Adaptive Feature Generation for Online Continual Learning from Imbalanced Data

Yingchun Jian[1], Jinfeng Yi[2], and Lijun Zhang[1(✉)]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{jianyc,zhanglj}@lamda.nju.edu.cn
[2] JD AI Research, Beijing, China

Abstract. Online continual learning (OCL) is the setting where deep neural network (DNN) incrementally learns new tasks with online data streams. The major problem in OCL is *catastrophic forgetting*, that DNN forgets the acquired knowledge on previous tasks quickly. Recently emerged studies tackle a more realistic problem that the data follows an imbalanced class distribution in OCL by storing particular exemplars. However, preserving exemplars causes memory burden and privacy issues. In this paper, we propose a non-exemplar based method—Adaptive Feature Generation (AdaFG) for OCL from imbalanced data, which tackles the class imbalance and catastrophic forgetting problems simultaneously. Specifically, we argue that one common reason for these problems is the decision boundaries of minority or old classes with few or no samples are affected by majority classes. Therefore, we first maintain a representative prototype for each class in the feature space, which dynamically changes with the streaming data to approximate the class mean feature. Then, we generate new features adaptively for old and minority classes based on their prototypes and train the DNN's classifier to adjust the decision boundaries. Experiments on three popular datasets demonstrate AdaFG's effectiveness in consolidating previous knowledge and addressing the class imbalance problem without preserving exemplars.

Keywords: Online continual learning · Imbalanced learning · Data augmentation

1 Introduction

In the last decade, Deep Neural Network (DNN) has achieved human-level or even better performance in many individual tasks [8,20,26]. When applying the DNN to practice, a typical paradigm is training the DNN sufficiently on a prepared dataset, then fixing the model parameters to deploy on various devices. However, the well-trained model can only tackle a specific task, lacking the capacity of continually learning from data when the environment changes, e.g., new

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13280, pp. 276–289, 2022.
https://doi.org/10.1007/978-3-031-05933-9_22

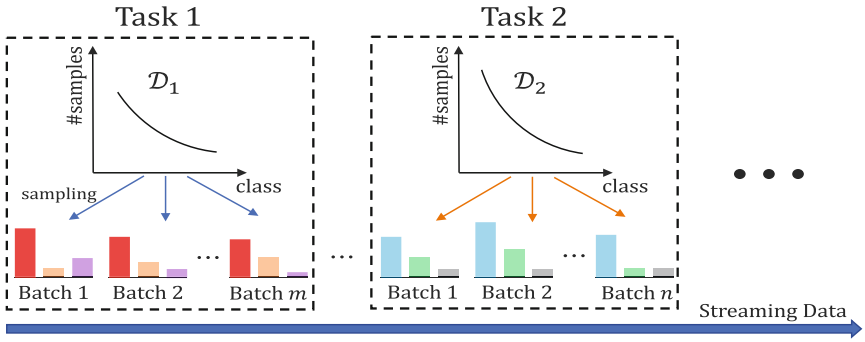


Fig. 1. Online continual learning from imbalanced data. Each batch is sampled from an imbalanced distribution.

classes not seen in the prepared dataset occur. Motivated by human's lifelong-learning ability [25], *continual learning* has been proposed to incrementally learn new tasks without access to previous data while maintaining the acquired knowledge on old tasks at the same time [15].

In this paper, we consider classification tasks and focus on the *online continual learning (OCL)* setting where the task data are coming in an online manner [18]. Different from the offline continual learning (Off-CL) [21] setting where the entire data of the new task are accessible and can be processed numerous times, we can only obtain a tiny batch of data at a time in OCL, which resembles the way humans learn more closely [3]. The major problem in OCL is *catastrophic forgetting* [19], i.e., DNN forgets previous knowledge quickly when learning a new task. Existing methods in OCL can be divided into two categories: exemplar based methods, which keep previous knowledge by storing observed samples (i.e., *exemplars*), and non-exemplar based methods that remember important parameters. However, most studies implicitly assume that the data follows a balanced distribution in each task [12,16,22,30], ignoring many realistic scenarios of imbalanced distributions, e.g., fraud detection and spam classification.

In this paper, we tackle the problem of online continual learning from imbalanced data (OCL-Imb) that each batch is sampled from an imbalanced distribution, as shown in Fig. 1. Besides catastrophic forgetting, we also need to solve the *class imbalance* problem in OCL-Imb, i.e., minority classes that have few samples in the new task are hard to learn [10]. Recently emerged studies address these problems in OCL-Imb by storing particular exemplars, e.g., Class-Balancing Reservoir Sampling [5] and Partitioning Reservoir Sampling [11]. However, these exemplar based methods bring memory burden for resource-constrained devices and cause privacy issues that arouse wide attention nowadays. Inspired by the non-exemplar based method protoAug [31] in Off-CL, we propose Adaptive Feature Generation (AdaFG) to address the class imbalance and catastrophic forgetting problems in OCL-Imb without preserving any exemplars. Specifically, we argue that one common reason of these problems is the decision boundaries for minority or old classes with few or no samples are affected by majority

classes that dominate the learning process. To solve this problem, we maintain a representative prototype in the feature space for each class, which dynamically changes with the streaming data to approximate the class mean feature on all seen samples. The prototype contains rich information and can be used to generate new features by adding gaussian noises [31]. AdaFG generates features for old and minority classes based on their prototypes, which are used to train the DNN's classifier along with the new coming data. The generated feature shows to be very helpful to alleviate the bias and adjust the decision boundaries. Moreover, since the distribution of observed samples dynamically changes, we adopt an adaptive strategy that controls the number of generated features according to the dynamic distribution to balance the learning process better.

To verify the effectiveness of the proposed AdaFG, we construct imbalanced tasks on three popular datasets (i.e., CIFAR-100 [13], Food-101 [2] and Mini-ImageNet [27]), and compare it with the state-of-the-art exemplar and non-exemplar based methods. Our empirical results show that AdaFG outperforms previous methods by large margins.

2 Related Works

In this section, we briefly review the related works of the OCL-Imb problem.

2.1 Online Continual Learning (OCL)

The existing OCL methods can be divided into two categories: exemplar based and non-exemplar based methods. Exemplar based methods use a memory buffer to store exemplars selected from previous tasks, which are retrieved to train the model along with the new coming data. Experience Replay [23] takes a naive approach that updates the memory with reservoir sampling and randomly retrieves the exemplars. Various memory updating and retrieving strategies are proposed to improve the performance, such as diversifying the gradients of the exemplars in the memory update [1] and leveraging Shapley Value adversarially in the memory retrieval [24]. As for the OCL-Imb problem, recent works design balanced schemes to make the memory updating and retrieving processes friendly to minority classes, e.g., Class-Balancing Reservoir Sampling [5] and Partitioning Reservoir Sampling [11]. Exemplar based methods try to maintain previously acquired knowledge by exploiting the information of exemplars as more as possible, but they may bring memory burden and cause privacy issues in many resource-restricted and data-sensitive applications.

Non-exemplar based methods usually use various regularization terms to consolidate the acquired knowledge on previous tasks. As representative methods, EWC [12] uses Fisher information matrix to estimate the importance of model parameters and penalizes the drastic changes of important parameters, and LwF [16] adopts knowledge distillation terms to prevent the feature drift of old classes. In addition to designing regularization terms, protoaug [31], a pioneering work in the field of Off-CL, maintains prototypes for old classes in

the feature space and generates prototype-based features to keep and expand the decision boundaries of old classes. The class-representative prototype shows to be very effective in keeping previous knowledge. Moreover, Lange and Tuytelaars [14] propose to use the prototype for nearest neighbor classification in OCL-Imb. However, this method needs to store exemplars to update the prototype with online data streams. In this paper, we extend the prototype-based method with no exemplars preserved to tackle the catastrophic forgetting problem in OCL-Imb.

2.2 Online Imbalanced Learning

Different from OCL that incrementally learns new tasks, online imbalanced learning focuses on learning a single task, and the streaming data is sampled from an imbalanced distribution. Due to the lack of samples, minority classes are highly under-represented and harder to learn compared with majority classes [7]. Many re-sampling strategies are proposed to solve this problem. For instance, Wang and Pineau [28] introduce online bagging techniques for online binary classification by randomly oversampling and undersampling samples of minority and majority classes, respectively. Furthermore, Wang *et al.* [29] extend online bagging techniques to tackle the multi-class imbalance problem. Besides re-sampling samples, data augmentation strategies are used to address the class imbalance problem. For instance, Generative Adversarial Networks [6] produce virtual samples by approximating the distribution of minority classes, and SMOTE [4] generates new samples for minority classes around the neighbor of original data. In this paper, we tackle the class imbalance problem in OCL-Imb also from the perspective of data augmentation.

3 Method

In this section, we first introduce the OCL-Imb setting and analyze the major problems in OCL-Imb, then illustrate the framework and details of our proposed method AdaFG.

3.1 Problem Analysis

Firstly, we consider the OCL setting. When learning the t-th task T_t, the model receives a tiny batch of samples of size b at a time, which is denoted as $\mathcal{B}_t^i$ for the i-th step. The entire data of T_t are $D_t = \{\mathcal{B}_t^0, \mathcal{B}_t^1, \cdots, \mathcal{B}_t^\tau\}$, where τ is the number of the totally received batches. When it comes to the OCL-Imb problem, T_t is not a balanced task and $\mathcal{B}_t^i$ are sampled from an imbalanced distribution $\mathcal{D}_t$, which is unknown in advance. As for the DNN model, we divide it into two parts: the feature extractor $\mathcal{G}$ and unified classifier $\mathcal{F}$. The goal is to minimize the statistical risk incurred by all seen tasks, which is formulated as:

$$\min \sum_{n=1}^{t} \mathbb{E}_{(x,y)\in D_n} \left[\ell\left(\mathcal{F}_t\left(\mathcal{G}_t\left(x;\phi_t\right);\theta_t\right),y\right)\right], \tag{1}$$

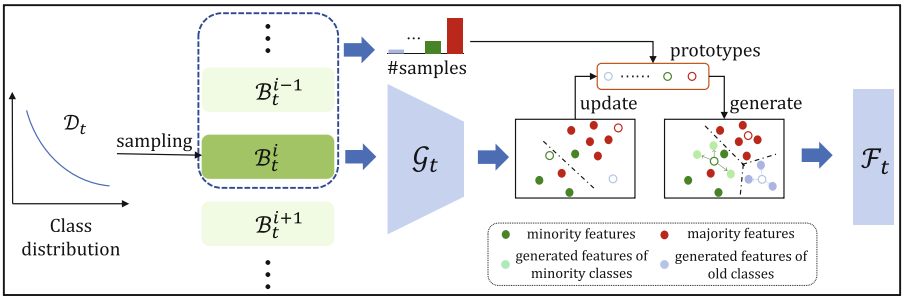


Fig. 2. The framework of Adaptive Feature Generation (AdaFG).

where ϕ_t and θ_t are the parameters of $\mathcal{G}_t$ and $\mathcal{F}_t$ after learning T_t, $\mathcal{G}_t(x;\phi_t)$ extracts the feature of a sample (x, y), $\mathcal{F}_t(\mathcal{G}_t(x;\phi_t);\theta_t)$ gets the outputs of the classifier, and ℓ is the loss function. Note that the data D_n of previous tasks $T_n (n < t)$ are not accessible, and we can only receive b samples of T_t at a time. The statistical risk of T_t can be approximated by the empirical risk [15]

$$\frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{(x,y)\in\mathcal{B}_t^i}\ell\left(\mathcal{F}_t\left(\mathcal{G}_t(x;\phi_t^i);\theta_t^i\right),y\right), \tag{2}$$

where ϕ_t^i and θ_t^i are parameters after the model trained on the i-th batch $\mathcal{B}_t^i$.

There are two major problems in OCL-Imb. One is catastrophic forgetting, which is caused by the drastic changes of decision boundaries for old classes when learning new classes in T_t [31]. Moreover, the learning process mainly focuses on majority classes, which contribute most to the changes of decision boundaries. The other is class imbalance that the decision boundaries are close to minority classes, because the optimization process of minimizing Eq. (2) is dominated by majority classes. The catastrophic forgetting and class imbalance problems are correlated, and one common reason of them is the decision boundaries for both old and minority classes are affected by the majority classes.

3.2 Adaptive Feature Generation (AdaFG)

In this subsection, we propose Adaptive Feature Generation (AdaFG) to adjust the decision boundaries by generating new features for old and minority classes, which are used to train the DNN's classifier.

Firstly, we handle the class imbalance problem by generating different numbers of features for classes in the current task T_t. For the k-th class in T_t (denoted as $c_{t,k}$), we maintain a prototype $\mu_{t,k}^i$, which is a feature approximating the class mean on all observed samples and dynamically changes with the incoming batch $\mathcal{B}_t^i$ (illustrated in Sect. 3.3). The new feature is generated based on $\mu_{t,k}^i$:

$$g_{t,k}^i = \mu_{t,k}^i + e \cdot r, \tag{3}$$

where e is a Gaussian noise with the same dimension as $\mu^i_{t,k}$, and r is a predefined value (e.g., 0.1). The generated feature can be seen as a certain disturbance of the prototype, which is used to train the classifier $\mathcal{F}_t$ to consolidate the just learned knowledge and expand the decision boundaries [31]. When receiving a new batch $\mathcal{B}^i_t$, the number of observed samples of $c_{t,k}$ is denoted as $s^i_{t,k}$. Since the distribution of $s^i_{t,k}$ $(k = 1, 2, \cdots, C_t$, C_t is the number of classes in $T_t)$ changes with incoming batches, we adopt an adaptive strategy to control the number of generated features for each class:

$$a^i_{t,k} = \left[N_1 \cdot \left(1 - \frac{s^i_{t,k}}{s^i_{max}} \right) \right], \tag{4}$$

where s^i_{max} is the maximum number of $s^i_{t,k}$ $(k = 1, 2, \cdots, C_t)$, N_1 is a predefined positive value (e.g., 10), and $[a]$ returns the integer closest to a. $a^i_{t,k}$ is close to N_1 if $s^i_{t,k} \ll s^i_{max}$ and close to 0 if $s^i_{t,k} \approx s^i_{max}$. In this way, minority classes have superiority in the number of generated features, making the decision boundaries far from minority classes, thus improving their performance.

Furthermore, we tackle the catastrophic forgetting problem by generating features for old classes in a similar way as Eq. (3) to keep previous decision boundaries. For the k-th class in the old task T_n $(n < t)$, the maintained prototype is $\mu_{n,t}$ and the new feature is generated as:

$$g_{n,k} = \mu_{n,k} + e \cdot r. \tag{5}$$

To generate features in a balanced way, we randomly select an old class to generate one feature by Eq. (5) and repeat N_2 times when receiving a new batch. To reduce the computation, N_2 is set to be a constant (e.g., 10) instead of a variable proportional to the number of seen classes.

The framework of AdaFG is illustrated in Fig. 2. When training on a new batch sampled from an imbalanced distribution, we update the number of observed samples for each class and the prototypes by current features. Then, we generate new features for old and minority classes in the ways mentioned above, which are used for training the classifier $\mathcal{F}_t$ to adjust the decision boundaries.

3.3 Online Prototype Update

For each class, the class mean in the feature space contains rich information and can be used for data augmentation [17]. Recent work [31] in Off-CL generates features for old classes to alleviate forgetting based on the maintained class mean features, which are computed until the feature extractor $\mathcal{G}_t$ is sufficiently trained on T_t. However, since only the current batch $\mathcal{B}^i_t$ are accessible and feature extractor parameters change over time in OCL-Imb, the class mean feature on all observed samples can't be computed directly. To overcome this problem, we adopt a *moving average* strategy to update the maintained prototype online

to approximate the true class mean feature. When receiving $\mathcal{B}_t^i$, the previously maintained prototype for $c_{t,k}$ is $\mu_{t,k}^{i-1}$ and will be updated by:

$$\begin{aligned} \mu_{t,k}^i &= (1-\alpha_{t,k}^i)\cdot\mu_{t,k}^{i-1} + \alpha_{t,k}^i\cdot\widetilde{\mu}_{t,k}^i, \\ \widetilde{\mu}_{t,k}^i &= \frac{1}{|X_{t,k}^i|}\sum_{x\in X_{t,k}^i}\mathcal{G}_t(x;\phi_t^{i-1}) \ , \end{aligned} \tag{6}$$

where $X_{t,k}^i$ are samples of $c_{t,k}$ in $\mathcal{B}_t^i$, $|X_{t,k}^i|$ is the size of $X_{t,k}^i$, and ϕ_t^{i-1} is the parameters of $\mathcal{G}_t$ after training on last batch. $\alpha_{t,k}^i$ is a factor controlling the prototype update, which can be adopted as:

$$\alpha_{t,k}^i = \frac{|X_{t,k}^i|}{\sum_{j=1}^i |X_{t,k}^j|}.$$

The prototype $\mu_{t,k}^i$ will not be updated if $|X_{t,k}^i|$ is 0. After learning the last batch $X_{t,k}^\tau$, the obtained prototype $\mu_{t,k}^\tau$ (also denoted as $\mu_{t,k}$) will be maintained for continually learning later tasks. In Sect. 4.3, we conduct experiments to demonstrate that the prototype updated by Eq. (6) is a good approximation to the true class mean on previously observed samples.

3.4 Training Process

When learning from the streaming data of a new task T_t, the training process can be divided into two parts. The first part is training on the new coming batch $\mathcal{B}_t^i$. For a new sample (x, y) in $\mathcal{B}_t^i$, the outputs of the current and last model are ξ_t and ξ_{t-1}, respectively. Typically, we adopt the cross-entropy loss $\mathcal{L}_{ce}(\xi_t, y)$ for classification and use the well-known knowledge distillation loss $\mathcal{L}_{kd}(\xi_t, \xi_{t-1})$ [16,22,30] to mitigate forgetting by making outputs of the current model close to those of the last model, which are defined as:

$$\begin{aligned} \mathcal{L}_{ce}(\xi_t, y) &= -\sum_{c=1}^{C} y_c \log(\sigma(\xi_t)_c), \\ \mathcal{L}_{kd}(\xi_t, \xi_{t-1}) &= -\sum_{c=1}^{C} \sigma(\xi_{t-1})_c \log(\sigma(\xi_t)_c), \end{aligned} \tag{7}$$

where C is the number of classes seen so far, $y \in \mathbb{R}^C$ is a label vector, and $\sigma(\cdot)$ is a softmax function. The overall loss of learning from the new data can be defined as previous works [22,30]:

$$\mathcal{L}_{new} = \frac{1}{t}\mathcal{L}_{ce}(\xi_t, y) + \left(1-\frac{1}{t}\right)\mathcal{L}_{kd}(\xi_t, \xi_{t-1}). \tag{8}$$

With the growth of tasks, the proportion of $\mathcal{L}_{kd}$ increases to remember more and more knowledge.

The second part is training the classifier on the generated features by AdaFG. For the generated data (g_o, y_o) and (g_m, y_m) of old and minority classes, we adopt $\mathcal{L}_{fgt}$ and $\mathcal{L}_{imb}$ to train the classifier $\mathcal{F}_t$:

$$\begin{aligned} \mathcal{L}_{fgt} &= \mathcal{L}_{ce}(\mathcal{F}_t(g_o; \theta_t^{i-1}), y_o), \\ \mathcal{L}_{imb} &= \mathcal{L}_{ce}(\mathcal{F}_t(g_m; \theta_t^{i-1}), y_m) + \mathcal{L}_{kd}(\mathcal{F}_t(g_m; \theta_t^{i}), \mathcal{F}_{t-1}(g_m; \theta_{t-1})). \end{aligned} \tag{9}$$

$\mathcal{L}_{fgt}$ and $\mathcal{L}_{imb}$ focus on mitigating forgetting and learning minority classes, respectively. Notice that in the first batch of each new task, the prototypes of new classes are not available, and only $\mathcal{L}_{fgt}$ is calculated.

The total loss is comprised of the above terms:

$$\mathcal{L} = \mathcal{L}_{new} + \eta \mathcal{L}_{fgt} + \gamma \mathcal{L}_{imb}, \tag{10}$$

where η and γ are coefficients that control the impact of corresponding terms.

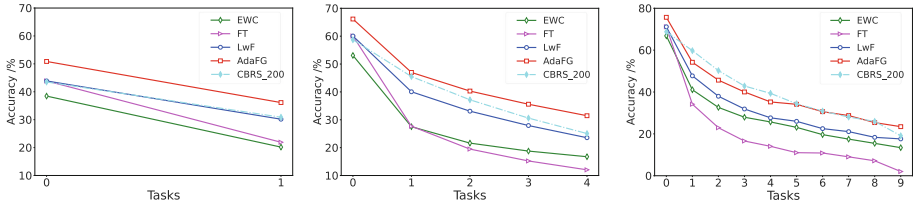
4 Experiments

In this section, we compare AdaFG with several state-of-the-art methods and analyze the results to validate our approach. Furthermore, we visualize the generated features to verify the effectiveness of the maintained prototype in AdaFG.

4.1 Setup

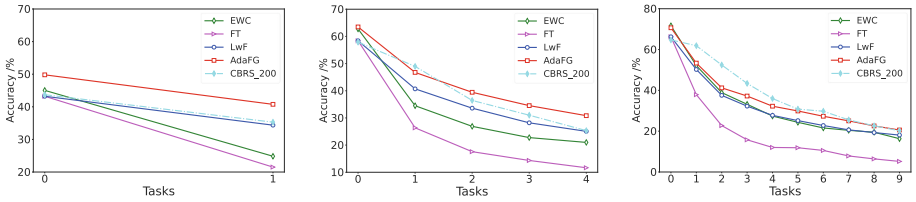
Datasets. CIFAR-100, Food-101 and Mini-ImageNet are used in our experiments, which are both balanced datasets. CIFAR-100 contains 50k training images and 10k test images in 100 classes. Food-101 contains 75k training images and 25k test images in 100 classes. Mini-ImageNet contains 60k images in 100 classes, and we split them into 50k training images and 10k test images.

OCL-Imb Settings. Similar to previous works [9,31], we divide the whole classes into two parts: base classes (20 classes for CIFAR-100 and Mini-ImageNet, and 21 classes for Food-101) and rest classes (80 classes). The base classes are used to train a base feature extractor offline, which is beneficial for the DNN model to cope with the streaming data. The rest classes are divided



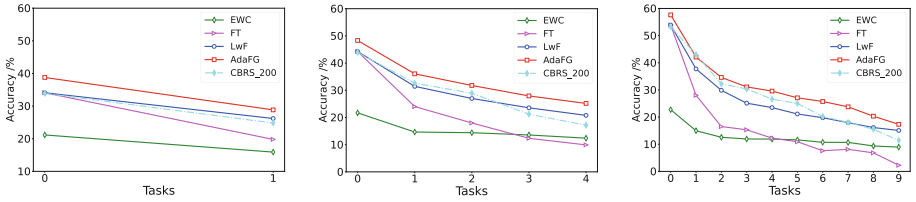
(a) $N = 2$ (b) $N = 5$ (c) $N = 10$

Fig. 3. Accuracy for each incremental task on CIFAR-100 when $N = 2$, 5, and 10.



(a) $N = 2$ (b) $N = 5$ (c) $N = 10$

Fig. 4. Accuracy for each incremental task on Food-101 when $N = 2$, 5, and 10.



(a) $N = 2$ (b) $N = 5$ (c) $N = 10$

Fig. 5. Accuracy for each incremental task on Mini-ImageNet when $N = 2$, 5, and 10.

into N tasks, and N can be 2, 5, or 10, which means each task contains 40, 16, and 8 classes, respectively. Following Chrysakis and Moens [5], we select a random percentage p of instances in the original dataset for each rest class to construct imbalanced streams. p is randomly selected from a retention set $\{1, 10^{-r}, 10^{-2r}, 10^{-3r}, 10^{-4r}\}$. In this paper, we use $r = 0.25$ for all experiments, i.e., the maximum imbalance between two classes is 10. When learning the sequential tasks, the number of samples received at a time is set to 10 (i.e., $b = 10$), and each sample can only be processed once.

Evaluation. After learning a new task, the performance is evaluated on test images of the observed rest classes by computing the average accuracy. We use two popular criteria to measure the ability to incrementally learn new tasks [22,30]. One is the *last accuracy*, which is the performance after learning the last task. The other is the *average incremental accuracy*, which computes the mean value of the performance over all incremental tasks. For each task division N, we construct 15 different imbalanced streams by setting 15 random seeds and report the average result. Additionally, we show the results of the accuracy for each incremental task in Fig. 3, 4 and 5.

Experimental Details. We use ResNet-18 [8] for all experiments. To train a base feature extractor, we use the SGD optimizer with the batch size of 32, and the initial learning rate is 0.1. For CIFAR-100, the learning rate is divided by 10 after 30, 60, and 90 epochs (100 epochs in total). For Food-101 and Mini-ImageNet, it is divided by 10 after 100, 150, and 180 epochs (200 epochs in total). When learning the sequential N tasks online, we adopt the SGD optimizer

with the learning rate of 0.1. The hyper-parameters used in AdaFG are set to: $N_1 = N_2 = 10$, $\eta = \gamma = 1.0$, and $r = 0.1$.

Table 1. Last accuracy (Last) and average incremental accuracy (Aver) on CIFAR-100.

CIFAR-100	$N = 2$		$N = 5$		$N = 10$	
	Last	Aver	Last	Aver	Last	Aver
FT	21.9 ± 1.2	32.9 ± 1.4	12.0 ± 0.8	26.9 ± 0.7	2.0 ± 3.3	19.9 ± 0.4
LwF	30.2 ± 1.8	37.0 ± 2.0	23.6 ± 1.3	37.0 ± 1.6	17.6 ± 1.1	32.2 ± 1.4
EWC	20.2 ± 1.0	29.3 ± 1.6	16.7 ± 1.4	27.6 ± 1.5	13.4 ± 2.0	28.3 ± 1.7
AdaFG	**36.1** ± 1.1	**43.5** ± 1.4	**31.5** ± 1.2	**44.1** ± 0.8	**23.4** ± 1.8	**39.3** ± 1.0
CBRS-50	25.8 ± 0.9	34.7 ± 1.4	17.3 ± 0.7	32.5 ± 0.9	9.8 ± 3.6	30.3 ± 1.2
CBRS-100	28.5 ± 1.0	36.0 ± 1.5	20.8 ± 0.9	36.0 ± 0.8	13.6 ± 3.5	35.3 ± 0.8
CBRS-200	30.9 ± 1.0	37.2 ± 1.5	25.1 ± 0.6	39.4 ± 0.6	19.1 ± 3.1	**39.9** ± 1.0

Table 2. Last accuracy (Last) and average incremental accuracy (Aver) on Food-101.

Food-101	$N = 2$		$N = 5$		$N = 10$	
	Last	Aver	Last	Aver	Last	Aver
FT	21.5 ± 2.0	32.4 ± 2.5	11.7 ± 0.9	25.7 ± 1.0	5.2 ± 1.7	19.7 ± 0.7
LwF	34.4 ± 2.5	38.8 ± 2.9	25.1 ± 1.3	37.2 ± 1.6	18.1 ± 1.2	32.0 ± 2.0
EWC	24.8 ± 1.9	34.9 ± 2.0	21.0 ± 1.9	33.6 ± 1.5	16.3 ± 1.5	32.5 ± 1.9
AdaFG	**40.8** ± 1.6	**45.3** ± 2.4	**30.8** ± 1.9	**43.0** ± 2.0	**20.6** ± 1.8	**36.0** ± 1.9
CBRS-50	29.3 ± 1.2	36.3 ± 2.0	16.7 ± 1.1	32.2 ± 0.9	11.4 ± 0.7	28.3 ± 0.9
CBRS-100	32.0 ± 1.4	37.9 ± 2.2	20.0 ± 1.4	35.4 ± 1.2	15.2 ± 1.3	32.6 ± 0.9
CBRS-200	35.3 ± 1.8	39.5 ± 2.5	25.3 ± 1.3	39.9 ± 0.9	20.2 ± 1.1	**38.7** ± 0.7

Compared Methods. We compare our proposed method AdaFG with several state-of-the-art non-exemplar and exemplar based methods:

- **FT:** (non-exemplar) A naive but important method that fine-tunes the model on the receiving data without any approach for avoiding forgetting.
- **LwF** [16]**:** (non-exemplar) Learning without Forgetting trains the model with the classification loss $\mathcal{L}_{ce}$ and knowledge distillation loss $\mathcal{L}_{kd}$.
- **EWC** [12]**:** (non-exemplar) Elastic Weight Consolidation uses a regularization term to constrain the updates of important parameters.
- **CBRS** [5]**:** (exemplar) Class-Balanced Reservoir Sampling uses a memory buffer to solve the OCL-Imb problem by storing particular samples. The buffer size is set to 50, 100 and 200 in our experiments.

4.2 Results

The results are reported in Tables 1, 2 and 3. For different task divisions ($N = 2, 5$ or 10), FT gets poor results, e.g., only 2.0% last accuracy on CIFAR-100 when $N = 10$. Since the importance of model parameters is hard to estimate,

Table 3. Last accuracy (Last) and average incremental accuracy (Aver) on Mini-ImageNet.

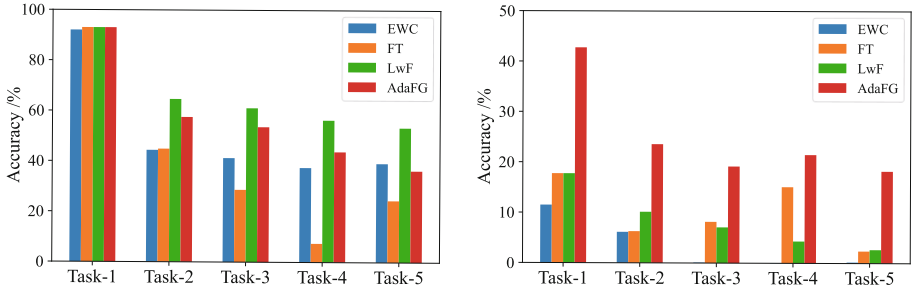
Mini-ImageNet	$N = 2$		$N = 5$		$N = 10$	
	Last	Aver	Last	Aver	Last	Aver
FT	19.8 ± 0.9	26.9 ± 1.4	9.9 ± 0.8	21.7 ± 0.7	2.3 ± 2.7	16.2 ± 0.8
LwF	26.2 ± 1.3	30.1 ± 1.8	20.8 ± 1.5	29.4 ± 1.5	15.1 ± 1.3	26.0 ± 1.6
EWC	15.9 ± 1.0	18.5 ± 1.4	12.4 ± 1.2	15.3 ± 2.0	8.9 ± 1.4	12.5 ± 2.1
AdaFG	**28.8** ± 1.2	**33.8** ± 1.4	**25.2** ± 1.2	**33.8** ± 1.2	**17.3** ± 2.3	**30.9** ± 1.1
CBRS-50	22.3 ± 0.6	28.0 ± 1.4	12.3 ± 0.5	24.4 ± 0.6	6.5 ± 2.9	21.0 ± 0.6
CBRS-100	23.3 ± 0.8	28.6 ± 1.5	14.3 ± 1.0	26.5 ± 0.8	8.7 ± 2.9	23.8 ± 0.9
CBRS-200	24.8 ± 0.7	29.4 ± 1.6	17.2 ± 0.9	28.8 ± 0.9	11.4 ± 2.3	27.6 ± 0.7

the results of EWC are not good compared with the distillation-based method LwF. In contrast, AdaFG achieves superior performance over all compared non-exemplar based methods whether on CIFAR-100, Food-101 or Mini-ImageNet. This is because these methods don't consider the class imbalance problem when designing the algorithms, thus shows unsatisfactory performance in the OCL-Imb setting. On Mini-ImageNet, AdaFG outperforms LwF by 3.7%, 4.4%, and 4.9% on the average incremental accuracy when $N = 2$, 5, and 10, respectively. On Food-101, the gaps are 6.5%, 6.2%, and 4.0%. As for CIFAR-100, the gaps are increased to 6.5%, 7.1%, and 7.1%. The performance of AdaFG on CIFAR-100 is much better than that on Mini-ImageNet, because the data in Mini-ImageNet are more complex (84×84 pixels v.s. 32×32 pixels) and the representation learning of the DNN model becomes more difficult.

As for the exemplar-based method, CBRS heavily depends on the memory size, and the performance gain of increasing the size is quite obvious. For instance, CBRS achieves 9.3% improvements on the last accuracy on CIFAR-100 ($N = 10$) when enlarging the memory size from 50 to 200. Compared with CBRS, AdaFG achieves comparable or even better results even if the memory size is 200. Specifically, AdaFG outperforms CBRS by large margins on Mini-ImageNet, e.g., 8.0% improvements on the last accuracy when $N = 5$, demonstrating the effectiveness of AdaFG to cope with complex data.

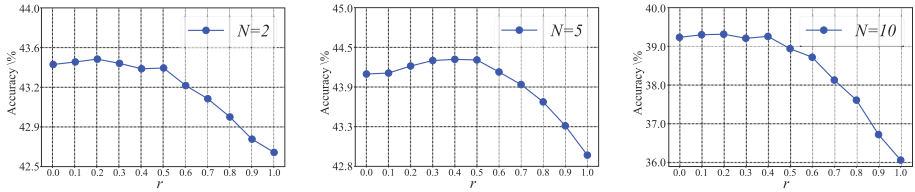
Performance of the Largest and Smallest Classes. After learning a new task, we compute the average accuracy of the largest and smallest observed classes (i.e., $p = 1.0$ and 0.1). As shown in Fig. 6, compared with other methods, AdaFG achieves comparable results on the largest classes but surpasses others greatly on the smallest classes, which shows that AdaFG can improve the learning of minority classes effectively at a slight expense of performance degradation on majority classes.

Ablation Study. We analyze the impact of hyper-parameter r that controls the scope of the generated features in AdaFG, and the results of the average incremental accuracy on CIFAR-100 are shown in Fig. 7. When $r > 0.4$, the accuracy drops rapidly with the increasing value of r. In this paper, we set $r = 0.1$ as it obtains good performance when learning a long sequence of tasks ($N = 10$).



(a) Largest observed classes ($p = 1.0$) (b) Smallest observed classes ($p = 0.1$)

Fig. 6. Performance of specific classes when continually learning 5 tasks on CIFAR-100 ($N = 5$). (a) The mean accuracy of the observed largest classes ($p = 1.0$). (b) The mean accuracy of the observed smallest classes ($p = 0.1$).



(a) $N = 2$ (b) $N = 5$ (c) $N = 10$

Fig. 7. Average incremental accuracy on CIFAR-100 w.r.t. r when $N = 2$, 5, and 10.

4.3 Visualization of Generated Features

To verify the quality of generated features by AdaFG, we visualize the generated features based on the prototypes and the real features of classes with different percentages p by collecting all seen samples. As shown in Fig. 8, the generated features are in the core of real features and can almost cover the real features of minority classes (e.g., $p = 0.10$ and 0.18), which demonstrates the prototype computed by Eq. (6) is close to the true class mean feature.

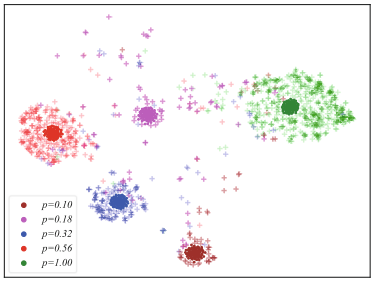


Fig. 8. Real (light color) and generated (dark color) features (Color figure online)

5 Conclusion

In this paper, we tackle the realistic problem of online continual learning from imbalanced data (OCL-Imb) and analyze the catastrophic forgetting and class imbalance problems encountered in OCL-Imb. To address these problems, we

propose a simple yet effective method AdaFG that gets rid of storing exemplars. AdaFG maintains a representative prototype for each class and generates new features based on the prototype to mitigate forgetting and improve the performance of minority classes. Experiments on CIFAR-100, Food-101 and Mini-ImageNet show that AdaFG achieves better performance than the state-of-the-art methods.

Acknowledgements. This work was partially supported by JiangsuSF (BK20200064), and the Open Research Projects of Zhejiang Lab (NO. 2021KB0AB02).

References

1. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. In: NeurIPS, pp. 11816–11825 (2019)
2. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 446–461. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10599-4_29
3. Cangelosi, A., Schlesinger, M.: Developmental Robotics: From Babies to Robots. MIT Press, Cambridge (2015)
4. Chawla, N.-V., Bowyer, K.-W., Hall, L.-O., Kegelmeyer, W.-P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**(1), 321–357 (2002)
5. Chrysakis, A., Moens, M.-F.: Online Continual Learning from Imbalanced Data. In: ICML, pp. 1952–1961 (2020)
6. Douzas, G., Bacao, F.: Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Syst. Appl. **91**, 464–471 (2018)
7. He, H., Garcia, E.-A.: Learning from imbalanced data. TKDE **9**(21), 1263–1284 (2009)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
9. Hou, S., Pan, X., Loy, C.-C., Wang, Z., Lin, D.: Learning a Unified Classifier Incrementally via Rebalancing. In: CVPR, pp. 831–839 (2019)
10. Johnson, J., Khoshgoftaar, T.: Survey on deep learning with class imbalance. J. Big Data **6**(1), 1–54 (2019)
11. Kim, C.D., Jeong, J., Kim, G.: Imbalanced continual learning with partitioning reservoir sampling. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12358, pp. 411–428. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58601-0_25
12. Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. PNAS **114**(13), 3521–3526 (2017)
13. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report (2009)
14. Lange, M., Tuytelaars, T.: Continual prototype evolution: learning online from non-stationary data streams. In: ICCV, pp. 8250–8259 (2021)
15. Lange, M., et al.: A continual learning survey: defying forgetting in classification tasks. arXiv preprint arXiv:1909.08383 (2019)
16. Li, Z., Hoiem, D.: Learning without forgetting. In: ECCV, pp. 614–629 (2016)

17. Liu, J., Sun, Y., Han, C., Dou, Z., Li, W.: Deep representation learning on long-tailed data: a learnable embedding augmentation perspective. In: CVPR, pp. 2970–2979 (2020)
18. Mai, Z., et al.: Online continual learning in image classification: an empirical survey. arXiv preprint arXiv:2101.10423 (2021)
19. McCloskey, M., Cohen, J.-N.: Catastrophic interference in connectionist networks: the sequential learning problem. Psychol. Learn. Motiv. **24**, 109–165 (1989)
20. Mnih, V., et al.: Playing atari with deep reinforcement learning. In: NeurIPS Workshop (2013)
21. Prabhu, A., Torr, P.H.S., Dokania, P.K.: GDumb: a simple approach that questions our progress in continual learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12347, pp. 524–540. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58536-5_31
22. Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.: iCaRL: incremental classifier and representation learning. In: ICCV, pp. 5533–5542 (2017)
23. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience replay for continual learning. In: NeurIPS, pp. 350–360 (2019)
24. Shim, D., Mai, Z., Jeong, J., Sanner, S., Kim, H., Jang, J.: Online class-incremental continual learning with adversarial shapley value. In: AAAI, pp. 9630–9638 (2021)
25. Tani, J.: Exploring Robotic Minds: Actions, Symbols, and Consciousness as Self-Organizing Dynamic Phenomena. Oxford University Press, Oxford (2016)
26. Vaswani, A., et al.: Attention is all you need. In: NeurIPS, pp. 6000–6010 (2017)
27. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NeurIPS, pp. 3637–3645 (2016)
28. Wang, B., Pineau, J.: Online bagging and boosting for imbalanced data streams. TKDE **28**(12), 3353–3366 (2016)
29. Wang, S., Minku, L.-L., Yao, X.: Dealing with multiple classes in online class imbalance learning. In: IJCAI, pp. 2118–2124 (2016)
30. Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S.-T.: Maintaining discrimination and fairness in class incremental learning. In: ICCV, pp. 13205–13214 (2020)
31. Zhu, F., Zhang, X.-Y., Wang, C., Yin, F., Liu, C.-L.: Prototype augmentation and self-supervision for incremental learning. In: CVPR, pp. 5871–5880 (2021)