig. 4. For convenience, we use G for the generator, D for the discriminator, and C for the classifier, F for the feature extractor hereinafter. In Sections 3.3 and 3.4, we elaborate on the non-incremental formulation and incremental training respectively.

3.3. Method formulation

AC-GAN [36] is originally used for image generation. Since it is endowed with a classifier, it has the potential to perform generation and classification at the same time, which is a good starting point to further incorporate the idea of prototypes in the previous subsection. The original **generator loss** and the **discriminator loss** in AC-GAN are as follows:

$$L_G = -E_{z\sim P_z, c\sim P_c}[log(D(G(z,c)))] \tag{1}$$

$$-\alpha E_{z\sim P_z, c\sim P_c}[y_c log(C(G(z,c)))] \tag{2}$$

$$L_D = -E_{(x,c)\sim X_{train}}[log(D(x))] \tag{3}$$

$$-E_{z\sim P_z, c\sim P_c}[log(1-D(G(z,c)))] \tag{4}$$

$$-\beta E_{(x,c)\sim X_{train}}[y_c log(C(x))] \tag{5}$$

where α and β are weighting factors. z is a noise vector that obeys Gaussian distribution P_z. c is the class label. P_c is the uniform distribution that covers all seen classes. x is an input image. y_c is the one-hot embedding of the ground-truth label. The losses above are the original non-saturating GAN loss [50] (Eq. (1), (3), (4)), cross entropy loss on generated samples (Eq. (2)) and cross entropy loss on real samples (Eq. (5)) respectively.

To incorporate prototype-based classification, we replace the inner product in the conventional softmax function with the minus distance between a given sample and the class prototype. The prototype is the

[2] For simplicity, we use "S" to denote the reference to the supplementary material hereinafter (e.g. Sec. S2 indicates Sec. 2 in the supplementary material).

[3] By inner-product we mean that the network has the form $\mathbf{W}^T f(\mathbf{x}) + \mathbf{b}$ which is implemented by attaching a fully connected layer at the end.

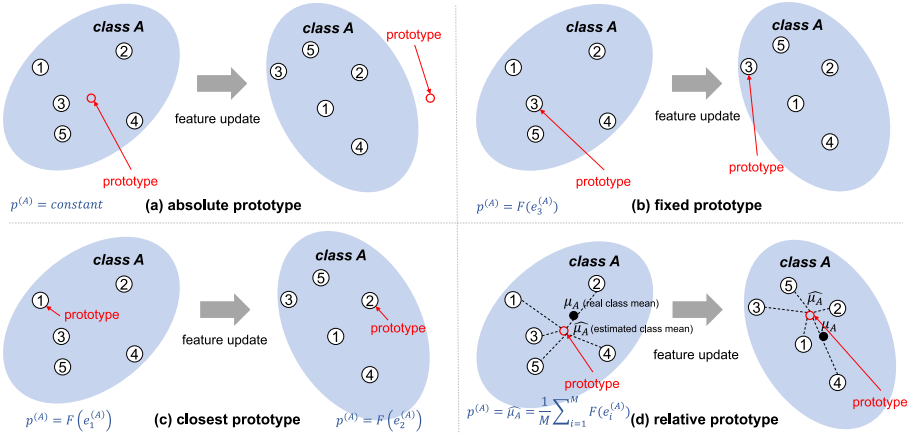

Fig. 3. Illustration diagrams of different prototype selection strategies before and after the feature update. The light blue oval denotes the class distribution. The circle with number i is the ith exemplar in the feature space. p^A is/are the prototype(s) of class A. $e_i^{(A)}$ is the ith exemplar of class A ($i \in \{1, \ldots, M\}$).

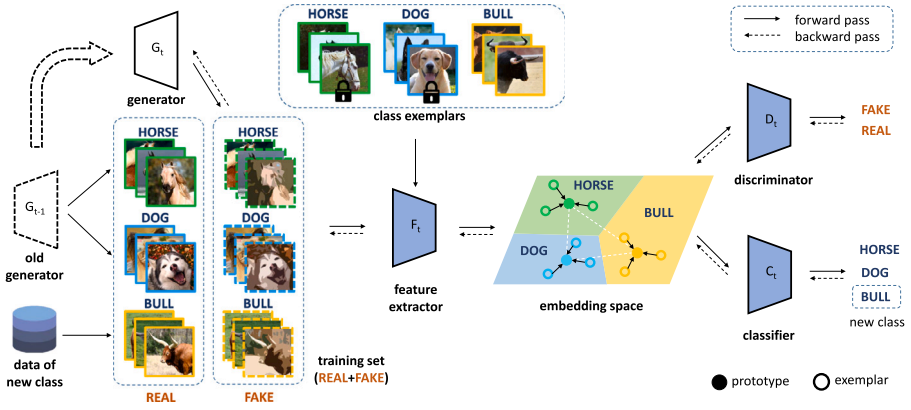

Fig. 4. Schematic illustration of IntroGAN pipeline at time t when it has learned *horse* and *dog* and the incoming class is *bull*. We train the model by first constructing a balanced training set that consists of: samples of *horse* and *dog* generated by the old generator G_{t-1}, exemplars of *horse* and *dog*, and the real samples of *bull*. Exemplars of the new class are carefully selected, whereas those of the old classes are fixed.

mean of the embedded exemplars of the corresponding class in the feature space:

$$p^{(c)} = \frac{1}{M}\Sigma_{i=1}^{M} F(e_j^{(c)}) \tag{6}$$

In Eq. (6), $p^{(c)}$ is the prototype of class c, $e_j^{(c)}$ is the jth exemplar of class c. M is the number of exemplars for class c. F is the feature extractor. The probability of a given input x on class c at time t is defined as:

$$C(c|x) = \frac{\exp(-\gamma d(F(x), p^{(c)}))}{\Sigma_{i=1}^{t} \exp(-\gamma d(F(x), p^{(i)}))} \tag{7}$$

In Eq. (7), γ is the parameter that controls the smoothness of the output probabilities, and the readers could refer to Sec. S5 for how to set a proper γ. The distance function d is simply defined as the most commonly used squared L2 distance (a.k.a. squared Euclidean distance), which can be roughly seen as the posterior probability based on the assumptions that: (1) features of each class obeys a multivariate Gaussian distribution with an isotropic covariance matrix; (2) each class has equal prior probability. The full derivation can be found in Sec. S3.

Note that since we employ the nearest-prototype classifier, the classifier does not have any trainable parameters. The feature extractor F and the prototypes are what we need to learn that may change the behavior of the classifier.

For prediction, we simply choose the label with the highest probability as the predicted class label as most prototype-based classifiers do:

$$\hat{c} = \arg\max_{c \in [1,t]} C(c|x) \tag{8}$$

For training, instead of directly choosing the feature mean as the prototype shown in Eq. (7), we want to make the best use of the exemplar labels since the feature mean is definitely inside the convex hull spanned by exemplars, making "classifying a sample to the closest feature mean" a very relaxed constraint. To emphasize more on the importance of each exemplar, we adopt a *selection* operation (denoted as $s(\cdot)$) to aggregate the information from all exemplars:

$$C(c|x) = \frac{\exp(-\gamma s(d(F(x), F(e_j^{(c)}))))}{\Sigma_{i=1}^{t} \exp(-\gamma s(d(F(x), F(e_j^{(i)}))))} \tag{9}$$

The above equation may add some useful perturbation and increase the discriminability because the model that satisfies Eq. (9) will be very likely to satisfy Eq. (7). Different choices of the *selection* functions (e.g. *min*, *max*, *random*, *mean*) are further evaluated in Sec. S6.4.

3.4. Training in incremental learning scenarios

By now, we have defined the losses of IntroGAN. The generator loss and the discriminator loss are optimized in turn over the corresponding parameters respectively. There are different choices to select the exemplars for each class, for example using the real samples that are closest to the cluster centers by K-means. These choices will be analyzed in Sec. S6.4.

In the first class increment, there are no old classes and the training of IntroGAN is similar to the conventional AC-GAN [36] training except that we need to calculate the prototype as well. In the subsequent class increments (let us assume time t), we construct a balanced training set that covers both old and new classes: for old class c ($1 \leq c \leq (t-1)k$), we use the trained generator G to generate the samples $X_{gen}^{(c)}$ combined with real exemplars $X_{exem}^{(c)}$; for new class c ($(t-1)k < c \leq tk$), we simply use all the training samples $X_{train}^{(c)}$. Thus, the training set at time t is $\{X_{gen}^{(1)}, \ldots, X_{gen}^{((t-1)k)}, X_{exem}^{(1)}, \ldots, X_{exem}^{((t-1)k)}, X_{train}^{((t-1)k+1)}, \ldots, X_{train}^{(tk)}\}$. The corresponding pseudo-code is shown in Alg. 1.

Algorithm 1 Training of IntroGAN at time t

Input: $X_{train}^{((t-1)k+1)}, \ldots, X_{train}^{(tk)}$ // new class samples at time t

Output: $\{D_t, C_t, G_t, F_t\}$ and $\{X_{exem}^{((t-1)k+1)}, \ldots, X_{exem}^{(tk)}\}$ // new model and exemplars of the new classes at time t

Require: exemplars $X_{exem}^{(1)}, \ldots, X_{exem}^{(t-1)k}$ and model $\{D_{t-1}, C_{t-1}, G_{t-1}, F_{t-1}\}$

$X_{gen}^{(1)}, \ldots, X_{gen}^{(t-1)k} \leftarrow G_{t-1}$ // replay old class samples using G_{t-1}

$X_{train} \leftarrow \{X_{gen}^{(1)}, \ldots, X_{gen}^{((t-1)k)}, X_{exem}^{(1)}, \ldots, X_{exem}^{((t-1)k)}, X_{train}^{((t-1)k+1)}, \ldots, X_{train}^{(tk)}\}$ // create a balanced training set

$\{D_t, C_t, G_t, F_t\} \leftarrow \{D_{t-1}, C_{t-1}, G_{t-1}, F_{t-1}\}$ // improved initialization

Initialize exemplars $X_{exem}^{((t-1)k+1)}, \ldots, X_{exem}^{(tk)}$ for new classes at time t

for $i \leftarrow 1$ **to** *iterations* **do**

  Update $\{D_t, C_t, G_t, F_t\}$ using a batch of X_{train}

  if i **mod** *interval* $= 0$ **then** update $X_{exem}^{((t-1)k+1)}, \ldots, X_{exem}^{(tk)}$

GAN training is time-consuming! It usually takes over 10^5 iterations to train GANs from scratch and the number will be multiplied by T if there are T class increments in incremental learning scenarios. A better alternative is to introduce fine-tuning technique commonly used in CNN into GAN, which is a hardly visited topic. Considering that the network has learned a handful of classes and is going to learn a closely related class, it is natural to borrow knowledge from before based on the class similarity instead of initializing randomly. We achieve this (forward) knowledge transfer by first finding which old class the new class is most confused with using the existing classifier, then initializing the exclusive weights of the new class with those of the most related old class (the exclusive weights mean the final fully-connected layer in the classifier or the first layer of the generator where the random noise and the one-hot class label are concatenated). The mathematical form of this **improved initialization** is:

$$W_{new} = W_{old}^{*} + W_{noise} \tag{10}$$

where W_{old}^{*} is the weight of the most confused old class. W_{noise} is an extra perturbation term to prevent completely duplicating the old weight from old classes which may result in the extra burdens for the network to tell those two similar classes apart later on. It is a tensor sampled from normal Gaussian distribution $N(0, \lambda\sigma)$ with the same shape as W_{old}^{*}, where σ is the standard deviation of W_{old} and λ is a weighting factor. We find that improved initialization is a general technique that can be applied to IntroGAN and other methods as well (refer to the experimental results in Sec. S6.4).

3.5. Differences with relevant methods

Incremental Learning Methods. Before IntroGAN, there is no method specifically designed to perform incremental generation and classification simultaneously. However, there are works that have the potential to do both. In Table 1, we list typical works in incremental generation and classification together with their potentials and characteristics. Note that only the first five have the potential to perform incremental generation and classification, where others can handle one task only. Among them, DGR [23] has a generator, a discriminator and a classifier, while ESGR [18] has independent generators and one classifier. Both of them are not end-to-end and are memory inefficient. DGM [45] is an interesting work that introduces hard attention [51] and expandable models [27] to GANs, which is quite different from other works. The most similar work to IntroGAN is MeRGAN-JTR [44]. Since MeRGAN-JTR is also based on AC-GAN, it has the potential to perform incremental generation and classification as well. From the perspective of method formulation, the difference of IntroGAN with MeRGAN-JTR mainly lies in the use of prototypes which have the following merits: prototypes can give superior classification performance which is attributed to the more robust prototype-based classifier already mentioned in Section 3.2; the stored exemplars can serve as extra training samples which offer more authentic information for both generation and classification. Although MeRGAN-JTR has the potential for the joint task, we should notice that it is oriented for incremental generation only and the classification accuracy reported in its paper is to evaluate the quality of the generated images using an oracle classifier, which is intrinsically a performance measure for generation. We use a slightly modified version when reporting its performance throughout the paper.[4] The idea of using a prototype-based classification strategy is similar to iCaRL [9]. However, iCaRL learns feature representation and the classifier separately, making it sensitive to the choice of the feature extractor while we train the classifier in an end-to-end manner which is more robust. Also, we further analyze how to obtain the prototype in the training and test phase (Sec. S6.4), where a special case that uses a *mean* selection function is similar to the prototype-based classifier in iCaRL.

Few-Shot Learning Methods. In the spectrum of few-shot learning, [52] also trains classification and generation network in an end-to-end fashion. Apart from the task-specific considerations, the technical differences with ours are two-folds: (1) *Generating images* (ours) v.s. *generating features*: since the feature extractor needs to be updated continually in incremental learning as elaborated in Section 3.2 in the main paper. Generating features in [52] is impractical since the features cannot be used when the feature extractor updates. (2) *Learning class distribution* (ours) v.s. *learning perturbation*: [52] tries to learn a perturbation over each real sample for augmentation, which is practical for generating feature since features of the same class already cluster and the useful perturbation added to each feature may be similar and learnable. But in the original image space, samples of the same class are much distant and the perturbation for each sample is much more different. Therefore, directly learning class distribution is more practical in our problem.

As for few-shot learning methods that leverages prototypes in classification only, Prototypical Network [8] is a typical method of this kind. The difference in using prototypes is that we further analyze how to obtain the prototype in the training and test phase (Sec. S6.4), which provides more useful insights.

4. Experiments

4.1. Experimental setups

Datasets. As far as we are concerned, there is no benchmark that evaluates incremental generation and classification simultaneously. Since incremental generation is more challenging, we mainly refer to datasets adopted in this field when designing the comprehensive benchmark. In the context of incremental generation, MNIST [54] and SVHN [55] are must-do datasets. Apart from MNIST and SVHN, we add a harder version of MNIST called Fashion-MNIST [56] and a subset of down-sampled ImageNet called ImageNet-Dogs (difficulty: MNIST<Fashion-MNIST<SVHN<ImageNet-Dogs). The statistics of them are shown in Table 2.

[4] The original MeRGAN-JTR gives poor accuracy because it is oriented for incremental generation only. By creating a balanced training set combining old generated samples and new real samples as in IntroGAN, the classification accuracy can become much normal as other GAN-based methods and we report the result of this improved version.

Table 1
A comparison of mainstream incremental learning methods and their capabilities and characteristics. GAN is in fact a trivial anti-forgetting technique and we do not list it in the table above. Gen.=Generation. Cls.=Classification. E2E.=End-to-End.

Method	Gen.	Cls.	E2E.	Anti-forgetting Techniques
IntroGAN (ours)	✓	✓	✓	Prototypes
DGM [45]	✓	✓	✓	Attention [51], expandable [27]
MeRGAN-JTR [44]	✓	✓	✓	
ESGR [18]	✓	✓		Prototypes, multiple GANs
DGR [23]	✓	✓		
DDGR [53]	✓	✓		
MeRGAN-RA [44]	✓			Replay alignment loss
iCaRL [9]		✓		Prototypes, distillation loss [21]
LwF [21]		✓		Distillation loss

Table 2
Statistics of datasets in terms of the number of training images per class (Tr./Cls.), the number of the test images per class (Ts./Cls.), the image resolution (Res.), gray or RGB (Color), the number of total classes (Cls.), and the number of total class orders experimented (Ord.), the number of classes to add in each training session (Cls./s). "~" means it is an approximate number. "⋆" means that the training/test set is imbalanced for each class.

Dataset	Tr./Cls.	Ts./Cls.	Res.	Color	Cls.	Ord.	Cls./s
MNIST [54]	6000⋆	1000⋆	28 × 28	Gray	10	5	2
Fashion-MNIST [56]	6000	1000	28 × 28	Gray	10	5	2
SVHN [55]	~7325⋆	~2603⋆	32 × 32	RGB	10	5	2
ImageNet-Dogs [57]	~1300⋆	50	64 × 64	RGB	30	2	10

Evaluation Protocols. FID [58] is used to evaluate generation performance by measuring the distance between the learned distribution and the ground-truth distribution in a fixed feature space. Since FID assumes the features obey a multivariate Gaussian distribution, we think that for conditional generators it is more natural to calculate FID over each class independently and average them (which is more reasonable since it treats each class multivariate Gaussian and emphasizes more on the generation correctness of each class). We denote it average condition FID (FID hereinafter for convenience): $\frac{1}{K}\Sigma_{c=1}^{T} FID_c$. We can get an FID at different time; by connecting those values, we can get an FID curve. To reduce the randomness caused by different class orders, we use multiple class orders (shown in Table 2) and average them. For evaluation of classification, we use the average accuracy (ACC) [9]. Similar to FID, we can get an ACC curve over time. To reflect the overall performance via only one quantitative value to facilitate method comparisons, we first average the values along the curve of each task and then average them over classes. The reason is that the number of points along the curve decreases as the task index increases, but we want to prevent early tasks from dominating this overall value (i.e. to treat each task equally). We denote the resulting metrics TA-ACC (task-average accuracy) for classification and TA-FID for generation respectively.

Compared Methods. In Table 1, DGM, MeRGAN-JTR (denoted as MeRGAN for convenience hereinafter), DGR and ESGR have the potential to perform generation and classification at the same time. Among those, ESGR trains one generator for each class which is memory inefficient and takes a long time to train, thus it is not added for comparison (refer to the memory cost in Sec. S6.5). However, we expect ESGR to achieve excellent results because multiple independent GANs have sufficient capacity and is likely to generate samples with higher quality. For DGM, we use its original codes and hyper-parameters. We only change the setting from adding one class to adding two classes at a time to match our experimental setups. Note that DGM uses a fixed number of epochs instead of a fixed number of iterations, making it hard to draw ACC and FID curves with others, so we only report its TA-ACC and TA-FID. For classification, we add two widely compared methods Learning without Forgetting (LwF) and iCaRL. We also add the result of joint training which uses all historical data to train a classifier (a performance upperbound for classification methods) and a variant of IntroGAN named IntroNet for ablation study (Section 4.4). Trivial solutions like joint training and fine-tuning which are usually seen as the upperbound and lowerbound respectively are also added for comparison.

Implementation Details. The codes are implemented in Tensorflow [59]. To make fair comparisons, the compared methods adopt nearly the same network architecture and hyper-parameters (except DGM and DDGR). For Fashion-MNIST and SVHN, we employ a four layer LeNet-like network; for difficult ImageNet-Dogs, we adopt a 4-ResBlock ResNet with spectral normalization [37]. By default, we apply the improved initialization in Section 3.4 for all methods (for the effectiveness of this technique, refer to Sec. S6.4). For all experiments, the loss is optimized via Adam optimizer [60] for 10,000 iterations with the base learning rate 2×10^{-4}. The number of exemplars for each class M is set to 20 which conforms to other works [61,62]. More analyses on M is presented in Sec. S6.4. λ is heuristically set to 0.5 for all methods and all datasets, which incorporates a tolerable random noise and basically preserves the knowledge from the most related old class. More details are elucidated in the Sec. S5.

4.2. Comparisons with SOTAs

Experiments on four datasets are conducted based on the settings in Table 2. Supposing that two classes are added at a time and the 2nd task is to classify the first four classes (*Task II (1–4)*), the performance of the first task (*Task I (1–2)*) can also be evaluated at this moment. Consequently, we can evaluate *Task* 1 to *Task* t when we are learning *Task* t, and the final result graph resembles an upper triangular matrix (Figs. 5, 6) similar to [28]. The reason for such a form is that comparing the accuracy of a 2-class classifier and the accuracy of a 4-class classifier does not make much sense, since these two tasks are intrinsically different. Learning 4 classes is naturally harder, and we cannot say that the drop of accuracy from the 2-class classifier to a 4-class classifier is forgetting, since these two accuracies are not comparable. However, in our upper triangular form, we can still track the 2-class accuracy of the first 2 classes even we have already learned 4 classes, which can reflect forgetting more precisely.

The overall generation and classification results measured by TA-FID/ACC are shown in Table 3.[5] First of all, DDGR shows very competitive results (esp. TA-ACC), which is due to the higher generation quality of the diffusion model over GAN's. However, since it is based on diffusion models, it also suffers from the extraordinary slowness in training and sampling,[6] which will restrict its actual use in incremental scenarios. More discussions with DDGR is left in Sec. S2, and we focus on other methods below.

On MNIST and Fashion-MNIST, DGR is unsatisfactory and by analyzing the generated images we notice that the generator fails to generate certain class samples for certain class orders, indicating a mode collapse problem which will be further explained in the following paragraph. DGM performs very well on SVHN but less satisfactory on (Fashion-)MNIST, which is probably due to the choice of hyper-parameters instead of the method itself (we simply use its best hyper-parameters but only change the code to adding two classes at a time). On ImageNet-Dogs, we find that DGM tends to predict each sample towards new classes, which accounts for the unsatisfactory results on this dataset. The reason might be that the current hyper-parameters are not suitable for this fine-grained dataset. DGM is a dynamically expandable network while other methods are almost fixed (exactly

[5] The training of GAN-based methods may fail occasionally due to randomness in the training procedure (the generated images are blank). When that happens, we simply rerun the code under the same setting. The numbers reported are all based on successful training.

[6] The sampling speed of DDGR is over 600 times of IntroGAN. More details are in Sec. S2.

Table 3
Overall performance of compared methods measured by TA-ACC and TA-FID on three datasets. ACC and FID are short for TA-ACC and TA-FID respectively. '-' implies that the method does not have the generation ability, which corresponds to Table 1. Dogs is short for ImageNet-Dogs. F-MNIST is short for Fashion-MNIST.

Method	MNIST		F-MNIST		SVHN		Dogs	
	ACC	FID	ACC	FID	ACC	FID	ACC	FID
Joint training	99.16	–	94.18	–	89.64	–	34.45	–
Fine-tuning	41.41	–	39.23	–	38.30	–	18.48	–
DDGR [53]	95.31	58.36	96.73	44.96	85.70	61.91	–	–
DGM [45]	92.15	77.72	79.88	110.90	75.00	95.91	13.69	270.70
MeRGAN [44]	97.66	10.23	82.93	25.56	53.70	99.67	27.22	163.71
DGR [23]	89.88	49.16	68.35	107.84	69.35	108.52	24.40	176.75
iCaRL [9]	89.17	–	81.27	–	70.07	–	11.16	–
LwF [21]	81.70	–	64.99	–	66.58	–	28.74	–
IntroNet (ours)	94.77	–	85.15	–	72.36	–	26.92	–
IntroGAN (ours)	97.41	8.90	88.17	25.45	77.26	91.64	35.47	169.88

speaking, only the exclusive weights for each class is changeable), thus comparing with DGM is not very fair. Among other methods, IntroGAN has the highest TA-ACC and a relatively low TA-FID comparable to others, indicating its strong discriminative power which is endowed by prototype-based classification robust to feature update. Another noteworthy result is that iCaRL has lower TA-ACC than LwF on ImageNet-Dogs. The reason is probably that the representation and classifier in iCaRL are learned separately, making it less stable and sensitive to the choice of the feature extractor as also pointed out in [18].

The ACC and FID curves provide more detailed views of TA-ACC and TA-FID. Due to space limits, we only show the result curves on Fashion-MNIST and SVHN in the main text (refer to Sec. S6.1 for MNIST and ImageNet-Dogs). On Fashion-MNIST (Fig. 5) it can be observed that DGR is the most unstable one. From the generated images in Fig. 7, it can be seen that DGR fails to generate samples of certain classes (black images), which is very obvious on Fashion-MNIST. It is because DGR uses an unsupervised GAN, making it prone to generating easy samples and avoid difficult ones as pointed out by [63]. This imbalance becomes more severe when an old GAN replays what it has learned and teaches a new GAN, since this knowledge transfer will have information loss in practice. It can also be seen that IntroGAN constantly outperforms MeRGAN and the reason might be two-fold: first, as illustrated in Fig. 4 the prototype-based classification is robust to feature change because it averages the features of the exemplars, which can lower the risk of estimating the real class mean especially when certain exemplars deviate too much from the class centroid; second, the stored exemplars serve as extra training samples and can offer more authentic information, while the replayed samples by GANs are less authentic and may carry some noise in practice. Further analyses over these two methods are in Section 4.3.

On SVHN (Fig. 6), IntroGAN outperforms MeRGAN and it is again attributed to the superiority of prototype-based classification and the extra use of exemplars as training samples mentioned above. One noteworthy phenomenon is that in Fig. 6 that IntroGAN is not the highest for the first one or two tasks. The reason is probably that the classifier and the feature extractor are disentangled in IntroGAN, therefore it does not directly guarantee that a test sample can be correctly classified and a slightly lower accuracy occurs. However, its potential is unleashed when more classes are encountered, which can also be observed in the accuracy change of iCaRL.

4.3. Ablation studies

MeRGAN is the most similar to our work and the technical differences mainly lie in: (1) prototype classifier; (2) some old training data (i.e. old exemplars) added to the training set. To perform ablation studies of these two techniques we conduct the following experiments in Table 4 where IntroGAN* does not have (2) and its difference with MeRGAN mainly lies in whether to use the prototype classifier or not.

Table 4
The TA-ACC and TA-FID of IntroGAN and MeRGAN given different number of exemplars. Exem.=exemplars. Unlike other tables, here IntroGAN* is the version with prototype-based classifier but the exemplars are not added into the training set.

Method	MNIST		F-MNIST		SVHN	
	ACC	FID	ACC	FID	ACC	FID
IntroGAN*	**97.42**	9.39	88.06	25.92	75.45	100.59
+ 20 exem.	97.41	**8.90**	88.17	25.45	77.26	**91.64**
+ 100 exem.	97.40	10.32	**88.80**	**25.33**	**82.41**	107.79
MeRGAN	97.66	10.23	82.93	25.56	53.70	**99.67**
+ 20 exem.	97.97	10.04	85.81	**24.93**	61.63	102.01
+ 100 exem.	**98.14**	**8.67**	**87.92**	24.95	**73.51**	102.27

Comparing the 1st and the 4th rows in the result table, it can be concluded that prototype-based classifier boosts much of the accuracy but the effect over the generation quality is less certain (the FID is on par with that of MeRGAN). By examining the remaining rows, it can be seen that adding exemplars to the training set generally has a positive effect on the accuracy for both IntroGAN and MeRGAN (the more, the better), but its effect on the FID is less obvious. The reason is probably that the calculation of FID is based on the mean and covariance matrix in the feature space, and adding these small numbers of exemplars have little influence on changing the mean and covariance matrix.

4.4. Mutual benefits of generation and classification

Generation Avails Classification. For non-incremental scenarios, there are many cases including [64–66]. In incremental scenarios (Section 1), we mentioned that generation can offer memory replay and more generalization ability. To validate them, we first remove the GAN from the IntroGAN framework and name it IntroNet which only has the classification ability. IntroNet is the baseline in this experiment. From Table 3 and Fig. 5, 6 it can be observed that IntroGAN constantly outperforms the IntroNet. The reason might be that the GAN provides an extra generative memory about old classes, which offers more sample diversity and can improve the generalization ability of the classifier (i.e. less overfitting).

Classification Helps Generation. The superiority of AC-GAN [36] over conditional GAN [67] has already demonstrated that classification helps generation in ordinary image generation. Here, we also want to show that classification helps generation in incremental learning scenarios by better controlling the generation process. To demonstrate that, we dive into the behavior of DGR with that of IntroGAN since the training of the generator and classifier are independent in DGR (DGR is seen as the baseline in this experiment). As shown in Section 4.2, DGR fails to generate certain hard classes because its GAN has no class supervision. For further analysis, we let the final DGR model generate 10^5 samples and calculate the statistics of class samples based on DGR's classifier. The full statistics are in Sec. S6.3, and a summarization in pie charts is shown in Fig. 8, from which it can be seen that the generated

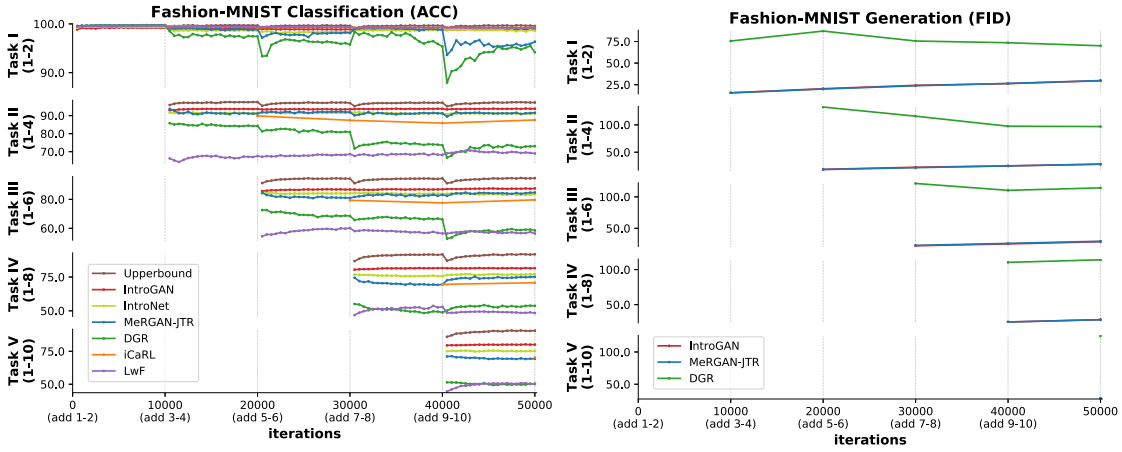

Fig. 5. The ACC and FID curves of different methods on Fashion-MNIST. Five sub-figures vertically indicate five different tasks (e.g. *Task II (1–4)* means that the 2nd task is to classify the first four classes), and the horizontal axis is the number of iterations. New classes add at a multiple of 10,000 iterations. Each point on the graph corresponds to an accuracy or FID. Note that for FID, the lower, the better. Refer to Section 4.2 for the reason to show in such a graph. Magnify for better view.

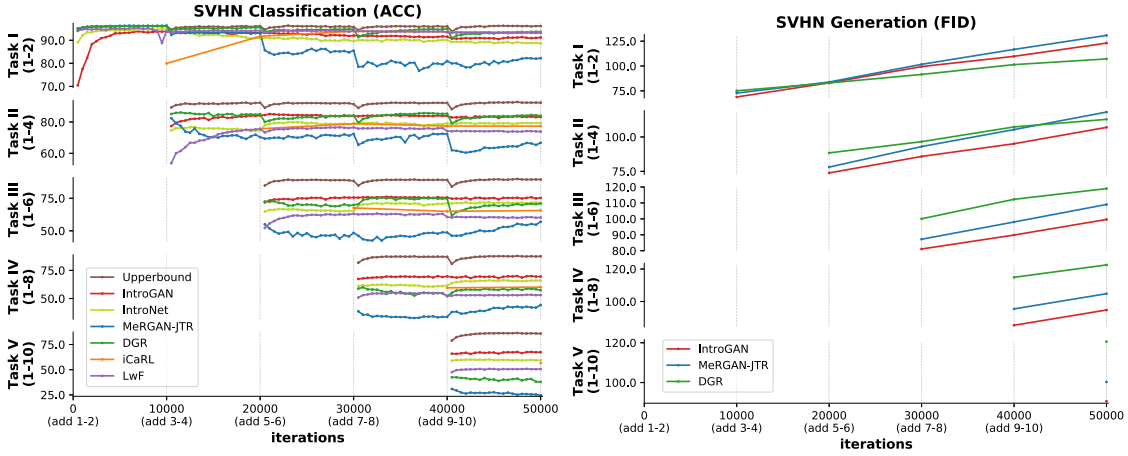

Fig. 6. The accuracy and FID curves of different methods on SVHN similar to Fig. 5.

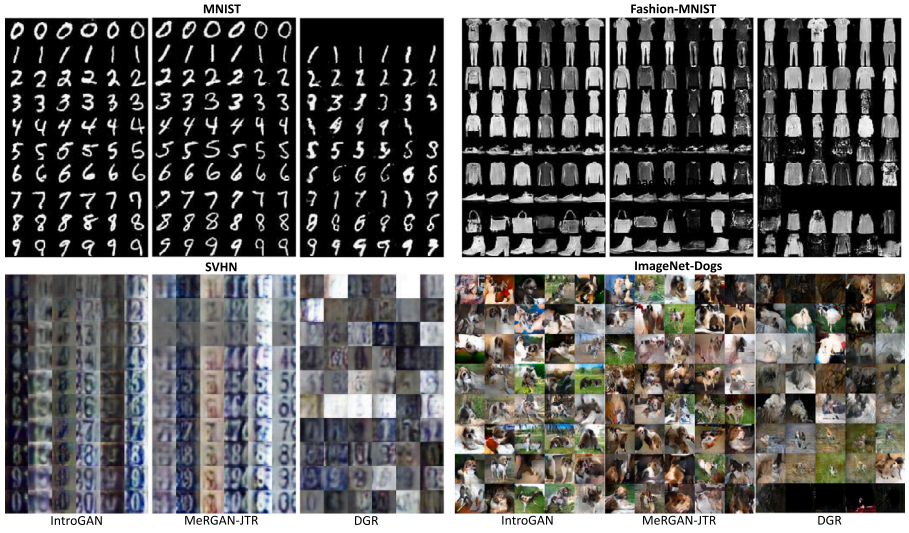

Fig. 7. Generated images of IntroGAN, MeRGAN and DGR in the final test on four datasets. For (Fashion-)MNIST and SVHN, each row is a different class and each column of IntroGAN/MeRGAN uses the same latent vector (i.e. z in Eq. (1)). For ImageNet-Dogs, we show samples of a certain class because different class samples with the are visually insignificant, indicating this dataset is difficult for image generation. Note that black images indicate failure in generation, which is explained in Section 4.4. The generated images over time are shown in Sec. S6.2.

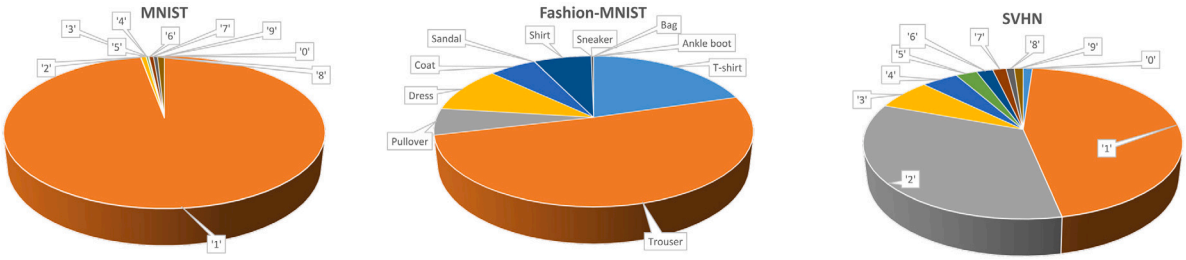

Fig. 8. The pie charts that show the portions of different class samples generated by DGR on different datasets. The annotation indicates the class name.

Table 5
The TA-ACC and TA-FID of IntroGAN and MeRGAN on different datasets when there are only 50 training samples for each category.

Method	MNIST		F-MNIST		SVHN	
	ACC	FID	ACC	FID	ACC	FID
IntroGAN (50 samples)	**97.31**	**9.13**	**87.93**	**25.56**	**43.28**	**218.05**
MeRGAN (50 samples)	91.97	23.96	82.90	26.43	29.83	245.16

Table 6
Accuracy of different methods on a 100-class subset of ImageNet over time.

Increment	IntroGAN	baseline	iCaRL	EEIL	LSIL	Joint Training
1	85.40	87.60	85.60	87.60	87.60	87.60
2	68.00	57.30	66.00	62.60	66.60	75.30
3	56.60	36.20	51.60	42.13	42.47	62.93
4	49.95	30.55	46.65	35.15	38.40	58.20
5	45.44	23.80	40.64	27.88	31.68	53.24

samples are highly imbalanced. Instead, IntroGAN can generate an arbitrary number of samples for each class and does not exhibit such a problem. Class imbalance is an important problem in incremental learning [68–70] and that may account for the unsatisfactory performance of DGR on benchmark datasets.

From a broader view, the problem with generation or classification alone mentioned above is also related to the universal shortcut learning in deep networks [71]: without the supervision of the classifier, the generator takes shortcut to only generate certain classes (as can be seen in DGR); without the generator, the classifier might focus on feature that can discriminate the current classes alone. The idea of combining generative and discriminative learning can let the two tasks regularize each other to alleviate shortcut learning in either side, which forms a nice symmetry. Moreover, the idea of combining generative and discriminative models is also related to dual learning [72], introspective learning [73], and hybrids of generative and discriminative models [74,75].

4.5. Further analyses and discussions

Low-Shot Learning Setting. IntroGAN also proves effective under low-shot scenarios. From Table 5, it can be observed that IntroGAN consistently outperforms MeRGAN even when there are only 50 training samples (in ordinary GAN training, 50 is a much smaller number and can be roughly seen as "low-shot"). The reasons are again attributed to the robustness of prototype-based classifier and the exemplars used as additional training samples.

Large-Scale Datasets. Although IntroGAN shows promising results in previous sections, one may still be concerned that those datasets are still much easier than ImageNet is often used in incremental classification. To demonstrate IntroGAN's potential on large-scale datasets, we extend it to generate features instead of images for efficiency concerns. Specifically, we extract an 100-class ImageNet subset, select the first 50 classes to train the feature extractor, and use the feature extractor to obtain features for the remaining 50 classes. Such a setting also mimics the scenario that the algorithm has already seen a bunch of classes, which is more similar to human learning that not everything is learned from scratch. After that, 10 classes are added at a time and there are 5 increments in total. More SOTAs in incremental classification like iCaRL [9], End-to-End Incremental Learning (EEIL) [76], Large-Scale Incremental Learning (LSIL) [68] are compared and the average ACCs are shown in Table 6.

Since these methods classify features instead of images, the results may be different from those reported in their original papers. Among them, IntroGAN and iCaRL generally perform better and it may be attributed to the prototype classifier that disentangles representation and classifier learning, making it more robust to the unusual "feature classification" setting here. Such a phenomenon is not so surprising since [77] also demonstrates that this kind of embedding network may be less prone to catastrophic forgetting than softmax classifier-based methods. The higher performance of IntroGAN over iCaRL's is probably attributed to the *selection functions* in the prototype-based classifier (Section 3.2) and the generalization ability provided by GANs.

Foreground Clarity and Background Diversity Over Time. Although IntroGAN shows very high TA-ACC, the TA-FID is only slightly better or on par with SOTAs. By observing the generated images of IntroGAN on SVHN over time, we notice that the foreground clarity decreases and the background diversity gets lower (number 6 and 2 respectively in Fig. 9). Since clarity and

diversity are important factors in judging the image quality, we suggest that a future improvement over IntroGAN might be disentangling foreground and background and generate them separately as in [78]. It may largely boost the TA-FID since the foreground and background are mutually irrelevant in most cases. By performing random combination over foregrounds and backgrounds, more diverse images can be generated.

5. Conclusion

In this paper, we propose a novel GAN framework called **Introspective GAN (IntroGAN)** which can perform incremental generation and classification simultaneously with the guidance of prototypes. We design a comprehensive benchmark and evaluation metrics for the joint task, perform experiments and show that our method achieves promising results. Moreover, we also verify from preliminary studies that incremental generation and classification can be mutually beneficial, which is a very inspiring area to be further exploited. Future works might be performing more in-depth analyses on when or why combining these two tasks works (direct justification of Fig. 2), and making more improvements over IntroGAN such as disentangling the foreground and background for better performance, reducing the memory overhead, making it expandable etc.

CRediT authorship contribution statement

Chen He: Conceptualization, Formal analysis, Methodology, Validation, Writing –original draft. **Ruiping Wang:** Conceptualization,

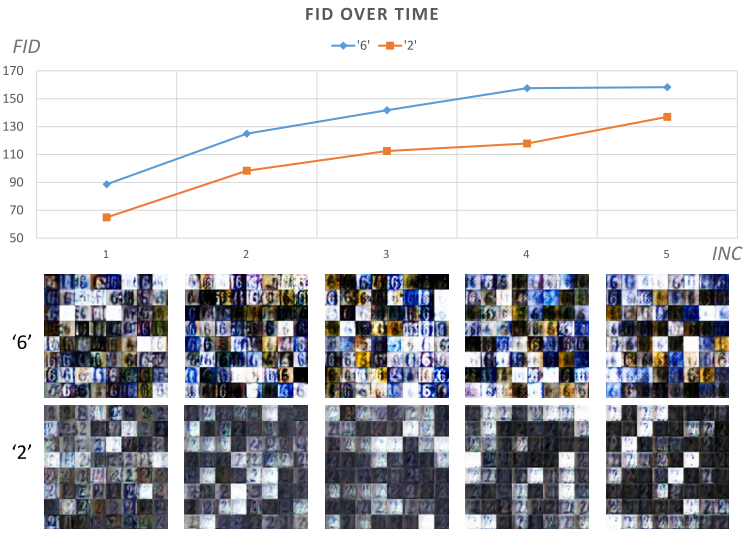

Fig. 9. Quantitative (in FIDs) and the corresponding qualitative results (generated images) of the generation performance of IntroGAN over time for class "6" and "2" on SVHN. *Top*: FID is increasing over time for "6" and "2". *Bottom*: foreground clarity and background diversity are decreasing judging from the generated images over time. "INC" indicates the index of the current class **inc**rement. Magnify for better view.

Formal analysis, Funding acquisition, Methodology, Supervision, Writing –review & editing. **Shiguang Shan:** Conceptualization, Formal analysis, Writing –review & editing. **Xilin Chen:** Conceptualization, Formal analysis, Funding acquisition, Supervision, Writing –review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the code on GitHub. The link is provided in the paper.

Acknowledgments

This work is partially supported by National Key R&D Program of China No. 2021ZD0111901, and Natural Science Foundation of China under contracts Nos. U21B2025, U19B2036.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patcog.2024.110383.

References

[1] Z. Chen, B. Liu, Lifelong machine learning, in: Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 10, (3) Morgan & Claypool Publishers, 2016, pp. 1–145.
[2] D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, in: NeurIPS, 2017, pp. 6467–6476.
[3] G. Zhou, K. Sohn, H. Lee, Online incremental feature learning with denoising autoencoders, in: AISTATS, 2012, pp. 1453–1461.
[4] A. Zeman, M. Dewar, S. Della Sala, Lives without imagery-congenital aphantasia., Cortex J. Devoted Study Nervous Syst. Behav. 73 (2015) 378.
[5] M.J. Farah, Visual Agnosia, MIT Press, 2004.
[6] T. Grüter, M. Grüter, V. Bell, C.-C. Carbon, Visual mental imagery in congenital prosopagnosia, Neurosci. Lett. 453 (3) (2009) 135–140.
[7] H. Jiang, R. Wang, S. Shan, X. Chen, Learning class prototypes via structure alignment for zero-shot recognition, in: ECCV, 2018, pp. 118–134.
[8] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: NeurIPS, 2017, pp. 4077–4087.
[9] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, iCaRL: Incremental classifier and representation learning, in: CVPR, 2017, pp. 2001–2010.
[10] H.-M. Yang, X.-Y. Zhang, F. Yin, C.-L. Liu, Robust classification with convolutional prototype learning, in: CVPR, 2018, pp. 3474–3482.
[11] M.B. Ring, Continual Learning in Reinforcement Environments (Ph.D. thesis), University of Texas at Austin Austin, Texas 78712, 1994.
[12] M.B. Ring, CHILD: A first step towards continual learning, Mach. Learn. 28 (1) (1997) 77–104.
[13] R.J. Solomonoff, A system for incremental learning based on algorithmic probability, in: Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition, 1989, pp. 515–527.
[14] A. Gepperth, B. Hammer, Incremental learning algorithms and applications, in: European Symposium on Artificial Neural Networks, ESANN, 2016.
[15] D. Maltoni, V. Lomonaco, Continuous learning in single-incremental-task scenarios, Neural Netw. 116 (2019) 56–73.
[16] M. McCloskey, N.J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of Learning and Motivation, vol. 24, Elsevier, 1989, pp. 109–165.
[17] S. Grossberg, Competitive learning: From interactive activation to adaptive resonance, Cogn. Sci. 11 (1) (1987) 23–63.
[18] C. He, R. Wang, S. Shan, X. Chen, Exemplar-supported generative reproduction for class incremental learning, in: BMVC, 2018, pp. 3–6.
[19] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, Q. Sun, Mnemonics training: Multi-class incremental learning without forgetting, in: CVPR, 2020, pp. 12245–12254.
[20] A. Robins, Catastrophic forgetting, rehearsal and pseudorehearsal, Connect. Sci. 7 (2) (1995) 123–146.
[21] Z. Li, D. Hoiem, Learning without forgetting, in: ECCV, 2016, pp. 614–629.
[22] P. Dhar, R.V. Singh, K.-C. Peng, Z. Wu, R. Chellappa, Learning without memorizing, in: CVPR, 2019, pp. 5138–5146.
[23] H. Shin, J.K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, in: NeurIPS, 2017, pp. 2994–3003.
[24] G.M. van de Ven, H.T. Siegelmann, A.S. Tolias, Brain-inspired replay for continual learning with artificial neural networks, Nature Commun. 11 (1) (2020) 1–14.
[25] K. Deja, P. Wawrzyński, D. Marczak, W. Masarczyk, T. Trzciński, Binplay: A binary latent autoencoder for generative replay continual learning, in: IJCNN, 2021, pp. 1–8.
[26] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, 2016, arXiv preprint arXiv:1606.04671.
[27] J. Lee, J. Yun, S. Hwang, E. Yang, Lifelong learning with dynamically expandable networks, in: ICLR, 2018.

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, in: Proceedings of the National Academy of Sciences (PNAS), vol. 114, (13) National Acad Sciences, 2017, pp. 3521–3526.
[29] E.H. Rosch, Natural categories, Cogn. Psychol. 4 (3) (1973) 328–350.
[30] A.R. Webb, Statistical Pattern Recognition, John Wiley & Sons, 2003.
[31] X. Ao, X.-Y. Zhang, C.-L. Liu, Cross-modal prototype learning for zero-shot handwritten character recognition, Pattern Recognit. 131 (2022) 108859.
[32] H. Huang, Z. Wu, W. Li, J. Huo, Y. Gao, Local descriptor-based multi-prototype network for few-shot learning, Pattern Recognit. 116 (2021) 107935.
[33] T. Mensink, J. Verbeek, F. Perronnin, G. Csurka, Distance-based image classification: Generalizing to new classes at near-zero cost, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2624–2637.
[34] O. Li, H. Liu, C. Chen, C. Rudin, Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions, 2017, arXiv preprint arXiv:1710.04806.
[35] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv preprint arXiv:1511.06434.
[36] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier gans, 2016, arXiv preprint arXiv:1610.09585.
[37] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: ICLR, 2018.
[38] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in: ICLR, 2018.
[39] Y. Dong, Y. Zhang, L. Ma, Z. Wang, J. Luo, Unsupervised text-to-image synthesis, Pattern Recognit. 110 (2021) 107573.
[40] S. Wu, H. Tang, X.-Y. Jing, J. Qian, N. Sebe, Y. Yan, Q. Zhang, Cross-view panorama image synthesis with progressive attention GANs, Pattern Recognit. 131 (2022) 108884.
[41] Y. Xian, T. Lorenz, B. Schiele, Z. Akata, Feature generating networks for zero-shot learning, in: CVPR, 2018, pp. 5542–5551.
[42] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: ICCV, 2017.
[43] W. Wang, A. Wang, A. Tamar, X. Chen, P. Abbeel, Safer classification by synthesis, 2017, arXiv preprint arXiv:1711.08534.
[44] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al., Memory replay GANs: Learning to generate new categories without forgetting, in: NeurIPS, 2018, pp. 5962–5972.
[45] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, M. Nabi, Learning to remember: A synaptic plasticity driven framework for continual learning, in: CVPR, 2019, pp. 11321–11329.
[46] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, G. Mori, Lifelong GAN: Continual learning for conditional image generation, in: ICCV, 2019, pp. 2759–2768.
[47] T. Lesort, H. Caselles-Dupré, M. Garcia-Ortiz, A. Stoian, D. Filliat, Generative models from the perspective of continual learning, in: IJCNN, 2019, pp. 1–8.
[48] Q. Lao, M. Mortazavi, M. Tahaei, F. Dutil, T. Fevens, M. Havaei, FoCL: Feature-oriented continual learning for generative models, Pattern Recognit. 120 (2021) 108127.
[49] F.-A. Croitoru, V. Hondru, R.T. Ionescu, M. Shah, Diffusion models in vision: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (01) (2023) 1–20.
[50] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: NeurIPS, 2014, pp. 2672–2680.
[51] J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: ICML, 2018, pp. 4555–4564.
[52] Y.-X. Wang, R. Girshick, M. Hebert, B. Hariharan, Low-shot learning from imaginary data, in: CVPR, 2018, pp. 7278–7286.
[53] R. Gao, W. Liu, DDGR: continual learning with deep diffusion-based generative replay, in: ICML, 2023, pp. 10744–10763.
[54] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[55] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[56] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.
[57] A. Khosla, N. Jayadevaprakash, B. Yao, F.-F. Li, Novel dataset for fine-grained image categorization: Stanford dogs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop on Fine-Grained Visual Categorization, Vol. 2, FGVC, 2011, p. 1.
[58] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: NeurIPS, 2017, pp. 6626–6637.
[59] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016, arXiv preprint arXiv:1603.04467.
[60] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[61] S. Hou, X. Pan, C.C. Loy, Z. Wang, D. Lin, Learning a unified classifier incrementally via rebalancing, in: CVPR, 2019, pp. 831–839.
[62] X. Hu, K. Tang, C. Miao, X.-S. Hua, H. Zhang, Distilling causal effect of data in class-incremental learning, in: CVPR, 2021, pp. 3957–3966.
[63] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, A. Torralba, Seeing what a GAN cannot generate, in: ICCV, 2019, pp. 4502–4511.
[64] Y. Wang, T. Lu, Y. Zhang, Z. Wang, J. Jiang, Z. Xiong, FaceFormer: aggregating global and local representation for face hallucination, IEEE Trans. Circuits Syst. Video Technol. 33 (2022) 2533–2545.
[65] Y. Wang, T. Lu, Y. Yao, Y. Zhang, Z. Xiong, Learning to hallucinate face in the dark, IEEE Trans. Multimed. 14 (2023) 1–13.
[66] Y. Li, Y. Wang, Z. Cui, Decoupled multimodal distilling for emotion recognition, in: CVPR, 2023, pp. 6631–6640.
[67] M. Mirza, S. Osindero, Conditional generative adversarial nets, 2014, arXiv preprint arXiv:1411.1784.
[68] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: CVPR, 2019, pp. 374–382.
[69] E. Belouadah, A. Popescu, Il2m: Class incremental learning with dual memory, in: ICCV, 2019, pp. 583–592.
[70] B. Zhao, X. Xiao, G. Gan, B. Zhang, S.-T. Xia, Maintaining discrimination and fairness in class incremental learning, in: CVPR, 2020, pp. 13208–13217.
[71] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, F.A. Wichmann, Shortcut learning in deep neural networks, Nat. Mach. Intell. 2 (11) (2020) 665–673.
[72] Y. Xia, T. Qin, W. Chen, J. Bian, N. Yu, T.-Y. Liu, Dual supervised learning, in: ICML, 2017, pp. 3789–3798.
[73] L. Jin, J. Lazarow, Z. Tu, Introspective classification with convolutional nets, in: NeurIPS, 2017, pp. 823–833.
[74] J.A. Lasserre, C.M. Bishop, T.P. Minka, Principled hybrids of generative and discriminative models, in: CVPR, 2006, pp. 87–94.
[75] Z. Tu, Learning generative models via discriminative approaches, in: CVPR, 2007, pp. 1–8.
[76] F.M. Castro, M.J. Marín-Jiménez, N. Guil, C. Schmid, K. Alahari, End-to-end incremental learning, in: ECCV, 2018, pp. 233–248.
[77] L. Yu, B. Twardowski, X. Liu, L. Herranz, K. Wang, Y. Cheng, S. Jui, J.v.d. Weijer, Semantic drift compensation for class-incremental learning, in: CVPR, 2020, pp. 6982–6991.
[78] K.K. Singh, U. Ojha, Y.J. Lee, FineGAN: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery, in: CVPR, 2019, pp. 6490–6499.

Chen He received the B.S. degree in computer science from the Sichuan University, Chengdu, China, in 2016, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2022. His research interests mainly include computer vision, pattern recognition, machine learning and, in particular, incremental learning, lifelong learning, and open world recognition.

Ruiping Wang (S'08-M'11) received the B.S. degree in applied mathematics from Beijing Jiaotong University, Beijing, China, in 2003, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, in 2010. He was a Post-Doctoral Researcher with the Department of Automation, Tsinghua University, Beijing, from 2010 to 2012. He also spent one year as a Research Associate with the Computer Vision Laboratory, Institute for Advanced Computer Studies, University of Maryland at College Park, College Park, from 2010 to 2011. In 2012, he joined the Faculty of the Institute of Computing Technology, Chinese Academy of Sciences, where he has been a Professor since 2017. His research interests include computer vision, pattern recognition, and machine learning.

Shiguang Shan (M'04-SM'15) received the M.S. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1999, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2004. In 2002, he joined ICT, CAS, where he has been a Professor since 2010. He is currently the Deputy Director of the Key Laboratory of Intelligent Information Processing, CAS. He has authored over 200 papers in refereed journals and proceedings in computer vision and pattern recognition. His research interests include computer vision, pattern recognition, and machine learning. He especially focuses on face recognition related research topics. He was a recipient

of the Chinas State Natural Science Award in 2015 and the Chinas State S&T Progress Award in 2005 for his research work. He is an Associate Editor of several journals, including the IEEE TRANSACTIONS ON IMAGE PROCESSING, the Computer Vision and Image Understanding, the Neurocomputing, and the Pattern Recognition Letters. He has served as the Area Chair for some international conferences, including ICCV11, ICPR12/14/20, ACCV12/16/18, FG13/18/20, ICASSP14, BTAS18, and CVPR19/20.

Xilin Chen (M'00-SM'09-F'16) is a professor with the Institute of Computing Technology, Chinese Academy of Sciences (CAS). He has authored one book and more than 300 papers in refereed journals and proceedings in the areas of computer vision, pattern recognition, image processing, and multimodal interfaces. He is currently an associate editor of the IEEE Transactions on Multimedia, and a Senior Editor of the Journal of Visual Communication and Image Representation, a leading editor of the Journal of Computer Science and Technology, and an associate editor-in-chief of the Chinese Journal of Computers, and Chinese Journal of Pattern Recognition and Artificial Intelligence. He served as an Organizing Committee member for many conferences, including general co-chair of FG13/FG18, program co-chair of ICMI 2010. He is/was an area chair of CVPR 2017/2019/2020, and ICCV 2019. He is a fellow of the IEEE, IAPR, and CCF.