Early Preparation Pays Off: New Classifier Pre-tuning for Class Incremental Semantic Segmentation

Zhengyuan Xie[1], Haiquan Lu[1], Jia-wen Xiao[1], Enguang Wang[1], Le Zhang[3], and Xialei Liu[1,2(✉)]

[1] VCIP, CS, Nankai University
[2] NKIARI, Shenzhen Futian
{xiezhengyuan}@mail.nankai.edu.cn
{xialei}@nankai.edu.cn
[3] SICE, UESTC

Abstract. Class incremental semantic segmentation aims to preserve old knowledge while learning new tasks, however, it is impeded by catastrophic forgetting and background shift issues. Prior works indicate the pivotal importance of initializing new classifiers and mainly focus on transferring knowledge from the background classifier or preparing classifiers for future classes, neglecting the flexibility and variance of new classifiers. In this paper, we propose a new classifier pre-tuning (NeST) method applied before the formal training process, learning a transformation from old classifiers to generate new classifiers for initialization rather than directly tuning the parameters of new classifiers. Our method can make new classifiers align with the backbone and adapt to the new data, preventing drastic changes in the feature extractor when learning new classes. Besides, we design a strategy considering the cross-task class similarity to initialize matrices used in the transformation, helping achieve the stability-plasticity trade-off. Experiments on Pascal VOC 2012 and ADE20K datasets show that the proposed strategy can significantly improve the performance of previous methods. The code is available at https://github.com/zhengyuan-xie/ECCV24_NeST.

Keywords: Class incremental learning · Semantic segmentation

1 Introduction

Nowadays, segmentation models driven by deep neural networks have achieved notable success in various fields [8, 9, 31]. However, these models are usually trained and tested in a static environment [18], which is contrary to real-world scenarios [43]. When facing a continuous data stream [22], a model needs to handle several novel classes. Blending new and previous data to retrain the model, or called *Joint Training*, ensures the performance of the model. However, the computational cost becomes prohibitive when dealing with large datasets. Simply fine-tuning the model may lead to the loss of previously acquired knowledge, a

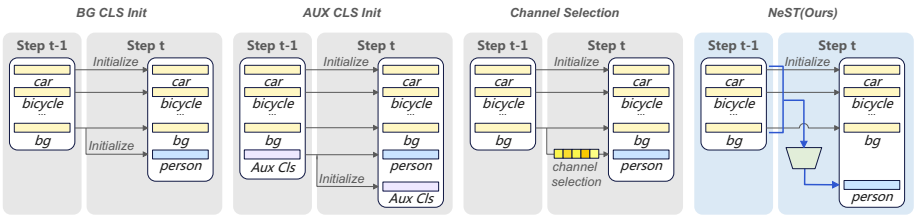


Fig. 1: Different classifier initialization methods for class incremental semantic segmentation. MiB [4] directly uses background classifiers to initialize new classifiers. Some methods [2,5] train an auxiliary classifier for future classes. AWT [19] selects the most relevant weights from the background classifier for new classifiers' initialization by gradient-based attribution. Our new classifier pre-tuning method learns a transformation from all old classifiers to generate new classifiers for initialization.

phenomenon known as catastrophic forgetting [27,34]. In particular, for semantic segmentation, it also faces the problem of background shift [4] (i.e., pixels labeled as background in the current step may belong to previous or future classes).

Class incremental semantic segmentation (CISS) [4, 5, 14, 51] has been proposed to address the challenges of catastrophic forgetting and background shift. It requires the segmentation model to learn concepts of new classes while preserving old knowledge. If no restrictions are imposed, parameters crucial to old classes may be updated in the wrong direction [26], causing the model to forget previously learned knowledge. On the contrary, only focusing on memorizing old knowledge may limit the model to learning new classes. Thus, the key issue to continual learning is actually the trade-off between stability and plasticity [20, 26].

As far as stability is concerned, existing methods rely on old models learned from the previous step for knowledge transfer [4, 14, 46, 51], *e.g. weight initialization*. Yet the old model does not have the ability to recognize new classes, finding a proper way to initialize the newly added classifier is crucial. Random initialization may cause the misalignment between new classifiers and features, leading to training instabilities [4]. In accordance with this situation, as Fig. 1 shows, MiB [4] utilizes the background classifier to initialize new classifiers, as future classes may appear in current data, labeled as the background. However, it may cause the model to make incorrect predictions for true background pixels [19]. AWT [19] selects the most relevant weight for initialization from the old background classifier via a gradient-based attribution technique, yet it neglects other old classifiers and brings a huge memory cost. Some methods train an auxiliary classifier to initialize new classifiers [2,5], while there is still a bias between the auxiliary classifier and true future classifiers because there is not any future data with ground truth. What's more, the above initialization methods treat each new classifier equally, ignoring the differences between new classes.

From the above observations, we propose **Ne**w cla**S**sifier pre-**T**uning (NeST) method to make new classifiers better align with the backbone and adapt to the training data, preventing drastic changes in the feature extractor caused

by unstable training process. To better utilize the old knowledge, instead of directly tuning the parameters of new classifiers, we tune a linear transformation from all old classifiers to generate new classifiers. Specifically, before the formal training process at the current step, we assign two transformation matrices, *i.e.*, an importance matrix, and a projection matrix, to each new class. As trainable parameters, the importance matrix learns a weighted score associated with each channel of the old classifier pertinent to the new class, while the projection matrix learns a linear combination from the weighted old classifiers to generate the new classifier. Subsequently, we use data from the current step to learn the transformation from old classifiers to each new classifier. Finally, we employ learned importance matrices and projection matrices to generate new classifiers from old ones, and then leverage the derived weights for classifier initialization of the formal training process. By using these two matrices, new classifiers are generated via old knowledge and are different from each other.

To achieve the stability-plasticity trade-off, we also find it crucial for the initial values of these transformation matrices and propose a strategy for initialization by considering cross-task class similarities between old and new classes, thus facilitating the learning of new classifiers.

To summarize, the main contributions of our paper are threefold:

1. We propose a new classifier pre-tuning (NeST) method that learns a transformation from all old classifiers to generate new classifiers with previous knowledge before the formal training process.
2. We further optimize the initialization of transformation matrices, striking a balance between stability and plasticity.
3. We conduct experiments on Pascal VOC 2012 and ADE20K. Results show that NeST can be readily integrated into other approaches and significantly enhance their performance.

2 Related Work

Semantic Segmentation. In recent years, the development of deep learning techniques has facilitated the performance of semantic segmentation models. Fully Convolutional Networks (FCN) [32] pioneers semantic segmentation, and a series of convolution-based segmentation networks have achieved high performance in many benchmarks [7, 17, 21, 39, 44, 52]. More recently, the Transformer architecture has become ever more popular and reached remarkable performance [10,11,30,38,41,47,53]. Thus, we conduct experiments on two different backbones: ResNet and Swin-Transformer.

Class Incremental Learning. Class incremental learning tackles the catastrophic forgetting of the task with ever-increasing categories. The whole training process is divided into several steps and in each step, the model is required to learn one or more classes, which is the most challenging. Existing methods can be categorized into three groups. The *model expansion* methods [24, 25, 48, 50] enlarge the model size in incremental steps to learn new knowledge while preserving old knowledge in fixed parameters. The *rehearsal based* methods store a

series of exemplars [3, 42] or prototypes along the task sequence [55], or using generative networks to maintain old knowledge [33, 40]. The *parameter regularization* methods either constrain the learning of some parameters [6, 28] or use knowledge distillation techniques [1, 14, 15, 45].

Class Incremental Semantic Segmentation. CISS requires a segmentation model to recognize all learned classes in continuous learning steps. Different from classification tasks, segmentation has its own background shift [4] issue. MiB [4] first proposes unbiased cross entropy and distillation to address background shift. PLOP [14] uses pseudo-label and intermediate features to transfer old knowledge. SDR [36] proposes a contrastive-learning-based method, minimizing intra-class feature distances. RCIL [51] introduces reparameterization to continual semantic segmentation with a complementary network structure. GSC [12] explores incremental semantic segmentation considering the gradient and semantic compensation. EWF [46] fuses old and new knowledge with a weight fusion strategy. Many methods have explored the classifier initialization in CISS. SSUL [5] employs auxiliary data to train an 'unknown' classifier and use it to initialize new classifiers. DKD [2] proposes a decomposed knowledge distillation, improving the rigidity and stability. AWT [19] applies gradient-based attribution to transfer the relevant weight of old classifiers to new classifiers. Different from the method above, we propose a new classifier pre-tuning method, achieving the goal of the trade-off between plasticity and stability.

3 Method

3.1 Preliminaries

Problem Definition. Following [4, 14], CISS contains several steps $\{t\}_{t=1}^{n}$ to receive sequential data stream $\{\mathcal{D}_t\}_{t=1}^{n}$ with classes $\{\mathcal{C}_t\}_{t=1}^{n}$. In each step t, the model is required to learn $|\mathcal{C}_t|$ new classes, while old training data is not available. For an image in the current step, pixels belonging to $\mathcal{C}_t$ are labeled as their ground truth classes, leaving other pixels labeled as background. Finally, after the last step, the model will be tested on the data of all learned classes.

At step t, the segmentation model is composed of a feature extractor f_θ^t and a classifier h_ϕ^t. ϕ_t is newly added and weights of previous classifiers are $\phi_{1:t-1}$. In this paper, our main concern is the initialization of ϕ_t.

Revisiting Previous Initialization Methods. In this part, we illustrate the shortcomings of previous initialization methods and further clarify our motivation. As mentioned before, classes in $\mathcal{C}_t$ will appear as background in previous steps, which is known as the *background shift*. According to this premise, previous methods either try to find a better way of utilizing the background classifier [4, 19], or train an auxiliary classifier for future classes, both neglecting the differences between new classes [2, 5]. Meanwhile, there will also be a mismatching issue between new classifiers and the feature extractor as the methods above do not have training processes with data from new classes. This may lead to drastic parameter changes in the feature extractor when training with new data, thus undermining previous knowledge.

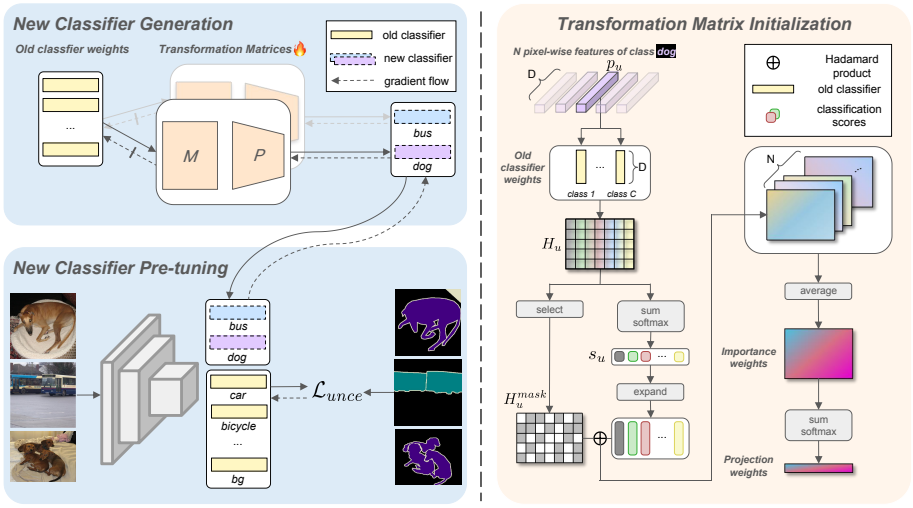


Fig. 2: Illustration of our new classifier pre-tuning (NeST) method. The left side of the figure is an iteration of the new classifier pre-tuning process. The right side of the figure represents the cross-task class similarity-based initialization of importance matrices and projection matrices before the pre-tuning process.

3.2 New Classifier Pre-Tuning

Considering the observations above, we propose a simple yet effective method, NeST, to improve the performance of CISS. In the pre-tuning stage before the formal training process, we first learn to generate each new classifier via old classifiers and then tune it with new data in each forward pass.

New Classifier Generation. As shown in Fig. 2, for each new class $c \in \mathcal{C}_t$, we assign an importance matrix and a projection matrix, learning the transformation from all old classifiers in each forward pass as follows:

$$\mathbf{w}_c = (\mathbf{M}_c \odot \mathbf{W}_{old})\mathbf{P}_c\,, \tag{1}$$

where $\mathbf{w}_c \in \mathbb{R}^d$ denotes the weight of new class c, $\mathbf{M}_c \in \mathbb{R}^{d \times n_{old}}$ denotes the importance matrix of class c and $n_{old} = \left|\bigcup_{i=1}^{t-1} \mathcal{C}_i\right| + 1$ denotes the number of old classes including the background, $\odot$ denotes the operation of Hadamard product, $\mathbf{W}_{old} \in \mathbb{R}^{d \times n_{old}}$ denotes the weight matrix of all old classifiers, $\mathbf{P}_c \in \mathbb{R}^{n_{old} \times 1}$ denotes the projection matrix of class c, and d denotes the output dimension of f_θ^{t-1}. $\mathbf{M}_c$ reflects the importance of each channel of the old classifiers for the new class c, and $\mathbf{P}_c$ combines weighted old classifiers, completing weight generation for the new class. After generating new weights, we concatenate them as the parameter of ϕ_t as follows:

$$\phi_t = \mathrm{concat}[\mathbf{w}_{\mathrm{n_{old}}}, ..., \mathbf{w}_{\mathrm{n_{old}}+|\mathcal{C}_t|-1}]\,, \tag{2}$$

where concat[] is the concatenation operation. Specifically, as the old background class may see new classes in previous steps and the score of background in the pre-tuning process may be high, preventing the model from learning new classes, we also learn a transformation from the old background classifier to the new background classifier during the pre-tuning process as follows:

$$\hat{\mathbf{w}}_0 = (\mathbf{M}_0 \odot \mathbf{w}_0)\mathbf{P}_0\,, \tag{3}$$

where $\mathbf{w}_0 \in \mathbb{R}^d$ denotes the previous background classifier, $\mathbf{M}_0 \in \mathbb{R}^{d\times 1}$ and $\mathbf{P}_0 \in \mathbb{R}^{1\times 1}$ are matrices specifically for the background, and $\hat{\mathbf{w}}_0 \in \mathbb{R}^d$ is new background classifier prepared for current task.

Pre-tuning and Initialization. In each forward pass, as Fig. 2 shows, after generating weights of new classifiers, we feed training data from the current step to the old model equipped with generated classifiers, using the output to calculate the loss and then backpropagating the loss to update learnable parameters. The loss function used in the pre-tuning is unbiased cross entropy $\mathcal{L}_{unce}$ [4], as it can help to avoid the overfitting problem [37]:

$$\mathcal{L}_{unce} = -\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \log \tilde{q}_x^t(i, y_i)\,, \tag{4}$$

where $\mathcal{I}$ denotes the pixel set of an image, y_i denotes the ground-truth label of pixel i at current step, $\tilde{q}_x^t$ denotes the modified output of the current model. It should be noted that during the whole pre-tuning process, the importance matrices and projection matrices are learnable and will be updated, while other components such as the feature extractor remain frozen. After the pre-tuning process, we use importance matrices and projection matrices to generate weights of each new classifier for initialization, and additional parameters will be removed.

3.3 Cross-Task Class Similarity-based Transformation Matrix Initialization

We found it crucial to initialize importance matrices and projection matrices. Random initialization disregards the importance of different channels among old classifiers and results in poor performance. AWT [19] reveals that only a few channels of the background classifier contribute to the classification of new classes, redundant channels may hinder the learning of new classifiers. Meanwhile, for different new classes, the important channels of old classifiers should be different. Hence, we introduce a cross-task class similarity-based transformation matrix initialization method. The core idea is that if an old class is more similar to a new class, then during the pre-tuning process, the contribution of this old classifier should be more significant, and thus, greater initial weights should be assigned to corresponding positions in the importance matrix and projection matrix.

Hence, we employ the predictions made by the old model as cross-task class similarity scores for each new class pixel, aiming to assess the resemblance between the new class and every old class. As shown in Fig. 2, before learning

transformation, we feed each training image in the current step to the old model and get the corresponding output of the layer before h_{ϕ}^{t-1}. Then considering the classification process of a new class pixel embedding $\mathbf{p}_u \in \mathbb{R}^d$ extracted from the old model, assuming that the total number of old classes is n_{old}, the matrix multiplication can be decomposed into an element-wise multiplication (also called Hadamard product) between $\mathbf{p}_u$ and the old classifier weight $\mathbf{W}_{old} \in \mathbb{R}^{d \times n_{old}}$ and a sum-softmax operation, as follows:

$$\begin{aligned} \mathbf{H}_u &= (\mathbf{W}_{old} \odot \mathbf{p}'_u), \\ \mathbf{s}_u &= \text{softmax}(\text{sum}(\mathbf{H}_u))^\top , \end{aligned} \tag{5}$$

where $\mathbf{p}'_u \in \mathbb{R}^{d \times n_{old}}$ denotes broadcasting $\mathbf{p}_u$ by n_{old} times, $\mathbf{H}_u \in \mathbb{R}^{d \times n_{old}}$ denotes the result of the Hadamard product, sum denotes summing $\mathbf{H}_u$ along the channel dimension, and $\mathbf{s}_u \in \mathbb{R}^{n_{old}}$ denotes the classification scores of $\mathbf{p}_u$. We consider that for each column (corresponding to an old class i) of $\mathbf{H}_u$, the positive elements contribute to $\mathbf{p}_u$ being classified into class i. Thus, to select relevant channels, we set all positions with positive values to 1 and get $\mathbf{H}_u^{mask}$, which can be regarded as a binary mask, as follows:

$$\mathbf{H}_u^{mask}(i,j) = \begin{cases} 1, \mathbf{H}_u(i,j) > 0 \\ 0, otherwise\,. \end{cases} \tag{6}$$

Finally, by utilizing ground-truth masks of the current step, for each pixel embedding belonging to new class $\boldsymbol{c_{new}}$, we calculate the Hadamard product between its corresponding $\mathbf{H}_{mask}$ and predicted score, then averaged the results to get the importance matrix weight $\mathbf{M}_{c_{new}}$ of class c_{new}, as follows:

$$\mathbf{M}_{c_{new}} = \frac{1}{N} \sum_{\mathbf{p}_u} \mathbf{H}_u^{mask} \odot \mathbf{s}'_u\,, \tag{7}$$

where $\mathbf{p}_u$ denotes the pixel embedding belonging to the new class c_{new}, $\mathbf{s}'_u \in \mathbb{R}^{d \times n_{old}}$ denotes broadcasting the predicted score $\mathbf{s}_u$ of $\mathbf{p}_u$ by d times, N denotes the number of pixels belonging to new class c_{new}. Here we use $\mathbf{s}'_u$ because the old class with a very small score will not contribute too much to the initialization of the new class, as the similarity between these two classes is relatively low. For projection matrix weight $\mathbf{P}_{c_{new}}$, we sum up $\mathbf{M}_{c_{new}}$ along the channel dimension and apply the softmax function to get the weight score for old classes, as follows:

$$\mathbf{P}_{c_{new}} = (\text{softmax}(\text{sum}(\mathbf{M}_{c_{new}})))^\top . \tag{8}$$

Then we use $\mathbf{P}_{c_{new}} \in \mathbb{R}^{n_{old} \times 1}$ to initialize the projection matrix of class c_{new} as these scores reflect the degree of similarity between the new class and old classes, deciding which old class the model should focus on for knowledge transfer. We provide the pseudo-code of our NeST in Algorithm 1.

Algorithm 1 Pseudo-code of the proposed NeST.

Input: Training samples $\mathcal{D}_t = \{(x_i, y_i)\}_t$ of current task t, feature extractor f_θ^0, classifier h_ϕ^0, task number T, new class set $\mathcal{C}_t$.
Output: initial weight of new classifier ϕ_t

for $t \in \{1, 2, ..., T\}$ **do**
  for $i \in \mathcal{C}_t$ **do** ▷ Initializing Transformation matrices. Sec. 3.3
    $(\mathbf{M}_i, \mathbf{P}_i) \leftarrow Init(\mathcal{D}_t, h_\phi^{t-1}, f_\theta^{t-1})$
  end for
  while not converged **do** ▷ Pre-tuning new classifiers. Sec. 3.2
    train $(\mathbf{M}, \mathbf{P})$ by minimizing $\mathcal{L}_{unce}$
  end while
  $\phi_t \leftarrow \text{Transform}(\mathbf{M}, \mathbf{P}, \phi_{1:t-1})$ ▷ Initializing new classifiers.
  $f_\theta^t \leftarrow f_\theta^{t-1}$
  $h_\phi^t \leftarrow (h_\phi^{t-1}, \phi_t)$
  while not converged **do** ▷ Formal training process.
    train f_θ^t and h_ϕ^t
  end while
end for

4 Experiments

4.1 Experimental setup

Protocols. In CISS, the whole training process contains T steps, and the task of each step may contain one or more classes. In step t, pixels belonging to previous steps are labeled as *background*. In evaluation, the model needs to identify all learned classes. There are two settings: *Disjoint* and *Overlapped* [4]. *Disjoint* setting assumes that in the current step, images do not contain any pixel belonging to classes that will be learned in the future. *Overlapped* setting is more realistic, as future classes may appear in images from the current step, and we mainly evaluate our method on *Overlapped* setting.

Datasets. Pascal VOC 2012 dataset [16] contains 20 classes including 10,582 images for training and 1449 images for validation. ADE20K dataset [54] contains 150 classes including 20,210 images for training and 2000 images for validation. In CISS, the training setting can be described as $X - Y$, X means the number of classes trained in the initial step and Y means the number of classes trained in incremental steps. We conduct experiments with *10-1*, *15-1*, *15-5* and *19-1* settings on Pascal VOC 2012 dataset [16]. For ADE20K dataset [54], we validate our method on *100-50*, *100-10*, *100-5* and *50-50* settings.

Implementation Details. Following previous methods [4, 14, 51], we use DeepLabV3 [7] with ResNet-101 [23] as our segmentation model. We also use Swin-B [30] as the Transformer-based backbone. Following [4,14], we use random crop and horizontal flip for data augmentation. We train the model with a batch size of 24 on 4 GPUs for all experiments. We use SGD as the optimizer in our experiments. The learning rate is set to 0.02 for the initial step and 0.001 for incremental steps, both for Pascal VOC 2012 and ADE20K. We also use *poly*

Table 1: The mIoU (%) of the last step on the Pascal VOC dataset for four different CISS scenarios. * implies results from the re-implementation of the official code.

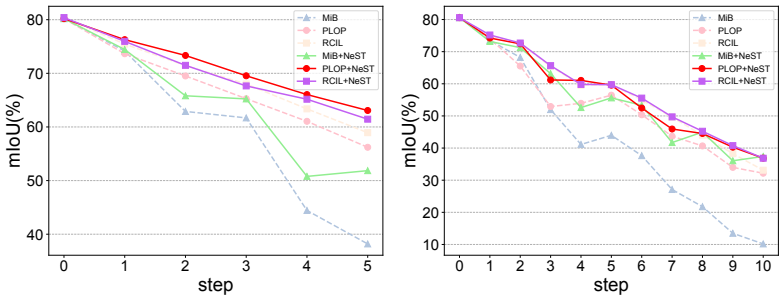
Method	Backbone	10-1(11 steps)			15-1(6 steps)			15-5(2 steps)			19-1(2 steps)		
		0-10	11-20	all	0-15	16-20	all	0-15	16-20	all	0-19	20	all
Joint	Res101	78.8	77.7	78.3	79.8	73.4	78.3	79.8	73.4	78.3	78.2	80.0	78.3
ILT [35]	Res101	7.2	3.7	5.5	9.6	7.8	9.2	67.8	40.6	61.3	68.2	12.3	65.5
SDR [36]	Res101	32.4	17.1	25.1	44.7	21.8	39.2	75.4	52.6	69.9	69.1	32.6	67.4
PLOP+UCD [49]	Res101	42.3	28.3	35.3	66.3	21.6	55.1	75.0	51.8	69.2	75.9	39.5	74.0
SPPA [29]	Res101	-	-	-	66.2	23.3	56.0	78.1	52.9	72.1	76.5	36.2	74.6
MiB+AWT [19]	Res101	33.2	18.0	26.0	59.1	17.2	49.1	77.3	52.9	71.5	-	-	-
ALIFE [37]	Res101	-	-	-	64.4	34.9	57.4	77.2	52.5	71.3	76.6	49.4	75.3
GSC [12]	Res101	50.6	17.3	34.7	72.1	24.4	60.8	78.3	54.2	72.6	76.9	42.7	75.3
MiB [4]	Res101	12.2	13.1	12.6	38.0	13.5	32.2	76.4	49.4	70.0	71.2	22.1	68.9
MiB* [4]	Res101	10.4	9.9	10.2	45.2	15.7	38.2	76.8	49.1	70.2	71.6	28.6	69.6
MiB+NeST (Ours)	Res101	52.3	21.0	37.4	61.7	20.4	51.8	77.1	50.1	70.7	71.7	28.2	69.7
PLOP [14]	Res101	44.0	15.5	30.5	65.1	21.1	54.6	75.7	51.7	70.1	75.4	37.4	73.5
PLOP* [14]	Res101	45.9	17.1	32.2	66.8	22.3	56.2	77.0	50.9	70.8	75.7	39.4	74.0
PLOP+NeST (Ours)	Res101	54.2	17.8	36.9	72.2	33.7	63.1	77.6	55.8	72.4	77.0	49.1	75.7
RCIL [51]	Res101	55.4	15.1	34.3	70.6	23.7	59.4	78.8	52.0	72.4	77.0	31.5	74.7
RCIL* [51]	Res101	47.8	17.0	33.1	69.9	23.9	58.9	78.8	52.4	72.5	76.8	28.9	74.5
RCIL+NeST (Ours)	Res101	51.4	20.9	36.8	71.9	28.0	61.4	79.0	52.8	72.8	77.0	33.3	74.9
Joint	Swin-B	80.4	79.7	80.1	81.1	76.7	80.1	81.1	76.7	80.1	80.0	80.7	80.1
MiB* [4]	Swin-B	11.4	18.9	15.0	35.0	43.2	36.9	80.7	66.5	77.3	79.2	60.2	78.3
MiB+NeST (Ours)	Swin-B	65.2	35.8	51.2	77.0	53.3	71.4	81.2	67.4	77.9	79.7	60.0	78.8
PLOP* [14]	Swin-B	37.8	23.1	30.8	74.1	52.1	68.9	80.1	68.1	77.2	77.0	65.8	76.4
PLOP+NeST (Ours)	Swin-B	64.3	28.3	47.2	76.8	57.2	72.2	80.5	70.8	78.2	79.6	70.2	79.1

schedule as the weight decay strategy. For Pascal VOC 2012, we pre-tune for 5 epochs with a learning rate of 0.001. For ADE20K, pre-tuning lasts 15 epochs, with a learning rate of 0.1 for experiments involving RCIL and Swin-B, and 0.5 for other experiments. Further details are provided in the supplementary material.

4.2 Results

In this section, we apply NeST to three classic methods, MiB [4], PLOP [14] and RCIL [51] with different backbones.

Pascal VOC 2012. We conduct experiments on *10-1*, *15-1*, *15-5* and *19-1* settings. As shown in Tab. 1, with NeST, MiB [4], PLOP [14] and RCIL [51] can achieve huge performance gains. On the *15-1* setting with Deeplab, by simply using the proposed new classifier strategy, we can improve the performance of MiB, PLOP, and RCIL by 13.6%, 6.9%, and 2.5%, respectively. In the more difficult *10-1* scenario, NeST can boost the performance of MiB by a large margin, achieving an improvement of 27.2%. Meanwhile, results with Swin-B show that NeST is also suitable for Transformer-based models. On MiB with Swin-B, NeST can improve the performance by 34.5% on the *15-1* setting and 36.2%



(a) *15-1* on PASCAL VOC 2012 **(b)** *10-1* on PASCAL VOC 2012

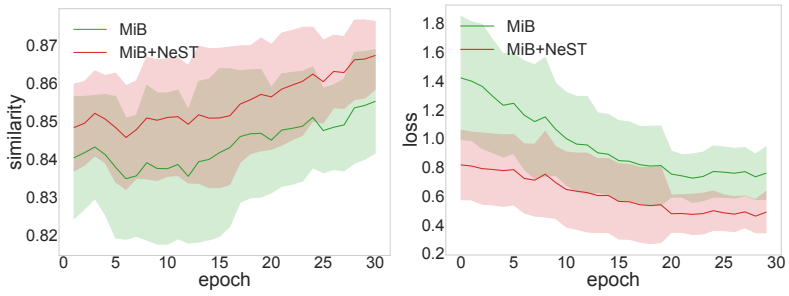
Fig. 3: The mIoU (%) at each step for the setting *15-1* (a) and *10-1* (b).

Table 2: The mIoU (%) of the last step on the ADE20K dataset for four different CISS scenarios. * implies results from the re-implementation of the official code.

Method	Backbone	100-50(2 steps)			100-10(6 steps)			100-5(11 steps)			50-50(3 steps)		
		0-100	101-150	all	0-100	101-150	all	0-100	101-150	all	0-50	51-150	all
Joint [51]	Res101	44.3	28.2	38.9	44.3	28.2	38.9	44.3	28.2	38.9	51.1	33.3	38.9
ILT [35]	Res101	18.3	14.8	17.0	0.1	2.9	1.1	0.1	1.3	0.5	13.6	6.2	9.7
PLOP+UCD [49]	Res101	42.1	15.8	33.3	40.8	15.2	32.3	-	-	-	47.1	24.1	31.8
SPPA [29]	Res101	42.9	19.9	35.2	41.0	12.5	31.5	-	-	-	49.8	23.9	32.5
MiB+AWT [19]	Res101	40.9	24.7	35.6	39.1	21.3	33.2	38.6	16.0	31.1	46.6	26.9	33.5
ALIFE [37]	Res101	42.2	23.1	35.9	41.0	22.8	35.0	-	-	-	49.0	25.7	33.6
GSC [12]	Res101	42.4	19.2	34.8	40.8	17.6	32.6	39.5	11.2	30.2	46.2	26.2	33.0
MiB [4]	Res101	40.5	17.2	32.8	38.2	11.1	29.2	36.0	5.6	25.9	45.6	21.0	29.3
MiB* [4]	Res101	40.5	23.5	34.9	37.8	12.1	29.3	35.8	6.0	25.9	45.9	23.9	31.3
MiB+NeST (Ours)	Res101	40.3	24.6	35.1	40.2	20.6	33.7	39.9	18.0	32.7	45.6	26.8	33.2
PLOP [14]	Res101	41.9	14.9	32.9	40.5	13.6	31.6	39.1	7.8	28.7	48.8	21.0	30.4
PLOP* [14]	Res101	42.2	15.3	33.3	41.0	13.7	32.0	39.6	8.2	29.2	48.5	20.8	30.2
PLOP+NeST (Ours)	Res101	42.2	24.3	36.3	40.9	22.0	34.7	39.3	17.4	32.0	48.7	27.7	34.8
RCIL [51]	Res101	42.3	18.8	34.5	39.3	17.6	32.1	38.5	11.5	29.6	48.3	25.0	32.5
RCIL* [51]	Res101	42.3	16.5	33.8	39.9	15.7	31.9	39.1	12.2	30.2	48.2	23.6	31.9
RCIL+NeST (Ours)	Res101	42.3	22.8	35.8	40.7	19.0	33.5	39.4	15.5	31.5	48.2	27.4	34.4
Joint	Swin-B	43.4	31.9	39.6	43.4	31.9	39.6	43.4	31.9	39.6	50.7	33.9	39.6
MiB*	Swin-B	42.7	26.1	37.2	40.2	15.0	31.8	39.1	8.6	29.0	48.3	26.8	34.1
MiB+NeST (Ours)	Swin-B	42.8	27.8	37.9	41.8	23.8	35.9	40.5	19.9	33.7	49.7	29.3	36.2
PLOP*	Swin-B	43.4	17.1	34.7	41.4	17.7	33.6	39.7	13.6	31.0	50.5	24.1	33.0
PLOP+NeST (Ours)	Swin-B	43.5	26.5	37.9	41.7	24.2	35.9	39.7	18.3	32.6	50.6	28.9	36.2

on the *10-1* setting. In Tab. 1, the results across all settings for old and new classes demonstrate that NeST significantly enhances stability by aligning new classifiers with the existing backbone through the pre-tuning process, thereby mitigating the forgetting of old knowledge. We also report the mIoU of our method and *baselines* at each step. As shown in Fig. 3, The performance of our method surpasses *baselines'* performance during the whole training process.

ADE20K. To further evaluate NeST, we conduct experiments on the more challenging settings of the ADE20K dataset. Results of *100-50*, *100-10*, *100-5* and



(a) feature map similarity **(b)** training loss

Fig. 4: The feature map similarity and training loss on Pascal VOC 2012 *15-1* overlapped setting.

50-50 settings are shown in Tab. 2. NeST outperforms other competing methods. On the most difficult *100-5* setting, NeST improves MiB [4], PLOP [14] and RCIL [51] by 6.8%, 2.8% and 1.3%, which indicates that NeST is also applicable to scenarios with a large number of classes.

Effectiveness of Our Method. It is crucial for pre-tuning new classifiers before the formal training step, not only for learning new classes but also for bridging the *stability gap* [13], which means that the model will undergo a short period of forgetting when the task changes and then gradually recover. This phenomenon is also discovered in CISS scenarios [2,46]. At the beginning of each formal training step, a well-tuned new classifier can make the loss smaller, thus there will also be less impact on other model parameters as the gradient values become smaller. As the model's parameters are inherited from the old model, the old knowledge learned from previous steps can be preserved. We conduct experiments on the *15-1* setting with the *baseline* MiB [4] and MiB+NeST. For a fair and plain comparison, we plot losses and cosine feature similarities by the mean and standard deviation of each epoch at the formal training process of the second step, while in the first step, we use the original MiB for all experiments. As shown in Fig. 4a, the feature similarity of NeST exceeds that of MiB throughout the entire recovery process, demonstrating the ability of NeST to help bridge the stability gap in the model. Fig. 4b shows that during the formal training process, the loss of NeST is lower than the loss of MiB, helping the model converge faster and learn better.

4.3 Ablation Study

Different Classifier Initialization Methods. We compare NeST with other initialization ways. As shown in Tab. 3, random initialization gets the worst performance because of the misalignment in the early stages of training. Initialization by the background classifier can get better performance. Directly tuning the weight of new classifiers initialized by the background classifier before the formal training step can bring a slight increase in performance. Unlike the afore-

mentioned approaches, NeST enables new classifiers to adapt to training data and enhances the performance of the *baseline* by 16.5% on old classes and 4.7% on new classes.

Table 3: Ablation study of different classifier initialization strategies. The performance is reported on Pascal VOC 2012 15-1 overlapped setting with MiB [4].

Method	*0-15*	*16-20*	*all*
Random	43.5	4.2	34.1
Background [4]	45.2	15.7	38.2
Two-Stage	46.0	15.3	38.7
NeST (Ours)	61.7	20.4	51.8

Table 4: Comparison between different transformation matrix initialization methods. The performance is reported on Pascal VOC 2012 15-1 overlapped setting.

Method	*0-15*	*16-20*	*all*
Random Matrix Initialization	53.8	7.5	42.8
NeST (Ours)	61.7	20.4	51.8

Transformation Matrices Initialization. According to Tab. 4, if we use random initialization instead of our designed strategy, the overall performance will still be higher than the performance of *baseline* [4]. However, the mIoU of new classes learned in incremental steps is significantly lower than *baseline*'s performance. It means that if we random initialize importance matrices and projection matrices, it will only help to maintain the stability of the model. The second row in Tab. 4 shows that after initializing importance matrices and projection matrices using our proposed strategy increases the performance of new classes to 20.4%, while also improving the performance of the old classes. The design of initial values for matrices used in the transformation can facilitate the model's learning of new classes. Thus, by initializing matrices with our strategy, NeST can achieve the trade-off between stability and plasticity.

Component-Level Ablation Study. We conduct a component-level ablation study on two kinds of matrices, as shown in Tab. 5. Eliminating the importance matrix means that we set values in the importance matrix to 1 and freeze it, *i.e.*, only learning a combination of old classifiers to get new classifiers. Similarly, eliminating the projection matrix indicates that we average the weighted old classifiers to generate new classifiers. Experimental results demonstrate that with equally treated channels, the performance dramatically drops due to the unawareness of different channels.

Computational Costs. The number of extra parameters can be represented as $n_{new} \times n_{old} \times (d+1) + d + n_{new} + 1$ and n_{old} increases linearly with steps

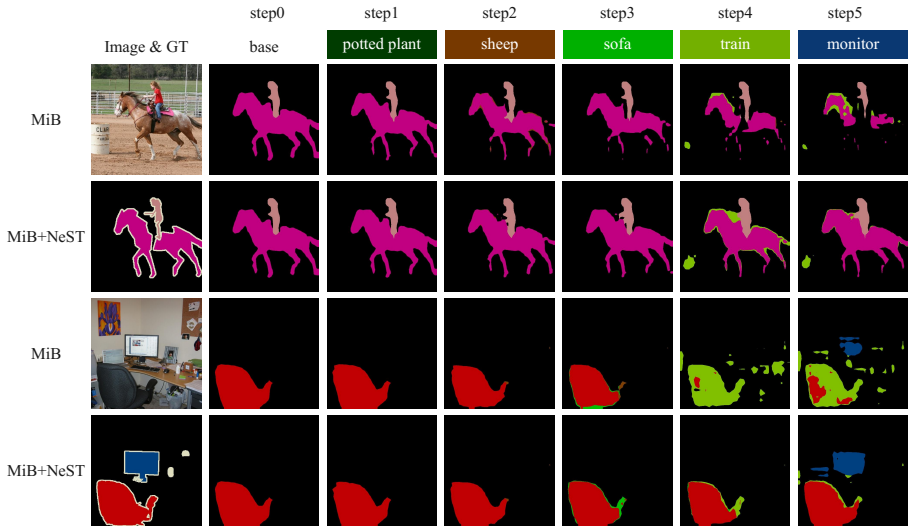


Fig. 5: Qualitative comparisons on Pascal VOC 2012 *15-1* overlapped setting.

Table 5: Component level ablation study on Pascal VOC 2012 *15-1* overlapped setting.

Method	Projection Matrix	Importance Matrix	*0-15*	*16-20*	*all*
baseline			45.2	15.7	38.2
Variants	✓		53.3	15.4	44.3
		✓	59.7	19.2	50.1
	✓	✓	**61.7**	**20.4**	**51.8**

while n_{new} is always fixed. The additional FLOPs (about 15.6K) and parameters (about 5.4K) are negligible during the pre-tuning process of Step 5 on the *15-1* setting. The whole training process of MiB+NeST takes an additional 2.7% (3.7 minutes) of the total training time (2.3 hours). Note that after pre-tuning, extra parameters are removed and the formal training process is the same as MiB.

Discussion on AWT v.s. NeST. AWT [19] utilizes the new training data and old background classifier to generate new classifiers. Here we discuss the differences between AWT and NeST. AWT employs gradient-based attribution to transfer the most relevant weights to new classifiers. However, it treats each new class equally, neglecting the role of other old classifiers and the differences between new classes. Meanwhile, the gradient-based attribution technique introduces a huge memory cost, AWT can not run on RTX 3090 even if the batch size is set to 1. NeST is based on learning, which can better make the generated classifiers' weight align with the backbone. Moreover, as shown in Tab. 6, the memory cost is acceptable, and NeST is a little faster than AWT.

Table 6: The memory costs and additional training time on Pascal VOC 2012 *15-1* overlapped setting.

Method	Memory per GPU	Additional Time
MiB+AWT	$>$ 24GB	6%
MiB+NeST (Ours)	**6.47GB**	**2.7%**

Table 7: The average performance of five different class orders on Pascal VOC 2012 *15-1* overlapped setting.

Method	*A*	*B*	*C*	*D*	*E*	avg
MiB	38.2	28.2	38.5	38.0	52.0	39.0
MiB+NeST (Ours)	51.8	43.4	50.4	60.0	59.8	**51.5**

Robustness of Different Class Orders. In CISS scenarios, the class order is crucial as the learning process of a class may boost or damage the performance of another class. To validate the robustness of NeST, we conduct experiments on the *15-1* setting with five different class orders. As shown in Tab. 7, the performance of NeST is higher than the performance of *baseline*.

4.4 Visualization

In Fig. 5, we show the visualization results of our proposed method based on MiB [4]. Samples are selected from step 0 and from step 5 to show the stability and plasticity of our NeST. In the first two rows, classes *person* and *horse* are learned in step 0, then in the following steps, the *baseline* MiB [4] gradually forget concepts learned in step 0, while NeST helps the model to preserve old knowledge. In the last two rows, class *chair* is learned in step 0 and class *tv/monitor* is learned in step 5. After learning class *train*, MiB has almost forgotten the knowledge of class *chair*, while NeST can give correct predictions for most of the pixels. In the last step, NeST can help the model learn new class *tv/monitor* better than MiB, showing the plasticity of our method.

5 Conclusions

In this work, we propose a simple yet effective new classifier pre-tuning method that can enhance the CISS ability by learning a transformation from old classifiers to new classifiers. We further find a way to initialize matrices by utilizing the information of cross-task class similarities between old classes and new classes, helping the model achieve the stability-plasticity trade-off. Experiments on two datasets show that NeST can significantly improve the performance of *baselines*, and it can be easily applied to other CISS methods. While our approach can lead to huge performance gains, the limitation is that it introduces additional computational overhead. In future work, we will try our method in other continual learning tasks.

Acknowledgments

This work is funded by NSFC (NO. 62206135), Key Program for International Cooperation of Ministry of Science and Technology, China (NO. 2024YFE0100700), Young Elite Scientists Sponsorship Program by CAST (NO. 2023QNRC001), Tianjin Natural Science Foundation (NO. 23JCQNJC01470), and the Fundamental Research Funds for the Central Universities (Nankai University). Computation is supported by the Supercomputing Center of Nankai University.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Eur. Conf. Comput. Vis. pp. 139–154 (2018)
2. Baek, D., Oh, Y., Lee, S., Lee, J., Ham, B.: Decomposed knowledge distillation for class-incremental semantic segmentation. In: Adv. Neural Inform. Process. Syst. (2022)
3. Bang, J., Kim, H., Yoo, Y., Ha, J.W., Choi, J.: Rainbow memory: Continual learning with a memory of diverse samples. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 8218–8227 (2021)
4. Cermelli, F., Mancini, M., Bulo, S.R., Ricci, E., Caputo, B.: Modeling the background for incremental learning in semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 9233–9242 (2020)
5. Cha, S., Yoo, Y., Moon, T., et al.: Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. In: Adv. Neural Inform. Process. Syst. vol. 34, pp. 10919–10930 (2021)
6. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Eur. Conf. Comput. Vis. pp. 532–547 (2018)
7. 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 Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2017)
8. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Eur. Conf. Comput. Vis. pp. 801–818 (2018)
9. Chen, Y.H., Chen, W.Y., Chen, Y.T., Tsai, B.C., Frank Wang, Y.C., Sun, M.: No more discrimination: Cross city adaptation of road scene segmenters. In: Int. Conf. Comput. Vis. pp. 1992–2001 (2017)
10. Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R.: Masked-attention mask transformer for universal image segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1290–1299 (2022)
11. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. In: Adv. Neural Inform. Process. Syst. vol. 34, pp. 17864–17875 (2021)
12. Cong, W., Cong, Y., Dong, J., Sun, G., Ding, H.: Gradient-semantic compensation for incremental semantic segmentation. IEEE Trans. Multimedia (2023)
13. De Lange, M., van de Ven, G., Tuytelaars, T.: Continual evaluation for lifelong learning: Identifying the stability gap. arXiv preprint arXiv:2205.13452 (2022)

14. Douillard, A., Chen, Y., Dapogny, A., Cord, M.: Plop: Learning without forgetting for continual semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. (2021)
15. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 86–102. Springer (2020)
16. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (2012)
17. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: A new multi-scale backbone architecture. IEEE Trans. Pattern Anal. Mach. Intell. **43**(2), 652–662 (2019)
18. Geng, C., Huang, S.j., Chen, S.: Recent advances in open set recognition: A survey. IEEE Trans. Pattern Anal. Mach. Intell. **43**(10), 3614–3631 (2020)
19. Goswami, D., Schuster, R., van de Weijer, J., Stricker, D.: Attribution-aware weight transfer: A warm-start initialization for class-incremental semantic segmentation. In: WACV. pp. 3195–3204 (2023)
20. Grossberg, S.T.: Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control, vol. 70. Springer Science & Business Media (2012)
21. Guo, M.H., Lu, C.Z., Hou, Q., Liu, Z., Cheng, M.M., Hu, S.M.: Segnext: Rethinking convolutional attention design for semantic segmentation. In: Adv. Neural Inform. Process. Syst. vol. 35, pp. 1140–1156 (2022)
22. Hadsell, R., Rao, D., Rusu, A.A., Pascanu, R.: Embracing change: Continual learning in deep neural networks. Trends in cognitive sciences **24**(12), 1028–1040 (2020)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conf. Comput. Vis. Pattern Recog. (2016)
24. Hu, Z., Li, Y., Lyu, J., Gao, D., Vasconcelos, N.: Dense network expansion for class incremental learning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 11858–11867 (2023)
25. Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. In: Adv. Neural Inform. Process. Syst. vol. 32 (2019)
26. Kim, D., Han, B.: On the stability-plasticity dilemma of class-incremental learning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 20196–20204 (2023)
27. 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)
28. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. **40**(12), 2935–2947 (2017)
29. Lin, Z., Wang, Z., Zhang, Y.: Continual semantic segmentation via structure preserving and projected feature alignment. In: Eur. Conf. Comput. Vis. pp. 345–361. Springer (2022)
30. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Int. Conf. Comput. Vis. pp. 10012–10022 (2021)
31. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3431–3440 (2015)

32. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3431–3440 (2015)
33. Maracani, A., Michieli, U., Toldo, M., Zanuttigh, P.: Recall: Replay-based continual learning in semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 7026–7035 (2021)
34. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
35. Michieli, U., Zanuttigh, P.: Incremental learning techniques for semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 0–0 (2019)
36. Michieli, U., Zanuttigh, P.: Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1114–1124 (2021)
37. Oh, Y., Baek, D., Ham, B.: Alife: Adaptive logit regularizer and feature replay for incremental semantic segmentation. In: Adv. Neural Inform. Process. Syst. vol. 35, pp. 14516–14528 (2022)
38. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Int. Conf. Comput. Vis. pp. 12179–12188 (2021)
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. pp. 234–241. Springer (2015)
40. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Adv. Neural Inform. Process. Syst. vol. 30 (2017)
41. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: Int. Conf. Comput. Vis. pp. 7262–7272 (2021)
42. Verwimp, E., De Lange, M., Tuytelaars, T.: Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In: Int. Conf. Comput. Vis. pp. 9385–9394 (2021)
43. Wang, E., Peng, Z., Xie, Z., Yang, F., Liu, X., Cheng, M.M.: Unlocking the multimodal potential of clip for generalized category discovery (2024), `https://arxiv.org/abs/2403.09974`
44. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **43**(10), 3349–3364 (2020)
45. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 374–382 (2019)
46. Xiao, J.W., Zhang, C.B., Feng, J., Liu, X., van de Weijer, J., Cheng, M.M.: Endpoints weight fusion for class incremental semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 7204–7213 (2023)
47. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. Adv. Neural Inform. Process. Syst. **34**, 12077–12090 (2021)
48. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3014–3023 (2021)
49. Yang, G., Fini, E., Xu, D., Rota, P., Ding, M., Nabi, M., Alameda-Pineda, X., Ricci, E.: Uncertainty-aware contrastive distillation for incremental semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **45**(2), 2567–2581 (2022)
50. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)

51. Zhang, C.B., Xiao, J.W., Liu, X., Chen, Y.C., Cheng, M.M.: Representation compensation networks for continual semantic segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 7053–7064 (2022)
52. Zhang, Z., Zhang, X., Peng, C., Xue, X., Sun, J.: Exfuse: Enhancing feature fusion for semantic segmentation. In: Eur. Conf. Comput. Vis. pp. 269–284 (2018)
53. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 6881–6890 (2021)
54. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 633–641 (2017)
55. Zhu, F., Zhang, X.Y., Wang, C., Yin, F., Liu, C.L.: Prototype augmentation and self-supervision for incremental learning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 5871–5880 (2021)