 Representation Compensation Networks for Continual Semantic Segmentation

Chang-Bin Zhang 1* Jia-Wen Xiao 1* Xialei Liu 1† Ying-Cong Chen 2,3 Ming-Ming Cheng 1
1 TMCC, CS, Nankai University 2 The Hong Kong University of Science and Technology (Guangzhou) 3 The Hong Kong University of Science and Technology

Abstract

In this work, we study the continual semantic segmentation problem, where the deep neural networks are required to incorporate new classes continually without catastrophic forgetting. We propose to use a structural re-parameterization mechanism, named representation compensation (RC) module, to decouple the representation learning of both old and new knowledge. The RC module consists of two dynamically evolved branches with one frozen and one trainable. Besides, we design a pooled cube knowledge distillation strategy on both spatial and channel dimensions to further enhance the plasticity and stability of the model. We conduct experiments on two challenging continual semantic segmentation scenarios, continual class segmentation and continual domain segmentation. Without any extra computational overhead and parameters during inference, our method outperforms state-of-the-art performance. The code is available at https://github.com/zhangchbin/RCIL.

1. Introduction

Data-driven deep neural networks [64, 72, 96, 108] have made many milestones in semantic segmentation. However, these fully-supervised models [16, 23, 93] can only handle a fixed number of classes. In real-world applications, it is preferable that a model can be dynamically extended to identify new classes. A straightforward solution is to rebuild the training set and retrain the model with all data available, known as Joint Training. However, considering the cost of re-training models, sustainable development of algorithms and privacy issues, it is particularly crucial to update the model with only current data to achieve the goal of recognizing both new and old classes. Nevertheless, naively fine-tuning a trained model with new data can result in catastrophic forgetting [48]. Therefore, in this paper, we seek continual learning, which can potentially enable a model to recognize new categories without catastrophic forgetting.

In the scenario of continual semantic segmentation [8, 27, 62, 63], given the previously trained model and the training data of the new classes, the model is supposed to distinguish all seen classes, including previous classes (old classes) and new classes. However, to save the labeling cost, the new training data often only has labels for the new classes, treating old classes as background. Learning with the new data directly without any additional designs is very challenging, which can easily lead to catastrophic forgetting [48].

As indicated in [28, 48, 51], fine-tuning the model on new data may lead to catastrophic forgetting, i.e., the model quickly fits the data distribution of the new classes, while losing the discrimination for the old classes. Some methods [43, 48, 66, 67, 80, 95] play regularization on model parameters to improve its stability. However, all parameters are updated on the training data of the new classes. This is however challenging, as new and old knowledge are entangled together in model parameters, making it extremely difficult to keep the fragile balance of learning new knowledge and keeping old ones. Some other methods [45, 57, 75, 76, 82, 91] increase the capacity of the model to have a better trade-off of stability and plasticity, but with the cost of growing memory of the network.

In this work, we propose an easy-to-use representation compensation module, aiming at remembering the old knowl-
edge while allowing extra capacity for new knowledge. Inspired by structural re-parameterization [24, 25], we replace the convolution layers in the network with two parallel branches during training, which is named as representation compensation module. As shown in Fig. 1, during training, the output of two parallel convolutions is fused before the non-linear activation layer. At the beginning of each continual learning step, we equivalently merge the parameters of the two parallel convolutions into one convolution, which will be frozen to retain the old knowledge. Another branch is trainable and it inherits the parameters from the corresponding branch in the previous step. The representation compensation strategy is supposed to remember the old knowledge using the frozen branch while allowing extra capacity for new knowledge using the trainable branch. Importantly, this module brings no extra parameters and computation cost during inference.

To further alleviate catastrophic forgetting [48], we introduce a knowledge distillation mechanism [70] between intermediate layers (shown in Fig. 1), named Pooled Cube Distillation. It can suppress the negative impact of errors and noises in local feature maps. The main contributions of this paper are:

- We propose a representation compensation module with two branches during training, one for retaining the old knowledge and one for adapting to new data. It always keeps the same computation and memory cost during inference as the number of tasks grows.

- We conduct experiments on continual class segmentation and continual domain segmentation, respectively. Experimental results demonstrate that our method outperforms the state-of-the-art performance on three different datasets.

2. Related Work

Semantic Segmentation. Early methods focused on modeling contextual relationships [2, 49, 103]. Currently methods pay more attention to multi-scale feature aggregation [3, 34, 52, 53, 59, 65, 68, 81]. Some methods [14, 22, 32, 37, 38, 50, 55] is inspired by Non-local [85], utilizing attention mechanisms to establish connections between image contexts. Another line of research [15, 61, 94] aimed at fusing features from different receptive fields. Recently, transformer architectures [7, 26, 86, 97, 104, 109] shine in semantic segmentation, focusing on multi-scale feature fusion [12, 84, 89, 101] and contextual feature aggregation [58, 79].

Continual Learning. Continual learning focuses on alleviating catastrophic forgetting while being discriminative for newly learned classes. To solve this problem, many work [4, 5, 11, 47, 77] propose to review knowledge by rehearsal-based mechanism. The knowledge can be stored by multiple types, like examples [4, 6, 9, 11, 73, 83], prototypes [35, 106, 107], generative networks [60], etc. Although these rehearsal-based methods usually achieve high performance, they need storage and authority for storing. In the more challenging scenario without any replay, many methods explore regularization to maintain old knowledge, including knowledge distillation [10, 18, 21, 28, 51, 69, 74, 100], adversarial training [29, 88], vanilla regularization [43, 48, 56, 66, 67, 80, 95, 98] and so on. Others focus on the capacity of the neural network. One of the research line [45, 57, 75, 76, 82, 91] is to expand the network architecture while learning new knowledge. Another research line [1, 44] explores the sparsity regularization for network parameters, which aims at activating as few neurons as possible for each task. This sparsity regularization reduces the redundancy in the network, while limiting the learning capacity for each task. Some work propose to learn better representations by combining self-supervised learning for feature extractor [9, 87] and solving class imbalance [39, 46, 54, 99, 102].

Continual Semantic Segmentation. Continual semantic segmentation is still an urgent problem to solve, mainly focusing on catastrophic forgetting [48] in semantic segmentation. In this field, continual class segmentation is a classic setting, with great progress made by several previous work: [41, 92] explore rehearsal-based methods to review old knowledge; MiB [8] models the potential classes to solve the ambiguous of background class; PLOP [27] applies knowledge distillation strategy to intermediate layers; SDR [63] takes advantage of prototype matching to perform consistency constraints in the latent space representation. While others [31, 78, 95] utilize high-dimensional information, self-training and model adaptation to overcome this problem. Moreover, continual domain segmentation is a novel setting proposed by PLOP [27], aiming at integrating new domain rather than new classes. Different from previous methods, we focus on expanding the network dynamically, decoupling the representation learning of old classes and new classes.

3. Method

3.1. Preliminaries

Let \( D = \{ x_i, y_i \} \) denotes the training set, where \( x_i \) denotes the input image and \( y_i \) is the corresponding segmentation ground-truth. In the challenging continual learning scenario, we call each training on the newly added dataset \( D_t \) as a step. At step \( t \), given a model \( f_{t-1} \) with parameter \( \theta_{t-1} \) trained on \( \{ D_0, D_1, ..., D_{t-1} \} \) with \( \{ C_0, C_1, ..., C_{t-1} \} \) classes continually, the model is supposed to learn the discrimination for \( \sum_{n=0}^{t} C_n \) classes when it encounters a newly added dataset \( D_t \) with extra \( C_t \) new classes. When training on \( D_t \), the training data of old classes are not accessible. Besides, to save the training cost, the ground-truth in \( D_t \) only contains the \( C_t \) new classes, while the old classes are labeled as
3.2. Representation Compensation Networks

To decouple the retaining of old knowledge and learning of new knowledge, as shown in Fig. 2, we introduce our representation compensation mechanism. In most of the deep neural networks, a $3 \times 3$ convolution followed by normalization and non-linear activation layer is a common component. We modify this architecture by adding a parallel $3 \times 3$ convolution followed by a normalization layer for each component. The output of two parallel convolution-normalization layers is fused, then is rectified by a non-linear activation layer. Formally, this architecture contains two parallel convolution layers with weight $\{W^0, W^1\}$ and bias $\{b^0, b^1\}$, followed by two independent normalization layers, respectively. Let $Norm^0 = \{\mu^0, \sigma^0, \gamma^0, \beta^0\}$ and $Norm^1 = \{\mu^1, \sigma^1, \gamma^1, \beta^1\}$ denote the mean, variance, weight and bias of two normalization layers $Norm^0$ and $Norm^1$. Thus, the calculation of input $x$ before non-linear activation function can be denoted as

$$\hat{x} = \sum_{i=0}^{1} Norm_i(W_i x + b_i)$$

$$= \sum_{i=0}^{1} \left( \gamma_i W_i x + b_i - \mu_i \right) \frac{1}{\sigma_i} + \beta_i$$

$$= \sum_{i=0}^{1} \frac{\gamma_i W_i x}{\sigma_i} + \sum_{i=0}^{1} \frac{\gamma_i b_i - \gamma_i \mu_i}{\sigma_i} + \beta_i$$

$$= W x + b.$$  

(1)

This equation demonstrates that two parallel branches can be equivalently represented as one with weight $W$ and bias $b$. We also display the transformation in the right part of Fig. 2. Therefore, for this modified architecture, we can equivalently merge the parameters of two branches into one convolution.

More precisely, in step 0, all parameters are trainable to train a model that can discriminate $C_0$ classes. For the subsequent learning steps, the model is supposed to segment newly added classes. In these continual learning steps, the network will be initialized with the parameters trained in the previous step, which is beneficial to transfer knowledge [8]. At the beginning of step $t$, since the model is supposed to avoid forgetting old knowledge, we merge the parallel branches trained in step $t - 1$ to one convolution layer. The parameters in this merged branch are frozen to memorize the old knowledge, as shown in Fig. 2. Another branch is trainable to learn new knowledge, which is initialized with the corresponding branch in the previous step. Besides, we design a drop-path strategy, which is applied on aggregating the output, $x_1$ and $x_2$ from two branches. During training, the output before the non-linear activation is denoted as

$$\hat{x} = \eta \cdot x_1 + (1 - \eta) \cdot x_2,$$

(2)

where $\eta$ is the random channel-wise weighted vector and sampling from the set $\{0, 0.5, 1\}$ uniformly. During inference, the element of vector $\eta$ is set as 0.5. Experimental results demonstrate that this strategy brings slight improvement.

Analysis on RC-Module’s Effectiveness. As shown in Fig. 3, the parallel convolution structure can be regarded as an implicit ensemble [36, 40] of numerous sub-networks. The parameters of some layers in these sub-networks are inherited from the merged teacher model (trained at previous step) and are frozen. During training, similar to [33, 90], these frozen teacher layers will impose regularization to trainable parameters, encouraging trainable layers to behave like the teacher model. In a special case where only one layer in the sub-network is trainable, as shown in Fig. 3(a), during training, this layer will take into account both adapting for the representation of frozen layers and learning for new knowledge. Therefore, this mechanism will alleviate catastrophic forgetting of the trainable layer. We further promote this effect to general sub-networks like Fig. 3(b),
which will also encourage the trainable layers to adapt to the representation of the frozen layers. Furthermore, all sub-networks are ensembled, integrating knowledge from different sub-networks to one network, like Fig. 3(c).

### 3.3. Pooled Cube Knowledge Distillation

In order to further alleviate the forgetting of old knowledge, following PLOP [27], we also explore feature distillation. As shown in Fig. 4(a), PLOP [27] introduces strip pooling [38] to integrate features. Pooling operation plays a key role in transferring knowledge. In our method, we design the average pooling-based knowledge distillation along the spatial dimension. Additionally, we use the average pooling in the channel dimension at each position as well to maintain their individual activation intensity. Overall, as shown in Fig. 4(b), we use the average pooling on both spatial and channel dimensions.

Formally, we select feature maps \( \{X^1, X^2, ..., X^L\} \) before the last non-linear activation layer for all \( L \) stages, including decoder and all stages in the backbone. For the features from the teacher model and the student model, we firstly calculate the square of value at each pixel to retain the negative information. Then, we perform multi-scale average pooling on spatial and channel dimensions, respectively. The features \( X^l_T, X^l_S \) of the teacher model and the student model can be calculated by the average pooling operation \( \odot \):

\[
\begin{align*}
\hat{X}^l_{T,m} &= M \odot [(X^l_{T,ij})^2] \\
\hat{X}^l_{S,m} &= M \odot [(X^l_{S,ij})^2],
\end{align*}
\]

where \( M \) denotes the \( m_{th} \) average pooling kernel, and \( l \) denotes the \( l_{th} \) stage. For the average pooling on the spatial dimension, we use the multi-scale windows to model the relationships between pixels in the local region. The size of kernel \( M \) belongs to \( M = \{4, 8, 12, 16, 20, 24\} \) and the step size is set to 1. And we simply set the window size as 3 for the average pooling on channel dimension. Then, the spatial knowledge distillation loss function \( L_{skd} \) for the intermediate layers can be denoted as

\[
L_{skd} = \frac{1}{L} \frac{1}{|M|} \sum_{l=1}^{L} \sum_{m=1}^{|M|} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{d=1}^{D} [(\hat{X}^l_{T,ijd} - \hat{X}^l_{S,ijd})^2],
\]

where \( H, W, D \) denote the height, width and the number of channels. The same equation can be applied on channel dimension with \( M = \{3\} \) to form \( L_{ckd} \). Overall, the distillation objective can be denoted as:

\[
L = L_{skd} + L_{ckd}.
\]

**Average pooling vs. Strip pooling.** Benefiting from its strong ability to aggregate features and model long-range dependency, strip pooling shines in many fully-supervised semantic segmentation models [38, 42]. The performance of continual segmentation is still much worse than that of fully-supervised segmentation. In the scenario of continual segmentation, there are often more noise or errors in the prediction results than fully-supervised segmentation. Thus, in the distillation process, when using strip pooling to aggregate features, this long-range dependency will introduce some uncorrelated noise to the cross point, causing noise diffusion. This will lead to further deterioration of the prediction results of the student model. In our method, we use average pooling in the local region to suppress the negative impact of noise. Specifically, because the semantics of local regions are often similar, the current key point can find more neighbors to support its decision by aggregating features in the local region. Thus, the current key point is less negatively affected by the noise in the local region.

As an example shown in Fig. 5(b) top, the strip pooling introduces noise or errors to the cross point for the teacher model. During the distillation process, the noise is further propagated to the student model, making the noise diffusion. For the average pooling in Fig. 5 bottom, the key point will...
1,525 test images. There are 19 classes from 21 cities. All other classes are treated as background. ADE20K [105] is a dataset for semantic segmentation covering daily life scenes. It contains 20,210 training images and 2,000 validation images with 20 object classes and the background class. Cityscapes [19] contains 2,975 training images, 500 validation images and 1,525 test images. There are 19 classes from 21 cities.

4. Experiments

In this section, we first demonstrate the details of our experimental setups, e.g., datasets, protocols and training details. Then we illustrate the effectiveness of our method from quantitative and qualitative experiments.

4.1. Experimental setups

4.1.1 Datasets

PASCAL VOC 2012 [30] is a commonly used dataset, which contains 10,582 training images and 1449 validation images with 20 object classes and the background class. ADE20K [105] is a dataset for semantic segmentation covering daily life scenes. It contains 20,210 training images and 2,000 validation images with 150 classes. Cityscapes [19] contains 2,975 training images, 500 validation images and 1,525 test images. There are 19 classes from 21 cities.

4.1.2 Protocols

Continual Class Segmentation. In continual class segmentation, the model is trained to recognize different classes sequentially in multiple steps. Each step the model learns one or several classes. Following [8, 27, 63], we assume training data of previous steps are not available, i.e., the model can only access data of the current step. Besides, only classes to be learned in the current step are labeled. All other classes are treated as background. There are two commonly used settings proposed by [8] for continual class segmentation, disjoint and overlapped. In the disjoint setting, assuming we know all classes in the future, the images in the current training step do not contain any classes in the future. The overlapped setting is more realistic. It allows potential classes in the future to appear in the current training images.

We conduct continual class segmentation experiments on the PASCAL VOC 2012 [30] and ADE20K [105]. Following [8, 27, 63], as defined in Sec. 3.1, we call each training on the newly added dataset as a step. Formally, $X$-$Y$ denotes the continual setting in our experiments, where $X$ denotes the number of classes that we need to train in the first step. In each subsequent learning step, the newly added dataset contains $Y$ classes. On PASCAL VOC 2012 [30], we conduct experiments on three settings, 15-5 (2 steps), 15-1 (6 steps) and 10-1 (11 steps). For example, 15-1 denotes that we train the model on the initial 15 object classes in the first step. In the subsequent five steps, the model is expected to be trained on new datasets, where each dataset contains one new added class. Thus, the model can discriminate 20 object classes in the last step. On ADE20K [105], we apply four settings, 100-50 (2 steps), 50-50 (3 steps), 100-10 (6 steps), and 100-5 (11 steps).

Continual Domain Segmentation. It is proposed by [27]. Different from continual class segmentation, this setting is to deal with the domain shift phenomenon rather than integrating new classes. In the real-world scene, domain shift can also occur frequently. We assume the classes in different domains are the same. The training data of the old domain is not accessible when training on new domain data. We conduct continual domain segmentation experiments on Cityscapes [19]. Following PLOP [27], we regard the training data in each city as a domain. We also apply three settings, 11-5 (3 steps), 11-1 (11 steps) and 1-1 (21 steps). In these experimental settings, we use the same recording as the continual class segmentation, but each step adds new domains (cities) instead of classes.

4.1.3 Implementation Details

Following [8, 27, 63], we use the Deeplab-v3 [13] architecture with ResNet-101 [36] as backbone. The output stride of Deeplab-v3 is set to 16. We also apply the in-place activated batch normalization [71] in the backbone pre-trained on the ImageNet [20], as the above methods. We utilized the loss function proposed by MiB [8] to assist our training process. And we apply the same training strategy as [8, 27, 63]. Specifically, we apply the same data augmentation, e.g., horizontal flip and random crop. The batch size is set to 24 for all experiments. We set the initial learning rate as 0.02 for the first training step and 0.001 for the next continual learning steps. The learning rate is adjusted by the poly schedule. We train the model using SGD optimizer for each step with 30 (PASCAL VOC 2012 [30]), 50 (Cityscapes [19]), and 60 epochs (ADE20K [105]), respectively. We also use 20% of the training set as validation following [8, 27, 63]. We report the mean Intersect over Union (mIoU) on the original validation set.

4.2. Continual Class Segmentation

PASCAL VOC 2012. Applying the same experimental settings as [8, 27, 63], we performed experiments on different
Table 1. The mIoU(%) of the last step on the Pascal VOC 2012 dataset for different continual class segmentation scenarios. The red denotes the highest results and the blue denotes the second highest results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Disjoint</th>
<th>Overlapped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-15</td>
<td>16-20</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>5.7</td>
<td>33.6</td>
</tr>
<tr>
<td>Joint</td>
<td>79.8</td>
<td>72.6</td>
</tr>
<tr>
<td>LwF [51]</td>
<td>60.4</td>
<td>37.4</td>
</tr>
<tr>
<td>ILT [62]</td>
<td>64.9</td>
<td>39.5</td>
</tr>
<tr>
<td>MiB [8]</td>
<td>73.0</td>
<td>43.3</td>
</tr>
<tr>
<td>SDR [63]</td>
<td>74.6</td>
<td>44.1</td>
</tr>
<tr>
<td>PLOP [27]</td>
<td>71.0</td>
<td>42.8</td>
</tr>
<tr>
<td>Ours</td>
<td>75.0</td>
<td>42.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Disjoint</th>
<th>Overlapped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-15</td>
<td>16-20</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>4.6</td>
<td>1.8</td>
</tr>
<tr>
<td>Joint</td>
<td>79.8</td>
<td>72.6</td>
</tr>
<tr>
<td>LwF [51]</td>
<td>6.6</td>
<td>33.1</td>
</tr>
<tr>
<td>ILT [62]</td>
<td>6.6</td>
<td>12.9</td>
</tr>
<tr>
<td>MiB [8]</td>
<td>7.9</td>
<td>39.9</td>
</tr>
<tr>
<td>SDR [63]</td>
<td>7.9</td>
<td>14.3</td>
</tr>
<tr>
<td>PLOP [27]</td>
<td>7.9</td>
<td>21.1</td>
</tr>
<tr>
<td>Ours</td>
<td>7.9</td>
<td>23.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Disjoint</th>
<th>Overlapped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-10</td>
<td>11-20</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>6.3</td>
<td>1.1</td>
</tr>
<tr>
<td>Joint</td>
<td>78.2</td>
<td>78.0</td>
</tr>
<tr>
<td>LwF [51]</td>
<td>6.4</td>
<td>1.2</td>
</tr>
<tr>
<td>ILT [62]</td>
<td>7.2</td>
<td>1.2</td>
</tr>
<tr>
<td>MiB [8]</td>
<td>7.2</td>
<td>1.2</td>
</tr>
<tr>
<td>SDR [63]</td>
<td>7.2</td>
<td>1.2</td>
</tr>
<tr>
<td>PLOP [27]</td>
<td>7.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Ours</td>
<td>7.2</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Figure 6. The mIoU (%) at each step in three experimental settings. (a)(b) are settings of continual class segmentation. (c) is the setting of continual domain segmentation.

Continuous learning settings, 15-5, 15-1 and 10-1. As shown in Tab. 1, we report the experimental results of the last step. The vanilla fine-tuning method suffers from the catastrophic forgetting phenomena. The model quickly forgets the old knowledge and is unable to learn the new knowledge well. Experimental results demonstrate that our method significantly improves the segmentation performance both on the overlapped and disjoint settings. Especially in the challenging 15-1 settings, our method outperforms the state-of-the-art by 6.0% (disjoint) and 4.8% (overlapped) in terms of mIoU, respectively. We also display the performance of each step for different methods as shown in Fig. 6a and Fig. 6b. This demonstrates that our method can reduce the forgetting of old knowledge in the continual learning process. In Tab. 1, we also report the performance over the old classes and new classes, respectively. For all settings, the performance of the old classes is greatly improved. This is benefited from the representation compensation module and distillation mechanism, which can effectively retain the old knowledge. On the other hand, our proposed representation module and distillation mechanism allow room for learning new knowledge. In Sec. 4.4, we will further analyze the effectiveness of these two mechanisms. We further show the qualitative results of different methods in the 15-1 overlapped setting in Fig. 7.

**ADE20K.** To verify the effectiveness of our method, we conduct experiments on a challenging semantic segmentation dataset, ADE20K [105]. Experimental results are shown in Tab. 2 and Tab. 3. On different continual learning tasks, 100-50, 100-10 and 50-50, our method achieves an average improvement of 1.4% over the state-of-the-art. To further verify our method, we also perform experiments on a more challenging scenario, 100-5, which contains 11 steps. In this scenario, our method also achieves the state-of-the-art, outperforming the previous method by about 0.9% in terms of mIoU, as shown in Tab. 3. The improvement is due to our proposed representation compensation module and pooled cube distillation mechanism.

### 4.3. Continual Domain Segmentation

In the context of continual semantic segmentation, in addition to the need to segment new classes, it is also of great significance to increase the processing capabilities of new domains. Following [27], we conducted experiments of continual domain semantic segmentation on Cityscapes [19]. Each city in Cityscapes [19] can be regarded as a domain,
which is widely used by domain adaptive semantic segmentation tasks [17]. In this scenario, we do not consider the difference in classes between domains. As shown in Tab. 4, experimental results demonstrate that our method achieves higher mIoU than previous methods [8, 27, 62] in all three settings. Our method outperforms the state-of-the-art by 3.7% on the challenging 1-1 setting with 21 learning steps. For this setting, we display the performance of each step in Fig. 6c. Since MiB [8] aims at solving the problem of semantic shift which is not existing in continual domain segmentation, MiB [8] performs slightly worse than Fine-tuning. These experiments indicate that our method is also effective for continual domain semantic segmentation, benefiting from the ability to retain old knowledge while allowing to learn new knowledge.

### 4.4. Ablation Study

In this section, we firstly analyze the effectiveness of our proposed representation compensation and pooled cube distillation mechanism. Then we discuss the robustness to class orders in the continual learning scenario.

**Representation Compensation.** We conduct ablation experiments on PASCAL VOC 2012 [30].

As shown in Tab. 5, our proposed representation compensation module achieves about 7% improvement than the MiB [8] baseline. With this module, our method reaches state-of-the-art performance. We argue this performance benefits from the scheme of remembering old knowledge in our method while allowing the learning for new knowledge. In our method, the operations of merging and freezing parameters aim at alleviating the forgetting of old knowledge. These experiments indicate that our method is also effective for continual domain semantic segmentation, benefiting from the ability to retain old knowledge while allowing to learn new knowledge.

**Distillation Mechanism.** In Tab. 5, we study the importance of knowledge distillation mechanism on spatial and channel dimensions, respectively. The knowledge distillation
on spatial and channel dimensions achieves similar performance, outperforming baseline by about 15.3% in terms of mIoU. With the representation compensation module, the combination of these two distillation schemes can reach state-of-the-art performance. We further compare the effectiveness of different pooling methods used in the knowledge distillation mechanism, as shown in Tab. 7. Experimental results demonstrate that average pooling outperforms strip pooling by 1.5%.

**Robustness to Class Orders.** In the scenario of continual semantic segmentation, the class orders in the pipeline is particularly important. To verify the robustness to class orders, we perform experiments on five different class orders, including four random orders and the original ascending order. In Fig. 8, we display the average performance and standard variance for different methods [8, 27, 62, 63]. Experimental results demonstrate that our method is more robust against different class orders than previous methods.

### 5. Conclusion and Limitation

In this work, aiming at remembering the knowledge for old classes while allowing capacity for learning new classes, we propose the representation compensation module, which dynamically expands the network without any extra inference cost. Besides, to further alleviate the forgetting for old knowledge, we propose Pooled Cube Distillation mechanism on spatial and channel dimensions. We conduct experiments on two commonly used benchmarks, continual class segmentation and continual domain segmentation. Our method outperforms state-of-the-art performance.

Although we have proposed two components, which outperform the state-of-the-art performance, we have a poor performance in the continual learning process with many steps, like 10-1 setting shown in Tab. 1. In these challenging scenarios, how to improve the performance of the model still has a long way to go. Besides, our method requires more computation costs during training.

**Acknowledgment** This work is funded by the National Key Research and Development Program of China (NO. 2018AAA0100400) ans NSFC (NO. 61922046), and S&T innovation project from Chinese Ministry of Education.
References


[100] Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. Delving deep into label smoothing. IEEE Trans. Image Process., 2021. 2