Continual Semantic Segmentation via Structure Preserving and Projected Feature Alignment

Zihan Lin, Zilei Wang✉, and Yixin Zhang

University of Science and Technology of China
{myustc, zhyx12}@mail.ustc.edu.cn, zlwang@ustc.edu.cn

Abstract. Deep networks have been shown to suffer from catastrophic forgetting. In this work, we try to alleviate this phenomenon in the field of continual semantic segmentation (CSS). We observe that two main problems lie in existing arts. First, attention is only paid to designing constraints for encoder (*i.e.*, the backbone of segmentation network) or output probabilities. But we find that forgetting also happens in the decoder head and harms the performance greatly. Second, old and new knowledge are entangled in intermediate features when learning new categories, making existing practices hard to balance between plasticity and stability. On these bases, we propose a framework driven by two novel constraints to address the aforementioned problems. First, a structure preserving loss is applied to the decoder's output to maintain the discriminative power of old classes from two different granularities in embedding space. Second, a feature projection module is adopted to disentangle the process of preserving old knowledge from learning new classes. Extensive evaluations on VOC2012 and ADE20K datasets show the effectiveness of our approach, which significantly outperforms existing state-of-the-art CSS methods.

Keywords: incremental learning, semantic segmentation

1 Introduction

Semantic segmentation aims to assign every pixel a semantic category for a given image, which is a fundamental but challenging computer vision task. In recent years, the state-of-the-art methods [7,47,48] based on Fully Convolutional Network (FCN) [30] have achieved great success on large-scale benchmarks [15,49,9]. These models are designed to be trained in a one-shot mode with all data prepared in advance. When fitted in real-world applications, they will inevitably encounter situations where new categories need to be gradually learned. A naive way is to fine-tune the models on new data. But they often fail to preserve the performance of learned classes when updating themselves, which is called catastrophic forgetting [16] and becomes a main obstacle in practical applications. Incremental learning, which enables models to continuously learn new knowledge like human beings, is considered to be a promising solution to this challenge.

Class incremental learning (CIL) has been widely studied in image classification [29,39,26,22,14], but it only received attention very recently in semantic

segmentation. Extending methods from classification, current CSS approaches have managed to make some progress, but two problems still remain unsolved. One of them is that existing practices add constraints on either output probabilities [3] or encoder [13,37] to prevent forgetting. But they fail to consider the decoder head, which is a component exclusively in segmentation model. We argue that forgetting phenomenon also exists in decoder and has a significant impact on performance (details in Section 5.5). On this demand, we aim to design a constraint specially for decoder to mitigate the forgetting. Another problem lies in the knowledge distillation (KD) [20] used by existing methods [14,13,12] to prevent changes in intermediate features (*i.e.*, the output of encoder). Fixing the model activations is reasonable in classification since the ground truth of a certain image is consistent across all learning steps, but it's not the case in CSS. The semantic information of a given image might gradually increase, which is known as the background shift [3]. This requires the intermediate features to be updated accordingly to integrate the knowledge of upcoming classes, making the aforementioned practices fail in this situation. Simply mimicking the intermediate features of the old model would cause a conflict, where the model tries to be persistent on the old activations while struggling to learn the new knowledge, and ultimately results in degradation of performance.

In response to these problems, we propose a novel framework called SPPA, standing for **S**tructure **P**reserving and **P**rojected feature **A**lignment. It consists of two main components to tackle the forgetting of encoder and decoder, respectively. For encoder, we adopt a feature projection module to extract from intermediate features some low-dimensional representations of old knowledge, on which alignment is performed. This practice disentangles the preservation of old knowledge and the integration of new knowledge, which enables us to maintain the integrity of old knowledge without hindering the update process of new knowledge. For decoder, inter-class and intra-class structures are first modeled from its output, which reflect both coarse and fine-grained relations in embedding space. The structures are then explicitly maintained during incremental learning, which can effectively preserve the discriminability of old classes while being more flexible for new classes. The extensive experiments on Pascal-VOC 2012 [15] and ADE20K [49] demonstrate the effectiveness of our method. The contributions of this paper are summarized as

- We unveil the forgetting phenomena of decoder head in CSS, and propose to mitigate this by maintaining the class-related structures in embedding space at two different granularities.
- We manage to solve the potential conflict of feature KD in CSS by disentangle the preservation of old knowledge and the integration of new knowledge. Consequently provides a better trade-off between stability and plasticity.
- We integrate both modules into a unified framework, which is evaluated on two popular datasets with diverse experimental settings. The results reveal that our method outperforms previous state-of-the-art methods.

2 Related Works

Continual Learning. Deep neural networks have achieved great success in many fields, such as image classification [19], semantic segmentation [7] and object detection [40]. However, they often suffer from catastrophic forgetting [16] when situated in real scenarios where continuous streams of new data are involved. Continual learning that aims at tackling this obstacle has become an active field recently.

Continual learning for image classification has been extensively studied these years. Current techniques can be mainly divided into rehearsal methods, architectural methods and regularization methods. Rehearsal methods store a limited amount of raw images [39,2,22] and interleave them with new data to relieve forgetting. Some methods store intermediate features [25,18,10] instead, as they require less storage and contain richer information. Generation-based methods are derived from them that resort to generative networks to obtain images [42,8] or features [43] of old classes. Architectural methods either use a sub-network [33,32] to solve each independent task at a cost of limited scalability, or dynamically expand the network [44,28,45] for learning new tasks with growing complexity. Most of these methods require additional task labels and are restricted to multi-head [34] setup. Regularization methods design extra loss functions to maintain the previous knowledge, which can be divided into parameter regularization [46,26] and distillation-based methods. Parameter regularization prevents updates on the most important parameters for old tasks based on various metrics like fisher information matrix [26] or gradient magnitude [46]. Distillation-based methods are the most common currently. It was first introduced by LwF [29] to penalize the changes in output logits. LwM [12] additionally performs distillation on attention maps. [22] introduces a stronger constraint by punishing changes in feature vectors. Distillation on intermediate features is used in [14] to further reduce forgetting.

Continual Semantic Segmentation. Modern deep neural networks for semantic segmentation are mostly based on the fully convolutional network (FCN) [30]. Recent developments mainly try to exploit spatial information to improve accuracy. For example, encoder-decoder architecture [41,1] is used to prevent spatial information loss. Atrous convolution [35] enlarges the field of view to incorporate more spatial context information. Multi-scale information is further considered in ASPP [5] and PSPNet [47]. More recently, attention mechanisms have been used to model spatial dependences [48,17,24].

Despite their great success, segmentation networks inevitably suffer from catastrophic forgetting when used in online scenarios. [36] is the first one targeting for CSS. But it requires the labels of both old and current classes be provided, which greatly limits its capability in real scenarios. [3] formalized background shift, which is a new obstacle specially for CSS. To overcome this, they modified traditional distillation loss and cross-entropy loss to an unbiased version considering the semantic inconsistency of background. More recently, some methods

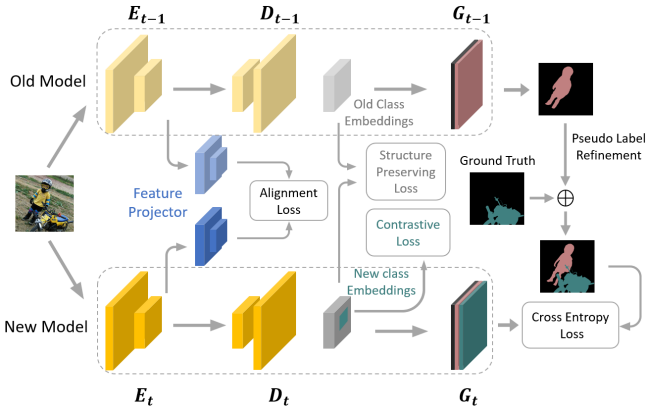

Fig. 1. Overview of the proposed framework. Alignment is performed on the representations generated by the feature projector to solve the potential conflict in feature distillation for encoder. The structure preserving loss is used to maintain discriminative structures for old classes, while the contrastive loss is used to optimize the structures of new classes. Finally, pseudo label is used to provide missing annotations for old classes.

proposed to tackle forgetting in encoder. [37] designed a framework combining prototype distillation, feature sparcification and contrastive learning. However, a plasticity issue might arise when trying to punish any changes in prototypes. [13] proposed a multi-scale pooling distillation to preserve long and short range statistics in intermediate features. But it cannot solve the conflict of old and new knowledge stated before. What's more, all of the existing works never considered the forgetting of decoder. Our work is the first attempt to tackle forgetting in both encoder and decoder in a single framework, with two novel distillation approaches to achieve a better plasticity-stability trade-off than existing methods.

3 Problem Definition

In semantic segmentation, let $\mathcal{X} \in \mathbb{R}^{H\times W\times 3}$ be the input space, and $\mathcal{Y} \in \mathcal{C}^{H\times W}$ be the label space where $\mathcal{C}$ is a given category set. Given a dataset $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_n)\}$ where $(\mathbf{x}_n, \mathbf{y}_n) \subset \mathcal{X} \times \mathcal{Y}$. The goal is to produce a segmentation map $\hat{\mathbf{y}} \subset \mathcal{Y} \in \mathcal{C}^{H\times W}$ by assigning each pixel $x_i \in \mathbf{x}$ a class in $\mathcal{C}$. This is usually done using a deep neural network $M : \mathcal{X} \mapsto \mathbb{R}^{H\times W\times \mathcal{C}}$ and the segmentation result is calculated as $\hat{\mathbf{y}} = \arg\max_c M(\mathcal{X})[h, w, c]$. Nowadays M is usually an encoder-decoder architecture made by a feature extractor E and a segmentation head D (*i.e.*, $M = E \circ D$). In our work, we treat the final classifier as an individual module G , which makes $M = E \circ D \circ G$. We use $\mathbf{F}$ to represent the intermediate feature output by encoder $E(\mathbf{x})$, and $\mathbf{E}$ to represent the embedding output by decoder $D(\mathbf{F})$ before final classifier.

In conventional training pipeline, the complete training set $\mathcal{T}$ is available and the model is trained only once to learn all classes $\mathcal{C}$. While in continual

learning, the training procedure is composed of multiple learning steps. At each step, a subset of the training set is provided together with a set of novel classes to be learned. More specifically, at the initial learning step $t = 0$, a standard supervised training is performed on a subset of training data $\mathcal{T}^0$ with labels of $\mathcal{C}^0$. Next moving to a more general step t, a set of novel classes $\mathcal{C}^t \in \mathcal{C}$ is introduced, expanding the learned label set to $\mathcal{C}^{0:t} = \mathcal{C}^{0:t-1} \cup \mathcal{C}^t$. A training set $\mathcal{T}^t$ with all $\mathcal{C}^{0:t-1}$ labeled as background and only $\mathcal{C}^t$ been labeled is provided to update the previous model $M_{t-1} : \mathcal{X} \mapsto \mathbb{R}^{H\times W\times \mathcal{C}^{0:t-1}}$ to $M_t : \mathcal{X} \mapsto \mathbb{R}^{H\times W\times \mathcal{C}^{0:t}}$. As in standard class incremental learning, we assume that the classes introduced in different steps are disjoint ($\mathcal{C}^i \cap \mathcal{C}^j = \emptyset$) except for the background class b.

4 Method

In this section, we will introduce the proposed modules one by one, following the structure depicted in Fig. 1 from left to right to make it easier to follow.

4.1 Projected Feature Alignment for Encoder

We first go through the feature distillation in existing works. [14,12,22] have attempted to perform distillation on intermediate features, which consists of a distance metric to minimize the differences between feature maps from old and new models. Due to the rich information contained in intermediate features, this approach often yields better performance than logits distillation. An Euclidean distance is usually applied, and the distillation loss can be formulated as

$$L = \|E_t(\mathbf{x}) - E_{t-1}(\mathbf{x})\|_2, \tag{1}$$

where $E_t(\mathbf{x})$, $E_{t-1}(\mathbf{x})$ indicate the feature of image $\mathbf{x}$ extracted by the encoder of step t and step $t-1$, respectively.

It's reasonable to keep the activations invariant in classification since the semantic information of an image will not change across all incremental learning steps. In CSS however, the semantic information might increase (*e.g.*, we want to learn a new class which was ignored and labeled as background before). This is known as the background shift phenomenon[3] exclusively in CSS. Ideally, the intermediate features should be updated to integrate new knowledge. It means that even if the current model does not forget the old knowledge, its intermediate features could still be quite different from the ones extracted by the old model. If we make the current model directly mimic the old one, a conflict arises between being unchanged in the activations and adapting to new classes at the same time.

Therefore, we try to disentangle the processes of retaining old knowledge and learning new classes for intermediate features. It is known that an auto-encoder can generate compact representations that contain sufficient information to reconstruct the original feature maps [21]. We leverage this property to extract low-dimensional representations of old knowledge from intermediate features.

These representations are then used for alignment. It's worth noting that we do not reduce the spatial resolution like traditional auto-encoder since the segmentation task requires high spatial precision. We define the auto-encoder architecture as a projector P and a reconstructor R to avoid being confused with the encoder and the decoder in segmentation model. Before starting learning step t, we use the intermediate features of $\mathcal{T}^t$ extracted by E_{t-1} to train a projection module P_{t-1} from scratch using the reconstruction loss L_{recon}.

$$L_{recon} = \|E_{t-1}(\mathbf{x}) - R_{t-1}(P_{t-1}(E_{t-1}(\mathbf{x})))\|_2 \tag{2}$$

After it converges, the reconstructor R_{t-1} is discarded, the old model M_{t-1} and the trained P_{t-1} is used to initialize the current model (*i.e.*, M_t and P_t). During the training of current step t, M_{t-1} and P_{t-1} are fixed, while P_t and M_t are updated. L1 distance is minimized between the output of P_{t-1} and P_t.

$$L_{ali} = \|P_{t-1}(E_{t-1}(\mathbf{x})) - P_t(E_t(\mathbf{x}))\|_1 \tag{3}$$

The design intuition is as follows: The output representations of P_{t-1} contain sufficient information to restore the old features $E_{t-1}(\mathbf{x})$ which stand for the old knowledge. If we can extract the same representations from the new feature maps $E_t(\mathbf{x})$ by P_t, it indicates that the new feature maps still hold the same old knowledge as that of the old. With this design, we make it possible to add constraints on the old knowledge without the potential impediment on learning new knowledge.

4.2 Class Structure Preserving for Decoder

Existing approaches utilizing pre-classifier embeddings commonly fall into adding constraints for feature vectors [22] or class prototypes [37,4]. However, when applied in CSS to prevent forgetting in decoder head, we find that these methods yield overly strong constraints on the output, which penalizes any possible changes with respect to its previous position in the embedding space. On the one hand, it might impede the model from finding a globally optimal position that benefits both old and new classes, thus hindering the learning of new classes. On the other hand, the model tries to be persistent in the absolute positions of old classes while integrating new classes. This might cause an optimization conflict, which results in uncontrollable small drifts of old classes in embedding space and ultimately leads to a chaotic distribution, as shown in Fig. 6.

Since it is undesirable to directly fix the embeddings of old classes, we turn to seeking a practice that can preserve the performance of old classes without sacrificing the freedom to learn new classes. It is known that the embeddings from a well-trained model form a space, in which the instances of the same class are close to each other and far from those of different class. This geometry in embedding space is the key to making each class linearly separable. Inspired by this, we propose to model the structures in embedding space and explicitly maintain them during incremental learning steps. In this work, we consider two

structures of different granularity: the structure between different classes (inter-class structure) and the structure within a single class (intra-class structure).

Class prototypes (*i.e.*, mean of feature vectors of a class) are needed when modeling these two structures. Generally, they can be computed in a global, in-batch or in-image manner, which represents the class information from coarse-grained (global-level) to fine-grained (instance-level). In segmentation, the embeddings of the same class vary across images and, in our observation, form image-level mini-clusters. Adopting in-image prototypes to obtain a fine-grained relation in our case helps to reflect these differences at an instance level and slightly boosts the performance. Similar practices are also observed in other designs for segmentation, like [23] uses in-image prototypes for contrastive learning. The in-image prototype $\mathbf{p}_c$ of class c for an input image is defined as

$$\mathbf{p}_c = \frac{1}{|\mathbf{y}^* = c|} \sum_{\substack{\mathbf{e}_i \in \mathbf{E} \\ i=c}} \mathbf{e}_i. \tag{4}$$

Since $\mathbf{E}$ has lower spatial resolution than input due to the network architecture, $\mathbf{y}$ should be resized to match the spatial dimension of $\mathbf{E}$ and is denoted as $\mathbf{y}^*$. $i = c$ means the position i is labeled as class c in $\mathbf{y}^*$. Because we don't have labels for old classes, the pseudo label $\hat{\mathbf{y}}^*$ output by the old model is used to obtain old class prototypes.

The inter-class structure is formulated as a distance matrix A between each in-image prototype of old classes obtained in the current batch, where A_{ij} is the cosine distance between prototypes i and j. The consistency constraint is then applied between A^t from M_t and A^{t-1} from M_{t-1}. $\bar{\mathbf{p}}_i$ is l2-normalized version of $\mathbf{p}_i$ and $\langle \bar{\mathbf{p}}_i, \bar{\mathbf{p}}_j \rangle$ calculates the cosine similarity between $\mathbf{p}_i$ and $\mathbf{p}_j$.

$$A_{ij} = 1 - \langle \bar{\mathbf{p}}_i, \bar{\mathbf{p}}_j \rangle, \quad i, j \in \mathcal{C}^{0:t-1} \tag{5}$$

$$L_{inter} = \|A^t - A^{t-1}\|_F \tag{6}$$

As for the intra-class structure, the embedding vectors of the same class are clustered closely with each other, so we leverage Euclidean distance to better reflect the small changes in the structure. Intuitively, the intra-class structure can be maintained by keeping the distance between embedding vectors and their prototypes. We further integrate the direction information to prevent the embedding vectors from rotating (See Fig. 2 right), making the constraints as fixing the *relative position* of embedding vectors with regard to their prototypes. The loss function for a class c is defined as

$$L_{intra}(c) = \frac{1}{|\hat{\mathbf{y}}^* = c|} \sum_{\substack{\mathbf{e}_i \in \mathbf{E} \\ i=c}} \|(\mathbf{e}_i^{t-1} - \mathbf{p}_c^{t-1}) - (\mathbf{e}_i^t - \mathbf{p}_c^t)\|_2, \tag{7}$$

in which $\mathbf{e}_i^{t-1}$, $\mathbf{p}_c^{t-1}$ is obtained from M_{t-1}. $\mathbf{e}_i^t$, $\mathbf{p}_c^t$ is obtained from M_t. ν is used to balance between inter-class term and intra-class term.

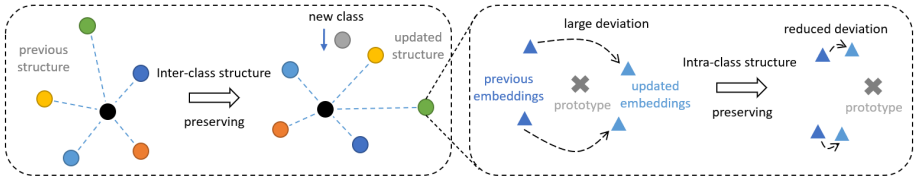

Fig. 2. Illustration of the structure preserving loss, which maintains inter-class structure and intra-class structure respectively.

$$L_{str} = L_{inter} + \nu \sum_{c \in \mathcal{C}^{0:t-1}} L_{intra}(c). \tag{8}$$

Compared to previous approaches, the structure preserving loss allows the model to freely re-arrange the feature vectors in the embedding space as long as the discriminative structures are intact during this procedure. By maintaining both coarse and fine-grained structures, it effectively mitigates the forgetting of old classes while at the same time avoiding the optimization dilemma, thus providing a better stability-plasticity trade-off.

The proposed structure preserving loss mainly focuses on old classes. We want to obtain good initial structures in the embedding space when learning new classes. Then the good structures can be explicitly maintained in future steps by the structure preserving loss. We adopt contrastive learning to achieve this. The losses are extended from [37,22], and we confirm that a modern contrastive loss with similar effects can also be used. L_{comp} encourages the embedding vectors to be close to their prototypes, while L_{mar} is to ensure enough distance between different classes. By doing this, a structure with compact distributions within classes and wide boundaries between classes can be obtained, which is less prone to catastrophic forgetting. The loss functions are formulated as follows, with μ a hyper-parameter and Δ a pre-defined distance.

$$L_{comp}(c) = \frac{1}{|\mathbf{y}^* = c|} \sum_{\substack{\mathbf{e}_i \in \mathbf{E} \\ i=c}} \|\mathbf{e}_i^t - \mathbf{p}_c^t\|_2, \tag{9}$$

$$L_{mar} = \sum_i \sum_j \max\{0, \Delta - \langle \bar{\mathbf{p}}_i, \bar{\mathbf{p}}_j \rangle\}, \quad i, j \in \mathcal{C}^t, \tag{10}$$

$$L_{cont} = L_{mar} + \mu \sum_{c \in \mathcal{C}^t} L_{comp}(c). \tag{11}$$

4.3 Pseudo Label for Old Classes

Training the model directly on given labels using cross-entropy loss will aggravate catastrophic forgetting because previous classes are labeled as background. A common practice is to adopt the pseudo label [27] technique to provide missing

annotations for old classes. At step t, we use the prediction of M_{t-1} on the current training set $\mathcal{T}^t$ to label the background regions. The pseudo labels are then refined by only accepting pixels with certainty above a threshold. Considering that the model cannot learn each class equally well. If the same threshold is applied to all classes, the poorly-learned classes might be overly rejected and unnecessarily sacrifice the integrity of pseudo label. It's better to let each class has its own threshold. We use entropy (denoted as u) as certainty measurement. An entropy threshold t_c is selected for each class to keep a fixed percentage of the raw pseudo label whose prediction entropy is below t_c.

$$y_i^{pseudo} = \begin{cases} y_i & \text{if } y_i \in \mathcal{C}^t, \\ \text{argmax}_{c \in \mathcal{C}^{0:t-1}} M_{t-1}(\mathbf{x}) & \text{elif } u < t_c, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

4.4 Loss Function

Combining the losses introduced above, the overall loss is formulated as follows, with α, β, γ being hyper-parameters:

$$L = L_{ce} + \alpha L_{ali} + \beta L_{str} + \gamma L_{cont}. \tag{13}$$

5 Experiments

We compare the performance of our method against four strong CSS methods, including ILT[36], MiB[3], SDR[37], and PLOP[13]. All methods in comparison (including ours) do not use the replay strategy [39], where a small amount of previous data is stored and rehearsed. The theoretical performance is obtained by training the model in an offline manner with all data and labels (given as Upper Bound).

5.1 Experiments Setup

Datasets and Metrics. We evaluated our framework on Pascal-VOC 2012 [15] with 20 foreground classes and a background class, and ADE20K [49] with 150 foreground classes. We use mean Intersection over Union (mIoU) as the performance metric. After the whole training process, the performance on initial classes $\mathcal{C}^0$ is denoted as *old*. On classes learned in incremental steps $\mathcal{C}^{1:T}$, it is denoted as *new*. And on all classes $\mathcal{C}^{0:T}$, it is denoted as *all*.

Incremental Protocols. Two different settings of CSS are described in [3] concerning different split methods for training set. **Disjoint setting:** at step t, the given images contain pixels only belonging to either old classes or current classes $\mathcal{C}^{0:t-1} \cup \mathcal{C}^t$. **Overlapped setting:** at step t, images that contain at least one pixel belonging to the current classes $\mathcal{C}^t$ are provided. In both settings, Only labels

for $\mathcal{C}^t$ is provided, and $\mathcal{C}^{0:t-1}$ are labeled as background. The main difference is that, the background in disjoint setting contains only seen classes $\mathcal{C}^{0:t-1}$. While in overlapped setting, it might contain old and future classes $\mathcal{C}^{0:t-1} \cup \mathcal{C}^{t+1:T}$. This is a more realistic setting for CSS because it is hard to guarantee that there will not be a demand to learn new concepts that were ignored previously. It is also more challenging due to severer semantic shifts.

Training Procedure. For fair comparison, we follow most of the settings in previous works [3,13]. We adopt the Deeplabv3 [6] with an output stride of 16. ResNet-101 [19] serves as the backbone and is pretrained on ImageNet [11]. The model is optimized using SGD with momentum, with a learning rate of $7e-3$ for the initial step and $7e-4$ for the following steps. The learning rate is decreased according to the polynomial decay rule of power 0.9. The model is trained with a batch size of 16 for 30 epochs on VOC2012 and 60 epochs on ADE20K. The input image is cropped to 512×512 for both training and validation. The data augmentation for training consists of a random scaling with a factor from 0.5 to 2.0 and a random left-right flip.

5.2 Evaluation on Pascal VOC2012

We present the results on Pascal-VOC 2012 in this section. Following [3,37,13], we use three experiment setups: adding one class for a step (19-1), adding five classes for a step (15-5), and adding one class sequentially for five steps (15-1). The results of disjoint and overlapped are reported in Table 1 and Table 2. The results of ours are averaged over three runs to increase statistical significance.

Single step addition of one class (19-1). In this setup, the model first learns 19 classes and then the last class is added. Directly fine-tuning the model always leads to the worst performance. ILT, which is the first method targeting CSS, struggles to learn the new class but still suffers from forgetting. The methods from MiB start to explicitly consider the background shift in CSS and all of them have steady improvements. Our method achieves the best overall performance in both disjoint and overlapped settings. We notice that despite of only a single class being added, this setting is actually difficult since the training set is rather small and monotonous.

Table 1. Results of different methods on **overlapped** Pascal-VOC 2012. Best in **bold**.

	19-1 (2 tasks)			15-5 (2 tasks)			15-1 (6 tasks)		
Method	old	new	all	old	new	all	old	new	all
Finetune	34.7	14.9	33.8	12.5	36.9	18.3	4.9	3.2	4.5
ILT[36]	67.1	12.3	64.4	66.3	40.6	59.9	4.9	7.8	5.7
MiB[3]	70.2	22.1	67.8	75.5	49.4	69.0	46.2	12.9	37.9
SDR[37]	69.1	32.6	67.4	75.4	52.6	69.9	44.7	21.8	39.2
PLOP[13]	75.3	**37.3**	73.5	75.7	51.7	70.0	65.1	21.1	54.6
Ours	**76.5**	36.2	**74.6**	**78.1**	**52.9**	**72.1**	**66.2**	**23.3**	**56.0**
Upper Bound	77.6	76.7	77.5	79.0	72.8	77.5	79.0	72.8	77.5

Table 2. Results of different methods on **disjoint** Pascal-VOC 2012. Best in **bold**.

	19-1 (2 tasks)			15-5 (2 tasks)			15-1 (6 tasks)		
Method	old	new	all	old	new	all	old	new	all
Finetune	35.2	13.2	34.2	8.4	33.5	14.4	5.8	4.9	5.6
ILT[36]	69.1	16.4	66.4	63.2	39.5	57.3	3.7	5.7	4.2
MiB[3]	69.6	25.6	67.4	71.8	43.3	64.7	35.1	13.5	29.7
SDR[37]	69.9	37.3	68.4	73.5	47.3	67.2	59.2	12.9	48.1
PLOP[13]	75.3	**38.8**	73.6	71.0	42.8	64.2	57.8	13.6	46.4
Ours	**75.5**	38.0	**73.7**	**75.3**	**48.7**	**69.0**	**59.6**	**15.6**	**49.1**
Upper Bound	77.6	76.7	77.5	79.0	72.8	77.5	79.0	72.8	77.5

Single step addition of five classes (15-5). In this setup, the remaining 5 classes are added in a single step after learning the first 15 classes. It has the most severe semantic shift due to the most classes being added in one go. Such a setup requires the model to have enough plasticity to learn 5 classes in one go while being stable on old knowledge. In this scenario, our method outperforms all previous works by a large margin in both disjoint and overlapped settings. In the disjoint setting, our method improves by about 2% compared to SDR and about 5% compared to PLOP. In the overlapped setting, our method has another gain of about 2% compared to the second place. What's more, our method boosts the performance of old classes to only 1% lower than the upper bound, together with a notable gain in new class performance.

Multi-step addition of five classes (15-1). This is similar to 15-5 except that the last 5 classes are added one by one, making it the most challenging because of multiple learning phases. As presented in Table 1 and Table 2, all methods suffer a great performance drop. ILT almost forgets all the knowledge. Even the second place, PLOP, has a significant drop compared to 15-5. Our method again achieves state-of-the-art performance on both the disjoint and overlapped setups, proving its ability under longer learning sequences.

Visual results. We visualize the results of all methods on different setups to unveil more details. As shown in Fig. 3, our method consistently produces better segmentation results than all competitors. For example, in 19-1 our method successfully distinguishes all three classes while the competitors fail to recognize at least one class. The same situation can be observed in 15-5 and 15-1, in which our method generates a segmentation map closer to the ground truth and less confusion between classes is observed. We further provide a visualization of each step in 15-1 setup to present the knowledge shift across each step. As shown in Fig. 4, all the competitors suffer more or less from catastrophic forgetting. The change is especially great in step 5, in which most of the old classes are misclassified into the new class *train* learned in this step. But our method shows good robustness against forgetting, which presents constant segmentation results across each step.

5.3 Evaluation on ADE20K

In this section, we evaluate our method on the ADE20K dataset. Following [13,3], we adopt three setups on the overlapped protocol: adding the last 50 classes in

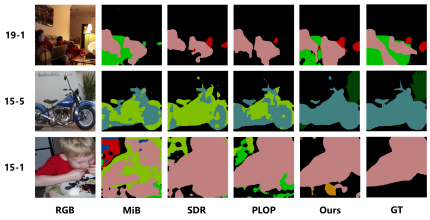

Fig. 3. Visual results of competing methods on different setups of overlapped VOC2012. Best viewed in color.

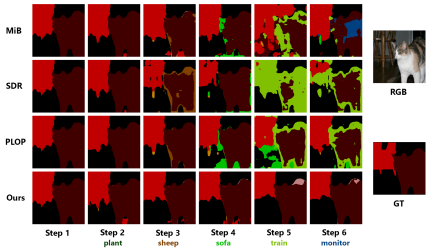

Fig. 4. Visual results of competing methods on each step of 15-1 overlapped VOC2012. Best viewed in color.

a single step (100-50), adding 50 classes each step (50-50), and adding 10 classes each step for the last 50 classes (100-10). ADE20K is much more complex than VOC2012, as we can only get 37.9% mIoU under the offline training. We compare the same methods as in Section 5.2 except for SDR since it does not provide their results on ADE20K.

The full results are presented in Table 3. ILT suffers from critical forgetting, while PLOP and MiB achieve much better results. Our method outperforms all previous methods in both 100-50 and 50-50 setups with an improvement of more than 2% compared to PLOP. When diving into longer learning sequences, our method shows higher performance on old classes, and achieves comparable overall performance compared to PLOP. These experiments show the effectiveness of our method on a large-scale dataset.

Table 3. Results of different methods on overlapped ADE20K in mIoU, best in **bold**

	100-50 (2 tasks)			50-50 (3 tasks)			100-10 (6 tasks).		
Method	old	new	all	old	new	all	old	new	all
ILT[36]	18.2	14.4	17.0	3.5	12.8	9.7	0.1	3.0	1.0
MiB[3]	40.5	17.1	32.7	45.5	21.0	29.3	38.2	11.1	29.2
PLOP[13]	41.8	14.8	32.9	48.8	20.9	30.4	40.4	**13.6**	**31.5**
Ours	**42.9**	**19.9**	**35.2**	**49.8**	**23.9**	**32.5**	**41.0**	12.5	**31.5**
Upper Bound	43.5	26.7	37.9	50.3	31.7	37.9	43.5	26.7	37.9

5.4 Ablation Study

We investigate the effects of key components in our framework using an ablation study on 15-1 overlapped VOC2012. Starting from the basic fine-tuning, we gradually add the proposed components upon it. The full results are shown in Table 4. It can be seen that the performance is boosted steadily as more components are added. Each module in our framework can bring benefits to both old and new classes. We conducted another experiment by removing a single component from the framework to better show their individual contribution to per-

formance. Among all the components, the structure preserving loss contributes the most to performance (-4.7% if removed), and the second is the projected feature alignment (-2.1% if removed). This well demonstrates the effectiveness of the two proposed techniques.

Table 4. Performance contribution of each component on 15-1 overlapped VOC2012

L_{ce}	*pseudo*	L_{ali}	L_{str}	L_{cont}	old	new	all
✓					4.9	3.2	4.5
✓	✓				23.5	6.6	19.5
✓	✓	✓			60.8	17.2	50.4
✓	✓	✓	✓		65.3	21.2	54.8
✓	✓	✓	✓	✓	**66.2**	**23.0**	**55.9**
✓	✓		✓	✓	63.7	22.0	53.8
✓	✓	✓		✓	61.7	17.8	51.2

5.5 Further Analysis

Forgetting in Decoder Head. To give a straightforward view of the forgetting phenomenon in decoder head, we utilize Canonical Correlation Analysis [38] to measure the model output similarity between adjacent incremental steps. Results are shown in Fig. 5. The X-axis indicates different layers of the network. The Y-axis represents learning steps. Lower value on step 1 indicates more forgetting. (a) only performs probability distillation and results in the most forgetting. (b) adds constraints on intermediate features, which alleviates the forgetting of encoder, but a clear gap can be observed in decoder output ($0.943 \rightarrow 0.765$). (c) further applies constraints on decoder as in our framework, which alleviates this forgetting and improves the overall performance.

Effect of Structure Preserving Loss. To demonstrate the effects of the structure preserving loss against the widely used prototype alignment, we performed an experiment by replacing structure preserving loss with the prototype alignment used in [37]. We select five old classes (*plane, bicycle, bird, boat, bottle*) and use t-SNE [31] to visualize their distributions in embedding space after training on new classes. To erase the randomness in initial learning phase, we use the same checkpoint learned the first 15 classes to start from. Results are shown in Fig. 6. Our structure preserving loss better maintains the discriminative power (*i.e.*, a clearer boundary) of old classes than prototype alignment.

Effect of Projected Feature Alignment. We conducted an experiment on 15-1 overlapped VOC by adding different feature space constraints on a simple baseline, which is made of L_{ce} + pseudo label. Results of *all* classes are shown in Table 5. Pixel-wise aligns the feature map directly in a pixel-wise manner. Pooled indicates the practice in [13]. Our proposed module successfully surpasses its contenders by a clear margin of 2.3%.

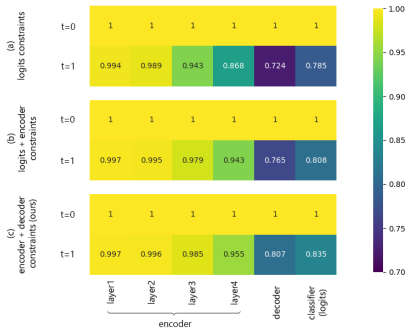

Fig. 5. Visualization of the forgetting phenomena in segmentation network.

Fig. 6. Comparing the embedding space distribution results by prototype alignment and structure preserving loss.

Table 5. Performance comparison of different feature space constraints

Baseline	Pixel-wise	Pooled[13]	Ours
18.9	33.7	48.2	**50.5**

6 Limitation and Conclusion

Continual learning is often applied to situations where computational and space overhead are important. Our method, which is exemplar-free, does not require any extra storage. But we do admit that there's some extra computational cost of our method. It is mainly from two aspects: The first is from the training of projector P, which takes roughly 10 minutes per incremental step. The second is from all the proposed modules, which makes the training about 10% slower than fine-tuning. We think this is within an acceptable range for most situations.

In this paper, we present a novel framework to deal with the forgetting in both encoder and decoder of segmentation network. In detail, the projected feature alignment module is designed to disentangle the preservation of old knowledge from integrating new knowledge. The structure preserving loss exploits inter-class and intra-class structures to maintain the discriminability of each class in the embedding space. Their effects are demonstrated by extensive experiments. Though our method managed to achieve a better stability-plasticity trade-off, the cause of large performance gap to the upper bound for new classes still remains to be explored in the future work.

Acknowledge This work is supported by the National Natural Science Foundation of China under Grant No.62176246 and No.61836008

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39**(12), 2481–2495 (2017)
2. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV (2018)
3. Cermelli, F., Mancini, M., Bulo, S.R., Ricci, E., Caputo, B.: Modeling the background for incremental learning in semantic segmentation. In: ICCV (2020)
4. Chaudhry, A., Gordo, A., Dokania, P.K., Torr, P., Lopez-Paz, D.: Using hindsight to anchor past knowledge in continual learning. In: AAAI (2021)
5. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
6. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
7. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018)
8. Cong, Y., Zhao, M., Li, J., Wang, S., Carin, L.: Gan memory with no forgetting. In: NIPS (2020)
9. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
10. De Lange, M., Tuytelaars, T.: Continual prototype evolution: Learning online from non-stationary data streams. In: ICCV (2021)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
12. Dhar, P., Singh, R.V., Peng, K.C., Wu, Z., Chellappa, R.: Learning without memorizing. In: CVPR (2019)
13. Douillard, A., Chen, Y., Dapogny, A., Cord, M.: Plop: Learning without forgetting for continual semantic segmentation. In: CVPR (2021)
14. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: ECCV (2020)
15. Everingham, M., Winn, J.: The pascal visual object classes challenge 2012 (voc2012) development kit. Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep (2011)
16. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
17. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual attention network for scene segmentation. In: CVPR (2019)
18. Hayes, T.L., Kafle, K., Shrestha, R., Acharya, M., Kanan, C.: Remind your neural network to prevent catastrophic forgetting. In: ECCV (2020)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
20. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
21. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science **313**(5786), 504–507 (2006)

22. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: ICCV (2019)
23. Hu, H., Cui, J., Wang, L.: Region-aware contrastive learning for semantic segmentation. In: ICCV (2021)
24. Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W.: Ccnet: Criss-cross attention for semantic segmentation. In: ICCV (2019)
25. Iscen, A., Zhang, J., Lazebnik, S., Schmid, C.: Memory-efficient incremental learning through feature adaptation. In: ECCV (2020)
26. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
27. Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML (2013)
28. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: ICML (2019)
29. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
30. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
31. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
32. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: ECCV (2018)
33. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: CVPR (2018)
34. Maltoni, D., Lomonaco, V.: Continuous learning in single-incremental-task scenarios. Neural Networks **116**, 56–73 (2019)
35. Mehta, S., Rastegari, M., Caspi, A., Shapiro, L., Hajishirzi, H.: Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In: ECCV (2018)
36. Michieli, U., Zanuttigh, P.: Incremental learning techniques for semantic segmentation. In: ICCVW (2019)
37. Michieli, U., Zanuttigh, P.: Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In: CVPR (2021)
38. Raghu, M., Gilmer, J., Yosinski, J., Sohl-Dickstein, J.: Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In: NIPS (2017)
39. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR (2017)
40. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28**, 91–99 (2015)
41. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention (2015)
42. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NIPS (2017)
43. Xiang, Y., Fu, Y., Ji, P., Huang, H.: Incremental learning using conditional adversarial networks. In: ICCV (2019)

44. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: CVPR (2021)
45. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)
46. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML. pp. 3987–3995 (2017)
47. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
48. Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C.C., Lin, D., Jia, J.: Psanet: Point-wise spatial attention network for scene parsing. In: ECCV (2018)
49. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR (2017)