Incremental Learning Techniques for Semantic Segmentation

Umberto Michieli and Pietro Zanuttigh
University of Padova, Italy
{umberto.michieli,zanuttigh}@dei.unipd.it

Abstract

Deep learning architectures exhibit a critical drop of performance due to catastrophic forgetting when they are required to incrementally learn new tasks. Contemporary incremental learning frameworks focus on image classification and object detection while in this work we formally introduce the incremental learning problem for semantic segmentation in which a pixel-wise labeling is considered. To tackle this task we propose to distill the knowledge of the previous model to retain the information about previously learned classes, whilst updating the current model to learn the new ones. We propose various approaches working both on the output logits and on intermediate features. In opposition to some recent frameworks, we do not store any image from previously learned classes and only the last model is needed to preserve high accuracy on these classes. The experimental evaluation on the Pascal VOC2012 dataset shows the effectiveness of the proposed approaches.

1. Introduction and Related Work

Deep neural networks are a key tool for computer vision systems. Despite their wide success on many visual recognition problems, neural networks struggle in learning new tasks whilst preserving good performance on previous ones since they suffer from catastrophic forgetting [11, 13, 21]. More precisely, the incremental learning problem is defined as the capability of machine learning architectures to continuously improve the learned model by feeding new data without losing previously learned knowledge. This has been widely studied in the context of problems like image classification and object detection [5, 18, 26, 30, 35]. Traditional learning models require that all the samples corresponding to old and new tasks are available during all steps of the training stage; a real world system, instead, should be able to update its knowledge with few training steps incorporating the new tasks while preserving unaltered the previous ones. Such a behavior is inherently present in human brain which is incremental in the sense that new tasks are continuously incorporated but the existing knowledge is preserved.

Catastrophic forgetting represents one of the main limitations of neural networks. It has been addressed even before the rise of neural networks popularity [6, 25, 33], but more recently it has been rediscovered and tackled in different ways. Some methods [16, 27, 28, 36] exploit network architectures which grow during the training process. A different strategy consists in freezing or slowing down the learning process on some relevant parts of the network [16, 17, 18, 24]. Another way of retaining high performance on old tasks is knowledge distillation. This idea was originally proposed in [4, 14] and then adapted in different ways in recent studies [5, 12, 18, 26, 30, 35, 37] to maintain stable the responses of the network on the old tasks whilst updating it with new training samples. However, differently from this paper, previous works focus only on object detection or image classification problems.

Some studies keep a small portion of data belonging to previous tasks and use them to preserve the accuracy on old tasks when dealing with new problems [5, 7, 15, 20, 26, 32]. The exemplar set to store is chosen at random or according to a relevance metric. In [5] the classifier and the features for selecting the samples to be added in the representative memory are learned jointly and herding selection is then used. Another method of this family is the only work considering an incremental setting for semantic segmentation [32], which however focuses on a very specific setup related to satellite images and has several limitations when applied to generic semantic segmentation problems. Indeed, it considers the segmentation as a multi-task learning problem, where a binary classification for each class replaces the multi-class labeling, and it stores some patches of previously seen images. Furthermore it assumes that training images corresponding to an incremental step only contain new classes while the capabilities on old ones are preserved by storing a subset of the old images. For large amount of classes and wide range of applications the methodology does not scale properly.

Storing previously seen data could represent a serious limitation for certain applications where privacy issues or limited storage budgets are present. For this reason, some recent methods [29, 35] do not store old data but com-

pensate this by training Generative Adversarial Networks (GANs) to generate images containing previous classes while new classes are learned. Some other approaches do not make use of exemplars set [2, 17, 18, 30, 31, 37]. In [30] an end-to-end learning framework is proposed where the representation and the classifier are learned jointly without storing any of the original training samples. In [18] previous knowledge is distilled directly from the last trained model. In [37] the current model distills the knowledge from pruned versions of all previous model snapshots.

Even if previous studies focus on different tasks and no work has been conducted on incremental learning for dense labeling task, semantic segmentation is a key task that computer vision systems must face frequently in various applications e.g., in robotics or autonomous driving [3, 22]. Notice that, differently from image classification, in semantic segmentation each image contains together pixels belonging to multiple classes and the labeling is dense. In particular the pixels could represent newly added classes and previously existing ones, making the problem conceptually different from incremental learning in image classification where typically a single object is present in the image and the outcome is a unique value. Furthermore, contrary to many existing methods, we consider the most challenging setting where images from old tasks are not stored and cannot be used to help the incremental process, which is particularly relevant for the vast majority of applications with privacy concerns or storage requirements.

In the first part of this paper we formalize the problem and we present possible settings for the incremental learning task. Then we introduce a novel framework to perform incremental learning for semantic segmentation. In particular we re-frame the distillation loss concept used in other fields and we propose a novel approach where the distillation loss is applied to the intermediate features level. Furthermore, we exploited the idea of freezing the encoder part of the network to preserve the feature extraction capabilities. To the best of our knowledge this is the first work on incremental learning for semantic segmentation which does not retain previously seen images and that has been evaluated on standard datasets, i.e., Pascal VOC2012 [10]. Experimental results demonstrate that the proposed approaches obtain high accuracy even without storing any of the previous examples thanks to the proposed distillation schemes.

2. Problem Formulation

The incremental learning task, when referring to semantic segmentation, can be defined as the ability of a learning system (e.g., a neural network) to learn the segmentation and the labeling of the new classes without forgetting or deteriorating too much the performance on previously learned ones. The performance of an incremental learning algorithm should be evaluated considering the accuracy on the new classes as well as the accuracy on the old ones. While the first should be as large as possible, meaning that the algorithm is able to learn the new classes, the second should be as close as possible to the one before the addition of the new classes, thus avoiding catastrophic forgetting. The key challenge then is how to balance between the preservation of previous segmentation and labeling knowledge and the capability of learning the new classes. Additionally, the considered problem is particularly hard when no data of previous tasks can be preserved, which is the scenario of interest in the majority of the applications. In this work we focus on the most general incremental learning framework in which: previously seen images are not used; the new images contain examples of the unseen classes combined together with pixels belonging to the old ones; the complexity of the approach scales well as the number of classes grows.

Let us assume that the available set of samples is $\mathcal{D}$ and is composed of N images. As usual part of the data is used for training and part for testing: we refer to the training split of $\mathcal{D}$ as $\mathcal{D}^{tr}$. Each pixel in each image of $\mathcal{D}$ is associated to a unique class belonging to the set $\mathcal{C} = \{c_0, c_1, c_2, ..., c_{C-1}\}$ of C possible classes. In case a background class is present we associate it to class c_0 because it is considered a special class with a non-conventional behavior being present in almost all the images and having by far the largest occurrence among the elements of $\mathcal{C}$.

In the incremental learning setting we assume that we have trained our network to recognize a subset $\mathcal{S}_0 \subset \mathcal{C}$ of *seen* classes using a labeled subset $\mathcal{D}_0^{tr} \subset \mathcal{D}^{tr}$, whose images contain only pixels belonging to the classes in $\mathcal{S}_0$. We then perform some incremental steps $k = 1, 2, ...$ in which we want to recognize a new subset $\mathcal{U}_k \subset \mathcal{C}$ of *unseen* classes. Notice that at the k-th incremental step the set of seen classes $\mathcal{S}_{k-1}$ is the union of all the classes previously learned and after the step we add the ones learned during the current step k: more formally, $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \mathcal{U}_k$ and $\mathcal{S}_{k-1} \cap \mathcal{U}_k = \emptyset$. At each step a new set of training samples is available, i.e., $\mathcal{D}_k^{tr} \subset \mathcal{D}^{tr}$, whose images contain only pixels belonging to $\mathcal{S}_{k-1} \cup \mathcal{U}_k$. The set is disjoint from previously used samples, i.e., $\left(\bigcup_{j=0,...,k-1} \mathcal{D}_j^{tr}\right) \cap \mathcal{D}_k^{tr} = \emptyset$. It is important to notice that, differently from image classification, images in $\mathcal{D}_k^{tr}$ could also contain classes belonging to $\mathcal{S}_{k-1}$, however their occurrence is limited since $\mathcal{D}_k^{tr}$ is restricted to consider only images containing at least one class belonging to $\mathcal{U}_k$. Furthermore, the specific occurrence of a particular class belonging to $\mathcal{S}_{k-1}$ is highly correlated to the set of classes being added (i.e., $\mathcal{U}_k$). For example if we assume that $\mathcal{S}_{k-1} = \{chair, airplane\}$ and that $\mathcal{U}_k = \{dining\ table\}$, then it is reasonable to expect that $\mathcal{D}_k^{tr}$ contains some images having the $chair$ class, that typically appears together with the $dining\ table$, while the class $airplane$ is extremely unlikely.

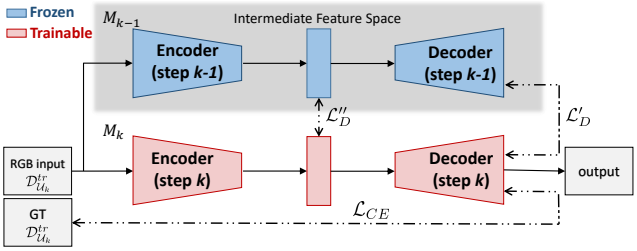

Figure 1. Overview of the k-th incremental step of our learning framework for semantic segmentation of RGB images.

Given this scenario, there exist many different ways of sampling the set $\mathcal{U}_k \subset \mathcal{C}$ of unseen classes and of selecting the cardinality of the sets $\mathcal{U}_k$ at each step, leading to different experiments. Previous work [30] ordered the classes using the sequence provided by the creators of the dataset and analyzed the behavior of the algorithms to the addition of a single class, the addition of a batch of classes and the sequential addition of classes. Our results stick to these settings to reproduce the same scenarios.

3. Methodology

In this work we start by re-framing incremental learning techniques developed for other fields in the semantic segmentation task. Then we propose some novel strategies explicitly targeted to this problem.

The proposed approaches can be fitted into any deep network architecture, however for the evaluation we chose the Deeplab v2 network (without the post-processing based on CRFs) with ResNet-101 as feature extractor [8] pre-trained [23] on the MSCOCO dataset [19]. The pre-training of the feature extractor (as done also in other incremental learning works as [18]) is needed since the Pascal VOC 2012 is too small to be used for training the Deeplab v2 from scratch. However MSCOCO data are used only for the initialization of the feature extractor and the contained labeling information, even if there are overlapping classes, is related to a different task (i.e., image classification).

The various procedures to achieve incremental learning in semantic segmentation are now introduced: see Fig. 1 for a general overview of the approach. We start by training the chosen network architecture in the first stage to recognize the classes in $\mathcal{S}_0$ with the corresponding training data $\mathcal{D}^{tr}_0$. The network is trained in a supervised way with a standard cross-entropy loss and after training we save the obtained model as M_0. Then, we perform a set of incremental steps indexed by $k = 1, 2, ...$ to make the model learn every time a new set of classes $\mathcal{U}_k$. At the k-th incremental step, the current training set $\mathcal{D}^{tr}_k$ is built with images that contain samples from at least one of the new classes. Notice that they can possibly contain also pixels belonging to previously seen classes and of course the background class is present in almost all images. During step k, the model M_{k-1} is loaded and updated exploiting a linear combination of two losses: a cross-entropy loss $\mathcal{L}_{CE}$, which learns how label the classes, and a distillation loss $\mathcal{L}_D$, which helps to retain knowledge of previously seen classes and will be detailed in the following. After the k-th incremental step, we save the current model as M_k and the described procedure is repeated every time a new set of classes to be learned is taken into account. The total loss $\mathcal{L}$ to train the model is:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_D \mathcal{L}_D \tag{1}$$

The parameter λ_D balances the two terms. If we set $\lambda_D = 0$ then we are considering the simplest scenario of fine-tuning in which no knowledge distillation is applied and the cross-entropy loss is applied to both unseen and seen classes (but in $\mathcal{D}^{tr}_k$ there is a large unbalance toward the new ones, see Section 2). As already pointed out, we expect this case to exhibit catastrophic forgetting.

During the k-th incremental step the cross-entropy loss $\mathcal{L}_{CE}$ is applied to all the classes and it is defined as:

$$\mathcal{L}_{CE} = -\frac{1}{|\mathcal{D}^{tr}_k|} \sum_{\mathbf{X}_n \in \mathcal{D}^{tr}_k} \sum_{c \in \mathcal{S}_{k-1} \cup \mathcal{U}_k} \mathbf{Y}_n[c] \cdot \log\left(M_k\left(\mathbf{X}_n\right)[c]\right) \tag{2}$$

where $\mathbf{Y}_n[c]$ and $M_k\left(\mathbf{X}_n\right)[c]$ are respectively the one-hot encoded ground truth and the output of the network corresponding to the estimated score for class c. Notice that the sum is computed on both old and new classes because in practice old classes will continue to appear. However since the new classes are much more likely in $\mathcal{D}^{tr}_k$, there is a clear unbalance toward them leading to catastrophic forgetting [34]. We introduce two possible strategies for defining the distillation loss $\mathcal{L}_D$ which only depend on the previous model M_{k-1} avoiding the need for large storage.

3.1. Distillation on the Output Layer ($\mathcal{L}'_D$)

The first considered distillation term $\mathcal{L}'_D$ for semantic segmentation is the masked cross-entropy loss between the logits produced by the output of the softmax layer in the previous model M_{k-1} and the output of the softmax layer in the current model M_k (assume that we currently are at the k-th incremental step). The cross-entropy is masked to consider already seen classes only since we want to guide the learning process to retain them, i.e.:

$$\mathcal{L}'_D = -\frac{1}{|\mathcal{D}^{tr}_k|} \sum_{\mathbf{X}_n \in \mathcal{D}^{tr}_k} \sum_{c \in \mathcal{S}_{k-1}} M_{k-1}(\mathbf{X}_n)[c] \cdot \log\left(M_k(\mathbf{X}_n)[c]\right) \tag{3}$$

The loss $\mathcal{L}'_D$ is our baseline model and some enhancements of the scheme have been evaluated. A first modification moves from the consideration that the encoder E aims at extracting some intermediate feature representation

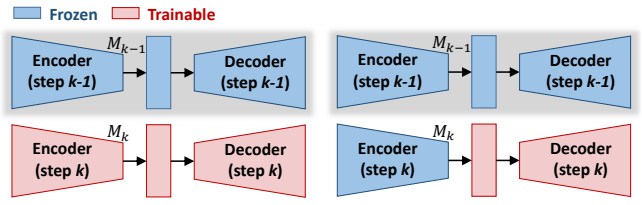

Figure 2. Freezing schemes of the encoder at k-th incremental step. The whole model at previous step, i.e. M_{k-1}, is always completely frozen and is employed only for knowledge distillation.

from the input information: hence the encoder part of the network can be frozen to the status it reached after the previous steps (E_F in short, see Fig. 2). In this way the network is constrained to learn new classes only through the decoder, while preserving the features extraction capabilities unchanged from the previous training stage. We evaluated this approach both with and without the application of the distillation loss in Eq. (3).

3.2. Distillation on Intermediate Feature Space ($\mathcal{L}''_D$)

A different approach we designed to preserve previous knowledge by keeping the encoder similar to the already learned model is to apply a knowledge distillation function to the intermediate level of the features space before the decoding stage. The distillation function on the features space in this case should be no longer the cross-entropy but rather the $L2$ loss. This choice is due to the fact that the considered layer is not anymore a classification layer but instead just an internal stage where the output should be kept close to the previous one in, e.g., $L2$-norm. Empirically, we found that using cross-entropy or $L1$ lead to worse results. Considering that model M_k can be decomposed into an encoder E_k and a decoder, the distillation term would become:

$$\mathcal{L}''_D = \frac{\|E_{k-1}(\mathbf{X}_n) - E_k(\mathbf{X}_n)\|_2^2}{|\mathcal{D}_k^{tr}|} \tag{4}$$

where $E_k(\mathbf{X}_n)$ denotes the features computed by E_k when a generic image $\mathbf{X}_n \in \mathcal{D}_k^{tr}$ is fed as input.

A summary of the proposed strategies is shown in Fig. 1 where the different losses are shown. As a final remark, we also tried a combination of the described distillation losses but it did not provide relevant enhancements.

4. Experimental Results

For the experimental evaluation we selected the Deeplab v2 architecture and we performed the tests on the Pascal VOC2012 [10] benchmark. This widely used dataset consists of 10582 images in the training split and 1449 in the validation split with a total of 21 different classes (background included). Since the test set has not been made available, all the results have been computed on the validation split as done by most approaches in the literature.

We trained our network with Stochastic Gradient Descent (SGD) as done in [8]. The initial stage of training of the network on the set $\mathcal{S}_0$ is performed by setting the starting learning rate to 10^{-4} and training for $|\mathcal{S}_0| \cdot 1000$ steps decreasing the learning rate up to 10^{-6} with a polynomial decay rule with power 0.9. We included weight decay regularization of 10^{-4} and we employed a batch size of 4 images. The incremental training steps $k = 1, 2, ...$ have been performed employing a lower learning rate to better preserve previous weights. In this case the learning rate starts from $5 \cdot 10^{-5}$ and decreases up to 10^{-6} after $|\mathcal{U}_k| \cdot 1000$ steps of polynomial decay. Notice that we train the network for a number of steps which is proportional to the number of new classes to be learned. We used TensorFlow [1] to develop and train the network: the overall training procedure takes around 5 hours on a NVIDIA 2080 Ti GPU. The code is available online at `https://lttm.dei.unipd.it/paper_data/IL`. The metrics we considered are the most widely used for semantic segmentation: the per-class Intersection over Union (IoU), the mean Pixel Accuracy (mPA), the mean Class Accuracy (mCA) and the mean IoU (mIoU) [9].

4.1. Addition of One Class

Following [30] we first analyze the addition of the last class, in alphabetical order, to our network. Specifically, we consider $\mathcal{S}_0 = \{c_0, c_1, ..., c_{19}\}$ and $\mathcal{U}_1 = \{c_{20}\} = \{tv \backslash monitor\}$. A summary of the evaluation of the proposed methodologies on the VOC2012 validation split is reported in Table 1. We indicate as $M_0(0-19)$ the first standard training of the network using $\mathcal{D}_0^{tr}$ as training dataset. The network is then updated exploiting the dataset $\mathcal{D}_1^{tr}$ and the resulting model is referred to as $M_1(20)$. From the first row of Table 1 we can appreciate that fine-tuning the network leads to an evident degradation of the performance with a final mIoU of 65.1%. This is a clear confirmation of the catastrophic forgetting phenomenon in the semantic segmentation scenario even with the addition of just one single class. The reference model, indeed, where all the 21 classes are learned at once (we call it $M_0(0-20)$) achieves a mIoU of 73.6%. The main issue of the fine-tuning approach is that it predicts too frequently the last class, as proved by the fact that the model has a very high pixel accuracy for the $tv/monitor$ class but a very poor IoU of 20.1%. This is due to the high number of false positive detection of the considered class which are not taken into account by the pixel accuracy measure. On the same class, the proposed methods are all able to outperform the fine-tuning approach in terms of IoU by large margin. Knowledge distillation strategies and the procedure of freezing the encoder provide better results because they act as regularization constraints.

$M_1(20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	**mIoU old**	tv	**mIoU**	**mPA**	**mCA**
Fine-tuning	90.2	80.8	33.3	83.1	53.7	68.2	84.6	78.0	83.2	32.1	73.4	52.6	76.6	72.7	68.8	79.8	43.8	76.5	46.5	68.4	67.3	20.1	65.1	90.7	76.5
$\mathcal{L}'_D$	92.0	83.9	37.0	84.0	58.8	70.9	90.9	82.5	86.1	32.1	72.5	51.0	79.9	72.3	77.3	80.9	45.1	78.1	45.7	79.9	70.0	35.3	68.4	92.5	79.5
E_F	92.7	86.2	32.6	82.9	61.7	74.6	92.9	83.1	87.7	27.4	79.4	59.0	79.4	76.9	77.2	81.2	49.6	80.8	49.3	83.4	71.9	43.3	70.5	93.2	81.4
$E_F, \mathcal{L}'_D$	92.9	86.1	37.1	83.6	62.2	76.1	93.2	82.9	88.3	30.6	79.6	58.5	80.3	77.6	77.2	81.8	49.8	81.0	47.0	84.5	72.5	**51.4**	71.5	93.4	82.5
$\mathcal{L}''_D$	92.9	84.8	36.4	82.6	63.5	75.0	92.2	83.6	88.3	29.5	80.3	59.6	79.7	80.2	78.9	81.2	49.7	78.9	51.0	84.1	72.6	50.6	71.6	93.4	**83.4**
$E_F, \mathcal{L}''_D$	92.9	84.2	36.8	82.4	63.3	75.6	92.2	83.6	88.3	28.0	80.1	59.6	79.4	79.9	78.7	81.3	49.7	78.8	51.4	83.8	72.5	50.7	71.5	93.4	**83.4**
$\mathcal{L}'_D, \mathcal{L}''_D$	92.8	85.5	36.5	82.2	62.8	73.4	92.3	82.9	88.5	31.0	79.8	58.4	80.3	79.5	79.1	82.0	50.8	78.5	47.8	84.1	72.4	49.4	71.3	93.4	83.2
$E_F, \mathcal{L}'_D, \mathcal{L}''_D$	92.9	86.0	36.5	84.4	61.8	76.2	93.1	83.1	88.6	30.4	79.7	58.7	80.4	78.1	76.4	82.0	50.5	81.0	50.4	85.1	**72.8**	49.9	**71.7**	**93.5**	**83.4**
$M_0(0-19)$	93.4	85.5	37.1	86.2	62.2	77.9	93.4	83.5	89.3	32.6	80.7	57.3	81.5	81.2	77.7	83.0	51.5	81.6	48.2	85.0	73.4	-	73.4	93.9	84.3
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	52.3	82.4	51.1	86.4	73.7	70.5	73.6	93.9	84.2

Table 1. Per-class IoU on the Pascal VOC2012 under some settings when the last class, i.e. the tv/monitor class, is added.

Interestingly those procedures allow to achieve higher accuracy not only on previously learned classes but also on newly added ones, which might be unexpected if we do not consider the regularization behavior of those terms. We can appreciate that the distillation on the output $\mathcal{L}'_D$ alone is able to improve the average mIoU by 3.3% with respect to the standard case. Furthermore it leads to a much better IoU on the new class, greatly reducing the aforementioned false positives issue. If we completely freeze the encoder E without applying knowledge distillation the model improves the mIoU by 5.4%. If we combine the two mentioned approaches, i.e. we freeze E and we apply $\mathcal{L}'_D$ as distillation loss, the mIoU further improves to 71.5% with an improvement of 6.4%, higher than each of the two methods alone (also the performance on the new class is higher).

If we apply a $L2$ loss at the intermediate features space, i.e., to use $\mathcal{L}''_D$, the model achieves 71.6% of mIoU, which is 6.5% higher than the standard approach (in this case freezing the encoder does not change too much the results). It is noticeable that two completely different approaches to preserve knowledge from the previous model, namely "$M_1(20)$ with E_F, $\mathcal{L}'_D$" (which applies a cross-entropy between the outputs with encoder frozen) and "$M_1(20)$ with $\mathcal{L}''_D$" (which applies a $L2$-loss between features spaces), achieve similar and high results both on the new class and on old ones. Freezing the encoder in the latter case leads to similar results. Finally, if we combine the two proposed losses together we achieve very similar results to $\mathcal{L}''_D$ alone, thus we argue that the extra complexity is not necessary.

An interesting aspect is that the changes in performance on previously seen classes are correlated with the class being added. Some classes have even higher results in terms of mIoU than before because their prediction has been reinforced through the new training set. For example, objects of the classes *sofa* or *dining table* are typically present in scenes containing a $tv/monitor$. Classes containing uncorrelated objects that are not present inside the new set of samples $\mathcal{D}_1^{tr}$ instead get more easily lost, for example the *bird* or *horse* which are not present in indoor scenes typically associated with the $tv/monitor$ class being added.

4.2. Addition of Five Classes

In this section we tackle a more challenging scenario where the initial learning stage is followed by one step of incremental learning with the last 5 classes to learn. First, the addition of the last 5 classes at once (referred to as $15-20$) is discussed and the results are shown in Table 2. In this setting the results are much lower than in the previous cases where a single class was added at a time since there is a larger amount of information to be learned. In particular, the fine-tuning approach exhibits an even larger drop in accuracy because it overestimates the presence of the new classes. We can confirm this by looking at the IoU scores of the newly added classes which are often lower than the proposed approaches by a large margin. In this setting the distillation on the output layer, "$M_1(16-20)$ with $\mathcal{L}'_D$", achieves the highest accuracy. In general in this case the approaches based on $\mathcal{L}'_D$ outperform the other ones. It is interesting to notice that some previously seen classes exhibit a clear catastrophic forgetting phenomenon because the updated models mislead them with visually similar classes belonging to the set of new classes. This is particularly true, for example, for the *cow* and *chair* classes which are often misled (low IoU and low pixel accuracy for these classes) with the newly added classes *sheep* and *sofa* that have similar shapes (low IoU but high pixel accuracy for these classes). This can be seen also in the qualitative results in Fig. 3. For example, in the first two rows the $tv/monitor$ and the *sofa* classes (which are added during the incremental step) are erroneously predicted in the *background* region, while these classes are correctly handled by applying $\mathcal{L}'_D$ and freezing the encoder. Additionally, in the third row the naïve approach predicts the *person* class in spite of the *sofa* while this artifact is not present when using $\mathcal{L}'_D$.

The last experiment presented here is the one in which the last 5 classes are progressively added one by one: the final model is referred to as $M_5(16 \rightarrow 20)$. The results are reported in Table 3 where we can appreciate a large gain of 20% of mIoU between the best proposed method (i.e., "$M_5(16 \rightarrow 20)$ with $E_F, \mathcal{L}'_D$") and the standard approach.

$M_1(16-20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	**mIoU old**	plant	sheep	sofa	train	tv	**mIoU new**	**mIoU**	**mPA**	**mCA**
Fine-tuning	89.7	59.5	34.6	68.2	58.1	58.8	59.2	79.2	80.2	30.0	12.7	51.0	72.5	61.7	74.4	79.4	60.6	36.4	32.4	27.2	55.2	42.4	38.7	55.4	88.4	70.6
$\mathcal{L}'_D$	91.4	85.0	35.6	84.8	61.8	70.5	85.6	77.9	83.6	30.7	72.0	45.4	76.1	76.9	77.0	81.3	**71.0**	33.8	54.9	30.8	73.9	51.6	**49.0**	**65.7**	**91.6**	**78.0**
$E_F, \mathcal{L}'_D$	91.7	83.4	35.6	78.7	60.9	73.0	65.8	82.2	87.0	30.2	58.0	55.3	80.0	78.3	78.5	81.4	70.0	35.3	46.1	32.3	62.1	53.5	45.8	64.2	91.5	76.1
$\mathcal{L}''_D$	90.9	81.4	33.9	80.3	61.9	67.4	73.1	81.8	84.8	31.3	0.4	55.8	76.1	72.2	77.7	81.2	65.6	39.4	31.8	31.3	64.1	52.9	43.9	60.5	90.0	74.9
$M_0(0-15)$	94.0	83.5	36.1	85.5	61.0	77.7	94.1	82.8	90.0	40.0	82.8	54.9	83.4	81.2	78.3	83.2	75.5	-	-	-	-	-	-	75.5	94.6	86.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	75.1	52.3	82.4	51.1	86.4	70.5	68.5	73.6	93.9	84.2

Table 2. Per-class IoU on the Pascal VOC2012 under some settings when 5 classes are added all at once.

$M_5(16 \rightarrow 20)$	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	**mIoU old**	plant	sheep	sofa	train	tv	**mIoU new**	**mIoU**	**mPA**	**mCA**
Fine-tuning	87.9	25.6	29.0	51.2	1.7	57.8	10.5	64.8	80.5	30.8	22.9	52.7	66.8	52.1	51.9	78.1	47.8	**36.5**	44.7	**31.8**	35.1	17.1	33.0	44.2	86.1	55.7
$\mathcal{L}'_D$	89.7	51.2	29.7	77.5	15.0	62.7	29.1	78.5	75.7	24.4	55.6	44.8	76.2	62.5	65.6	80.1	57.4	25.5	35.7	30.8	42.3	40.4	34.9	52.0	88.6	63.2
$E_F, \mathcal{L}'_D$	91.1	73.9	31.9	81.4	59.5	71.9	73.1	82.1	87.1	27.2	77.4	56.4	79.1	79.9	76.1	80.7	**70.5**	31.6	55.3	30.4	**62.2**	**41.4**	44.2	**64.3**	**91.3**	**75.2**
$\mathcal{L}''_D$	90.3	54.2	28.2	78.4	52.5	69.8	59.5	78.5	86.3	28.8	72.3	57.4	76.3	77.1	65.8	79.3	65.9	36.3	**65.5**	31.6	54.7	38.9	**45.4**	61.0	90.4	71.0
$M_0(0-15)$	94.0	83.5	36.1	85.5	61.0	77.7	94.1	82.8	90.0	40.0	82.8	54.9	83.4	81.2	78.3	83.2	75.5	-	-	-	-	-	-	75.5	94.6	86.4
$M_0(0-20)$	93.4	85.4	36.7	85.7	63.3	78.7	92.7	82.4	89.7	35.4	80.9	52.9	82.4	82.0	76.8	83.6	75.1	52.3	82.4	51.1	86.4	70.5	68.5	73.6	93.9	84.2

Table 3. Per-class IoU on the Pascal VOC2012 under some settings when 5 classes are added sequentially.

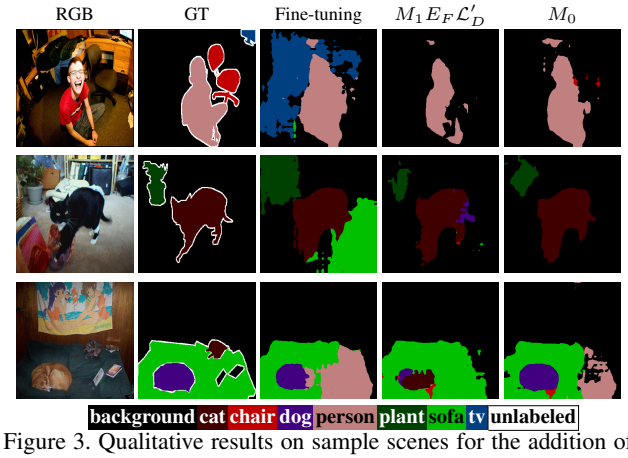

Figure 3. Qualitative results on sample scenes for the addition of five classes all at once (*best viewed in colors*).

	Fine-tuning			$\mathcal{L}'_D$			$E_F, \mathcal{L}'_D$			$\mathcal{L}''_D$		
	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA	mIoU	mPA	mCA
$M_1(16)$	71.2	93.7	82.5	72.4	94.2	83.0	72.5	94.1	83.5	72.2	93.9	84.3
$M_2(17)$	53.8	90.0	61.8	68.1	93.4	78.5	68.4	93.3	79.5	60.0	91.6	69.4
$M_3(18)$	57.7	87.7	68.7	63.3	90.8	74.5	66.5	91.5	79.4	65.5	90.7	76.8
$M_4(19)$	39.3	85.9	47.4	54.1	89.2	64.3	61.3	90.6	72.5	52.1	89.0	60.6
$M_5(20)$	44.2	86.1	55.7	52.0	88.6	63.2	64.3	91.3	75.2	61.0	90.4	71.0

Table 4. Mean IoU, mPA and mCA on the Pascal VOC2012 under some settings when 5 classes are added sequentially.

In this case freezing the encoder and distilling the knowledge is the best approach because the addition of one single class do not alter too much the responses of the whole network: distilling the knowledge from the previous model when the encoder is fixed guides the decoder to modify only the responses for the new class.

The evolution of the models' mean performance over time is reported in Table 4 where the distribution of the drop of performance during the different steps is analyzed. In particular we can notice how the accuracy drop is affected by the specific class being added. As expected the larger drop is experienced when the classes *sheep* or *train* are added (models $M_2(17)$ and $M_4(19)$) because such classes are only sparsely correlated with other classes (they mainly appear alone or with the *person* class).

5. Conclusion and Future Work

In this work we formally introduced the problem of incremental learning for semantic segmentation. A couple of novel distillation loss functions have been designed ad-hoc for the task. They have been combined with a cross-entropy loss and with the idea of freezing the encoder module to optimize the performance on new classes while preserving old ones. Our method does not need any stored image of previous datasets and only the previous model is used to update the current one thus reducing memory consumption.

Experiments on the Pascal VOC2012 dataset show that the proposed methods were able to largely outperform the standard fine-tuning approach, thus alleviating the catastrophic forgetting phenomenon. However, the problem of incremental learning for semantic segmentation is a novel challenging task that needs advanced strategies to be tackled. This is proved by the fact that the results are lower than the ones achieved by the same architecture after a one-step training, i.e., when all training examples are available and employed at the same time. In the future we plan to expand our set of experiments, to develop novel incremental learning strategies and to employ GANs to generate images containing already seen classes. Finally we will consider the scenario in which classes that will appear in the future are present from the beginning but labeled as background.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016. 4

[2] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 139–154, 2018. 2

[3] M. Biasetton, U. Michieli, G. Agresti, and P. Zanuttigh. Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019. 2

[4] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 535–541. ACM, 2006. 1

[5] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 233–248, 2018. 1

[6] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 409–415, 2001. 1

[7] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 1

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2018. 3, 4

[9] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan. What is a good evaluation measure for semantic segmentation? In *BMVC*, volume 27, page 2013. Citeseer, 2013. 4

[10] 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. 2, 4

[11] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 1

[12] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan. Active long term memory networks. *arXiv preprint arXiv:1606.02355*, 2016. 1

[13] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *International Conference on Learning Representations (ICLR)*, 2014. 1

[14] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *Neural Information Processing Systems (NIPS) Deep Learning and Representation Learning Workshop*, 2015. 1

[15] S. Hou, X. Pan, C. Change Loy, Z. Wang, and D. Lin. Lifelong learning via progressive distillation and retrospection. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 437–452, 2018. 1

[16] R. Istrate, A. C. I. Malossi, C. Bekas, and D. Nikolopoulos. Incremental training of deep convolutional neural networks. *arXiv preprint arXiv:1803.10232*, 2018. 1

[17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017. 1, 2

[18] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(12):2935–2947, 2018. 1, 2, 3

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. 3

[20] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6467–6476, 2017. 1

[21] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[22] U. Michieli and L. Badia. Game Theoretic Analysis of Road User Safety Scenarios Involving Autonomous Vehicles. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1377–1381, 2018. 2

[23] V. Nekrasov. Pre-computed weights for ResNet-101. https://github.com/DrSleep/tensorflow-deeplab-resnet, Accessed: 2019-07-04. 3

[24] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014. 1

[25] R. Polikar, L. Upda, S. S. Upda, and V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, part C (Applications and Reviews)*, 31(4):497–508, 2001. 1

[26] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010, 2017. 1

[27] D. Roy, P. Panda, and K. Roy. Tree-CNN: a hierarchical deep convolutional neural network for incremental learning. *arXiv preprint arXiv:1802.05800*, 2018. 1

[28] S. S. Sarwar, A. Ankit, and K. Roy. Incremental learning in deep convolutional neural networks using partial network sharing. *arXiv preprint arXiv:1712.02719*, 2017. 1

[29] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2990–2999, 2017. 1

[30] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 3400–3409, 2017. 1, 2, 3, 4

[31] R. V. Singh, P. Dhar, K.-C. Peng, Z. Wu, and R. Chellappa. Learning without memorizing. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[32] O. Tasar, Y. Tarabalka, and P. Alliez. Incremental Learning for Semantic Segmentation of Large-Scale Remote Sensing Data. *arXiv preprint arXiv:1810.12448*, 2018. 1

[33] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, pages 640–646, 1996. 1

[34] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[35] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018. 1

[36] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM International Conference on Multimedia (ACMMM)*, pages 177–186. ACM, 2014. 1

[37] P. Zhou, L. Mai, J. Zhang, N. Xu, Z. Wu, and L. S. Davis. M2kd: Multi-model and multi-level knowledge distillation for incremental learning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2