Learning and Transforming General Representations to Break Down Stability-Plasticity Dilemma

Kengo Murata[0000−0002−6835−7810], Seiya Ito[0000−0002−7079−9116], and Kouzou Ohara[0000−0002−7399−2472]

Aoyama Gakuin University, Kanagawa, Japan
d5621007@aoyama.jp
{s.ito,ohara}@it.aoyama.ac.jp

Abstract. In the Class Incremental Learning (CIL) setup, a learning model must have the ability to incrementally update its knowledge to recognize newly appeared classes (plasticity) while maintaining the knowledge to recognize the classes it has already learned (stability). Such conflicting requirements are known as the stability-plasticity dilemma, and most existing studies attempt to achieve a good balance between them by stability improvements. Unlike those attempts, we focus on the generality of representations. The basic idea is that a model does not need to change if it has already learned such general representations that they contain enough information to recognize new classes. However, the general representations are not optimal for recognizing the classes a model has already learned because the representations must contain unrelated and noisy information for recognizing them. To acquire representations suitable for recognizing known classes while leveraging general representations, in this paper, we propose a new CIL framework that learns general representations and transforms them into suitable ones for the target classification tasks. In our framework, we achieve the acquisition of general representations and their transformation by self-supervised learning and attention techniques, respectively. In addition, we introduce a novel knowledge distillation loss to make the transformation mechanism stable. Using benchmark datasets, we empirically confirm that our framework can improve the average incremental accuracy of four types of CIL methods that employ knowledge distillation in the CIL setting.

1 Introduction

Deep neural network models can provide superior performance in image recognition tasks [49, 24, 17] if all the classes to be recognized are available when training them. However, this prerequisite is unrealistic in most practical situations. To overcome this limitation, many studies have tackled the problem of Class Incremental Learning (CIL) [38, 6], in which a model learns new classes incrementally through multiple classification tasks that are sequentially provided. In the CIL setup, a learning model needs to update its knowledge to recognize newly emerged classes (plasticity), while maintaining the knowledge to recognize the classes it has already learned (stability). Such conflicting requirements are known as the stability-plasticity dilemma [40], which is the major issue in CIL.

In general, deep neural network models tend to learn task-oriented knowledge and overwrite the knowledge learned for the previous tasks with the one for the new task in CIL. This problem is known as catastrophic forgetting [39, 32], which implies that the balance of the two conflicting requirements is biased toward plasticity. Thus, most existing studies attempt to achieve a good balance between stability and plasticity by improving stability. For instance, regularization methods [9] prevent important parameters of a neural network model from moving. Knowledge distillation [25, 35] attempts to preserve features a model has already acquired. However, all those approaches achieve improvement in stability in exchange for the plasticity of a learning model, which implies their performance degrades for the new tasks when the model needs to be highly plastic to adapt to the new tasks.

The above observation indicates that the CIL methods with knowledge distillation cannot achieve high performance both for the new and learned tasks simultaneously without a mechanism to decrease the required plasticity for adapting to the new tasks. Therefore, in this paper, we focus on the generality of learned representation because the model does not need to change if the representations it has already learned are so general that they contain enough information to recognize new classes. The existing approaches cannot learn such general representations because they optimize model parameters with the cross-entropy loss to solve the classification task. The cross-entropy loss encourages learning task-oriented representations, but it does not ensure the model acquires representations not directly related to the current and previous tasks, even if they may be helpful for the subsequent tasks. Thus, we propose a new CIL framework and adopt the self-supervised learning technique [14] to learn general representations that could be useful for classes involved in future tasks. Here, it is noted that the general representations are not optimal for recognizing the classes a model has already learned because they must contain unrelated and noisy information for the known classes to be recognized. Thus, we further introduce into our framework a mechanism based on attention techniques [44, 27] to transform the general representations into suitable ones for the target classification tasks. Furthermore, we introduce a novel knowledge distillation loss to improve the stability of our transformation mechanism.

Our contributions are summarized as follows:

1. We propose a novel CIL framework that makes a model learn general representations and transforms them into suitable ones for the target classification tasks.
2. We design a novel knowledge distillation loss that improves the stability of our transformation mechanism.
3. Through the experiments on benchmark datasets in the CIL setting, we empirically confirm that our framework improves the average incremental accuracy of existing CIL models employing knowledge distillation.

2 Related Work

2.1 Class Incremental Learning

A straightforward solution to the problem of CIL is to learn classes involved in the current task with all the classes the model has already learned in the past tasks. However, this solution is unrealistic because of the considerable training time and the huge memory space. Experience replay [11] achieves this ideal solution within a reasonable training time by memorizing only a few exemplars used in the previous tasks and replaying them while learning the current task. Thanks to its simplicity and good performance in the CIL setting, experience replay has been one of the standard methods in CIL to date.

Even if using experience replay, the model is still unstable through the process of a CIL scenario and does not perform well for classes in the previous tasks because the number of memorized exemplars is very limited. For stability improvements, many CIL methods [47, 26, 18, 50, 48, 34] employ knowledge distillation, which preserves the input-output relations the model has already learned. For instance, while learning the t-th task, the less-forget constraint proposed in [26] forces a model to minimize the distance between its output feature vector and the one of the latest previous model, which has just finished learning the $(t-1)$-th task.

The other issue involved in experience replay is the class-imbalance problem between the previous and current tasks. This class-imbalance problem causes predictions biased toward the classes in the current task [54]. To tackle this problem, several studies propose the bias correction methods such as the bias correction layer [54], the normalization of weight vectors in the classification layer [26, 5, 4, 58], and the imbalance-aware loss functions [1, 30, 41].

2.2 Representation Learning in CIL

A few studies on CIL focus on representation learning of deep neural network models. We categorize them into three groups: contrastive learning methods [37, 42, 8], meta-learning methods [46, 28, 3], and self-supervised learning methods [20, 57, 59, 29]. Firstly, contrastive learning methods optimize a model with respect to supervised contrastive loss [31], which encourages the model to acquire intra-class concentrated and inter-class separated features. Secondly, meta-learning methods learn features that accelerate future learning via meta-learning frameworks [19]. However, due to the nature of the meta-learning framework, each task must be explicitly identified when using the model for prediction. This necessity for task identification leads to the requirements of the additional task prediction system [46], which should be continually updatable without forgetting. Thirdly, self-supervised learning methods utilize the self-supervised learning approaches [36] as with our framework. For instance, the existing studies [59, 57] use the angle of training data as self-supervision. They optimize one feature space with respect to original and self-supervised labels. In contrast, our framework optimizes two kinds of feature space with respect to each label. To our best knowledge, DualNet [45] is the most similar method to our framework. In DualNet, two completely different networks are optimized with each label, interacting with each other. In contrast to DualNet, our framework works with a single neural network, so the number of parameters is lower than the one used in DualNet. We note that DualNet is specialized in online continual learning [2], which is similar but different to the CIL scenario targeting in our work. Therefore, we do not compare our framework to DualNet in this paper.

2.3 Self-supervised Learning

Recently, self-supervised learning has been shown effective in learning general representations without label supervision [12, 23, 14]. The earlier studies propose the proxy task, e.g., prediction rotations [21], patch permutation [43], image colorization [56], and clustering [7], which enables a model to learn the general representations. More recently, contrastive self-supervised learning has shown great success [36]. It encourages a model to learn the features that can identify one view of an image with the other view and distinguish the image from any other images. For instance, SimCLR [12] produces two types of views by sophisticated data augmentations, and the variants of MoCo [23, 13, 15] introduce a momentum encoder to produce the different views. The key issue of contrastive self-supervised learning methods is the necessity of the vast amount of negative samples. BYOL [22], SimSiam [14], and Barlow Twins [55] are representative methods that only rely on two distinct views of the same image. In our methods, we use SimSiam as a self-supervised learning method because of its simplicity and the unnecessity of negative samples.

3 Problem Setup and Preliminaries

In this section, we firstly introduce the problem setup of CIL and then briefly describe CIL methods with knowledge distillation, the baseline methods for our framework. Let $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_N\}$ be a sequence of N tasks. For the t-th task $\mathcal{T}_t$, there is a training dataset $\mathcal{D}_t$ and a memorized exemplar set $\mathcal{M}_t$. More specifically, $\mathcal{D}_t = \{(x_t^i, y_t^i)\}_{i=1}^{n_t}$ is composed of pairs of an image $x_t^i \in \mathcal{X}_t$ and its corresponding label $y_t^i \in \mathcal{Y}_t$. The memorized exemplar set $\mathcal{M}_t = \{(x_{\text{mem}}^i, y_{\text{mem}}^i)\}_{i=1}^{M}$ is composed of M previously learned instances $(x_{\text{mem}}^i, y_{\text{mem}}^i) \in \bigcup_{j=1}^{t-1} \mathcal{D}_j$. Note that the label sets of different tasks do not overlap, meaning $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ if $i \neq j$. We consider a deep neural network model with two components: a feature extractor f and a classifier g. While learning the t-th task, the model updates its parameters with both the training dataset $\mathcal{D}_t$ and the exemplar set $\mathcal{M}_t$ to achieve correct prediction for any instance $x \in \bigcup_{j=1}^{t} \mathcal{X}_j$. We refer to the model that is being updated for the current task $\mathcal{T}_t$ as the current model and call its components the current feature extractor f_t and the current classifier g_t. Similarly, the model that has already been updated for the task $\mathcal{T}_{t-1}$ is referred to as the previous model, whose components are the previous feature extractor f_{t-1} and classifier g_{t-1}.

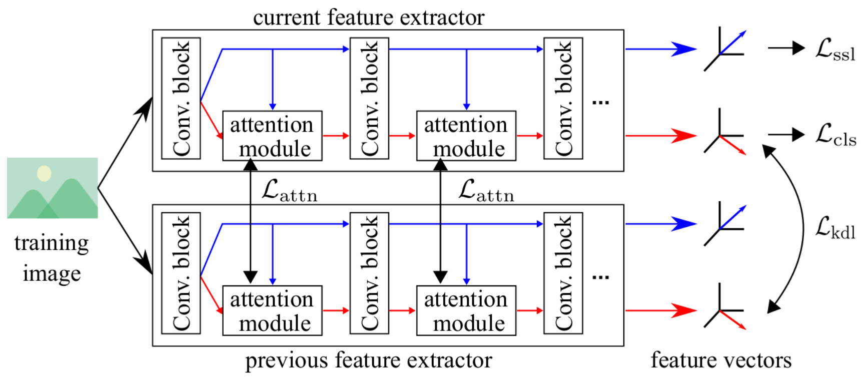
current feature extractor
Conv. block
attention module
Conv. block
attention module
Conv. block
...
$\mathcal{L}_{\text{ssl}}$
$\mathcal{L}_{\text{cls}}$
$\mathcal{L}_{\text{attn}}$
$\mathcal{L}_{\text{attn}}$
$\mathcal{L}_{\text{kdl}}$
training image
previous feature extractor
feature vectors

Fig. 1. Illustration of our framework. The feature extractor consists of the attention modules and convolutional blocks, e.g., Residual blocks. It outputs two feature vectors: one for the calculation of the self-supervised loss $\mathcal{L}_{\text{ssl}}$ and the other for the calculation of the loss term $\mathcal{L}_{\text{cls}}$ for classification in addition to knowledge distillation loss $\mathcal{L}_{\text{kdl}}$. The attention-wise knowledge distillation loss $\mathcal{L}_{\text{attn}}$ keeps each attention module stable. For simplicity, we omit the other network components, e.g., the classifier.

The CIL methods with knowledge distillation optimize parameters with the loss function defined as:

$$\mathcal{L}_{\text{kd}} = \beta_t^{\text{cls}}\mathcal{L}_{\text{cls}} + \beta_t^{\text{kdl}}\mathcal{L}_{\text{kdl}} + \mathcal{L}_{\text{other}} \ , \tag{1}$$

where $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{kdl}}$, and $\mathcal{L}_{\text{other}}$ denote the loss terms for classification, knowledge distillation, and other purposes, respectively. In addition, β_t^{cls} and β_t^{kld} are the scalar values that control the impact of each loss term depending on the number of classes the model learned. The implementation of loss terms depends on individual CIL methods. For instance, UCIR [26] uses cross-entropy loss as the loss term $\mathcal{L}_{\text{cls}}$ and the cosine distance between output vectors of the previous and current feature extractors as the loss term $\mathcal{L}_{\text{kdl}}$. However, the fundamental purpose of the loss terms is the same; $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{kdl}}$ evaluate the classification performance of the current model and the dissimilarity between the current and previous models, respectively.

4 Methodology

Fig. 1 shows an overview of our framework combined with CIL methods employing knowledge distillation. In our framework, the feature extractor outputs two kinds of feature vectors from an image. One vector derived through convolutional blocks (blue line in Fig. 1) is used for the calculation of the self-supervised loss $\mathcal{L}_{\text{ssl}}$. Through the optimization of the self-supervised loss, the parameters of convolutional blocks are updated to extract less task-oriented but more general features. The other vector (red line in Fig. 1) affected by attention modules is used for the calculation of the loss term for classification $\mathcal{L}_{\text{cls}}$. The optimization

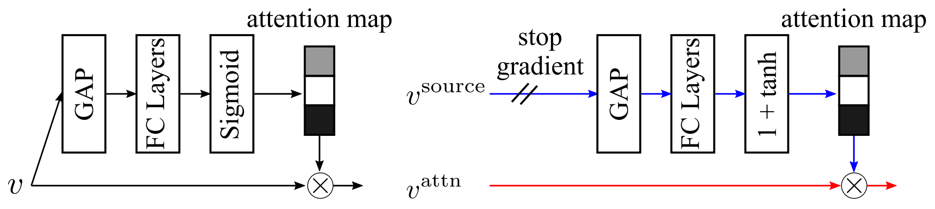

Fig. 2. The schemas of the original SE block (left) and our modified SE block (right). In our block, an attention map is generated from the intermediate feature vector v^{source} derived from the convolutional blocks. The attention map acts on the other intermediate feature vector v^{attn} derived from the convolutional blocks and attention modules.

of the loss term $\mathcal{L}_{\text{cls}}$ enables the attention modules to transform general representations extracted by convolutional blocks into suitable ones for classification. Moreover, a new knowledge distillation loss $\mathcal{L}_{\text{attn}}$ improves the stability of our transformation mechanism. To sum up, our framework consists of three components: representation learning with self-supervision, the transformation mechanism with attention modules, and knowledge distillation for the transformation mechanism.

4.1 Representation Learning with Self-supervision

To encourage the model to acquire the general representation, we introduce the loss function used on SimSiam [14]. More specifically, for a training image x, we first generate two views of the image denoted as $\dot{x}$ and $\ddot{x}$ through random data augmentation. Then, we produce the two types of vectors for each view through the feature extractor f_t, the projector $proj_t$, and the predictor $pred_t$. More precisely, these vectors are denoted as $\dot{p} = (pred_t \circ proj_t \circ f_t)(\dot{x})$ and $\dot{q} = (proj_t \circ f_t)(\dot{x})$, respectively. Finally, we calculate $\mathcal{L}_{\text{ssl}}$ as below:

$$\mathcal{L}_{\text{ssl}} = \frac{1}{2}(\mathcal{D}_{\cos}(\dot{p}, \text{stopgrad}(\ddot{q})) + \mathcal{D}_{\cos}(\ddot{p}, \text{stopgrad}(\dot{q}))) \ , \tag{2}$$

where stopgrad denotes the stop-gradient operation, and $\mathcal{D}_{\cos}(p, q)$ denotes the cosine distance between two vectors p and q. With the optimization of self-supervised loss $\mathcal{L}_{\text{ssl}}$, the parameters of the convolutional layers are updated so as to extract meaningful features from an image belonging to not only the learned classes but also the new upcoming classes.

4.2 Attention Module

Thanks to the optimization of self-supervised loss, the representations extracted from the convolutional layers are general. However, they definitely contain unrelated information to recognize the classes involved in the current and previous tasks. Thus, we introduce a transformation mechanism that weakens such unrelated information and strengthens the useful one for classification. We realize such transformation by introducing the attention module similar to the SE block [27]. As shown in Fig. 2, we change the original SE block with respect to the generation process of an attention map and the activation function. This modification is intended to mitigate the potential stability decrease caused by the attention modules, enable strengthening of information, and avoid unnecessary transformations.

The original SE block generates an attention map from an intermediate feature vector, so the attention map changes with the fluctuation of the intermediate vector even if the parameters of the SE block do not change while learning new tasks. In addition, the fluctuation of the intermediate vector further enlarges due to the multiplication with fluctuated attention map. Such property indicates that the fluctuation of parameters in a convolutional layer or an SE block could significantly fluctuate the final output vector even if the parameters of their succeeding layers do not change. We experimentally confirmed that SE-ResNet [27] was less stable than ResNet [24], which does not have any attention mechanism (as for the detailed experimental results, please see the supplementary materials). To mitigate the potential stability decrease, we use the intermediate feature vector derived through only convolutional blocks for the generation of the attention map. With this modification, the procedure to generate attention maps in our SE block is not affected by its proceeding attention modules. Additionally, we introduce a stop-gradient operation because this gradient could prevent the optimization with self-supervision.

We alter the activation function from sigmoid function to $1 + \tanh(\cdot)$. This altered function has two desirable properties which the sigmoid function does not have. First, the range of $1 + \tanh(\cdot)$ is from 0 to 2, so the attention module with our activation function can strengthen or weaken the values of intermediate feature vectors, while the one with a sigmoid function can only weaken the values. Second, the attention module with the activation function of $1 + \tanh(\cdot)$ becomes an identity mapping if all the values of its parameters are 0. Owing to this property, our attention modules can avoid generating unnecessary attention maps which lead to destructive transformations of general representations.

4.3 Knowledge Distillation for Attention Mechanism

Formally, the l-th attention module generates the attention map $a_t^l = 1 + \tanh(h_t^l(z_t^l))$ by applying the calculation on the FC layers h_t^l contained in the attention module to the pooled intermediate feature vector z_t^l. In addition, let $a_{t-1}^l = 1 + \tanh(h_{t-1}^l(z_{t-1}^l))$ be the attention map generated by the l-th attention module in the previous feature extractor, where h_{t-1}^l is its FC layers and z_{t-1}^l is the pooled intermediate feature vector derived from its preceding layers. To stabilize the process of generating attention maps, the input vector to each attention module should less fluctuate in addition to preserving the input-output relations defined by each attention module. More specifically, the pooled intermediate feature vector z_t^l should be close to the vector z_{t-1}^l derived through the previous feature extractor, and the l-th attention module should preserve its input-output relations. To achieve such preservation, we design the attention-wise knowledge distillation loss composed of two types of loss terms; one preserves the input-output relations, and the other prevents the fluctuation of the intermediate feature vectors.

We implement the loss term for the retention of the input-output relations as the weighted sum of the distances between the output attention maps generated by the current and previous attention modules from pooled intermediate feature vectors z_t^l and z_{t-1}^l. For the l-th attention module, we define the loss term as:

$$\mathcal{L}_{\text{attn}}^{\text{map},l} = \lambda_{\text{attn}}^{\text{map,new}} \mathcal{L}_{\text{attn}}^{\text{map,new},l} + \lambda_{\text{attn}}^{\text{map,old}} \mathcal{L}_{\text{attn}}^{\text{map,old},l} \ , \tag{3}$$

$$\mathcal{L}_{\text{attn}}^{\text{map,new},l} = \| \tanh(h_t^l(z_t^l)) - \tanh(h_{t-1}^l(z_t^l)) \|_2 \ , \tag{4}$$

$$\mathcal{L}_{\text{attn}}^{\text{map,old},l} = \| \tanh(h_t^l(z_{t-1}^l)) - \tanh(h_{t-1}^l(z_{t-1}^l)) \|_2 \ , \tag{5}$$

where $\lambda_{\text{attn}}^{\text{map,new}}$ and $\lambda_{\text{attn}}^{\text{map,old}}$ are the hyperparameters controlling the impact of each distance regularization.

To prevent the fluctuation of the intermediate feature vectors, we minimize the weighted distance between the intermediate feature vectors derived from the current and previous feature extractor. Formally, for the l-th attention module, we define the loss term as:

$$\mathcal{L}_{\text{attn}}^{\text{source},l} = \lambda_{\text{attn}}^{\text{source}} \hat{\omega}_l \left\| \overline{z_t^l} - \overline{z_{t-1}^l} \right\|_2 \ , \tag{6}$$

$$\hat{\omega}_l = \frac{\omega_l}{\sum_{j=1}^{L} \omega_j} \ , \tag{7}$$

where $\overline{z} = z / \|z\|_2$ denotes the l_2-normalized vector, L denotes the number of attention modules, $\lambda_{\text{attn}}^{\text{source}}$ is the hyperparameter controlling the impact of this loss term, and ω_l denotes the weighting factor. We consider the sensitivity to change in the values of the attention map with respect to change in the source vector as the weighting factor, which is calculated and updated when each task $\mathcal{T}_{t'}$ has been learned as:

$$\omega_l \leftarrow \frac{1}{2} \left(\omega_l + \mathbb{E}_{\mathcal{D}_{t'}} \left[\frac{1}{D^2} \sum_{d=1}^{D} \sum_{d'=1}^{D} \frac{\partial \tanh(h_{t'}^l(z_{t'}^l))_d^2}{\partial (z_{t'}^l)_{d'}} \right] \right) \ , \tag{8}$$

where D denotes the dimension of the source vector. With the introduction of this weighting factor, the fluctuation of the intermediate feature vector is highly prevented when it causes a large fluctuation of the corresponding attention map.

To sum up, we define the attention-wise knowledge distillation loss $\mathcal{L}_{attn}$ as:

$$\mathcal{L}_{\text{attn}} = \sum_{l=1}^{L} \mathcal{L}_{\text{attn}}^{\text{map},l} + \mathcal{L}_{\text{attn}}^{\text{source},l} \ . \tag{9}$$

4.4 Overall Loss

When the proposed framework is introduced to CIL methods with knowledge distillation, the loss function is defined as:

$$\mathcal{L} = \beta_t^{\text{cls}} (\alpha \mathcal{L}_{\text{cls}} + (1 - \alpha) \mathcal{L}_{\text{ssl}}) + \beta_t^{\text{kdl}} (\mathcal{L}_{\text{kdl}} + \mathcal{L}_{\text{attn}}) + \mathcal{L}_{\text{other}} \ , \tag{10}$$

where $\alpha \in [0, 1]$ is the hyperparameter that controls the balance between $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{ssl}}$. We note that the self-supervised loss $\mathcal{L}_{\text{ssl}}$ is calculated from two views of an image generated by complex data augmentation techniques, while the others are calculated from another view generated by the other simple data augmentation techniques. The reason why we utilize these two types of data augmentation is that complex data augmentation is unsuitable for classification learning. We describe the entire learning process and learning tips in supplementary materials.

5 Experiments

5.1 Experimental Settings

Datasets and task configurations. We employed two image datasets for our experiments: CIFAR100 [33] and ImageNet100 [16, 26]. CIFAR100 contains 60,000 images of size 32×32 from 100 classes, and each class includes 500 training samples and 100 test samples. ImageNet100 is a subset of the original ImageNet1000 [16] with 100 classes randomly selected from the original one. According to the common CIL setting [47], the arrangement of all classes was shuffled with Numpy random seed 1993. We used the first 50 classes for the first task and the rest for subsequent tasks by dividing them into K sets. We constructed two kinds of CIL scenarios for each dataset by setting K to 5 and 10. Each CIL scenario is referred to as a K-phase setup.

Baselines. We used the following four knowledge distillation-based methods as baselines of our framework:

- IL-Baseline [30]: IL-Baseline optimizes the parameters of a model through the cross-entropy loss and the original knowledge distillation loss [25], which restricts the output probability of the model from moving.
- UCIR [26]: UCIR introduces cosine normalization in a classification layer to deal with the class-imbalance problem. Its implementation of knowledge distillation loss is the cosine similarity between the output vectors of the previous and current feature extractors.
- PODNet [18]: PODNet constrains the intermediate feature vectors of a feature extractor in addition to its output vector. In addition, it employs the local similarity classifier, which is the extended version of the classification layer used in UCIR.
- BSCE [30]: BSCE has the same learning procedure as IL-Baseline except for introducing the balanced softmax cross-entropy loss instead of the standard cross-entropy loss.

The detailed learning procedure of each baseline method is described in supplementary materials. We evaluated all of the methods through the average incremental accuracy (AIA) [47], defined as the average taken over accuracies of the model for test data after training for each task.

Table 1. Average incremental accuracy of four baselines *w/* and *w/o* our framework. Each result is in the form of the average ± standard deviation obtained from three independent trials using different random seeds. On ImageNet100, the scores of the baselines are reported from their respective papers.

	CIFAR100		ImageNet100	
method	5-phase	10-phase	5-phase	10-phase
IL-Baseline [30]	52.92 ±0.15	43.14 ±0.31	51.52	42.22
w/ Ours	56.49 ±0.20	48.13 ±0.51	61.57 ±0.05	59.13 ±0.09
UCIR [26]	69.29 ±0.15	63.16 ±0.12	70.47	68.09
w/ Ours	71.03 ±0.28	66.24 ±0.42	76.80 ±0.08	76.06 ±0.19
PODNet [18]	68.99 ±0.36	66.64 ±0.15	75.54	74.33
w/ Ours	70.61 ±0.12	69.02 ±0.05	**78.05** ±0.19	**76.66** ±0.10
BSCE [30]	71.64 ±0.16	64.87 ±0.39	72.57	68.25
w/ Ours	**74.09** ±0.24	**70.30** ±0.38	74.96 ±0.39	72.32 ±0.23

Implementation details. We used ResNet18 [24] as the CNN backbone. Our attention modules were attached to residual blocks in a similar manner to the SE blocks [27]. In addition, we used the same architectures of the projector and predictor as the ones of SimSiam [14]. All parameters were optimized through stochastic gradient descent (SGD) with a momentum of 0.9. For the first task of CIFAR100, the network was trained with 400 epochs using a cosine learning rate decay for the base learning rate of 0.2 and weight decay of 5e-4. Then, it was further optimized using the remaining tasks one by one with 200 epochs per task. For ImageNet100, we set the number of epochs and the base learning rate to 150 and 0.2 for the first task. For the other tasks, the number of epochs was set to 90. We adopted 1e-4 as the weight decay for all the tasks of ImageNet100. For all the experiments, the batch size was set to 128, and the number of exemplars was set to 20 per class. The exemplars were sampled based on the herd selection method [52, 47]. We employed the same data augmentation technique used in SimSiam for self-supervised learning, while using image flipping, random cropping, and color jittering for calculating the loss functions other than the self-supervised loss. Based on the hyperparameter tuning process in the existing work [10], we tuned the other hyperparameters through the three-task CIL setting constructed from the training data belonging to the first task. We describe the detailed hyperparameter tuning process and resulting hyperparameters in the supplementary materials.

5.2 Main Results

Tab. 1 shows the AIA of four baselines with and without our framework on CIFAR100 and ImageNet100. As shown in Tab. 1, our framework increases the AIA of all baselines on every dataset and setup. More precisely, the minimum increase of AIA is 1.62 points, and the maximum is 5.43 points on CIFAR100.

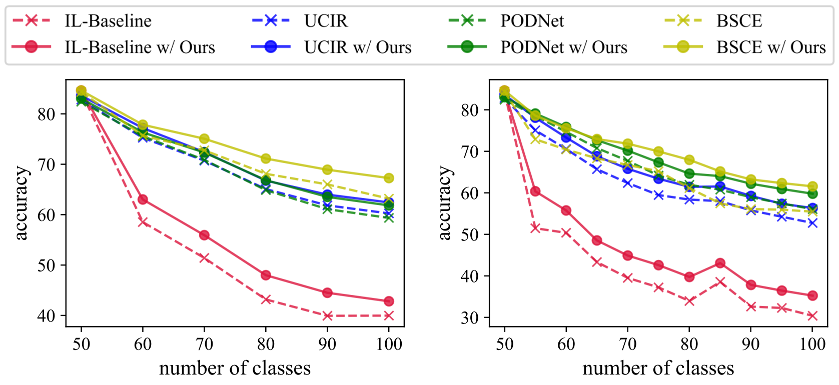

Fig. 3. The curve of accuracy over the classes on CIFAR100 (left: 5-phase, right: 10-phase). Each point indicates the average accuracy obtained from three independent trials using different random seeds.

Table 2. Three types of the average accuracies of BSCE with a part of our framework on CIFAR100 under the 5-phase setup. The first three columns indicate whether self-supervised learning, attention mechanism, and attention-wise knowledge distillation are introduced, respectively. Each result is in the form of the average ± standard deviation obtained from three independent trials using different random seeds.

SSL	AM	AwKD	AIA	Avg. Acc. (the first task)	Avg. Acc. (the new tasks)
			71.64 ±0.16	68.94 ±0.50	78.26 ±0.18
✓			71.76 ±0.53	68.68 ±1.02	76.78 ±0.21
	✓		70.34 ±0.14	66.30 ±0.43	**79.16** ±0.22
✓	✓		71.32 ±0.43	67.23 ±0.33	78.74 ±0.48
	✓	✓	72.54 ±0.51	73.11 ±0.45	72.48 ±1.17
✓	✓	✓	**74.09** ±0.24	**74.17** ±0.57	74.35 ±0.20

On ImageNet100, the increase of AIA is from 2.33 to 16.91 points. This common tendency of increase in AIA indicates the high flexibility of our framework. We also show the curve of accuracy over the classes on CIFAR100 in Fig. 3. As shown in Fig. 3, our framework consistently exceeds the accuracy of any baseline method. This result implies that the increase in AIA is not by chance.

5.3 Ablation Study

In this section, we present the results of several ablation experiments to clarify the effect of our framework. We report the AIA on CIFAR100 with the 5-phase setup in all the ablation experiments. Following the existing work [18], we use the average accuracy on the first task and the one on the new tasks as the additional evaluation metrics to analyze the effect of our framework in detail.

Table 3. Three types of the average accuracies on CIFAR100 with the 5-phase setup for different implementations of the transformation mechanism. "Original" means the implementation of the transformation mechanism is the original SE block, and "Ours" means the implementation is our modified SE block. The second column denotes whether attention-wise knowledge distillation is introduced. Note that all the methods optimize a model through the learning procedure of BSCE and self-supervision. Each result is in the form of the average ± standard deviation obtained from three independent trials using different random seeds.

AM	AwKD	AIA	Avg. Acc. (the first task)	Avg. Acc. (the new tasks)
Original		71.09 ±0.23	67.13 ±0.50	77.20 ±0.10
Ours		71.32 ±0.43	67.23 ±0.33	78.74 ±0.48
Original	✓	72.99 ±0.34	72.98 ±0.66	71.62 ±0.51
Ours	✓	**74.09** ±0.24	**74.17** ±0.57	74.35 ±0.20

Intuitively, the former accuracy gets a high value if the model is stable without the influence of forgetting, while the latter gets high if the model is plastic enough to learn the new tasks. We give the definition of each additional metric in supplementary materials. The baseline method is BSCE because it shows the highest AIA on CIFAR100 with the 5-phase setup regardless of whether our framework is introduced.

First, we verify the effect of the components of our framework, namely, self-supervised learning, attention mechanism, and attention-wise knowledge distillation. Tab. 2 shows the scores of the three evaluation metrics for the six types of methods that differ with the existence of each component. We summarize the comparing results as follows:

- The results on the first two rows show that self-supervised learning only increases AIA slightly (71.64 to 71.76) when the self-supervised loss is simply added as with the existing work [57].
- From the comparison between the results on the first and third rows, we can confirm that the attention mechanism boosts the accuracy for the new tasks while decreasing the accuracy for the first task and AIA. The results on the second and fourth rows indicate that the same tendency appears when introducing self-supervised learning.
- Even if self-supervised learning is not introduced (the fifth row), its AIA is higher than the baseline (the first row), however, the difference is only 0.9 points because of the low accuracy for the new tasks.
- Our framework (the sixth row) increases the AIA of the baseline by 2.45 points by improving the accuracy for the first task while mitigating the decrease in the accuracy for the new tasks.

The above findings suggest that the increase of AIA with our framework results from the synergistic effect of the components, but not from an independent one of each component.

Table 4. Three types of average accuracies on CIFAR100 with the 5-phase setup for different implementation of attention-wise knowledge distillation when the baseline method is BSCE. In the first column, "None", "*w/o* weight", and "*w/* weight" mean the loss term $\mathcal{L}_{\text{attn}}^{\text{source},l}$ is not introduced, introduced without weighting, and introduced with weighting, respectively. The second column denotes whether the loss term $\mathcal{L}_{\text{attn}}^{\text{map},l}$ is introduced. Each result is in the form of the average ± standard deviation obtained from three independent trials using different random seeds.

Source	Map	AIA	Avg. Acc. (the first task)	Avg. Acc. (the new tasks)
None		71.32 ±0.43	67.23 ±0.33	**78.74** ±0.48
w/o weight		73.57 ±0.35	73.99 ±0.35	72.52 ±0.35
w/ weight		73.68 ±0.36	74.04 ±0.47	72.61 ±0.47
None	✓	71.46 ±0.22	67.50 ±0.32	78.09 ±0.17
w/o weight	✓	73.84 ±0.24	**74.31** ±0.18	72.55 ±0.46
w/ weight	✓	**74.09** ±0.24	74.17 ±0.57	74.35 ±0.20

Next, we compare the original SE block [27] and our modified SE block to clarify the effect of our modification on the transformation mechanism. Tab. 3 shows the scores of the three evaluation metrics for the methods whose transformation mechanism is implemented by the original SE block or our modified SE block. Without attention-wise knowledge distillation (the first and second rows), AIA slightly increases (71.09 to 71.32) with our modification of the SE block. However, with attention-wise knowledge distillation (the last two rows), the increase of AIA by our modification becomes 1.10 points. In addition, the accuracy for the first task and the one for the new tasks both increase through the modification of the transformation mechanism. These results indicate that our modification only has a negligible impact on the performance of the model if the attention-wise knowledge distillation is not applied, however, the impact becomes relatively large when it is introduced.

Finally, we clarify the effect of the attention-wise knowledge distillation loss by the comparing results in Tab. 4. From the result on the first three rows, we can verify that the constraints on the input vector to the attention modules imposed by the loss term $\mathcal{L}_{\text{attn}}^{\text{source},l}$ increase AIA by improving the stability of a model, whether with or without weighting. The effect of weighting is very limited without the loss term $\mathcal{L}_{\text{attn}}^{\text{map},l}$. Similarly, the comparison between the results on the first and fourth rows shows that introducing the loss term $\mathcal{L}_{\text{attn}}^{\text{map},l}$ increases AIA by only 0.14 points when the loss term $\mathcal{L}_{\text{attn}}^{\text{source},l}$ is not employed. However, when the weighting and the loss term $\mathcal{L}_{\text{attn}}^{\text{map},l}$ are introduced (the sixth row), the accuracy for the new tasks is 2.17 points higher than the accuracy of the method employing only the loss term $\mathcal{L}_{\text{attn}}^{\text{source},l}$ without weighting (the second row). Thus, introducing the weighting and the loss term $\mathcal{L}_{\text{attn}}^{\text{map},l}$ is effective in improving the plasticity of a model. This is because they control the power of constraints for each attention module. In fact, the weighting procedure controls the power of the constraints on the input vector to each attention module. In

Table 5. Average incremental accuracy and the last accuracy (LA) of Co2L *w/* and *w/o* our framework on CIFAR10 [33] with 200 exemplars. The experimental configurations are all the same as the ones described in the original paper [8]. Each result is in the form of the average ± standard deviation obtained from three independent trials using different random seeds.

method	AIA	LA
Co2L	79.17 ±0.15	64.46 ±0.33
w/ Ours	**80.69** ±0.21	**66.76** ±0.79

addition, the constraints imposed by the loss term $\mathcal{L}_{\text{attn}}^{\text{map},l}$ affect the parameter of each attention module only when its input-output relation changes. Such selective constraint strongly regularizes a part of the parameters, which means the other parameters are relatively free to adapt to new tasks.

5.4 The Effect on Contrastive Learning Methods

To show the effect on the contrastive learning methods, we conducted the comparative experiment using Co2L [8] as the baseline method, which learned classification through the supervised contrastive loss. We followed the experimental settings reported in the original paper [8] on CIFAR10 [33] with 200 exemplars. Tab. 5 shows the results in AIA and the last accuracy (LA), the accuracy of a model having learned all the tasks. Note that we compare the methods in terms of LA because it is the evaluation metric in [8]. The experimental results show that our framework increases AIA by 1.52 points and LA by 2.3 points. This indicates that our framework can improve the performance of not only methods that learn classification with the cross-entropy loss but also those which learn it with supervised contrastive loss.

6 Conclusion

In this paper, we proposed a novel CIL framework that incorporates the representation learning with self-supervision, the transformation mechanism with attention modules, and the attention-wise knowledge distillation. We empirically confirmed our framework can improve the average incremental accuracy of four types of knowledge distillation-based methods using benchmark datasets. In addition, we showed that our framework can also improve the average incremental accuracy and the last accuracy of Co2L, a contrastive learning method. These results assure the high generality of our framework. For future work, we plan to verify the effect of our framework in the other continual learning scenarios, including the CIL scenario without any exemplar [53], few-shot CIL [51], and online continual learning [2].

Acknowledgement. This work was partially supported by JSPS KAKENHI Grant Number JP21J21785.

References

1. Ahn, H., Kwak, J., Lim, S., Bang, H., Kim, H., Moon, T.: Ss-il: Separated softmax for incremental learning. In: ICCV. pp. 844–853 (2021)
2. Aljundi, R., Kelchtermans, K., Tuytelaars, T.: Task-free continual learning. In: CVPR. pp. 11254–11263 (2019)
3. Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K.O., Clune, J., Cheney, N.: Learning to continually learn. arXiv preprint arXiv:2002.09571 (2020)
4. Belouadah, E., Popescu, A.: Scail: Classifier weights scaling for class incremental learning. In: WACV. pp. 1266–1275 (2020)
5. Belouadah, E., Popescu, A., Kanellos, I.: Initial classifier weights replay for memoryless class incremental learning. In: BMVC (2020)
6. Belouadah, E., Popescu, A., Kanellos, I.: A comprehensive study of class incremental learning algorithms for visual tasks. Neural Networks **135**, 38–54 (2021)
7. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: ECCV. pp. 132–149 (2018)
8. Cha, H., Lee, J., Shin, J.: Co2l: Contrastive continual learning. In: ICCV. pp. 9516–9525 (2021)
9. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: ECCV. pp. 532–547 (2018)
10. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with A-GEM. In: ICLR (2019)
11. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486 (2019)
12. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML. pp. 1597–1607 (2020)
13. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
14. Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR. pp. 15750–15758 (2021)
15. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV. pp. 9640–9649 (2021)
16. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
17. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020)
18. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: ECCV. pp. 86–102 (2020)
19. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML. pp. 1126–1135 (2017)
20. Gallardo, G.J., Hayes, T.L., Kanan, C.: Self-supervised training enhances online continual learning. In: BMVC (2021)
21. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)
22. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., Kavukcuoglu,

K., Munos, R., Valko, M.: Bootstrap your own latent - a new approach to self-supervised learning. In: NeurIPS. vol. 33, pp. 21271–21284 (2020)
23. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR. pp. 9729–9738 (2020)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
25. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
26. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR. pp. 831–839 (2019)
27. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR. pp. 7132–7141 (2018)
28. Javed, K., White, M.: Meta-learning representations for continual learning. In: NeurIPS. vol. 32 (2019)
29. Ji, Z., Li, J., Wang, Q., Zhang, Z.: Complementary calibration: Boosting general continual learning with collaborative distillation and self-supervision. arXiv preprint arXiv:2109.02426 (2021)
30. Jodelet, Q., Liu, X., Murata, T.: Balanced softmax cross-entropy for incremental learning. In: ICANN. vol. 12892, pp. 385–396 (2021)
31. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. In: NeurIPS. vol. 33, pp. 18661–18673 (2020)
32. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017)
33. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
34. Kurmi, V.K., Patro, B.N., Subramanian, V.K., Namboodiri, V.P.: Do not forget to attend to uncertainty while mitigating catastrophic forgetting. In: WACV. pp. 736–745 (2021)
35. Li, Z., Hoiem, D.: Learning without forgetting. IEEE TPAMI **40**(12), 2935–2947 (2017)
36. Liu, X., Zhang, F., Hou, Z., et al.: Self-supervised learning: Generative or contrastive. IEEE TKDE (2021). https://doi.org/10.1109/TKDE.2021.3090866
37. Mai, Z., Li, R., Kim, H., Sanner, S.: Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In: CVPR. pp. 3589–3599 (2021)
38. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277 (2020)
39. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
40. Mermillod, M., Bugaiska, A., Bonin, P.: The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. Frontiers in Psychology **4**, 504 (2013)
41. Mittal, S., Galesso, S., Brox, T.: Essentials for class incremental learning. In: CVPR. pp. 3513–3522 (2021)

42. Ni, Z., Shi, H., Tang, S., Zhuang, Y.: Alleviate representation overlapping in class incremental learning by contrastive class concentration. arXiv preprint arXiv:2107.12308 (2021)
43. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV. pp. 69–84 (2016)
44. Pei, W., Mayer, A., Tu, K., Yue, C.: Attention please: Your attention check questions in survey studies can be automatically answered. In: WWW. pp. 1182–1193 (2020)
45. Pham, Q., Liu, C., Hoi, S.: Dualnet: Continual learning, fast and slow. In: NeurIPS. vol. 34, pp. 16131–16144 (2021)
46. Rajasegaran, J., Khan, S., Hayat, M., Khan, F.S., Shah, M.: itaml: An incremental task-agnostic meta-learning approach. In: CVPR. pp. 13588–13597 (2020)
47. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 2001–2010 (2017)
48. Simon, C., Koniusz, P., Harandi, M.: On learning the geodesic path for incremental learning. In: CVPR. pp. 1591–1600 (2021)
49. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
50. Tao, X., Chang, X., Hong, X., Wei, X., Gong, Y.: Topology-preserving class-incremental learning. In: ECCV. pp. 254–270 (2020)
51. Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., Gong, Y.: Few-shot class-incremental learning. In: CVPR. pp. 12183–12192 (2020)
52. Welling, M.: Herding dynamical weights to learn. In: ICML. pp. 1121–1128 (2009)
53. Wu, G., Gong, S., Li, P.: Striking a balance between stability and plasticity for class-incremental learning. In: ICCV. pp. 1124–1133 (2021)
54. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)
55. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: ICML. pp. 12310–12320 (2021)
56. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV. pp. 649–666 (2016)
57. Zhang, S., Shen, G., Huang, J., Deng, Z.H.: Self-supervised learning aided class-incremental lifelong learning. arXiv preprint arXiv:2006.05882 (2020)
58. Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S.T.: Maintaining discrimination and fairness in class incremental learning. In: CVPR. pp. 13208–13217 (2020)
59. Zhu, F., Zhang, X.Y., Wang, C., Yin, F., Liu, C.L.: Prototype augmentation and self-supervision for incremental learning. In: CVPR. pp. 5871–5880 (2021)