

DER: Dynamically Expandable Representation for Class Incremental Learning

Shipeng Yan^{1,3,4*} Jiangwei Xie^{1*} Xuming He^{1,2}

¹School of Information Science and Technology, ShanghaiTech University

²Shanghai Engineering Research Center of Intelligent Vision and Imaging

³Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences

⁴University of Chinese Academy of Sciences

{yanshp, xiejw, hexm}@shanghaitech.edu.cn

Abstract

We address the problem of class incremental learning, which is a core step towards achieving adaptive vision intelligence. In particular, we consider the task setting of incremental learning with limited memory and aim to achieve better stability-plasticity trade-off. To this end, we propose a novel two-stage learning approach that utilizes a dynamically expandable representation for more effective incremental learning. Specifically, at each incremental step, we freeze the previously learned representation and augment it with additional feature dimensions from a new learnable feature extractor. This enables us to integrate new visual concepts with retaining learned knowledge. We dynamically expand the representation according to the complexity of novel concepts by introducing a channel-level mask-based pruning strategy. Moreover, we introduce an auxiliary loss to encourage the model to learn diverse and discriminate features for novel concepts. We conduct extensive experiments on the three class incremental learning benchmarks and our method consistently outperforms other methods with a large margin.¹

1. Introduction

Human can easily accumulate visual knowledge from past experiences and incrementally learn novel concepts. Inspired by this, the problem of class incremental learning aims to design algorithms that can learn novel concepts in a sequential manner and eventually perform well on all observed classes. Such capability is indispensable for many real-world applications such as the intelligent robot [31], face recognition [19] and autonomous driving [25]. However, achieving human-level incremental learning remains

*Both authors contributed equally. This work was supported by Shanghai NSF Grant (No. 18ZR1425100)

¹Code is available at <https://github.com/Rhyssiyan/DER-ClassIL.pytorch>.

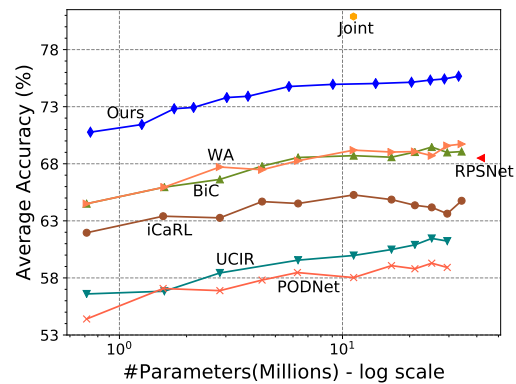


Figure 1: The average incremental accuracy for different model size. We compare our model with prior methods (WA[39], BiC[12], RPSNet[26], iCaRL[27], UCIR[12], PODNet[6]) and the model trained on all the data (Joint) on the experiment CIFAR100-B0 of 10 steps.

challenging for modern visual recognition systems.

There has been much effort attempting to address the incremental learning in literature [36, 23, 27, 3, 12, 33, 39]. Among them, perhaps the most effective strategy is to keep a memory buffer that stores part of observed data for the rehearsal [28, 29] in future. However, due to the limited size of data memory, such the incremental learning method still faces several challenges in the general continual learning task. In particular, it requires a model to effectively incorporate novel concepts without forgetting the existing knowledge, which is also known as *stability-plasticity dilemma* [9]. In detail, excessive plasticity often causes large performance degradation of the old categories, referred to as catastrophic forgetting [8]. On the contrary, excessive stability impedes the adaptation of novel concepts.

Most existing works attempt to achieve a trade-off between stability and plasticity by gradually updating data representation and class decision boundary for increasingly

larger label spaces. For instance, regularization methods [4] penalize the change of important weights of previously learned models, while knowledge distillation [27, 3, 12, 6, 34] preserves the network output with available data, and structure-based methods [26, 1] keep old parameters fixed when allocating more for new categories. Nevertheless, all those methods either sacrifice the model plasticity for the stability, or are susceptible to forgetting due to feature degradation of old concepts. As shown in Figure 1, a large performance gap still exists between the model (*Joint*) trained on all data and previous state-of-the-art models.

In this work, we aim to address the above weaknesses and achieve a better stability-plasticity trade-off in the class incremental learning. To this end, we adopt a two-stage learning strategy, decoupling the adaptation of feature representation and final classifier head (*or classifier for short*) of a deep network [15]. Within this framework, we propose a novel data representation, referred to as *super-feature*, capable of increasing its dimensionality to accommodate new classes. Our main idea is to freeze the previously learned representation and augment it with additional feature dimensions from a new learnable extractor in each incremental step. This enables us to retain the existing knowledge and provides enough flexibility to learn novel concepts. Moreover, our super-feature is expanded dynamically based on the complexity of novel concepts to maintain a compact representation.

To achieve this, we develop a modular deep classification network composed of a super-feature extractor network and a linear classifier. Our super-feature extractor network consists of multiple feature extractors with varying sizes, one for each incremental step. Specifically, at a new step, we expand the super-feature extractor network with a new feature extractor while keeping the parameters of previous extractors frozen. The features generated by all the extractors are concatenated together and fed into the classifier for the class prediction.

We train the new feature extractor and the classifier on the memory and the newly incoming data. To encourage the new extractor to learn diverse and discriminative features for new classes, we design an auxiliary loss on distinguishing new and old classes. Additionally, to remove the model redundancy and learn the compact features for novel classes, we apply a differentiable channel-level mask-based pruning method that dynamically prunes the network according to the difficulty of novel concepts. Finally, given the updated representation, we freeze the super-feature extractor and finetune the classifier on a balanced training subset to solve the class imbalance problem [33, 39].

We validate our approach on three commonly used benchmarks, including CIFAR-100, ImageNet-100, and ImageNet-1000 datasets. The empirical results and the ablation study demonstrate the superiority of our method over

prior state-of-the-art approaches. Interestingly, we also find that our method could achieve positive backward and forward transfer between steps. The main contributions of our work are three-fold:

- To achieve better stability-plasticity trade-off, we develop a dynamically expandable representation and a two-stage strategy for the class incremental learning.
- We propose an auxiliary loss to promote the newly added feature module to learn novel classes effectively and a model pruning step to learn compact features.
- Our approach achieves the new state of the art performance on all three benchmarks under a wide range of model complexity, as shown in Figure 1.

2. Related Work

Class incremental learning aims to learn new classes continuously. Some works [36, 23] try to solve the problem with no access to previously seen data. However, prevalent approaches are based on the rehearsal strategy with limited data memory, which can be mainly analyzed from representation learning and classifier learning.

Representation Learning Current works can be mainly divided into the following three categories. *Regularization-based* methods [16, 37, 18, 4, 2] adopt Maximum a Posterior estimation to expect small changes in the important parameters and update the posterior of model parameters sequentially. However, its intractable computation typically requires approximations with a strong model assumption. For example, EWC [16] uses Laplace approximation, which assumes weights falling into a local region of the optimal weights of last step. This severely restricts the model capacity to adapt to novel concepts.

Distillation-based methods [27, 39, 33, 12, 3, 6, 34] use knowledge distillation [11] to maintain the representation. iCaRL [27] and EE2L [3] compute the distillation loss on the network outputs. UCIR [12] uses normalized feature vectors to apply the distillation loss instead of the prediction of the network. PODNet [6] uses a spatial-based distillation loss to restrict the change of model. TPCIL [34] makes the model preserve the topology of CNN’s feature space. The performance of knowledge distillation depends on the quality and quantity of saved data.

Structure-based methods [21, 13, 30, 20, 22, 7, 35, 26, 1, 20] keep the learned parameters related to previous classes fixed and allocate new parameters in different forms such as unused parameters, additional networks to learn novel knowledge. CPG [13] proposes a compaction and selection/expansion mechanism that prunes the deep model and expands the architecture alternatively with selectively weight sharing. However, most structure-based [21, 13, 30, 20, 22, 7, 35] methods are designed for task continual learning, which needs task identity during inference. For class incremental learning, RPSNet [26] proposes a random

path selection algorithm that progressively chooses optimal paths as sub-network for the new classes. CCGN [1] equips each convolutional layer with task-specific gating modules to select filters to apply on the given input and uses a task predictor to choose the gating modules in inference.

Classifier Learning Class-imbalance problem is the main challenge for classifier learning due to limited memory. Some works like LWF.MC[27], RWalk[4] train the extractor and classifier jointly within one-stage training. By contrast, recently, there are many works to solve the class imbalance problem by introducing an independent classifier learning stage after representation learning. EEIL[3] finetunes the classifier on a balanced training subset. BiC[33] adds a bias correction layer to correct the model’s outputs, where the layer is trained on a separate validation set. WA[39] corrects the biased weights by aligning the norms of the weight vectors for new classes to those for old classes.

Discussion Our work is a structure-based method and the most similar work to ours are RPSNet and CCGN. RPSNet cannot retain the intrinsic structure of each old concept and tends to gradually forget the learned concepts by summing the previously learned features and the newly learned features at each ConvNet stage. In CCGN, the learned representation may slowly degrade over steps as only the parameters of part of layers are frozen. By contrast, we keep the previously learned representation fixed and augment it with novel features parameterized by a new feature extractor. This enables us to preserve the intrinsic structure of old concepts in the subspace of previously learned representation, and re-use the structure via the final classifier to mitigate forgetting.

3. Methods

In this section, we present our approach to the problem of class incremental learning, aiming to achieve a better trade-off between stability and plasticity. To this end, we propose a dynamically expandable representation (DER) that incrementally augments previously learned representation with novel features and a two-stage learning strategy.

Below we first present the formulation of class incremental learning and an overview of our method in Sec. 3.1. Then we introduce the expandable representation learning and its loss function in Sec. 3.2. After this, we describe the dynamic expansion of our representation in Sec. 3.3 and the second stage of classifier learning in Sec. 3.4.

3.1. Problem Setup and Method Overview

Firstly, we introduce the problem setup of class incremental learning. In contrast to task incremental learning, class incremental learning does not require task id during inference. Specifically, during the class incremental learning, the model observes a stream of class groups $\{\mathcal{Y}_t\}$ and

their corresponding training data $\{\mathcal{D}_t\}$. Particularly, the incoming dataset \mathcal{D}_t at step t has a form of (\mathbf{x}_i^t, y_i^t) where \mathbf{x}_i^t is the input image and $y_i^t \in \mathcal{Y}_t$ is the label within the label set \mathcal{Y}_t . The label space of the model is all seen categories $\tilde{\mathcal{Y}}_t = \cup_{i=1}^t \mathcal{Y}_i$ and the model is expected to predict well on all classes in $\tilde{\mathcal{Y}}_t$.

Our method adopts the rehearsal strategy, which saves a part of data as the memory \mathcal{M}_t for future training. For the learning of step t , we decouple the learning process into two sequential stages as follows.

1) *Representation Learning Stage.* To achieve better trade-off between stability and plasticity, we fix the previous feature representation and expand it with a new feature extractor trained on the incoming and memory data. We design an auxiliary loss on the novel extractor to promote it to learn diverse and discriminative features. To improve the model efficiency, we dynamically expand the representation according to the complexity of new classes via introducing a channel-level mask-based pruning method. The overview of our proposed representation is shown in Figure 2.

2) *Classifier Learning Stage.* After the learning of representation, we retrain the classifier with currently available data $\tilde{\mathcal{D}}_t = \mathcal{D}_t \cup \mathcal{M}_t$ at step t to deal with the class imbalance problem via adopting the balanced finetuning method in [3].

3.2. Expandable Representation Learning

We first introduce our expandable representation. At step t , our model is composed of a super-feature extractor Φ_t and the classifier \mathcal{H}_t . The super-feature extractor Φ_t is built by expanding the feature extractor Φ_{t-1} with a newly created feature extractor \mathcal{F}_t . Specifically, given an image $\mathbf{x} \in \tilde{\mathcal{D}}_t$, the feature \mathbf{u} extracted by Φ_t is obtained by concatenation as follows

$$\mathbf{u} = \Phi_t(\mathbf{x}) = [\Phi_{t-1}(\mathbf{x}), \mathcal{F}_t(\mathbf{x})] \quad (1)$$

Here we reuse the previous $\mathcal{F}_1, \dots, \mathcal{F}_{t-1}$ and encourage the new extractor \mathcal{F}_t to learn only novel aspect of new classes. The feature \mathbf{u} is then fed into the classifier \mathcal{H}_t to make prediction as follows

$$p_{\mathcal{H}_t}(\mathbf{y}|\mathbf{x}) = \text{Softmax}(\mathcal{H}_t(\mathbf{u})) \quad (2)$$

Then the prediction $\hat{y} = \arg \max p_{\mathcal{H}_t}(\mathbf{y}|\mathbf{x})$, $\hat{y} \in \tilde{\mathcal{Y}}_t$. The classifier is designed to match its new input and output dimensions for step t . The parameters of \mathcal{H}_t for the old features are inherited from \mathcal{H}_{t-1} to retain old knowledge and its newly added parameters are randomly initialized.

To reduce catastrophic forgetting, we freeze the learned function Φ_{t-1} at step t , as it captures the intrinsic structure of previous data. In detail, the parameters of last step super-feature extractor $\theta_{\Phi_{t-1}}$ and the statistics of Batch Normalization[14] are not updated. Besides, we instantiate

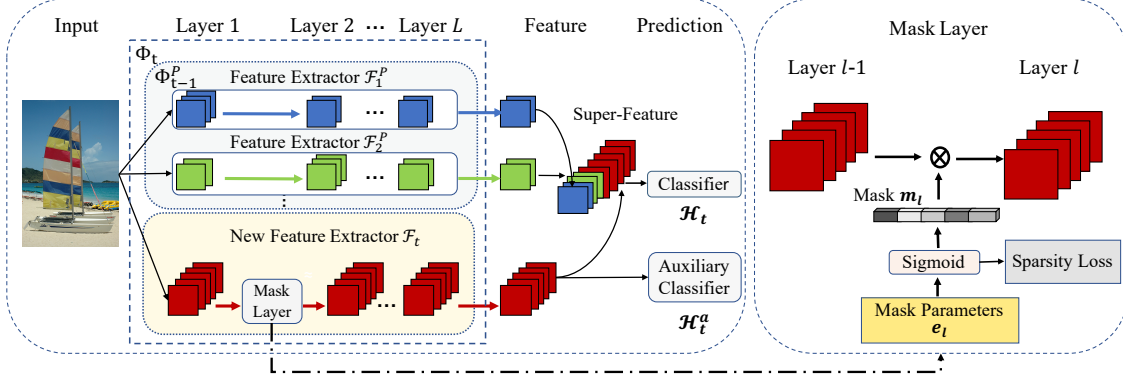


Figure 2: Dynamically Expandable Representation Learning. At step t , the model is composed of super-feature extractor Φ_t and classifier \mathcal{H}_t , where Φ_t is built by expanding the existing super-feature extractor Φ_{t-1}^p with new feature extractor \mathcal{F}_t . We also use an auxiliary classifier to regularize the model. Besides, the layer-wise channel-level mask is jointly learned with the representation, which is used to prune the network after the learning of model.

\mathcal{F}_t with \mathcal{F}_{t-1} as initialization to reuse previous knowledge for fast adaptation and forward transfer.

We can shed the light on the problem from the perspective of estimating the prior distribution $p(\theta_{\Phi_t} | \mathcal{D}_{1:t-1})$ given the previous data $\mathcal{D}_{1:t-1}$. Unlike previous regularization methods like EWC, we do not assume the prior distribution for t -th step is unimodal, which restricts the model flexibility and is typically not the case in practice. For our method, the model expands with new parameters by creating a separate feature extractor \mathcal{F}_t for incoming data and take a uniform distribution as the prior distribution $p(\theta_{\mathcal{F}_t} | \mathcal{D}_{1:t-1})$ which provides enough flexibility for the model to adapt to novel concepts. Meanwhile, for simplicity, we approximate the prior distribution $p(\theta_{\Phi_{t-1}} | \mathcal{D}_{1:t-1})$ on the old parameters $\theta_{\Phi_{t-1}}$ as the Dirac distribution, which maintains the information learned on $\mathcal{D}_{1:t-1}$. By integrating two prior distribution assumptions on $p(\theta_{\Phi_{t-1}} | \mathcal{D}_{1:t-1})$ and $p(\theta_{\mathcal{F}_t} | \mathcal{D}_{1:t-1})$, we have more flexibility in achieving a better stability-plasticity trade-off.

Training Loss We learn the model with cross-entropy loss on memory and incoming data as follows

$$\mathcal{L}_{\mathcal{H}_t} = -\frac{1}{|\tilde{\mathcal{D}}_t|} \sum_{i=1}^{|\tilde{\mathcal{D}}_t|} \log(p_{\mathcal{H}_t}(y = y_i | \mathbf{x}_i)) \quad (3)$$

where \mathbf{x}_i is image and y_i is the corresponding label.

To enforce the network to learn the diverse and discriminative features for novel concepts, we further develop an auxiliary loss operating on the novel feature $\mathcal{F}_t(\mathbf{x})$. Specifically, we introduce an auxiliary classifier \mathcal{H}_t^a , which predicts the probability $p_{\mathcal{H}_t^a}(y | \mathbf{x}) = \text{Softmax}(\mathcal{H}_t^a(\mathcal{F}_t(\mathbf{x})))$. To encourage the network to learn features to discriminate between old and new concepts, the label space of \mathcal{H}_t^a is $|\mathcal{Y}_t| + 1$ including the new category set \mathcal{Y}_t and the other class by treating all old concepts as one category. Thusly, we intro-

duce the auxiliary loss and obtain the expandable representation loss as follows

$$\mathcal{L}_{\text{ER}} = \mathcal{L}_{\mathcal{H}_t} + \lambda_a \mathcal{L}_{\mathcal{H}_t^a} \quad (4)$$

where λ_a is the hyper-parameter to control the effect of the auxiliary classifier. It is worth noting that $\lambda_a = 0$ for first step $t = 1$.

3.3. Dynamical Expansion

To remove the model redundancy and maintain a compact representation, we dynamically expand the super-feature according to the complexity of novel concepts. Specifically, we adopt a differentiable channel-level mask-based method to prune filters of the extractor \mathcal{F}_t , in which the masks are learned with the representation jointly. After the learning of the mask, we binarize the mask and prune the feature extractor \mathcal{F}_t to obtain the pruned network \mathcal{F}_t^p .

Channel-level Masks Our pruning method is based on differentiable channel-level masks, which is adapted from HAT [30]. For the novel feature extractor \mathcal{F}_t , the input feature map of convolutional layer l for a given image \mathbf{x} is denoted as \mathbf{f}_l . We introduce the channel mask $\mathbf{m}_l \in \mathbb{R}^{c_l}$ to control the size of layer l where $m_l^i \in [0, 1]$ and c_l is the number of channels of layer l . \mathbf{f}_l is modulated with the mask as follows

$$\mathbf{f}_l' = \mathbf{f}_l \odot \mathbf{m}_l \quad (5)$$

where \mathbf{f}_l' is the masked feature map, \odot means channel-level multiplication. To make the value of \mathbf{m}_l fall into the interval $[0, 1]$, the gating function is adopted as follows

$$\mathbf{m}_l = \sigma(s\mathbf{e}_l) \quad (6)$$

where \mathbf{e}_l means learnable mask parameters, the gating function $\sigma(\cdot)$ uses the sigmoid function in this work and s is the

scaling factor to control the sharpness of the function. With such a mask mechanism, the super-feature \tilde{u} of step t can be rewritten as

$$\tilde{u} = \Phi_t^P(\mathbf{x}) = [\mathcal{F}_1^P(\mathbf{x}), \mathcal{F}_2^P(\mathbf{x}), \dots, \phi_t(\mathbf{x})] \quad (7)$$

During training, $\phi_t(\mathbf{x})$ is $\mathcal{F}_t(\mathbf{x})$ with the soft masks. For inference, we assign s a large value to binarize masks and obtain the pruned network \mathcal{F}_t^P , and $\phi_t(\mathbf{x}) = \mathcal{F}_t^P(\mathbf{x})$

Mask Learning During a epoch, a linear annealing schedule is applied for s as follows

$$s = \frac{1}{s_{\max}} + (s_{\max} - \frac{1}{s_{\max}}) \frac{b-1}{B-1} \quad (8)$$

where b is the batch index, $s_{\max} \gg 1$ is the hyper-parameter to control the schedule, B is the number of batches in one epoch. The training epoch starts with all channels activated in a uniform way. Then the mask is progressively binarized with the increasing of batch index within a epoch.

One of the problems of the sigmoid function is that the gradient is unstable due to the s schedule. We compensate the gradient g_{e_l} with respect to e_l to remove the influence of s as follows

$$g'_{e_l} = \frac{\sigma(e_l)[1 - \sigma(e_l)]}{s\sigma(se_l)[1 - \sigma(se_l)]} g_{e_l} \quad (9)$$

where g'_{e_l} is the compensated gradient.

Sparsity Loss At every step, we encourage the model to maximally reduce the number of parameters with a minimal performance drop. Motivated by this, we add a sparsity loss based on the ratio of used weights in all available weights.

$$\mathcal{L}_S = \frac{\sum_{l=1}^L K_l \|m_{l-1}\|_1 \|m_l\|_1}{\sum_{l=1}^L K_l c_{l-1} c_l} \quad (10)$$

where L is the number of layers, K_l is the kernel size of convolution layer l , layer $l=0$ refers to the input image, and $\|m_0\|_1=3$.

After adding the sparsity loss, the final loss function is

$$\mathcal{L}_{\text{DER}} = \mathcal{L}_{\mathcal{H}_t} + \lambda_a \mathcal{L}_{\mathcal{H}_t^a} + \lambda_s \mathcal{L}_S \quad (11)$$

where λ_s is the hyper-parameter to control the model size.

3.4. Classifier Learning

At the representation learning stage, we re-train the classifier head in order to reduce the bias in the classifier weight introduced by the imbalanced training. Specifically, we first re-initialize the classifier with random weights and then sample a class-balanced subset from currently available data $\tilde{\mathcal{D}}_t$. We train the classifier head only using the cross-entropy loss with a temperature δ in the Softmax [38]. The temperature controls the smoothness of the Softmax function to improve the margins between classes.

4. Experiments

In this section, we conduct extensive experiments to validate the effectiveness of our algorithm. Especially, we evaluate our method on CIFAR-100[27], ImageNet-100[27] and ImageNet-1000[27] datasets with two widely used benchmark protocols. We also perform a series of ablation studies to evaluate the importance of each component and provides more insights into our method. Below we first start with the introduction of experiment setup and implementation details in Sec. 4.1, followed by the experimental results on the CIFAR100 dataset in Sec. 4.2. Then we present the evaluation results on both ImageNet-100 and ImageNet-1000 datasets in Sec. 4.3. Finally, we introduce the ablation study and analysis for our method in Sec. 4.4.

4.1. Experiment Setup and Implementation Details

Datasets CIFAR-100 [17] consists of 32x32 pixel color images with 100 classes. It contains 50,000 images for training with 500 images per class, and 10,000 images for evaluation with 100 images per class. ImageNet-1000 [5] is a large-scale dataset from 1,000 classes which includes about 1.2 million RGB images for training and 50,000 images for validation. ImageNet-100 [27, 12] is built by selecting 100 classes from the ImageNet-1000 dataset.

Benchmark Protocols For the CIFAR-100 benchmark, we test our methods on two popular protocols including 1) *CIFAR100-B0*: we follow the protocol proposed in [27], which trains all 100 classes in several splits including 5, 10, 20, 50 incremental steps with fixed memory size of 2,000 exemplars over batches; 2) *CIFAR100-B50*: we follow the protocol introduced in [12], which starts from a model trained on 50 classes, and the remaining 50 classes are divided into splits of 2, 5, and 10 steps with 20 examples as memory per class. We compare the top-1 average incremental accuracy which takes the average of the accuracy for each step.

We also evaluate our method on ImageNet-100 with two protocols that are 1) *ImageNet100-B0*: the protocol [27] trains the model in batches of 10 classes from scratch with fixed memory size 2,000 over batches; 2) *ImageNet100-B50*: the protocol [12] starts from a model trained on 50 classes, and the remaining 50 classes come in 10 steps with 20 examples per class as memory. For the sake of fairness, we use the same ImageNet subset and class order following the protocols [27, 12]. For ImageNet-1000, we evaluate our method on the protocol [27], known as *ImageNet1000-B0* benchmark, that trains the model in batches of 100 classes with 10 steps in total and set the fixed memory size as 20,000. Detailedly, we use the same class order as [27] for ImageNet-1000. Moreover, we compare the top-1 and top-5 average incremental accuracy and the last step accuracy on ImageNet-100 and ImageNet-1000 datasets.

Methods	5 steps		10 steps		20 steps		50 steps	
	#Paras	Avg	#Paras	Avg	#Paras	Avg	#Paras	Avg
Bound	11.2	80.40	11.2	80.41	11.2	81.49	11.2	81.74
iCaRL[27]	11.2	71.14 \pm 0.34	11.2	65.27 \pm 1.02	11.2	61.20 \pm 0.83	11.2	56.08 \pm 0.83
UCIR[12]	11.2	62.77 \pm 0.82	11.2	58.66 \pm 0.71	11.2	58.17 \pm 0.30	11.2	56.86 \pm 3.74
BiC[12]	11.2	73.10 \pm 0.55	11.2	68.80 \pm 1.20	11.2	66.48 \pm 0.32	11.2	62.09 \pm 0.85
WA[39]	11.2	72.81 \pm 0.28	11.2	69.46 \pm 0.29	11.2	67.33 \pm 0.15	11.2	64.32 \pm 0.28
PODNet[6]	11.2	66.70 \pm 0.64	11.2	58.03 \pm 1.27	11.2	53.97 \pm 0.85	11.2	51.19 \pm 1.02
RPSNet[26]	60.6	70.5	56.5	68.6	-	-	-	-
Ours(w/o P)	33.6	76.80 \pm 0.79(+3.7)	61.6	75.36 \pm 0.36(+5.9)	117.6	74.09 \pm 0.33(+6.76)	285.6	72.41 \pm 0.36(+8.09)
Ours	2.89	75.55 \pm 0.65(+2.45)	4.96	74.64 \pm 0.28(+5.18)	7.21	73.98 \pm 0.36(+6.65)	10.15	72.05 \pm 0.55(+7.73)

Table 1: Results on CIFAR100-B0 benchmark which is averaged over three runs. #Paras means the average number of parameters used during inference over steps, which is counted by million. Avg means the average accuracy (%) over steps. Ours(w/o P) means our method without pruning.

Methods	2Steps		5Steps		10Steps	
	#Paras	Avg	#Paras	Avg	#Paras	Avg
Bound	11.2	77.22	11.2	79.89	11.2	79.91
iCaRL[27]	11.2	71.33 \pm 0.35	11.2	65.06 \pm 0.53	11.2	58.59 \pm 0.95
UCIR[12]	11.2	67.21 \pm 0.35	11.2	64.28 \pm 0.19	11.2	59.92 \pm 2.4
BiC[12]	11.2	72.47 \pm 0.99	11.2	66.62 \pm 0.45	11.2	60.25 \pm 0.34
WA[39]	11.2	71.43 \pm 0.65	11.2	64.01 \pm 1.62	11.2	57.86 \pm 0.81
PODNet[6]	11.2	71.30 \pm 0.46	11.2	67.25 \pm 0.27	11.2	64.04 \pm 0.43
Ours(w/o P)	22.4	74.61 \pm 0.52(+2.14)	39.2	73.21 \pm 0.78(+5.96)	67.2	72.81 \pm 0.88(+8.77)
Ours	3.90	74.57 \pm 0.42(+2.10)	6.13	72.60 \pm 0.78(+5.35)	8.79	72.45 \pm 0.76(+8.41)

Table 2: Results on CIFAR100-B50 (average over 3 runs). #Paras means the average number of parameters used during inference over steps, which is counted by million. Avg means the average accuracy (%) over steps. Ours(w/o P) means our method without pruning.

Implementation Details Our method is implemented with PyTorch [24]. For CIFAR-100, we adopt ResNet-18 as feature extractor \mathcal{F}_t following RPSNet [26]. We note that most previous works use a modified 32-layers ResNet[27], which has fewer channels and residual blocks compared to standard ResNet-32. We argue that such a small network is not suitable because it cannot achieve competitive results on CIFAR100 compared with standard 18-layers ResNet[10] and may underestimate the performance of methods. We run these methods with standard ResNet-18 on the same class orders based on their code implementation. For those without releasing the codes, we report the results based on our implementation. For RPSNet, we use the results in their paper directly. For ImageNet-100 and ImageNet-1000 benchmarks, we use 18-layers ResNet as the basic network. In these experiments, we select exemplars as memory based on the herding selection strategy[32] following the previous works[27]. Furthermore, we run experiments on three different class orders and report average \pm standard deviations in the results. We also provide the experimental results on CIFAR-100 based on modified 32-layers ResNet [27] in the appendix, which proves the superiority of our method again. We follow the protocol in [6, 30] and tune the hyper-

parameters on a validation set created by holding out a part of original training data. The details of the hyperparameters are added to the appendix.

4.2. Evaluation on CIFAR100

Quantitative Results Table 1 summarizes the results of CIFAR100-B0 benchmark. We can see that our method consistently outperforms other methods by a sizable margin at different incremental splits. As the number of steps increases in the split, it is observed that the margin between our method and other methods continuously increases which indicates that our method performs better on the difficult splits with longer steps. Particularly, under the incremental setting of 50 steps, we improve the average incremental accuracy from 64.32% to 72.05%(+7.73%) of ours with fewer parameters. It is worth noting that although huge model parameters are reduced, the performance degradation of our method caused by pruning can be ignored, which demonstrates that the success of our pruning method. As shown in the left panel of Figure 3, it is observed that our method consistently surpasses other methods at every step for different splits. Moreover, the gap between our method and other methods increases with the continuous adding of

Methods	ImageNet100-B0					ImageNet1000-B0					Methods	ImageNet100-B50				
	#Paras	top-1		top-5		#Paras	top-1		top-5			#Paras	top-1		top-5	
		Avg	Last	Avg	Last		Avg	Last	Avg	Last			Avg	Last	Avg	Last
Bound	11.2	-	-	-	95.1	11.2	89.27	-	-	-	Bound	11.2	81.20	81.5	-	-
iCaRL[27]	11.2	-	-	83.6	63.8	11.2	38.4	22.7	63.7	44.0	UCIR[12]	11.2	68.09	57.3	-	-
BiC[12]	11.2	-	-	90.6	84.4	11.2	-	-	84.0	73.2	PODNet[6]	11.2	74.33	-	-	-
WA[39]	11.2	-	-	91.0	84.1	11.2	65.67	55.6	86.6	81.1	TPCIL[34]	11.2	74.81	66.91	-	-
RPSNet[26]	-	-	-	87.9	74.0	-	-	-	-	-	Ours(w/o P)	67.20	78.20	74.92	94.20	91.30
Ours(w/o P)	61.6	77.18	66.70	93.23	87.52	61.6	68.84	60.16	88.17	82.86	Ours	8.87	77.73	72.06	94.01	91.64
Ours	7.67	76.12	66.06	92.79	88.38	14.52	66.73	58.62	87.08	81.89						

Table 3: Results on ImageNet-100 and ImageNet-1000 datasets. **Left:** The results on ImageNet100-B0 and ImageNet1000-B0 benchmark. **Right:** The results on ImageNet100-B50 benchmark. #Paras means the average number of parameters during inference over steps, which is counted by million. Avg means the average accuracy (%) over steps. Last is the accuracy (%) of the last step. Ours(w/o P) means our method without pruning.

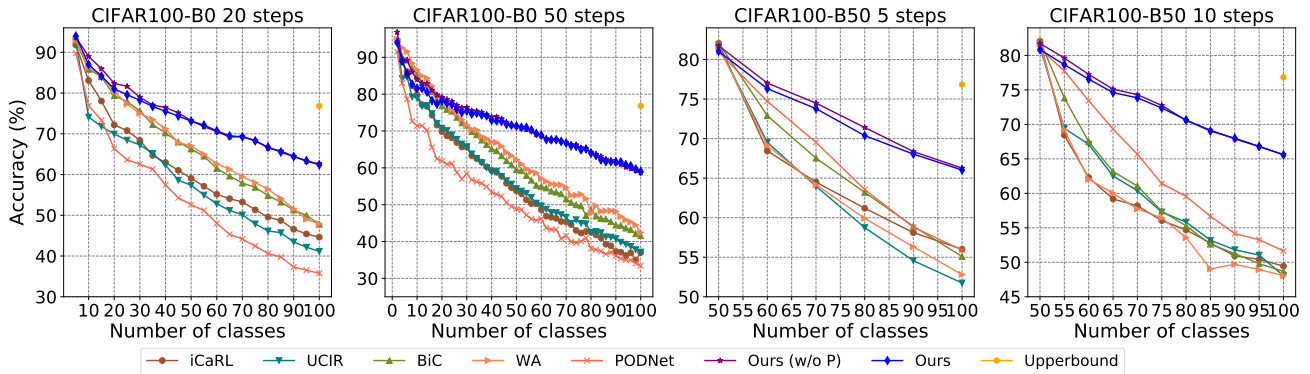


Figure 3: The performance for each step. **Left** is evaluated on CIFAR100-B0 of 20 and 50 steps and **Right** is evaluated on CIFAR100-B50 of 5 and 10 steps.

novel classes. Specifically, under the incremental split of 50 steps, the last step accuracy is boosted from 42.75% to 58.66%(+15.91%), which further proves that the effectiveness of our method.

We also compare the performance of our method with previous methods in Table 2 on the CIFAR100-B50 benchmark, which shows our method improves the performance with a significant gain in all splits. In particular, under the incremental setting of 10 steps, our method outperforms PODNet by 8.41% average incremental accuracy. As the right panel of Figure 3 shows, our method performs better than other methods at each step for all splits. Especially, our method improves from 52.56% to 65.58%(+13.02%) for the last step accuracy in the split of 10 steps. Moreover, our method achieves similar performance with much fewer parameters compared to our method without pruning.

It is worth noting that previous methods often perform well only on one of the protocols where WA is the state-of-the-art on CIFAR100-B0 and PODNet is state-of-the-art on CIFAR100-B50. By contrast, our method consistently surpasses other methods on both protocols.

The effects of model size We conduct extensive experiments to study the effect of model size on performance. As

shown in Figure 1, we can see that our method consistently and significantly performs better than other methods at various model sizes. We also note that the improvement of our method compared to most other methods becomes more significant with the increasing of model size, which illustrates that our method can exploit the potential of a large model.

4.3. Evaluation on ImageNet

Table 3 summarizes the experimental results for the ImageNet-100 and ImageNet-1000 datasets. We can see that our method consistently surpasses other methods with a considerable margin for all splits on ImageNet-100 and ImageNet-1000 datasets, especially the last step accuracy. Specifically, our method outperforms the state-of-the-art with about 1.79% for the average top-5 accuracy on the ImageNet100-B0 benchmark. For ImageNet100-B50 benchmark, the last step top-1 accuracy is improved from 66.91% to 72.06%(+5.15%). Furthermore, our method improves the final step top-1 accuracy from 55.6% to 58.62%(+3.02%) on ImageNet1000-B0 benchmark. While the top-5 accuracy gap is smaller, we believe it is because top-5 accuracy is more tolerant to slightly inaccurate predictions and thus less sensitive to forgetting.

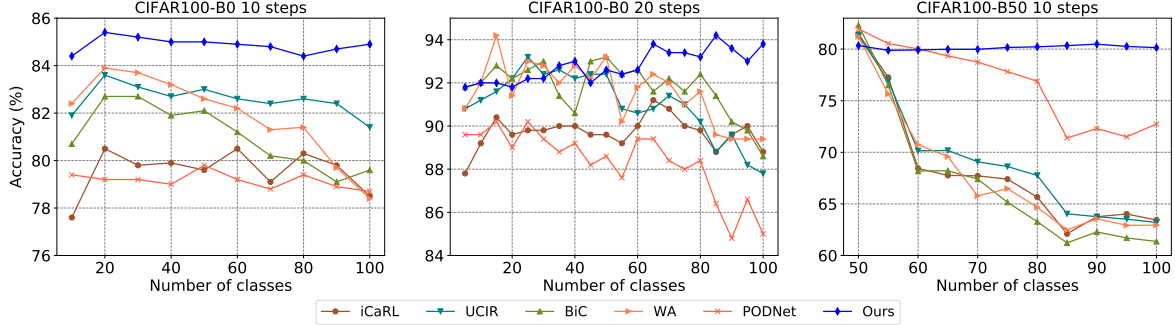


Figure 4: **Analysis.** The backward transfer of representation by observing the changes of $A_{\mathcal{Y}_1}^t$ for different splits.

Components		Avg	Last
E.R.	Aux.		
✗	✗	61.84	40.81
✓	✗	73.26	63.07
✓	✓	75.36	65.34

Table 4: The contribution of each component. *E.R.* means expandable representation. *Aux.* means using auxiliary loss.

4.4. Ablation Study and Analysis

We conduct exhaustive ablation study to evaluate the contribution of each component for our method. We also conduct sensitive study for hyper-parameters stated in the appendix. Moreover, we study the backward transfer and forward transfer of the representation for each method.

The effect of each component Table 4 summarizes the results of our ablative experiments on CIFAR100-B0 with 10 steps. We can see that the average accuracy is improved significantly from 61.84% to 73.26% by representation expansion. We also show that the performance of the model is further improved with **2.10%** gain using auxiliary loss.

Backward Transfer for Representation To assess the quality of representation, we introduce an ideal decision boundary obtained by finetuning the classifier with all observed data, which allows us to exclude the influence of the classifier. We then define classification accuracy $A_{\mathcal{Y}_k}^t$ at step t as the accuracy on the test images of class set \mathcal{Y}_k , where the prediction space of the model is restricted to \mathcal{Y}_k . By observing the $A_{\mathcal{Y}_k}^t$ curve over t , we can see how the representation quality evolves along the increments. Figure 4 shows the results of CIFAR100-B0 with 10 incremental steps. We also compute a backward transfer value for different methods as follows:

$$\text{BWT} = \frac{1}{T-1} \sum_{i=2}^T \frac{1}{i} \sum_{j=1}^{i-1} A_{\mathcal{Y}_j}^i - A_{\mathcal{Y}_j}^1 \quad (12)$$

The results are shown in Table 5. We can see that other methods suffer from severe forgetting. In contrast, our method even achieves positive backward transfer **+1.36%** and the accuracy increases with respect to steps, which further proves the superiority of our method.

Methods	iCaRL	UCIR	BiC	WA	PODNet	Ours
BWT (%)	-4.14	-8.52	-3.40	-3.18	-16.27	+1.36
FWT (%)	-4.91	-5.56	-0.17	+0.82	-5.58	+1.49

Table 5: Backward transfer and Forward transfer (FWT) for representation.

Forward Transfer for Representation We also measure the influence of existing knowledge on the performance of subsequent concepts on CIFAR100-B0 with 10 incremental steps, known as forward transfer. Specifically, we define a forward transfer rate for representation as follows

$$\text{FWT} = \frac{1}{T-1} \sum_{i=2}^T A_{\mathcal{Y}_i}^i - \bar{A}_{\mathcal{Y}_i}^i \quad (13)$$

where $\bar{A}_{\mathcal{Y}_i}^i$ is the test accuracy obtained by model trained on available data $\tilde{\mathcal{D}}_t$ with only cross-entropy loss at random initialization. As shown in Table 5, it is observed that most methods have negative forward transfer, which indicates that they sacrifice the flexibility of adaptation to novel concepts. By contrast, our method achieves **+1.49%** FWT which implies that our method not only makes the model highly flexible but also brings the positive forward transfer.

5. Conclusion

In this work, we propose dynamically expandable representation to improve the representation for class incremental learning. At each step, we freeze previously learned representation and augment it with novel parameterized feature. We also introduce channel-level mask-based pruning to dynamically expand representation according to the difficulty of novel concepts and an auxiliary loss to learn the novel discriminative features better. We conduct exhaustive experiments on the three major incremental classification benchmarks. The experimental results show that our method consistently performs better than other methods with a sizable margin. Interestingly, we also find that our method can even achieve positive backward and forward transfer.

References

- [1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020. 2, 3
- [2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2
- [3] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3
- [4] 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)*, 2018. 2, 3
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2009. 5
- [6] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 6, 7, 12
- [7] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 2
- [8] Robert M. French and Nick Chater. Using noise to compute error surfaces in connectionist networks: A novel means of reducing catastrophic forgetting. *Neural Comput.*, 2002. 1
- [9] Stephen Grossberg. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, 2013. 1
- [10] 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 (CVPR)*, 2016. 6
- [11] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, 2015. 2
- [12] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2019. 1, 2, 5, 6, 7, 12
- [13] Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2
- [14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 3
- [15] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations (ICLR)*, 2020. 2
- [16] 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 (PNAS)*, 2017. 2
- [17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. 5
- [18] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems (NeurIPS)*, 2017. 2
- [19] Lufan Li, Zhang Jun, Jiawei Fei, and Shuohao Li. An incremental face recognition system based on deep learning. In *International Conference on Machine Vision Applications (MVA)*, 2017. 1
- [20] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning (ICML)*, 2019. 2
- [21] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2
- [22] Arun Mallya and Svetlana Lazebnik. Packetnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2018. 2
- [23] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jah-nichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2
- [24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6
- [25] John M Pierre. Incremental lifelong deep learning for autonomous vehicles. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2018. 1
- [26] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 6, 7

- [27] 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 (CVPR)*, 2017. [1](#), [2](#), [3](#), [5](#), [6](#), [7](#), [11](#), [12](#)
- [28] Anthony V. Robins. Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In *International Two-Stream Conference on Artificial Neural Networks and Expert Systems, ANNES*, 1993. [1](#)
- [29] Anthony V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.*, 1995. [1](#)
- [30] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018. [2](#), [4](#), [6](#)
- [31] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 1995. [1](#)
- [32] Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning (ICML)*, 2009. [6](#)
- [33] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2019. [1](#), [2](#), [3](#)
- [34] Tao Xiao, Chang Xinyuan, Hong Xiaopeng, Wei Xing, and Gong Yihong. Topology-preserving class-incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. [2](#), [7](#), [12](#)
- [35] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations (ICLR)*, 2018. [2](#)
- [36] Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [1](#), [2](#)
- [37] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 2017. [2](#)
- [38] Xu Zhang, Felix Xinnan Yu, Svebor Karaman, Wei Zhang, and Shih-Fu Chang. Heated-up softmax embedding. *arXiv preprint arXiv:1809.04157*, 2018. [5](#)
- [39] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020. [1](#), [2](#), [3](#), [6](#), [7](#), [12](#)

Appendices

A. Hyperparameters

Representation learning stage For CIFAR-100, we use SGD to train the model with batch size 128, weight decay 0.0005. We adopt the warmup strategy with the ending learning rate 0.1 for 10 epochs. After the warmup, we run the SGD with 160 epochs and the learning rate decays at 100, 120 epochs with 0.1. For ImageNet100 and ImageNet1000, we adopt SGD with batch size 256, weight decay 0.0005. We also use the same warmup strategy as in CIFAR100. After the warmup, model is trained for 120 epochs. Learning rate starts from 0.1 and decays by 0.1 rates after 30, 60, 80, and 90 epochs.

For the coefficients in the loss function, λ_a is set to 1 for all experiments in the paper. λ_s is tuned to ensure our learned model has comparable number of parameters to other methods for fair comparison. Moreover, for simplicity, we set the same λ_s for every steps in each experiment. For experiments of CIFAR100-B0, λ_s is set as 0.75. For experiments of CIFAR100-B50, λ_s is set as 0.25. For experiments on ImageNet100 and ImageNet, λ_s is set as 0.75 for ImageNet100-B0, 0.5 for ImageNet100-B50 and 0.75 for ImageNet.

Classifier learning stage We adopt SGD optimizer with weight decay 0.0005 to update the classifier only for 30 epochs with SGD optimizer. Learning rate is 0.1 and decays with 0.1 rate at 15 epochs. The temperature of cross-entropy loss are set as $\delta = 5$ for CIFAR-100 and $\delta = 1$ for ImageNet-100 and ImageNet-1000.

B. Sensitive Study of Hyper-parameters

We conduct a sensitive study of our method on CVIFAR100-B0 10 steps with different λ_a . The results are shown in Table 6, which demonstrates our method is robust to λ_a . We also conduct experiments for different λ_s , which is shown in the Figure 1 in the main body.

λ_a	0.1	0.5	1	5	10
Avg	74.12 \pm 0.06	74.41 \pm 0.16	74.64 \pm 0.28	74.52 \pm 0.33	73.54 \pm 0.22

Table 6: Sensitive study on effects of λ_a

C. The Quality of Decision Boundary

In this section, our goal is to verify that the high-quality linear classifier can be obtained even re-learning the old classes’ decision boundary with memory \mathcal{M} . Specifically, we compare our method with an ‘ideal’ strategy that uses all the previous data to train the classifier in the second

stage. Such an upper bound achieves $76.14 \pm 0.80\%$ on the CIFAR100-B0 10 steps, which is only slightly higher than our method ($74.64 \pm 0.28\%$). We also observed similar results on the other benchmarks, which show the efficacy of our second-stage learning.

D. Latency

Regarding inference latency, we conduct an experimental comparison on the ImageNet with GTX 1080Ti. Our method achieves 1.07ms/image, which is comparable to other baseline methods, such as BiC and WA, which are 0.99ms/image.

E. Results for modified 32-layer ResNet

Most works use a modified 32-layer ResNet following iCaRL[27]. We also compare the results of our method with other methods based on the modified 32-layer ResNet. The results on CIFAR100-B0 are shown in Table 7 and the results on CIFAR100-B50 are shown in Table 8. The results of other methods are reported in their papers. It can be found that our method still outperforms other methods on both CIFAR100-B0 and CIFAR100-B50 even with a small network like modified ResNet-32.

F. More detailed results on CIFAR100

Figure 5 shows the performance with respect to steps on CIFAR100-B0 with 5 incremental steps and 10 incremental steps and CIFAR100-B0 with 2 incremental steps. This also illustrates the superiority of our method.

G. Detailed results on ImageNet

We also show the curves of performance with respect to steps on ImageNet100-B0, ImageNet100-B50 and ImageNet1000-B0 in Figure 6, which proves the effectiveness of our method on complex datasets.

Methods	5 steps		10 steps		20 steps		50 steps	
	#Paras	Avg	#Paras	Avg	#Paras	Avg	#Paras	Avg
iCaRL [27]	0.46	67.20	0.46	64.04	0.46	61.16	0.46	57.00
BiC [12]	0.46	68.92	0.46	66.15	0.46	63.80	0.46	-
WA [39]	0.46	70.00	0.46	67.25	0.46	64.33	0.46	-
Ours(w/o P)	1.38	73.00(+3.00)	2.53	71.29(+4.04)	4.83	71.07(+6.74)	11.73	70.58(+13.58)
Ours	0.42	72.31(+2.31)	0.52	69.41(+2.16)	0.45	68.82(+4.49)	0.70	67.29(+10.29)

Table 7: Results on CIFAR100-B0 benchmark using modified 32-layer ResNet. #Paras means the average number of parameters used during inference over steps, which is counted by million. Avg means the average accuracy (%) over steps. Ours(w/o P) means our method without pruning.

Methods	2Steps		5Steps		10Steps	
	#Paras	Avg	#Paras	Avg	#Paras	Avg
UCIR [12]	0.46	66.76	0.46	63.42	0.46	60.18
PODNet [6]	-	-	0.46	64.83	0.46	64.03
TPCIL [34]	-	-	0.46	65.34	0.46	63.58
Ours(w/o P)	0.92	70.18(+3.42)	1.61	68.52(+3.18)	2.76	67.09(+3.06)
Ours	0.32	69.52(+2.76)	0.59	67.60(+2.26)	0.61	66.36(+2.33)

Table 8: Results on CIFAR100-B50 using modified 32-layer ResNet. #Paras means the average number of parameters used during inference over steps, which is counted by million. Avg means the average accuracy (%) over steps. Ours(w/o P) means our method without pruning.

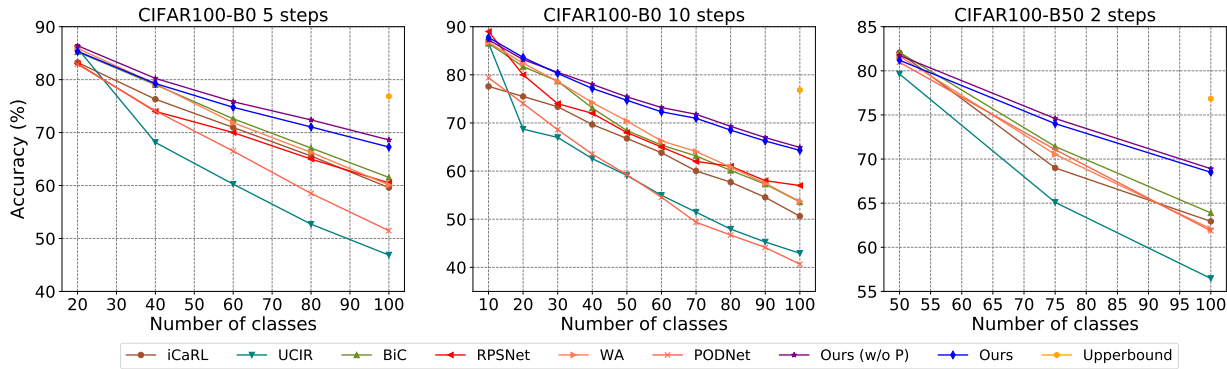


Figure 5: The performance for each step. **Left** is evaluated on CIFAR100-B0 of 5 steps. **Middle** is evaluated on CIFAR100-B0 of 10 steps. **Right** is evaluated on CIFAR100-B50 of 2 steps.

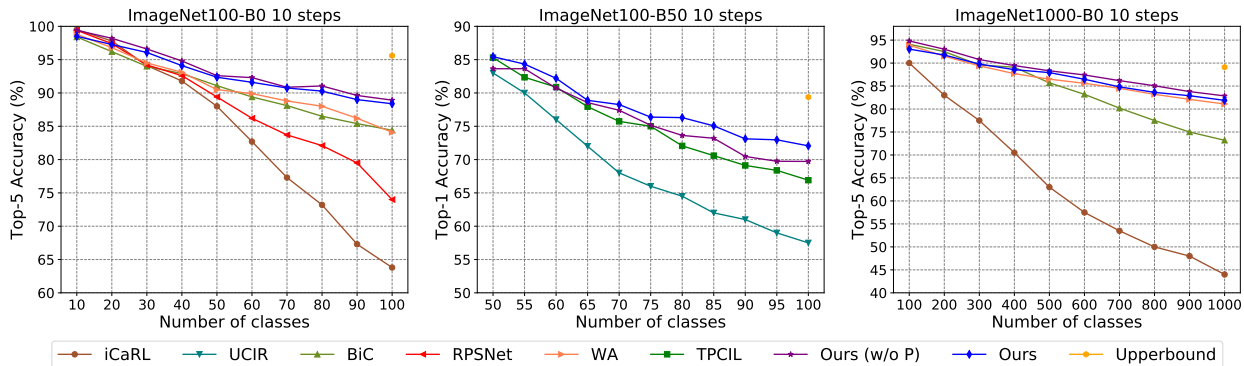


Figure 6: The performance for each step. **Left** is evaluated on ImageNet100-B0 of 10 steps. **Middle** is evaluated on ImageNet100-B50 of 10 steps. **Right** is evaluated on ImageNet1000-B0 of 10 steps.