FOSTER: Feature Boosting and Compression for Class-Incremental Learning

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye✉, and De-Chuan Zhan

State Key Laboratory for Novel Software Technology, Nanjing University
wangfuyun@smail.nju.edu.cn,{zhoudw, yehj, zhandc}@lamda.nju.edu.cn

Abstract. The ability to learn new concepts continually is necessary in this ever-changing world. However, deep neural networks suffer from catastrophic forgetting when learning new categories. Many works have been proposed to alleviate this phenomenon, whereas most of them either fall into the stability-plasticity dilemma or take too much computation or storage overhead. Inspired by the gradient boosting algorithm to gradually fit the residuals between the target model and the previous ensemble model, we propose a novel two-stage learning paradigm FOSTER, empowering the model to learn new categories adaptively. Specifically, we first dynamically expand new modules to fit the residuals between the target and the output of the original model. Next, we remove redundant parameters and feature dimensions through an effective distillation strategy to maintain the single backbone model. We validate our method FOSTER on CIFAR-100 and ImageNet-100/1000 under different settings. Experimental results show that our method achieves state-of-the-art performance. Code is available at https://github.com/G-U-N/ECCV22-FOSTER.

Keywords: class-incremental learning, gradient boosting

1 Introduction

The real world is constantly changing, with new concepts and categories continuously springing up [14,48,35,46]. Retraining a model every time new classes emerge is impractical due to data privacy [5] and expensive training costs. Therefore, it is necessary to enable the model to continuously learn new categories, namely class-incremental learning [44,49,34]. However, directly fine-tuning the original neural networks on new data causes a severe problem known as catastrophic forgetting [11] that the model entirely and abruptly forgets previously learned information. Inspired by this, class-incremental learning aims to design a learning paradigm that enables the model to continuously learn novel categories in multiple stages while maintaining the discrimination ability for old classes.

In recent years, many approaches have been proposed from different aspects. So far, the most widely recognized and utilized class-incremental learning strategy is based on knowledge distillation [19]. Methods [27,32,1,38,43,47] retain an old model additionally and use knowledge distillation to constrain output

for original tasks of the new model to be similar to that of the old one [27]. However, these methods with a single backbone may not have enough plasticity [17] to cope with the coming new categories. Besides, even with restrictions of KD, the model still suffer from feature degradation [40] of old concepts due to limited access [5] to old data. Recently, methods [40,28,9] based on dynamic architectures achieve state-of-the-art performance in class-incremental learning. Typically, they preserve some modules with their parameters frozen to maintain important sections for old categories and expand new trainable modules to strengthen plasticity for learning new categories. Nevertheless, they have two inevitable defects: First, constantly expanding new modules for coming tasks will lead to a drastic increase in the number of parameters, resulting in severe storage and computation overhead, which makes these methods not suitable for long-term incremental learning. Second, since old modules have never seen new concepts, directly retaining them may harm performance in new categories. The more old modules kept, the more remarkable the negative impact.

In this paper, we propose a novel perspective from gradient boosting to analyze and achieve the goal of class-incremental learning. Gradient boosting methods use the additive model to gradually converge the ground-truth target model where the subsequent one fits the residuals between the target and the prior one. In class-incremental learning, since distributions of new categories are constantly coming, the distribution drift will also lead to the residuals between the target label and model output. Therefore, we propose a similar boosting framework to solve the problem of class-incremental learning by applying an additive model, gradually fitting residuals, where different models mainly handle their special tasks (with nonoverlapping sets of classes). And as we discuss later, our boosting framework is a more generalized framework for dynamic structure methods (*e.g.*, DER[40]). It has positive significance in two aspects: On the one hand, the new model enhances the plasticity and thus helps the model learn to distinguish between new classes. On the other hand, training the new model to classify all categories might contribute to discovering some critical elements ignored by the original model. As shown in Fig. 1, when the model learns old categories, including tigers, cats, and monkeys, it may think that stripes are essential information but mistakenly regard auricles as meaningless features. When learning new categories, because the fish and birds do not have auricles, the new model will discover this mistake and correct it.

However, as we discussed above, creating new models not only leads to an increase in the number of parameters but also might cause inconsistency between the old and the new model at the feature level. To this end, we compress the boosting model to remove unnecessary parameters and inconsistent features, thus avoiding the above-mentioned drawbacks of dynamic structure-based methods, preserving crucial information, and enhancing the robustness of the model.

In conclusion, our paradigm can be decoupled into two steps: boosting and compression. The first step can be seen as boosting to alleviate the performance decline due to the arrival of new classes. Specifically, we retain the old model with all its parameters frozen. Then we expand a trainable new feature extractor and

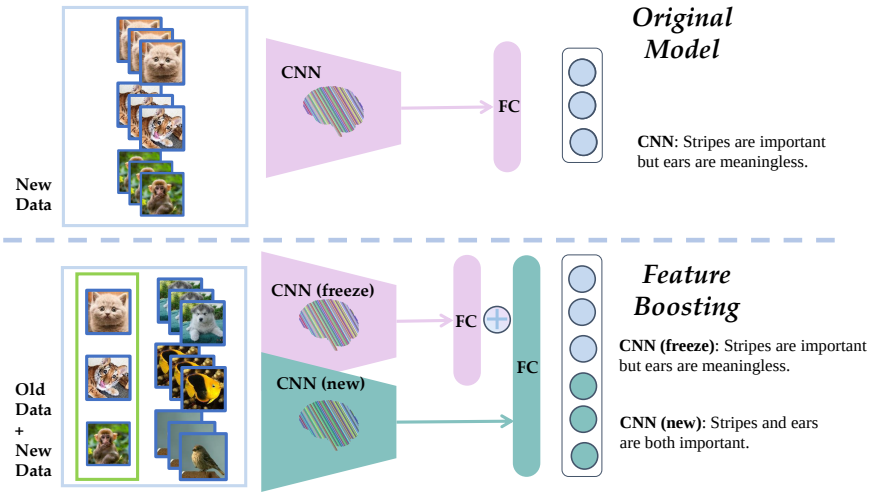


Fig. 1: **Feature Boosting.** Illustration of feature boosting. When the task comes, we freeze the old model and create a new module to fit the residuals between the target and the output. The new module helps the model learn both new and old classes better.

concatenate it with the extractor of the old model and initialize a constrained, fully-connected layer to transform the super feature into logits, which we will demonstrate later in detail. In the second step, we aim to eliminate redundant parameters and meaningless dimensions caused by feature boosting. Specifically, we propose an effective distillation strategy that can transfer knowledge from the boosting model to a single model with negligible performance loss, even if the data is limited when learning new tasks. Extensive experiments on three benchmarks, including CIFAR-100, ImageNet-100/1000 show that our method **F**eature Bo**OST**ing and Compr**E**ssion for class-inc**R**emental learning (**FOSTER**) obtains the state-of-the-art performance.

2 Related Work

Many works have been done to analyze the reasons for performance degradation in class-incremental learning and alleviate this phenomenon. In this section, we will give a brief discussion of these methods and boosting algorithms.

Knowledge Distillation. Knowledge distillation [19] aims to transfer dark knowledge [24] from the teacher to the student by encouraging the outputs of the student model to approximate the outputs of the teacher model [27]. LwF [27] retains an old model additionally and applies a modified cross-entropy loss to constrain the outputs for old categories of the new model to preserve the capability for the old one. Bic [38], WA [43] propose effective strategies to alleviate the bias of the classifier caused by imbalanced training data after distillation.

Rehearsal. The rehearsal strategy enables the model to have partial access to old data. [32,38,43,36] allocate a memory to store exemplars of previous tasks for replay when learning tasks. [21] preserves low dimensional features instead of raw instances to reduce the storage overhead. In [39], instances are synthesized by a generative model [16] for rehearsal. [30] test various exemplar selection strategies, showing that different ways of exemplar selection have a significant impact on performance and herding surpass other strategies in most settings.
Dynamic Architectures. Many works [10,15,20,33,37] create new modules to handle the growing training distribution [41,26] dynamically. However, an accurate task id, which is usually unavailable in real-life, is needed for most of these approaches to help them choose the corresponding id-specific module. Recently, methods [40,28,9] successfully apply the dynamic architectures into class incremental learning where the task id is unavailable, showing their advantages over the single backbone methods. However, as we illustrate in Sec. 1, they have two unavoidable shortcomings: (i) Continually adding new modules causes unaffordable overhead. (ii) Directly retaining old modules leads to noise in the representations of new categories, harming the performance in new classes.
Boosting. Boosting represents a family of machine learning algorithms that convert weak learners to strong ones [50]. AdaBoost [12] is one of the most famous boosting algorithms, aiming to minimize the exponential loss of the additive model. The crucial idea of AdaBoost is to adjust the weights of training samples to make the new base learner pay more attention to samples that the former ensemble model cannot recognize correctly. In recent years, gradient boosting [13] based algorithms [2,23,7] achieve excellent performance on various tasks.

3 Preliminary

In this section, we first briefly discuss the basic process of gradient boosting in Sec. 3.1. Then, we describe the setting of class-incremental learning in Sec. 3.2. In Sec. 4, we will give an explicit demonstration of how we apply the idea of gradient boosting to the scenario of class-incremental learning.

3.1 Gradient Boosting

Given a training set $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathcal{X}$ is the instance and $y_i \in \mathcal{Y}$ is the corresponding label, the gradient boosting methods seek a hypothesis $\mathrm{F} : \mathcal{X} \to \mathcal{Y}$ to minimize the empirical risk (with loss function $\ell(\cdot,\cdot)$)

$$\mathrm{F}^* = \arg\min_{\mathrm{F}} \mathbb{E}_{(x,y)\in\mathcal{D}_{train}} \left[\ell\left(y, \mathrm{F}(x)\right)\right] , \tag{1}$$

by iteratively adding a new weighted weak function $h_i(\cdot)$ chosen from a specific function space $\mathcal{H}_i$ (*e.g.*, the set of all possible decision trees) to gradually fit residuals. After m iterations, the hypothesis F can be represented as

$$\mathrm{F}(x) = \mathrm{F}_m(x) = \sum_{i=1}^{m} \alpha_i h_i(x) , \tag{2}$$

where α_i is the coefficient of $h_i(\cdot)$. Then we are supposed to find F_{m+1} for further optimization of the objective

$$\mathrm{F}_{m+1}(x) = \mathrm{F}_m(x) + \underset{h_{m+1}\in\mathcal{H}_{m+1}}{\arg\min}\ \mathbb{E}_{(x,y)\in\mathcal{D}_{train}}\left[\ell\left(y, \mathrm{F}_m(x) + h_{m+1}(x)\right)\right] . \quad (3)$$

However, directly optimizing the above function to find the best h_{m+1} is typically infeasible. Therefore, we use the steepest descent step for iterative optimization:

$$\mathrm{F}_{m+1}(x) = \mathrm{F}_m(x) - \alpha_m \nabla_{\mathrm{F}_m}\mathbb{E}_{(x,y)\in\mathcal{D}_{train}}\left[\ell\left(y, \mathrm{F}_m(x)\right)\right] , \quad (4)$$

where $-\nabla_{\mathrm{F}_m}\mathbb{E}_{(x,y)\in\mathcal{D}_{train}}\left[\ell\left(y, \mathrm{F}_m(x)\right)\right]$ is the objective for $h_{m+1}(x)$ to approximate. Specifically, if $\ell(\cdot,\cdot)$ is the mean-squared error (MSE), it transforms into

$$-\nabla_{\mathrm{F}_m}\mathbb{E}_{(x,y)\in\mathcal{D}_{train}}\left[\left(y - \mathrm{F}_m(x)\right)^2\right] = 2\times\mathbb{E}_{(x,y)\in\mathcal{D}_{train}}\left[y - \mathrm{F}_m(x)\right] . \quad (5)$$

Ideally, let $\alpha_m = 1/2$, if $h_{m+1}(x)$ can fit $2\alpha_m(y - \mathrm{F}_m(x)) = (y - \mathrm{F}_m(x))$ for each $(x, y) \in \mathcal{D}_{train}$, F_{m+1} is the optimal function, minimizing the empirical error.

3.2 Class-Incremental Learning Setup

Unlike the traditional case where the model is trained on all classes with all training data available, in class-incremental learning, the model receives a batch of new training data $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^n$ in the t^{th} stage. Specifically, n is the number of training samples, $\boldsymbol{x}_i^t \in \mathcal{X}_t$ is the input image, and $y_i^t \in \mathcal{Y}_t$ is the corresponding label for $\boldsymbol{x}_i^t$. Label space of all seen categories is denoted as $\hat{\mathcal{Y}}_t = \cup_{i=0}^t \mathcal{Y}_i$, where $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \emptyset$ for $t \neq t'$. In the t^{th} stage, rehearsal-based methods also save a part of old data as $\mathcal{V}_t$, a limited subset of $\cup_{i=0}^{t-1}\mathcal{D}_i$. Our model is trained on $\hat{\mathcal{D}}_t = \mathcal{D}_t \cup \mathcal{V}_t$ and is required to perform well on all seen categories.

4 Method

In this section, we give a description of FOSTER and how it works to prompt the model to simultaneously learn all classes well. Below, we first give a full demonstration of how the idea of the gradient boosting algorithm is applied to class-incremental learning in Sec. 4.1. Then we propose novel strategies to further enhance and balance the learning, which greatly improves the performance in Sec. 4.2. Finally, in order to avoid the explosive growth of parameters and remove redundant parameters and feature dimensions, we utilize a straightforward and effective compression method based on knowledge distillation in Sec. 4.3.

4.1 From Gradient Boosting to Class-Incremental Learning

Assuming in the t^{th} stage, we have saved the model F_{t-1} from the last stage. F_{t-1} can be further decomposed into feature embedding and linear classifier: $\mathrm{F}_{t-1}(\boldsymbol{x}) = (\mathbf{W}_{t-1})^\top \Phi_{t-1}(\boldsymbol{x})$, where $\Phi_{t-1}(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$ and $\mathbf{W}_{t-1} \in \mathbb{R}^{d\times|\hat{\mathcal{Y}}_{t-1}|}$.

When a new data stream comes, directly fine-tuning F_{t-1} on the new data will impair its capacity for old classes, which is inadvisable. On the other hand, simply freezing F_{t-1} causes it to lose plasticity for new classes, making the residuals between target y and $\mathrm{F}_{t-1}(\boldsymbol{x})$ large for $(\boldsymbol{x}, y) \in \mathcal{D}_t$. Inspired by gradient boosting, we train a new model to fit the residuals. Specifically, the new model $\mathcal{F}_t$ consists of a feature extractor $\phi_t(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$ and a linear classifier $\mathcal{W}_t \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_t|}$. $\mathcal{W}_t$ can be further decomposed into $\left[\mathcal{W}_t^{(o)}, \mathcal{W}_t^{(n)}\right]$, where $\mathcal{W}_t^{(o)} \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_{t-1}|}$ and $\mathcal{W}_t^{(n)} \in \mathbb{R}^{d \times |\mathcal{Y}_t|}$. Accordingly, the training process can be represented as

$$\mathrm{F}_t(\boldsymbol{x}) = \mathrm{F}_{t-1}(\boldsymbol{x}) + \arg\min_{\mathcal{F}_t} \mathbb{E}_{(\boldsymbol{x},y)\in\hat{\mathcal{D}}_t} \left[\ell\left(y, \mathrm{F}_{t-1}(\boldsymbol{x}) + \mathcal{F}_t(\boldsymbol{x})\right)\right] . \tag{6}$$

Similar to Sec. 3.1, let $\ell(\cdot,\cdot)$ be the mean-squared error function, considering the strong feature representation learning ability of neural networks, we expect $\mathcal{F}_t(\boldsymbol{x})$ can fit residuals of y and $\mathrm{F}_{t-1}(\boldsymbol{x})$ for every $(\boldsymbol{x}, y) \in \hat{\mathcal{D}}_t$. Ideally, we have

$$\boldsymbol{y} = \mathrm{F}_{t-1}(\boldsymbol{x}) + \mathcal{F}_t(\boldsymbol{x}) = \mathcal{S}\left(\begin{bmatrix}\mathbf{W}_{t-1}^\top \\ \mathbf{O}\end{bmatrix} \Phi_{t-1}(\boldsymbol{x})\right) + \mathcal{S}\left(\begin{bmatrix}(\mathcal{W}_t^{(o)})^\top \\ (\mathcal{W}_t^{(n)})^\top\end{bmatrix} \phi_t(\boldsymbol{x})\right) , \tag{7}$$

where $\mathcal{S}(\cdot)$ is the softmax operation, $\mathbf{O} \in \mathbb{R}^{d \times |\mathcal{Y}_t|}$ is set to zero matrix or fine-tuned on $\hat{\mathcal{D}}_t$ with Φ_{t-1} frozen, and $\boldsymbol{y}$ is the corresponding one-hot vector of y. We set $\mathbf{O}$ to zero matrix as default in our discussion.

Denote the parameters of $\mathcal{F}_t$ as θ_t and $\mathrm{Dis}(\cdot,\cdot)$ as a distance metric (*e.g.*, euclidean metric), this process can be represented as the following optimization problem:

$$\theta_t^* = \arg\min_{\theta_t} \mathrm{Dis}\left(\boldsymbol{y}, \mathcal{S}\left(\begin{bmatrix}\mathbf{W}_{t-1}^\top \\ \mathbf{O}\end{bmatrix} \Phi_{t-1}(\boldsymbol{x})\right) + \mathcal{S}\left(\begin{bmatrix}(\mathcal{W}_t^{(o)})^\top \\ (\mathcal{W}_t^{(n)})^\top\end{bmatrix} \phi_t(\boldsymbol{x})\right)\right) . \tag{8}$$

We replace the $\mathcal{S}(\cdot) + \mathcal{S}(\cdot)$ with $\mathcal{S}(\cdot + \cdot)$ and substitute the $\mathrm{Dis}(\cdot,\cdot)$ for the Kullback-Leibler divergence (KLD), then the objective function changes into:

$$\theta_t^* = \arg\min_{\theta_t} \mathrm{KL}\left(y \,\middle\|\, \mathcal{S}\left(\begin{bmatrix}\mathbf{W}_{t-1}^\top & (\mathcal{W}_t^{(o)})^\top \\ \mathbf{O} & (\mathcal{W}_t^{(n)})^\top\end{bmatrix} \begin{bmatrix}\Phi_{t-1}(\boldsymbol{x}) \\ \phi_t(\boldsymbol{x})\end{bmatrix}\right)\right) . \tag{9}$$

We provide an illustration about the reasons for this substitution in the supplementary material. Therefore, F_t can be further decomposed as an expanded linear classifier $\mathbf{W}_t$ and a concatenated super feature extractor $\Phi_t(\cdot)$, where

$$\mathbf{W}_t^\top = \begin{bmatrix}\mathbf{W}_{t-1}^\top & (\mathcal{W}_t^{(o)})^\top \\ \mathbf{O} & (\mathcal{W}_t^{(n)})^\top\end{bmatrix} , \qquad \Phi_t(\boldsymbol{x}) = \begin{bmatrix}\Phi_{t-1}(\boldsymbol{x}) \\ \phi_t(\boldsymbol{x})\end{bmatrix} . \tag{10}$$

Note that $\mathbf{W}_{t-1}^\top$, $\mathbf{O}$, and Φ_{t-1} are all frozen, the trainable modules are the $\phi_t, \mathcal{W}_t^{(o)}, \mathcal{W}_t^{(n)}$. Here we explain their roles. Eventually, logits of F_t is

$$\mathbf{W}_t^\top \Phi_t(\boldsymbol{x}) = \begin{bmatrix}\mathbf{W}_{t-1}^\top \Phi_{t-1}(\boldsymbol{x}) + (\mathcal{W}_t^{(o)})^\top \phi_t(\boldsymbol{x}) \\ (\mathcal{W}_t^{(n)})^\top \phi_t(\boldsymbol{x})\end{bmatrix} . \tag{11}$$

The lower part is the logits of new classes, and the upper part is that of old ones. As we claimed in Sec. 1, the lower part requires the new module $\mathcal{F}_t$ to learn how to correctly classify new classes, thus enhancing the model's plasticity to redeem the performance on new classes. The upper part encourages the new module to fit the residuals between y and F_{t-1}, thus encouraging $\mathcal{F}_t$ to exploit more pivotal patterns for classification.

4.2 Calibration for Old and New

When training on new tasks, we only have an imbalanced training set $\hat{\mathcal{D}}_t = \mathcal{D}_t \cup \mathcal{V}_t$. The imbalance on categories of $\mathcal{D}_t$ will result in a strong classification bias in the model [22,43,38,1]. Besides, the boosting model tends to ignore the residuals of minor classes due to insufficient supervision. To alleviate the classification bias and encourage the model to equally learn old and new classes, we propose Logits Alignment and Feature Enhancement strategies in the following sections.

Logits Alignment. To strengthen the learning of old instances and mitigate the classification bias, we add a scale factor to the logits of the old and new classes in Eq. 11 respectively during training. Thus, the logits during training are:

$$\boldsymbol{\gamma}\mathbf{W}_t^\top \Phi_t(\boldsymbol{x}) = \begin{bmatrix} \gamma_1 \left(\mathbf{W}_{t-1}^\top \Phi_{t-1}(\boldsymbol{x}) + (\mathcal{W}_t^{(o)})^\top \phi_t(\boldsymbol{x}) \right) \\ \gamma_2 (\mathcal{W}_t^{(n)})^\top \phi_t(\boldsymbol{x}) \end{bmatrix}, \tag{12}$$

where $0 < \gamma_1 < 1$, $\gamma_2 > 1$, and $\boldsymbol{\gamma}$ is a diagonal matrix composed of γ_1 and γ_2. Through this scaling strategy, the absolute value of logits for old categories is reduced, and the absolute value of logits for new ones is enlarged, thus forcing the model F_t to produce larger logits for old categories and smaller logits for new categories.

We get the scale factors γ_1, γ_2 trough the normalized effective number E_n [4] of each class, which can be seen as the summation of proportional series, where n equal to the number of instances and β is an adjustable hyperparameter

$$E_n = \begin{cases} \frac{1-\beta^n}{1-\beta}, & \beta \in [0, 1) \\ n, & \beta = 1 \end{cases}, \tag{13}$$

concretely, $(\gamma_1, \gamma_2) = \left(\frac{E_{n_{\text{old}}}}{E_{n_{\text{old}}} + E_{n_{\text{new}}}}, \frac{E_{n_{\text{new}}}}{E_{n_{\text{old}}} + E_{n_{\text{new}}}} \right)$. Hence the objective is formulated as:

$$\mathcal{L}_{LA} = \mathrm{KL}\big(y \,\big\|\, \mathcal{S}\left(\boldsymbol{\gamma}\mathbf{W}_t^\top \Phi_t(\boldsymbol{x})\right)\big). \tag{14}$$

Feature Enhancement. We argue that simply letting a new module $\mathcal{F}_t(\boldsymbol{x})$ fit the residuals of $\mathrm{F}_{t-1}(\boldsymbol{x})$ and label y is sometimes insufficient. At the extreme,, for instance, the residuals of $\mathrm{F}_{t-1}(\boldsymbol{x})$ and y is zero. In that case, the new module $\mathcal{F}_t$ can not learn anything about old categories, and thus it will damage the performance of our model for old classes. Hence, we should prompt the new module $\mathcal{F}_t$ to learn old categories further.

Our Feature Enhancement consists of two parts. First, we initialize a new linear classifier $\mathbf{W}_t^{(a)} \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_t|}$ to transform the new feature $\phi_t(\boldsymbol{x})$ into logits of

all seen categories and require the new feature itself to correctly classify all of them:

$$\mathcal{L}_{FE} = \mathrm{KL}\Big(y \,\Big\|\, \mathcal{S}\left((\mathbf{W}_t^{(a)})^\top \phi_t(\boldsymbol{x})\right)\Big)\,. \tag{15}$$

Hence, even if the residuals of $\mathrm{F}_{t-1}(\boldsymbol{x})$ and y is zero, the new feature extractor ϕ_t can still learn how to classify the old categories. Besides, it should be noted that simply using one-hot targets to train the new feature extractor in an imbalanced dataset might lead to overfitting to small classes, failing to learn a feature representation with good generalization ability for old categories. To alleviate this phenomenon and provide more supervision for old classes, we utilize knowledge distillation to encourage $\mathrm{F}_t(\boldsymbol{x})$ to have similar output distribution as F_{t-1} on old categories,

$$\mathcal{L}_{KD} = \mathrm{KL}\Big(\mathcal{S}\left(\mathrm{F}_{t-1}(\boldsymbol{x})\right) \,\Big\|\, \mathcal{S}\left(\mathrm{F}_{t-1}(\boldsymbol{x}) + (\mathcal{W}_t^{(o)})^\top \phi_t(\boldsymbol{x})\right)\Big)\,. \tag{16}$$

Note that this process requires only one more time matrix multiplication computation because the forward process of the original model F_{t-1} and the expanded model F_t are shared, except for the final linear classifier.

Summary of Feature Boosting. To conclude, feature-boosting consists of three components. First, we create a new module to fit the residuals between targets and the output of the original model, following the principle of gradient boosting. With reasonable simplification and deduction, the optimization objective is transformed into the minimization of KL divergence of the target and the output of the concatenated model. To alleviate the classification bias caused by imbalanced training, we proposed logits alignment (LA) to balance the training of old and new classes. Moreover, we argued that simply letting the new module fit the residuals is sometimes insufficient. To further encourage the new module to learn old instances, we proposed feature enhancement, where $\mathcal{L}_{FE}$ aims to make the new module learn the difference among all categories by optimizing the cross-entropy loss of target and the output of the new module, and $\mathcal{L}_{KD}$ utilize the original output to instruct the expanded model through knowledge distillation. The final FOSTER loss for boosting combines the above three components:

$$\mathcal{L}_{Boosting} = \mathcal{L}_{LA} + \mathcal{L}_{FE} + \mathcal{L}_{KD}\,. \tag{17}$$

4.3 Feature Compression

Our method FOSTER achieves excellent performance through gradient boosting. However, gradually adding a new module $\mathcal{F}$ to our model F_t will lead to the growing number of parameters and feature dimensions of our model F_t, making it unable to be applied in long-term incremental learning tasks. Do we really require so many parameters and feature dimensions? For example, we create the same module $\mathcal{F}$ to learn tasks with 2 classes and 50 classes and achieve similar effects.

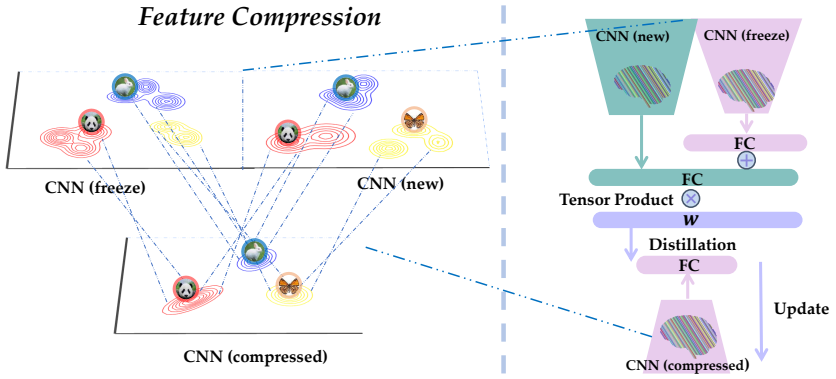


Fig. 2: **Feature Compression.** Left: the process of feature compression. We remove insignificant dimensions and parameters to make the distribution of the same categories more compact. Right: the implementation of feature compression. Outputs of the dual branch model are used to instruct the representation learning of the compressed model. Different weights are assigned to old and new classes to alleviate the classification bias.

Thus, there must be redundant parameters and meaningless feature dimensions in the task with 2 classes. Are we able to compress the expanded feature space of F_t to a smaller one with almost no performance degradation?

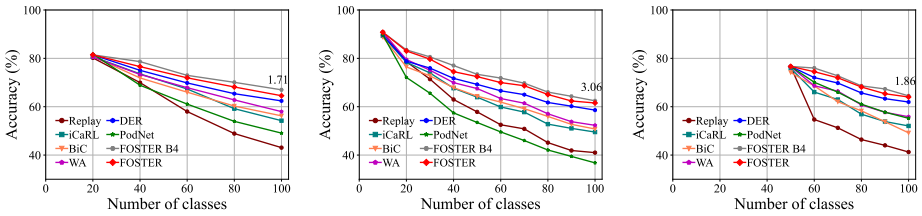
Knowledge distillation [19] is a simple yet effective way to achieve this goal. Since our model F_t can handle all seen categories with excellent performance, it can give any input a soft target, namely the output distribution on all known categories. Therefore, except for the current training set $\hat{\mathcal{D}}_t$, we can sample other unlabeled data from a similar domain for further distillation. Note that these unlabeled data can be obtained from the Internet during distillation and discarded after that, so it does not occupy additional memory.

Here, we do not expect any additional auxiliary data to be available and achieve remarkable performance with only the imbalanced dataset $\hat{\mathcal{D}}_t$.

Balanced Distillation. Suppose there is a single backbone student model $\mathrm{F}_t^{(s)}$to be distilled. To mitigate the classification bias caused by imbalanced training datasets $\hat{\mathcal{D}}_t$, we should consider the class priors and adjust the weights of distilled information for different classes [42]. Therefore, the Balanced Distillation loss is formulated as:

$$\mathcal{L}_{\mathrm{BKD}} = \mathrm{KL}\Big(\boldsymbol{w} \otimes \mathcal{S}\left(\mathrm{F}_t(\boldsymbol{x})\right) \Big\| \mathcal{S}(\mathrm{F}_t^{(s)}(\boldsymbol{x}))\Big), \tag{18}$$

where $\otimes$ means the tensor product (*i.e.*, automatically broadcasting to different batchsizes.) and $\boldsymbol{w}$ is the weighted vector obtained from Eq. 13 to make classes with fewer instances have larger weights.



(a) CIFAR-100 B0 5 steps (b) CIFAR-100 B0 10 steps (c) CIFAR-100 B50 5 steps

Fig. 3: **Incremental Accuracy on CIFAR-100.** Replay is the baseline with naive rehearsal strategy. FOSTER B4 records the accuracy of the dual branch model after feature boosting. FOSTER records the accuracy of the single backbone model after feature compression. The performance gap is annotated at the end of each curve.

5 Experiments

In this section, we compare our FOSTER with other SOTA methods on benchmark incremental learning datasets. We also perform ablations to validate the effectiveness of FOSTER components and their robustness to hyperparameters.

5.1 Experimental Settings

Datasets. We validate our methods on widely used benchmark of class-incremental learning CIFAR-100 [25] and ImageNet100/1000 [6]. **CIFAR-100**: CIFAR-100 consists of 50,000 training images with 500 images per class, and 10,000 testing images with 100 images per class. **ImageNet-1000**: ImageNet-1000 is a large scale dataset composed of about 1.28 million images for training and 50,000 for validation with 500 images per class. **ImageNet-100**: ImageNet-100 is composed of 100 classes randomly chosen from the original ImageNet-1000.
Protocol. For both the CIFAR-100 and ImageNet-100, we validate our method on two widely used protocols: (i) **CIFAR-100/ImageNet-100 B0 (base 0)**: In the first protocols, we train all 100 classes gradually with 5, 10, 20 classes per step with the fixed memory size of 2,000 exemplars. (ii) **CIFAR-100/ImageNet-100 B50 (base 50)**: We also start by training the models on half the classes. Then we train the rest 50 classes with 2, 5, 10 classes per step with 20 exemplars per class. For ImageNet-1000, we train all 1000 classes with 100 classes per step (10 steps in total) with a fixed memory size of 20,000 exemplars.
Implementation Details. Our method and all compared methods are implemented with Pytorch [31] and PyCIL [45]. For ImageNet, we adopt the standard ResNet-18 [18] as our feature extractor and set the batch size to 256. The learning rate starts from 0.1 and gradually decays to zero with a cosine annealing scheduler [29] (170 epochs in total). For CIFAR-100, we use a modified ResNet-32 [32] as the most previous works as our feature extractor and set the batch size to 128. The learning rate also starts from 0.1 and gradually decays to zero with a cosine annealing scheduler (170 epochs in total). For both ImageNet and

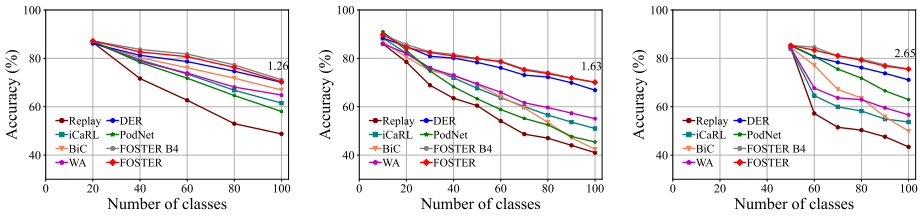
Table 1: Average incremental accuracy on CIFAR-100 for FOSTER vs. state-of-the-art. DER uses the same number of backbone models as incremental sessions, while the other methods, including FOSTER, retain only one backbone after each session.

Methods	Average accuracy of all sessions (%)			
	B0 10 steps	**B0 20 steps**	**B50 10 steps**	**B50 25 steps**
Bound	80.40	80.41	81.49	81.74
iCaRL [32]	64.42	63.5	53.78	50.60
BiC [38]	65.08	62.37	53.21	48.96
WA [43]	67.08	64.64	57.57	54.10
COIL[47]	65.48	62.98	59.96	-
PODNet [8]	55.22	47.87	63.19	60.72
DER [40]	69.74	67.98	66.36	-
Ours	**72.90**	**70.65**	**67.95**	**63.83**
Improvement	**(+3.06)**	**(+2.67)**	**(+1.59)**	**(+3.11)**

CIFAR-100, we use SGD with the momentum of 0.9 and the weight decay of 5e-4 in the boosting stage. In the compression stage, we use SGD with the momentum of 0.9 and set the weight decay to 0. We set the temperature scalar T to 2. For data augmentation, AutoAugment [3], random cropping, horizontal flip, and normalization are employed to augment training images. The hyperparameter β in Eq. 18 is set to 0.97 in most settings, while the β in Eq. 14 on CIFAR-100 and ImageNet-100/1000 is set to 0.95 and 0.97, respectively.

5.2 Quantitative results

CIFAR-100. Table 1 and Fig. 3 summarize the results of CIFAR-100 benchmark. We use replay as the baseline method, which only uses rehearsal strategy to alleviate forgetting. Experimental results show that our method outperforms the other state-of-the-art strategies in all six settings on CIFAR-100. Our method achieves excellent performance on both long-term incremental learning tasks and large-step incremental learning tasks. Particularly, we achieve 3.11% and 2.67% improvement under the long-term incremental setting of base 50 with 25 steps and base 0 with 20 steps, respectively. We also surpass the state-of-the-art method by 1.71% and 3.06% under the large step incremental learning setting of 20 classes per step and 10 classes per step. It should also be noted that although our method FOSTER expands a new module every time, we compress it to a single backbone every time. Therefore, the parameters and feature dimensions of our model do not increase with the number of tasks, which is our advantage over methods [40,28,9] based on dynamic architecture. From Fig. 3, we can see that the compressed single backbone model FOSTER has a tiny gap with FOSTER B4 in each step, which verifies the effectiveness of our distillation method.
ImageNet. Table 2 and Fig. 4 summarize the experimental results for ImageNet-100 and ImageNet-1000 benchmarks. Our method, FOSTER, still outperforms the other method in most settings. In the setting of ImageNet-100 B0, we surpass the state-of-the-art method by 1.26, 1.63, and 0.7 percent points for, respectively,



(a) ImageNet-100 B0 5 steps (b) ImageNet-100 B0 10 steps (c) ImageNet-100 B50 5 steps

Fig. 4: **Incremental Accuracy on ImageNet-100.** Replay is the baseline with naive rehearsal strategy. FOSTER B4 records the accuracy of the dual branch model after feature boosting. FOSTER records the accuracy of the single backbone model after feature compression. The performance gap is annotated at the end of each curve.

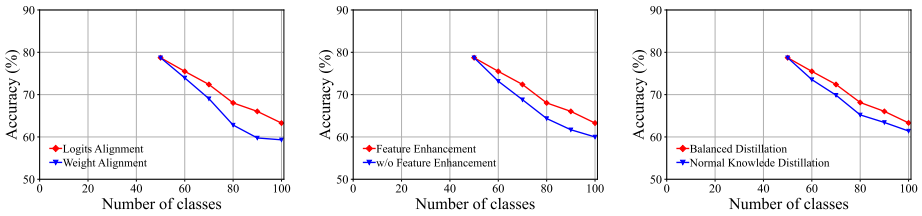
Table 2: Average incremental accuracy on ImageNet for FOSTER vs. state-of-the-art. DER uses the same number of backbone models as incremental sessions, while the other methods, including FOSTER, retain only one backbone after each session. The left three columns are experimental results on ImageNet-100. The rightmost column is the results of ImageNet-1000 with 100 classes per step (10 steps in total).

Methods	Average accuracy of all sessions (%)			
	B0 20 **steps**	**B**50 10 **steps**	**B**50 25 **steps**	**ImageNet-1000**
Bound	81.20	81.20	81.20	89.27
iCaRL [32]	62.36	59.53	54.56	38.4
BiC [38]	58.93	65.14	59.65	-
WA [43]	63.2	63.71	58.34	54.10
PODNet [8]	53.69	74.33	67.28	-
DER [40]	73.79	77.73	-	66.73
Ours	**74.49**	**77.54**	**69.34**	**68.34**
Improvement	**(+0.7)**	(−0.19)	**(+2.06)**	**(+1.61)**

5, 10, and 20 steps. The results shown in Fig. 4 again verify the effectiveness of our distillation strategy, where the performance degradation after compression is negligible. The results on ImageNet-1000 benchmark is shown in the rightmost column in Tabel 2. Our method improves the average top-1 accuracy on ImageNet-1000 with 10 steps from 66.73% to 68.34% (+1.61%), showing that our method is also efficacious in large-scale incremental learning.

5.3 Ablation Study

Different Components of FOSTER. Table 5 demonstrates the results of our ablative experiments on CIFAR-100 B50 with 5 steps. Specifically, we replace logits alignments (LA) with the post-processing method weight alignment (WA) [43]. The performance comparison is shown in Fig. 5a, where LA surpasses WA by about 4% in the final accuracy. This shows that our LA is a more efficacious strategy than WA in calibration for old and new classes. We remove feature enhancement and compare its performance with the original result



(a) Logits alignment (b) Feature enhancement (c) Balanced distillation

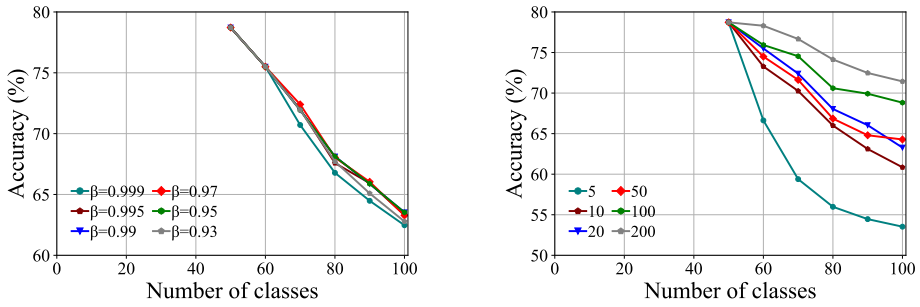
Fig. 5: **Ablations** of the different key components of FOSTER. (a): Performance comparison between logits alignment and weight alignment [43]. (b): Performance comparison with or without Feature Enhancement. (c): Performance comparison between balanced distillation and normal knowledge distillation [19].

in Fig. 5b, the model suffers from more than 3% performance decline in the last stage. We find that, in the last step, there is almost no difference in the accuracy of new classes between the model with feature enhancement and the model without that. Nevertheless, the model with feature enhancement outperforms the model without that by more than 4 % on old categories, showing that feature enhancement encourages the model to learn more about old categories. We compare the performance of balanced knowledge distillation (BKD) with that of normal knowledge distillation (KD) in Fig. 5c. BKD surpasses KD in all stages, showing that BKD is more effective when training on imbalanced datasets.

Sensitive Study of Hyper-parameters. To verify the robustness of FOSTER, we conduct experiments on CIFAR-100 B50 5 steps with different hyperparameters $\beta \in (0, 1)$. Typically, β is set to more than 0.9. We test $\beta = 0.93$, 0.95, 0.97, 0.99, 0.995, 0.999 respectively. The experimental results are shown in Fig. 6a. We can see that the performance changes are minimal under different βs.

Effect of Number of Exemplars. In Fig. 6b, We gradually increase the number of exemplars from 5 to 200 and record the performance of the model on CIFAR-100 B50 with 5 steps. The accuracy in the last step increases from 53.53% to 71.4% as the number of exemplars for every class changes from 5 to 200. From the results, we can see that with the increase in the number of exemplars, the accuracy of the last stage of the model gradually improves, indicating that our model can make full use of more exemplars to improve performance. In addition, notice that our model achieves more than 60% accuracy in the last round, even when there are only 10 exemplars for each class, surpassing most state-of-the-art methods using 20 exemplars shown in Fig. 3c. This indicates that FOSTER is more effective and robust; it can overcome forgetting even with fewer exemplars.

Visualization of Grad-CAM. We visualize the grad-CAM before and after feature boosting. As shown in Fig. 7 (left), the freeze CNN only focuses on the head of the birds, ignoring the rest of their bodies, while the new CNN learns that the whole body is important for classification, which is consistent with our claim in Sec. 1. Similarly, the middle and right figures show that the new CNN also discovers some essential but ignored patterns of the mailbox and the dog.



(a) Sensitive study of hyper-parameters (b) Influence of number of exemplars

Fig. 6: **Robustness Testing.** Left: Performance under different hyperparameter βs. Right: Performance with different numbers of exemplars. Both of them are evaluated on CIFAR-100 B50 with 5 steps.

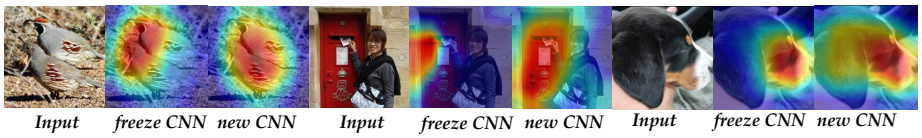


Fig. 7: **Grad-CAM before and after feature boosting.** The freeze CNN only focuses on some areas of an object and is not accurate enough, but the new CNN can discover those important but ignored patterns and correct the original output.

6 Conclusions

In this work, we apply the concept of gradient boosting to the scenario of class-incremental learning and propose a novel learning paradigm FOSTER based on that, empowering the model to learn new categories adaptively. At each step, we create a new module to learn residuals between the target and the original model. We also introduce logits alignment to alleviate classification bias and feature enhancement to balance the representation learning of the old and new classes. Furthermore, we propose a simple yet effective distillation strategy to remove redundant parameters and dimensions, compressing the expanded model into a single backbone model. Extensive experiments on three widely used incremental learning benchmarks show that our method obtains state-of-the-art performance.

Acknowledgments. This research was supported by National Key R&D Program of China (2020AAA0109401), NSFC (61773198, 61921006,62006112), NSFC-NRF Joint Research Project under Grant 61861146001, Collaborative Innovation Center of Novel Software Technology and Industrialization, NSF of Jiangsu Province (BK20200313), CCF-Hikvision Open Fund (20210005). Han-Jia Ye is the corresponding author.

References

1. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 233–248 (2018)
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: KDD. pp. 785–794 (2016)
3. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: CVPR. pp. 113–123 (2019)
4. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: CVPR. pp. 9268–9277 (2019)
5. Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. Ieee (2009)
7. Dorogush, A.V., Ershov, V., Gulin, A.: Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363 (2018)
8. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: ECCV. pp. 86–102. Springer (2020)
9. Douillard, A., Ramé, A., Couairon, G., Cord, M.: Dytox: Transformers for continual learning with dynamic token expansion. arXiv preprint arXiv:2111.11326 (2021)
10. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734 (2017)
11. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
12. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics **28**(2), 337–407 (2000)
13. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
14. Golab, L., Özsu, M.T.: Issues in data stream management. ACM Sigmod Record **32**(2), 5–14 (2003)
15. Golkar, S., Kagan, M., Cho, K.: Continual learning via neural pruning. arXiv preprint arXiv:1903.04476 (2019)
16. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
17. Grossberg, S.: Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. Neural networks **37**, 1–47 (2013)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
19. Hinton, G., Vinyals, O., Dean, J., et al.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 **2**(7) (2015)
20. Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. Advances in Neural Information Processing Systems **32** (2019)
21. Iscen, A., Zhang, J., Lazebnik, S., Schmid, C.: Memory-efficient incremental learning through feature adaptation. In: ECCV. pp. 699–715. Springer (2020)

22. Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., Kalantidis, Y.: Decoupling representation and classifier for long-tailed recognition. arXiv preprint arXiv:1910.09217 (2019)
23. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30** (2017)
24. Korattikara Balan, A., Rathod, V., Murphy, K.P., Welling, M.: Bayesian dark knowledge. Advances in Neural Information Processing Systems **28** (2015)
25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
26. Lesort, T., Caselles-Dupré, H., Garcia-Ortiz, M., Stoian, A., Filliat, D.: Generative models from the perspective of continual learning. In: IJCNN. pp. 1–8. IEEE (2019)
27. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
28. Li, Z., Zhong, C., Liu, S., Wang, R., Zheng, W.S.: Preserving earlier knowledge in continual learning with the help of all previous feature extractors. arXiv preprint arXiv:2104.13614 (2021)
29. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
30. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277 (2020)
31. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
32. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 2001–2010 (2017)
33. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
34. Wang, L., Yang, K., Li, C., Hong, L., Li, Z., Zhu, J.: Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In: CVPR. pp. 5383–5392 (2021)
35. Wang, L., Zhang, M., Jia, Z., Li, Q., Bao, C., Ma, K., Zhu, J., Zhong, Y.: Afec: Active forgetting of negative transfer in continual learning. NeurIPS **34**, 22379–22391 (2021)
36. Wang, L., Zhang, X., Yang, K., Yu, L., Li, C., Lanqing, H., Zhang, S., Li, Z., Zhong, Y., Zhu, J.: Memory replay with data compression for continual learning. In: ICLR (2022)
37. Wen, Y., Tran, D., Ba, J.: Batchensemble: an alternative approach to efficient ensemble and lifelong learning. arXiv preprint arXiv:2002.06715 (2020)
38. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)
39. Wu, Z., Baek, C., You, C., Ma, Y.: Incremental learning via rate reduction. In: CVPR. pp. 1125–1133 (2021)
40. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: CVPR. pp. 3014–3023 (2021)
41. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)
42. Zhang, S., Chen, C., Hu, X., Peng, S.: Balanced knowledge distillation for long-tailed learning. arXiv preprint arXiv:2104.10510 (2021)

43. Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S.T.: Maintaining discrimination and fairness in class incremental learning. In: CVPR. pp. 13208–13217 (2020)
44. Zhou, D.W., Wang, F.Y., Ye, H.J., Ma, L., Pu, S., Zhan, D.C.: Forward compatible few-shot class-incremental learning. In: CVPR. pp. 9046–9056 (2022)
45. Zhou, D.W., Wang, F.Y., Ye, H.J., Zhan, D.C.: Pycil: A python toolbox for class-incremental learning. arXiv preprint arXiv:2112.12533 (2021)
46. Zhou, D.W., Yang, Y., Zhan, D.C.: Learning to classify with incremental new class. IEEE Transactions on Neural Networks and Learning Systems (2021)
47. Zhou, D.W., Ye, H.J., Zhan, D.C.: Co-transport for class-incremental learning. In: ACM MM. pp. 1645–1654 (2021)
48. Zhou, D.W., Ye, H.J., Zhan, D.C.: Learning placeholders for open-set recognition. In: CVPR. pp. 4401–4410 (2021)
49. Zhou, D.W., Ye, H.J., Zhan, D.C.: Few-shot class-incremental learning by sampling multi-phase tasks. arXiv preprint arXiv:2203.17030 (2022)
50. Zhou, Z.H.: Ensemble methods: foundations and algorithms. CRC press (2012)