Efficient Perturbation Inference and Expandable Network for Continual Learning

Fei Du[a], Yun Yang[b,*], Ziyuan Zhao[c], Zeng Zeng[c,d]

[a]School of Information Science and Engineering, Yunnan University, Kunming 650091, China
[b]National Pilot School of Software, Yunnan University, Kunming 650091, China
[c]Institute for Infocomm Research (I2R), A*STAR, 138632, Singapore
[d]School of Microelectronics, Shanghai University, Shanghai, China

Abstract

Although humans are capable of learning new tasks without forgetting previous ones, most neural networks fail to do so because learning new tasks could override the knowledge acquired from previous data. In this work, we alleviate this issue by proposing a novel Efficient Perturbation Inference and Expandable Network (EPIE-Net), which dynamically expands lightweight task-specific decoders for new classes and utilizes a mixed-label uncertainty strategy to improve the robustness. Moreover, we calculate the average probability of perturbed samples at inference, which can generally improve the performance of the model. Experimental results show that our method consistently outperforms other methods with fewer parameters in class incremental learning benchmarks. For example, on the CIFAR-100 10 steps setup, our method achieves an average accuracy of 76.33% and the last accuracy of 65.93% within only 3.46M average parameters.

Keywords: Continual Learning, Dynamic Networks, Class Incremental Learning, Uncertainty Inference

1. Introduction

A long-standing goal of general artificial intelligence is to develop a continual learning system that learns new tasks over time and keeps the performance of old tasks. However, most trained neural networks will inevitably override the acquired knowledge for learning new tasks, which is commonly known as catastrophic forgetting [1]. Ideally, Joint Training [2], optimizing a model on all tasks simultaneously, can effectively address the catastrophic forgetting problem, but in practice, the data of previous tasks will inevitably be discarded as tasks increase. Therefore, Continual Learning (CL) or Incremental Learning (IL) is developed to accommodate new tasks and acquired knowledge.

There has been much effort attempting to address catastrophic forgetting along different lines [3, 4, 5, 6, 7, 8]. Among them, the existing CL approaches can be roughly classified into three categories: regularization-based methods [4, 6], memory replay methods [3, 9, 10], and parameter isolation methods [8, 11, 12]. For the regularization-based branch, these works preserve prior knowledge by introducing an extra regularization term into the loss function to control the plasticity of network weights. But as new tasks continuously arrive, the available model parameters rapidly decrease, and regularization-based methods are hard to maintain the long-term trade-off between new tasks and acquired knowledge. For the memory replay methods, these works mainly focus on storing explicit or implicit data in memory to review prior knowledge. Although memory replay methods can effectively mitigate catastrophic forgetting, these methods are heavily dependent on the amount of data that can be saved and also need to consider the imbalance between new data and memory exemplars [13]. As for the parameter isolation methods, they assign different sub-model parameters to specific tasks to prevent forgetting. But as the model grows, the dynamic architecture may become redundant and inefficient [8, 11]. In human learning, compressing acquired knowledge, taking notes, and expanding the boundaries of new knowledge are unified and instinctive behaviors. Therefore, in recent research work, the above three types of continual learning methods are usually integrated to build a powerful model to defy forgetting.

The core capability of CL models is to effectively maintain the right trade-off between new concepts and acquired knowledge, also known as the stability-plasticity dilemma. In detail, although high stability can effectively retain previous knowledge, it also impedes the acquisition of new knowledge. On the contrary, models with high plasticity, while adaptable to novel knowledge, often cause catastrophic forgetting. To address this problem, one of the intuitive thoughts is to assign isolated parameters to tasks by dynamically expanding the network. However, it also raises two problems: how to reduce parameter redundancy in long-term dynamic growth and identify the task branches corresponding to the data at inference?

Around reducing memory overhead, recent works [8, 14, 15, 16, 17, 18] utilize pruning [19] techniques to compress the model or re-arrange internal structures by regularization or knowledge distillation [20]; however, they also add a lot of extra post-processing time and training tricks. Although dynamically expandable networks have an inherent advantage in mitigating catastrophic forgetting, the task-ID must be given

*Corresponding author
Email address: yangyun@ynu.edu.cn (Yun Yang)

at inference. Previous work oversimplified the problem by assuming that having an oracle provided such prior knowledge to the model. More recently, [8, 11, 14, 21] have started addressing the more difficult class-incremental learning setting (without task-ID at inference), yet, these strategies only concatenate all sub-classifiers into a single classifier and take the highest score of merged classifiers as the best candidate. Besides, iCaRL [3] utilizes clustering [22, 23] to calculate the similarity of mean-of-exemplars. StackNet [24] utilizes GANs to generate the task-related feature. And Expert gate [7] uses autoencoder as a task control gate to learn task correlation. But these methods also bring additional parameter overhead.

In this work, we aim to design a more efficiently expandable architecture for class-incremental learning (CIL). The proposed Efficient Perturbation Inference and Expandable Network (EPIE-Net) consist of two parts: (1) a lightweight multi-scale hierarchical encoder is used to embed the image into the shared latent space, and (2) some dynamic lightweight task-specific multi-scale hierarchical decoders forward the latent code to the task-specific classifying space. The lightweight multi-scale shared encoder and task-specific decoder framework can effectively alleviate the parameter redundancy problem of Dynamic growing models. Moreover, we propose a mixed-label uncertainty learning strategy to alleviate the representation drift phenomenon in continual learning and utilize the average probability of perturbed samples at inference to reduce the uncertainty of the model.

Our method is mainly evaluated on CIFAR10, CIFAR100, and ImageNet-Sub datasets. Experimental results show that our method consistently outperforms other state-of-the-art methods with fewer parameters. The main contributions of our work can be summarized as follows:

- We propose lightweight multi-scale hierarchical encoder/decoder architecture to achieve efficient growth for continual learning.
- The proposed mixed-label uncertainty learning strategy can effectively improve the robustness of the model.
- The proposed mean perturbation inference can be used as a plug-and-play module to boost the prediction accuracy of the model.
- Our approach achieves a new state-of-the-art with fewer parameters on CIL and blurry-CIL setups.

The rest of the paper is organized as follows. We discuss related work in Section 2. The Efficient Perturbation Inference and Expandable Network (EPIE-Net) is detailed in Section 3, followed by experiments in Section 4. Finally, we summarize this paper and discuss some potential research directions in Section 5.

2. Related work

The EPIE-Net framework builds on the insights of multiple earlier attempts to address class-incremental learning. In the section, we summarize the most related ones to our work.

2.1. Dynamically Architectures for Continual Learning

A major trend in CL research has been proposing dynamical network architectures to cope with the growing learning tasks. Rusu et al. [12] propose a progressive network structure that laterally connects new branches to previously networks, but the complexity of the model increases quadratically with the tasks. Aljundi et al. [7] use an autoencoder as a task control gate to learn task correlation, but this method is parameter-consuming because an additional classification network and an autoencoder are constructed at each stage of CL. PathNet [25] uses evolutionary algorithms to find a specific path for each learning task in a super network and reuses these paths in new tasks to speed up learning. However, this method cannot incrementally learn new classes, and the use of evolutionary algorithms is relatively inefficient. Moreover, RPS-Net [11] combines a variety of methods such as path selection strategies, knowledge distillation, and retrospection to learn a dynamic growth model for class-incremental setting. Yet, previous works induce dramatic parameter overhead in long-term growth. To have less parameter redundancy, Kim et al. [24] propose StackNet that dynamically expands filters for new tasks and utilizes GAN as an index module to distinguish the origin of a given input sample. DER [8] utilizes a two-stage learning approach to expand an additional representation extractor with a differentiable channel-level mask-based pruning strategy and then retrain the classifier with memory data. Unfortunately, DER is hyperparameter sensitive and adds a lot of extra post-processing time. Simple-DER [14] uses a unified pruning method to reduce the hyperparameter's selection, but the compression efficiency is lower than DER. More recently, DyTox [21] directly adopts a transformer-based lightweight encoder/decoder architecture to add task-specific decoders for the new task. In a similar spirit, we design a multi-scale hierarchical convolutional encoder/decoder architecture that allows for more significant parameter savings. Compared with DER, EPIE-Net does not need to design a pruning strategy to compress the model, which saves training time and reduces the model instability caused by pruning.

2.2. Mixed-Label Data Augmentation

Mixed-label data augmentation technique, which mixes multiple images and their labels to encourage model learning smooth prediction, has been adopted successfully to improve the performance of deep learning models in various tasks. For instance, MixUp [26] generates new vicinity samples and their labels by computing the convex combination of pairs of samples. To be specific, it does a pixel-level interpolation between images and linear interpolation between one-hot labels. Moreover, MixUp variants [27] perform feature-level interpolation to encourage neural networks to softer prediction on interpolations of hidden representations. However, MixUp-based method samples are locally ambiguous and unnatural. To handle this issue, CutMix [28] combines two images by replacing the image region with a patch from another training image and does linear interpolation between one-hot labels. Other MixUp-based methods have been recently explored in [29, 30, 31, 32]

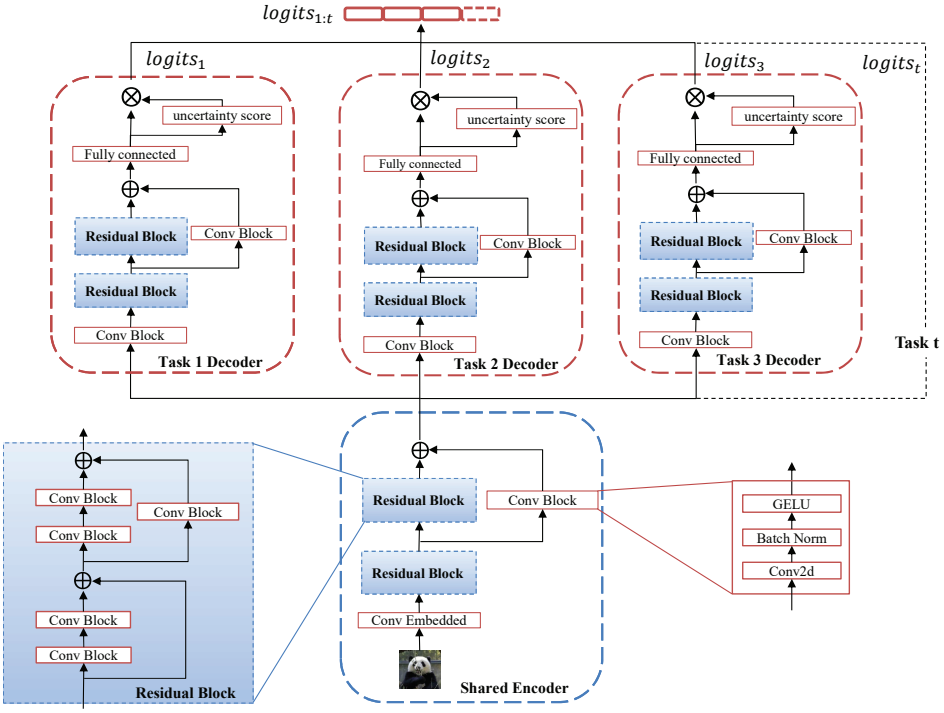

Figure 1: An overview of our EPIE-Net: The network architecture comprises a shared lightweight encoder (In Section 3.2.1) that maps images into the middle-level representation, and some dynamic lightweight task-specific decoders (In Section 3.2.2) that forward the representation to task-specific classifying space. All output embeddings are finally concatenated together as one classifier.

to improve the robustness of feature extractors. However, these methods mainly mix images or features in an unsupervised manner. In the class-incremental learning context, Bang et al. [33] introduce mixed-label data augmentation to enhance the diversity of exemplars and alleviate the side effects caused by the change of class distribution between current tasks and previous task. Our work differs from the previous works: we redesign the mixed-label data augmentation to construct local and global mixed images and their smooth labels to approximate the uncertainty of class distribution. Our label-mixed data augmentation can better learn global and local consistency to improve the robustness of representations.

3. Our Method

Our goal is to learn a lightweight model that can efficiently expandable grow parameters to handle class-incremental learning problems. Next, we present the formulation of class incremental learning in Section Section 3.1 to explain the class-incremental learning setup. Then we introduce the lightweight multi-scale hierarchical convolutional encoder/decoder architecture in Section 3.2 to construct the EPIE-Net framework. After this, we describe how to use a two-stage training strategy to train the EPIE-Net framework In Section 3.3.

3.1. Class Incremental Learning Setups

We consider the case of continual learning where tasks and their corresponding data arrived one after another in a sequential matter. The term "task" in continual learning refers to an isolated training phase where the coming data can belong to a different set of classes, a domain shifted dataset, or a new output space. Specifically, Class Incremental Learning (CIL) setup demands model training with multi-phases classification datasets $\{X^t, Y^t\}$. At any t training phase, $x^t \in X^t$ is the input image and $y^t \in Y^t$ is the corresponding label. The CIL setup is strictly required that training labels at all phases do not intersect. Although it can effectively assess catastrophic forgetting, it deviates from the real world where arrived data do not contain new classes exclusively. For more practical applications, [34, 33] consider a more general and realistic blurry-CIL setup where the previous classes will still appear with a lower probability in subsequent learning. The blurry-CIL setup makes the task boundaries vague and requires that each task is given sequentially as a stream. Following Bang et al. [33], we formulate either blurry or disjoint CIL setups by intersecting labels or not.

$$\begin{aligned} disjoint - CIL &\Rightarrow Y^{t-1} \cap Y^t = \emptyset \\ blurry - CIL &\Rightarrow Y^{t-1} \cap Y^t \neq \emptyset \end{aligned} \tag{1}$$

For both blurry and disjoint CIL setups, In the t testing phase, the model needs to predict all seen test classes $Y^t_{all} = \bigcup_{i=1}^{t} Y^i$. Note that class-incremental learning allows holding a few old samples (belonging to tasks 1 to $t-1$) as rehearsing data in memory to help prevent model forgetting classes.

3.2. Lightweight Multi-scale Hierarchical Encoder/Decoder Architecture for Class Incremental Learning

The proposed EPIE-Net comprises a shared encoder that maps images into the middle-level representation and some dy-

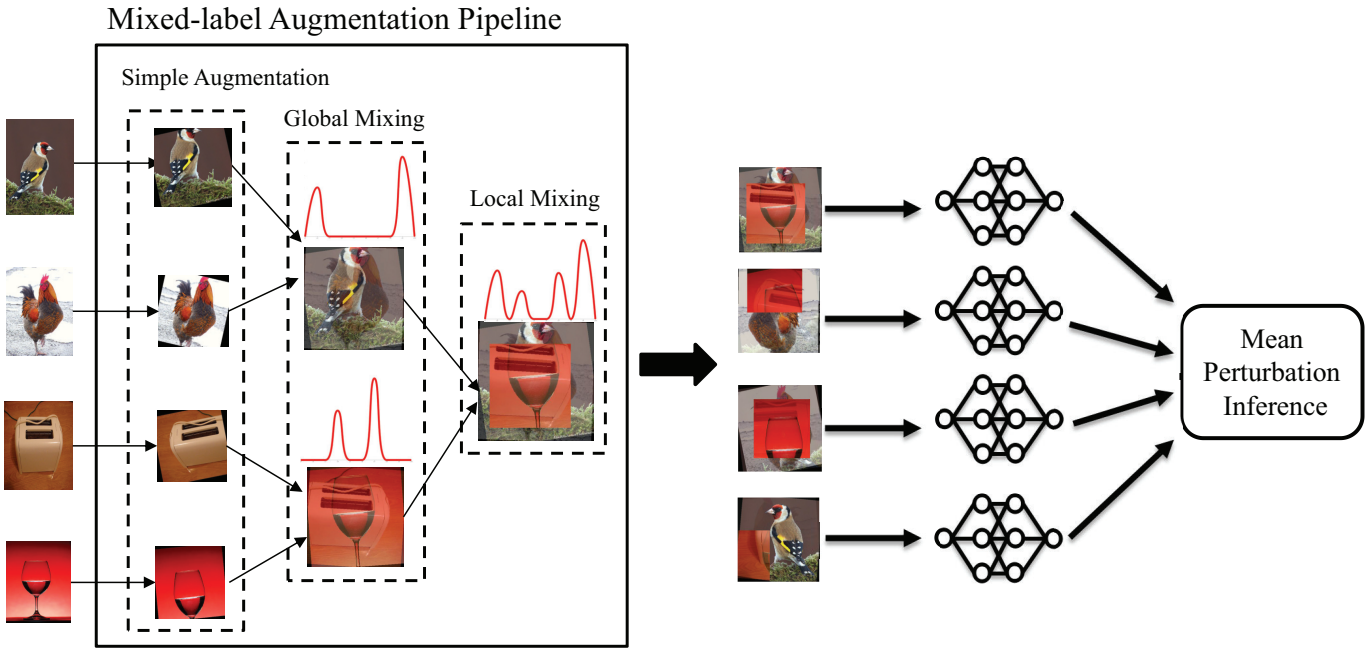

Figure 2: Mixed-label Uncertainty Learning. Images are augmented into n mixed images by our mixed-label augmentation strategy. They are then input into the network to calculate the average probability of these mixed images.

namic task-specific decoders that forward the representation to task-specific classifying space. For example, the panda photo of the new task is first input to the shared encoder for feature extraction. Then the middle-level representation is input to each task-decoder, respectively. Finally, we concatenate the output layers of all task decoders as a long embedding vector. Contrary to previous transformer-based architecture, we use lightweight multi-scale hierarchical convolution modules to achieve efficient growth.

3.2.1. Shared multi-scale hierarchical encoder

As opposed to the transformer-based encoder, for the sake of saving parameters, we use a multi-scale hierarchical encoder to extract features. The main motivation behind this is to fuse information from multi-level features into hidden variables, which can help the decoder to capture specific correlations of different level features with tasks in the subsequent classification. As shown in Figure 1, the encoder is a redesigned ResNet-like structure that consists of an embedding layer, two residual blocks for multi-level feature extraction, and a convolution layer for adapting low-level feature sizes to high-level features. Each convolution block contains three operations: convolution, batch normalization [35] and GELU [36] activation. We use an additive operation to fuse different level features.

3.2.2. Dynamic multi-scale hierarchical decoder expansion

As shown in Figure 1, our decoder also uses a lightweight multi-scale hierarchical convolutional module for specific tasks. To accommodate the growth of tasks, our idea is to expand the parameter space by creating a new decoder while keeping the previous task decoder. Thus, after training t tasks, we have t task-specific decoders ($\theta_i \quad for \quad i \in \{1 \dots t\}$). Given an image x, belonging to any of the seen tasks $\{1 \dots t\}$, our model first processes it through the shared encoder to get the hidden variables h. Then the h is passed to each decoder to obtain the task-specific embedding e_i.

We expect the task-specific decoders can not only predict the current task data with low entropy but also make high-entropy predictions on other task data. For example, if a natural image containing a bird is fed into a cat/dog decoder, the model should produce an uncertain result. Specifically, we design a task entropy module to formulate the uncertainty u of the model:

$$\mu = \frac{-\sum_{i=1}^{C} p_i log(p_i)}{log(C)} \tag{2}$$

Here, p_i is the probability of the i-th dimension, and C is the class number for a given task. We multiply the output of each model by its uncertainty score and then concatenate all embeddings together into one classifier.

$$out = concat([e_1 \cdot \mu_1, e_2 \cdot \mu_2, ..., e_t \cdot \mu_t]) \tag{3}$$

3.3. Two-stage Training Pipeline for EPIE-Net

As shown in Figure 3, EPIE-Net decouples the incremental training process into two sequential stages. In the first training stage, we expand a lightweight decoder to learn new classes using a mixed-label uncertainty learning method. And then, we finetune all decoders with class-balanced memory data for the second training stage.

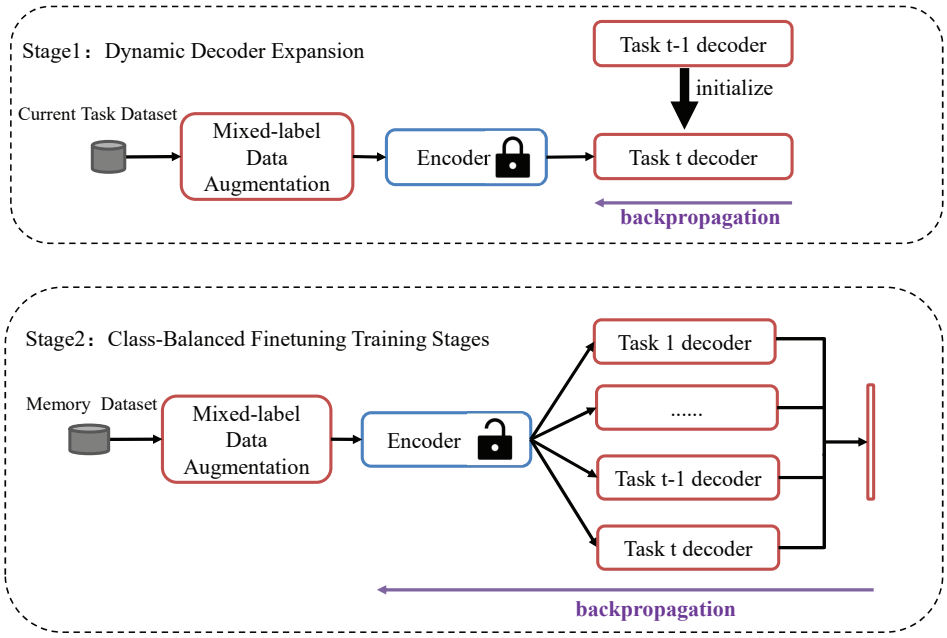

Figure 3: Two-stage Training Pipeline for EPIE-Net. We first fix the learned encoder and expand a lightweight decoder to learn new classes at stage one training. For the second stage of training, we concatenate the output layers of all task-specific decoders as a long embedding vector and then finetune EPIE-Net with memory data M^t.

3.3.1. *Mixed-label Uncertainty Learning for Dynamic Decoder Expansion*

At any t training step, we first fix the learned encoder and task-specific decoders in previous training phases and then expand a lightweight decoder to learn new classes. However, training the task-specific decoder may cause bias. Thus, we propose a mixed-label uncertainty learning method to improve the robustness of the task-specific decoder.

Following [37, 33], we use the Monte-Carlo (MC) method to approximate the uncertainty of the distribution $p(y = c|x)$.

$$\begin{aligned} p(y = c|x) &= \int p(y = x|\tilde{x}_i) \\ &\approx \frac{1}{m}\sum_{y=1}^{m} p(t = c|\tilde{x}_i), \tilde{x} \sim aug(x) \end{aligned} \tag{4}$$

where $\tilde{x}$ denote an augmented sample of x. By default, the hyperparameter m is set to 2.

Unlike the simple augmentation method such as rotation and scaling. Our mixed-label data augmentation pipeline, as shown in Figure 2, consists of three consecutive steps. Firstly, one batch images are expanded into m augmented batches by general data augmentation such as simple rotation, translation, flipping, and scaling. Then a global mixing technique MixUp [26] is used for mixing m augmented batches and their labels by linear-interpolation, respectively. Finally, a local mixing technique CutMix [28] is further to process the result of global mixing by replacing the mixed image region with a patch from another mixed image.

MixUp is a global mixed-label method for generating new vicinity samples by computing the convex combination of two images of different classes. For a pair of two samples and their labels probabilities $(x_i; p_i)$ and $(x_j; p_j)$, we calculate $(x'; p')$ by

$$\begin{aligned} \lambda &\sim Beta(\alpha, \alpha), \\ x' &= \lambda x_i + (1 - \lambda)x_j, \\ p' &= \lambda p_i + (1 - \lambda)p_j. \end{aligned} \tag{5}$$

where λ is sampled from a Beta distribution parameterized by the α hyper-parameter.

Different from MixUp, CutMix combines two images by locally replacing the image region with a patch from another training image. We define the combining operation as

$$\begin{aligned} \tilde{x} &= \boldsymbol{M} \odot x'_i + (\boldsymbol{1} - \boldsymbol{M}) \odot x'_j, \\ \tilde{p} &= \lambda p'_i + (1 - \lambda)p'_j, \end{aligned} \tag{6}$$

where $\boldsymbol{M} \in [0, 1]^{W \times H}$ denotes the randomly selected pixel region for the image x'_i and fill in x'_j, 1 is a binary mask filled with ones, and $\odot$ is element-wise multiplication. To be specific, we sample the bounding box coordinates $B = (r_x, r_y, r_w, r_h)$ indicating the cropping regions on x'_i and x'_j. The box coordinates are uniformly sampled according to

$$\begin{aligned} r_x &\sim Uniform(0, W), r_w = W\sqrt{1 - \lambda} \\ r_y &\sim Uniform(0, H), r_h = H\sqrt{1 - \lambda} \end{aligned} \tag{7}$$

where λ is also sampled from the Beta distribution $Beta(\alpha, \alpha)$. After constructing the augmented data pair $(\tilde{x}; \tilde{p})$, we calculate the mixed-label uncertainty cross-entropy loss:

$$£ = -\frac{1}{N \times m}\sum_{n=1}^{N}\sum_{i=1}^{m} \tilde{p}_i^n log(f(\tilde{x}_i^n)) \tag{8}$$

Algorithm 1 Learning algorithm of EPIE-Net

Input: current task dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, rehearsal data M, *encoder*, previous $decoders = \{decoder_i, ..., decoder_{t-1}\}$, perburtation times k, $epoch_1$, $epoch_2$.
Output: *encoder* and *decoders* of EPIE-Net

$decoder_t \leftarrow$ initialize a decoder for current task t
for i in $epoch_1$ **do** // new decoder training stage
  $(\mathbf{x}, \mathbf{y}) \leftarrow$ sample a batch from dataset D.
  $[(\mathbf{x'_1}, \mathbf{y'_1})..., (\mathbf{x'_k}, \mathbf{y'_k})] \leftarrow$ Augment $(\mathbf{x}, \mathbf{y})$ k times.
  $[\mathbf{feat_1}, ..., \mathbf{feat_k}] \leftarrow encoder([\mathbf{x'_1}, ..., \mathbf{x'_k}])$
  $[\mathbf{p_1}, ..., \mathbf{p_k}] \leftarrow decoder_t([\mathbf{feat_1}, ..., \mathbf{feat_k}])$
  $loss \leftarrow £([\mathbf{p_1}, ..., \mathbf{p_k}], [\mathbf{y'_1}, ..., \mathbf{y'_k}])$
  Update $decoder_t$ through gradient descent
end for
$decoders.append(decoder_t)$ // Adds the $decoder_t$ to the previous decoders set.
for i in $epoch_2$ **do** // class-balanced finetuning stage
  $(\mathbf{x}, \mathbf{y}) \leftarrow$ sample a class-balanced batch from M.
  $[(\mathbf{x'_1}, \mathbf{y'_1})..., (\mathbf{x'_k}, \mathbf{y'_k})] \leftarrow$ Augment $(\mathbf{x}, \mathbf{y})$ k times.
  $[\mathbf{feat_1}, ..., \mathbf{feat_k}] \leftarrow encoder([\mathbf{x'_1}, ..., \mathbf{x'_k}])$
  for *decoder* in *decoders* **do**
    $[\mathbf{em_1}, ..., \mathbf{em_k}] \leftarrow decoder([\mathbf{feat_1}, ..., \mathbf{feat_k}])$
    $outList.append([\mathbf{em_1}, ..., \mathbf{em_k}])$
  end for
  $[\mathbf{out_1}, ..., \mathbf{out_t}] \leftarrow$ concatenate outList to long vector
  $loss \leftarrow £([\mathbf{out_1}, ..., \mathbf{out_t}], [\mathbf{y'_1}, ..., \mathbf{y'_k}])$
  Update *encoder* and *decoders* through gradient descent
end for
return *encoder* and *decoders* of EPIE-Net

where m, N, $f(\cdot)$ and $\tilde{p}$ denote augmented number, batch size, predicted probability and label mixing probabilities of sample x, respectively.

3.3.2. Class-Balanced Finetuning Training Stages

After the newly expanded decoder is trained, the next question is how to combine all task-specific decoders to recognize all seen classes? Unlike previous works [8] that fix extractors and retrain a classifier, we concatenate the output layers of all task-specific decoders as a long embedding vector and finetune EPIE-Net with memory data M^t. Note that previous work [5, 13, 38] has shown that the imbalance between new tasks and memory data significantly affects the performance of continual learning. We thus design a simple new-old class rebalancing sampling strategy to train the merged network. Assuming that a constant number of U classes are observed in each task, we rebalance current and rehearsal data in each sampling batch:

$$b_r = B \times \frac{\sum_{t=1}^{T-1} U_t}{\sum_{t=1}^{T} U_t} \qquad b_c = B - b_r \tag{9}$$

where U_t, B, b_r, and b_c denote the classes number of t-th task, batch size hyperparameter, rehearsal batch, and current batch, respectively. Note that M^t is just randomly selected from the previous data stream. Unlike the previous memory strategies saving fixed "key" data in memory, we found that the well-designed fixed data could lead to overfitting, so we randomly selected class-balanced data into the memory buffer.

Mean Perturbation Inference. Typically, data augmentation techniques are removed during the inference phase to reduce noise in model predictions. However, we find that the augmented data can amplify the uncertainty of decoders for non-corresponding tasks. Therefore, we calculate the mean probability of perturbed samples at the inference phase:

$$p_{mean} = \frac{1}{m} \sum_{i=1}^{m} p(\tilde{x}_i), \tilde{x} \sim aug(x) \tag{10}$$

where $\tilde{x}$ denote a perturbed sample that is rotated, translated, and scaled from the original x. By default, the hyperparameter m is set to 2.

4. Experiments

4.1. Implementation Details

In this section, we compare EPIE-Net with several state-of-the-art methods in various CIL experimental setups. We also conducted a series of ablation studies to more fully assess the importance of each component of EPIE-Net.

Experimental setup. Our experiments are mainly carried out as CIL and blurry-CIL setups. In the CIL experiments, we follow Rebuffi et al. [3] evaluation protocols that split dataset evenly in different increments, and the learned classes do not appear in subsequent learning (unless it is in the memory). In the blurry-CIL experiments, following Bang et al. [33], the arrived data are composed of the majority of new classes and minor seen classes. We denote blurry-CIL setup as 'BlurryM' where the M denotes the portion of the previous classes (i.e., Blurry10 means that 90% of the coming data are new classes, and the remaining 10% are known). In addition, we also consider online and offline setups. In offline setups, the incoming samples can be trained multiple epochs until the model converges. However, online setups require that samples are trained only once.

Evaluation Metrics. For the evaluation method, we use three popular metrics in the literature, such as last accuracy, last forgetting and average incremental accuracy $\bar{A}$. following Rebuffi et al. [3], $\bar{A}$ is defined as:

$$\bar{A} = \frac{1}{T} \sum_{i=1}^{T} \bar{a}_i \tag{11}$$

where $\bar{a}_i$ is the average accuracy on dataset $D_{1:i}^{test}$ when the model learned task i.

Dataset. We use CIFAR10/100 [41] and ImageNet-Sub [3] datasets to configure CIL setups for evaluations. We randomly split and assign with different random seeds a set of all classes into n tasks to generate CIL task setup. Following [3], ImageNet-sub datasets are randomly 100 classes sampled from ImageNet [42] with an identical random seed (1993).

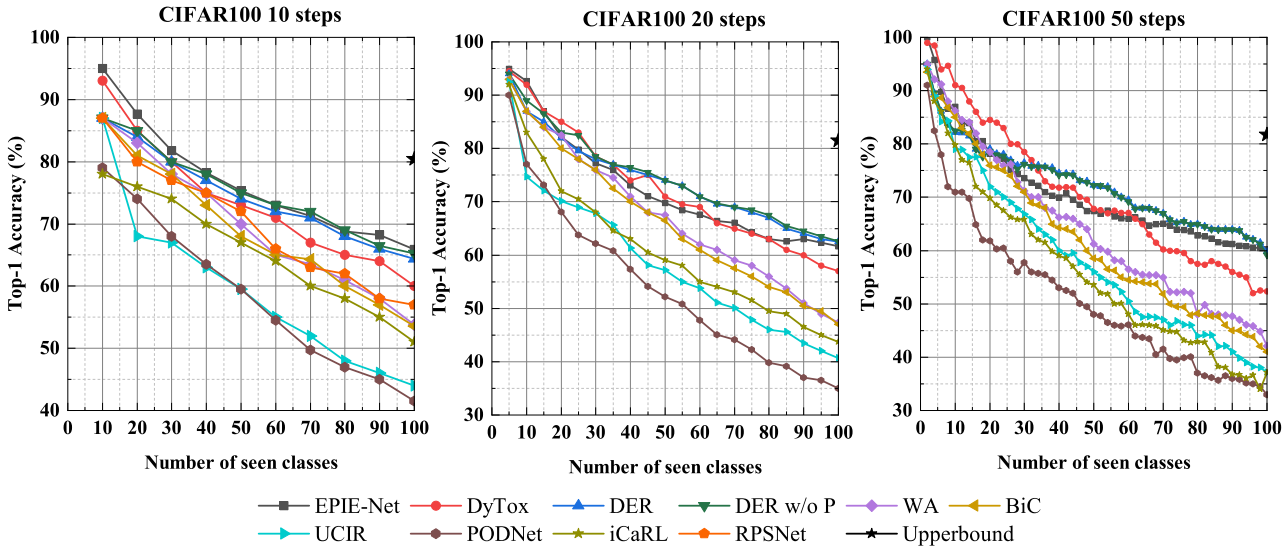

Figure 4: Experimental results of class-incremental training at each step on CIFAR-100. We report top-1 accuracy (%) after learning each step. Left is evaluated with 10 steps, middle with 20 steps, and right with 50 steps.

	10 steps			20 steps			50 steps		
methods	#Paras	Avg Acc	Last Acc	#Paras	Avg Acc	Last Acc	#Paras	Avg Acc	Last Acc
Joint	11.22	-	80.41	11.22	-	81.49	11.22	-	81.74
iCaRL [3]	11.22	65.27±1.02	50.74	11.22	61.20 ±0.83	43.75	11.22	56.08±0.83	36.62
UCIR [13]	11.22	58.66±0.71	43.39	11.22	58.17±0.30	40.63	11.22	56.86±0.83	37.09
BiC [5]	11.22	68.80±1.20	53.54	11.22	66.48±0.32	47.02	11.22	62.09±0.85	41.04
WA [39]	11.22	69.46±0.29	53.78	11.22	67.33±0.15	47.31	11.22	64.32±0.28	42.14
PODNet [40]	11.22	58.03±1.27	41.05	11.22	53.97±0.85	35.02	11.22	51.19±1.02	32.99
RPSNet [11]	56.5	68.6	57.05	-	-	-	-	-	-
DER w/o P [8]	last 112.2 avg 61.6	75.36±0.36	65.22	last 224.5 avg 117.6	**74.09**±0.33	62.48	last 561.3 avg 285.6	**72.41**±0.36	59.08
DER† [8]	avg 4.96	74.64±0.28	64.35	avg 7.21	73.98±0.36	**62.55**	avg 10.15	72.05±0.55	59.76
DyTox [21]	last 10.73	73.66±0.02	60.67±0.34	last 10.74	72.27±0.18	56.32±0.61	last 10.77	70.2±0.16	52.34±0.26
EPIE-Net(ours)	**last 5.16** **avg 3.46**	**76.33**±0.31	**65.93**±0.26	**last 5.91** **avg 3.78**	72.35±0.28	61.69±0.35	**last 7.04** **avg 4.18**	70.68±0.33	**60.23**±0.37

Table 1: Results on CIFAR100 benchmark. All results are averaged over three runs. Avg Acc means the average incremental accuracy (%). Note that baselines results come from [21, 8]. DER† [8] means require setting-sensitive pruning operation.

Implementation details. For each task, we use Adam [43] optimizer to train the model with batch size 256, weight decay 0.0001, and 200 epochs. The learning rate starts from 0.001 and decays by cosine annealing scheduler [44]. Note that our approach requires different learning rates in two sequential stages: in the decoder expansion stage, the learning rate starts from 0.001, while in the class-balanced fine-tuning stage, the initial learning rate is 0.0005×0.9^i and decays with cosine annealing scheduler, here i is the current task number. Following [3, 8, 21], the memory buffer of CIFAR100 and ImageNet-100 are both $K = 2000$.

4.2. Results on Offline CIL

In this part of the experiments, we address the offline CIL setup where labels at all phases do not intersect, and the model can be trained multiple epochs until it converges. We extensively compare the proposed technique with existing state-of-the-art methods, including iCaRL [3], UCIR [13], BiC [5], WA [39], POD-Net [40], RPS-Net [11], DER [8] and DyTox [21].

Evaluation on CIFAR100. Table 3.3.2 summarizes the experimental results on CIFAR100 for all approaches. DER w/o P means the DER without pruning, and DER† is to set sensitive hyperparameters to prune. #Paras (counted by million) means the average parameters overhead during inference over steps. Note that the parameter overhead of DyTox was reported as the final parameters count, and we likewise report it for a fair comparison with DyTox. We can see that EPIE-Net consistently has minimal parameter overhead at different incremental splits. Compared to the pruned DER†, EPIE-Net does not require setting-sensitive pruning operation, which can save much training time and is more suitable for practical application. Specifically, for the difficult 50 steps setup, our method reaches 70.68% in "Avg" accuracy and 60.23% in "Last" accuracy within 7.04M parameters. This outperforms the previ-

	ImageNet100-B0					ImageNet100-B50		
	#Paras	top-1		top-5		#Paras	top-1	
Methods		Avg Acc	Last Acc	Avg Acc	Last Acc		Avg Acc	Last Acc
ResNet18 joint	11.22	-	-	-	95.1	11.22	81.2	81.5
E2E [38]	11.22	-	-	89.92	80.29	-	-	-
WA [39]	11.22	-	-	91	84.1	-	-	-
UCIR [13]	-	-	-	-	-	11.22	68.32	57.3
iCaRL [3]	11.22	-	-	83.6	63.8	11.22	59.88	50.3
BiC [5]	11.22	-	-	90.6	84.4	11.22	64.96	56.7
DER w/o P [8]	61.6	77.18	66.7	93.23	87.52	67.20	78.20	74.92
DER [8]	7.67	76.12	66.06	92.79	88.38	8.87	77.73	72.06
DyTox [21]	11.01	76.53	67.76	92.26	88.5	-	-	-
EPIE-Net(ours)	**6.72**	**79.55**	**68.55**	**93.43**	**88.64**	**6.72**	**79.52**	**75.66**

Table 2: Results on ImageNet100 benchmark. Avg Acc means the average incremental accuracy (%). Note that ImageNet100-B0 protocol [3] means training the model from scratch with 10 steps, and the ImageNet100-B50 [13] means starting from a model trained on 50 classes, and the remaining 50 classes come in 10 steps.

ous state-of-the-art DyTox (70.2% in "Avg", 52.34% in "Last") which has 10.77M parameters. Moreover, although DER w/o P reaches a higher average accuracy (72.41%), its need for 561.3M parameters is too inefficient. Since UCIR, iCaRL, and BiC adopt static network design, the parameters used to identify old classes are forced to be overwritten to learn new classes, so their overall performance deteriorates rapidly, while EPIE-Net protects learned knowledge by isolating the parameters of previous task decoders. Figure 4 shows the detailed performance for each step with different incremental steps. All models degrade as the number of learned classes increases, but EPIE-Net maintains a high performance after learning long steps, and the decline slope is lower than other models.

Evaluation on ImageNet100. Table 1 summarizes the experimental results for all approaches on ImageNet100 datasets. ImageNet100-B0 protocol [3] means training the model from scratch with 10 steps, and the ImageNet100-B50 [13] means starting from a model trained on 50 classes, and the remaining 50 classes come in 10 steps. Following [8], we compare the average accuracy and last accuracy of Top-1 and Top-5. However, since few works have reported the performance of Top5 in the ImageNetB50 setting, we remove the relevant comparison. We can see that our method consistently outperforms other methods with minimal parameter overhead.

4.3. Results on Online Blurry-CIL

In this part of the experiments, we address a realistic and real-world online blurry-CIL setup where tasks share the classes, and samples are trained only once. Following [33], we compare the proposed EPIE-Net with RM [33], Rwalk [45], iCaRL [3], GDumb [34] and BiC [5]. To adapt the online blurry-CIL setup, we firstly train task-specific decoders offline for new classes and then adds them to EPIE-Net for online learning.

Table 3 summarizes the 'Blurry10-Online' setup results on the CIFAR10 and CIFAR100 datasets. Following [33], we split each dataset into 5 steps, and in each step, 90% of the coming data are new classes and the remaining 10% are known. We use Last Accuracy and Last Forgetting metrics to compare the models. Since we use different decoders for each task, EPIE-Net preserves learned knowledge more effectively than RM on the Online Blurry-CIL setting. We can observe that EPIE-Net outperforms other methods even with fewer parameters. Specifically, under the CIAR100 online setups, EPE-Net can achieve 60.31% accuracy, which surpasses the state-of-the-art RW by 8.94% last accuracy.

4.4. Ablation Studies and Analysis

To further analyze the mechanism of EPIE-Net, a large number of ablation experiments are performed in this section to evaluate the contribution of each component of EPIE-Net more comprehensively.

The effect of reserved exemplars. It has been proven that pre-reserved exemplars are useful in maintaining the performance of old classes [13, 3]. We use the mixed-label uncertainty learning strategy to improve the generalization performance of representations so that EPIE-Net can reduce the requirement of rehearsal samples compared with other continual learning models. Figure 5 illustrates the effect of EPIE-Net on different reserved exemplars. Both methods have improved performance as the number of reserved exemplars increases, and EPIE-Net can achieve high performance with fewer exemplars. In particular, EPIE-Net merely uses 100 exemplars (1 exemplar per class), and the average incremental accuracy reaches 63.89%, which exceeds the 68.33% achieved by the iCaRL with 2000 exemplars.

The effect of model size. The trade-off between performance and parameter capacity for dynamically growing networks is a dilemma. Thanks to the proposed efficient task decoder module, EPIE-Net only needs to grow 0.1M parameters per task (CIFAR-100 with 10 class increments). As shown in Figure 6, We conduct experiments to study the effect of model size on performance. Our method can achieve high performance with only a small number of parameters. Specifically, RPS requires 56.5M parameters to achieve an average incremental accuracy of 68.6%, while our model requires only 2.04M parameters to achieve an accuracy of 73.2%. In the case of unrestricted parameters, DER without pruning requires 64M average parame-

Methods	CIFAR10 (K=500)			CIFAR100 (K=2000)		
	#Parms	Online		#Parms	Online	
		Last Accuracy(↑)	Last Forgetting(↓)		Last Accuracy(↑)	Last Forgetting(↓)
EWC	11.2	55.65±4.60	16.06±3.89	22.4	26.95±0.36	11.47±1.26
Rwalk	11.2	53.66±3.18	17.04±0.31	22.4	32.31±0.78	15.57±2.17
iCaRL	11.2	45.98±3.04	4.75±1.70	22.4	17.39±1.04	5.38±0.88
GDumb	11.2	49.47±1.08	1.44±2.77	22.4	27.19±0.65	7.49±0.88
BiC	11.2	42.06±2.41	1.34±2.27	22.4	13.01±0.24	4.63±0.46
RM	11.2	71.13±0.25	-0.85±0.28	22.4	41.35±0.95	4.99±0.89
EPIE-Net (ours)	**4.11**	**72.11**±0.15	0.76±0.12	**4.11**	**60.31**±0.55	**2.45**±0.67

Table 3: Blurry10-Online setup results on the CIFAR10 and CIFAR100 datasets. Following [33], we split the dataset into 5 steps and then compare the model last accuracy (%) and last forgetting (%). Note that baselines results come from [33].

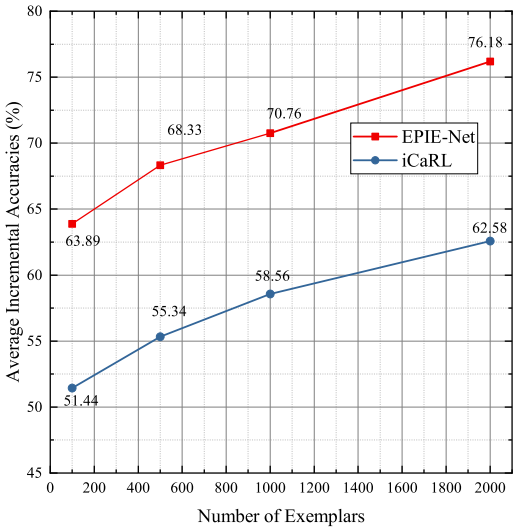

Figure 5: Ablation analysis on CIFAR-100 with 10 class increments for different exemplars.

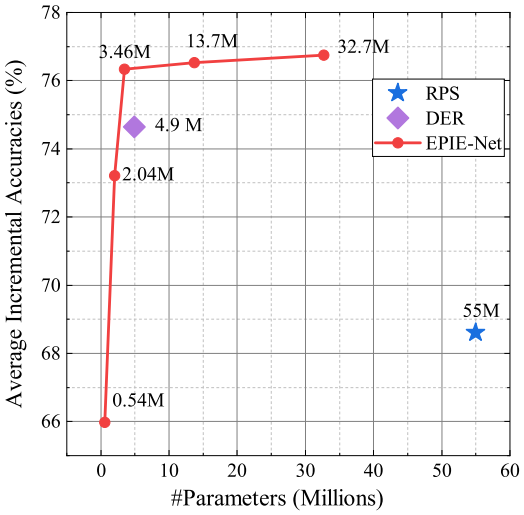

Figure 6: Ablation analysis on CIFAR-100 with 10 classes increments for different model size.

ters to achieve 75.36% accuracy, but EPIE-Net only needs 34M parameters to reach 76.77% high performance.

The effect of each component. EPIE-Net contains three important components:(1) Label-mixed Uncertainty Learning, (2) Perturbation Inference, and (3) Class-Balanced Finetuning. Table 4 summarizes the ablation results of EPIE-Net on CIFAR100 10 steps. Note that the without Class-Balance Finetuning setting means EPIE-Net directly merges the trained task-decoder module for prediction, so no reserved exemplars data is used in this setting. We can see that all three greatly improve the baseline method (55.44% → 76.33%). Analyzing each component individually, Class-Balanced Finetuning based on reserved exemplars is crucial, which improves performance by an average of 12.22% (59.83% → 72.05%). Moreover, Perturbation Inference can be used as a plug-and-play module, improving the base model by 4.81% (57.94% → 62.05%).

Exemplars overfitting for long-term continual learning. Reserved samples can effectively alleviate catastrophic forgetting, but in the case of long-term continual learning, repeated training of reserved data will also cause overfitting. We com-

Label-mixed Uncertainty Learning	Perturbation Inference	Class-Balanced Finetuning	Avg
×	×	×	55.44
✓	×	×	56.71
×	✓	×	59.23
✓	✓	×	63.55
×	×	✓	61.67
✓	×	✓	74.36
×	✓	✓	65.48
✓	✓	✓	76.33

Table 4: Ablations of the different key components of our EPIE-Net architecture. We report the average incremental accuracies (%) on CIFAR100 with 10 classes per-task.

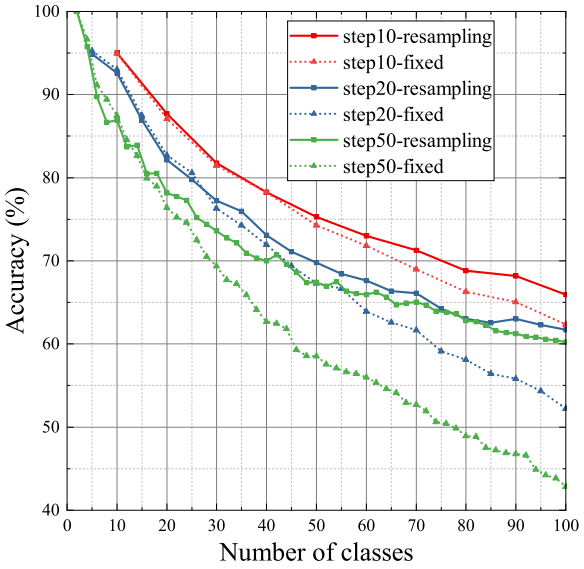

Figure 7: Exemplars overfitting for long-term continual learning on CIFAR100 dataset. We compare the effect of fixed reserved exemplars and resampling rehearsal exemplars under different increment steps.

m	0	1	2	3	4	10
Avg	74.36	74.63	76.33	76.55	76.79	77.11

Table 5: Average incremental accuracy on CIFAR100 with 10 steps for different perturbation numbers m.

pare the performance of fixed reserved exemplars and resampling rehearsal exemplars under different increment steps. As shown in Figure 7, the solid line represents the resampling rehearsal, and the dashed line represents the fixed reserved exemplars. We can see that the performance of the two strategies is similar in the early steps. However, the performance of the fixed reserved method drops significantly as the training progresses.

The effect of perturbation number. As shown in Table 5, we conduct a sensitive study of different perturbation inference numbers m on CIFAR100 10 tasks split. Note that $m = 0$ means standard inference, $m = 1$ means using data augmentation techniques in the test data, and $m \geqslant 2$ means calculating the mean probability of perturbed samples. We can see that the higher the value of m, the higher the performance, but when m is greater than 2, its performance improvement is slight. Considering the impact of computing resources, in the default case is recommended to use $m = 2$.

The effect of learning rate on finetuning stage. As shown in Table 6, we conduct a sensitive study of different learning rate on CIFAR100 10 tasks split. We can see that, in the class-balanced training stage, the initial learning rates of 0.001 and 0.0005 differ by 2.98% and 7.62% in average and final accuracy, respectively. We, therefore, use an initial learning rate that decays by task and finetune the model with a lower learning rate as tasks increase.

fixed lr 0.001		fixed lr 0.0005		decay lr 0.0005×0.9^i	
avg	last	avg	last	avg	last
71.64	56.43	74.62	64.05	76.33	65.93

Table 6: The effect of learning rate on class-balanced finetuning training stage. We report the average incremental accuracies (%) and Last accuracies (%) on CIFAR100 with 10 classes per-task.

5. Conclusions and future work

In this work, we aim to design a more efficiently expandable architecture for class-incremental learning. The proposed Efficient Perturbation Inference and Expandable Network (EPIE-Net) has two key novelties: (1) the proposed lightweight multi-scale hierarchical encoder/decoder architecture can achieve efficient growth for continual learning. (2) mixed-label uncertainty learning strategies can effectively improve the robustness of the model in continual learning scenarios. The experimental results show that our method consistently outperforms other methods with fewer parameter overhead.

In future work, we will improve our framework from two aspects. First, the encoder of EPIE-Net would suffer from the feature shift phenomenon under long-term continual learning, so our framework has to rely on a class-balanced fine-tuning strategy to mitigate this phenomenon. However, continual fine-tuning will also cause overfitting of rehearsal data. Next, we will introduce contrastive learning to improve the generalization of the encoder. Second, our framework uses a lightweight decoder to expand new task branches, and we will delve into the relationships among task branches to improve their reusability.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to acknowledge the financial support provided by the Postgraduate Research and Innovation Foundation of Yunnan University with No.2021Z113, Yunnan provincial major science and technology special plan projects: digitization research and application demonstration of Yunnan characteristic industry under Grant: No. 202002AD080001, The Natural Science Foundation of China (NSFC) under Grant: No. 61876166, and Yunnan Basic Research Program for Distinguished Young Youths Project, under Grant: 202101AV070003.

References

[1] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of learning and motivation, volume 24, Elsevier, 1989, pp. 109–165.
[2] R. Caruana, Multitask learning, Machine learning 28 (1997) 41–75.

[3] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, icarl: Incremental classifier and representation learning, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[4] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proceedings of the national academy of sciences 114 (2017) 3521–3526.

[5] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 374–382.

[6] Z. Li, D. Hoiem, Learning without forgetting, IEEE transactions on pattern analysis and machine intelligence 40 (2017) 2935–2947.

[7] R. Aljundi, P. Chakravarty, T. Tuytelaars, Expert gate: Lifelong learning with a network of experts, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3366–3375.

[8] S. Yan, J. Xie, X. He, Der: Dynamically expandable representation for class incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3014–3023.

[9] H. Shin, J. K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, Advances in neural information processing systems 30 (2017).

[10] D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, Advances in neural information processing systems 30 (2017).

[11] J. Rajasegaran, M. Hayat, S. Khan, F. S. Khan, L. Shao, Random path selection for incremental learning, Advances in Neural Information Processing Systems (2019).

[12] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, arXiv preprint arXiv:1606.04671 (2016).

[13] S. Hou, X. Pan, C. C. Loy, Z. Wang, D. Lin, Learning a unified classifier incrementally via rebalancing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 831–839.

[14] Z. Li, C. Zhong, S. Liu, R. Wang, W.-S. Zheng, Preserving earlier knowledge in continual learning with the help of all previous feature extractors, arXiv preprint arXiv:2104.13614 (2021).

[15] S. Golkar, M. Kagan, K. Cho, Continual learning via neural pruning, arXiv preprint arXiv:1903.04476 (2019).

[16] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, C.-S. Chen, Compacting, picking and growing for unforgetting continual learning, Advances in Neural Information Processing Systems 32 (2019).

[17] A. Mallya, S. Lazebnik, Packnet: Adding multiple tasks to a single network by iterative pruning, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2018, pp. 7765–7773.

[18] Y. Yang, Y. Hu, X. Zhang, S. Wang, Two-stage selective ensemble of cnn via deep tree training for medical image classification, IEEE Transactions on Cybernetics (2021).

[19] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744.

[20] G. Hinton, O. Vinyals, J. Dean, et al., Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 2 (2015).

[21] A. Douillard, A. Ramé, G. Couairon, M. Cord, Dytox: Transformers for continual learning with dynamic token expansion, arXiv preprint arXiv:2111.11326 (2021).

[22] Y. Yang, J. Jiang, Adaptive bi-weighting toward automatic initialization and model selection for hmm-based hybrid meta-clustering ensembles, IEEE transactions on cybernetics 49 (2018) 1657–1668.

[23] Y. Yang, J. Jiang, Hybrid sampling-based clustering ensemble with global and local constitutions, IEEE transactions on neural networks and learning systems 27 (2015) 952–965.

[24] J. Kim, J. Kim, N. Kwak, Stacknet: Stacking feature maps for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 242–243.

[25] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra, Pathnet: Evolution channels gradient descent in super neural networks, arXiv preprint arXiv:1701.08734 (2017).

[26] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, arXiv preprint arXiv:1710.09412 (2017).

[27] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, Y. Bengio, Manifold mixup: Better representations by interpolating hidden states, in: International Conference on Machine Learning, PMLR, 2019, pp. 6438–6447.

[28] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, Y. Yoo, Cutmix: Regularization strategy to train strong classifiers with localizable features, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6023–6032.

[29] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, D. Larlus, Hard negative mixing for contrastive learning, Advances in Neural Information Processing Systems 33 (2020) 21798–21809.

[30] S. Kim, G. Lee, S. Bae, S.-Y. Yun, Mixco: Mix-up contrastive learning for visual representation, arXiv preprint arXiv:2010.06300 (2020).

[31] Z. Shen, Z. Liu, Z. Liu, M. Savvides, T. Darrell, E. Xing, Un-mix: Rethinking image mixtures for unsupervised visual representation learning, arXiv preprint arXiv:2003.05438 (2020).

[32] V. Verma, T. Luong, K. Kawaguchi, H. Pham, Q. Le, Towards domain-agnostic contrastive learning, in: International Conference on Machine Learning, PMLR, 2021, pp. 10530–10541.

[33] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, J. Choi, Rainbow memory: Continual learning with a memory of diverse samples, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8218–8227.

[34] A. Prabhu, P. H. Torr, P. K. Dokania, Gdumb: A simple approach that questions our progress in continual learning, in: European conference on computer vision, Springer, 2020, pp. 524–540.

[35] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448–456.

[36] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), arXiv preprint arXiv:1606.08415 (2016).

[37] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, PMLR, 2016, pp. 1050–1059.

[38] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, K. Alahari, End-to-end incremental learning, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 233–248.

[39] B. Zhao, X. Xiao, G. Gan, B. Zhang, S.-T. Xia, Maintaining discrimination and fairness in class incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13208–13217.

[40] A. Douillard, M. Cord, C. Ollion, T. Robert, E. Valle, Podnet: Pooled outputs distillation for small-tasks incremental learning, in: European Conference on Computer Vision, Springer, 2020, pp. 86–102.

[41] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International journal of computer vision 115 (2015) 211–252.

[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[44] I. Loshchilov, F. Hutter, Sgdr: Stochastic gradient descent with warm restarts, arXiv preprint arXiv:1608.03983 (2016).

[45] A. Chaudhry, P. K. Dokania, T. Ajanthan, P. H. Torr, Riemannian walk for incremental learning: Understanding forgetting and intransigence, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 532–547.