Overcoming catastrophic forgetting with classifier expander

Xinchen Liu xinchen_liu@bupt.edu.cn
Hongbo Wang[*] hbwang@bupt.edu.cn
Yingjian Tian tianyj@bupt.edu.cn
Linyao Xie xiely2000@bupt.edu.cn
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

It is essential for models to gradually adapt to the world's increasing complexity, and we can use models more effectively if they keep up with the times. However, the technique of continuous learning (CL) has an issue with catastrophic forgetting, and effective continuous learning methods can only be attained by effectively limiting forgetting and learning new tasks. In this study, we offer the Classifier Expander(CE) method, which combines the regularization-based and replay-based approaches. By undergoing two stages of training, it fulfills the aforementioned standards. The training content for the new task is limited to the portion of the network relevant to that task in the first stage, which uses the replay approach to reduce the forgetting problem. This strategy minimizes disruption to the old task while facilitating efficient learning of the new one. Utilizing all of the data available, the second stage retrains the network and sufficiently trains the classifier to balance the learning performance of the old and new tasks. Our method regularly outperforms previous CL methods on the CIFAR-100 and CUB-200 datasets, obtaining an average improvement of 2.94% on the class-incremental learning and 1.16% on the task-incremental learning compared to the best method currently available. Our code is available at https://github.com/EmbraceTomorrow/CE.

Keywords: catastrophic forgetting; continuous learning; classifier expander

1. Introduction

AlexNet Krizhevsky et al. (2017) greatly beat SVM Hearst et al. (1998) in picture classification, which has led to an explosion in deep neural network research. ResNet He et al. (2016) was able to beat average human accuracy in the field of picture categorization for the first time in 2015. It seems that the learning ability of deep neural networks in the field of image classification is no doubt. However, if you look closely at the datasets used in their experiments or competitions, you will find that the training dataset and test dataset are independently and identically distributed. Deep neural networks are capable of working well under this presumption; otherwise, performance will drastically decline. The independently identically distribution is simple to violate because the actual application scenario is frequently subject to change. The user experience will be subpar if the model is unable to implement sensible changes.

[*] Corresponding author

Continuous learning is a paradigm that involves acquiring new knowledge while maintaining existing competence. A deep neural network-based application must be able to keep learning in order to be useful. For instance, if a recognition robot can only recognize items that it was trained to recognize before it left the factory, a large number of things might not be recognized in the dynamic world of today, providing users with a bad experience.

Humans are easily able to build on their prior knowledge and solve new problems. Parametric models such as deep neural networks can get into trouble in continuous learning. The performance of the model on the old tasks may significantly deteriorate if the model parameters are updated to adapt to the new task when learning a new task, a phenomenon known as catastrophic forgetting McCloskey and Cohen (1989). However, in order to ensure the performance of the old tasks, we can only limit the updating range of model parameters, which will cause the network model cannot learn the features of the new task. This is the stability-plasticity dilemma in continuous learning Mermillod et al. (2013).

We initially establish the boundaries of our continuous learning task before proposing a solution. Because if there were no restrictions, we could obtain all the training data required for joint training, the model could align its features jointly and perform well on each target class, and the stability-plasticity paradox would not exist. In reality, experiences (we refer to each learning task as an experience Lomonaco et al. (2021) to help us distinguish the meaning of the term "task" in task incremental learning) typically arrive in sequential order, and we might not be able to save all the data (possibly due to storage issues) or perform joint training even if we do(possibly due to computational requirements issues). As a result, our algorithm design needs to satisfy the following properties Rebuffi et al. (2017):

1) It can be trained in a sequential flow of experience to get a good classifier.

2) At the end of any experience training in the experience stream, competitive classifiers are provided for categories of current and past experience.

3) Its computing requirements and memory footprint should remain limited, or at least grow very slowly relative to the number of classes that have been seen.

The current regularization-based method is unable to produce a strong classification result in continuous learning tasks Lesort (2020), and the dynamic architecture method has high computational and memory requirements, which violates the third property. Data must be added to memory during training for the replay-based approach, but this rise in computational and memory needs is acceptable when compared to the dynamic architecture approach. Our approach combines regularization-based and replay-based methods, which can fully utilize memory to enhance model performance under low computational demand and memory constraints.

Our approach is a two-stage method. In the first stage, we train the network with the old task data stored in memory and the data for the new task to preserve the performance of the previous task and learn the new task. We confine the classifier to learning solely within the new task at this part of new task learning. The classifier's learnable range is expanded to include all previous tasks in the second stage during which we train the network using the in-memory data, which increases the overall prediction accuracy. Therefore, we refer to this technique as Classifier Expander(CE), which extends the training range for linear classifier.

The properties specified in three aspects are fully supported by the method(CE) proposed in this work. It stores a set number of images (much fewer than the whole amount of training data), does not alter the network structure (other than the necessary modification of the numbers of classifier classes), and outputs a reliable classifier at the conclusion of any training. It makes use of knowledge distillation to its full potential, reduces the problem of catastrophic forgetting, and shows strong classification performance.

In Section 2, we will introduce the related works and contrast our viewpoints. I'll go into more detail regarding our method and the rationale behind its creation in Section 3. In Section 4, we evaluate the CIFAR-10, CIFAR-100, and CUB-200 datasets to show the efficacy of CE in continuous learning. Finally, the remaining limitations and future work are discussed in section 5.

2. Related work

Methods for continual learning generally are divided into three categories: regularization methods, dynamic network architectures methods, and replay-based methods De Lange et al. (2021); Hadsell et al. (2020); Parisi et al. (2019). The CE method is an attempt to combine regularization-based methods and replay-based methods. In this section, we will first briefly introduce these three categories of continual learning methods, then introduce the relevant techniques used in our method, and finally provide a detailed comparison between our approach and the methods that are similar to ours.

2.1. Continuous Learning Methods

2.1.1. Regularization-based Methods

The regularization-based methods alleviate catastrophic forgetting from the perspective of the parameter. When the model learns a new task, the performance of the previous task decreases sharply due to the greatly updated parameter. The regularization-based methods try to prevent large parameter modifications to reduce the forgetting effect. Regularization-based methods can be divided into data-focused methods such as Li and Hoiem (2017); Jung et al. (2016); Zhang et al. (2020) and prior-focused methods such as Kirkpatrick et al. (2017); Zenke et al. (2017). The most classical Lwf Li and Hoiem (2017) added knowledge distillation loss to limit the update of model parameters, slow down model forgetting, and realize state-of-the-art at that time. Kirkpatrick et al. (2017); Zenke et al. (2017) determined the importance of the parameters of the model by using the diagonal elements of the Fisher information matrix and the sensitivity of the loss function to the parameters respectively. Subsequently, they imposed penalties on the modification of important parameters to limit the updating of parameters and reduce the forgetting of the model.

2.1.2. dynamic architecture methods

Dynamic network architecture methods modify the structure of the network model, such as PNN Rusu et al. (2016), DEN Yoon et al. (2017). PNN does not change the parameters of the network model corresponding to the old task, but adds a new network structure directly for the new task, freezes all the parameters related to the old task during training, and only trains the network structure related to the new task. DEN selectively retrains the old network, expanding its capacity as necessary, thereby dynamically determining its optimal capacity as it continues to operate. This type of approach is often able to achieve state-of-the-art in terms of accuracy, but this approach, where the computational demand and memory consumption grow linearly with the number of tasks, is undesirable in the long-term continuous learning process.

2.1.3. replay-based methods

Retraining the data of old tasks can also alleviate model forgetting, which is referred to as playback methods Bagus and Gepperth (2021). replay-based methods are generally divided into two categories: retaining-based methods such as iCaRL Rebuffi et al. (2017), DER Buzzega et al. (2020) and generating-based methods such as DGR Shin et al. (2017). Retaining-based methods fix the capacity of the network to store old data. As the number of old tasks increases, the number of retained samples corresponding to each old task will decrease. Generating-based methods train a generator, then the model is trained with the data generated by the generator and the new task data together. This method can alleviate the forgetting problem and may achieve better task results. Nevertheless, both methods require either sufficient storage or additional training of the generator.

2.2. Comparative Analysis

The most relevant works to our method are DER Buzzega et al. (2020) and TwF Boschini et al. (2022). In reproducing these two works, we found that the performance improvement of DER for task increment is very significant, therefor our work continues to follow the dark experience replay-based approach. The dark experience replay method refers to that when learning a new task, the saved data of the old task is put into the training data set, and the knowledge distillation method is used to maintain the old task performance. As mentioned earlier, our work is a two-stage approach, with the first stage using a variant of the dark experience replay-based approach and the second stage leveraging the old task data stored in memory to improve the performance of class-Incremental tasks. Our first-stage approach is a variation on DER, and we also adopt TwF's approach of using pre-trained model weights for training.

In addition, LwF Li and Hoiem (2017) is also a two-stage method, its first stage is warm-up training, and the second stage starts the overall training, and it does not use memory to store old information. In contrast, Our method trains the whole network in the first stage and then trains the classifier in the second stage. In the experiment section, we will add LwF and LwF with memory(LwF.M) as comparison methods.

3. Method

By the definition of continuous learning, we have a series of tasks to learn. To distinguish them from tasks in task-incremental learning, we refer to avalanche Lomonaco et al. (2021) using the term "experience", and we denote these continuous experiences as $\mathcal{E} = \{E_1, E_2, \ldots, E_T\}$, T is the total number of continuous learning experiences. And for the image classification network f_θ, we usually divide it simply into a feature extractor g and a linear classifier h, $f_\theta = g \circ h$. The objective function of learning is:

$$\arg\min_\theta \sum_{i=1}^{T} \mathcal{L}_i = \arg\min_\theta \sum_{i=1}^{T} \mathbb{E}_{(\boldsymbol{x},y)\in E_i}\left[\ell\left(y, f_\theta\left(x\right)\right)\right] \tag{1}$$

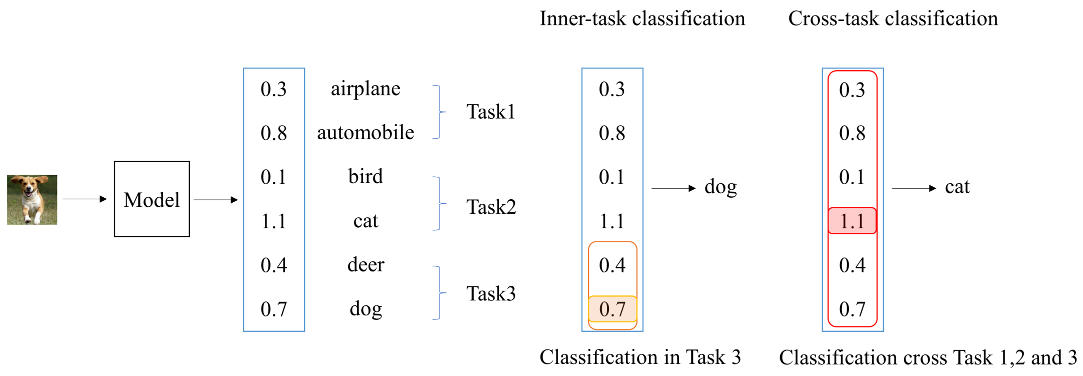


Figure 1: Inner-task classification and Cross-task classification.

Where ℓ is the loss function. If we have all the experiences data, we can do joint training and directly optimize the above loss function by setting the hyperparameters to get the best results.

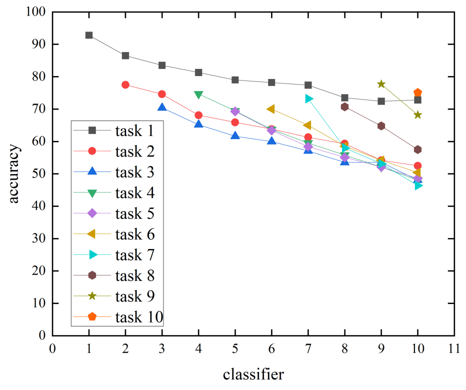
However, as we mentioned in the related work when defining the task, we acknowledged that we cannot obtain all the data. During training, we only have access to data for the current experience and a limited amount of past data stored in memory. Therefore we can only approximate the minimization of the loss function.

3.1. Definition of terms

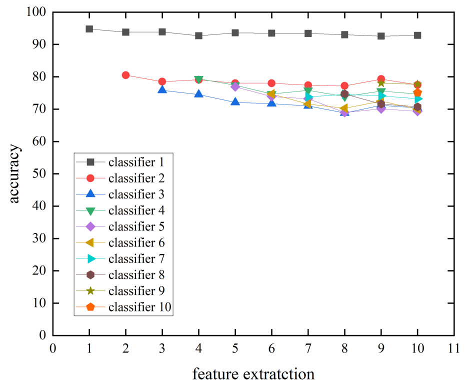
To facilitate the detail of our approach next, we define two terms. Assuming that the t^{th} task has arrived and the previous $t-1$ tasks have already been trained, each experience is divided into c classes. When inputting an image, the output of the current network is a vector of dimension $t*c$, and the subscripts of the $t*c$ floating-point numbers contained in this vector are denoted by 0 to $t*c-1$. The output is represented in a Python-like list format: $O[0:t*c]$. Define two terms below:
Inner-task classification: The training goal is to classify correctly in $O[(t-1)*c:t*c]$.
Cross-task classification: The training goal is to classify correctly in $O[0:t*c]$.
Fig. 1 depicts the two terms. As show in the figure, the result of Inner-task classification is the dog, while the prediction of Cross-task classification is the cat.

3.2. Classifier: Primary Cause of Forgetting

Based on the DER Buzzega et al. (2020) method, we did some experiments to demonstrate that the linear classifier forgot much more than the feature extractor after using the knowledge distillation method. We save the model at the end of training for each task in continuous learning.



(a)



(b)

Figure 2: (a) depicts the performance of the model on the test dataset with a different linear classifier using a fixed feature extractor. Where the i^{th} experience curve indicates the prediction accuracy of the i^{th} feature extractor and different linear classifiers on the test dataset of the i^{th} experience with a sharp drop. (b) depicts the performance of the model on the test dataset with a fixed linear classifier with different feature extractors. Where the i^{th} classifier curve represents the prediction accuracy of the i^{th} linear classifier and different feature extractors on the i^{th} experience test dataset with a flat decline.

As shown in Fig. 2(a)subfigure. Fixing the feature extractor parameters and comparing different linear classifier parameters for the same experience learning shows a significant decrease in accuracy on the same task.

As shown in Fig. 2(b)subfigure. We again fixed the linear classifier parameters and compared different feature extractor parameters for the same experience learning and found a flat decrease in accuracy on the same task.

Thus we reasonably suspect that the linear classifier is more forgetful during continuous learning and we need to pay extra attention to the learning of the linear classifier parameters during training. In the Wu et al. (2019) there was also mentioned the existence of bias in the fully connected layer, it was done by adding Bias Correction Layer, but our approach does not add additional network structure.

3.3. Classifier Expander(CE) Method

The model structure of the network is shown in Fig. 3. We divide the model into two parts: feature extractor and linear classifier. The parameters of feature extraction are the task-sharing parameters represented by θ_s, the parameters of the old task linear classifier are represented by θ_o, and the parameters of the new task linear classifier are represented by θ_n.

As mentioned in the Introduction, our approach is a two-stage method. The CE model training process is shown in Fig. 4. In the first stage, our training process was divided into two parts. In the first part, we employ a knowledge distillation approach which utilizes

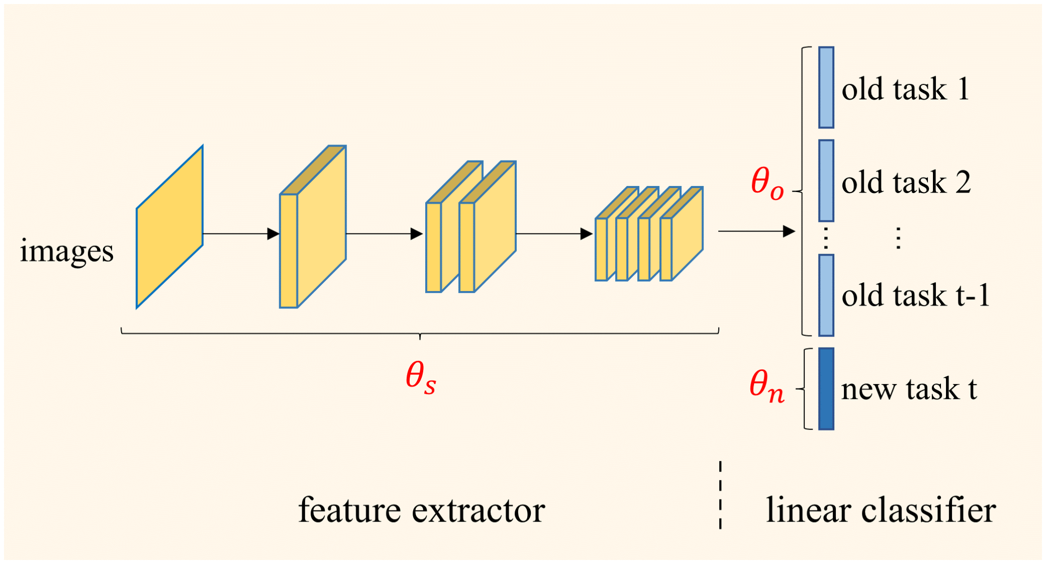


Figure 3: Classifier Expander(CE) Model Architecture.

data retained in memory for the training of the network parameters(θ_s, θ_o) to preserve the performance of previously learned tasks. In the second part, the model train the network parameters(θ_s, θ_n) to learn the new task by using the dataset of the new task. At this point, the new task is trained using the Inner-task classification approach. After training, we add the new task dataset to memory and adjust the amount of data from the old task to keep the total amount of storage in memory constant. In the second stage, we use the data in updated memory to train the network(θ_o, θ_n) to improve the overall prediction accuracy. At this point, the new task is trained using the Cross-task classification method. So we are referred to this linear classifier training range expansion method as Classifier Expander(CE).

The algorithm flow is as follows Algo. 1. As the algorithm demonstrates: we now train the t^{th} task called new task E_t which data is $\mathcal{D}^t$, and the data of 1$\sim$ t-1 tasks will be saved in the memory $\mathcal{M}$. The size of the memory is K, and the number of data of each task stored in memory is consistent $j = K/t$. We use two forms(data perspective and task perspective) to represent $\mathcal{M}$ to facilitate the understanding of the algorithm. ℓ is the cross-entropy loss function, and ℓ' is the MSE loss function. λ_α, λ_β, and λ_{ftc} are hyperparameters, which are set in the experimental section of Section 4.

4. Experiment

4.1. Evaluation Method

Some continuous learning metrics are proposed in Lopez-Paz and Ranzato (2017); Chaudhry et al. (2018). We choose Final Average Accuracy(FAA) and Final Forgetting(FF) Boschini et al. (2022) as the metrics of our experiments.

Algorithm 1 Classifier Expander(CE)

```
Input: The index of the current task: t.
    The dataset of the current task: $\mathcal{D}^t = \left\{\left(\boldsymbol{x_n^t}, y_n^t\right) | 1 \leq n \leq r\right\}$(data perspective).
    // $r$ represents the total amount of data.
    The memory size: K.
    The t-1 task model: $F^{t-1} = (\theta_s^{t-1}, \theta_o^{t-1})$.
    The dataset of the memory: $\mathcal{M} = \left\{\mathcal{M}^1, \mathcal{M}^2, \ldots, \mathcal{M}^{t-1}\right\}$(task perspective),
    $\mathcal{M}^i = \left\{\left(\boldsymbol{x_1^i}, y_1^i\right), \left(\boldsymbol{x_2^i}, y_2^i\right), \ldots, \left(\boldsymbol{x_j^i}, y_j^i\right)\right\}, 1 \leq i \leq t-1, j = K/(t-1)$(data perspective),
    $\mathcal{M} = \{(\boldsymbol{x_m}, y_m) | 1 \leq m \leq K\}$(data perspective).
Output: The trained model $F^{t*}$.
1: $F^t \leftarrow (F^{t-1}, \theta_n^t)$;
   // First stage
2: $(\theta_s^{t*}, \theta_o^{t*}, \theta_n^{t*}) \leftarrow \arg\min_{\theta_s^t, \theta_o^t, \theta_n^t} \Big(\ell\left(F^t\left(\boldsymbol{x_n^t}\right), y_n^t\right)$
   $+ \lambda_\alpha \cdot \ell\left(F^t\left(\boldsymbol{x_m}\right), y_m\right) + \lambda_\beta \cdot \ell'\left(F^t\left(\boldsymbol{x_m}\right), F^{t-1}\left(\boldsymbol{x_m}\right)\right)\Big)$;
3: $j' \leftarrow K/t$;
4: for $i = 1 \rightarrow i = t-1$ do
5:     $\mathcal{M}^{i'} \leftarrow RANDOM\left(\mathcal{M}^i\right)[: j']$; // Shuffle and select the first $j'$ items.
6: end for
7: $\mathcal{M}^t \leftarrow RANDOM\left(\mathcal{D}^t\right)[: j']$; // Shuffle and select the first $j'$ items.
8: $\mathcal{M}' \leftarrow \left\{\mathcal{M}^{1'}, \mathcal{M}^{2'}, \ldots, \mathcal{M}^{t-1'}, \mathcal{M}^t\right\}$; // $\mathcal{M}' = \{(\boldsymbol{x'_m}, y'_m) | 1 \leq m \leq K\}$.
   // Second stage
9: $\left(\theta_o^{t**}, \theta_n^{t**}\right) \leftarrow \arg\min_{\theta_o^{t*}, \theta_n^{t*}} \left(\lambda_{ftc} \cdot \ell\left(F^t\left(\boldsymbol{x'_m}\right), y'_m\right)\right)$;
10: $F^{t*} \leftarrow \left(\theta_s^{t*}, \theta_o^{t**}, \theta_n^{t**}\right)$.
```

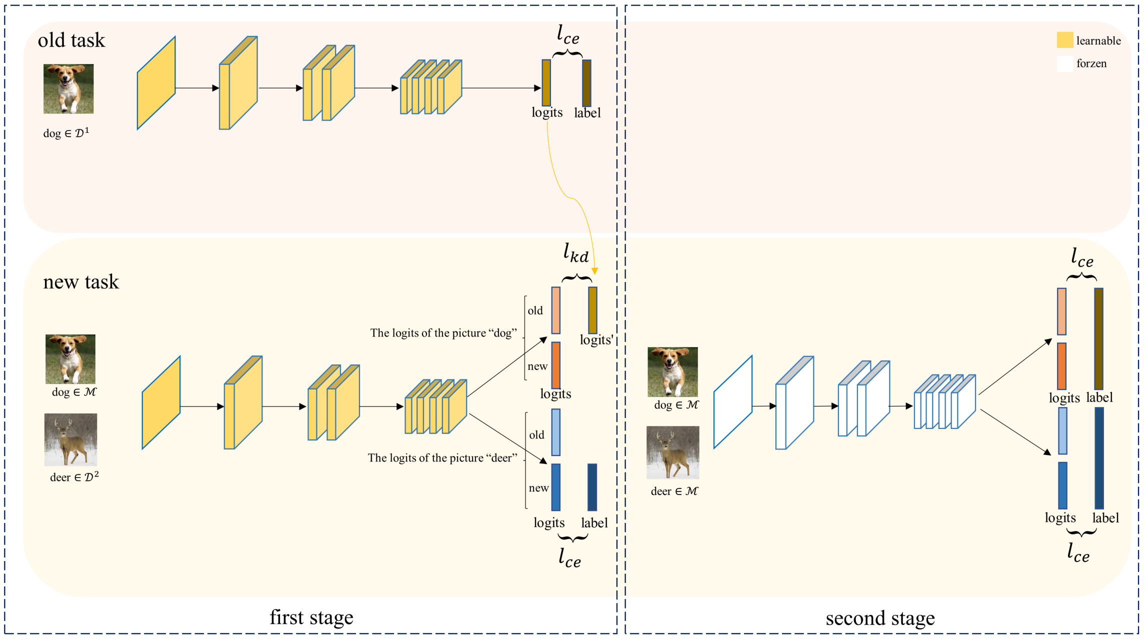


Figure 4: Classifier Expander(CE) Model Training Process.

$$FAA = \frac{1}{T}\sum_{i=1}^{T} a_i^t \tag{2}$$

$$FF = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{t\in 1,2,\ldots,T-1} a_i^t - a_i^{T-1} \tag{3}$$

where a_i^t denotes the accuracy for experience E_i after training on the t^{th} experience.

4.2. Experimental Setting

4.2.1. Dataset

The dataset is described in Tab. 1.

Table 1: The number of training and test samples, the total number of classes, and the split of the dataset.

Dataset	Training/Test Samples	Total Classes	Num Of Experiences	Classes Per Experience	Experiment Name
CIFAR-10	$50,000/10,000$	10	5	2	Split CIFAR-10
CIFAR-100	$50,000/10,000$	100	10	10	Split CIFAR-100
CUB-200	$5,994/5,794$	200	10	20	Split CUB-200

4.2.2. Model And Parameters Setting

Details of the experiment are in Tab. 2. We introduce the backbone model and the weights of the pre-trained model used for each experiment. We train ResNet-18 for CIFAR-10 and CIFA-100 trials with a low number of classes and ResNet-50 for CUB-200 experiments with a big number of classes. For the selection of the pre-trained model, we refer to Boschini et al. (2022). We choose a dataset with much more classes than the current experiment for pre-trained, and it is important to note that we use the checkpoint supplied in Boschini et al. (2022). In the first stage, we use the number of batch size data from the new experience dataset and memory. In the second stage, we only use the data in the updated memory, we use the num of batch size data for training.

Table 2: The model used in the experiments, the corresponding pre-trained model, and the experimental parameter settings under different buffer sizes.

Experiment Name	**Backbone Model**	**Pre-trained Model**	**Buffer Size**	$lr/epoch/batch\ size$	$\lambda_\alpha/\lambda_\beta/\lambda_{ftc}$
Split CIFAR-10	ResNet18	CIFAR-100	500	0.05/50/32	0.4/2.5/0.1
			5120	0.08/50/32	0.4/2/0.1
Split CIFAR-100	ResNet18	Tiny ImageNet	500	0.04/50/32	0.9/1/0.1
			2000	0.05/50/32	1/1.2/0.1
Split CUB-200	ResNet50	ImageNet	400	0.02/50/32	5/2/1
			1000	0.04/50/32	2/1/0.5

4.2.3. Compared Methods

We compare CE with the following scenarios:

1. *Joint*. Train the network using all the data at once, with the accuracy result as an upper limit.
2. *Finetune*. Train the network using only new task data and a tiny learning rate at a time without any restrictions.
3. *oEWC* Kirkpatrick et al. (2017), using online estimation of Fisher information matrix to penalize modifications to important parameters.
4. *LwF* Li and Hoiem (2017), a distillation method using only new task data.
5. *LwF.M*, LwF method that uses memory to retain old task data.
6. *ER* Robins (1995), the method that mixes training with in-memory data sampling and current task data.
7. CO^2L Cha et al. (2021), a continuous learning method that introduces contrast learning.
8. *iCaRL* Rebuffi et al. (2017), which uses episodic memory to prevent catastrophic forgetting.
9. $DER++$ Buzzega et al. (2020), a method for training using self-distillation on in-memory data.
10. *ER_ACE* Caccia et al. (2022), cross-entropy loss method for separating memory and current task data.
11. *TWF* Boschini et al. (2022), a method to improve continuous learning performance based on migrating knowledge from pre-trained networks

4.3. Results

As shown in Table. 3, the FF of our method is the lowest on the Split CIFAR-10 experiment, making full use of the data in memory to mitigate catastrophic forgetting. The FAA of our method is second only to the TwF method and higher than the other CL methods. This is because the advantage of classifier expansion is not obvious when the number of classes is small. On the continuous learning task with a small number of classes, both TwF and our method are close to the results of joint training, and the difference in our results is not significant.

Table 3: The results on the Split CIFAR-10 dataset, including both class incremental learning and task incremental learning for continuous learning, using Final Average Accuracy(FAA) and Final Forgetting(FF) as metrics. The pre-trained experiment is training on CIFAR-100.

FAA(FF)	**Split CIFAR-10** (pretr. CIFAR-100)			
Method	**Class-IL**		**Task-IL**	
Joint(UB)	92.89(-)		98.38(-)	
Finetune	19.76(98.11)		81.84(19.50)	
oEwc Kirkpatrick et al. (2017)	26.10(88.85)		81.84(19.50)	
LwF Li and Hoiem (2017)	19.80(97.96)		86.41(14.35)	
Buffer Size	500	5120	500	5120
LwF.M	26.08(90.03)	29.18(85.76)	92.46(7.10)	94.32(4.36)
ER Robins (1995)	67.24(38.24)	86.27(13.68)	96.27(2.23)	97.89(0.55)
CO^{2}L Cha et al. (2021)	75.47(21.80)	87.59(9.61)	96.77(1.23)	97.82(0.53)
iCaRL Rebuffi et al. (2017)	76.73(14.70)	77.95(12.90)	97.25(0.74)	97.52(0.15)
DER++ Buzzega et al. (2020)	78.42(20.18)	87.88(8.02)	94.25(4.46)	96.42(1.99)
ER-ACE Caccia et al. (2022)	77.83(10.63)	86.20(5.58)	96.41(2.11)	97.60(0.66)
TwF Boschini et al. (2022)	**83.65(11.59)**	**89.55(6.85)**	97.49(0.86)	**98.35(0.17)**
CE(Ours)	83.61(9.89)	88.22(5.99)	**97.81(0.31)**	98.08(0.09)

As shown in Tab. 4 and Tab. 5, the advantages of our method come to the fore on datasets with a high number of classification categories. In Split CIFAR-100 and Split CUB-200 experiments, the FF and FAA of our method outperform other CL methods, achieving an average improvement of 2.94% on the class increment task and 1.16% on the task increment task compared to the current best method. The experiments demonstrate that our method can alleviate forgetting of old tasks and can learn knowledge from new tasks, thus effectively improving the accuracy of continuous learning tasks.

4.4. Ablation studies

We perform ablation studies to verify the validity of each part of our model. As shown in Tab. 6, we denote the loss of distillation training in the first stage by $\mathcal{L}_{KD}$ and the loss of the CE of the second stage by $\mathcal{L}_{FTC}$. By comparing the first and second rows or the third and fourth rows it can be seen that using the CE method can effectively improve the accuracy of the model. By comparing the first and third rows or the third and fourth rows it can be seen that the distillation method is the most influential factor in performance improvement.

Table 4: The results on the Split CIFAR-100 dataset, including both class incremental learning and task incremental learning for continuous learning, using Final Average Accuracy(FAA) and Final Forgetting(FF) as metrics. The pre-trained experiment is training on Tiny ImageNet.

FAA(FF)	**Split CIFAR-100** (pretr. Tiny ImageNet)			
Method	**Class-IL**		**Task-IL**	
Joint(UB)	75.20(-)		93.40(-)	
Finetune	09.52(92.31)		73.50(20.53)	
oEwc Kirkpatrick et al. (2017)	10.95(81.71)		65.56(21.33)	
LwF Li and Hoiem (2017)	10.83(90.87)		86.17(4.77)	
Buffer Size	500	2000	500	2000
LwF.M	13.23(88.21)	14.37(86.57)	76.52(18.03)	80.56(13.61)
ER Robins (1995)	31.30(65.40)	46.80(46.95)	85.95(6.14)	87.59(4.85)
CO^{2}L Cha et al. (2021)	33.40(45.21)	50.95(31.20)	68.51(21.51)	82.96(8.53)
iCaRL Rebuffi et al. (2017)	56.00(19.27)	58.10(16.89)	89.99(2.32)	90.75(1.68)
DER++ Buzzega et al. (2020)	43.65(48.72)	58.05(29.65)	73.86(20.08)	86.63(6.86)
ER-ACE Caccia et al. (2022)	53.38(21.63)	57.73(17.12)	87.21(3.33)	88.46(2.46)
TwF Boschini et al. (2022)	56.83(23.89)	64.46(15.23)	89.82(3.06)	91.11(2.24)
CE(Ours)	**59.88(15.91)**	**65.12(15.87)**	**91.13(1.48)**	**91.77(0.89)**

Table 5: The results on the Split CUB-200 dataset, including both class incremental learning and task incremental learning for continuous learning, using Final Average Accuracy(FAA) and Final Forgetting(FF) as metrics. The pre-trained experiment is training on ImageNet.

FAA(FF)	**Split CUB-200** (pretr. ImageNet)			
Method	**Class-IL**		**Task-IL**	
Joint(UB)	78.54(-)		86.48(-)	
Finetune	8.56(82.38)		36.84(50.95)	
oEwc Kirkpatrick et al. (2017)	8.20(71.46)		33.94(40.36)	
LwF Li and Hoiem (2017)	8.59(82.14)		22.17(67.08)	
Buffer Size	400	1000	400	1000
LwF.M	11.50(72.72)	13.79(71.70)	67.29(11.35)	72.23(7.72)
ER Robins (1995)	45.82(40.76)	59.88(25.65)	75.26(9.82)	80.19(4.52)
CO^{2}L Cha et al. (2021)	8.96(32.04)	16.53(20.99)	22.91(26.42)	35.79(16.61)
iCaRL Rebuffi et al. (2017)	46.55(12.48)	49.07(11.24)	68.90(3.14)	70.57(3.03)
DER++ Buzzega et al. (2020)	56.38(26.59)	67.35(13.47)	77.16(7.74)	82.00(3.25)
ER-ACE Caccia et al. (2022)	48.18(25.79)	58.19(16.56)	74.34(9.78)	78.27(6.09)
TwF Boschini et al. (2022)	57.78(18.32)	68.32(6.74)	79.35(5.77)	82.81(2.14)
CE(Ours)	**63.62(13.36)**	**70.53(6.62)**	**80.85(1.77)**	**83.97(1.43)**

5. Conclusion

We propose a CE method that effectively mitigates the problem of catastrophic forgetting in continuous learning. Experiments show that our method outperforms other CL methods

Table 6: The results of the ablation experiment, $w/o/buf.$ indicates that Memory is not used, ✓indicates that this loss is used, and ✗indicates that this loss is not used. There are two data for each term, the result of the class-incremental learning and the task-incremental learning are shown on the left/right.

$\mathcal{L}_{KD}$	$\mathcal{L}_{FTC}$	**Split CIFAR-10**			**Split CIFAR-100**			**Split CUB-200**		
Buffer Size		w/o/buf.	500	5120	w/o/buf.	500	2000	w/o/buf.	400	1000
✓	✓	-	83.61/97.81	88.22/98.08	-	59.88/91.13	65.12/91.77	-	63.62/80.85	70.53/83.97
✓	✗	-	76.15/97.79	75.48/96.96	-	51.33/91.12	55.79/92.09	-	55.79/80.77	61.66/83.31
✗	✓	-	59.35/93.65	73.54/96.19	-	37.69/79.24	44.38/81.82	-	34.04/60.55	39.53/65.95
✗	✗	53.14/93.37	-	-	25.33/76.64	-	-	14.60/5.31	-	-

in tasks with a large number of continuous learning classes. However, in tasks with a small number of classes, the advantage of our CE method is no longer obvious. In Section 3.1, we mentioned that linear classifiers are the key to forgetting, and we use retraining to overcome forgetting, which can be subsequently considered using a new classifier network to consider solving this problem. Lesort (2020)

References

Benedikt Bagus and Alexander Gepperth. An investigation of replay-based approaches for continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2021.

Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*, pages 692–709. Springer, 2022.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *arXiv preprint arXiv:2203.03798*, 2022.

Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 9516–9525, 2021.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Timothée Lesort. Continual learning: Tackling catastrophic forgetting in deep neural networks with replay processes. *arXiv preprint arXiv:2007.00487*, 2020.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas Tolias, Simone Scardapane, Luca Antiga, Subutai Amhad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: an end-to-end library for continual learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2nd Continual Learning in Computer Vision Workshop, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020.