NeCa: Network Calibration for Class Incremental Learning

Zhenyao Zhang and Lijun Zhang(✉)

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{zhangzhenyao,zhanglj}@lamda.nju.edu.cn

Abstract. Class incremental learning (CIL) aims to continually learn unseen classes in new tasks without forgetting the previous ones. However, deep neural networks are prone to make a biased prediction towards classes in the most recently learned task, dubbed task-recency bias. Most recent studies make a post-training adjustment on the last fully connected layer to alleviate this problem but ignore the feature extractor. This work proposes a novel training framework termed network calibration (NeCa) that simultaneously adjusts the last fully connected layer and the feature extractor. Specifically, we combine the post-training adjustment process with the training process into a balanced learning module, whose loss function is corrected based on the prior probabilities of classes. In this module, the parameters of the whole network are well-calibrated via backpropagation. Additional knowledge transmission and decaying regularization modules further mitigate catastrophic forgetting in CIL. Experiment results manifest that NeCa outperforms the state-of-the-art methods on three mainstream datasets, including MNIST, CIFAR-100, and ImageNet-100, which validates the effectiveness of our framework. Furthermore, we conduct experiments with the prevalent vision transformer backbone, and the consistently excellent performance demonstrates that NeCa is also competently suited for attention-based models.

Keywords: Class incremental learning · Catastrophic forgetting · Class imbalance · Classification · Deep learning

1 Introduction

In recent years, deep neural networks (DNNs) have achieved excellent performance on various visual tasks [9,10]. However, their success is generally limited to a single task. Compared with humans, DNNs seem to be tremendously forgetful. Specifically, when a new task arrives, DNNs are prone to forget the knowledge gained on the old task, which is known academically as catastrophic forgetting [4,22]. Nowadays, the increasing volume of data and privacy issues create an urgent need for a learning paradigm to continually adapt to new tasks while maintaining performance on old ones.

To delve into catastrophic forgetting, researchers have proposed three incremental learning scenarios: task-incremental learning, domain-incremental learning, and class-incremental learning (CIL) [27]. They differ in whether the task identities are provided at inference and whether the task identities must be inferred. CIL is the most difficult of the three because it has to predict unknown task identities. Researchers widely adopt a size-limited exemplar memory [25] and knowledge distillation technique [19] for CIL. With their help, the overall accuracy does improve, but predictions are still extremely biased towards the classes in the most recently learned task. The limitation is termed task-recency bias (TRB) [20], illustrated in Fig. 1.

Recently, several works aim to correct TRB, most of which argue that an imbalanced training set is the main reason and the last fully connected layer suffers the most. So they perform a post-training adjustment on the last fully connected layer, which does improve the performance to some extent [25,30,34]. However, their methods leave the feature extractor, which is also trained on the same imbalanced dataset, untouched. To improve this, we present a simple and effective training framework named network calibration (NeCa) that mitigates TRB by simultaneously tuning the last fully connected layer and the feature extractor in a one-stage training.

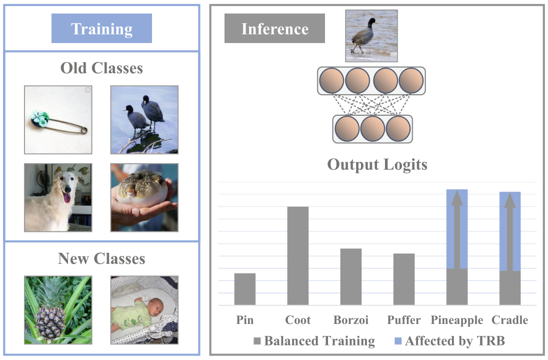

Fig. 1. Compared to the model trained in the balanced dataset, the TRB-affected model produced extremely high output logits for the new classes, leading to inference errors.

In total, NeCa is composed of three parts. **Balanced Learning**: We confront an imbalanced training set in each task because of the size limitation of exemplar memory. To address this issue, we calibrate the traditional classification loss via the prior probability of each class and propose the calibrated cross-entropy (CCE) loss. Through minimizing CCE loss, we calibrate the whole network (including the feature extractor ignored by previous work) to obtain a more balanced classification boundary. **Knowledge Transmission**: To transfer the knowledge along with the task, we apply knowledge distillation (KD) [12] between the models of two adjacent tasks. We restrict the output logits of the models to be as consistent as possible. This module can pass on the performance gains from other modules. **Decaying Regularization**: As the training set starts from a small part of the whole dataset, we heuristically adopt a stronger regularization at the beginning to prevent overfitting and weaken it as the task progresses to combat forgetting.

To sum up, our contribution is threefold: (i) Taking the information of prior probability into account, we propose CCE loss to confront the imbalanced training set in CIL. Optimizing CCE loss will calibrate the whole network especially the feature extractor neglected in previous works. (ii) We propose a novel training framework called NeCa consisting of three parts. They can promote each other very well and make the model's prediction more balanced throughout the task continuum. (iii) Extensive experiment results manifest that, compared with the current state-of-the-art methods, our framework achieves the best performance on three mainstream datasets, including MNIST, CIFAR-100, and ImageNet-100. Furthermore, experiments on vision transformers demonstrate that NeCa is equally applicable to attention-based models.

2 Related Work

Class incremental learning (CIL) approaches are roughly classified into the following four categories: (i) Structure-based methods fix the parameters related to previous tasks and assign more to new tasks [1,24,31]. (ii) Regularization-based methods try to minimize the impact of learning a new task imposing on the old ones [2,3,15,19]. (iii) Rehearsal methods provide training samples of previous tasks by storing a small number of exemplars or generating synthetic images or features [5,25,26,30]. (iv) Bias-correction methods aim to alleviate task-recency bias (TRB) [5,13,25,30,34]. Our framework network calibration (NeCa) belongs to (ii), (iii), and (iv) at the same time, and the following is a brief overview of these three categories.

Knowledge Distillation. Knowledge distillation (KD) [12] belongs to the broad category of regularization-based methods. Li and Hoiem [19] first utilize KD in incremental learning to keep the model's response for old tasks. After this, KD became a popular technique used to retain the knowledge of old tasks in CIL. Rebuffi *et al.* [25] store a limited number of previous exemplars for training and expand the KD loss for them. Wu *et al.* [30] introduce a scalar to adjust the weight between KD loss for remembering the previous knowledge and classification loss for learning the new ones. Ahn *et al.* [2] impose KD independently on the output logits associated with each task and propose a novel separated softmax layer to implement it.

Rehearsal. CIL can proceed without preserving any examples from the previous task [3,14,19,33]. In contrast, rehearsal-based methods use a limited amount of previously seen exemplars or learn to generate them [26]. As far as we know, Rebuffi *et al.* [25] first introduce the exemplar rehearsal method into CIL with a nearest-means-of-exemplars classification strategy at inference. Due to its performance improvement and ease of deployment, rehearsal is prevalent in recent works [2,13,18,30,31,34]. Prabhu *et al.* [23] keep a balanced exemplar memory by greedily collecting samples over the previous tasks, and they train a model from scratch in the next task using samples only in the memory. Verwimp *et al.* [28] provide both conceptual and strong empirical evidence to interpret the benefits and harms of the rehearsal-based method in the CIL scenario.

Bias Correction. The bias in the bias correction refers to the TRB. Despite resorting to sorts of techniques to prevent forgetting, the model tends to make a biased prediction towards the newest classes. To correct TRB, Rebuffi *et al.* [25] observe the last fully connected layer of a model manifests higher bias than the modules before it, *i.e.*, the feature extractor. So they abandon the fully connected layer and utilize only the feature extractor at inference. Castro *et al.* [5] take advantage of data augmentation and propose a second-stage training phase called balanced fine-tuning. Wu *et al.* [30] add a bias correction layer following the last fully connected layer and train it on a balanced dataset obtained by down-sampling while freezing the feature extractor. Hou *et al.* [13] utilize the hard negative classes of a new class to rectify the imbalanced class embeddings. Furthermore, they note that the weight and bias of the last fully connected layer are distorting between old and new classes. Inspired by this finding, Zhao *et al.* [34] propose a simple and effective method to adjust the weight of the last fully connected layer via its norms. Unlike previous approaches, our framework NeCa adjusts the parameters of the feature extractor and merges the bias correction stage into the training phase.

3 Method

In this section, we first give a formal formulation of class incremental learning (CIL) and then detail our proposed framework, namely network calibration (NeCa), which consists of three main components: balanced learning, knowledge transmission, and decaying regularization (illustrated in Fig. 2).

3.1 Problem Formulation

CIL is composed of T tasks coming in a row. In each task $t = 1, 2, \cdots, T$, we get a new training set $\mathcal{D}^t$ that consists of C^t classes we have never seen before, which implies that $\mathcal{L}^i \cap \mathcal{L}^j = \varnothing, \forall i \neq j$, where $\mathcal{L}^i$ denotes the label space of $\mathcal{D}^i$. Note that we need to expand the model parameters for the new classes on the last fully connected layer. It is easy to get a model that performs well on task t by simply adopting a traditional cross-entropy (CE) loss on $\mathcal{D}^t$:

$$L_{CE}(\mathcal{D}^t) = \frac{1}{|\mathcal{D}^t|} \sum_{(X,y)\in\mathcal{D}^t} \mathcal{L}^t_{CE}(X, y). \tag{1}$$

Let $C^t_{old} = \sum_{i=1}^{t-1} C^i$ and $C^t_{all} = \sum_{i=1}^{t} C^i$, we can express CE loss for a single item as

$$\mathcal{L}^t_{CE}(X, y) = \sum_{c=1}^{C^t_{all}} -\mathbb{1}_{\{c=y\}} \log u_c(X), \tag{2}$$

in which $\mathbb{1}_{\{c=y\}}$ is the indicator function and $u_c(\cdot)$ denotes the prediction probability for the c-th class.

Note that before task t, our model has seen C^t_{old} classes from the previous datasets $\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^{t-1}$. Only optimizing (1) will tremendously deteriorate the performance of the model on the previous tasks, which is termed catastrophic forgetting. To alleviate this phenomenon, we typically apply a size-limited exemplar memory $\mathcal{M}$ (size is limited to $|\mathcal{M}| < m$ and $m \ll \sum_{i=1}^{t} |\mathcal{D}^i|$) to randomly preserve $\lfloor m/C^t_{all} \rfloor$ items in each class after a task in a balanced manner and recall them at the next task. Thus, the CE loss at task t is adopted on both $\mathcal{D}^t$ and $\mathcal{M}$, *i.e.*, $L_{CE}(\mathcal{D}^t \cup \mathcal{M})$. However, the prior probability of each class in $\mathcal{D}^t \cup \mathcal{M}$ is extremely biased, and the CE loss does not apply to the situation anymore.

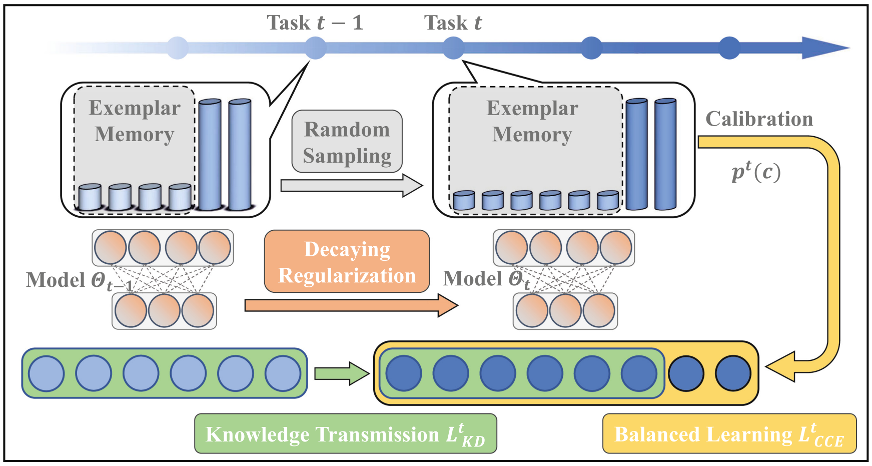

Fig. 2. An overview of our framework NeCa, containing balanced learning (the yellow part), knowledge transmission (the green part), and decaying regularization (the orange part) (Color figure online)

3.2 Balanced Learning

Biased prior probability leads to biased prediction, which is exactly the cause of task-recency bias (TRB). However, the distribution information of prior probability depicts the degree of imbalance of each class, which we can utilize to correct the imbalanced output logits for each class. We intend to construct a more balanced learning process by using prior probabilities.

Considering a general situation, at task t, our training set contains C^t_{all} different classes. Suppose the number of samples from c-th class is $n(c)$, the prior probability of c-th class can be estimated by

$$p^t(c) = \frac{n(c)}{\sum_{i=1}^{C^t_{all}} n(i)}. \tag{3}$$

Then we demonstrate how this information corrects the imbalanced prediction probability. As the main reason for TRB, while training with an imbalanced dataset, the prediction probabilities $u(\cdot)$ of the dominant classes (*i.e.*, the new classes in class incremental learning) are prone to be unexpectedly high. Inspired by recent approaches to class imbalance [21,32], we address this problem by correcting the prediction probability of c-th class $u_c(\cdot)$ with the prior probability of c-th class $p^t(c)$ at inference:

$$v_c(\cdot) = \frac{u_c(\cdot)}{p^t(c)}, \tag{4}$$

applying normalization on $v_c(\cdot)$, we get the calibrated prediction probability as

$$\hat{u}_c(\cdot) = \frac{v_c(\cdot)}{\sum_{i=1}^{C^t_{all}} v_i(\cdot)}. \tag{5}$$

Through the above post-training calibration, the prediction probabilities $\hat{u}_c(\cdot)$ become more balanced and result in a considerable overall accuracy. But post-training adjustment optimizes no parameter of the model. To complement this, we introduce the calibration into the CE loss (2).

As for post-training calibration at inference, we turn it into a pre-training scaling imposed on the prediction probabilities $u_c(\cdot)$. The pre-training scaling can be considered as the inverse process of the post-training calibration (4):

$$v'_c(\cdot) = p^t(c) \cdot u_c(\cdot), \tag{6}$$

after the same normalization in (5), we get $\hat{u}'_c(\cdot)$. The calibrated cross-entropy (CCE) loss can be obtained by replacing prediction probabilities $u_c(\cdot)$ in the CE loss (2) with $\hat{u}'_c(\cdot)$:

$$\mathcal{L}^t_{CCE}(X, y) = \sum_{c=1}^{C^t_{all}} -\mathbb{1}_{\{c=y\}} \log \hat{u}'_c(X). \tag{7}$$

Combining the above Eqs. (5), (6), and (7), we obtain the complete formulation of CCE loss:

$$\mathcal{L}^t_{CCE}(X, y) = \sum_{c=1}^{C^t_{all}} -\mathbb{1}_{\{c=y\}} \log \frac{e^{o_c(X)+\mu \log p^t(c)}}{\sum_{i=1}^{C^t_{all}} e^{o_i(X)+\mu \log p^t(i)}}, \tag{8}$$

where $o_i(\cdot)$ denotes the i-th class output of the current model and μ, like temperature in the knowledge distillation (KD) technique, is an additional hyperparameter controlling the smoothness magnitude. When $\mu = 0$, it degenerates as the traditional CE loss, and the neural network will not put additional attention on the old classes. When $\mu = 1$, (8) is equivalent to (7). As μ increases, the model begins to focus more on the old classes.

At inference, we use the prediction probabilities $u_c(\cdot)$ without additional operation. As shown in Table 1, the pre-training scaling in (6) shares the same effect as the post-training calibration in (4), *i.e.*, it produces a balanced classification boundary. However, unlike the post-training calibration leaves all the parameters untouched, optimizing CCE loss will influence each parameter positively (including the last fully connected layer and the feature extractor). When the model becomes a teacher model of the knowledge transmission module in Sect. 3.3, its output logits will be more balanced and teach more useful knowledge than a sub-balanced one.

Table 1. The relation between the post-training calibration and the pre-training scaling

Method	Training w/	Inference w/
Post-training calibration	$u_c(\cdot)$	$u_c(\cdot)/p^t(c)$
Pre-training scaling	$p^t(c) \cdot u_c(\cdot)$	$u_c(\cdot)$

3.3 Knowledge Transmission

To transmit the knowledge, we preserve the teacher model Θ_{t-1} at the end of task $(t-1)$, then utilize KD to train the model Θ_t at task t. Specifically, the KD loss can be formulated as

$$L_{KD}(\mathcal{D}^t \cup \mathcal{M}) = \frac{1}{|\mathcal{D}^t \cup \mathcal{M}|} \sum_{(X,y)\in\mathcal{D}^t\cup\mathcal{M}} \mathcal{L}_{KD}(X), \tag{9}$$

in which KD loss for a single item $\mathcal{L}_{KD}(\cdot)$ is defined as

$$\mathcal{L}_{KD}(X) = \sum_{c=1}^{C^t_{old}} -\hat{q}_c(X)\log q_c(X), \tag{10}$$
$$\hat{q}_c(X) = \frac{e^{\frac{\hat{o}_c(X)}{\tau}}}{\sum_{i=1}^{C^t_{old}} e^{\frac{\hat{o}_i(X)}{\tau}}}, \quad q_c(X) = \frac{e^{\frac{o_c(X)}{\tau}}}{\sum_{i=1}^{C^t_{old}} e^{\frac{o_i(X)}{\tau}}},$$

where $\hat{o}_i(\cdot)$ denote the i-th class output of the previous model Θ_{t-1}, and τ is the temperature controlling the smoothness magnitude of the target distribution during KD.

Finally, we combine (8) with (10) as Wu *et al.* [30] did, and the overall loss on task t becomes

$$L^t = (1-\lambda^t)L_{CCE}(\mathcal{D}^t \cup \mathcal{M}) + \lambda^t L_{KD}(\mathcal{D}^t \cup \mathcal{M}), \tag{11}$$

in which $\lambda^t = C^t_{old}/C^t_{all}$ is a hyperparameter of the trade-off between the balanced learning loss and the knowledge transmission loss.

Note that every parameter in our teacher model Θ_{t-1} has been calibrated by optimizing CCE loss at the previous task, thus the student Θ_t will learn a more balanced classification boundary, which will also promote a more balanced training in the current task.

3.4 Decaying Regularization

At the practical training phase, we often add a regularization term $\gamma^t\|\Theta_t\|_2$ ($\|\Theta_t\|_2$ denotes the 2-norm sum of each weight and bias in the model Θ_t) outside the overall loss (11). This term drives all model parameters toward zero to prevent overfitting.

As we should impose a stronger regularization on the parameters when trained on a smaller dataset, we replace the fixed regularization in previous works with a decaying way. At the beginning of a class incremental learning problem, we always fit the model on a tiny dataset, which is a small part of the complete dataset. A stronger regularization helps prevent overfitting. After that, overly strong regularization tends to make the parameters vanish, thus we weaken the regularization to protect the parameters fitted in previous tasks. Specifically, we formulate the decaying regularization parameter as

$$\gamma^t = \frac{1}{t}\cdot\gamma^1, \tag{12}$$

in which γ^1 is the regularization parameter of the first task, relying on the dataset and the total number of tasks T. In addition, we have tried linear decaying regularization

$$\gamma^t = \left(1 - \frac{t-1}{T}\right) \cdot \gamma^1, \tag{13}$$

but it is not as good as the reciprocal decaying in (12).

Algorithm 1.. Network Calibration

```
Input: Datasets $\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^T$
Output: Classification models $\Theta_1, \Theta_2, \cdots, \Theta_T$
Initialize: Exemplar memory $\mathcal{M} = \varnothing$, decaying regularization factor $\gamma^1$
Parameter: Random initialized model $\Theta_0$
 1: for $t = 1, \cdots, T$ do
 2:     Input the dataset $\mathcal{D}^t$ for task $t$
 3:     if $t = 1$ then
 4:         $\Theta_t \leftarrow$ Optimize $\Theta_{t-1}$ with $L_{CE}(\mathcal{D}^1) + \gamma^1 \|\Theta_{t-1}\|_2$
 5:     else
 6:         Update decay regularization factor $\gamma^t$ by (12)
 7:         Calculate $\lambda^t$ and estimate prior probabilities $p^t(\cdot)$ by (3)
 8:         $\Theta_t \leftarrow$ Optimize $\Theta_{t-1}$ with $(1-\lambda^t)L_{CCE}(\mathcal{D}^t \cup \mathcal{M}) + \lambda^t L_{KD}(\mathcal{D}^t \cup \mathcal{M}) + \gamma^t \|\Theta_{t-1}\|_2$
 9:     end if
10:     $\mathcal{M} \leftarrow$ Sample averagely and randomly from $\mathcal{D}^t \cup \mathcal{M}$
11:     Output a classification model $\Theta_t$ for task $t$
12: end for
```

Finally, summarizing the above three parts yields Algorithm 1. As can be seen, NeCa is easy to deploy because no additional training phase and complex data sampling method (*e.g.*, herding in iCaRL [25]) are needed.

4 Experiments

In this section, we evaluate our proposed framework network calibration (NeCa) in the class incremental learning (CIL) scenario and compare NeCa with some state-of-the-art methods. Moreover, we apply NeCa to both ResNet [10,11] and vision transformer (ViT) [8], to demonstrate its model-independent attribute.

4.1 Datasets, Baselines and Experimental Details

Our experiments involve three datasets with significantly diverse data volumes, including MNIST [17], CIFAR-100 [16], and a subset of ImageNet ILSVRC 2012 [7], *i.e.*, ImageNet-100. MNIST contains 60,000 bi-level images of handwritten digits 0 to 9 for training and 10,000 for testing. CIFAR-100 contains 50,000 RGB images at 32×32 resolution from 100 classes for training and 10,000 images for testing. ImageNet

ILSVRC 2012 is a larger dataset containing about 1.2 million images at different resolutions from 1,000 classes for training and 50,000 for evaluation. As for ImageNet-100, we randomly select 100 classes from the former.

We choose the following typical or state-of-the-art methods (also mentioned in Sect. 2) for comparison:

- **LwF** [19]: Retain the previous knowledge via knowledge distillation only.
- **iCaRL** [25]: Use an exemplar memory and predict using the outputs of the feature extractor in a nearest-means-of-exemplars way.
- **EEIL** [5]: Introduce an additional balanced training phase and impose stronger data augmentation.
- **BiC** [30]: Add additional modules, which require an independent training phase, to correct the task-recency bias (TRB) in the last fully connected layer.
- **WA** [34]: Make post-training adjustments, utilizing the information of the magnitude of its norms, only to the parameters of the last fully connected layer.
- **SS-IL** [2]: Propose a novel separate softmax module, which calculates loss independently for the output logits of each task, especially suitable for large-scale datasets.

For CIFAR-100, we only adopt random cropping, horizontal flipping, and normalization for data augmentation in all experiments. For all methods, we use a 32-layer ResNet [10, 11] and an SGD optimizer with a momentum of 0.9. We vary the total number of tasks as $T = \{2, 5, 10, 20\}$ with correspondence to batch size $= 32$, $m = 2000$, $\tau = 4$, $\gamma^1 = 0.00005 \cdot T$, and $\mu = 1.5$. The learning rate starts from 0.1 and reduces to 1/10 of the previous after 100, 150, and 200 epochs (250 epochs in total).

For MNIST, we only adopt normalization for data augmentation. All details are the same as in CIFAR-100 except $\gamma^1 = 0.00004 \cdot T$ and $\mu = 1$. The learning rate starts from 0.01 and reduces to 0.001 after the 5-th epoch (10 epochs in total).

For ImageNet-100, we use an 18-layer ResNet. Except for $\gamma^1 = 0.00001 \cdot T$, all other details are the same as in CIFAR-100. The learning rate starts from 0.1 and reduces to 1/10 of the previous after 30, 60, 80, and 90 epochs (100 epochs in total).

4.2 Results

On MNIST, we take the average of 5 experiments and report the accuracy of the last task A_T in Table 2 and the average accuracy across all tasks $\overline{A}$ in Table 3. The performance gaps among these methods are not very significant because the task is quite simple, but our method NeCa still achieves a slight lead. Note that, in Table 3 we use Joint to denote the method of saving all previous data for joint training, which is generally regarded as the upper bound indicator of performance that can be obtained in class incremental learning tasks.

On CIFAR-100, we take the average of 5 experiments and report the top-1 accuracy at each task in Fig. 3, the accuracy of the last task A_T in Table 2, and the average accuracy across all tasks $\overline{A}$ in Table 3. We denote the performance of jointly training all data as the upper bound. Among the methods of comparison, WA is the strongest baseline method. Compared with it, NeCa exceeds its accuracy at each task, and the

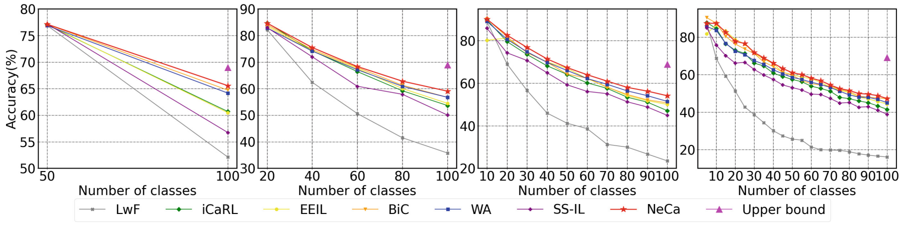

Fig. 3. CIL results on CIFAR-100 with a total number of 2, 5, 10, and 20 tasks respectively

average accuracy of all tasks increases by about 0.82, 1.71, 1.76, and 2.93 for 2-, 5-, 10-, and 20-step CIL scenarios, respectively. The improvement achieved by NeCa is more pronounced when the total number of tasks increases.

On ImageNet-100, we take the average of 3 experiments and report the top-1 and top-5 accuracy (denoted by 10 and 10* respectively in the T attribute) in Table 2 and Table 3. The results further indicate that NeCa still performs well on a larger dataset. As a method designed for large-scale datasets, SS-IL becomes the strongest baseline method. Compared with it, NeCa still surpasses about 4.12 and 0.96 on the average top-1 and top-5 accuracy of all tasks, respectively. Our framework especially improves a large margin on the top-1 accuracy.

Table 2. Last accuracy A_T (%) at different CIL settings

Model	ResNet-32						ResNet-18		ViT-M	ViT-B
Dataset	MNIST		CIFAR-100				ImageNet-100		CIFAR-100	
T	2	5	2	5	10	20	10	10*	10	
LwF	98.13	92.17	52.13	35.75	23.48	16.03	21.25	37.22	12.34	70.78
iCaRL	98.26	93.18	60.69	53.62	47.09	41.37	49.88	78.42	27.91	86.93
EEIL	98.37	96.61	60.47	54.42	50.02	44.28	52.74	80.02	41.80	84.89
BiC	98.62	96.89	64.71	56.69	50.75	46.96	58.88	81.96	45.40	86.37
WA	98.86	97.22	64.15	56.70	51.44	44.97	56.78	79.20	47.49	73.32
SS-IL	97.83	87.96	56.74	50.03	44.87	38.77	59.24	85.70	44.72	87.28
NeCa	**99.21**	**98.10**	**65.55**	**59.01**	**54.10**	**47.15**	**64.22**	**86.56**	**49.72**	**87.87**
Joint	99.67		68.93				82.34	95.21	67.52	96.46

4.3 Experiments on Vision Transformer

ViT [8] is attention-based, which separates the input image into several parts and calculates the potential correlation between them. ViT has the same prediction head (*i.e.*, the fully connected layer behind the [class] token) as ResNet does, thus can be rationally calibrated via NeCa. In this section, we transfer NeCa into the ViT model. Table 4 presents the details of our ViT models.

Table 3. Average accuracy $\overline{A}$ (%) at different CIL settings

Model	ResNet-32						ResNet-18		ViT-M	ViT-B
Dataset	MNIST		CIFAR-100				ImageNet-100		CIFAR-100	
T	2	5	2	5	10	20	10	10*	10	
LwF	98.98	96.05	64.49	54.47	45.19	32.84	49.26	64.75	36.47	83.40
iCaRL	99.08	97.65	68.83	67.62	64.46	59.21	69.78	90.10	46.79	91.89
EEIL	99.11	98.15	68.77	67.92	64.94	62.23	69.69	89.92	55.96	90.71
BiC	99.26	98.39	70.88	68.91	65.87	62.22	71.60	90.27	57.46	91.61
WA	99.39	98.79	70.50	68.31	66.40	60.58	71.16	89.72	59.85	87.32
SS-IL	98.88	92.52	66.94	64.72	61.12	55.16	71.04	91.96	56.58	91.94
NeCa	**99.57**	**99.21**	**71.32**	**70.02**	**68.16**	**63.51**	**75.16**	**92.92**	**60.83**	**93.12**

Table 4. Specific information about ViT-M and ViT-B

	Input size	Patch size	Depth	Head	Feature dimension	Pre-training status
ViT-M	32×32	2×2	6	8	512	From scratch
ViT-B	224×224	16×16	12	12	768	Pretrained on ImageNet-1000

On CIFAR-100, ViT-M is prone to overfit because it contains no convolutional operation. To alleviate this situation, we additionally apply RandAugment [6] in data augmentation. We train around 20000 steps for every task with an SGD optimizer and utilize the cosine annealing learning rate with the warmup technique. Besides, due to the prevalence of big data, it is urgent to investigate a continual learning method for strongly pre-trained models [29]. The ViT-B is a stronger model that has been pre-trained on ImageNet-1000 with a top-1 accuracy of around 0.81. On CIFAR-100, we apply ViT-B in the CIL setting. We train around 4000 steps for every task with an SGD optimizer and utilize the cosine annealing learning rate. We report their top-1 accuracy results in the two rightmost columns of Table 2 and Table 3. The results show that on the pre-trained ViT-B, the WA method has a very drastic performance degradation due to the hard modification of the classifier parameters, while NeCa achieves the best performance through adaptively calibrating the whole network, showing its generalizability and effectiveness.

4.4 Ablation Studies

To further analyze the role of each factor in NeCa, we consider some variants of our method. After evaluating them in a 10-step CIL scenario on CIFAR-100, we report the accuracy of the last task A_T, the average accuracy of all tasks $\overline{A}$, and forgetting rate $A_1 - A_T$ (the accuracy gap between the first and last task) in the three rightmost columns of Table 5, where KT and DR denote knowledge transmission and decaying regularization modules respectively. In V_3, CCE* indicates using post-training calibration mentioned in Sect. 3.2 rather than optimizing CCE loss. This method has poor performance because, like the comparison method in Sect. 4.1, it does not calibrate the relevant parameters of the model. Ablation studies show that every part of NeCa plays an active role and is indispensable. When used in combination, they can outperform state-of-the-art methods.

4.5 Analysis on Hyperparameter

In this section, we analyze the hyperparameter μ in the CCE loss (8). As discussed in Sect. 3.2, the CCE loss imposes greater emphasis on older classes when μ increases. However, an overhigh μ causes the model to ignore the new classes. It is a trade-off between the level of concern for the old and new classes, and we can set an optimal value for μ. Our experiment finds that the optimal μ is always in the range from 1 to 2 for every dataset. As shown in Table 6, we report the average accuracy across all tasks $\overline{A}$ when μ varies in [1,2]. Results demonstrate that our method is not very sensitive to the value of μ, which implies that our approach is easy to deploy across datasets without excessive hyperparameter tuning.

Table 5. Accuracy results of the variants

Variant	Loss	KT	DR	A_T (%)	$\overline{A}$ (%)	$A_1 - A_T$ (%)
V_1	CE	✓	×	$34.75_{\pm 1.25}$	$56.66_{\pm 1.08}$	51.35
V_2	CE	✓	✓	$34.04_{\pm 0.56}$	$58.43_{\pm 0.45}$	55.40
V_3	CCE*	✓	✓	$40.67_{\pm 0.66}$	$62.75_{\pm 0.36}$	50.03
V_4	CCE	×	✓	$43.83_{\pm 1.07}$	$62.52_{\pm 1.15}$	46.60
V_5	CCE	✓	×	$52.96_{\pm 0.41}$	$67.17_{\pm 0.75}$	36.58
NeCa	CCE	✓	✓	**54.10**$_{\pm 0.54}$	**68.16**$_{\pm 0.43}$	**36.00**

Table 6. Influence of μ in CCE loss

μ	1.0	1.2	1.4	**1.5**	1.6	1.8	2.0
$\overline{A}$	66.88	66.48	67.15	**68.16**	65.67	66.34	66.43

4.6 Balanced Classification Boundaries

In this section, we demonstrate the effect of the NeCa training framework on the classification boundaries between the old and new classes of the classification model.

As shown in Fig. 4, when optimizing the CE loss (2), due to the inconsistent data distribution between the training and testing sets, the classification boundary that performs well on the training set may lead to many misjudgments when stepping into the test phase. The optimization of the CCE loss (8) will shift the original classification boundary by a distance toward the center of the new class, allowing the model to make more balanced inferences about the old and new classes, thus solving the TRB problem.

Experimentally, we record the accuracy of each CIL method on Cifar-100 for multiple tasks on both old and new classes and then calculate their average gap. The results in Table 7 demonstrate that NeCa allows the final classification boundary to reach the most balanced state thanks to the calibration of the whole network parameters, thus resulting in the minimal accuracy gap between the old and new classes and the highest overall accuracy.

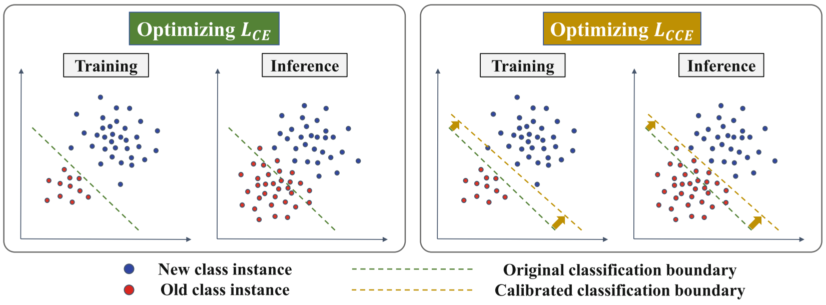

Fig. 4. Impact of optimizing CCE loss on the classification boundary between old and new classes

Table 7. The accuracy gap between the old and new classes of various CIL methods

Method	Classes	Task 2	Task 4	Task 6	Task 8	Task 10	Accuracy gap
iCaRL	old	83.4	66.6	59.1	51.2	45.5	12.34
	new	74.0	70.6	69.5	66.0	68.6	
EEIL	old	78.0	69.3	60.6	52.1	49.0	9.11
	new	84.3	72.2	69.7	68.9	59.4	
BiC	old	75.8	67.8	61.2	53.8	50.1	8.13
	new	87.8	75.9	68.4	60.7	56.7	
WA	old	79.3	67.5	62.0	56.2	51.5	7.64
	new	85.5	80.2	69.2	62.3	57.5	
NeCa	old	85.0	70.9	63.5	58.4	53.7	**6.50**
	new	81.1	77.1	72.4	62.1	63.5	

5 Conclusion

In this work, we combat the task-recency bias problem in the class incremental learning scenario. We propose a novel framework network calibration (NeCa) to calibrate the whole network via the prior probability and pass the ability of balanced prediction through knowledge distillation. Finally, the decaying regularization further improves the performance. Extensive experiments confirm the effectiveness and generalizability of NeCa. However, the bottleneck of having to preserve some of the samples remains unresolved. In the future, we will try to extend NeCa into the memory-free scenario.

Acknowledgements. This work was partially supported by NSFC (61976112), and Fundamental Research Funds for the Central Universities (2023300246).

References

1. Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., Bejnordi, B.E.: Conditional channel gated networks for task-aware continual learning. In: CVPR, pp. 3930–3939 (2020)
2. Ahn, H., Kwak, J., Lim, S., Bang, H., Kim, H., Moon, T.: SS-IL: separated softmax for incremental learning. In: ICCV, pp. 844–853 (2021)
3. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: learning what (not) to forget. In: ECCV, pp. 144–161 (2018)
4. Belouadah, E., Popescu, A., Kanellos, I.: A comprehensive study of class incremental learning algorithms for visual tasks. Neural Netw. **135**, 38–54 (2021)
5. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV, pp. 241–257 (2018)
6. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: practical automated data augmentation with a reduced search space. In: CVPR Workshops, pp. 3008–3017 (2020)
7. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR, pp. 248–255 (2009)
8. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale. In: ICLR (2021)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: ICCV, pp. 2980–2988 (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
11. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
12. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
13. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR, pp. 831–839 (2019)
14. Jian, Y., Yi, J., Zhang, L.: Adaptive feature generation for online continual learning from imbalanced data. In: PAKDD, vol. 13280, pp. 276–289 (2022)
15. Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. PNAS **114**(13), 3521–3526 (2017)
16. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report (2012)
17. LeCun, Y., Cortes, C.: MNIST handwritten digit database. Public (2010)
18. Lee, K., Lee, K., Shin, J., Lee, H.: Overcoming catastrophic forgetting with unlabeled data in the wild. In: ICCV, pp. 312–321 (2019)
19. Li, Z., Hoiem, D.: Learning without forgetting. In: ECCV, pp. 614–629 (2016)
20. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation. arXiv preprint arXiv:2010.15277 (2020)
21. Menon, A.K., Jayasumana, S., Rawat, A.S., Jain, H., Veit, A., Kumar, S.: Long-tail learning via logit adjustment. In: ICLR (2021)
22. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: a review. Neural Netw. **113**, 54–71 (2019)

23. 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
24. Rajasegaran, J., Hayat, M., Khan, S.H., Khan, F.S., Shao, L.: Random path selection for continual learning. In: NeurIPS, pp. 12648–12658 (2019)
25. Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: incremental classifier and representation learning. In: CVPR, pp. 5533–5542 (2017)
26. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NeurIPS, pp. 2990–2999 (2017)
27. van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv preprint arXiv:1904.07734 (2019)
28. Verwimp, E., Lange, M.D., Tuytelaars, T.: Rehearsal revealed: the limits and merits of revisiting samples in continual learning. In: ECCV, pp. 9385–9394 (2021)
29. Wu, T., et al.: Class-incremental learning with strong pre-trained models. In: CVPR, pp. 9591–9600 (2022)
30. Wu, Y., et al.: Large scale incremental learning. In: CVPR, pp. 374–382 (2019)
31. Yan, S., Xie, J., He, X.: DER: dynamically expandable representation for class incremental learning. In: CVPR, pp. 3014–3023 (2021)
32. Ye, H., Chen, H., Zhan, D., Chao, W.: Identifying and compensating for feature deviation in imbalanced deep learning. arXiv preprint arXiv:2001.01385 (2020)
33. Yu, L., et al.: Semantic drift compensation for class-incremental learning. In: CVPR, pp. 6980–6989 (2020)
34. Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S.: Maintaining discrimination and fairness in class incremental learning. In: CVPR, pp. 13205–13214 (2020)