Efficient Class-Incremental Segmentation Learning via Expanding Visual Transformers

Anonymous authors
Paper under double-blind review

Abstract

Incrementally learning new semantic concepts while retaining existing information is fundamental for several real-world applications. While behaviors of different sizes of backbones and architectural choices have been studied to propose efficient limited-sized architectures within many non-incremental computer vision applications, only large convolutional and Visual Transformer (ViT) backbones have been explored for class-incremental semantic segmentation, without providing a fair comparison wrt model size. In this work, we propose a fair study across existing class-incremental semantic segmentation methods, focusing on the models efficiency wrt their memory footprint. Moreover, we propose TILES (Transformer-based Incremental Learning for Expanding Segmenter), a novel approach exploiting small-size ViT backbones efficiency to offer an alternative solution where severe memory constraints are applied. It is based on expanding the architecture with the increments, allowing to learn new tasks while retaining old knowledge within a limited memory footprint. Besides, in order to tackle the background semantic shift, we apply adaptive losses specific to the incremental branches, while balancing old and new knowledge. Furthermore, we exploit the confidence of each incremental task to propose an efficient branch merging strategy. TILES provides state-of-the-art results on challenging benchmarks using up to 14 times fewer parameters.

1 Introduction

In a traditional non-incremental learning context, machine learning models process the whole training data at once. However, incrementally learning new information held by new collected data while retaining past knowledge is a critical capacity needed in real-word applications such as robotics, self-driving vehicles or video surveillance, as past data is not always available for storage or privacy reasons. In a more specific context, Class-Incremental (CI) learning can be more constrained when the new data features new classes not seen previously without bringing anymore knowledge about previously learnt classes. While humans can learn to recognize new objects continuously without forgetting the ones previously seen, deep learning models can suffer from performance degradation on previously learned tasks. This phenomenon introduced by Mccloskey & Cohen (1989) is known as *catastrophic forgetting*, and happens especially if the new classes are learned with no supervision on past knowledge.

This problem has received much attention from the research community for the task of image classification (IC). Several approaches have proposed to expand the model architecture to learn new tasks such as Yan et al. (2021) or to retain a portion of the previous dataset and use it in subsequent steps. Other researchers have tried to solve CI learning in a more challenging setting, where no old data is available and with no or limited model expansion. Approaches based on transferring information from the old to the new network known as *Knowledge Distillation* (KD) presented by Hinton et al. (2015) have shown great success in alleviating catastrophic forgetting for IC. Architecture-wise, all first methods were CNN based until DyTox (Douillard et al. (2022)) which demonstrated good performances while using a Visual Transformer (ViT) based architecture (Dosovitskiy et al. (2021)) with very few additional parameters at each step.

However, less attention has been given to Class-Incremental Semantic Segmentation (CI-SS). This task holds one more challenge which is the *semantic shift of the background*: the semantic meaning of the background changes at each step since past and future classes are considered as *background* relative to the current step (which only relies on labels of current classes). This is why direct adaptations of CI-IC approaches do not perform well for CI-SS. Several solutions have been proposed to deal with this challenge such as self-inpainting by Cermelli et al. (2020). Besides, while most approaches dealing with CI-SS use Deeplab-v3 (Chen et al. (2017)) with a CNN backbone, more recently, CoinSeg (Zhang et al. (2023)), NeST (Xie et al. (2024)) and Incrementer (Shang et al. (2023)) leveraged ViT-based architectures and outperformed CNN-based models while employing large ViT backbones. However, to the best of our knowledge, no benchmark exists to fairly compare these various methods. Moreover, despite the various studies providing smaller sized ViT backbones on several computer vision tasks (Strudel et al. (2021); Liu et al. (2021)), no such small models have been proposed for CI-SS.

In this work, we provide a fair comparison across previous state-of-the-art (SOTA) CI-SS approaches, by taking into account the performance of the used backbones having different memory footprints. Besides, we demonstrate that previous ViT-based CI-SS methods using large backbones are not adapted for use cases with severe memory constraints where smaller backbones should be used. As an alternative, we propose a novel approach to solve limited footprint CI-SS based on an expanding ViT architecture. Similarly to Incrementer, we choose Segmenter architecture (Strudel et al. (2021)) for its class embeddings which represent an efficient way to retain knowledge on classes. In contrast, our method features an expanding architecture which helps alleviate the semantic shift of the background, especially when using small backbones. Therefore, our Transformer-based Incremental Learning for Expanding Segmenter (TILES) appears better suited to the constrained-memory CI-SS scenarios as it largely outperforms previous SOTA methods when limited-sized backbones are applied. It is also able to outperform previous approaches while using up to 14 times fewer parameters. The effectiveness of the proposed framework is demonstrated through extensive experiments on the CI-SS benchmarks Pascal-VOC (Everingham et al. (2010)) and ADE20k (Zhou et al. (2019)). Our contributions can be formulated as follows:

- We provide a fair comparison across different CI-SS methods while studying their behaviors and efficiencies. Besides, we demonstrate a big performance drop of the best SOTA approach when using smaller backbones.
- Alternatively, we propose TILES to solve CI-SS with critical memory constraints. We demonstrate that the proposed expanding mechanism dedicating balanced task-specialized branches alongside a suitable loss computation and a branch merging technique, are the more efficient to cope with constrained-memory cases.
- We improve the SOTA performance for CI-SS using limited-sized backbones, on two challenging benchmarks.

2 Related work

2.1 Class-incremental learning

We can distinguish three main families of approaches solving CI learning for image classification, object detection or semantic segmentation:

Replay methods: The strategy here is to use data holding past knowledge in the subsequent steps (Liu et al. (2020); Maracani et al. (2021)). This idea can provide good performances since there is limited or no catastrophic forgetting, but it is contrary to privacy constraints (*e.g* health, industrial or defense sensitive data). In this work, we do not consider this family of approaches.

Expansion methods: These approaches try to find an efficient way to make the model *evolve* throughout the epochs by allocating new parameters to the new classes. This strategy has already proven its worth in CI-IC by achieving good performance in Yan et al. (2021). DyTox (Douillard et al. (2022)) is the first approach to use an expanding transformer-based architecture with limited sized backbones for CI-IC. It

can dynamically process new knowledge by moderately increasing the number of parameters leading to performance similar to SOTA CNN-based approaches.

Regularization methods: This family of approaches focuses on how the model learns at each step and can be further divided into weight regularization and functional-based methods. On the one hand, *weight regularization* approaches, especially used for CI-IC, put constraints on the weights having high impact on previous tasks (Kirkpatrick et al. (2017)). On the other hand, *functional-based* approaches compute losses measuring the distance between a specific layer outputs produced by the previous and the new models respectively (Cermelli et al. (2020); Shmelkov et al. (2017); Phan et al. (2022)). These methods are the most popular thanks to their simplicity and capacity to continuously learn new classes. In particular, *Knowledge Distillation* (KD) proposed by Hinton et al. (2015) has shown great success reducing the catastrophic forgetting especially for CI-IC.

2.2 Class-incremental semantic segmentation

Several methods have tried to solve CI-SS using KD, given its success for CI-IC. However, the Cross Entropy Loss (L_{CE}) and the KD Loss (L_{KD}) contradict each other for CI-SS because of the semantic shift of the background. In fact, previously seen classes are labeled in the new ground-truth as background, hence a contradiction with the old model's predictions. Similarly, the new classes are classified as background by the old model, hence a contradiction with the new ground-truth labels. Therefore, this opposition makes the model cut the trade-off between rigidity and flexibility in an inefficient way.

MiB (Cermelli et al. (2020)) is the first to tackle the background semantic shift challenge by changing the new model outputs depending on which loss is computed. Indeed, the new model predictions corresponding to new classes are considered as background in the L_{KD}. Similarly, those corresponding to old classes are considered as background in the L_{CE}. PLOP (Douillard et al. (2021)) deals with the background semantic shift in a different manner, by joining the predictions made by the previous model and the new ground truth to generate a new target. Later, SDR (Michieli & Zanuttigh (2021)) and UCD (Yang et al. (2022)) show that using contrastive learning on lower dimension representations offers a powerful way to retain knowledge. RCIL (Zhang et al. (2022a)) uses representation compensation to merge old and new parameters while MicroSeg (Zhang et al. (2022b)) addresses the challenge of background shift by leveraging regional objectness to identify and preserve previously learned classes. Moreover, RBC (Zhao et al. (2022)) corrects context bias through a context-rectified image-duplet learning scheme, a biased-context-insensitive consistency loss, and an adaptive class-balanced learning strategy. Finally, Bg_Adapt (Zhang & Gao (2024)) leverages a background adaptation mechanism based on residual modeling to better handle background category evolution. Even though these methods propose different incremental strategies, they are all based on the Deeplab-v3 (Chen et al. (2017)) architecture with a ResNet (He et al. (2016)) backbone as the core model on which the CI-SS strategies are applied.

More recently, ViT-based architectures have been used to solve CI-SS trying to benefit from these architecture's efficiencies. On the one hand, two recent methods are based on a SwinB backbone (Liu et al. (2021)): first, CoinSeg (Zhang et al. (2023)) uses a discriminative feature representation thanks to inter-class and intra-class contrastive losses, while NeST (Xie et al. (2024)) proposes a pre-training method that transforms existing classifiers to initialize new ones, enhancing alignment with the model backbone. On the other hand, Incrementer (Shang et al. (2023)) takes advantage from the Segmenter (Strudel et al. (2021)) class embeddings to add new knowledge while retaining old information, based on a ViT-B backbone (Dosovitskiy et al. (2021)). It also proposes to adapt the L_{KD} to only focus on old class regions and to regularise training alleviating both overfitting on new classes and confusion of similar semantic concepts. Since these methods are based on different backbones, they re-implement some previous methods using the same backbone for performance comparison such as MiB(ViT-B), RBC(ViT-B) or MicroSeg(SwinB).

2.3 Positioning of our method

At each step, the new data has most likely different statistical properties from the data seen previously. This *distributional shift* has a huge impact on the optimization of the model weights. This is enhanced in the case of CNNs where Batch Normalization (BN) layers are usually present, having a high dependency to the data

distribution. Thus, if the distributional shift is not detected, the model will inevitably suffer from stronger catastrophic forgetting (Zhao et al. (2021)). Therefore, we believe that transformer-based architectures can be more appropriate for incremental learning since they do not rely on BN, making them better continual learners (Li et al. (2022)).

Moreover, a very important aspect of real world applications is the memory footprint. However, contrary to the non-incremental semantic segmentation approaches where the efficiency and the performance drop using smaller backbones have been extensively studied, only big backbones (ViT-B and SwinB) have been tested for ViT-based CI-SS methods. Uncertainty on how the incremental strategies would perform with smaller backbones makes them impractical for real-world use. Besides, it would be interesting to build new strategies for these constrained cases.

Furthermore, during the training of previous SOTA approaches, the rigidity-elasticity trade-off is usually handled by a hyperparameter balancing the L_{CE} of the predictions with relation to the new ground truth labels, and the L_{KD} between the new and the old model's predictions. However, the values of the two losses vary depending on the confidence acquired by the old model, the number of classes for each task, and the number of images used for the training of each step. Previous models did not take this into account and applied a fixed hyperparameter independent of the task. In this work, we aim to tackle this, for a better trade-off between old and new classes, by introducing an adaptive balancing parameter during training, alongside a weighting branch merging parameter during inference.

To sum up our positioning, the proposed method TILES aims i) to present a solution for scenarios with severe memory constraints by exploiting the efficient transformer architectures for CI-SS; ii) to leverage the effectiveness of both knowledge distillation and expansion methods for incremental learning to tackle the semantic shift of the background iii) while minimizing the parameter expansion at each increment and iv) without retaining data for privacy concerns.

3 Method

3.1 Problem definition

The goal of CI-SS is to learn a model able to perform well on an increasing set of semantic classes. Let T denote the total number of steps or increments at which a model has to learn a new task to cope with a new subset of classes in addition to previously learned classes. At step $t \in \{1, \dots, T\}$, let D^t denote an additional subset of annotated data: $D^t = \{(x_i^t, y_i^t)\}_i$ where x_i^t is the i-th image and y_i^t is the corresponding ground truth of the same size, where each pixel is classified in C^t, the set of semantic classes at step t. The challenge is that the foreground classes are supervised only at one step, *i.e* $C^n \bigcap C^m = \varnothing, \quad \forall n, m \in \{1, \dots, T\}, n \neq m$, even if older classes continue to appear in new images or if future classes appeared in the older steps they are both considered as background. However, at the end, the model should still be able to retain knowledge on all seen classes $C^1 \bigcup C^2 \bigcup ... \bigcup C^T$.

3.2 Overview

We propose TILES as an expanding CI-SS approach that is convenient for scenarios with severe memory constraints. In fact, while it is possible for big encoders/decoders to encompass several semantic concepts learnt during different steps with minimum forgetting thanks to their considerable number of parameters, we argue that this is not possible when much smaller backbones are used, which we demonstrate in sec. 4.2. Therefore, we design TILES allowing limited expansion, particularly useful and necessary for these cases, while assuring a limited memory footprint. Similar to Incrementer (Shang et al. (2023)) our approach is based on the Strudel et al. (2021) architecture: Segmenter which uses a fully transformer encoder-decoder to generate class masks for SS. To adapt this architecture to CI learning, at each step t, we add K_t class embeddings to represent the K_t newly added classes C^t. This technique offers an efficient encoding of the class knowledge which could be saved in the next steps helping the model to retain information through its weights. We use a shared encoder between all tasks. However, we adopt an expanding architecture where each task has its specific specialized decoder branch b. The training of different steps of TILES is presented

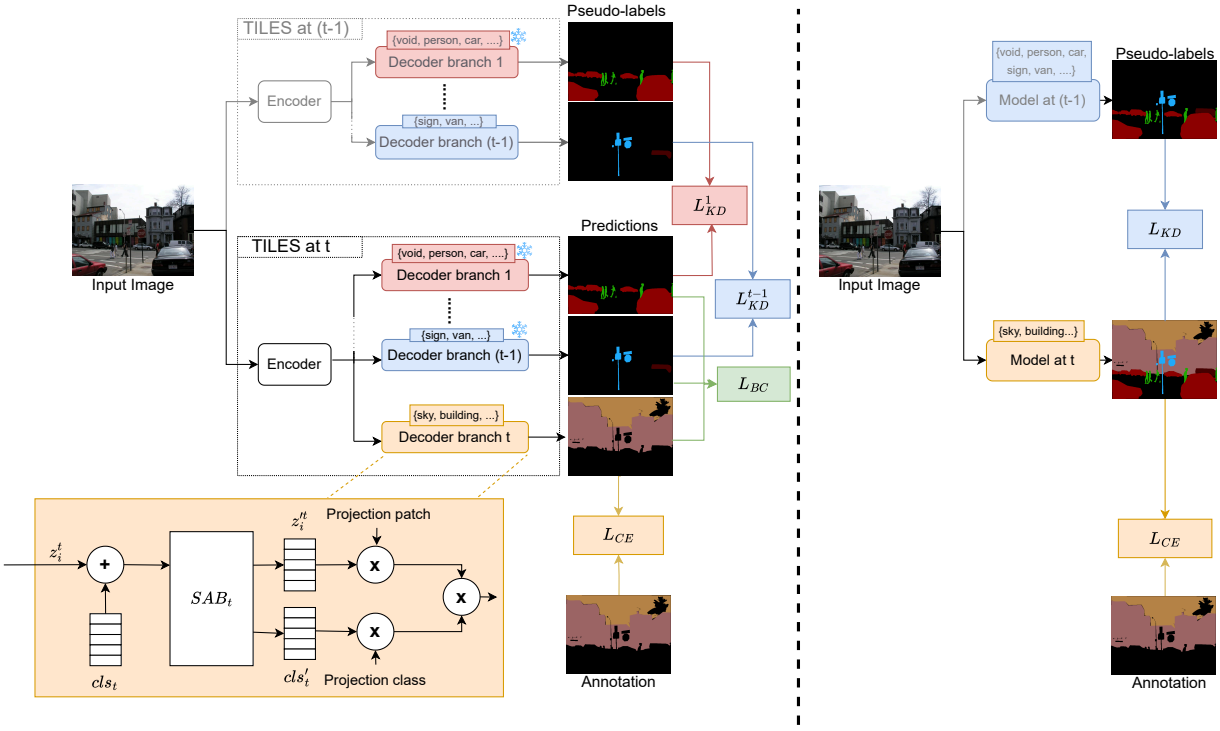

Figure 1: Training process of TILES (left) vs. KD-based SOTA methods (right). In TILES, the input image is first processed by the encoder, *e.g* ViT-Tiny, then each of the task branches (the tiles) generates a prediction map specific to the task it learned. L_{KD} losses are computed between old and new predictions of each old branch separately to retain old knowledge, while L_{CE} is used to learn to predict new classes via the new branch. L_{BC} helps the differentiation and specification of each branch. In previous KD-based CI-SS approaches, both L_{KD} and L_{CE} are applied on the same output with possibly opposed goals. The architectural design is explained in sections 3.3, 3.4 and the learning strategy is detailed in section 3.6.

in Figure 1 (left). Note that the decoders are adapted so that the expansion is not prodigal in terms of number of parameters.

During inference, the input image is processed by the encoder, then, by each of the T branches of the decoder in parallel. A final branch merging module is used to aggregate the predictions of different branches such as illustrated in Figure 2. Thanks to this expanding strategy, there is no semantic shift in each branch individually. The branch merging module carries out the choice of the final result using branch weights.

The different learning strategies in the initial step and then, how new branches are built on top of the previous model to segment all classes are detailed hereafter.

3.3 Encoder

The input image is denoted by $x_i^t \in \mathbb{R}^{H\times W\times C}$ where H, W and C are respectively the height, width and number of channels of the image. x_i^t is divided into patches of size $P \times P$ pixels to generate the sequence of patches $\mathbf{x}_i^t = [\mathbf{x}_{i,1}^t, ..., \mathbf{x}_{i,N}^t] \in \mathbb{R}^{N\times P^2\times C}$, where N is the number of patches *i.e* $N = \frac{HW}{P^2}$. These patches are then flattened, linearly projected and added to learnable position embeddings to generate the sequence $w_i^t = [w_{i,1}^t, \ldots, w_{i,N}^t] \in \mathbb{R}^{N\times D}$, D being the embedding dimension. The encoder generates an output $z_i^t = [z_{i,1}^t, \ldots, z_{i,N}^t] \in \mathbb{R}^{N\times D}$. The encoder weights are updated at each increment.

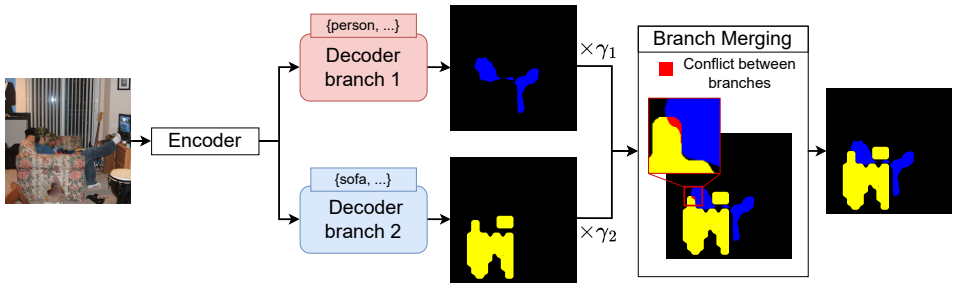

Figure 2: Branch merging module used during inference to merge two segmentation maps learned during different steps. Illustration is on a PascalVoc image using TILES-S trained on the [15-1] *overlapped* scenario. See sec. 3.5 for explanations.

3.4 Decoder incremental branches

To retain knowledge on classes, we use the learnable class embeddings as introduced in Segmenter (Strudel et al. (2021)) which are noted for the b-th branch (*i.e* task) as $cls_b = [cls_b^1, \dots, cls_b^{K_b}] \in \mathbb{R}^{K_b \times D}$. At the entry of each branch, the class embeddings cls_b are concatenated to the encoder outputs z_i^t. Hence, the sequence fed into the Mask Decoder is $d_{i,b}^t = [z_{i,1}^t, \dots, z_{i,N}^t, cls_b^1, \dots, cls_b^{K_b}] \in \mathbb{R}^{(N+K_b)\times D}$. The self-attention block decoder (SAB) then generates the sequence ${d'_{i,b}}^t = [{z'_{i,1}}^t, \dots, {z'_{i,N}}^t, {cls'_b}^1, \dots, {cls'_b}^{K_b}] \in \mathbb{R}^{(N+K_b)\times D}$ made of the transformed class embeddings $cls'_b \in \mathbb{R}^{K_b \times D}$ and transformed encoded patches $z_i'^t \in \mathbb{R}^{N\times D}$. The class embeddings and encoded patches are then respectively linearly projected, and the K_b masks are generated from their scalar product. For each branch, this output of dimension $N \times K_b$ is then upsampled to obtain a segmented image of the same size as the input in order to get the prediction at step t: $pred_{i,b}^t \in \mathbb{R}^{K_b \times H \times W}$ where $b \in \{1, \dots, t\}$. As a result, each decoder branch is specialized in a task where classes from other tasks are considered as background. At step t, only the decoder weights of branch b_t are changed while we freeze old step branches.

3.5 Branch merging

During inference, the predictions made by each task branch $pred_{i,b}^t$ are combined to generate the segmented image on all classes $pred_i^t$ such as presented in Figure 2. In particular, for each pixel, we merge the predictions p^b of the different branches to decide the final value p of that pixel using the following rules:

- if all task branches predict the pixel as background, *i.e* $\{p^b = 0,\ \forall b \in \{1, \dots, T\}\}$, then the final label is the background: $p \leftarrow 0$ (label for background);
- if only one branch predicts the pixel as a foreground class, *i.e* $\exists!\ b_1 \in \{1, \dots, T\}$ such that $\{p^{b_1} = c,\ c \neq 0\} \cap \{p^b = 0,\ \forall b \neq b_1\}$, then the final label is that foreground class: $p \leftarrow c$;
- if two or more branches predict the pixel as different foreground classes *i.e* $\exists\, b_1, b_2 \in \{1, \dots, T\},\ b_1 \neq b_2$ such that $\{p^{b_1} = c1,\ c1 \neq 0\} \cap \{p^{b_2} = c2,\ c2 \neq 0\}$ (red area in Figure 2), then the final label is the one scoring the highest confidence which is computed as a weighted probability $P(p^b) \cdot \gamma_b$, where γ_b is the probability compensation weight for the branch b.

Probability compensation weight: Background preponderance in images varies a lot depending on the dataset *e.g* no background class in ADE20k and a very present background class in Pascal-VOC such as detailed in sec. 4.1.1. It also highly depends on the nature of the task. For instance, the majority of pixels are considered as background in learning step 6 of the $100 - 10$ scenario, whereas background is much less present in the first step. This preponderance underrates, in different ratios, the probabilities of foreground classes, which causes different probability values of foreground classes while branch merging. To circumvent this, our branch merging strategy takes into account the relative sparsity of task classes C^b by choosing

$\gamma_b = \frac{\#p(background)}{\#p(images)}$, where $\#p(images)$ and $\#p(background)$ are respectively the number of pixels of all images used in train, and the corresponding number of background pixels.

3.6 Learning

Figure 1 illustrates the fundamental difference of loss computation between TILES and previous KD-based approaches. At a given step t, TILES (*left*) has t branches specialized in predicting t different tasks. It is trained to learn a new task on the new branch and to retain old tasks knowledge on old branches, while differentiating between semantic concepts of different tasks.

Learn a new task: To learn the new task, the ground-truth y_i^t contains only labels belonging to the new set C^t as foreground while all other classes are set to background. Hence, $L_{CE}(pred^t_{i,b=t}, y_i^t)$ is applied on the branch t specific to the new task that estimates $pred^t_{i,b=t}$. The goal is to be able to distinguish between the new classes and the old ones which are considered as background in this branch.

Retain knowledge on old tasks: Since the encoder weights are shared between all tasks and are updated while learning the new task, we have to make sure that these weights will not change so much on the new background pixels (containing old classes' pixels) so that they will no longer fit the old tasks, leading to catastrophic forgetting. Therefore, L^b_{KD} is computed for each branch $b \in \{1, .., t-1\}$ as a cross-entropy on the output softmax probabilities of predictions $pred^t_{i,b}$ and $pred^{t-1}_{i,b}$ at current and previous steps only on new background pixels (*i.e* where $y_i^t = 0$). Such as introduced in Incrementer (Shang et al. (2023)), a L_{KD} applied only on new background pixels instead of the whole image enables more elasticity on new foreground pixels ($y_i^t != 0$) to learn the new tasks and more rigidity on new background pixels which contain old foreground classes (see ablation in sec. 4.3).

Differentiate between semantic concepts of different branches: Incrementer (Shang et al. (2023)) demonstrated that close semantic concepts seen in different tasks could be confused since they are learnt in independent steps. This is further heightened in our case since we use totally independent decoders to learn different tasks. To alleviate this problem we add a Branch Classification Loss L_{BC} defined as the mean values of the mask $M_i = \{m_{j,k}, 1 \leq j \leq W, 1 \leq k \leq H\}$ for each image $x_i \in \mathbb{R}^{H \times W \times C}$, where:

$$m_{j,k} = \begin{cases} 0 \; if \; p^b_{j,k} = 0 \; \forall \; b \; \in \{1, \ldots, t\} \\ 0 \; if \; \exists! \; b_1 \in \{1, \ldots, t\}; \{p^{b_1}_{j,k} = c, \; c \neq 0\} \cap \{p^b_{j,k} = 0, \; \forall b \neq b_1\} \\ 1 \; otherwise \end{cases} \tag{1}$$

This loss encourages having at most one non-zero (foreground) value per pixel, alleviating confusion of close semantic classes learnt at different steps (see ablation in sec. 4.3).

Total loss: For step 1 the total loss is the presented $L_{CE}(pred^1_{i,1}, y_i^1)$. For steps $t > 1$, the total loss $Loss_i^t$ is computed as follows, where $\lambda_{old} > 0$ to balance rigidity vs. elasticity of the model:

$$Loss_i^t = \underbrace{\lambda_{old} \sum_{b \in [1:t-1]} L^t_{KD}(pred^t_{i,b}[y_i^t = 0], pred^{t-1}_{i,b}[y_i^t = 0])}_{\text{Loss on older tasks}} + \underbrace{L_{CE}(pred^t_{i,t}, y_i^t)}_{\text{Loss on the new task}} + \underbrace{L_{BC}(pred^1_{i,b}, .., pred^{t-1}_{i,b}, pred^t_{i,b})}_{\text{Branch classification loss}} \tag{2}$$

Balancing losses: At the beginning of each training step $t > 1$, we initialize the model parameters using those of the model at step $t-1$. Hence, the different L_{KD} losses are much smaller than the new L_{CE}. This causes the model to learn considerably the new task and forget the old one *i.e* catastrophic forgetting, since the L_{KD} have minor impact in the final loss. To balance the rigidity vs. the elasticity of the model, we set λ_{old} to equalize the two terms at the beginning of training of each step: $\lambda_{old} = \frac{L_{CE}(iteration=1)}{L_{KD}(iteration=1)}$. This parameter takes into consideration the old model confidence on the old tasks and the nature of the tasks (number of images, preponderance of the background) which cause losses variations. Differently, previous KD-based methods (*right* in Figure 1) compute both L_{KD} and L_{CE} on the same output predicting all classes seen in tasks $\{1, .., t\}$ and balance them with a fixed weight per step which is a hyperparameter to find.

4 Experiments and results

4.1 Experimental settings

4.1.1 Datasets, protocols and scenarios

To evaluate the effectiveness of TILES, we use two commonly used datasets, Pascal-VOC and ADE20k, and follow the evaluation protocols and scenarios associated.

Pascal-VOC (Everingham et al. (2010)): contains 11,530 images segmented and labeled into 20 possible semantic classes plus a background class which is highly predominant. Indeed, the segmentation is object-centric and 56% of all pixels are labeled as background. Following previous work by Yang et al. (2022), we evaluate our models using both *disjoint* and *overlapped* protocols. In the *disjoint* protocol, each learning step contains a unique set of images, whose pixels belong to classes seen either in the current or previous step. Differently, in the *overlapped* protocol, each training step contains all the images that have at least one pixel of a novel class. Thus, images may contain pixels of classes that will be learned in the future. Notice that, since the sets of images are unique to each step in the *disjoint* protocol, the number of images used for training each step is drastically lower than in the *overlapped* protocol. The following three scenarios are evaluated for both protocols: $[19-1]$, $[15-5]$ and $[15-1]$ referring to [number of classes seen at step 1 - number of classes new seen at each step >1 until reaching the 20 classes of Pascal-VOC].

ADE20k (Zhou et al. (2019)): is made of 20,000 images, each segmented into 150 possible classes. The segmentation is exhaustive in this case: images are segmented into stuff classes (*e.g* , *wall, sky*) and things (*e.g* , *cars, person*). Like previous works (Shang et al. (2023); Zhang et al. (2023)), models are evaluated on ADE20k with the *overlapped* protocol since it is the more realistic one, on three scenarios: $[100-50]$, $[100-10]$ and $[50-50]$.

4.1.2 Implementation details

Similar to previous incremental approaches (Cermelli et al. (2020); Douillard et al. (2021)), random crops of size 512×512 pixels are used for training for both datasets. Moreover, the incremental learning configurations concerning the semantic classes learned at each step are retrieved from Cermelli et al. (2020). We optimize the models using SGD with a constant learning rate of 1e-3 throughout the steps. The models are trained with a batch size of 8, for 30 epochs for Pascal-VOC and for 64 epochs for ADE20k. The confidence weight γ_b and the loss weight on old tasks λ_{old} are computed online (*c.f* as explained in sections 3.5 and 3.6).

We compute three variants of TILES with suffixes -B, -S and -T denoting Base, Small or Tiny. In fact, we use a ViT (Dosovitskiy et al. (2021)) pre-trained on Imagenet (Deng et al. (2009)) for image classification (-B, -S or -T) as encoder. TILES-T uses the same encoder and decoder as Segmenter-T (Strudel et al. (2021)), resulting in 6.7M (million) parameters for one step and adding 0.4M parameters at each subsequent step as we add a branch decoder for each increment. For scalability reasons, TILES-S and TILES-B use the original ViT-B and ViT-S encoders respectively but adopt a custom decoder having a smaller hidden size of 256. This reduction helps limit the expansion of parameters for many step's scenarios. A dense layer is used to resize the resp. 384 and 768 hidden sizes output of ViT-S and ViT-B encoders to 256. Thus, TILES-S (TILES-B) uses 23.8M (88.2M) parameters at the initial step with additional 1.8M parameters at each step for both configurations. Table 5 details the number of parameters used for each scenario for TILES and previous methods.

For Incrementer (Shang et al. (2023)), we use the same values reported in the their paper, where ViT-B encoder and decoders are used. We perform the missing experiments for the ViT-S and ViT-T variants of Incrementer for comparison with TILES variants using the same encoder backbones.

4.1.3 Evaluation metrics

The evaluation metric usually used for CI-SS is the *mean Intersection over Union (mIoU)* which is the mean of the IoU per class. Results of the last step models are compared: a mIoU score is computed for classes

learned in the first step, another for those learnt in subsequent step(s), and the *all* column (in the tables hereafter) represents the average mIoU over all classes.

Knowledge Remaining (KR): Different approaches use different models and backbones, which are based on different *joint* performances (see Table 4): the non-incremental scenario in which the model learns from the whole data at once which can be considered as an upper bound for a given architecture and backbone. Hence, it is not fair to evaluate absolute *all mIoU* performances to evaluate the gain of the incremental strategies, while using different architectures. To this end, we propose an additional evaluation metric: the *Knowledge Remaining (KR)*. This metric is actually inspired by work done by Hoyer et al. (2022) in domain adaptation to fairly compare adaptation strategies while using different architectures and backbones. It is calculated as the ratio of the performance in the incremental setup by the performance in the *i.e joint* setup.

4.2 Results

Backbone	Method	#P (M)	**19-1** (2 steps)				**15-5** (2 steps)				**15-1** (6 steps)			
			1-19	**20**	**all**	**KR**	**1-15**	**16-20**	**all**	**KR**	**1-15**	**16-20**	**all**	**KR**
ResNet-101	MiB	66	70.2	22.1	67.8	87.6	75.5	49.4	69.0	89.1	35.1	13.5	29.7	38.4
	PLOP*	66	75.0	39.1	73.2	94.6	74.7	49.8	68.5	88.5	65.2	22.4	54.5	70.4
	SDR	66	69.1	32.6	67.4	87.1	75.4	52.6	69.9	90.3	44.7	21.8	39.2	50.6
	UCD	66	71.4	47.3	70.0	90.4	77.5	53.1	71.3	92.1	49.0	19.5	41.9	54.1
	RBC	66	77.3	55.6	76.2	98.1	76.6	52.8	70.6	90.9	69.5	38.4	61.7	76.7
	RCIL$^+$	66	-	-	-	-	78.8	52.0	72.4	93.5	70.6	23.7	59.4	76.7
	MicroSeg$^+$	66	78.8	14.0	75.7	97.4	80.4	52.8	73.8	**95.0**	80.1	36.8	69.8	89.8
	Bg_Adapt$^+$	66	78.2	42.2	76.4	**98.7**	-	-	-	-	77.6	45.9	79.4	**90.4**
SwinB	MicroSeg$^+$	104	79.0	25.3	76.4	92.4	81.9	54.0	75.2	90.9	80.5	40.8	71.0	85.9
	CoinSeg$^+$	104	79.6	70.2	79.1	95.6	80.5	70.8	78.2	**94.6**	76.8	57.2	72.2	87.3
	PLOP+NeST$^+$	104	81.5	44.8	79.8	**96.5**	82.1	63.2	77.6	93.8	82.7	52.5	75.5	**91.3**
ViT-B	RBC	102	80.2	38.8	78.1	95.4	78.9	62.0	74.7	91.2	75.9	40.2	67.0	81.8
	MiB	102	79.9	47.7	78.3	95.6	78.6	63.1	74.7	91.2	72.6	23.1	60.2	73.5
	Incrementer	102	82.5	61.0	81.4	**99.4**	82.5	69.3	79.2	96.7	79.6	59.7	74.6	91.1
	TILES-B	90 to 97	81.0	53.4	79.6	**99.4**	81.9	69.4	78.8	**98.4**	79.8	55.4	73.7	**92.0**
ViT-S	Incrementer	26	76.4	47.5	75.0	95.4	74.0	56.1	69.5	88.4	72.6	46.1	66.0	84.0
	TILES-S	26 to 33	79.1	53.3	77.8	**99.0**	79.7	65.3	76.1	**96.8**	77.1	47.8	69.8	**88.8**
ViT-T	Incrementer	7	67.2	39.7	65.8	90.4	56.7	38.0	52.0	71.4	53.3	20.0	45.0	61.8
	TILES-T	7 to 9	73.6	38.0	71.8	**98.6**	71.7	47.2	65.6	**90.1**	66.3	21.1	55.0	**75.5**

Table 1: CI performances (mIoU and KR in %) on Pascal-VOC for different *overlapped* scenarios. Best KR per backbone per scenario in **bold**. *Reported by Yang et al. (2022). $^+$ mIoU computation is biased by considering background IoU. **#P** is the number of parameters (in millions).

Backbone	Method	#P (M)	**19-1** (2 steps)				**15-5** (2 steps)				**15-1** (6 steps)			
			1-19	**20**	**all**	**KR**	**1-15**	**16-20**	**all**	**KR**	**1-15**	**16-20**	**all**	**KR**
ResNet-101	MiB	66	69.6	25.6	67.6	87.3	71.8	43.3	64.7	83.6	46.2	12.9	37.9	49.0
	PLOP*	66	75.1	38.2	73.2	94.6	66.5	39.6	59.8	76.6	49.0	13.8	40.2	51.9
	SDR	66	69.9	37.3	68.4	88.4	73.5	47.3	67.2	86.8	59.2	12.9	48.1	62.1
	UCD	66	73.4	33.7	71.5	92.4	71.9	49.5	66.2	85.6	53.1	13.0	42.9	55.4
	RBC	66	76.4	45.8	74.9	**96.4**	75.1	49.7	68.8	**88.5**	61.7	19.5	51.1	65.8
	RCIL$^+$	66	-	-	-	-	75.0	42.8	67.3	87.0	66.1	18.2	54.7	**70.7**
ViT-B	RBC	102	80.9	42.1	79.0	96.4	77.7	59.1	73.1	89.2	69.0	28.4	58.9	71.9
	MiB	102	80.6	45.2	78.8	96.3	75.0	59.9	71.2	87.0	66.7	26.3	56.6	69.1
	Incrementer	102	82.4	64.2	81.5	**99.5**	81.6	62.2	76.8	**93.8**	81.4	57.1	75.3	**91.9**
	TILES-B	90 to 97	80.5	55.2	79.2	98.9	77.6	49.3	70.5	88.0	74.1	35.7	64.5	80.5
ViT-S	Incrementer	26	76.4	19.9	73.6	93.6	75.2	26.9	63.1	80.3	71.9	38.6	63.6	80.9
	TILES-S	26 to 33	79.1	51.5	77.7	**98.9**	76.1	47.7	69.0	**87.8**	73.9	35.1	64.2	**81.7**
ViT-T	Incrementer	7	68.9	13.6	66.1	90.8	65.7	22.9	55.0	75.5	59.1	14.2	47.9	65.8
	TILES-T	7 to 9	72.9	44.5	71.5	**98.2**	68.5	37.4	60.7	**83.4**	61.6	26.5	51.7	**71.0**

Table 2: CI performances (mIoU and KR in %) on Pascal-VOC for different *disjoint* scenarios. Best KR per backbone per scenario in **bold**. *Reported by Yang et al. (2022). $^+$ mIoU computation is biased by considering background IoU. **#P** is the number of parameters (in millions).

Backbone	Method	#P (M)	**100-50** (2 steps)				**100-10** (6 steps)				**50-50** (3 steps)			
			1-100	**101-150**	**all**	**KR**	**1-100**	**101-150**	**all**	**KR**	**1-50**	**51-150**	**all**	**KR**
ResNet-101	MiB	66	37.9	27.9	34.6	88.9	31.8	14.1	25.9	65.6	35.5	22.9	27.0	69.4
	PLOP*	66	29.8	4.2	22.2	57.1	32.1	2.8	22.3	57.3	19.2	0.4	6.6	17.0
	SDR	66	37.4	24.8	33.2	85.3	28.9	7.3	21.7	55.8	40.9	23.8	29.5	75.8
	UCD	66	40.4	27.3	36.0	**92.5**	28.6	12.4	23.2	59.6	39.3	22.2	27.9	71.7
	RBC	66	42.9	21.5	35.8	92.0	39.0	21.7	33.2	85.3	49.6	26.3	34.1	**87.7**
	RCIL	66	42.3	18.8	34.5	88.6	39.3	17.7	32.1	82.5	48.3	25.0	32.5	83.5
	MicroSeg$^+$	66	40.2	18.8	33.1	85.1	41.5	21.6	34.9	89.7	48.6	24.8	32.9	84.6
	Bg_Adapt$^+$	66	42.0	23.0	35.7	91.8	41.1	23.1	35.2	**90.5**	47.9	26.5	33.7	86.6
SwinB	MicroSeg$^+$	104	41.1	24.1	35.4	92.7	41.0	22.6	34.8	91.1	49.8	23.9	32.5	85.1
	CoinSeg$^+$	104	41.6	26.7	36.6	93.4	42.1	24.5	36.2	**92.3**	49.0	28.9	35.6	90.8
	PLOP+NeST$^+$	104	43.5	26.5	37.9	**96.9**	41.7	24.2	35.9	91.8	50.6	28.9	36.2	**92.6**
ViT-B	MiB	102	46.4	35.0	42.6	88.6	43.0	30.8	38.9	80.9	52.2	35.6	41.1	85.4
	Incrementer	102	49.4	35.6	44.8	93.1	48.5	34.6	43.6	**91.3**	56.2	37.8	43.9	91.3
	TILES-B	90 to 97	51.9	39.3	47.7	**99.2**	50.6	18.6	39.9	82.7	58.9	41.2	47.1	**97.9**
ViT-S	Incrementer	26	48.7	29.9	42.4	90.6	42.3	15.1	33.2	70.9	55.7	32.6	40.3	86.1
	TILES-S	26 to 33	50.2	34.1	44.8	**98.2**	46.7	15.8	36.4	**79.8**	55.5	37.4	43.4	**95.2**
ViT-T	Incrementer	7	47.6	8.6	34.6	89.9	34.1	8.6	24.6	63.9	49.9	23.4	32.2	83.6
	TILES-T	7 to 9	43.1	24.8	37.0	**96.1**	36.8	16.8	30.1	**78.2**	50.1	28.1	35.4	**91.9**

Table 3: CI performances (mIoU and KR in %) on ADE20k for different *overlapped* scenarios. Best KR per backbone per scenario in **bold**. *Reported by Yang et al. (2022). $^+$ mIoU computation is biased by considering background IoU. **#P** is the number of parameters (in millions).

We compare in Table 1, Table 2 and Table 3 different state-of-the-art CI-SS methods on the predefined scenarios and protocols of Pascal-VOC and ADE20k datasets. Since these methods are based on different architectures and backbones, their corresponding *joint* setups have different values such as illustrated in table Table 4. Moreover, it is important to highlight that the memory footprint varies a lot depending on the backbone, which has not been considered by previous CI-SS methods, by comparing absolute $mIoU$ while having different *joint* values and model sizes. For these reasons and to ensure a fair comparison between the different incremental techniques, we focus on comparing the *knowledge remaining* (KR) which evaluates the capacity of retaining old knowledge while learning new tasks, regardless of the used architecture. The goal here is to fairly compare the different approaches, and to propose a new light-weight option for use-cases with severe memory constraints. Note that methods with $^+$ consider the background in their $mIoU$ computation which distorts results since generally all methods have very good background IoU.

On the one hand, we can notice in the three tables that previous ViT based approaches assure in general better absolute and KR performances than the CNN based approaches. In addition to the incremental techniques used, this is also due to the fact that visual transformers are better continual learners than CNNs such as discussed in sec. 2.3, to the better *joint* values and to the bigger memory footprint used. In fact, using more parameters improves the models' capacity to encompass the old and the new knowledge with minimum forgetting. Nevertheless, despite using less parameters than previous ViT based approaches thanks to the smaller decoder (see sec. 4.1.2 for details), TILES-B is able to achieve interesting results and even outperforms them in the more realistic *overlapped* scenario for both datasets.

On the other hand, we compare Incrementer (Shang et al. (2023)) with TILES using smaller backbones to study behaviors when severe memory constraints are applied. We choose Incrementer as a reference because it has best results among most scenarios on both ADE20k and Pascal-VOC datasets, and because it uses the same semantic segmentation base method as TILES: Segmenter (Strudel et al. (2021)). We can see that fine-tuning parameters when using smaller backbones results in a big performance drop since the limited number of parameter is not able to encompass both old and new knowledge while ensuring a good rigidity vs. elasticity trade-off. As an alternative, TILES-S and TILES-T provide always better performances, with a KR p.p ranging from: i) for ViT-S from 0.8 to 8.4 for Pascal-VOC and from 7.6 to 9.1 for ADE20k, and ii) for ViT-T from 5.2 to 18.4 for Pascal-VOC and from 6.2 to 14.3 for ADE20k. Indeed, the performance gap between TILES and Incrementer is heightened for smaller backbones. These improvements are especially thanks to the adopted expanding mechanism which seems necessary to learn new tasks without big forgetting, while adding a limited number of parameters at each step. Indeed, depending on the balance between the old and new losses, applying smaller backbones to Incrementer seems to either cause catastrophic forgetting or to limit learning new tasks.

Model	Backbone	Pascal-VOC	ADE20k
Deeplab-v3	ResNet-101	77.4	38.9
Deeplab-v3	SwinB	82.7	39.1
Incrementer	ViT-B	81.9	48.1
TILES-B	ViT-B	80.1	48.1
Incrementer	ViT-S	78.6	46.8
TILES-S	ViT-S	78.6	45.6
Incrementer	ViT-T	72.8	38.5
TILES-T	ViT-T	72.8	38.5

Table 4: Performance (mIoU in %) of the *joint* setting of different models and backbones for Pascal-VOC and ADE20k datasets.

Model	Backbone	1 step	2 steps	3 steps	6 steps
Deeplab-v3	ResNet-101	66	66	66	66
Deeplab-v3	SwinB	104	104	104	104
Incrementer	ViT-B	102	102	102	102
TILES-B	ViT-B	88	90	92	97
Incrementer	ViT-S	26	26	26	26
TILES-S	ViT-S	24	26	27	33
Incrementer	ViT-T	6.7	6.7	6.7	6.7
TILES-T	ViT-T	6.7	7.1	7.5	8.7

Table 5: Number of parameters (in million) used with relation to the number of steps of the scenario.

Note that, despite adopting an expansion mechanism, the number of parameters added in all these cases is paramount (see Table 5 for details). In fact, TILES-B uses less parameters than other ViT-B and SwinB based approaches thanks to the light-weight decoder used. For TILES-S and TILES-T, we can prove efficiency by providing major improvements compared to Incrementer while adding 1.8M and 0.4M parameters per step respectively.

It is also important to highlight that TILES-S achieves the same absolute results as Incrementer (ViT-B) for the $[100-50]$ and $[50-50]$ ADE20k protocols despite displaying different *joint* performances and while using up to 4 times fewer parameters (25 vs. 102) thanks to the adopted expanding mechanism along with the corresponding losses and branch merging module. Similarly, TILES-T achieves and even surpasses CNN based absolute approaches in some cases, despite the smaller *joint* performances and while using up to 9 times fewer parameters (7.1 vs. 66). TILES-T also provides similar or better absolute performances compared to SwinB-based methods while using up to 14 times fewer parameters (7.1 vs. 104) for the same ADE20k protocols. This proves the over-allocation of parameters by previous CI-SS methods and the importance of studying the efficacy of models for highly constrained tasks.

Moreover, despite being an expanding method, TILES-T shows a good scalability with the number of increments (239 steps would be necessary to surpass the $102M$ parameters used by previous ViT-based approaches). Therefore, TILES-T is convenient for applications with extremely severe memory constraints or needing a large number of increments. Besides, TILES-S shows closest performances to previous ViT-based methods while keeping the number of parameters lower until an expansion of 43 increments. Thus, TILE-S is adapted to applications requiring a lower number of increments, and where improved performance is more important than severe memory constraints.

4.3 Ablation study

TILES-S using Segmenter-S decoder: In Table 6, we compare ADE20k results using TILES-S but with two different decoders: Segmenter-S decoders which add 4M parameters at each step and our light-weight decoders adding $1.8M$ parameters per step. It demonstrates that, despite the much bigger Segmenter-S decoder, the absolute $mIoU$ improvement compared to the TILES-S decoder is null or minor compared to

the memory footprint increase (ranging from 15% to 48%). This proves that the adopted decoder architecture is sufficient to encompass the new knowledge while adding a limited memory footprint at each step.

Method	**#P** (M)	**100-50** (2 steps)				**100-10** (6 steps)				**50-50** (3 steps)				**Joint**
		1-100	**101-150**	**all**	**KR**	**1-100**	**101-150**	**all**	**KR**	**1-50**	**51-150**	**all**	**KR**	
TILES-S (SS-D)	30 to 46	49.8	35.2	44.9	95.9	42.1	31.6	38.6	82.5	56.1	41.2	43.6	93.2	46.8
TILES-S (ours)	26 to 33	50.2	34.1	44.8	98.2	46.7	15.8	36.4	79.8	55.5	37.4	43.4	95.2	45.6

Table 6: Influence of decoder architecture on TILES-S performance (mIoU and KR in %) on 3 *overlapped* scenarios on ADE20k. SS-D denotes Segmenter-S decoder. **#P** is the number of parameters (in millions).

Balancing losses: Table 7 (A1 and TILES) shows that loss balancing in TILES ($\lambda_{old} \neq 1$ as detailed) is beneficial (+1.4 p.p. $mIoU$ for TILES-T on Pascal-VOC $[15-5]$ *disjoint*). We can notice that equalizing losses at the beginning, alleviates forgetting and creates a better rigidity vs. elasticity trade-off. This can be even more important for several-step scenarios where the model forgets old knowledge at each increment.

Ablation	λ_{old}	L_{KD}	γ_b	L_{BC}	1-15	16-20	all
A1	$\mathbf{= 1}$	on new background pixels	$\neq 1$	✓	67.2	55.0	64.2
A2	$\neq 1$	***on all pixels***	$\neq 1$	✓	71.5	42.6	64.3
A3	$\neq 1$	on new background pixels	$\mathbf{= 1}$	✓	71.0	45.1	64.5
A4	$\neq 1$	on new background pixels	$\neq 1$	✗	72.4	42.7	65.0
TILES	$\neq 1$	on new background pixels	$\neq 1$	✓	71.7	47.2	**65.6**

Table 7: Ablation study of loss balancing λ_{old} (A1), applying L_{KD} on all pixels or only on new background pixels (A2), probability compensation weight γ_b (A3) and binary classification loss L_{BC} (A4) on TILES-T performance (mIoU in %) on Pascal-VOC [15-5] *disjoint* scenario.

Applying L_{KD} on all pixels: Table 7 (A2 and TILES) proves that applying the knowledge distillation loss L_{KD} only on new background pixels improves significantly the new classes performances compared to applying this loss to whole image. In fact, this technique provides more elasticity on the pixels corresponding to new classes by retaining knowledge only on the new background pixels that could be potentially old classes learnt in old steps.

Compensation weight of probability for branch merging: Table 7 (A3 and TILES) shows that compensating the branch prediction probabilities is beneficial as it gives more weight to ignored branches *i.e* in this case the new branch as the training-set for this step is smaller.

Impact of branch classification loss: Since different decoders are used for different tasks for TILES, semantically close concepts could be learnt by separate decoders causing a confusion between them and thus a performance drop (see sec. 3.6). Table 7 (A4 and TILES) shows the big degradation of new classes performances if the branch classification loss between the branches L_{BC} is retrieved.

5 Conclusion

In this work, we elaborated a complete comparison across previous SOTA CI-SS methods based on different backbones. While different performance trends can be remarked, all these methods use quite large memory footprint without any study about their efficiency wrt this aspect. Thus, we proposed TILES, a new CI learning method based on a ViT architecture for SS and specifically convenient for use cases with severe memory constraints. Indeed, we demonstrated a big performance drop for a previous SOTA method when smaller backbones are used, unlike TILES which is more adapted for these cases. Moreover, TILES can even outperform previous SOTA models which use much bigger backbones (up to 14× bigger) when comparing absolute performance. We hope that this work stimulates the AI community's interest to study models efficiency for CI-SS.

References

Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9230–9239, 2020. doi: 10.1109/CVPR42600.2020.00925.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. URL http://arxiv.org/abs/1706.05587.

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, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. PLOP: Learning without Forgetting for Continual Semantic Segmentation. In *CVPR*, Nashville, United States, 2021. URL https://hal.archives-ouvertes.fr/hal-03503831.

Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. URL http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL http://arxiv.org/abs/1503.02531.

Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9924–9935, June 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017.

Duo Li, Guimei Cao, Yunlu Xu, Zhanzhan Cheng, and Yi Niu. Technical report for ICCV 2021 challenge sslad-track3b: Transformers are better continual learners. *CoRR*, abs/2201.04924, 2022. URL https://arxiv.org/abs/2201.04924.

Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew Bagdanov, Shangling Jui, and Joost Weijer. Generative feature replay for class-incremental learning. In *CVPR Workshop*, pp. 915–924, 06 2020. doi: 10.1109/CVPRW50498.2020.00121.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

A. Maracani, U. Michieli, M. Toldo, and P. Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7006–7015, Los Alamitos, CA, USA, Oct 2021. IEEE Computer Society. doi: 10.1109/ICCV48922.2021.00694. URL https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00694.

Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

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 (CVPR)*, pp. 1114–1124, June 2021.

Minh Hieu Phan, The-Anh Ta, Son Lam Phung, Long Tran-Thanh, and Abdesselam Bouzerdoum. Class similarity weighted knowledge distillation for continual semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16866–16875, June 2022.

Chao Shang, Hongliang Li, Fanman Meng, Qingbo Wu, Heqian Qiu, and Lanxiao Wang. Incrementer: Transformer for class-incremental semantic segmentation with knowledge distillation focusing on old class. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7214–7224, June 2023.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental Learning of Object Detectors without Catastrophic Forgetting. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 3420–3429, Venice, Italy, October 2017. doi: 10.1109/ICCV.2017.368. URL https://hal.inria.fr/hal-01573623.

Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. In *International Conference on Computer Vision (ICCV)*, Virtual, France, October 2021. URL https://hal.archives-ouvertes.fr/hal-03481207.

Zhengyuan Xie, Haiquan Lu, Jia-wen Xiao, Enguang Wang, Le Zhang, and Xialei Liu. Early preparation pays off: New classifier pre-tuning for class incremental semantic segmentation. In *European Conference on Computer Vision*, pp. 183–201, 2024.

Shipeng Yan, Jiangwei Xie, and Xuming He. DER: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Guanglei Yang, Enrico Fini, Dan Xu, Paolo Rota, Mingli Ding, Moin Nabi, Xavier Alameda-Pineda, and Elisa Ricci. Uncertainty-aware contrastive distillation for incremental semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. doi: 10.1109/tpami.2022.3163806. URL https://doi.org/10.1109%2Ftpami.2022.3163806.

Anqi Zhang and Guangyu Gao. Background adaptation with residual modeling for exemplar-free class-incremental semantic segmentation. 2024.

Chang-Bin Zhang, Jia-Wen Xiao, Xialei Liu, Ying-Cong Chen, and Ming-Ming Cheng. Representation compensation networks for continual semantic segmentation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022a.

Zekang Zhang, Guangyu Gao, Zhiyuan Fang, Jianbo Jiao, and Yunchao Wei. Mining unseen classes via regional objectness: A simple baseline for incremental segmentation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24340–24353. Curran Associates, Inc., 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/99b419554537c66bf27e5eb7a74c7de4-Paper-Conference.pdf.

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 (ICCV)*, pp. 843–853, October 2023.

Hanbin Zhao, Hui Wang, Yongjian Fu, Fei Wu, and Xi Li. Memory efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–12, 05 2021. doi: 10.1109/TNNLS.2021.3072041.

Hanbin Zhao, Fengyu Yang, Xinghe Fu, and Xi Li. Rbc: Rectifying the biased context in continual semantic segmentation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision – ECCV 2022*, pp. 55–72, Cham, 2022. Springer Nature Switzerland.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *Int. J. Comput. Vis.*, 127(3):302–321, 2019. URL `http://dblp.uni-trier.de/db/journals/ijcv/ijcv127.html#ZhouZPXFBT19`.