Preparing the Future for Continual Semantic Segmentation

Zihan Lin[1] Zilei Wang[* 1] Yixin Zhang[1,2]
[1] University of Science and Technology of China
[2] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
myustc@mail.ustc.edu.cn, {zlwang, zhyx12}@ustc.edu.cn

Abstract

In this study, we focus on Continual Semantic Segmentation (CSS) and present a novel approach to tackle the issue of existing methods struggling to learn new classes. The primary challenge of CSS is to learn new knowledge while retaining old knowledge, which is commonly known as the rigidity-plasticity dilemma. Existing approaches strive to address this by carefully balancing the learning of new and old classes during training on new data. Differently, this work aims to avoid this dilemma fundamentally rather than handling the difficulties involved in it. Specifically, we reveal that this dilemma mainly arises from the greater fluctuation of knowledge for new classes because they have never been learned before the current step. Additionally, the data available in incremental steps are usually inadequate, which can impede the model's ability to learn discriminative features for both new and old classes. To address these challenges, we introduce a novel concept of pre-learning for future knowledge. Our approach entails optimizing the feature space and output space for unlabeled data, which thus enables the model to acquire knowledge for future classes. With this approach, updating the model for new classes becomes as smooth as for old classes, effectively avoiding the rigidity-plasticity dilemma. We conducted extensive experiments and the results demonstrate a significant improvement in the learning of new classes compared to previous state-of-the-art methods.

1. Introduction

Deep neural networks have demonstrated their superiority in many computer vision tasks [26, 31, 14]. Conventionally, they are trained in an offline manner with all data collected beforehand. When novel classes need to be handled, the networks need to be re-trained using either updated training set with both new and previous data, or directly finetuned on new data. The former has higher costs and is even impracticable sometimes like, when privacy raises concerns. The latter is prone to forget previously learned knowledge, which is called catastrophic forgetting (CF) [25]. To better fit neural networks into real-world scenarios, we want them to have the ability to learn new concepts while preserving the old knowledge (or even improve it using new knowledge). This is known as the continual learning (CL) problem and receives increasing attention recently. It has been explored first in the field of image classification [28, 42, 17, 11, 57, 44, 43], and some methods are proposed recently targeting semantic segmentation [5, 10, 6, 54, 51, 37]. Generally, it is often achieved by applying forgetting-preventing constraints between current and previous networks to prevent CF, which raises the rigidity (*i.e.*, the ability to preserve old states) but harms the plasticity (*i.e.*, the ability to learn new knowledge). It is known as the rigidity-plasticity (R-P) dilemma. A key aspect in designing constraints is to balance between rigidity and plasticity, which is what most works focus on.

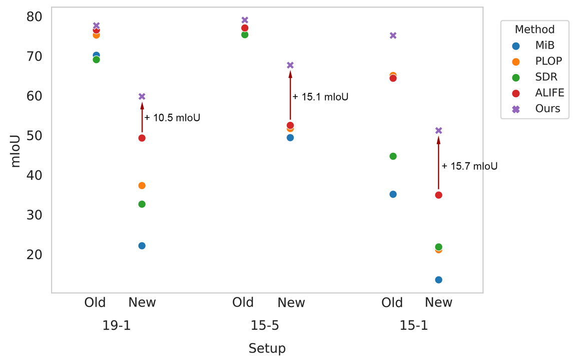

Figure 1. On VOC2012 benchmark, our method boosts the performance of new classes by a significant margin, while also achieving promising performance for old classes, compared to recent state-of-the-art methods.

Despite that several new constraints have been proposed for CSS to better tackle the dilemma, existing works still struggle to learn new classes. Taking the VOC2012 benchmark as an example, most methods can only achieve a performance around 70% of that of the upperbound for *15-5* setup and around 50% for *19-1* setup, which is even worse for harder setups. To get a clear picture of why this happens, we first analyze the intermediate features of the network and find it suffers serious underfitting for new classes, which indicates a large amount of knowledge is not learned. We then examine the CSS training pipeline and find two main obstacles causing this phenomenon. (1) R-P Dilemma: the network has no knowledge w.r.t. new classes until they are introduced. This generally requires a rather large update to incorporate the new knowledge, which will be significantly hindered by the forgetting-preventing constraints punishing the network from drifting away from its previous states. (2) Inadequate data for training: the amount of data available in incremental steps is usually limited, and this problem becomes more severe when a single class is added, resulting in a tiny training set with only a few negative samples. This can significantly limit the network to properly discriminate between old and new classes. This problem has never been explicitly considered before.

Unlike existing works trying to tackle the R-P dilemma directly, we are the first to argue that this obstacle can be elegantly avoided by exploring future knowledge in an unsupervised way. Assume that the network has obtained partial knowledge of future classes in advance, then the network needs fewer updates to learn the upcoming knowledge, which consequently lowers the impacts of the R-P dilemma. The more future knowledge it learns in advance, the fewer the impacts will be caused. Considering an extreme situation where all the future knowledge is learned beforehand, the network does not need any update, and thus the influence of constraints could be totally avoided. On top of this, it makes us the first attempt to explicitly tackle the second obstacle by utilizing the abundant data in previous steps to optimize discriminability between old and new classes. Based on these understandings, we design a novel framework where unsupervised contrastive learning is performed on intermediate features to optimize future classes with both visual similarity and feature affinity as supervisions. One step further, an auxiliary classifier is trained using pseudo labels of future classes generated via clustering, which is used to initialize the actual classifier for future steps. Our method surpasses existing works by a huge margin as shown in Figure 1 and the effectiveness is further demonstrated with extensive experiments. Our contributions are summarized as follows.

- We propose to pre-learn future knowledge for CSS with the aim of improving the performance especially for new classes, which can simultaneously tackle the R-P dilemma and the obstacle of inadequate data.
- We design a novel framework that can pre-learn future knowledge in an unsupervised way, where both the feature space and output space are handled.
- We have validated the effectiveness of our proposed approach through extensive experiments. The results demonstrate significant improvement on standard CSS benchmarks.

2. Related Work

Continual Learning (CL). Continual learning has been studied for years in image classification. Existing techniques can be roughly divided into regularization methods, rehearsal methods, and architectural methods. Regularization methods apply consistency constraints between current network and previous network to avoid drifts from previous states. They can be applied on network weights [53, 25, 1], the intermediate features [9, 11, 45, 13, 22] or the output logits [28, 42, 17, 48]. Parameter regularization is rarely seen recently due to the lack of plasticity, while the other two are mostly based on knowledge distillation (KD) [16]. To better solve the forgetting problem, rehearsal methods replay previous data in the incremental steps, including image replay [42, 41, 3], generative replay [49, 39], feature replay [20, 57], and memory management [30, 29]. Architectural methods use a dynamic network to handle incoming knowledge, like sub-netwok [32, 33], complementary network [47, 40] or expandable network [27, 50, 52].

Continual Semantic Segmentation (CSS). Some works extended the aforementioned techniques to semantic segmentation tasks. For the KD-based methods, MiB [5] formalized and targeted background shift problem with output space distillation. PLOP [10], SDR [36], and RCIL [54] use feature space distillation to further mitigate forgetting. For the rehearsal-based methods, SSUL [6] and RECALL [34] utilize memory or external data to overcome forgetting and work well in multiple-step learning. For the architectural methods, RCIL [54] proposes a structural reparameterization design to better learn new knowledge.

Unsupervised Semantic Segmentation. These works focus on learning to cluster semantically related pixels without human labeling. Existing works can be roughly divided into three types. PiCIE [8] iteratively performs clustering and uses it to refine network features. This relies on the architectural prior and is prone to degenerate solutions. IIC [21] maximizes the mutual information between two augmentations of the same image which enforces spatial stability on them. But this solution has trouble dealing with complex scenes. The last type uses visual similarity. These works [19, 55, 46, 23] use handcrafted priors, like boundary, superpixels, or saliency detectors to first group pixels by visual similarity. This is then used to learn pixel-level representations using contrastive learning.

3. Preliminaries

In this section, we first explain the definition of CSS (Sec. 3.1), and present the underfitting phenomenon of new

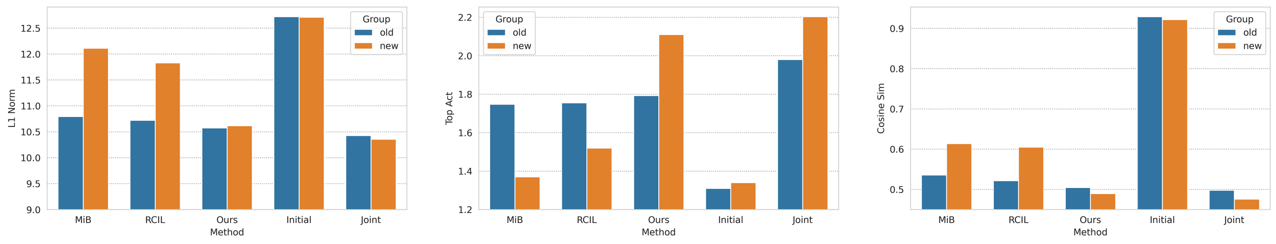

Figure 2. Feature fitness analysis of different methods using L1 norm (**Left**), activation strength (**Mid**), and cosine similarity (**Right**) on VOC2012 15-5 setup. It is demonstrated that existing works suffer underfitting on both intra-class level (left and mid) and inter-class level (right) for new classes. Our method effectively addresses this problem.

classes (Sec. 3.2). After that, we analyze how the performance is impacted in CSS (Sec. 3.3). Finally, we discuss the solutions to boost new class performance (Sec. 3.4).

3.1. CSS Definition

In CSS, the training procedure is composed of a series of T learning steps. At each step t where $t \in \{1, ..., T\}$, a set of new categories $\mathcal{C}_t$ are introduced with a training set $D_t = \{(x_n, y_n)\}$ where x_n is an image and y_n is the corresponding label. Only $\mathcal{C}_t$ is labeled in y_n and all the other classes are labeled as background. It is assumed that the categories learned at each step are disjoint, *i.e.*, $\mathcal{C}_i \cap \mathcal{C}_j = \varnothing$. The network is trained sequentially on each step, where it learns the knowledge w.r.t. $\mathcal{C}_t$ and expands its output space to $\mathcal{C}_{1:t} = \mathcal{C}_{1:t-1} \cup \mathcal{C}_t$. After all learning steps, the network is expected to perform well on all classes.

3.2. Analysis of Feature Fitness

To get a clear view of why new classes perform poorly in existing methods, we analyze the intermediate features of two works: MiB [5] based on conventional logits distillation, and a more recent work, RCIL [54], based on feature distillation and structural re-parameterization aiming to better tackle the R-P dilemma. To simplify the situation, we opt for the two-step 15-5 setup on VOC2012 to conduct the analysis. The results are reported in Figure 2 (Detailed information about incremental setups and validation set can be found in Section 5.). The results of joint training (*i.e.*, performance upperbound) and initial state (*i.e.*, backbone is initialized with ImageNet pre-trained weights while the other parts are randomly initialized) are also provided for reference. Ideally, the results are expected to be close to that of the upperbound in order to achieve good performance.

Intra-class Fitness. We first evaluate how well the network fits to each individual class. In neural networks, each channel in the feature vectors could represent a certain pattern [4] which composes the learned classes [58]. When the network fits a class well, the output feature vector should have specific channels being activated while the others deactivated, resulting in sparsity in the feature vector. The activation strength further shows how strong the network's response is to each pattern. Both of them can serve as clues indicating how well the network fits a class. We use the average L1 norm of L2-normalized feature vectors to measure the sparsity, and the average top activation values to measure the response strength. As can be seen in Figure 2 (left and mid), the new classes of both MiB and RCIL have high L1 norm and low activation strength, indicating a serious underfitting problem.

Inter-class Fitness. We further evaluate how well the network learns to distinguish different classes, which is measured by the cosine similarity between the feature vectors of different classes. The better the network can distinguish each class, the lower the cosine similarities will be. We compute the mean cosine distance of a class to any other classes, which is then averaged based on class group to get the final results, as shown in Figure 2 (right). It can be seen that the network has good discriminative power on old classes, but the new classes again present poor results. It indicates the network cannot distinguish a new class between other classes well.

Discussion. From the above analyses, it is evident that new classes suffer serious underfitting on both intra-class and inter-class aspects. This exactly explains why most methods show low performance on new classes, even in simple setups with only a single incremental step.

3.3. Understanding Training Pipeline

In this part, we dive deeper to investigate why the underfitting problem happens for new classes in the conventional training pipeline. We use Figure 3 to give a clear view of how the two obstacles impact the performance in a conventional pipeline (green background). Without loss of generality, a two-step situation is adopted as an example.

Starting from step 1 where $\mathcal{C}_1$ is labeled in the given data D_1 while others are marked as background, the network acquires most knowledge of $\mathcal{C}_1$ (*i.e.*, distinguish between classes in $\mathcal{C}_1$) after finishing this step. Considering that it only accesses partial data at this step compared to joint training, the knowledge of $\mathcal{C}_1$ remains to be updated

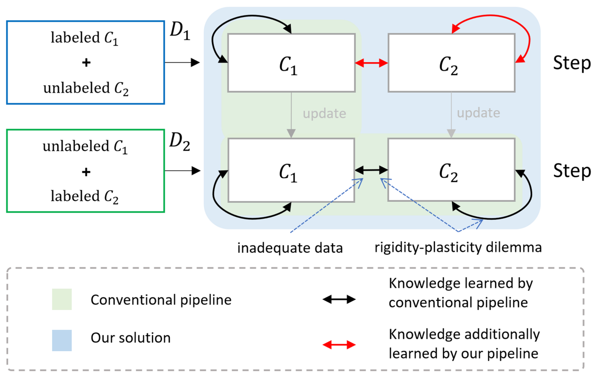
labeled C_1 + unlabeled C_2
D_1
unlabeled C_1 + labeled C_2
D_2
C_1
C_2
update
update
Step 1
Step 2
inadequate data
rigidity-plasticity dilemma
Conventional pipeline
Our solution
Knowledge learned by conventional pipeline
Knowledge additionally learned by our pipeline

Figure 3. Illustration of the difference between knowledge learning in a conventional training pipeline (green background) and our solution (blue background). We jointly optimize new classes together with old classes in an unsupervised way *before* new classes are introduced.

in the next step. However, even though the future classes $\mathcal{C}_2$ do exist in D_1, they do not receive explicit supervision, thus the network obtains almost no knowledge of $\mathcal{C}_2$ after finishing step 1.

Moving to step 2, a constraint is usually applied to avoid forgetting previous knowledge by punishing the network from drifting away from its previous states. In this step, the network needs to update its knowledge of $\mathcal{C}_1$ on D_2. Given that $\mathcal{C}_1$ is already properly optimized at step 1, the network only needs a small adjustment to complete its knowledge of $\mathcal{C}_1$ and receives little punishment from the constraint. However, the network is also intended to learn the knowledge w.r.t. $\mathcal{C}_2$ (*i.e.*, distinguish the classes in $\mathcal{C}_2$ and distinguish between $\mathcal{C}_2$ and $\mathcal{C}_1$) from the start, which requires a rather large update. Based on the fact that the constraints make more punishment for greater changes, this update will be suppressed more severely compared to that of $\mathcal{C}_1$, making more knowledge w.r.t. $\mathcal{C}_2$ cannot be learned. This is why the **R-P dilemma** often leads to low performance for new classes. To make matters worse, the scale of the training set is usually limited in incremental steps, especially for the situation when a single class is added with only a few samples of $\mathcal{C}_1$. Such limited data could impede the network from successfully distinguishing between $\mathcal{C}_1$ and $\mathcal{C}_2$, which is the **inadequate data** obstacle. Both of these factors result in the inability to properly learn knowledge w.r.t. $\mathcal{C}_2$ and contribute to the aforementioned underfitting phenomenon.

3.4. Discussions for Solutions

Pre-learning future knowledge. Based on the analysis, if we modify the training pipeline by adding explicit supervision to $\mathcal{C}_2$ *before* it is introduced (blue background in Figure 3). The knowledge w.r.t. $\mathcal{C}_2$ could be learned in advance. By doing so, we first avoid the R-P dilemma by reducing the great update to learning the knowledge w.r.t. $\mathcal{C}_2$ at the next step. Meanwhile, we can use the abundant samples of $\mathcal{C}_1$ at step 1 to make the network better learn how to distinguish between $\mathcal{C}_2$ and $\mathcal{C}_1$, mitigating the influence of inadequate data of $\mathcal{C}_1$ at next step. Consequently, both obstacles are tackled in a uniform and elegant way. The main challenge falls to how we can learn knowledge w.r.t. $\mathcal{C}_2$ as accurately as possible in an unsupervised way in order to reduce the extent of updates required in the future.

Memory. Memory is a widely used technique in CL and has been adapted to CSS recently [6, 34]. Based on our analysis above, here we clearly explain the effect of memory in CSS. Memory is applied only in incremental steps. With the help of it, new classes are possible to be optimized with partial previous data, which can also alleviate the impact of inadequate data. Meanwhile, memory helps to ease the forgetting problem, which makes it possible to relax the constraints without forgetting. As a result, new knowledge can be more easily learned. We would like to point out that memory can be used to solve the two obstacles at a different aspect. It could further boost the performance in conjunction with our solution.

4. Method

In this section, we introduce a novel framework based on pre-learning future knowledge as depicted in Figure 4. It takes into account future knowledge in an unsupervised way, both in the feature space (green background) and the output space (blue background).

4.1. Learning Future Knowledge in Feature Space

Contrastive learning. Contrastive learning is a widely used technique in unsupervised learning [7, 15]. In our work, we resort to it to optimize future classes in feature space. In semantic segmentation, directly using feature vectors at pixel level to perform *pixel-pixel* contrastive is redundant and results in high computational cost. Therefore, we follow [24, 46, 18] to perform contrast in a *pixel-segment* manner. Formally, supposing an image I is partitioned into segments S by a given prior. Let $\boldsymbol{f}_i$ be the unit-length feature vector of pixel i, we define the (normalized) prototype $\boldsymbol{p}$ of a segment $s \in S$ by the average of feature vectors within this segment.

$$\boldsymbol{p}_s = \frac{1}{|s|}\sum_{i \in s} \boldsymbol{f}_i, \quad \|\boldsymbol{p}_s\| = 1 \tag{1}$$

We adopt the InfoNCE [38] loss and formalize our loss as

$$\mathcal{L}(\boldsymbol{f}_i, S) = -\log \frac{\exp(\boldsymbol{f}_i \cdot \boldsymbol{p}_{s_+}/\tau)}{\exp(\boldsymbol{f}_i \cdot \boldsymbol{p}_{s_+}/\tau) + \sum_{s_-} \exp(\boldsymbol{f}_i \cdot \boldsymbol{p}_{s_-}/\tau)} \tag{2}$$

where τ is a temperature hyper-parameter. Ideally, it pulls $\boldsymbol{f}_i$ close to the positive segment s_+ and pushes it away from

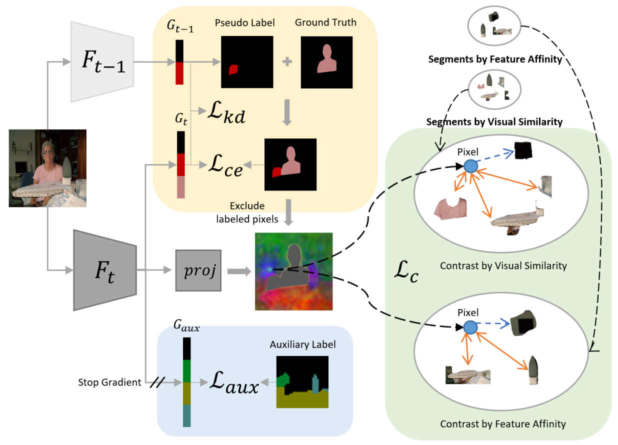

Figure 4. **Overall framework.** Given an image at step t, a projection head maps the features from feature extractor F_t into an embedding space where $\mathcal{L}_c$ is used to optimize future classes (green block). Next, the features from F_t are also used to train an auxiliary classifier G_{aux} to initialize the classifier for future classes (blue block). Last, the pseudo label technique together with a knowledge distillation loss are adopted to mitigate forgetting (yellow block).

all the negative segments s_-. For a given pixel, the segment it belongs to serves as s_+, whereas other segments, including those from other images, serve as s_-.

Contrast by Visual Similarity. In pixel-segment contrastive learning, the partition scheme determines how the network is optimized. We first resort to visual boundaries as a prior to partition the images into visual coherent segments S_v by OWT-UCM [2], which is a decent choice in unsupervised segmentation [19, 55, 23]. By utilizing visual similarity as supervision, we can bootstrap the representation learning and ground the features to respect low-level visual cues. But using this supervision alone makes the network hard to capture semantic-level information of future classes, and still needs further updates to properly learn them. This contradicts our goal to make the necessary updates as few as possible in the future.

Contrast by Feature Affinity. We also need a partition method that can match actual future classes accurately to serve as another supervision. Due to the bootstrap effect of the previous contrast, we can resort to clustering methods to explore feature affinity as semantic-level supervision without relying on architectural prior like [8]. However, applying clustering directly cannot generate satisfying results. In practice, we find that by hierarchically combining an over-clustering result, we could obtain much more accurate results. More in detail, starting from an over-clustering generated using intermediate features, we merge the two most similar segments each time till all segments are merged as one. The similarity between segments is measured by the cosine similarity between their prototypes. This procedure

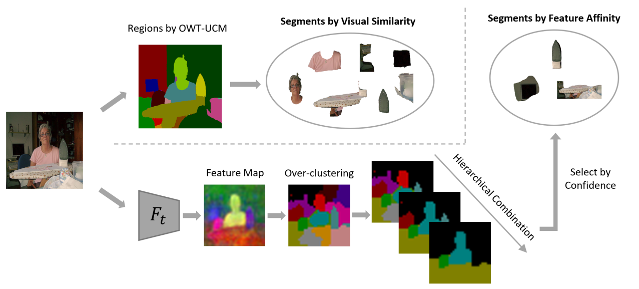

Figure 5. Segments obtained using visual similarity and feature affinity respectively. **Top:** Segments generated by OWT-UCM [2] using boundaries. **Bottom:** Segments that are likely to match the actual classes are selected from a hierarchical combination of fine-grained clustering results.

generates consistent segments across all granularities. But different classes naturally have different levels of granularities, directly selecting segments from a single granularity is often suboptimal. We further propose a confidence score to evaluate how likely a segment matches a class accurately to select segments across granularities. The design of the confidence score is based on the following observations. In the combination procedure where the most similar segments are merged each time, the longer a segment can remain not merged, the more different it is to all other segments, and is more likely to match a class. What's more, given that this procedure starts from an over-clustering, the segments in later steps have a higher chance to match a class and should be assigned higher weights. A possible choice is to use the cosine distance between the segments to be merged at each step as a weight factor, which monotonously increases by step. Formally, assuming segments a and b are merged into a new segment c at step n in this procedure, set S_n indicates all segments that are merged from start till now (include a and b). The confidence score u for a segment s at step n is defined recursively as follows.

$$u_s^n = \begin{cases} 0 & \text{if } s = c, \\ u_s^{n-1} & \text{elif } s \in S_n, \\ u_s^{n-1} + (1 - \boldsymbol{p}_a \cdot \boldsymbol{p}_b) & \text{otherwise.} \end{cases} \tag{3}$$

After finishing this procedure, we use the final score of each segment to select those with scores above the threshold to serve as high-confidence segments S_f and participate in contrastive learning. If two selected segments are partially overlapped, we only reserve the one with a higher score. This procedure is performed for each training sample before each epoch to select high-confidence segments as the network updates itself, except for the first epoch since the feature cannot be clustered at that moment.

Overall Contrastive Loss. In our contrastive learning, the pixels of $\mathcal{C}_t$ with ground truth annotations and $\mathcal{C}_{1:t-1}$ with pseudo labels are excluded. Meanwhile, they can naturally serve as high-confidence negative segments in contrastive learning. Since the batch size is limited in segmentation while the number of negative samples is crucial in contrastive learning [7, 15], we use a memory bank to cache the features of recent four batches to increase the number of negative samples. Given the segments S_v generate by visual similarity and segments S_f generated by feature affinity, our contrastive loss is formalized as

$$\mathcal{L}_c = \frac{1}{|S_v|} \sum_{\substack{i \in S_v \\ i \notin \mathcal{C}_{1:t}}} \mathcal{L}(\boldsymbol{f}_i, S_v) + \frac{1}{|S_f|} \sum_{\substack{i \in S_f \\ i \notin \mathcal{C}_{1:t}}} \mathcal{L}(\boldsymbol{f}_i, S_f) \quad (4)$$

where $|S|$ means the pixel number in all segments within S.

4.2. Learning Future Knowledge in Output Space

Previous work [5] has claimed that a proper initialization of classifier weight at step $t > 1$ could also benefit the performance and is widely adopted by other works [10, 54]. This motivates us to further introduce future knowledge to the output space. Existing practice eases the misalignment between features and classifier weights by using the weight of the background to initialize that of the new classes. But in the update process, the weights of new classes should be updated to separate them from the background. Making the classifier weights to be identical to that of the background has the risk of increasing the necessary update, which has a negative effect based on previous analysis.

To address this issue, we want the classifier to be initialized as close to the final optimal value as possible. When optimizing feature space, we manage to obtain high-confidence segments of future classes, and the intermediate features are also optimized accordingly. We could naturally utilize them to train an auxiliary classifier that can classify future classes in advance and use its weights as initialization. More in detail, we first perform clustering on the prototypes of segments to assign an auxiliary semantic label to each segment, which is then used to optimize the auxiliary classifier using cross-entropy loss. Note that the network receives no gradient from this classifier. In the next step when initialization is needed, we compute the prototype of each new class and select the auxiliary classifier weight which is closest to the prototype to initialize the actual classifier weight of it. On the one hand, we can make the classifier converge more easily and avoid the R-P dilemma. On the other hand, it solves the inadequate data obstacle in output space by utilizing the data of current step to serve as the negative samples of future classes, which can further boost the discriminability of the classifier compared to training it directly on the future steps.

4.3. Overall Framework

Our proposed framework is depicted in Figure 4. $\mathcal{L}_{ce}$ and $\mathcal{L}_{aux}$ are both cross-entropy loss to learn new classes and to train the auxiliary classifier, respectively. To complete our framework, we still need a KD loss to overcome forgetting. Since our focus is not on this part, we adopt the CKD loss in MiB [5] as $\mathcal{L}_{kd}$ to assist our training, with the addition of a temperature parameter as in original KD loss [16] to enhance its ability in knowledge preserving. This makes the overall loss function of our framework as follows.

$$\mathcal{L} = \mathcal{L}_{ce} + \alpha\mathcal{L}_{kd} + \beta\mathcal{L}_c + \mathcal{L}_{aux}, \quad (5)$$

where α and β are hyper-parameters.

5. Experiments

5.1. Implement Details

Datasets. Following previous works, we evaluate our method on VOC2012 [12] and ADE20K [56], which are the standard CSS benchmarks. We report the performance on the official validation set of each dataset. We also exclude 20% of the training set as a validation set to tune hyper-parameters and perform all analyses.

Protocols. There are two settings in CSS benchmarks: disjoint and overlapped. The disjoint setting limits the given images to contain pixels only belong to $\mathcal{C}_{1:t}$, while the latter does not have this restriction, which means the pixels can also belong to the future classes, *i.e.*, $\mathcal{C}_{t+1:T}$. Note that the overlapped setting is considered to be more realistic and more challenging by all existing works [5, 10, 6, 37], and the disjoint setting is even ignored in some works [6, 37]. In design, our work is fully based on the overlapped setting.

Evaluation. The CSS benchmark setups are denoted as $N_o - N_n$ where N_o and N_n are the number of classes to be learned at the initial step and each incremental step, respectively. For example, the 15-1 setup of VOC2012 gives 15 classes at the initial step and adds one class per incremental step, resulting in a total of 6 steps. On VOC2012, we conduct experiments on four setups: 19-1 (2 steps), 15-5 (2 steps), 15-1 (6 steps), and 10-1 (11 steps). On ADE20K, we use three setups: 100-50 (2 tasks), 50-50 (3 tasks), and 100-10 (6 tasks). The network is evaluated after finishing all learning steps using mean Intersection over Union (mIoU) as metric. The performance of initial classes $\mathcal{C}_1$, incremental classes $\mathcal{C}_{2:T}$, and all classes $\mathcal{C}_{1:T}$ are denoted as old, new, and all, respectively.

Training. Following previous works [5, 10], we adopt DeepLab-V3 with an output stride of 16. ResNet-101 serves as the backbone, which is pre-trained on ImageNet. We use the SGD optimizer with momentum set to 0.9. The learning rate is set to 1e-2 for the initial learning step and 1e-3 for the following learning step. The learning rate is adjusted by the polynomial decay schedule with power set to 0.9. The network is trained on VOC2012 for 30 epochs, and 60 epochs on ADE20K, both with a batch size of 16. The image is cropped to 512×512 for both training and validation, with

Table 1. Results of different methods on VOC2012. The best is in **bold** while the second best is in underline. Results are averaged over 3 runs with standard deviation provided.

Method	19-1 (2 tasks)			15-5 (2 tasks)			15-1 (6 tasks)			10-1 (11 tasks)		
	old	new	all	old	new	all	old	new	all	old	new	all
ILT [35]	67.1	12.3	64.4	66.3	40.6	59.9	4.9	7.8	5.7	-	-	-
MiB [5]	70.2	22.1	67.8	75.5	49.4	69.0	35.1	13.5	29.7	12.2	13.0	12.6
PLOP [10]	75.3	37.3	73.5	75.7	51.7	70.0	65.1	21.1	54.6	44.0	15.5	22.9
SDR [36]	69.1	32.6	67.4	75.4	52.6	69.9	44.7	21.8	39.2	-	-	-
SSUL [6]	**77.7**	29.6	75.4	77.8	50.1	71.2	**77.3**	36.5	67.6	**71.3**	45.9	59.2
RCIL [54]	-	-	-	78.8	52.0	72.4	70.6	23.7	59.4	55.4	15.1	34.3
ALIFE [37]	76.6	49.3	75.3	77.1	52.5	71.3	64.4	34.9	57.4	-	-	-
Ours	**77.7** ±0.22	**59.8** ±1.38	**76.8** ±0.35	**79.1** ±0.21	**67.7** ±0.46	**76.4** ±0.29	75.2 ±0.75	**52.2** ±1.33	**69.7** ±1.09	68.1 ±0.96	**54.5** ±1.57	**61.6** ±1.13
RECALL [34]	68.1	55.3	68.6	67.7	54.3	65.6	67.8	50.9	64.8	65.0	53.7	60.7
SSUL-M [6]	77.8	49.7	76.4	78.4	55.8	73.0	**78.3**	49.0	71.3	**74.0**	53.2	64.1
ALIFE-M [37]	76.7	52.2	75.5	77.6	55.2	72.3	66.0	38.8	59.5	-	-	-
Ours-M	**77.9** ±0.19	**67.7** ±0.74	**77.4** ±0.32	**79.3** ±0.22	**70.2** ±0.29	**77.1** ±0.24	77.1 ±0.65	**60.4** ±0.73	**73.1** ±0.68	69.5 ±0.60	**63.2** ±1.25	**66.5** ±0.78
Joint	77.6	77.7	77.6	79.5	71.5	77.6	79.5	71.5	77.6	78.5	76.6	77.6

Table 2. Results of different methods on ADE20K. The best is in **bold** while the second best is in underline. Results are averaged over 3 runs with standard deviation provided.

Method	100-50 (2 tasks)			50-50 (3 tasks)			100-10 (6 tasks).		
	old	new	all	old	new	all	old	new	all
ILT [35]	18.2	14.4	17.0	3.5	12.8	9.7	0.1	1.3	0.5
MiB [5]	40.5	17.1	32.7	45.5	21.0	29.3	38.2	11.1	29.2
PLOP [10]	41.8	14.8	32.9	48.8	20.9	30.4	40.4	13.6	31.5
SSUL [6]	41.2	18.0	33.5	48.3	20.1	29.5	40.2	18.7	33.1
RCIL [54]	42.3	18.8	34.5	48.3	25.0	32.9	39.3	17.6	32.1
ALIFE [37]	42.1	23.0	35.8	48.9	25.6	33.5	41.0	22.7	34.9
Ours	**43.1** ±0.17	**26.0** ±0.36	**37.4** ±0.25	**49.1** ±0.24	**28.3** ±0.43	**35.2** ±0.33	**41.4** ±0.28	**25.5** ±0.45	**36.1** ±0.35
SSUL-M [6]	42.7	17.5	34.3	49.1	20.1	29.7	**42.8**	17.6	34.4
ALIFE-M [37]	42.2	23.5	36.0	49.0	26.1	33.8	41.1	23.0	35.1
Ours-M	**43.1** ±0.15	**26.2** ±0.29	**37.5** ±0.21	**49.2** ±0.12	**28.5** ±0.27	**35.4** ±0.18	41.6 ±0.21	**25.8** ±0.26	**36.3** ±0.22
Joint	43.5	29.4	38.8	50.3	32.7	38.8	43.5	29.4	38.8

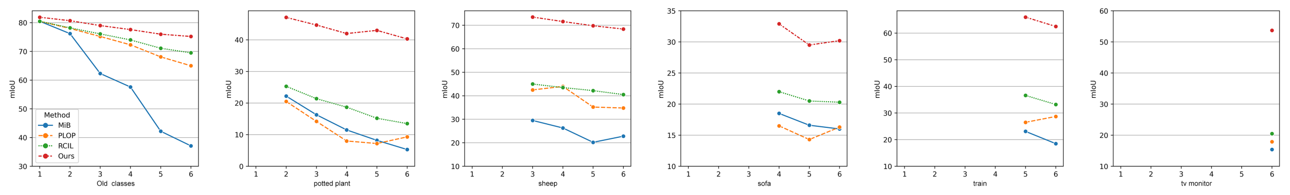

Figure 6. mIoU comparison between CSS methods over steps on VOC 15-1 setups. The performance of initial classes and each new class added per step are plotted separately from left to right. Our method shows great ability in learning new classes.

a random scaling and a random left-right flip additionally applied to training.

5.2. Results

We provide the results on VOC2012 and ADE20K in Table 1 and Table 2, respectively. Some works [6, 34] exploit memory or external data to assist training. We also report the results obtained using memory (denoted by -M) following the implementation of [6] to fairly compare with these methods. The result of joint training is also provided as the upperbound for reference.

For all setups on VOC2012, our method surpasses all existing methods by a large margin on new classes. This demonstrates the strong ability of our method to learn new classes. Note that our method also shows promising results on old classes even with a simple approach to preserve old knowledge. This is due to that a higher performance on new classes indicates that there also might be less confusion between old and new classes, which also benefits the mIoU of old classes. On the more challenging setups with multiple steps (15-1, 10-1), the new classes cannot be properly learned in the first place for most methods. To make matters worse, these poorly optimized new classes will suffer forgetting in the following steps which further reduces their performance. Even in these challenging circumstances, our method still shows great performance in learning new classes, as shown in Figure 6. We can also see that even though SSUL shows higher performance on old classes on long-term setups, it relies on completely freezing the feature extractor to prevent forgetting, which does help a lot for old classes on these steps, but also harms its flexibility greatly. We believe that if equipped with a modern knowledge-preserving constraint, our method could offer another boost in performance.

On the more challenging ADE20K benchmarks, we obtain similar results as that of VOC2012. Our method shows some improvement on old classes and is significantly better on new classes. On top of the results of VOC2012 benchmark, this further demonstrates that our approach works well on more complex datasets, and has the ability to handle both scenes and objects centric datasets.

When memory is applied on both benchmarks, our method receives further improvements and holds the leading position. This also indicates that our proposed training method has a non-overlapping effect with the memory technique. They can be used as two complementary techniques as pointed out in Section 3.4

5.3. Ablation Study

We investigate the effects of the proposed components in our framework using both 15-1 and 15-5 setups on VOC2012. Starting from a baseline composed of L_{ce}, L_{kd} and pseudo label, we add the components one by one and show the ablation results in Table 3. As can be seen, the performance increase steadily as each proposed component is gradually added. Only applying contrast by visual similarity does not receive much improvement since it cannot instruct the network to capture semantic-level information. After applying contrast by feature affinity, the performance receives the highest gain, indicating the necessity to add semantic-level supervision and make the network learn more accurate knowledge. On top of that, the performance receives another boost from our classifier initialization technique which introduces future knowledge to output space.

Table 3. Ablation study on VOC2012 15-1 and 15-5 setup.

Method	15-1 (6 tasks) old	new	all	15-5 (2 tasks) old	new	all
baseline	65.3	18.8	54.2	76.9	51.3	70.8
+ visual similarity contrast	67.1	26.3	57.4	77.2	54.5	71.8
+ feature affinity contrast	74.5	45.1	67.5	78.7	65.4	75.5
+ classifier initialization	75.2	52.2	69.7	79.1	67.7	76.4

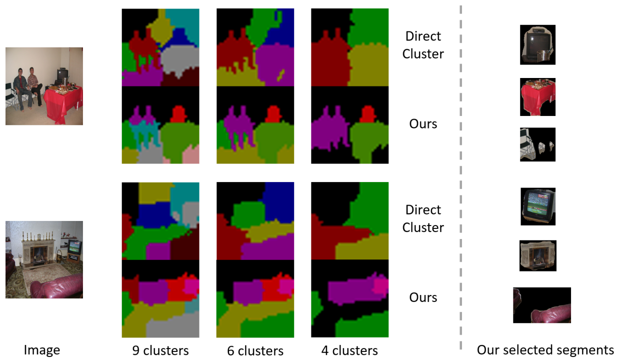

Figure 7. Visualization of clustering results of direct clustering and our practice. The high-confidence segments selected by our criterion are also presented on the right.

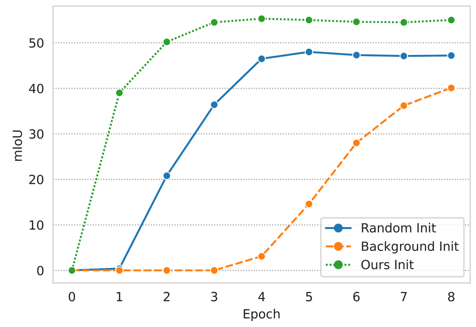

Figure 8. Performance comparison between different classifier initialization over training epochs on VOC2012 19-1 setup.

5.4. Further Discussion

Feature Quality Analysis. To further demonstrate the effect of our method to solve the underfitting problem pointed out in Section 3.2, we conduct the same analysis on our method and incorporate the findings into Figure 2 (denoted by Ours). As can be seen, pre-learning future classes can significantly improve both intra-class and inter-class fitness.
Segment Generation. We provide a qualitative comparison between direct clustering and our proposed hierarchical combination at different granularities to validate our design in Section 4.1. The results are generated from the same intermediate features and are plotted in Figure 7. As can be seen, our practice consistently generates better results across different granularities. We also present the high-confidence segments selected by our criterion for each image, which match the actual classes with high precision.
Classifier Initialization. We give a comparison of performance over training epochs for different classifier initialization methods in Figure 8. As can be seen, the initialization technique proposed by MiB [5] converges much slower due to larger update paces. Random initialization converges fast but the performance stops increasing shortly afterward. Among all of them, our method stands out with its fast convergence speed and superior performance.

6. Limitation and Conclusion

Our method has two limitations. Firstly, it relies on the natural attribute of CSS that unlabeled future classes exist in current training data. So it may not perform well when there are only a few unlabeled future classes in the current training data. However, in such cases, we can still leverage external unlabeled data, such as web-crawled data as used in RECALL [34]. Secondly, we point out that even though our proposed framework achieves great performance, it as a practicable solution might not be able to unleash the full potential of the training method, which requires further exploration in future works.

In this work, we present an in-depth analysis of the reason why existing methods suffer low performance on new classes. Building on this analysis, we introduce a novel training pipeline for CSS that optimizes potential future classes in the current training step. This offers a new perspective for enhancing new class performance in contrast to existing works that focus on addressing the R-P dilemma. We propose a framework based on this pipeline and it shows a significant improvement for new classes, while also benefiting the performance of old classes. We believe that this work can provide valuable insights for the research community on CSS.

7. Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant 62176246 and Grant 61836008. This work is also supported by Anhui Province Key Research and Development Plan (202304a05020045), Anhui Provincial Natural Science Foundation 2208085UD17 and the Fundamental Research Funds for the Central Universities (WK3490000006).

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[3] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, 2021.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[5] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *ICCV*, 2020.

[6] Sungmin Cha, YoungJoon Yoo, Taesup Moon, et al. Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. *NIPS*, 2021.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[8] Jang Hyun Cho, Utkarsh Mall, Kavita Bala, and Bharath Hariharan. Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *CVPR*, 2021.

[9] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *CVPR*, 2019.

[10] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *CVPR*, 2021.

[11] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, 2020.

[12] Mark Everingham and John Winn. The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011.

[13] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *ECCV*, 2018.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *ICCV*, 2019.

[18] Hanzhe Hu, Jinshi Cui, and Liwei Wang. Region-aware contrastive learning for semantic segmentation. In *ICCV*, 2021.

[19] Jyh-Jing Hwang, Stella X Yu, Jianbo Shi, Maxwell D Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen. Segsort: Segmentation by discriminative sorting of segments. In *ICCV*, 2019.

[20] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *ECCV*, 2020.

[21] Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

[22] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*, 2022.

[23] Tsung-Wei Ke, Jyh-Jing Hwang, Yunhui Guo, Xudong Wang, and Stella X Yu. Unsupervised hierarchical semantic segmentation with multiview cosegmentation and clustering transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2571–2581, 2022.

[24] Tsung-Wei Ke, Jyh-Jing Hwang, and Stella Yu. Universal weakly supervised segmentation by pixel-to-segment contrastive learning. In *ICLR*, 2021.

[25] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[27] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, 2019.

[28] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[29] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *NIPS*, 2021.

[30] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[32] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

[33] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[34] Andrea Maracani, Umberto Michieli, Marco Toldo, and Pietro Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. In *ICCV*, 2021.

[35] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *ICCVW*, 2019.

[36] Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In *CVPR*, 2021.

[37] Youngmin Oh, Donghyeon Baek, and Bumsub Ham. Alife: Adaptive logit regularizer and feature replay for incremental semantic segmentation. *NIPS*, 2022.

[38] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[39] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, 2019.

[40] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *NIPS*, 2021.

[41] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

[42] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[43] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip HS Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: an initial phase decorrelation approach for class incremental learning. In *CVPR*, 2022.

[44] Pravendra Singh, Pratik Mazumder, Piyush Rai, and Vinay P Namboodiri. Rectification-based knowledge retention for continual learning. In *CVPR*, 2021.

[45] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 254–270. Springer, 2020.

[46] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Unsupervised semantic segmentation by contrasting object mask proposals. In *ICCV*, 2021.

[47] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 2022.

[48] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

[49] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *ICCV*, 2019.

[50] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, 2021.

[51] Guanglei Yang, Enrico Fini, Dan Xu, Paolo Rota, Mingli Ding, Moin Nabi, Xavier Alameda-Pineda, and Elisa Ricci. Uncertainty-aware contrastive distillation for incremental semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[52] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[53] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017.

[54] Chang-Bin Zhang, Jia-Wen Xiao, Xialei Liu, Ying-Cong Chen, and Ming-Ming Cheng. Representation compensation networks for continual semantic segmentation. In *CVPR*, 2022.

[55] Xiao Zhang and Michael Maire. Self-supervised visual representation learning from hierarchical grouping. *NIPS*, 2020.

[56] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[57] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, 2021.

[58] Yixiong Zou, Shanghang Zhang, Ke Chen, Yonghong Tian, Yaowei Wang, and José MF Moura. Compositional few-shot recognition with primitive discovery and enhancing. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 156–164, 2020.