TOWARDS REALISTIC INCREMENTAL SCENARIO IN CLASS INCREMENTAL SEMANTIC SEGMENTATION

Jihwan Kwak[1], **Sungmin Cha**[2] **& Taesup Moon**[1,3*]
[1]Department of Electrical and Computer Engineering, Seoul National University
[2]Computer Science Department at the Courant Institute of Mathematical Sciences, New York University
[3]ASRI / INMC / IPAI / AIIS, Seoul National University
kkwakzi@snu.ac.kr, sungmin.cha@nyu.edu, tsmoon@snu.ac.kr

ABSTRACT

This paper addresses the unrealistic aspect of the *overlapped* scenario, a commonly adopted incremental learning scenario in Class Incremental Semantic Segmentation (CISS). We highlight that the *overlapped* scenario allows the **same** image to reappear in future tasks with different pixel labels, creating unwanted advantage or disadvantage to widely used techniques in CISS, such as pseudo-labeling and data replay from the exemplar memory. Our experiments show that methods trained and evaluated under the *overlapped* scenario can produce biased results, potentially affecting algorithm adoption in practical applications. To mitigate this, we propose a practical scenario called ***partitioned***, where the dataset is first divided into distinct subsets representing each class, and then these subsets are assigned to corresponding tasks. This efficiently addresses the data reappearance artifact while meeting other requirements of CISS scenario, such as capturing the background shifts. Additionally, we identify and resolve the code implementation issues related to replaying data from the exemplar memory, previously overlooked in other works. Lastly, we introduce a simple yet competitive memory-based baseline, MiB-AugM, that handles background shifts in the exemplar memory. This baseline achieves state-of-the-art results across multiple tasks involving learning many new classes. Codes are available at https://github.com/jihwankwak/CISS_partitioned.

1 INTRODUCTION

Due to increasing industrial demands, recent studies have placed significant emphasis on understanding the behavior of models when learning from non-stationary streams of data. One area of particular interest is the Class Incremental Learning (CIL) problem, where a model learns new classes from incrementally arriving training data. The primary challenge in CIL lies in addressing the plasticity-stability dilemma (Carpenter & Grossberg, 1987; Mermillod et al., 2013), whereby models must learn new concepts while mitigating the risk of *catastrophic forgetting* (McCloskey & Cohen, 1989), a phenomenon of inadvertently forgetting previously acquired knowledge when learning new concepts. To date, research in this field has expanded beyond methodological approaches (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Yoon et al., 2017) to include discussions on practical scenarios (Wu et al., 2019; Tao et al., 2020; He et al., 2020) that closely mirror real-world learning processes.

Motivated by its applications to autonomous driving and robotics, CIL has extended its reach to semantic segmentation tasks, known as Class Incremental Semantic Segmentation (CISS). In CISS, the model additionally encounters the challenge of *background shift* (Cermelli et al., 2020), which refers to a semantic drift of the background class between tasks. Specifically, since all pixels whose ground truth class does not correspond to the current task classes are annotated as background, objects from previous and future classes may be mislabeled as background. This exacerbates the forgetting of previous classes and hinders the knowledge acquisition of new classes.

To apply and evaluate CISS methods, two incremental scenarios, *disjoint* and *overlapped* were initially introduced by Cermelli et al. (2020). Since the *disjoint* scenario failed to capture the background shift of unseen classes, most studies (Douillard et al., 2021; Baek et al., 2022; Zhang et al., 2022a; 2023) have focused on the *overlapped* scenario. Notably, recent methods have demonstrated state-of-the-art performance by either freezing parameters (Cha et al., 2021; Zhang et al., 2022b) or applying strong regularization (Baek et al., 2022; Zhang et al., 2023). However, despite the active discussion on methodologies, there is a lack of awareness about the limitations of the *overlapped* scenario, which have been overlooked until now.

* Corresponding author

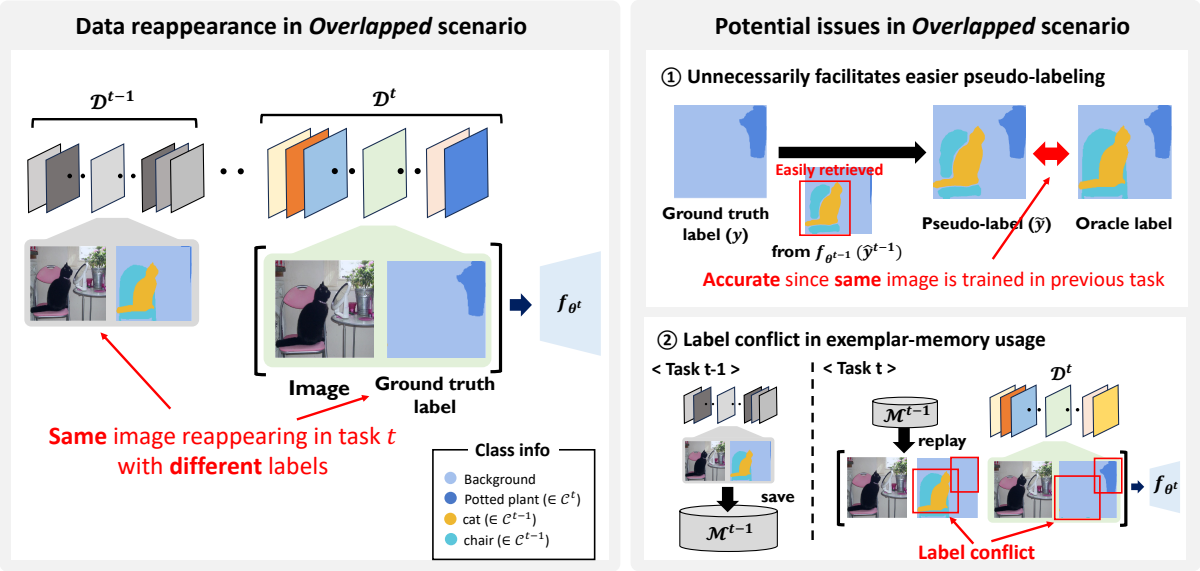


Figure 1: (**Left**) Illustration of data reappearance issue in the *overlapped* scenario. At task t, an image seen at task $t-1$ is given with different labels, which is far from a practical incremental learning scenario. (**Right**) Illustration of potential problems caused by the reappearance issue in the *overlapped* scenario.

We focus on addressing the unrealistic aspect of the widely adopted *overlapped* scenario. As shown in Figure 1 (left), in the *overlapped*, the **same** image can reappear in future tasks with **different** pixel labels. For example, an image seen in task $t-1$ labeled with `cat` and `chair` class can reappear at task t labeled only with `potted plant` class. We term this as *overlapping* data and show that this artifact can lead to unwanted advantage or disadvantage for certain techniques, resulting in biased outcomes for certain algorithms. Firstly, pseudo-labeling classes from the previous task becomes relatively easy for *overlapping* data since the model has already seen the image from previous tasks. Secondly, if the *overlapping* data saved in memory during the previous tasks are replayed with its saved labels, an image may be trained with two distinct labels, causing label conflicts. For example, in task t, the model learns the potted plant object with two labels: `potted plant` class from current data and `background` class from data replayed from the exemplar memory. Therefore, in the *overlapped*, models unfairly benefit from learning pseudo-labels that are close to the oracle labels and encounter unnecessary label conflict when exemplar memory is utilized.

As an alternative, we propose a practical scenario, dubbed as *partitioned*, that facilitates accurate and objective evaluation of CISS algorithms. This involves partitioning the dataset into mutually distinct subsets, each representing a specific class, and assigning each subset to its corresponding task. This methodology eliminates unnecessary artifact of the *overlapped* scenario while satisfying the preconditions of the CISS scenario, such as capturing the background shifts of both previous and unseen classes. Furthermore, we address the overlooked code implementation issues in the exemplar memory, which have been ignored in prior studies. Prior code frameworks rather provide labels of the current task to the data replayed from the memory.

Lastly, motivated by the issues above, we introduce an efficient memory-based baseline, named MiB-AugM, that effectively handles the background shift of the current task class in the exemplar memory. Experiments with reproduced baselines show that our method outperforms state-of-the-art methods across several tasks involving the learning of numerous new classes at every task.

2 Related Works

2.1 Class Incremental Learning (CIL)

Methodologies Research on CIL has primarily focused on image classification tasks, with methods falling into three main categories: 1) *Regularization*-based (Kirkpatrick et al., 2017; Chaudhry et al., 2018; Ahn et al., 2019; Jung et al., 2020), 2) *Rehearsal*-based (Lopez-Paz & Ranzato, 2017; Shin et al., 2017; Prabhu et al., 2020), and 3) *Architecture*-based (Rusu et al., 2016; Yoon et al., 2017; Hung et al., 2019; Yan et al., 2021a) solutions. Among them, approaches

that combine *regularization* through knowledge distillation (KD) (Hinton et al., 2015) and *rehearsal* of exemplar memory (Rolnick et al., 2019) have achieved state-of-the-art performance (Li & Hoiem, 2017; Buzzega et al., 2020). Consequently, this integration has spurred a line of research dedicated to addressing by-product problems such as prediction bias (Wu et al., 2019; Ahn et al., 2021).

Incremental scenarios Since the research demands of continual learning started from the non-stationary property of the real-world, several works have claimed the necessity of building up a practical incremental scenario that resembles the learning process of the real-world. For example, Wu et al. (2019); Ahn et al. (2021) suggested an evaluation on *large* scale CIL scenarios and Hou et al. (2019); Douillard et al. (2020) proposed a scenario that includes a *large base task* where the model has a chance to initially learn knowledge from many classes.

With growing concerns about realistic scenarios that align with industrial demands, CIL studies have stretched out its application to settings with additional data constraints such as *few-shot* (Tao et al., 2020; Yang et al., 2023) or *online* (He et al., 2020; Lin et al., 2023). Similarly, a new body of discussions on *online*-CIL regarding practical scenarios (Koh et al., 2022; Chrysakis & Moens, 2023) or constraints (Ghunaim et al., 2023) has been introduced.

2.2 Class incremental semantic segmentation (CISS)

Methodologies Inspired by recent works of CIL, initial works in CISS (Cermelli et al., 2020; Douillard et al., 2021; Michieli & Zanuttigh, 2021) took KD-based regularization as a general approach. MiB (Cermelli et al., 2020) proposed a novel KD-based regularization to address the semantic drift of background label. PLOP (Douillard et al., 2021) brought the idea of Douillard et al. (2020) and utilized the feature-wise KD. On the other hand, few works (Yan et al., 2021b; Zhu et al., 2023) focused on employing the exemplar memory. Yan et al. (2021b) proposed to use a class-balanced memory but only focused on *online* setting. Zhu et al. (2023) proposed a memory sampling mechanism to ensure diversity among samples but has a reproduction issue[1]. Recently, SSUL (Cha et al., 2021) which introduced a method to freeze the feature extractor have demonstrated outperforming performance. As a result, the recent state-of-the-art CISS methods have been focused on either freezing (Zhang et al., 2022b) or regularizing the extractor with hard constraint (Zhang et al., 2023; Chen et al., 2023), while considering memory usage as an extra factor to enhance performance slightly.

Incremental scenarios Cermelli et al. (2020) first established the learning scenarios of CISS, *overlapped* and *disjoint* on Pascal VOC 2012 (Everingham et al., 2010) and ADE 20K (Zhou et al., 2017). After many works (Douillard et al., 2021; Baek et al., 2022) pointed out the impractical assumption of *disjoint* where the existence of semantic shift of unseen class is excluded, current CISS research is actively being explored in *overlapped* scenario. However, in contrast to CIL, where both practical scenarios and methodologies have been actively discussed, there has been no further discussion on the learning scenarios in CISS until now.

Building upon insights acquired from previous studies of CIL in classification, regularization-based methods have achieved impressive performance in CISS. However, previous studies have often overlooked the practical considerations in the learning scenarios. In this work, we point out an overlooked issue in the learning scenario (*e.g., overlapped*) and propose a new scenario and a competitive baseline method that effectively leverages the exemplar memory.

3 Problems of overlapped scenario and a proposed realistic scenario

3.1 Notation and problem setting

Class Incremental Learning (CIL) for image classification operates under the assumption that pairs of input data and its corresponding label for *new* classes are accessible for training a model in each incremental *task*. At each task t, the model is trained on a new dataset, $\mathcal{D}^t$ annotated with a set of new classes $\mathcal{C}^t$. In the evaluation phase, the model is expected to distinguish between all the seen classes up to task t, denoted as $\mathcal{C}^{0:t} = \mathcal{C}^0 \cup \cdots \cup \mathcal{C}^t$. Each task is organized with disjoint classes, meaning there is no overlap of classes between the tasks, denoted as $\mathcal{C}^i \cap \mathcal{C}^j = \emptyset$ for $\forall i, j$. Note that the model initially acquires knowledge of a large number of classes at the base task (task 0), and gradually learns the remaining classes in subsequent tasks (task 1 to T).

Class Incremental Semantic Segmentation (CISS) considers incremental learning in semantic segmentation, involving the pixel-level prediction of labels. At task t, an input image of size N[2], $x \in \mathbb{R}^{N \times 3}$, is paired with its corresponding ground-truth pixel labels $y \in \mathbb{R}^N$, denoted as $(x, y) \sim \mathcal{D}^t$. There exists at least one pixel that is annotated as one of

[1] Code implementation not available

[2] For notational convenience, the image, originally considered as a 2-D array with height (H) and width (W), is now treated as a 1-D array with size N, where $N = H * W$.

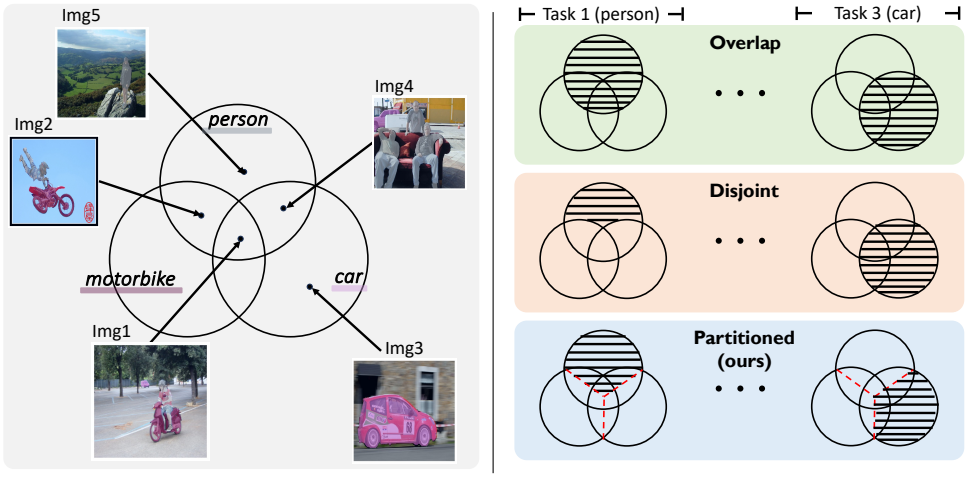


Figure 2: (**Left**) A Venn diagram illustrating the relationship between ground truth class for each datum. Each datum is assigned to a class set based on the object it contains. (**Right**) Comparison of $\mathcal{D}^t$ (highlighted with black line) for each scenario. Table 5 in the Appendix provides a detailed summary of the assigned labels for each image in every incremental task.

the classes in $\mathcal{C}^t$ and pixels not belonging to $\mathcal{C}^t$ are labeled as the background label c_{bg}. Therefore, each task dataset $\mathcal{D}^t$ in CISS contains ground truth labels corresponding to $\mathcal{C}^t \cup c_{bg}$. Note that objects from the *past* or *future* task's classes may be labeled as the background label in the current dataset $\mathcal{D}^t$, even though these same objects have their actual labels in other tasks. This label shift between tasks, known as *background shift*, presents a significant challenge in CISS. Previous works have considered two scenarios of CISS, *disjoint* and *overlapped*, which will be discussed in Section 3.2. Following previous works (Cha et al., 2021; Baek et al., 2022), we also introduce an exemplar memory $\mathcal{M}^{t-1}$, which stores a subset of data from $\mathcal{D}^{0:t-1}$ and is used for task t. The size of memory remains consistent across the tasks, denoted as $|\mathcal{M}^t| = M \ \forall t$.

At task t, our model, parameterized by θ^t, consists of a feature extractor and a classifier, denoted as $f_{\theta^t}(\cdot)$. Given an input x, the model produces a consolidated score of seen classes for each pixel $z_i^t = f_{\theta^t}(x)_i \in \mathbb{R}^{|\mathcal{C}^{0:t} \cup \{c_{bg}\}|}$. The class prediction for each pixel is obtained by selecting the class with the highest score:

$$\hat{y}_i^t = \underset{c \in \mathcal{C}^{0:t} \cup \{c_{bg}\}}{\arg\max} \ z_{i,c}^t, \tag{1}$$

where $z_{i,c}^t$ represents the output logit (before softmax) of the i^{th} pixel to the c^{th} class in $\mathcal{C}^{0:t} \cup \{c_{bg}\}$.

3.2 Incremental scenarios in CISS: disjoint and overlapped

Two incremental scenarios have been recognized as key scenarios of CISS (Cermelli et al., 2020): *disjoint* and *overlapped*. The dataset construction for each scenario can be compared using a set diagram, as depicted in Figure 2.

Suppose we are constructing a CISS scenario using a dataset that has three object classes (`motorbike`, `car`, and `person`), as annotated in Figure 2. First, we define three sets representing each class, where each element in a set represents a datum (*i.e.,* a pair of (x, y)). Each datum is then assigned to one or more class sets based on its *oracle* pixel labels. For example, the figure illustrates that `Img1` and `Img2` have objects whose pixels are labeled as {*car, motorbike, person*} and {*motorbike, person*}, respectively. In this case, both `Img1` and `Img2` are elements of `motorbike` class set. If an incremental scenario is defined to sequentially introduce the `person`, `motorbike`, and `car` classes as ground truth for each task, the Venn diagram of the three sets can be used to demonstrate the dataset given at each task in CISS. Namely, the data points covered by black lines indicate the dataset utilized for each task.

As illustrated in the right figure, *disjoint* ensures the separation of the dataset for each task but necessitates prior knowledge of unseen classes to achieve this separation (Cermelli et al., 2020). Moreover, *disjoint* does not capture the background shift of classes from future tasks, leading to the widespread adoption of the *overlapped* scenario in recent works (Cermelli et al., 2020; Douillard et al., 2020; Cha et al., 2021). However, we argue that the *overlapped* scenario has its own set of challenges.

3.3 UNWANTED ADVANTAGE AND DISADVANTAGE IN THE OVERLAPPED SCENARIO

The problem arises from the fact that, in *overlapped*, previously seen images may be reintroduced in future tasks with different pixel labels, as illustrated in Figure 1. For example, in task $t-1$, the pixels corresponding to the cat and chair object are labeled with their respective classes, while all other pixels are annotated as the `background` class. However, in task t, the **same** image may reappear with being labeled only with the `potted plant` class, leaving the remaining pixels, including the cat and chair object, as the `background` class. This phenomenon in *overlapped* is far from a practical incremental learning scenario, and we term this problematic data as *overlapping* data. In this section, we will show that overlapping data can create artificial advantage or disadvantage for techniques commonly used in current CISS, potentially resulting in misleading conclusions and impacting the development and adoption of algorithms in practical applications.

Unwanted advantage of pseudo-labeling with previously learned model To mitigate the issues arising from background shift caused by classes from the previous task, many recent studies (Douillard et al., 2021; Cha et al., 2021; Zhang et al., 2023; Chen et al., 2023) employ pseudo-labeling for the background region using predictions from the previously learned model. Formally, the pseudo-label $\tilde{y}_i$ can be defined as follows:[3]

$$\tilde{y}_i = \begin{cases} y_i & \text{if } y_i \in \mathcal{C}^t \\ \hat{y}_i^{t-1} & \text{if } (y_i = c_{bg}) \wedge (s_i^{t-1} > \tau) \\ c_{bg} & \text{else,} \end{cases} \tag{2}$$

where $s_i^{t-1} = \max_{c \in \mathcal{C}^{0:t-1} \cup \{c_{bg}\}} \frac{z_{i,c}^{t-1}}{\sum_k z_{i,k}^{t-1}}$ represents the output probability (after softmax) for $\hat{y}_i^{t-1} \in \mathcal{C}^{0:t-1} \cup \{c_{bg}\}$ and τ indicates the threshold for pseudo-labeling, respectively. An issue arises when the pseudo-labeling with the previously learned model is applied to overlapping data which is a pair of previously **seen** image and pixel labels annotated based on current task classes and the background class. The old class objects in the image were learned in previous tasks with the **same** image and corresponding label. Therefore, when pseudo-labeling this part with the past model, it produces accurate pseudo-labels, as shown in Figure 1 (right). This implies that overlapping data unnecessarily facilitates easier pseudo-labeling of classes from past task.

To showcase the aforementioned phenomenon, we conducted experiments with the Pseudo-labeling Retrieval Rate (PRR) metric, defined as follows:

$$PRR(\hat{\mathcal{D}}, f_{\theta^{t-1}}) = \frac{1}{|\hat{\mathcal{D}}|} \sum_{(x, y_{oracle}) \sim \hat{D}} \text{mIoU}_{\mathcal{C}^{0:t-1} \cup \{c_{bg}\}}(y_{oracle}, \tilde{y}^{t-1}) \tag{3}$$

where $\hat{\mathcal{D}}$ indicates the dataset under evaluation that includes image x with y_{oracle} which indicates an oracle pixel labels containing all foreground and background classes. Also, $\tilde{y}^{t-1}$ denotes the pseudo-label of x generated by $f_{\theta^{t-1}}$. To assess the retrieval rate of previous classes, Intersection-over-Unions (IoU) between $\tilde{y}^{t-1}$ and y_{oracle} is averaged among previous classes, denoted as $\text{mIoU}_{\mathcal{C}^{0:t-1} \cup \{c_{bg}\}}(\cdot, \cdot)$. For an in-depth description of the PRR metric, please refer to Figure 5 in the Appendix.

For verification, we modify the overlapped scenario by randomly dividing the overlapping data of two consecutive tasks, $\mathcal{D}^0 \cap \mathcal{D}^1$, into two parts of the same size. One part will be seen at the previous task, task 0, denoted by $\mathcal{D}_{seen}$, and the other will not be seen, denoted by $\mathcal{D}_{unseen}$. As a result, at task 0, the model is trained with $\mathcal{D}^0 \setminus \mathcal{D}^{unseen}$. After training the model f_{θ^0}, we evaluate the PRR results of $\mathcal{D}_{seen}$ and $\mathcal{D}_{unseen}$ by comparing IoU between oracle labels and pseudo-labels generated from f_{θ^0}.

Table 1: Pseudo-labeling Retrieval Rate (PRR) on $\mathcal{D}_{seen}$ and $\mathcal{D}_{unseen}$. The figure with the downarrow(↓) inside the parentheses indicates the difference between the two.

Task	***PRR*** $(\cdot, f_{\theta^0})$	
	$\mathcal{D}_{seen}$	$\mathcal{D}_{unseen}$
15-1	86.81	**66.91 (19.90 ↓)**
15-5	84.15	**60.27 (23.88 ↓)**
10-1	86.82	**66.66 (20.16 ↓)**
10-5	87.36	**68.40 (18.96 ↓)**

Table 1 demonstrates PRR results across two different incremental tasks. Task 15-1 indicates that the model initially learns 15 classes and incrementally learns 1 class at every task. Since the evaluation is done on task 1, PRR is evaluated after learning 15 classes at task 0. For robustness, we report averaged results over 3 random overlapping data splits, $\mathcal{D}_{seen}$ and $\mathcal{D}_{unseen}$ construction, and 3 different class-ordering. For further details, please refer to Section A.2

In Table 1, compared to PRR results of $\mathcal{D}_{seen}$, PRR results of $\mathcal{D}_{unseen}$ shows a notable decline in all tasks. For example, in 15-1 task, PRR of $\mathcal{D}_{unseen}$ shows a 19.90 decreased result compared to PRR of $\mathcal{D}_{seen}$. These experimental

[3]Since each study follows different rules for pseudo-labeling, we present the simplest format. Note that the core idea of pseudo-labeling is to use the predicted labels with a high prediction score, which is steered by hyper-parameter τ.

findings illustrate that the retrieval of previous labels is relatively straightforward when it is done on already seen overlapping data, which can cause unnecessary advantage to pseudo-labeling techniques.

Unwanted disadvantage of using exemplar memory Recent studies (Cha et al., 2021; Baek et al., 2022; Zhu et al., 2023; Chen et al., 2023) also actively utilize exemplar memory to store and replay past task data to alleviate forgetting. However, when overlapping data is saved in the memory, it can rather cause a *label conflict* as illustrated in Figure 1 (right). For example, if the image with cat and chair objects labeled accordingly and the potted plant object marked as `background` is saved in task $t-1$ and replayed in task t, the model encounters the **same** image twice with disparate labels. For the potted plant object, data from the current task has the ground truth pixel label correctly labeled with `potted plant` class but the pixel labels from the memory are annotated as `background` class. This implies that overlapping data can worsen learning new concepts and remembering old knowledge when saved and replayed by the exemplar memory.

To observe the actual impact of the label conflict, we construct a following ablation experiment on the overlapping data in the exemplar memory. After training is done on the base task, class-balanced memory of size M is constructed with $\mathcal{D}^0$ and is used to fine-tune the model along with current task data $\mathcal{D}^1$. Since overlapping data between $\mathcal{D}^0$ and $\mathcal{D}^1$ may exist in the exemplar memory, we denote this memory as *overlapping memory* and report its average ratio. For comparison, we also construct *non-overlapping memory* where overlapping data in the *overlapping memory* is replaced with non-overlapping data in D^0. The model is expected to encounter label conflict when fine-tuning is done on $\mathcal{D}^1$ and *overlapping memory* in contrast to the use of *non-overlapping memory*. After training is done on the above two settings respectively, we compare the test mIoU to compare the impact of label conflict. For further implementation details, please refer to Section A.3.

Table 2 shows the test mIoU of models fine-tuned on $\mathcal{D}^1$ with each memory. Test mIoU of the model increases in all incremental tasks when *overlapping memory* is replaced with *non-overlapping memory*. This increase is further amplified when overlapping ratio gets higher, which implies that the overlapping data in the memory results in label conflict in the *overlapped* scenario.

Through the above experiments, we have shown that the unrealistic artifact of the *overlapped* scenario, namely overlapping data, is not negligible since it may lead to biased results toward several techniques. Therefore, in the following section, we will introduce a new scenario that handles the overlapping issue while satisfying other conditions of CISS scenario.

Table 2: Test mIoU of models fine-tuned on $\mathcal{D}^1$ with each memory. Other than exemplar memory, we fix all the other training implementations. To reduce the randomness caused by memory construction, we report averaged results of models trained under 3 different class-balanced memory.

Incremental Task		***Overlapping ratio in memory***	***Test overall mIoU***	
Task	Class order		Overlapping memory	Non-overlapping memory
15-1	type a	3%	50.02	50.15 (**0.13** ↑)
	type b	17%	24.26	27.02 (**2.56** ↑)
15-5	type a	9%	50.30	51.85 (**1.55** ↑)
	type b	35%	35.13	37.41 (**2.28** ↑)
10-1	type a	1%	71.98	72.27 (**0.29** ↑)
	type b	36%	25.19	27.35 (**2.16** ↑)
10-5	type a	16%	43.94	45.38 (**1.44** ↑)
	type b	46%	36.21	39.31 (**3.10** ↑)

3.4 Proposed scenario: Partitioned

The issue of *overlapped* primarily arises from overlapping data that provide different labels across various tasks. Therefore, we reconsider the disjointness property in CISS scenario by suggesting a scenario that meets the core requirement of *disjoint* and *overlapped*: 1) capturing background shifts of both previous and unseen classes 2) disjointness for eliminating overlapping data. our proposed scenario can be summarized in two steps: 1) partitioning the dataset into distinct subsets representing each class and 2) assigning each class subset to a corresponding task dataset.

Figure 3 demonstrates the process of constructing the proposed ***partitioned*** scenario. First, for each data (image-pixel labels pair), we assign a class for partitioning based on the oracle pixel labels. If the oracle pixel labels consist of at least two classes, we randomly select one class. Experimental results in the later section show that baseline results are robust to this random selection. Second, we partition the whole dataset ($\mathcal{D}$) into distinct subsets based on the assigned class for partitioning. Third, following the required task information (*e.g.,*

Table 3: Comparison of scenarios including *partitioned* in two criteria towards realistic CISS scenario.

Incremental scenario	Removal of overlapping data	Capturing background shifts
disjoint	✓	✗
overlapped	✗	✓
partitioned (ours)	✓	✓

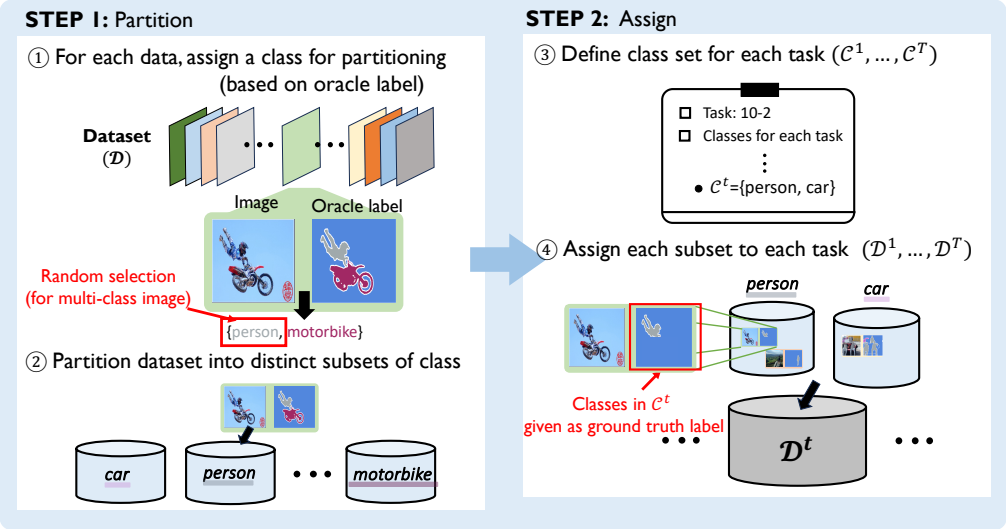


Figure 3: Illustration of dataset construction steps for *partitioned* scenario.

task 10-2), we define the newly learned class set for each task ($\mathcal{C}^0, \ldots, \mathcal{C}^T$). Finally, we assign each subset to the corresponding task dataset ($\mathcal{D}^0, \ldots, \mathcal{D}^T$).

As summarized in Table 3, our proposed *partitioned* 1) guarantees disjointness between task datasets which eliminates overlapping data and 2) captures the background shift of both previous and unseen classes. Figure 2 (right) also visualizes the dataset for each task in the Venn diagram for comparing *partitioned* with *disjoint* and *overlapped*. Experimental results on *partitioned* with reproduced baselines are reported in Section 4.

3.5 New baseline for exemplar memory replay

Since the issue of overlapping data is mitigated in our proposed scenario, it is natural to expect an effective usage of memory in *partitioned*. Therefore, we propose a simple yet competitive baseline that integrates MiB (Cermelli et al., 2020) with an extra loss function, tailored for the case of using exemplar memory.

Following notations in MiB (Cermelli et al., 2020), the overall loss function of our method at task t is defined as:

$$\mathcal{L}(\theta^t) = \underbrace{\frac{1}{|\mathcal{D}^t|}\sum_{(x,y)\in\mathcal{D}^t}\mathcal{L}_{unce}(y, f_{\theta^t}(x)) + \frac{\lambda}{|\mathcal{D}^t \cup \mathcal{M}^{t-1}|}\sum_{(x,y)\in\mathcal{D}^t\cup\mathcal{M}^{t-1}}\mathcal{L}_{unkd}(f_{\theta^{t-1}}(x), f_{\theta^t}(x))}_{\text{From MiB (Cermelli et al., 2020)}} + \underbrace{\frac{1}{|\mathcal{M}^{t-1}|}\sum_{(x,y)\in\mathcal{M}^{t-1}}\mathcal{L}_{mem}(y, f_{\theta^t}(x))}_{\text{Our proposed memory loss}} \tag{4}$$

where $\mathcal{L}_{unce}(\cdot)$ and $\mathcal{L}_{unkd}(\cdot)$ are the loss functions employed in MiB[4]. For the specific formulation of $\mathcal{L}_{unce}(\cdot)$ and $\mathcal{L}_{unkd}(\cdot)$ in our notation, please refer to Section A.5. Regarding $\mathcal{L}_{mem}(\cdot)$, we suggest utilizing cross-entropy loss with the augmented predictions $\dot{p}^t_{i,c}$ as follows:

$$\mathcal{L}_{mem}(y, f_{\theta^t}(x)) = -\frac{1}{N}\sum_{i=1}^{N}\log \dot{p}^t_{i,y_i}. \tag{5}$$

[4]The original paper utilizes the terminologies $\mathcal{L}_{ce}(\cdot)$ and $\mathcal{L}_{kd}(\cdot)$ to represent loss functions. However, to avoid potential confusion with conventional cross-entropy and knowledge distillation losses, we choose to adjust the terminology.

Here, $\dot{p}^t_{i,yi}$ represents the augmented output probability (after softmax) for the ground truth label of a i^{th} pixel. Moreover, the augmented prediction $\dot{p}^t_{i,c}$ can be defined as follows:

$$\dot{p}^t_{i,c} = \begin{cases} p^t_{i,c} & \text{if } c \neq c_{bg} \\ p^t_{i,c_{bg}} + \sum_{k \in \mathcal{C}^t} p^t_{i,k} & \text{if } c = c_{bg}, \end{cases} \tag{6}$$

where $p^t_{i,c} = \exp^{z^t_{i,c}} / \sum_{k \in \mathcal{C}^{0:t} \cup \{c_{bg}\}} \exp^{z^t_{i,k}}$ is the output probability (after softmax) for class c using f_{θ^t}.

Note that the motivation behind employing Equation (5) stems from the background shift in $\mathcal{M}^{t-1}$. For example, consider data stored in the exemplar memory containing objects belonging to the class of the current task. Since this data only has classes from past tasks as a ground truth label, objects of the current task are annotated as `background`. This background shift of the new class confuses the acquisition of knowledge of classes from new tasks. To mitigate this conflict, we adopt the intuition of Zhang et al. (2022b), wherein both background and unseen classes are treated as the same class. Additionally, inspired by the prediction augmentation technique discussed in Cermelli et al. (2020), we aggregate the prediction values of c_{bg} and $c \in \mathcal{C}^t$, specifically in the case of $c = c_{bg}$ in Equation (6). Note that this allows for a positive update of prediction scores for both background and new classes when the model is trained with pixels labeled as `background`, thereby alleviating the label conflict of the data in $\mathcal{M}^{t-1}$.

3.6 Implementation issues in previous CISS studies

In addition to the issues discussed in the previous sections, we will highlight a certain overlooked aspect of code implementation errors in CISS studies. It is worth noting that the labeling implementation for data replayed from the exemplar memory is incorrect in recent studies (Cha et al., 2021; Baek et al., 2022; Zhang et al., 2022b; 2023). Upon examination of their official code repositories, it becomes evident that the data from the exemplar memory **do not provide** ground truth labels for previous objects; instead, they only assign ground truth labels based on $\mathcal{C}^t$ or c_{bg} (refer to Figure 6 in the Appendix). We would like to emphasize the importance of accurate implementation as such discrepancies in implementation can lead to inherent experimental biases. To address this, we rectified the error in the implementation and conducted all experiments with the corrected version, of which codes are at `https://github.com/jihwankwak/CISS_partitioned`.

We believe that addressing these implementation corrections and providing reproduced baselines constitutes a valuable contribution of our work, which can benefit future researchers in CISS studies.

4 Experimental results

4.1 Experimental setups

Dataset and protocols We evaluated our methods based on two incremental scenarios, namely *overlapped* and *partitioned* (ours), using Pascal VOC 2012 (Everingham et al., 2010) dataset. To assess the overall performance, we conducted evaluations across various tasks (*e.g.,* 15-1 task) with differing characteristics (*e.g.,* large/small base tasks). Note that the total number of training data for all the tasks is different between *overlapped* and *partitioned* due to disjointness property of the *partitioned*. Detailed data configurations for two scenarios are provided in Table 8.

Evaluation metrics We utilize the mean Intersection-over-Union (mIoU) as our evaluation metric, which represents the averaged IoU over defined classes. The range of the average is provided in the table, such as 1-15 and 16-20, distinguishing between base classes and incrementally learned classes. Following Cermelli et al. (2020), the mIoU of the `background` class is only included in the *all* category, as it exists in both the base task and incremental tasks.

Baseline Since DKD (Baek et al., 2022) stands as one of the state-of-the-art methods in CISS, we reproduced the general regularization approaches, MiB (Cermelli et al., 2020) and PLOP (Douillard et al., 2021), based on its implementation. Note that pseudo-labeling is applied in PLOP and is not applied in DKD and MiB. The detailed implementation of memory usage, such as PLOP-M, DKD-M† [5], is explained in Section A.5. In this work, we assume the non-usage of the off-the-shelf detectors, thus we do not compare with the latest methods such as MicroSeg (Zhang et al., 2022b) and CoinSeg (Zhang et al., 2023).

Implementation details For all experiments, following Cermelli et al. (2020), we employed a DeepLab v3 segmentation network (Chen et al., 2017) with a ResNet-101 (He et al., 2016) backbone, pre-trained on ImageNet (Deng et al.,

[5]Implementation of DKD with memory is modified from the original implementation, dubbed as DKD-M† as the original implementaiton, DKD-M, cannot be directly used after memory modification mentioned in Section 3.6

Table 4: Test mIoU results after the final incremental task, averaging the results over 3 different runs.

	Overlapped											
	15-1 Task			5-3 Task			10-1 Task			10-5 Task		
	1-15	16-20	all	1-5	6-20	all	1-10	11-20	all	1-10	11-20	all
MiB (Cermelli et al., 2020)	31.92	17.33	30.98	60.72	49.62	53.94	9.15	22.42	18.93	67.77	58.16	64.25
PLOP (Douillard et al., 2021)	62.77	12.25	49.99	17.46	36.08	34.05	23.53	11.66	17.26	58.78	50.39	56.29
PLOP-M (Douillard et al., 2021)	63.71	25.85	55.00	65.02	43.41	50.64	15.97	9.78	15.32	74.41	56.53	66.71
DKD (Baek et al., 2022)	76.23	40.03	68.14	64.42	51.08	56.01	70.79	46.08	59.67	71.33	59.56	66.64
DKD-M† (Baek et al., 2022)	**76.66**	**45.29**	**69.84**	**68.14**	54.15	59.14	**72.50**	**53.14**	**64.04**	72.19	59.80	67.18
MiB + AugM (ours)	73.21	36.20	65.06	67.93	**59.42**	**62.76**	64.02	37.89	52.30	**74.53**	**61.77**	**69.19**
	Partitioned											
	15-1 Task			5-3 Task			10-1 Task			10-5 Task		
	1-15	16-20	all	1-5	6-20	all	1-10	11-20	all	1-10	11-20	all
MiB (Cermelli et al., 2020)	22.45	13.04	23.01	50.47	45.65	48.71	2.58	16.53	12.88	62.42	54.61	60.00
PLOP (Douillard et al., 2021)	63.62	11.72	49.12	16.48	27.66	27.67	14.85	9.90	11.79	49.86	44.96	49.48
PLOP-M (Douillard et al., 2021)	63.64	23.48	54.45	57.30	38.50	45.22	13.72	12.51	15.55	70.04	51.75	62.36
DKD (Baek et al., 2022)	73.86	35.86	65.51	61.72	47.92	53.14	67.24	43.11	56.67	64.88	55.68	64.12
DKD-M† (Baek et al., 2022)	**76.82**	**44.40**	**69.79**	**66.69**	52.16	57.40	**70.91**	**52.03**	**62.80**	70.65	56.53	64.92
MiB + AugM (ours)	73.15	34.02	64.29	65.47	**59.57**	**62.34**	63.85	41.24	53.82	**72.59**	**59.36**	**67.13**

2009). For training our method (MiB-AugM), we optimize the network with the learning rate of 10^{-3} for the backbone model and 10^{-2} for the rest. SGD with a Nesterov momentum value of 0.9 is used for optimization in all incremental steps. Other training implementations are equal to details written in Cermelli et al. (2020). For detailed implementations of our model and other baselines, please refer to Section A.6. In terms of exemplar memory, consistent with prior research (Cha et al., 2021; Baek et al., 2022), we utilized memory with a fixed size of $M = 100$.

4.2 Main Results

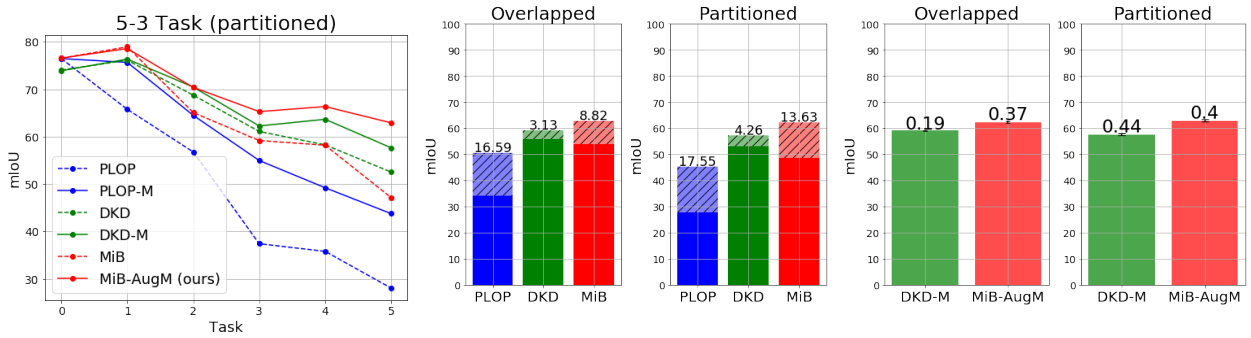
4.2.1 Key Observations of Partitioned Scenario via Baseline Results

Label conflicts of the exemplar memory resolved in the *partitioned* scenario Table 4 reports the mIoU results of test data after training the final incremental task. In every tasks, all methods demonstrate a greater improvement when exemplar memory is used in the *partitioned* compared to the *overlapped*. For example, in 5-3 task, MiB, PLOP, and DKD shows 8.82, 16.59, and 3.13 gains respectively in the *overlapped* while 13.63, 17.55, and 4.26 improvements were observed in the *partitioned*, which is also visualized in Figure 4b. Moreover, in 10-1 task, PLOP demonstrates a decline from 17.26 to 15.32, while an increase of 3.76 is observed in the *partitioned*. These results implies that, in the *overlapped* scenario, label conflicts caused by the exemplar memory impacts the final results of current CISS methods as our previous observations, and that the *partitioned* scenario can efficiently resolve this issue, enabling unbiased comparison between methods.

Robust to random selection in constructing the *partitioned* scenario Given that the data splitting rule used in *partitioned* entails randomness, we analyze the impact of this variance in terms of the final results of each method trained on different dataset construction seeds. We also report the variance of models learned on *overlapped* that does not involve randomness for data construction to represent the default variance for training each method. The figures displayed above the bar plot in Figure 4c illustrate the performance variance of each method. Notably, the models trained on the *partitioned* dataset exhibit similar randomness to the default variance observed in the *overlapped* scenario. This suggests that the final results of each model are robust to the randomness during the construction of the *partitioned* dataset.

4.2.2 Experimental Results on MiB-AugM

MiB-AugM as a new baseline for plasticity MiB with our proposed memory loss, MiB-AugM, exhibits state-of-the-art results across several incremental tasks. In Table 4, in both the *partitioned* 5-3 and 10-5 tasks, MiB-AugM outperforms the state-of-the-art method by 4.94 and 2.21, respectively. Additionally, Figure 4a demonstrates that this superiority remains consistent for every task.



(a) mIoU on each incremental tasks (b) Memory gain of each method (c) Variance for each setting (3 seeds)

Figure 4: Results of each method on 5-3 Task

While our proposed memory-based baseline may not demonstrate superior results in every task, we highlight two key points that support MiB-AugM as a promising future baseline for plasticity. Firstly, MiB-AugM shows better performance in tasks that are more complex in terms of learning new concepts. As more classes are introduced in the new task, the model must distinguish not only between old and new classes but also among the new classes themselves. Hence, a higher level of plasticity is required, particularly in the 5-3 and 10-5 tasks. Secondly, MiB-AugM is an efficient method in terms of hyper-parameters. Unlike other methods that necessitate multiple hyper-parameters (at least 4 for DKD and PLOP), MiB-AugM has only λ. Given that MiB-AugM can efficiently achieve state-of-the-art performance in several incremental tasks requiring complex plasticity, we insist that it holds promise as a valuable baseline for plasticity in future CISS studies.

5 Concluding Remarks, Limitation, and Future Work

Our work addresses the unrealistic aspect of the *overlapped* setting in CISS, where identical images are reused in future tasks with different pixel labels, which is not practical for real-world learning scenarios. We demonstrate that this artifact can lead to unfair advantage or disadvantage for commonly used techniques in CISS, resulting in biased comparisons among algorithms. To address this, we propose an alternative *partitioned* scenario that eliminates data reappearnce while meeting incremental learning requirements, such as capturing background shifts between previous and new classes. Additionally, we introduce a simple yet competitive exemplar memory-based method that effectively handles background shifts in stored data. This method efficiently uses exemplar memory in the proposed setting and outperforms state-of-the-art methods on several tasks that involve learning multiple new classes incrementally.

Limitation and broader impact While promising, our work has a few limitations that warrant future exploration. Firstly, our proposed replay-based baseline exhibits inferior performance compared to state-of-the-art methods in the 15-1 and 10-1 tasks, which involve learning a small number of classes in common. Addressing the stability-plasticity dilemma regarding the number of new classes requires further research. Secondly, our findings are based on experiments conducted solely on the Pascal VOC dataset, which is relatively small. Future research should explore other datasets beyond Pascal VOC (Everingham et al., 2010) or ADE 20K (Zhou et al., 2017), with larger volumes of data. We intend to investigate incremental scenarios involving larger datasets as part of our future work.

ACKNOWLEDGEMENTS

This work was supported in part by the National Research Foundation of Korea (NRF) grant [No.2021R1A2C2007884] and by Institute of Information & communications Technology Planning & Evaluation (IITP) grants [RS-2021-II211343, RS-2021-II212068, RS-2022-II220113, RS-2022-II220959] funded by the Korean government (MSIT). It was also supported by SNU-NAVER Hyperscale AI Center and AOARD Grant No. FA2386-23-1-4079.

REFERENCES

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32, 2019.

Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 844–853, 2021.

Donghyeon Baek, Youngmin Oh, Sanghoon Lee, Junghyup Lee, and Bumsub Ham. Decomposed knowledge distillation for class-incremental semantic segmentation. *Advances in Neural Information Processing Systems*, 35: 10380–10392, 2022.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Gail A Carpenter and Stephen Grossberg. Art 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, 1987.

Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9233–9242, 2020.

Sungmin Cha, YoungJoon Yoo, Taesup Moon, et al. Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. *Advances in neural information processing systems*, 34:10919–10930, 2021.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

Jinpeng Chen, Runmin Cong, LUO Yuxuan, Horace Ip, and Sam Kwong. Saving 100x storage: Prototype replay for reconstructing training sample distribution in class-incremental semantic segmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

Aristotelis Chrysakis and Marie-Francine Moens. Online bias correction for task-free continual learning. *ICLR 2023 at OpenReview*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pp. 86–102. Springer, 2020.

Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4040–4050, 2021.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip HS Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new hope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11888–11897, 2023.

Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13926–13935, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 831–839, 2019.

Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhenchao Jin. Cssegmentation: An open source continual semantic segmentation toolbox based on pytorch. https://github.com/SegmentationBLWX/cssegmentation, 2023.

Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *Advances in neural information processing systems*, 33:3647–3658, 2020.

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

Hyunseo Koh, Minhyuk Seo, Jihwan Bang, Hwanjun Song, Deokki Hong, Seulki Park, Jung-Woo Ha, and Jonghyun Choi. Online boundary-free continual learning by scheduled data prior. In *The Eleventh International Conference on Learning Representations*, 2022.

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

Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24246–24255, 2023.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4:504, 2013.

Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1114–1124, 2021.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 524–540. Springer, 2020.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12183–12192, 2020.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 374–382, 2019.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3014–3023, 2021a.

Shipeng Yan, Jiale Zhou, Jiangwei Xie, Songyang Zhang, and Xuming He. An em framework for online incremental learning of semantic segmentation. In *Proceedings of the 29th ACM international conference on multimedia*, pp. 3052–3060, 2021b.

Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. Neural collapse inspired feature-classifier alignment for few-shot class incremental learning. *arXiv preprint arXiv:2302.03004*, 2023.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Chang-Bin Zhang, Jia-Wen Xiao, Xialei Liu, Ying-Cong Chen, and Ming-Ming Cheng. Representation compensation networks for continual semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7053–7064, 2022a.

Zekang Zhang, Guangyu Gao, Zhiyuan Fang, Jianbo Jiao, and Yunchao Wei. Mining unseen classes via regional objectness: A simple baseline for incremental segmentation. *Advances in Neural Information Processing Systems*, 35:24340–24353, 2022b.

Zekang Zhang, Guangyu Gao, Jianbo Jiao, Chi Harold Liu, and Yunchao Wei. Coinseg: Contrast inter-and intra-class representations for incremental segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 843–853, 2023.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

Lanyun Zhu, Tianrun Chen, Jianxiong Yin, Simon See, and Jun Liu. Continual semantic segmentation with automatic memory sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3082–3092, 2023.

A APPENDIX

A.1 DATA CONFIGURATION COMPARISON: OVERLAP, DISJOINT, AND PARTITIONED

The tables in Table 5 provide a summary of the ground truth labels for each image in every incremental tasks. The first two columns show the image name and its corresponding oracle classes, while subsequent columns display the ground truth labels for each task. It's important to note that pixels not belonging to classes in $\mathcal{C}^t$ are annotated as the `background` class (abbreviated as *bg* in the table). To illustrate background shifts, pixels affected by unseen classes are highlighted in red, while those affected by previous classes are highlighted in blue. A hatched line is used for data not present in each task.

Table 5: Labeling information of each image in Figure 2 for each incremental scenario: *overlapped*, *Disjoint*, and *Ours*.

(a) Overlapped

Image	**Oracle class**	**Given ground truth class**		
		Task 1 (person)	Task 2 (motorbike)	Task 3 (car)
Img1	person motorbike car	person bg bg	bg motorbike bg	bg bg car
Img2	person motorbike	person bg	bg motorbike	-
Img3	car	-	-	car
Img4	person car	person bg	-	bg car
Img5	person	person	-	-

(b) Disjoint

Image	**Oracle class**	**Given ground truth class**		
		Task 1 (person)	Task 2 (motorbike)	Task 3 (car)
Img1	person motorbike car	-	-	bg bg car
Img2	person motorbike	-	bg motorbike	-
Img3	car	-	-	car
Img4	person car	-	-	bg car
Img5	person	person	-	-

(c) Partitioned (Ours)

Image	**Oracle class**	**Given ground truth class**		
		Task 1 (person)	Task 2 (motorbike)	Task 3 (car)
Img1	person motorbike car	-	bg motorbike bg	-
Img2	person motorbike	-	bg motorbike	-
Img3	car	-	-	car
Img4	person car	person bg	-	-
Img5	person	person	-	-

Similar to data configuration in CIL, the *disjoint* scenario in CISS ensures the separation of data from each task. However, achieving disjointness in the *disjoint* scenario requires prior knowledge of unseen classes and excludes data that could cause background shifts of these unseen classes (indicated by the absence of bg in the table). Given that background shift poses a significant challenge in CISS, many studies opt for the *overlapped* scenario, which can result in background shifts of both previous and unseen classes. However, in this work, we highlight the issues associated with overlapping data in the *overlapped* scenario and propose a novel scenario called *partitioned*. Our proposed scenario ensures the separation between task data while accommodating background shifts of both unseen and previous classes.

A.2 Details for pseudo-labeling analysis

A.2.1 Overview of Pseudo Retrieval Rate (PRR)

Figure 5 illustrates the pseudo-labeling and the pseudo retrieval rate (PRR) evaluation procedure. First, the previously learned model $f_{\theta^{t-1}}$ predicts labels of the background region of a given image x. Then, the prediction $(\hat{y}^{t-1})$ is added on to the ground truth label y to construct the pseudo-label $\tilde{y}^{t-1}$. After IoU for each class is calculated, the mean IoU over classes from the background class and the classes from the old task $\{c_{bg}\} \cup \mathcal{C}^{0:t-1}$ is returned.

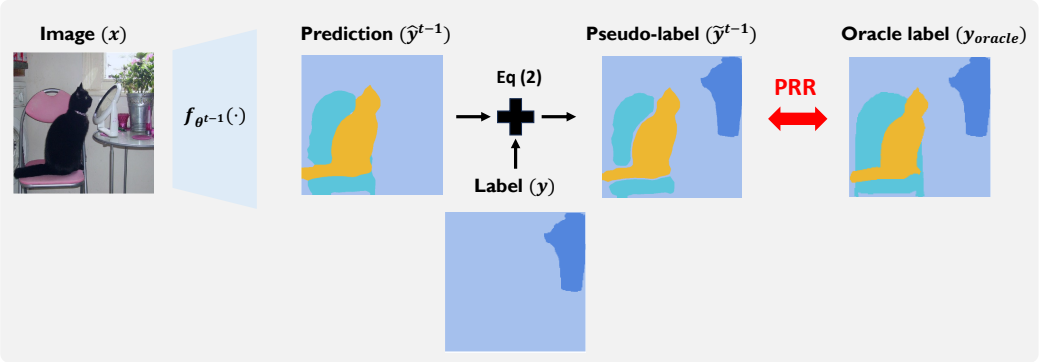


Figure 5: An overview of Pseudo-labeling Retrieval Rate (PRR) metric evaluation

A.2.2 Implementation details for experiments

Table 6: Data configuration for pseudo-label analysis experiments in Section 3.3

Task	Class order	Number of each subset in $\mathcal{D}^0$			
		Non-overlapping	overlapping		Total
			$\mathcal{D}_{seen}$	$\mathcal{D}_{unseen}$	
15-1	Type a	9234	167	167	9568
	Type b	9214	32	31	9277
	Type c	8449	197	197	8843
15-5	Type a	8437	566	565	9568
	Type b	8452	413	412	9277
	Type c	7506	669	668	8843

Data configuration The base task dataset $\mathcal{D}^0$ consists of overlapping dataset reintroduced in future tasks and non-overlapping dataset which solely appear in task 0. In the experiment, the overlapping dataset is randomly divided into half, $\mathcal{D}_{seen}$ and $\mathcal{D}_{seen}$. Table 6 shows the number of each dataset, non-overlapping, $\mathcal{D}_{seen}$, $\mathcal{D}_{seen}$, and $\mathcal{D}^0$. The model is trained on the modified training dataset $\mathcal{D}^0 \setminus \mathcal{D}_{unseen}$, and then PRR evaluation is done on $\mathcal{D}_{unseen}$ and $\mathcal{D}_{seen}$.

Training details The fine-tuning model is trained under the same training scheme used for PLOP in (Douillard et al., 2021). For training details, please refer PLOP details in Table 9.

A.3 Details for exemplar memory analysis

A.3.1 Implementation details for experiment

Data configuration After the model is trained with $\mathcal{D}^0$ at task 0, the model is fine-tuned on data from $\mathcal{D}^1$ and the exemplar memory at task 1. To construct an *overlapping memory*, we first calculate the average overlapping ratio in the memory which indicates the total amount of data from $\mathcal{D}^0 \cap \mathcal{D}^1$ out of the exemplar memory size M. Through 100 different class-balanced memory construction, we report the average ratio for each task and class order. Table 7 shows the number of training data used in each task and the ratio of overlapping data in the *overlapping memory*.

Training details The fine-tuning model is trained under the same training scheme used for PLOP in (Douillard et al., 2021). For training details, please refer PLOP details in Table 9.

Table 7: Data configuration for each dataset when *overlapping memory* is used for exemplar memory analysis experiment in Section 3.3

Task	Class order	Overlapping ratio in memory	$\mathcal{D}^0$	$\mathcal{D}^0 \cap \mathcal{D}^1$	$\mathcal{D}^1$	$\mathcal{M}^0$ (Non-overlapping/overlapping)
15-1	Type a	3.42 %	9568	334	487	97 / 3
	Type b	3.48 %	9214	394	1177	97 / 3
	Type c	16.6 %	8672	2425	3898	83 / 17
15-5	Type a	9.58 %	9568	1131	2145	90 / 10
	Type b	8.81 %	8843	1337	3076	91 / 9
	Type c	35.3 %	8672	3045	4955	65 / 35
10-1	Type a	5.61 %	6139	431	528	94 / 6
	Type b	0.36 %	7703	48	264	99 / 1
	Type c	35.81 %	5255	1831	3898	64 / 36
10-5	Type a	34.22 %	6139	2113	5542	66 / 34
	Type b	16.23 %	7703	1523	2663	84 / 16
	Type c	45.58 %	5255	2353	6406	54 / 46

A.4 Implementation error in previous stuides

As illustrated in Figure 6, the model is expected to see ground-truth masks annotated with *motorbike* class for current data and *person* class for data stored in memory, respectively. However, following code implementation of previous studies (Cha et al., 2021; Baek et al., 2022; Zhang et al., 2022b; 2023), the ground-truth mask for memory data is labeled with *motorbike* class.

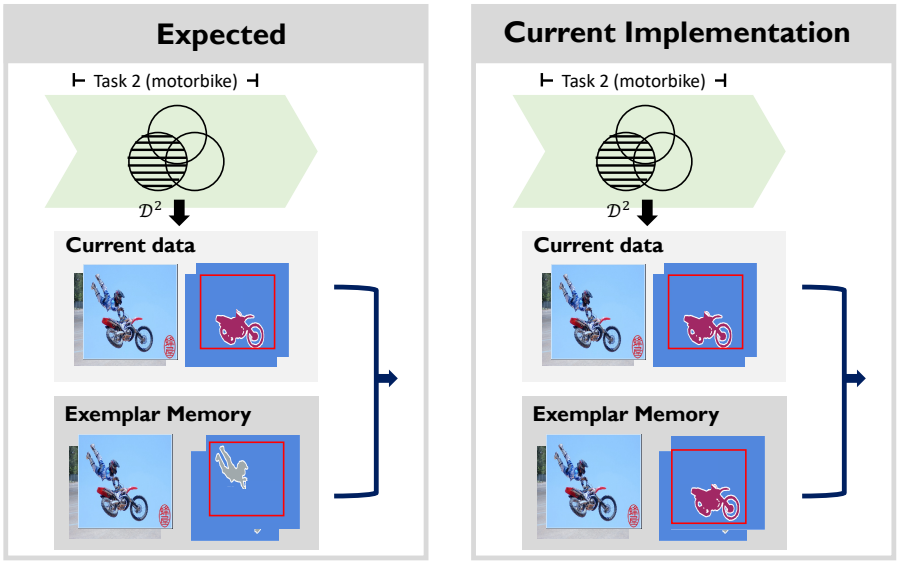


Figure 6: This figure illustrates the labeling issue in code implementation of current CISS studies (Cha et al., 2021; Baek et al., 2022; Zhang et al., 2022b; 2023). Note that the ground-truth mask of data from the exemplar memory does not provide labels for the classes of the previous task at which it was stored. Instead, it provides labels for the classes of the current task.

A.5 Implementation of MiB-AugM, PLOP-M, and DKD-M†

In this section, we explain the concrete formula of MiB-AugM, PLOP-M, and DKD-M† following the notation in Section 3.1.

A.5.1 MiB-AugM

The overall loss for MiB-AugM model can be defined as follows.

$$\mathcal{L}(\theta^t) = \underbrace{\frac{1}{|\mathcal{D}^t|}\sum_{(x,y)\in\mathcal{D}^t}\mathcal{L}_{unce}(y, f_{\theta^t}(x)) + \frac{\lambda}{|\mathcal{D}^t \cup \mathcal{M}^{t-1}|}\sum_{(x,y)\in\mathcal{D}^t\cup\mathcal{M}^{t-1}}\mathcal{L}_{unkd}(f_{\theta^{t-1}}(x), f_{\theta^t}(x))}_{\text{From MiB (Cermelli et al., 2020)}} + \underbrace{\frac{1}{|\mathcal{M}^{t-1}|}\sum_{(x,y)\in\mathcal{M}^{t-1}}\mathcal{L}_{mem}(y, f_{\theta^t}(x))}_{\text{Our proposed memory loss}} \tag{7}$$

Here, we rewrite the formula of $\mathcal{L}_{unce}$ and $\mathcal{L}_{unkd}$ defined in Cermelli et al. (2020) on our notation.

$$\mathcal{L}_{unce}(y, f_{\theta^t}(x)) = -\frac{1}{N}\sum_{i=1}^{N}\log \ddot{p}^t_{i,y_i} \tag{8}$$

$$\mathcal{L}_{unkd}(f_{\theta^{t-1}}(x), f_{\theta^t}(x)) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c\in\mathcal{C}^{0:t-1}} p^{t-1}_{i,c}\log \dot{p}^t_{i,c} \tag{9}$$

where $\ddot{p}^t_{i,c}$ and $\dot{p}^t_{i,c}$ indicates different augmentation technique of prediction probabilities. The augmented prediction for $\mathcal{L}_{unce}$, $\ddot{p}^t_{i,c}$, is defined as follows.

$$\ddot{p}^t_{i,c} = \begin{cases} p^t_{i,c} & \text{if } c \neq c_{bg} \\ p^t_{i,c_{bg}} + \sum_{k\in\mathcal{C}^{0:t-1}} p^t_{i,k} & \text{if } c = c_{bg} \end{cases} \tag{10}$$

A.5.2 PLOP-M

The loss function in PLOP-M remains unchanged, same as PLOP (Douillard et al., 2021). However, PLOP-M distinguishes itself by updating with concatenated data from both the current task and memory, using an equal ratio from each.

$$\mathcal{L}(\theta^t) = \frac{1}{|\mathcal{D}^t \cup \mathcal{M}^{t-1}|}\sum_{(x,y)\in\mathcal{D}^t\cup\mathcal{M}^{t-1}}\mathcal{L}_{ce}(\tilde{y}, f_{\theta^t}(x)) + \lambda\mathcal{L}_{pod}(f_{\theta^{t-1}}(x), f_{\theta^t}(x)) \tag{11}$$

A.5.3 DKD-M†

The loss function in DKD-M in the original paper remains also unchanged in its original paper (Baek et al., 2022). Namely, the data from concatenated set, *i.e.,* $(x, y) \sim \mathcal{D}^t \cup \mathcal{M}^{t-1}$, was forwarded to $\mathcal{L}_{kd}.\mathcal{L}_{dkd}, \mathcal{L}_{mbce}$, and $\mathcal{L}_{ac}$. However, looking at $\mathcal{L}_{mbce}$ in eq 12, it only updates the new class score, which is awkward for data in memory that does not have any labels of new classes.

$$\mathcal{L}_{mbce}(y, f_{\theta^t}(x)) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c\in\mathcal{C}^t}\gamma\mathbf{1}_{\{y_i=c\}}\log p^t_{i,c} + \mathbf{1}_{\{y_i\neq c\}}\log(1-p^t_{i,c}) \tag{12}$$

Later, we noticed that this wrong usage did not harm the performance because the labeling issue existed in memory retrieval mentioned in Section 3.6 (Manuscript). Therefore, after correction in memory target labeling, we add a $\mathcal{L}_{mbce}$ loss for memory, dubbed as $\mathcal{L}_{membce}$, which is defined as follows.

$$\mathcal{L}_{membce}(y, f_{\theta^t}(x)) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c\in\mathcal{C}^{0:t-1}} \gamma \mathbf{1}_{\{y_i=c\}} \log p_{i,c}^t + \mathbf{1}_{\{y_i\neq c\}} \log(1-p_{i,c}^t) \tag{13}$$

The overall loss function for DKD-M† is as follows.

$$\begin{aligned}\mathcal{L}(\theta^t) = \frac{1}{|\mathcal{D}^t \cup \mathcal{M}^{t-1}|} \sum_{(x,y)\in\mathcal{D}^t\cup\mathcal{M}^{t-1}} & \underbrace{\Big[\alpha\mathcal{L}_{kd}(f_{\theta^{t-1}}(x), f_{\theta^t}(x)) + \beta\mathcal{L}_{dkd}(f_{\theta^{t-1}}(x), f_{\theta^t}(x))\Big]}_{\text{From DKD (Baek et al., 2022)}} \\ & + \frac{1}{|\mathcal{D}^t|}\sum_{(x,y)\in\mathcal{D}^t} \underbrace{\Big[\mathcal{L}_{mbce}(y, f_{\theta^t}(x)) + \mathcal{L}_{ac}(y, f_{\theta^t}(x))\Big]}_{\text{From DKD (Baek et al., 2022)}} \\ & + \frac{1}{|\mathcal{M}^{t-1}|}\sum_{(x,y)\in\mathcal{M}^{t-1}} \underbrace{\mathcal{L}_{membce}(y, f_{\theta^t}(x))}_{\text{Added loss modified from } \mathcal{L}_{mbce}}\end{aligned} \tag{14}$$

A.6 IMPLEMENTATION DETAILS

Table 8: The number of train data for every task in the *overlapped* and *partitioned* scenarios.

Task	Scenario	Seed	The number of training data for each task	Total
15-1 Task	overlapped	-	9568 / 487 / 299 / 491 / 500 / 548	11893
	partitioned	0	9031 / 254 / 266 / 270 / 434 / 327	10582
		1	9059 / 263 / 271 / 256 / 413 / 320	10528
		2	9030 / 274 / 258 / 253 / 437 / 330	10582
5-3 Task	overlapped	-	2836 / 2331 / 1542 / 2095 / 4484 / 1468	14756
	partitioned	0	2222 / 1860 / 972 / 1574 / 2923 / 1031	10582
		1	2237 / 1859 / 991 / 1616 / 2890 / 989	10582
		2	2221 / 1855 / 993 / 1589 / 2904 / 1020	10582
10-1 Task	overlapped	-	6139 / 528 / 1177 / 444 / 482 / 3898 / 487 / 299 / 491 / 500 / 548	14993
	partitioned	0	4847 / 207 / 961 / 310 / 303 / 2403 / 254 / 266 / 270 / 434 / 327	10582
		1	4861 / 226 / 987 / 322 / 307 / 2356 / 263 / 271 / 256 / 413 / 320	10582
		2	4856 / 213 / 969 / 317 / 303 / 2372 / 274 / 258 / 253 / 437 / 330	10582
10-5 Task	overlapped	-	6139 / 5542 / 2145	13826
	partitioned	0	4847 / 4148 / 1551	10582
		1	4861 / 4198 / 1523	10582
		2	4856 / 4174 / 1552	10582

Baseline reproduction and experimental environment The code environment of CISS is divided into two branches: Distributed data parallel (DDP) implemented on Nvidia Apex (https://github.com/NVIDIA/apex) and Torch (Paszke et al., 2017). Initial studies primarily utilized the former, with Jin (2023) organizing numerous baselines for evaluation. Due to the transition of Nvidia Apex to Torch in deep learning community[6], recent CISS works began to work on Torch environment. However, recent works conducted on the latter omitted the process of re-implementing baselines, instead reporting figures from the original paper. Given the transition from Apex to Torch, resulting in significant changes in built-in operations, it's widely acknowledged that a mismatch in results exists between the two environments. To facilitate fair comparisons, we report our implementation of two baselines in the Torch version.

Dataset and protocols Pascal VOC 2012 (Everingham et al., 2010) consists of 10.582 training and 1,449 validation images for 20 object and background classes. For the training image, random crop with size 512, random resize with $(0.5, 2.0)$ ratio, and normalization are used. For the test image, only normalization is used.

Following Cermelli et al. (2020), incremental tasks are denoted by the number of classes used in base classes, $|\mathcal{C}^0|$, and the number of classes learned in each incremental task, $|\mathcal{C}^t| \; \forall t \in \{1, \ldots, T\}$. For example, if the base task class is composed of 15 classes and 1 class is learned at every task, it is denoted as 15-1 task.

Table 8 demonstrates the number of data $\mathcal{D}^t$ used for each task in both the *partitioned* and *overlapped* scenarios. Note that the dataset remains consistent across tasks in the *overlapped*, whereas variations are observed in the *partitioned* scenario across different seeds.

Training details Table 9 summarizes the training details used for each method. MiB, PLOP-M, and DKD-M use the same details as MiB-AugM, PLOP, and DKD, respectively. Note that some training details of MiB are different from those that were used in MiB. For methods that use exemplar memory, we replace half of the data from the current batch with the data from the memory.

Table 9: Training details for each method

	Common					Base task		Incremental task	
	Batch size	Epoch	Optimizer	Momentum	Lr schedule	Lr (backbone / aspp / classifier)	Weight decay	Lr (backbone / aspp / classifier)	Weight decay (backbone / aspp / classifier)
PLOP (Douillard et al., 2021)	24	30	SGD	Nesterov, 0.9	PolyLR	0.01 / 0.01 / 0.01	0.001	0.001 / 0.001 / 0.001	0.001 / 0.001 / 0.001
DKD (Baek et al., 2022)	32	60	SGD	Nesterov, 0.9	PolyLR	0.001 / 0.01 / 0.01	0.0001	0.0001 / 0.001 / 0.001	0 / 0 / 0.0001
MiB-AugM (Ours)	24	30	SGD	Nesterov, 0.9	PolyLR	0.01 / 0.01 / 0.01	0.001	0.001 / 0.01 / 0.01	0 / 0 / 0.001

A.6.1 HYPER-PARAMETERS

Table 10 summarizes the main hyper-parameters of each method used for training. Notations for each hyper-parameter are from the original paper. We also used same hyper-parameter used in MiB (Cermelli et al., 2020).

[6]Please refer to https://github.com/NVIDIA/apex/issues/818

Table 10: Hyper-parameters used for each method

	Base task training	Incremental task	Inference
PLOP (Douillard et al., 2021)	-	$\lambda_f = 0.01$ (features), $\lambda_l = 0.0005$ (logits), $\tau = 0.001$, pod scale= $[1, \frac{1}{2}, \frac{1}{4}]$	-
DKD (Baek et al., 2022)	$\gamma = 2$	$\alpha = 5, \beta = 5, \gamma = 1$	$\tau = 0.5$
MiB-AugM (Ours)	-	$\lambda = 5$	-

A.7 Computation details

The experiments were conducted using PyTorch (Paszke et al., 2017) 1.13.1 with CUDA 11.2 and were run on four NVIDIA Titan XP GPUs with 12GB memory per device. All experiments except PLOP (Douillard et al., 2021) were conducted with distributed data-parallel training on four GPUs. For PLOP (Douillard et al., 2021) experiments, we use 2 GPUs for parallel training since the results of the original paper could not be achieved by other numbers.

A.8 Software and Dataset Licenses

A.8.1 Datasets

- Pascal VOC (Everingham et al., 2010): CC BY-NC-SA 3.0 License
  http://host.robots.ox.ac.uk/pascal/VOC/

A.8.2 Models

- MiB (Cermelli et al., 2020): MiT License
  https://github.com/fcdl94/MiB
- PLOP (Douillard et al., 2021): MiT License
  https://github.com/arthurdouillard/CVPR2021_PLOP
- DKD (Baek et al., 2022): GPL-3.0 License
  https://github.com/cvlab-yonsei/DKD