RECALL: Replay-based Continual Learning in Semantic Segmentation

Andrea Maracani,* Umberto Michieli,* Marco Toldo*†, Pietro Zanuttigh

Department of Information Engineering, University of Padova

andreamaracani@gmail.com, {umberto.michieli,toldomarco,zanuttigh}@dei.unipd.it

Abstract

Deep networks allow to obtain outstanding results in semantic segmentation, however they need to be trained in a single shot with a large amount of data. Continual learning settings where new classes are learned in incremental steps and previous training data is no longer available are challenging due to the catastrophic forgetting phenomenon. Existing approaches typically fail when several incremental steps are performed or in presence of a distribution shift of the background class. We tackle these issues by recreating no longer available data for the old classes and outlining a content inpainting scheme on the background class. We propose two sources for replay data. The first resorts to a generative adversarial network to sample from the class space of past learning steps. The second relies on web-crawled data to retrieve images containing examples of old classes from online databases. In both scenarios no samples of past steps are stored, thus avoiding privacy concerns. Replay data are then blended with new samples during the incremental steps. Our approach, RECALL, outperforms state-of-the-art methods.

1. Introduction

A common requirement for many machine learning applications is the ability to learn a sequence of tasks in multiple incremental steps, *e.g.*, progressively introducing novel classes to be recognized, instead of using a single-shot training procedure on a large dataset [34]. This problem has been widely studied in image classification and many methods propose to alleviate the forgetting of previous tasks and intransigence of learning new ones [22, 35, 45]. When the model is exposed to samples of novel classes and is trained on them without additional provisions, the optimization leads to the so-called *catastrophic forgetting* phenomenon [36, 16], *i.e.*, knowledge about previously seen classes tends to be lost.

Incremental learning on dense tasks (*e.g.*, semantic segmentation), where pixel-wise predictions are performed, has
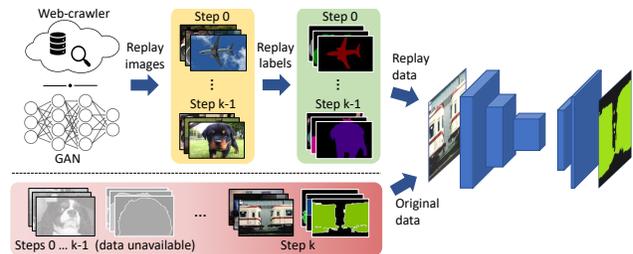
Figure 1: Replay images of previously seen classes are retrieved by a web crawler or a generative network and further labeled. Then, the network is incrementally trained with a mixture of new and replay data.

only recently been explored and the first experimental studies show that catastrophic forgetting is even more severe than on the classification task [29, 31]. Current approaches for class-incremental semantic segmentation re-frame knowledge distillation strategies inspired by previous works on image classification [29, 5, 23, 31]. Although they partially alleviate forgetting, they often fail when multiple incremental steps are performed or when background shift [5] (*i.e.*, change of statistics of the background across learning steps, as it incorporates old or future classes) occurs.

In this paper, we follow a completely different strategy and, instead of distilling knowledge from a teacher model (*i.e.*, the old one) to avoid forgetting, we propose to generate samples of old classes by using replay strategies. We propose RECALL (REplay in ContinuAL Learning), a method that re-creates representations of old classes and mixes them with the available training data, *i.e.*, containing novel classes being learned (see Fig. 1). To reduce background shift we introduce a self-inpainting strategy that re-assigns the background region according to predictions of the previous model.

To generate representations of past classes we pursue two possible directions. The first is based on a pre-trained generative model, *i.e.*, a Generative Adversarial Network (GAN) [15] conditioned to produce samples of an input class. The GAN has been trained beforehand on a dataset different than the target one (we chose ImageNet as it comprehends a wide variety of classes and domains), thus requiring a Class Mapping Module to perform the translation between the two label

spaces. The second strategy, instead, is based on crawling images from the web, querying the class names to drive the search. Both approaches allow to retrieve a large amount of weakly labeled data. Finally, we generate pseudo-labels for semantic segmentation using a side labeling module, which requires only minimal extra storage.

Our main contributions are: 1) we propose RECALL, which is the first approach to use replay data in continual semantic segmentation; 2) to the best of our knowledge, we are the first to introduce the webly-supervised paradigm in continual learning, showing how we can extract useful clues from extremely weakly supervised and noisy samples; 3) we devise a background inpainting strategy to generate pseudo-labels and overcome the background shift; 4) we achieve state-of-the-art results on a wide range of scenarios, especially when performing multiple incremental steps.

2. Related Works

Continual Learning (CL). Deep neural networks witnessed remarkable improvements in many fields; however, such models are prone to catastrophic forgetting when they are trained to continuously improve the learned knowledge (*e.g.*, new categories) from progressively provided data [16]. Catastrophic forgetting is a long-standing problem [36, 14] which has been recently tackled in a variety of visual tasks such as image classification [22, 35, 26, 43, 33], object detection [40, 25] and semantic segmentation [29, 31, 5, 23].

Current techniques can be grouped into four main (non mutually exclusive) categories [24]: namely, dynamic architectures, regularization-based, rehearsal and generative replays. Dynamic architectures can be explicit [42, 26], if new network branches are grown, or implicit [13, 37], if some network weights are available for certain tasks only. Regularization-based approaches mainly propose to compute some penalty terms to regularize training (*e.g.*, based on the importance of the weights for a specific task) [22, 45] or to distill knowledge from the old model [40, 26, 31]. Rehearsal approaches store a set of raw samples of past tasks into memory, which are then used while training for the new task [35, 28]. Finally, generative replays approaches [39, 43, 21] rely on generative models typically trained on the same data distribution, which are later used to generate artificial samples to preserve previous knowledge. Generative models are usually GANs [39, 43, 17] or auto-encoders [21]. In this work, we employ two kinds of generative replays: either resorting to a standard pre-trained GAN or to web-crawled images to avoid forgetting, without storing any of the samples related to previous tasks. When using the generative model, differently from previous works on continual image classification, we do not select real exemplars as anchor to support the learned distribution [17] nor we train or fine-tune the GAN architecture on the current data distribution [39, 17, 43], thus reducing memory and computation time.

CL in Semantic Segmentation. Semantic segmentation has experienced a wide research interest in the past few years and deep networks have achieved noticeable results on this task. Current techniques are based on the auto-encoder structure firstly employed by FCN [27] and subsequently improved by many approaches [6, 7, 46, 44]. Recently, increasing attention has been devoted to class-incremental semantic segmentation [29, 5, 30, 11] to learn new categories from new data. In [29, 31] the problem is first introduced and tackled with regularization approaches such as parameters freezing (*e.g.*, fixing the encoder after the initial training stage) and knowledge distillation. In [23] knowledge distillation is coupled with a class importance weighting scheme to emphasize gradients on difficult classes. Cermelli *et al.* [5] study the distribution shift of the background class. In [11] long- and short-range spatial relationships at feature level are preserved. In [30] the latent space is regularized to improve class-conditional features separation.

Webly-Supervised Learning is an emerging paradigm in which large amounts of web data is exploited for learning CNNs [8, 10, 32]. Recently, it was employed also in semantic segmentation to provide a plentiful source of both images [20, 38] and videos [19] with weak image-level class labels during training. The most active research directions are devoted toward understanding how to query images, how to filter and exploit them (*e.g.*, assigning pseudo-labels). To our knowledge, however, webly-supervised learning has not yet been explored in continual learning as a replay strategy.

3. Problem Formulation

The semantic segmentation task consists in labeling each pixel in an image by assigning it to a class from a collection of possible semantic classes $\mathcal{C}$, which typically also comprises a special background category that we denote as b. More formally, given an image $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$, we aim at producing a map $\hat{\mathbf{Y}} \in \mathcal{Y} \subset \mathcal{C}^{H \times W}$ that is a prediction of the ground truth map $\mathbf{Y}$. This is nowadays usually achieved by using a suitable deep learning model $M : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}|}$, commonly made by a feature extractor E followed by a decoding module D, *i.e.*, $M = D \circ E$.

In standard supervised learning, the model is learned in a single shot over a training set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$, available in its complete form to the training algorithm. In class-incremental learning, instead, we assume that the training is performed in multiple steps and only a subset of training data is made available to the algorithm at each step $k = 0, ..., K$. More in detail, we start from an initial step $k = 0$ where only training data concerning a subset of all the classes $\mathcal{C}_0 \subset \mathcal{C}$ is available (we assume that $b \in \mathcal{C}_0$). We denote with $M_0 : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}_0|}$, $M_0 = D_0 \circ E_0$ the model trained after this initial step. Moving to a generic step k, a new set of classes $\mathcal{C}_k$ is added to the class collection $\mathcal{C}_{0 \to (k-1)}$ learned up to that point, resulting in an expanded

set of learnable classes $\mathcal{C}_{0\to k} = \mathcal{C}_{0\to(k-1)} \cup \mathcal{C}_k$ (we assume $\mathcal{C}_{0\to(k-1)} \cap \mathcal{C}_k = \varnothing$). The model after the k-th step of training is $M_k : \mathcal{X} \mapsto \mathbb{R}^{H\times W\times|\mathcal{C}_{0\to k}|}$, where $M_k = D_k \circ E_0$, since in our approach the encoder E_0 is not trained during the incremental steps and only the decoder is updated [29].

Two main continual scenarios have been proposed (see [29, 30, 5] for a more detailed description) and we tackle both in a unified framework.

Disjoint setup: in the initial step all the images in the training set with at least one pixel belonging to a class of $\mathcal{C}_0$ (except for b) are assumed to be available. We denote with $\mathcal{Y}_{\mathcal{C}_0\cup\{b\}} \subset \mathcal{C}_0^{H\times W}$ the corresponding output space where labels can only belong to $\mathcal{C}_0$, while all the pixels not pertaining to these classes are assigned to b. The incremental partitions are built as *disjoint* subsets of the whole training set. The training data associated to k-th step, $\mathcal{T}_k \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_k\cup\{b\}}$, contains only images corresponding to classes in $\mathcal{C}_k$ with just classes of step k annotated (possible old classes are labeled as b), and is disjoint w.r.t. previous and past partitions.

Overlapped setup: in the first phase we select the subset of training images having only $\mathcal{C}_0$-labeled pixels. Then, the training set at each incremental step contains *all* the images with labeled pixels from $\mathcal{C}_k$ i.e. $\mathcal{T}_k \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_k\cup\{b\}}$. Similarly to the initial step, labels are limited to semantic classes in $\mathcal{C}_k$, while remaining pixels are assigned to b.

In both setups, b undergoes a semantic shift at each step, as pixels of ever changing class sets are assigned to it.

4. General Architecture

In the standard setup, the segmentation model M is trained with annotated samples from a training set $\mathcal{T}$. Data should be representative of the task we would like to solve, meaning that multiple instances of all the considered semantic classes $\mathcal{C}$ should be available in the provided dataset for the segmentation network to properly learn them. Once $\mathcal{T}$ has been assembled, the cross-entropy objective is commonly employed to optimize the weights of M:

$$\mathcal{L}_{ce}(M; \mathcal{C}, \mathcal{T}) = -\frac{1}{|\mathcal{T}|} \sum_{\mathbf{X},\mathbf{Y}\in\mathcal{T}} \sum_{c\in\mathcal{C}} \mathbf{Y}[c]\cdot\log\left(M(\mathbf{X})[c]\right) \quad (1)$$

In the incremental learning setting, when performing an incremental training step k only samples related to new classes $\mathcal{C}_k$ are assumed to be at our disposal. Following the simplest approach, we could initialize our model's weights from the previous step (M_{k-1}, $k \geq 1$) and learn the segmentation task over classes $\mathcal{C}_{0\to k}$ by optimizing the standard objective $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\to k}, \mathcal{T}_k)$ with data from the current training partition $\mathcal{T}_k$. However, simple fine-tuning leads to catastrophic forgetting, being unable to preserve previous knowledge.

Architecture of Replay Block. To cope with this issue, we opt for a replay strategy. Our goal is to retrieve task-related knowledge of past classes to be blended into the ongoing incremental step, all without accessing training data

of previous iterations. To this end, we introduce a Replay Block, whose target is twofold. First, it has to provide images resembling instances of classes from previous steps, whether generating them from scratch or retrieving them from an available alternative source (*e.g.*, a web database). Second, it has to obtain reliable semantic labels of those images, by resorting to learned knowledge from past steps.

The Replay Block's image retrieval task is executed by what we call Source Block:

$$S : \mathcal{C}_k \mapsto \mathcal{X}_{\mathcal{C}_k}^{rp} \quad (2)$$

This module takes in input a set of classes $\mathcal{C}_k$ (background excluded) and provides images whose semantic content can be ascribed to those categories (*e.g.*, $\mathbf{X}^{rp} \in \mathcal{X}_{\mathcal{C}_k}^{rp}$). We adopt two different solutions for the Source Block, namely GAN and web-based techniques, both detailed in Sec. 5.

The Source Block provides unlabeled image data (if we exclude the weak image-level classification labels), and for this reason we introduce an additional Label Evaluation Block $\{L_{\mathcal{C}_k}\}_{\mathcal{C}_k\subset\mathcal{C}}$, which aims at annotating examples provided by the replay module. This block is made of separate instances $L_{\mathcal{C}_k} = D_{\mathcal{C}_k}^H \circ E_0$, each denoting a segmentation model to classify a specific set of semantic categories $\mathcal{C}_k \cup \{b\}$ (*i.e.*, the classes in $\mathcal{C}_k$ plus the background) :

$$L_{\mathcal{C}_k} : \mathcal{X}_{\mathcal{C}_k} \mapsto \mathbb{R}^{H\times W\times(|\mathcal{C}_k\cup\{b\}|)} \quad (3)$$

All $L_{\mathcal{C}_k}$ modules share the encoder section E_0 from the initial training step, so that only a minimal portion of the segmentation network (*i.e.*, $D_{\mathcal{C}_k}^H$, which accounts for only few parameters, see Sec. 7.1) is stored for each block's instance. Notice that a single instance recognizing all classes could be used, leading to an even more compact representation, but it experimentally led to worse performance.

Provided that S and $L_{\mathcal{C}_k}$ are available, replay training data can be collected for classes in $\mathcal{C}_k$. A query to S outputs a generic image example $\mathbf{X}_{\mathcal{C}_k}^{rp} = S(\mathcal{C}_k)$, which is then associated to its prediction $\mathbf{Y}_{\mathcal{C}_k}^{rp} = \underset{c\in\mathcal{C}_k\cup\{b\}}{\arg\max} L_{\mathcal{C}_k}(\mathbf{X}_{\mathcal{C}_k}^{rp})[c]$. By retrieving multiple replay examples, we build a replay dataset $\mathcal{R}_{\mathcal{C}_k} = \{(\mathbf{X}_{\mathcal{C}_k}^{rp}, \mathbf{Y}_{\mathcal{C}_k}^{rp})_n\}_{n=1}^{N_r}$, where N_r is a fixed hyperparameter empirically set (see section 6).

Background Self-Inpainting. To deal with the background shift phenomenon, we propose a simple yet effective inpainting mechanism to transfer knowledge from the previous model into the current one. While the replay block re-creates samples of previously seen classes, background inpainting acts on background regions of current samples reducing the background shift and at the same time bringing a regularization effect similar to knowledge distillation [29, 5], although its implementation is quite different. At every step k with training set $\mathcal{T}_k$, we take the background region of each ground truth map and we label it with the associated prediction from the previous model M_{k-1} (see Fig. 3). We call it *background inpainting* since the background regions in label maps are changed according to a self-teaching scheme
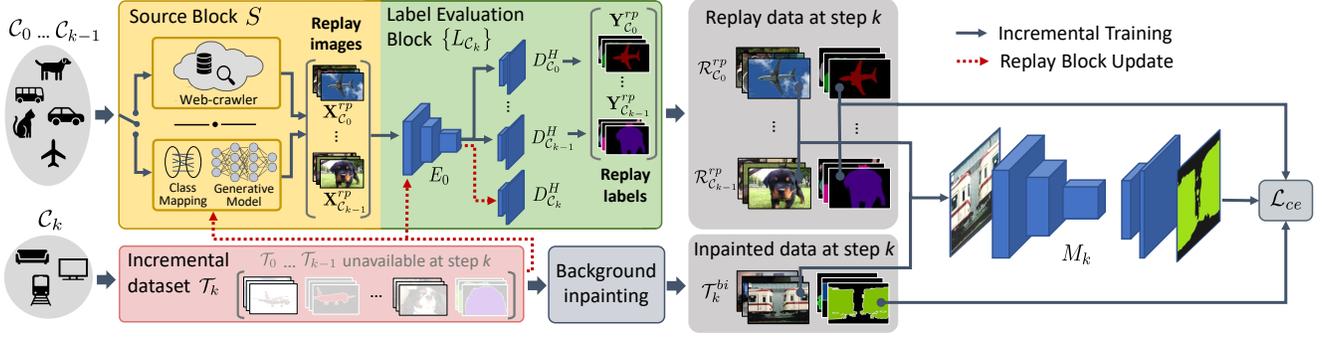
Figure 2: Overview of the proposed RECALL: class labels from past incremental steps are provided to a Source Block, either a web crawler or a pre-trained conditional GAN, which retrieves a set of unlabeled replay images for the past semantic classes. Then, a Label Evaluation Block produces the missing annotations. Finally, the segmentation network is incrementally trained with a replay-augmented dataset, composed of both new classes data and replay data.
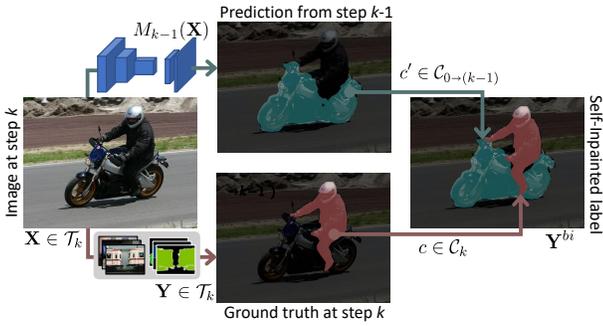
Figure 3: Background self-inpainting process.

based on the prediction of the old model. More formally, we replace each original label map $\mathbf{Y}$ available at step $k > 0$ with its inpainted version $\mathbf{Y}^{bi}$:

$$\mathbf{Y}^{bi}[h,w] = \begin{cases} \mathbf{Y}[h,w] & \text{if } \mathbf{Y}[h,w] \in \mathcal{C}_k \\ \arg\max_{c \in \mathcal{C}_{0\to k-1}} M_{k-1}(\mathbf{X})[h,w][c] & \text{otherwise} \end{cases}$$
(4)

where $(\mathbf{X}, \mathbf{Y}) \in \mathcal{T}_k$, while $[h, w]$ denotes the pixel coordinates. Labels at step $k = 0$ are not inpainted, as at that stage we lack any prior knowledge of past classes. When background inpainting is performed, each set $\mathcal{T}_k^{bi} \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_{0\to k}}$ ($k > 0$) contains all samples of $\mathcal{T}_k$ after being inpainted.

Incremental Training with Replay Block. The training procedure of RECALL is detailed and summarized in Algorithm 1 and the process is described in Fig. 2. Suppose we are at the incremental step k, with only training data of classes in $\mathcal{C}_k$ from partition $\mathcal{T}_k$ available. In a first stage, the Replay Block is fixed and it is used to retrieve annotated data for steps from 0 to $k - 1$ uniformly distributed among all the past classes. Following the described pipeline, the generative and labeling models are applied independently over each incremental class set $\mathcal{C}_i$, $i = 0, ..., k - 1$. The replay

---

Algorithm 1 RECALL: incremental training procedure.

Input: $\{\mathcal{T}_k\}_{k=0}^K$ and $\{\mathcal{C}_k\}_{k=0}^K$
Output: M_K
train $M_0 = E_0 \circ D_0$ with $\mathcal{L}_{ce}(M_0; \mathcal{C}_0, \mathcal{T}_0)$
train S on $(\mathcal{C}_0, \mathcal{T}_0)$
train $D_{\mathcal{C}_0}^H$ with $\mathcal{L}_{ce}(L_{\mathcal{C}_0}; \mathcal{C}_0, \mathcal{T}_0)$
for $k \leftarrow 1$ to K **do**
    background inpainting on $\mathcal{T}_k$ to obtain $\mathcal{T}_k^{bi}$
    train S on $(\mathcal{C}_k, \mathcal{T}_k)$
    train $D_{\mathcal{C}_k}^H$ with $\mathcal{L}_{ce}(L_{\mathcal{C}_k}; \mathcal{C}_k \cup \{b\}, \mathcal{T}_k)$
    generate $\mathcal{T}_k^{rp} = \mathcal{T}_k^{bi} \cup \mathcal{R}_{\mathcal{C}_{0\to(k-1)}}$
    train D_k with $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\to k}, \mathcal{T}_k^{rp})$
end for

---

training dataset for step k is the union of the single replay sets for each previous step: $\mathcal{R}_{\mathcal{C}_{0\to(k-1)}} = \bigcup_{i=0}^{k-1} \mathcal{R}_{\mathcal{C}_i}$. Once we have assembled $\mathcal{R}_{\mathcal{C}_{0\to(k-1)}}$, by merging it with $\mathcal{T}_k^{bi}$ we get an augmented step-k training partition $\mathcal{T}_k^{rp} = \mathcal{T}_k^{bi} \cup \mathcal{R}_{\mathcal{C}_{0\to(k-1)}}$. This new set, in principle, is complete of annotated samples containing both old and new classes, thanks to replay data. Therefore, we effectively learn the segmentation model M_k through the cross-entropy objective $\mathcal{L}_{ce}(M_k; \mathcal{C}_{0\to k}, \mathcal{T}_k^{rp})$ on replay-augmented training data. This mitigates the bias toward new classes, thus preventing forgetting.

In a second stage, we exploit $\mathcal{T}_k$ to train the Class Mapping Module if needed (see Sec. 5). In particular, we teach the Source Block S to produce samples of $\mathcal{C}_k$, and we optimize the decoder $D_{\mathcal{C}_k}^H$ to correctly segment, in conjunction with E_0, images from $\mathcal{T}_k$ by minimizing $\mathcal{L}_{ce}(L_{\mathcal{C}_k}; \mathcal{C}_k \cup \{b\}, \mathcal{T}_k)$. This stage is not exploited in the current step, but will be necessary in future ones.

During a standard incremental training stage, we follow

a mini-batch gradient descent scheme, where batches of annotated training data are sampled from $\mathcal{T}_k^{rp}$. However, to guarantee a proper stream of information, we opt for an interleaving sampling policy, rather than a random one. In particular, at a generic iteration of training, a batch of data $\mathcal{B}^{rp}$ supplied to the network is made of r_{new} samples from the current training partition $\mathcal{T}_k^{bi}$ and r_{old} replay samples from $\mathcal{R}_{\mathcal{C}_{0\to(k-1)}}$. The ratio between r_{new} and r_{old} controls the proportion of replay and new data (see also Sec. 7.1). We need, in fact, to carefully balance how new data is dosed with respect to replay one, so that enough information about new classes is provided within the learning process, while concurrently we assist the network in recalling knowledge acquired in past steps to prevent catastrophic forgetting.

5. Replay Strategies

In this section we describe more in detail the replay strategies employed for the image generation task of the Source Block S. As mentioned previously, we opt for a generative approach based on a GAN framework and for an online retrieval solution, where images are collected by a web crawler.
Replay by GAN. The GAN-based strategy exploits a deep generative adversarial framework to re-create the no longer available samples for previously seen classes. We use a conditional GAN, G, pre-trained on a generic large-scale visual dataset with data from a wide set of semantic classes $\mathcal{C}^G$ and different domains. For the experiments, we choose an ImageNet [9] based pre-training. On this regard, we remark that classes and domains are not required to be completely coherent: for instance *person* does not exist in ImageNet, but related classes (*e.g.*, *hat*) still allow to preserve its knowledge (further considerations on this are reported in Suppl. Mat.). When performing the k-th incremental step, we retrieve images containing previously seen classes by sampling the GAN's generator output, *i.e.*, $\mathbf{X}^{rp} = G(\mathbf{n}, c^G)$ conditioned on GAN's classes $c^G \in \mathcal{C}^G$ corresponding to the target ones from the original training data ($\mathbf{n}$ is a generic noise input).

Since the GAN is pre-trained on a separate dataset, typically it inherits a different label set. For this reason, the Source Block with GAN is composed of two main modules, namely the actual GAN for image generation and a Class Mapping Module to translate each class of the semantic segmentation incremental dataset to the most similar class of the GAN's training dataset. Provided that we have trained both the GAN and class mapping modules, first we use the latter to translate the class set $\mathcal{C}_k$ to the matching set $\mathcal{C}_k^G$. Then, a set of queries to the conditioned GAN's generator:
$$\mathbf{X}_{\mathcal{C}_k}^{rp} = G(\mathbf{n}, c^G), \quad c^G \in \mathcal{C}_k^G \qquad (5)$$
provides samples resembling the ones in $\mathcal{C}_k$, as long as the mapping is able to properly associate each original class to a statistically similar counterpart in the GAN's label space.

At each incremental step k, the Source Block with GAN goes through two separate training and inference stages. In

a first training phase, samples from $\mathcal{T}_k$ are fed to an Image Classifier I, which is pre-trained to solve an image classification task on the GAN's dataset. In particular, for each class $c \in \mathcal{C}_k$ we select the corresponding training subset $\mathcal{T}_k^c \subset \mathcal{T}_k$, *i.e.*, all the samples of set $\mathcal{T}_k$ associated to class c, and we sum the resulting class probability vectors from the classification output. Then, the GAN's class c^G with the highest probability score is identified by:
$$c^G = \arg\max_{j \in \mathcal{C}^G} \sum_{\mathbf{X} \leftarrow \mathcal{T}_k^c} I(\mathbf{X})[j] \qquad (6)$$
where $\mathbf{X}$ is extracted from $\mathcal{T}_k^c$ (labels are not used) and $I(\mathbf{X})$ denotes the vector output of the last softmax layer of I, whose j-th entry corresponds to the j-th GAN's class. By repeating this procedure for every class in $\mathcal{C}_k$, we build the mapped set $\mathcal{C}_k^G$. Class correspondence is stored, so that at each step we have access to class mappings of past iterations.

In a second evaluation phase, classes in $\mathcal{C}_{0\to(k-1)}$ are given as input to the Source Block. Thanks to the class correspondences saved in previous steps, $\mathcal{C}_{0\to(k-1)}$ are mapped to $\mathcal{C}_{0\to(k-1)}^G$. Next, image generation conditioned on each class of $\mathcal{C}_{0\to(k-1)}^{GAN}$ is performed, and the resulting replay images are fed to the Label Evaluation Block to be associated to their corresponding semantic labels. By following this procedure, we end up with self-annotated data of past classes suitable to support the supervised training at the current step, which otherwise would be limited to new classes.
Replay by Web Crawler. As an alternative we propose to retrieve training examples from an online source. For the evaluation, we searched images from the $Flickr$ website, but any other online database or search engine can be used.

Assuming we are at the incremental step k and we have access to the names of every class in the past iterations (*e.g.*, $\forall c \in \mathcal{C}_{0\to(k-1)}$), we download images whose tag and description happen to both contain the class name through the $Flickr$'s web crawler. Then, the web-crawled images are fed to the Label Evaluation Block for their annotation.

Compared to the GAN-based approach, the online retrieval solution is simpler as no learnable modules are introduced. In addition, we completely avoid to assume that a larger dataset is available, whose class range should be sufficiently ample and diverse to cope with the continuous stream of novel classes incrementally introduced. On the other side, this approach requires the availability of an internet connection and in some way exploits additional training data even if almost unsupervised. Plus, we lack control over the weak labeling performed by the web source.

6. Implementation Details

We use the DeepLab-V2 [7] as segmentation architecture with ResNet-101 [18] as backbone. Nonetheless, RE-CALL is independent of the specific network architecture. Encoder's weights are pre-trained on ImageNet [9] and all

network's weights are trained in the initial step 0. In the following steps, only the main decoder is trained, together with the additional $\{D_{\mathcal{C}_k}^H\}_k$ helper decoders, which are needed to annotate replay samples (as discussed in Sec. 4). For fair comparison, all competing approaches are trained with the same backbone. SGD with momentum is used for weights optimization, with initial learning rate set to 5×10^{-4} and decreased to 5×10^{-6} according to polynomial decay of power 0.9. Following previous works [29, 31], we train the model for $|\mathcal{C}_k| \times 1000$ learning steps in the disjoint setup and for $|\mathcal{C}_k| \times 1500$ steps in the overlapped setup. Each helper decoder $D_{C_k}^H$ is trained with a polynomially decaying learning rate starting from 2×10^{-4} and ending at 2×10^{-6} for $|\mathcal{C}_k| \times 1000$ steps. As Source Block, we use BigGAN-deep [4] pre-trained [1] on ImageNet. At each incremental step k, we generate 500 replay samples per old class, *i.e.* $N_r = 500$. To map classes from the segmentation dataset to the GAN's one, we use the EfficientNet-B2 [41] classifier implemented at [2] and pre-trained on ImageNet. The interleaving ratio r_{old}/r_{new} is set to 1.

As input pre-processing, random scaling and mirroring are followed by random padding and cropping to 321×321 px. The entire framework is developed in TensorFlow [3] and trained on a single NVIDIA RTX 2070 Super. Training time varies depending on the setup, with the longest run taking about 5 hours. Code and replay data are available at <https://github.com/LTTM/RECALL>.

7. Experimental Results

In this section we present the experimental evaluation on the Pascal VOC 2012 dataset [12]. Following previous works on this topic [40, 29, 31, 5], we start by analyzing the performance on three widely used incremental scenarios: *i.e.*, addition of the last class (19-1), addition of the last 5 classes at once (15-5) and addition of the last 5 classes sequentially (15-1). Moreover, we report the performance on three more challenging scenarios in which 10 classes are added sequentially one by one (10-1), in 2 batches of 5 elements each (10-5) and all at once (10-10). Classes for the incremental steps are selected according to the alphabetical order. We compare with the naïve fine-tuning approach (FT), which defines the lower limit to the accuracy of an incremental model, and with the joint training on the complete dataset in one step, which serves as upper bound. We also report the results of a simple Store and Replay (S&R) method, where at each incremental step we store a certain number of true samples for newly added classes, such that the respective size in average matches the size of the helper decoders needed by RECALL (see Fig. 6). As comparison, we include 2 methods extended from classification (*i.e.*, LwF [26] and its single-headed version LwF-MC [35]) and the most relevant methods designed for continual segmentation (*i.e.*, ILT [29], CIL [23], MiB [5] and SDR [30]). Exhaustive

quantitative results in terms of mIoU are shown in Table 1. For each setup we report the mean accuracy for the initial set of classes, for the classes in the incremental steps and for all classes, computed after the overall training.

Addition of the last class. First, we train over the first 19 classes during step 0. Then, we perform a single incremental step to learn *tv/monitor*. Looking at Table 1 (upper-left section), we notice that FT results in a drastic performance degradation w.r.t. joint training, due to catastrophic forgetting. RECALL, instead, shows higher overall mIoU than competitors and it is especially effective on the last class, whilst still retaining high accuracy on the past ones thanks to the regularization brought in by background inpainting and replay strategies. S&R, instead, heavily forgets previous classes, thus confirming the usefulness of replay data.

Addition of last 5 classes. In this setup, 15 classes are learned in the initial step, while the remaining 5 are added in one shot (15-5) or sequentially one at a time (15-1). Compared to the 19-1 setup, the addition of multiple classes in the incremental iterations makes catastrophic forgetting even more severe. The accuracy gap between FT and joint training, in fact, raises from about 41% of the 19-1 case to more than 70% of mIoU in the 15-1 scenario. Taking a closer look at the results in Table 1 (upper mid and right sections), our replay approaches strongly limit the degradation caused by catastrophic forgetting. This trend can be observed in the 15-5 setup and more evidently in the 15-1 one, both in the disjoint and overlapped settings: exploiting generated or web-derived replay samples proves to effectively restore knowledge of past classes, leading to a final mIoU approaching that of the joint training. Storing and replaying original samples, instead, improves the performance w.r.t. FT, but ultimately leads to a mIoU lower of more than 20% if compared to our approaches. This is due to the limited number of samples to be stored in order to match the helper decoder size: their sole addition is, in fact, insufficient to adequately preserve learned knowledge. Finally, we observe that RE-CALL can scale much better than competitors when multiple incremental steps are performed (scenario 15-1), as typically encountered in real-world applications.

Addition of last 10 classes. To analyze the previous claim, we introduce some new challenging experiments, not evaluated in previous works. In these tests only 10 classes are observed in the initial step, while the remaining ones are added in a single batch (10-10), in 2 steps of 5 classes each (10-5), or individually (10-1). Again, FT is heavily affected by the information loss that occurs when performing incremental training without regularization, leading to performance drops up to about 71% of mIoU w.r.t. the joint training in the most challenging 10-1 setting. Thanks to the introduction of replay data, RECALL brings a remarkable performance boost to the segmentation accuracy and becomes more and more valuable as the difficulty of the

Table 1: mIoU on Pascal VOC2012 for different incremental setups. Results of competitors in the upper part come from [30, 5], while we run their implementations for the new scenarios in the bottom part.

	19-1 Disjoint			19-1 Overlapped			15-5 Disjoint			15-5 Overlapped			15-1 Disjoint			15-1 Overlapped		
Method	1-19	20	all	1-19	20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all
FT	35.2	13.2	34.2	34.7	14.9	33.8	8.4	33.5	14.4	12.5	36.9	18.3	5.8	4.9	5.6	4.9	3.2	4.5
S&R	55.3	43.2	56.2	54.0	48.0	55.1	38.5	43.1	41.6	36.3	44.2	40.3	41.0	31.8	40.7	38.6	31.2	38.9
LwF [26]	65.8	28.3	64.0	62.6	23.4	60.8	39.7	33.3	38.2	67.0	41.8	61.0	26.2	15.1	23.6	24.0	15.0	21.9
LwF-MC [35]	38.5	1.0	36.7	37.1	2.3	35.4	41.5	25.4	37.6	59.8	22.6	51.0	6.9	2.1	5.7	6.9	2.3	5.8
ILT [29]	66.9	23.4	64.8	50.2	29.2	49.2	31.5	25.1	30.0	69.0	46.4	63.6	6.7	1.2	5.4	5.7	1.0	4.6
CIL [23]	62.6	18.1	60.5	35.1	13.8	34.0	42.6	35.0	40.8	14.9	37.3	20.2	33.3	15.9	29.1	6.3	4.5	5.9
MiB [5]	69.6	25.6	67.4	70.2	22.1	67.8	71.8	43.3	64.7	75.5	49.4	69.0	46.2	12.9	37.9	35.1	13.5	29.7
SDR [30]	69.9	37.3	68.4	69.1	32.6	67.4	73.5	47.3	67.2	75.4	52.6	69.9	59.2	12.9	48.1	44.7	21.8	39.2
RECALL (GAN)	65.2	50.1	65.8	67.9	53.5	68.4	66.3	49.8	63.5	66.6	50.9	64.0	66.0	44.9	62.1	65.7	47.8	62.7
RECALL (Web)	65.0	47.1	65.4	68.1	55.3	68.6	69.2	52.9	66.3	67.7	54.3	65.6	67.6	49.2	64.3	67.8	50.9	64.8
Joint	75.5	73.5	75.4	75.5	73.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4

	10-10 Disjoint			10-10 Overlapped			10-5 Disjoint			10-5 Overlapped			10-1 Disjoint			10-1 Overlapped		
Method	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all
FT	7.7	60.8	33.0	7.8	58.9	32.1	7.2	41.9	23.7	7.4	37.5	21.7	6.3	2.0	4.3	6.3	2.8	4.7
S&R	25.1	53.9	41.7	18.4	53.3	38.2	26.0	28.5	29.7	22.2	28.5	27.9	30.2	19.3	27.3	28.3	20.8	27.1
LwF [26]	63.1	61.1	62.2	70.7	63.4	67.2	52.7	47.9	50.4	55.5	47.6	51.7	6.7	6.5	6.6	16.6	14.9	15.8
LwF-MC [35]	52.4	42.5	47.7	53.9	43.0	48.7	44.6	43.0	43.8	44.3	42.0	43.2	6.9	1.7	4.4	11.2	2.5	7.1
ILT [29]	67.7	61.3	64.7	70.3	61.9	66.3	53.4	48.1	50.9	55.0	44.8	51.7	14.1	0.6	7.5	16.5	1.0	9.1
CIL [23]	37.4	60.6	48.4	38.4	60.0	48.7	27.5	41.4	34.1	28.8	41.7	34.9	7.1	2.4	4.9	6.3	0.8	3.6
MiB [5]	66.9	57.5	62.4	70.4	63.7	67.2	54.3	47.6	51.1	55.2	49.9	52.7	14.9	9.5	12.3	15.1	14.8	15.0
SDR [30]	67.5	57.9	62.9	70.5	63.9	67.4	55.5	48.2	52.0	56.9	51.3	54.2	25.5	15.7	20.8	26.3	19.7	23.2
RECALL (GAN)	62.6	56.1	60.8	65.0	58.4	63.1	60.0	52.5	57.8	60.8	52.9	58.4	58.3	46.0	53.9	59.5	46.7	54.8
RECALL (Web)	64.1	56.9	61.9	66.0	58.8	63.7	63.2	55.1	60.6	64.8	57.0	62.3	62.3	50.0	57.8	65.0	53.7	60.7
Joint	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4

settings increases. In the 10-10 case, our method achieves slightly lower mIoU results than competitors (although comparable). As we increase complexity, our approach is able to outperform competitors by about 8% of mIoU in 10-5 and by 37% of mIoU in 10-1. We remark how RECALL shows a convincing capability of providing a rather steady accuracy in different setups, regardless of the number of incremental steps used to introduce new classes. For example, in the disjoint scenario, when moving from simpler to more challenging setups (*i.e.*, from 10-10 to 10-1, passing through 10-5), the mIoU of FT drops as $33.0\% \rightarrow 23.7\% \rightarrow 4.3\%$ and the one of SDR (*i.e.*, the best compared approach) as $62.9\% \rightarrow 52.0\% \rightarrow 20.8\%$, while our approach maintains a stable mIoU trend of $61.9\% \rightarrow 60.6\% \rightarrow 57.8\%$. Finally, we report the mIoU after each incremental step on the 10-1 disjoint scenario in Fig. 4, where our approaches show much higher mIoU at every learning step, indicating improved resilience to forgetting and background shift, than competitors.

In the qualitative results in Fig. 5 we observe that RE-CALL effectively alleviates forgetting and reduces the bias towards novel classes. In the first row, the *bus* is correctly preserved while FT, S&R and inpainting wrongly classify it as *train* (*i.e.*, one of the novel classes); in the second row, FT places *sheep* and *tv* (newly added classes) in place of *cow*; in the third row, some *horse*'s features are either mixed with those of *person* and *cat* or completely destroyed, while they are preserved by our methods (the web scheme shows higher accuracy than the GAN here). Additional results, a few tests
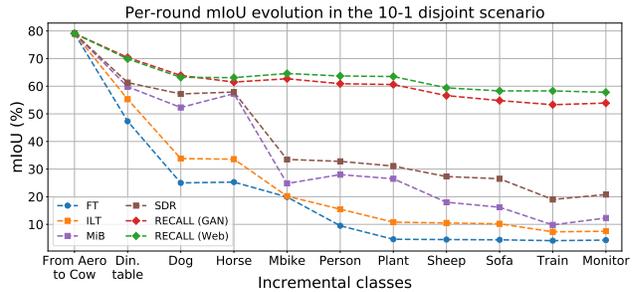
Figure 4: Evolution of mIoU on the 10 tasks of 10-1 disjoint.

combining RECALL with competitors and a preliminary evaluation on ADE20K [47] are presented in Suppl. Mat.

7.1. Ablation Study

To further validate the robustness of our approach, we perform some ablation studies. First of all, we analyze the memory requirements. The plot in Fig. 6 shows in semi-log scale the memory occupation (expressed in MB) of the data to be stored at the end of each incremental step, as a function of the number of classes learned up to that point. We denote with *standard* an incremental approach which do not store any sample (*e.g.*, FT, LwF, ILT, MiB, SDR). The saved model generally corresponds to a fixed size encoder and a decoder, whose dimension slightly increases at each step to account for additional output channels for the new learnable classes. *Saving images*, instead, refers to the extreme scenario where training images of past steps are stored, thus being available

Figure 5: Qualitative results on disjoint incremental setups: from top to bottom 15-1, 15-5 and 10-1. (*best viewed in colors*).

Figure 6: Memory occupation in the disjoint scenario.

Figure 7: Distinct interleaving policies in 15-1 disjoint.

Table 2: mIoU results showing the contribution of each module, D: Disjoint, O: Overlapped.

Method	19-1		15-5		15-1		10-10		10-5		10-1	
	D	O	D	O	D	O	D	O	D	O	D	O
Bgr inp.	65.6	66.7	52.2	52.5	49.7	49.9	58.8	60.7	47.5	47.1	34.0	39.0
GAN	54.5	56.2	49.8	49.1	47.9	48.2	45.8	48.8	38.1	43.7	36.6	40.8
Web	57.3	57.4	55.2	54.7	55.0	53.7	55.2	58.2	47.9	52.1	45.4	50.1
GAN+inp.	65.8	68.4	63.5	64.0	62.1	62.7	60.8	63.1	57.8	58.4	53.9	54.8
Web+inp.	65.4	68.6	66.3	65.6	64.3	64.8	61.9	63.7	60.6	62.3	57.8	60.7

observe that its effect tends to attenuate with multiple incremental steps. For example, moving from 10-10 to 10-1 overlapped setups, the mIoU drops more than 20%. On the other hand, the proposed replay techniques prove to be beneficial when multiple training stages are involved. On the same setting, replay techniques alone limit the degradation to only 8%. Yet, jointly employing replay and inpainting further boosts the final results in all setups (up to 15%), proving that they can be effectively combined.

Finally, we analyze how results vary w.r.t. the proportion of new (r_{new}) and replay (r_{old}) samples seen during training (Fig. 7): the mIoU is quite stable w.r.t. this ratio, however the maximum value is reached when the same number of old and replay samples is used, *i.e.*, $r_{new}/r_{old} = 1$.

8. Conclusions

In this paper we introduced RECALL, which targets continual semantic segmentation by means of replay strategies to alleviate catastrophic forgetting and background inpainting to mitigate background shift. Two replay schemes are proposed to retrieve data related to former training stages, either reproducing it via a conditional GAN or crawling it from the web. The experimental analyses proved the efficacy of our framework in improving accuracy and robustness to multiple incremental steps compared to competitors. New research will improve the generative model, coupling it more strictly with the incremental setup, and explore how to control and refine weak supervision during web-crawling. Evaluation on different datasets, such as ADE20K, will also be performed.

throughout the entire incremental process. As concerns our approach, to annotate originally weakly-labeled replay images, we devise a specific module (Sec. 4), which requires to save a set of helper decoders $\{D_{\mathcal{C}_i}^H\}_{i=0}^k$, one for each past step. Finally, for the GAN-based approach we add the storage required for the generative model. Fig. 6 shows that our web-based solution is very close to the *standard* ones in terms of memory occupation. The space required to store the GAN is comparable to that needed to save images in the very initial steps, but then remains constant while the space for saving all training data quickly grows.

We further analyze the contribution of the background inpainting and replay techniques in Table 2. While inpainting alone provides a solid contribution in terms of knowledge preservation acting similarly to knowledge distillation, we

References

[1] Tensorflow module of BigGAN-deep 512, https://tfhub.dev/deepmind/biggan-deep-512/1. Accessed on 18/03/2020.

[2] TensorFlow module of EfficientNet-b2, https://tfhub.dev/google/efficientnet/b2/classification/1. Accessed on 18/03/2020.

[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of the International Conference on Learning Representations*, 2019.

[5] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

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

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:834–848, 2018.

[8] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the International Conference on Computer Vision*, pages 1431–1439, 2015.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[10] Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3277, 2014.

[11] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[13] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[14] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[16] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[17] Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *Proceedings of the British Machine Vision Conference*, page 98, 2018.

[18] 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*, pages 770–778, 2016.

[19] Seunghoon Hong, Donghun Yeo, Suha Kwak, Honglak Lee, and Bohyung Han. Weakly supervised semantic segmentation using web-crawled videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7322–7330, 2017.

[20] Bin Jin, Maria V Ortiz Segovia, and Sabine Susstrunk. Webly supervised semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3635, 2017.

[21] Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017.

[22] 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.

[23] Marvin Klingner, Andreas Bär, Philipp Donn, and Tim Fingscheidt. Class-incremental learning for semantic segmentation re-using neither old data nor old labels. *International Conference on Intelligent Transportation Systems*, 2020.

[24] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.

[25] Dawei Li, Serafettin Tasci, Shalini Ghosh, Jingwen Zhu, Junting Zhang, and Larry Heck. Efficient incremental learning for mobile object detection. *arXiv preprint arXiv:1904.00781*, 2019.

[26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[28] David Lopez-Paz and MarcAurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

[29] Umberto Michieli and Pietro Zanuttigh. Incremental Learning Techniques for Semantic Segmentation. In *Proceedings of the International Conference on Computer Vision Workshops*, 2019.

[30] Umberto Michieli and Pietro Zanuttigh. Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[31] Umberto Michieli and Pietro Zanuttigh. Knowledge distillation for incremental learning in semantic segmentation. *Computer Vision and Image Understanding*, 2021.

[32] Li Niu, Ashok Veeraraghavan, and Ashutosh Sabharwal. Webly supervised learning meets zero-shot learning: A hybrid approach for fine-grained classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7171–7180, 2018.

[33] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11321–11329, 2019.

[34] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[36] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[37] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the International Conference on Machine Learning*, 2018.

[38] Tong Shen, Guosheng Lin, Chunhua Shen, and Ian Reid. Bootstrapping the performance of webly supervised semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1363–1371, 2018.

[39] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[40] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the International Conference on Computer Vision*, pages 3400–3409, 2017.

[41] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 6105–6114, 2019.

[42] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2471–2480, 2017.

[43] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018.

[44] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *Proceedings of the International Conference on Learning Representations*, 2016.

[45] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning*, pages 3987–3995, 2017.

[46] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.

[47] 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*, 2017.