Incremental Learning in Semantic Segmentation

Salvatore Stefano Furnari
Politecnico di Torino
s290057@studenti.polito.it

Giuseppe Gallipoli
Politecnico di Torino
s291086@studenti.polito.it

Andrea Ghiglione
Politecnico di Torino
s287845@studenti.polito.it

Abstract

Semantic segmentation is one of the most widespread Computer Vision tasks thanks to its applications in different domains. However, to be effectively deployed in real-life systems, it has to be adapted to the incremental learning scenario. In order to explore these challenging problems, we start from an offline implementation of BiSeNet, finding the best configuration of hyper-parameters. Then we move towards an incremental class learning scenario following MiB protocol which deals with the well-known catastrophic forgetting problem of deep networks through the modeling of the background. Finally, taking into account the difficulty of having a large dataset with full pixel-level annotations, we try to learn a segmentation network under an incremental weakly-supervised setting. All the experiments are performed on the Pascal-VOC 2012 dataset, confirming both the effectiveness of MiB method even with a different backbone network and the open issues represented by the incremental weakly-supervised approach.

1. Introduction

Semantic segmentation is undoubtedly among the most difficult tasks in the field of Computer Vision. Tracing a path from coarse to fine inference, semantic segmentation lies on the latter extreme, having as objective the assignment of a class label to each pixel in an image [1]. Since in an image there are very many objects and only a limited amount of classes, semantic segmentation is characterized by an additional special class called background, which indicates pixels not assigned to any of the given objects. If we consider an offline scenario, *i.e.* where all the classes are statically defined, the background class is semantically constant. However, in a more realistic scenario, there may be the need to keep learning new classes that were not considered in the early stages of the classification, resulting in a semantic shift of the background class. A naive solution could be the one of training again the model from scratch with these additional new classes, resulting in a very expensive procedure. To alleviate the burden of continuous re-training, a smarter solution is to perform classification in an incremental class learning setting, where the training is realized over multiple phases called learning steps and each of these steps introduces new classes to be learnt. In the latter scenario, despite the computation time is reduced, we have to consider that the semantics associated to the background class may change across different incremental steps, and so pixels associated to the background during a learning step may be assigned to a specific object class in the following learning steps, or vice-versa. In addition, this problem exacerbates the catastrophic forgetting [18], which is the tendency of deep architectures to worsen their performances when required to update their knowledge and the original training set is no longer available. Furthermore, semantic segmentation is a time-consuming task for human annotators because it requires images which are labeled at pixel level [1]. As a consequence, in a more realistic scenario the majority of images are provided with partial annotations that could be, for instance, a simple indication of which classes are present into the image or labeling few pixels only.

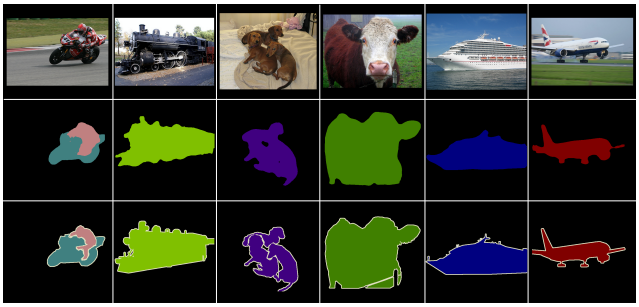

Figure 1. Semantic segmentation task using BiSeNet architecture in an offline scenario. From top to bottom: image, prediction and ground truth.

The first goal of our work is to implement the Bilateral Segmentation Network, *a.k.a.* BiSeNet [22], which provides near real-time inference, in an offline scenario. Looking for the best set of parameters, we test it on the Pascal-VOC 2012 dataset [8] obtaining satisfactory results. In second place we move to an incremental setting, repli-

cating some of the experiments performed in MiB [3], that deals with incremental learning in semantic segmentation by modeling the background. The authors use a DeepLab-V3 architecture [6], but having limited computational resources at our disposal, we use the lighter BiSeNet. Finally, we analyze the challenging scenario in which the learning steps provide a weak supervision only, in particular we work with image-wise labels. To summarize, the main contributions of this paper are:

- We implement BiSeNet in an offline scenario, testing several sets of parameters.
- We replicate the experiments of MiB paper using BiSeNet architecture, working in an incremental class learning scenario.
- We study the challenging task of working in an incremental scenario with a weak supervision.

2. Related works

2.1. Semantic segmentation

Semantic segmentation is one of the most ambitious tasks in Computer Vision and its results progressed in a significant way thanks to deep architectures spread (such as AlexNet [12], VGGNet [20], GoogleNet [21] and ResNet [10]): [17] was one of the first works to exploit the "representational power" of the DCNNs (implemented in a fully-convolutional fashion), which are able to extract high-level features useful for the detection of the presence of object classes in images. From this intuition, successive works further investigated to find the best way to conciliate the two main aspects of semantic segmentation, which are the classification of the objects and their localization pixel by pixel. DeepLab-V1[4] made use of the graphical model CRF [13] as a post-processing technique to better refine the boundaries of the various objects; besides, authors introduced a new type of convolution operation, which is the *atrous* one (also called *dilated*), in order to maintain a larger receptive field without having to learn additional parameters and alleviating the burden of the upsampling operation. This work entered among the state-of-the-art solutions for semantic segmentation tasks, with the benefit of further upgrades: in the second version [5], taking inspiration from the Spatial Pyramid Pooling technique [9], authors implemented ASPP, which consists of a parallel execution of atrous convolutions at different rates, to make inference at various scales and capture the presence of objects of any size. In the third version [6], CRF was abandoned in favor of an enrichment of the ASPP module.

2.2. Incremental learning

The necessity of neural networks suitable to incremental learning takes place from the objective of employing these systems in a realistic scenario of online training, with a stream of new incoming samples and previously unseen classes involved. The main obstacle in this task is the *catastrophic forgetting* [18] to which all neural networks are subject: during the learning procedure, the presence of new classes induces the network to update its parameters in a way that is destructive for the preservation of the knowledge of previous classes. In this sense, we can distinguish incremental learning solutions with respect to the way in which they deal with this undesired effect, going in the direction of natural systems (like human brain), that do not forget about previously acquired knowledge. Among the most prominent ones, we find [19], which falls into the category of the replay-based methods: indeed, the core of this work is that the network keeps in its memory the most representative sample images of all the classes seen until the current iteration (the so-called *exemplars*). In this way, the loss function can promote the learning of new classes, while trying to maintain the same scores obtained for the old classes. The majority of previous work on semantic segmentation has been carried out under an offline learning procedure and the challenges posed by the incremental learning framework have been treated in a limited manner. [3] is one of the first papers to address the topic of "background shift". In fact, if we do not have a fixed number of classes to be segmented, the network has to deal with the mutable placement of the background class, which could be located into pixels belonging to previously seen classes: so the loss function has to be constructed in a way that, following the general principle of [14], *distillates* the knowledge in all the training steps, without the need of storing exemplar sets.

2.3. Weak supervision

The works mentioned in 2.1 could learn a segmentation network with a full-supervised dataset. However, this type of annotation is expensive and requires a very long time, even with the efforts of expert annotators. To overcome this obstacle, it was investigated the possibility to obtain satisfactory performances adopting less time-expensive annotations, which include image-level labels, scribbles and points. In the first path, SEC [11] was one of the first leading works. Starting only from the indication of which classes are present into the images, the authors defined a three-terms loss function based on three guiding principles: to seed with weak localization cues, to expand objects based on the information about which classes can occur in an image and to constrain the segmentations to coincide with object boundaries. For what concerns the second direction, it is added a new level of supervision with a not so relevant amount of time required: annotators have just to draw some lines in correspondence of the objects present into the images, providing in this way a few labeled pixels for each class that can help to *propagate* this information to the other

pixels inferring a full-image segmentation mask [15].

Finally, considering the case in which points are provided (one or very few pixels) for each class present into the image, [1] delineates an interesting approach, based on a loss function suited to the point-supervision and on an *objectness prior* which helps the segmentation network to properly infer the spatial extent of the objects.

3. Method

3.1. BiSeNet in an offline setting

Our first step consists in implementing BiSeNet architecture in an offline scenario, testing it on the Pascal-VOC 2012 dataset. This architecture is composed of four main parts: a Spatial Path, a Context Path, an Attention Refinement Module (ARM) and a Feature Fusion Module (FFM). The Spatial Path produces a spatially accurate feature map, thanks to the moderate downsampling. In particular, it is composed of three layers, including a stride 2 convolution, a batch normalization and a ReLU activation function. The Context Path provides an appropriate receptive field and an efficient computation, to address performance and speed. It uses a lightweight model like Xception [7] to get a large receptive field. Then, on its tail, a global average pooling (GAP) [16] is added and finally the features of the pooling and the features of the lightweight model are combined. We work on the Context Path trying different networks; in particular we do experiments with ResNet18, ResNet50 and ResNet101 [10], modifying some model's parameters in order to achieve the best performance. ARM is used to refine the output feature of each stage in the Context Path. An image is processed in parallel by Spatial and Context Path and, finally, the FFM is used to combine the features provided by these two paths, which are low and high level features, respectively. A loss function is used to supervise the output of BiSeNet with two auxiliary loss functions to supervise the Context Path that are balanced through an hyper-parameter. Moreover, this network tackles the trade-off between inference speed and accuracy, reaching a satisfactory balance.

3.2. MiB protocol with BiSeNet architecture

As second step, we implement the incremental learning protocol following [3] using BiSeNet and we test it on four baselines: Fine-Tuning (FT), Learning without Forgetting (LwF), Incremental Learning Techniques (ILT) and Modeling the Background (MiB). The main contributions of MiB protocol are a new distillation-based framework that takes into account the background shift, by revisiting both cross-entropy and knowledge-distillation losses, and a novel strategy to initialize classifier's parameters.

The loss function with parameters θ at learning step t which is minimized is:

$$\mathcal{L}(\theta^t) = \frac{1}{|\mathcal{T}^t|} \sum_{(x,y)\in\mathcal{T}^t} (l_{ce}^{\theta^t}(x,y) + \lambda l_{kd}^{\theta^t}(x)) \quad (1)$$

where $\mathcal{T}$ is the training set, l_{ce} is a standard supervised loss (e.g. cross-entropy loss), l_{kd} is the distillation loss and λ is a hyper-parameter to balance the weights of the two terms. In order to take into account the possibility that the background class in step t may include pixels associated to previously seen classes, the cross-entropy loss in Eq.(1) is defined as follows:

$$l_{ce}^{\theta^t}(x,y) = -\frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \log \tilde{q}_x^t(i, y_i) \quad (2)$$

where $y_i \in \mathcal{Y}^t$ is the ground truth label of pixel i,

$$\tilde{q}_x^t(i,c) = \begin{cases} q_x^t(i,c) & \text{if } c \neq b \\ \sum_{k\in\mathcal{Y}^{t-1}} q_x^t(i,k) & \text{if } c = b \end{cases} \quad (3)$$

and $q_x^t(i,c) = f_{\theta^t}(x)[i,c]$.

To handle the fact that the background class is shared among different learning steps, which means that annotations for background in $\mathcal{T}^s$ with $s < t$ might include pixels of classes in $\mathcal{C}^t$, the distillation loss in Eq.(1) is written as:

$$l_{kd}^{\theta^t}(x) = -\frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \sum_{c\in\mathcal{Y}^{t-1}} q_x^{t-1}(i,c) \log \hat{q}_x^t(i,c) \quad (4)$$

where

$$\hat{q}_x^t(i,c) = \begin{cases} q_x^t(i,c) & \text{if } c \neq b \\ \sum_{k\in\mathcal{C}^t} q_x^t(i,k) & \text{if } c = b \end{cases} \quad (5)$$

and $\mathcal{C}^t$ is the set of new classes.

As for the initialization of classifier's parameters, in each learning step t, unless the appearance of a class in $\mathcal{C}^t$ is very similar to one in $\mathcal{Y}^{t-1}$, the old model will probably assign pixels of new classes to b. Instead of randomly initialize the classifier for the new classes, the authors propose to uniformly spread the probability of the background $q_x^{t-1}(i,b)$ among the classes in $\mathcal{C}^t$. Weights ω and bias β parameters for a class c are initialized as follows:

$$\omega_c^t = \begin{cases} \omega_b^{t-1} & \text{if } c \in \mathcal{C}^t \\ \omega_c^{t-1} & \text{otherwise} \end{cases} \quad (6)$$

$$\beta_c^t = \begin{cases} \beta_b^{t-1} - \log(|\mathcal{C}^t|) & \text{if } c \in \mathcal{C}^t \\ \beta_c^{t-1} & \text{otherwise} \end{cases} \quad (7)$$

To adapt BiSeNet architecture to an incremental scenario, we replace the last convolutional layers of the three outputs of BiSeNet with a list of classifiers, each with the appropriate number of output channels set to the number of classes

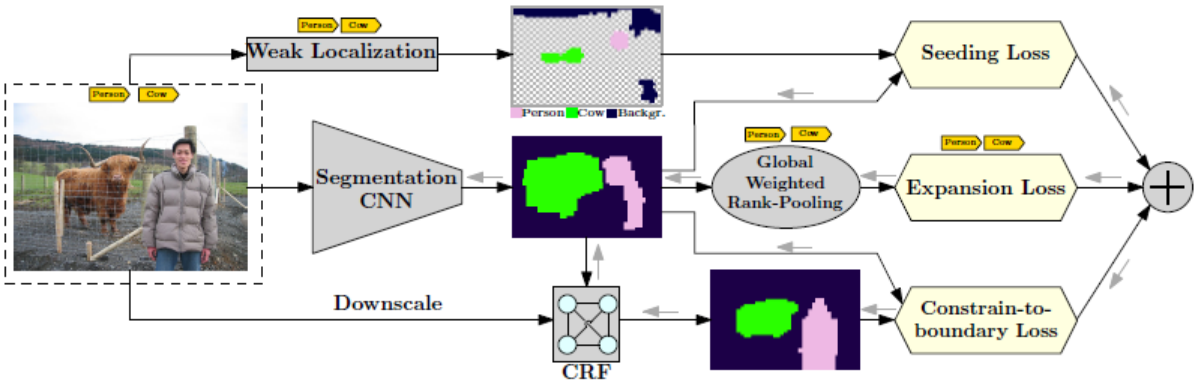

Figure 2. Weak supervision with SEC protocol, based on the minimization of a loss function consisting of three terms: *Seeding Loss* to provide insights about the localization of the most discriminative parts of the objects, *Expansion Loss* to look for the reasonable extent of the object instances and *Constrain-to-boundary Loss* to make the inferred segmentation fit the exact boundaries of the items.

of the previous and current learning steps. The classifiers are then applied in sequence to the outputs and the results are concatenated to generate the final predictions. Since the total number of lists of classifiers is three, the proposed initialization strategy is repeated for each of them. BiSeNet architecture described in 3.1 uses as internal number of channels in FFM the number of classes but, since in an incremental scenario it is not constant, we adopt a fixed value of 32 which has been chosen as the closest to the original one of 21 and also considering memory limitations. Another difference with respect to DeepLab-V3 architecture is that the loss function is computed for each of the three outputs of BiSeNet. As for the distillation techniques, we follow the same approach used by the authors while for ILT method we apply it on the pre-logits of the main output of BiSeNet.

3.3. SEC protocol in an incremental setting

In the third step, as an additional study we implement the weak supervision SEC protocol, shown in Figure 2, embedding it into the incremental learning scenario of the previous step. Also in this case we use BiSeNet architecture and we test it on the four baselines described in 4.3. The main contribution of SEC protocol is a new loss function composed of three terms which are respectively based on three principles: **S**eed, **E**xpand and **C**onstrain.

The first term, L_{seed}, provides localization hints to the network, the second term, L_{expand}, penalizes the prediction of segmentation masks with too small or wrong objects and the third term, $L_{constrain}$, encourages segmentations that respect both the spatial and color structure of the images. The loss function with parameters θ which is minimized is:

$$\min_{\theta} \sum_{(X,T)\in\mathcal{D}} [L_{seed} + L_{expand} + L_{constrain}] \quad (8)$$

where X is an image and T is the set of weak annotations; each term is a function of $(f(X,\theta),T)$ and they will be explained in detail in the following.

$$L_{seed}(f(X),T,S_c) = -\frac{1}{\sum_{c\in T}|S_c|}\sum_{c\in T}\sum_{u\in S_c}\log f_{u,c}(X) \quad (9)$$

where S_c is a set of locations u labeled with class c by the weak localization procedure and $f_{u,c}(X) = p(y_u = c|X)$ with the semantic label y_u. To reduce the computational effort and memory consumption, we use the set of localization cues S_c precomputed by the authors.

$$\begin{aligned} L_{expand}(f(X),T) = &-\frac{1}{|T|}\sum_{c\in T}\log G_c(f(X);d_+) \\ &-\frac{1}{|\mathcal{C}'\setminus T|}\sum_{c\in\mathcal{C}'\setminus T}\log(1-G_c(f(X);d_-)) \\ &-\log G_{c^{bg}}(f(X);d_{bg}) \end{aligned} \quad (10)$$

where $G_c(f(X);d_c) = \frac{1}{\sum_{j=1}^{n}(d_c)^{j-1}}(d_c)^{j-1}f_{i_j,c}(X)$, $f_{i_j,c}$ is the prediction score for class c, $\mathcal{C}'$ is the set of foreground labels and d_c a decay parameter but for simplicity the authors only distinguished between three groups: d_+ for object classes that occur in an image, d_- for those that do not occur and d_{bg} for background.

$$L_{constrain}(f(X),X) = \frac{1}{n}\sum_{u=1}^{n}\sum_{c\in\mathcal{C}} Q_{u,c}\log\frac{Q_{u,c}}{f_{u,c}(X)} \quad (11)$$

where $Q(X,f(X))$ is a fully-connected CRF for which we exploit the implementation of the authors, with unary potentials given by the logarithm of the predicted probabilities and pairwise potentials depending only on image pixels.
To embed SEC protocol into the incremental learning setting described in 3.2, the main changes with respect to the implementation of the previous step are related to the new

Batch size	Crop size	Learning rate	Context Path	Data augmentation	Loss	mIoU Val	Precision Val	mIoU Train	Precision Train
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomRotation(15)	0.6389	0.634	0.905	0.837	0.939
32	480	$1 \cdot 10^{-2}$	ResNet18	Default	0.5723	0.628	0.902	0.867	0.948
16	480	$0.5 \cdot 10^{-2}$	ResNet50	Default	0.3786	0.705	0.915	0.898	0.959
16	480	$1 \cdot 10^{-2}$	ResNet50	Default	0.3662	0.690	0.914	0.898	0.959
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default RandomRotation(15)	0.3310	0.721	0.919	0.891	0.958
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default ColorJitter(0.3, 0.3, 0.3, 0.1) RandomRotation(15)	0.3713	0.716	0.909	0.879	0.954

Table 1. Most remarkable results obtained with different Context Paths and parameters – 30 epochs

Batch size	Crop size	Learning rate	Context Path	Data augmentation	Loss	mIoU Val	Precision Val	mIoU Train	Precision Train
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default RandomRotation(15)	0.2963	0.729	0.921	0.902	0.961

Table 2. Best result obtained with different Context Paths and parameters – 40 epochs

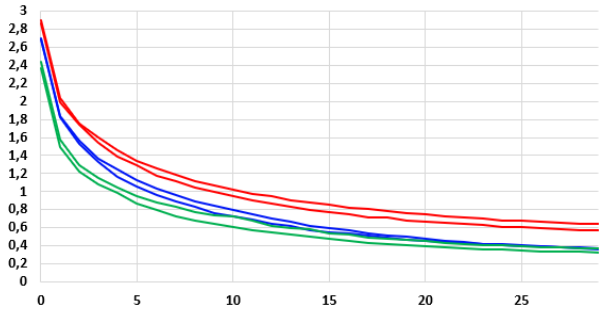

Figure 3. Loss functions related to Table 1

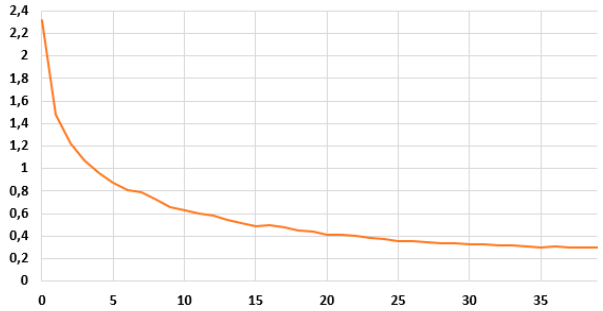

Figure 4. Loss function related to Table 2

loss function, composed of the three terms discussed above, which is applied to each of the three outputs of BiSeNet architecture, replacing in this way the supervised loss function of MiB protocol. As for the additional losses applied in the incremental steps, they are maintained with the respective distillation techniques of the different baselines we compare. To load the correct images with the related image-level labels and localization cues, we exploit the same strategy and masking procedure of MiB protocol, which means to mask training data related to classes that do not belong to the current learning step.

4. Experiments and Results

4.1. Pascal-VOC 2012

The Pascal Visual Object Classes dataset of 2012 has been widely used as a benchmark for Computer Vision tasks like classification, object detection, semantic segmentation, action classification and person layout. It contains 20 object classes, which belong to four groups that are person, animal, vehicle and indoor. The dataset is split in a training set composed of 10582 images and a validation set made of 1449 images. We use this dataset to generate pixel-wise segmentations, classifying pixels as object classes or as background. Specifically, in the first place we do experiments in an offline scenario, considering all the 20 object classes from the beginning. Secondly, we move to an incremental class learning scenario considering two settings. The first is called *15-5* and it corresponds to start the training with 15 classes and then adding 5 classes all at once in the following learning step. The second is called *15-1* and it consists in starting the training with 15 classes and then adding the 5 classes in sequential steps of 1 class per step. For both settings we use the overlapped scenario, *i.e.* assuming that each learning step contains all the images which have at least one pixel of a novel class, with only the latter annotated.

4.2. Offline BiSeNet implementation details

In the first step we considered an offline setting and we looked for the best performance obtained training BiSeNet on the Pascal-VOC 2012 dataset. We worked on batch size, crop size, learning rate and data augmentation. In addition, we tried 3 different Context Paths: ResNet18, ResNet50 and ResNet101, for a total of 54 experiments. We started experiments with ResNet18, since it is the least computationally heavy, and we followed a specific procedure:

1. We sought for the batch and crop sizes that saturated the GPU provided by Google Colaboratory, keeping learning rate and data augmentation as default. In par-

ticular, default value of learning rate is $1 \cdot 10^{-2}$ and the default data augmentation consists of:

- RandomResizedCrop(320, (0.5, 2.0))
- RandomHorizontalFlip()

2. For each tuple of batch and crop size, we tried the following learning rates: $\{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}\}$. We selected the configuration with the best result and we did further experiments multiplying the learning rate by a factor of $\{0.5, 2, 5\}$ which are a common good practise set of values.

3. Since the computation time is high, we exploited the combination of parameters with a low batch and crop size, in order to try different types of data augmentations analyzing how they are able to reduce overfitting.

Once we ended experiments with ResNet18, we had a general overview of which parameters worked well in terms of results and we exploited this information on the following experiments on ResNet50 and ResNet101, that would reasonably be as sensitive as ResNet18 to the parameters change. In both cases, however, we repeated the test on the saturation of the GPU since these two architectures are heavier. ResNet101 obtained the best results, as reported in Table 1. However, it is the network with the most limited number of experiments, since the GPU was saturated with very many combinations of batch and crop size. All the experiments were performed on 30 epochs, but the final best configuration model has been trained for 40 epochs (Table 2) for the sake of completeness. The respective loss functions are reported in Figure 3 and Figure 4.

Note: only some significant results are reported, while in Appendix A. [5] there is the entire set of 54 experiments and their results.

4.3. MiB & other ICL baselines experiments

The four baselines we compare are:

- Fine-Tuning (FT): simple fine tuning on each $\mathcal{T}^t$.
- Learning without Forgetting (LwF): original loss defined in Eq.(1) with basic cross-entropy and distillation losses.
- Incremental Learning Techniques (ILT): it uses a distillation loss in the features space, as described in 3.2.
- Modeling the Background (MiB): loss defined in Eq.(1) with revisited cross-entropy and distillation losses and initialization strategy.

The table also reports the final results obtained in 4.2 training on all classes offline (Joint). All results in Table 3 are reported as mean Intersection-over-Union (mIoU) on the validation set of the dataset, averaged over all the classes of a learning step and computed at the end of each task.

	15-5			**15-1**		
Method	*1-15*	*16-20*	*all*	*1-15*	*16-20*	*all*
FT	0.003	0.217	0.056	0.001	0.005	0.002
LwF	0.411	0.225	0.365	0.016	0.019	0.017
ILT	0.354	0.188	0.312	0.016	0.055	0.026
MiB	0.619	0.297	0.539	0.305	0.080	0.249
Joint	0.748	0.671	0.729	0.748	0.671	0.729

Table 3. mIoU on Pascal-VOC 2012 dataset for different baselines

All methods were implemented using BiSeNet architecture with the best configuration of parameters reported in Table 2. To improve the efficiency of the method, the Context Path has been initialized using the ImageNet pretrained model based on InPlace-ABN approach described in [2]. We followed MiB protocol by training the network with SGD, the same learning rate policy, momentum, weight decay and the same hyper-parameters of each method. We trained the model for 40 epochs with an initial learning rate of $0.5 \cdot 10^{-2}$ for the first learning step and $0.5 \cdot 10^{-3}$ for the followings.

Task 15-5. In this first task we can notice a significant drop in the performance which underlines how challenging the incremental scenario is and consequently the importance of address it properly to prevent, or better to limit, the well-known phenomenon of *catastrophic forgetting*. It is particularly evident observing the performance of the four methods on the first 15 classes, while for the second set of labels they perform almost similarly. FT forgets almost everything of the first learning step while the other approaches are able to retain a certain amount of knowledge and, if we rank their performance in an increasing order, we find ILT followed by LwF and MiB which outperforms the other baselines by nearly 15%. By comparing our results with those in [3], we can notice little differences due to the fact we used BiSeNet instead of DeepLab-V3 but the "ranking" of the four methods is maintained with the exception of ILT which performs worse than LwF. The possible cause is that ILT may be over-regularized, that means we could need a smaller weight since we used both a different network and different features for distillation with respect to the authors.

Task 15-1. In this second task the last 5 classes are learnt sequentially and this results in a more challenging scenario with a more profound drop in the performance. The predictions are biased towards new classes with the difference that, since they are added one at a time, there is a form of forgetting also for them. All the methods show poor performance, lower than 6%, on both old and new classes, while the outcomes of MiB are different since it is the only approach that is able to retain knowledge of old classes and outperforms the other baselines with an increase in both old (almost 30%) and new (more than 20%) classes. In this task the "ranking" of the methods is coherent with the one assessed by the authors, with the usual differences due to

the different segmentation network used.

In Appendix B. [5] it is possible to find some qualitative results of the four methods, their confusion matrices and the detailed mIoU per class on the validation set.

4.4. Incremental SEC experiments

We used BiSeNet architecture as segmentation network with the best configuration of parameters reported in 2 while for the localization procedure, we exploited the precomputed cues provided by the authors which are derived from the standard VGG architecture finetuned on the Pascal-VOC 2012 dataset. We adopted the suggested decay parameters involved in L_{expand} of $d_+ = 0.996$, $d_- = 0$, $d_{bg} = 0.999$ and we used the default values for the fully-connected CRF at training time. To refine the segmentation at validation time, we employed again a fully-connected CRF.

We tested the incremental version of SEC on the same four baselines described in 4.3. All results in Table 4 are reported as mean Intersection-over-Union (mIoU) on the validation set of the dataset, averaged over all the classes of a learning step and computed at the end of each task. We trained the network with the same policy adopted in the previous step, both in terms of learning rate (40 epochs, learning rate $0.5 \cdot 10^{-2}$ for the first learning step and $0.5 \cdot 10^{-3}$ for the followings) and hyper-parameters of each method.

Note: we are aware that using different parameters from the suggested ones by the authors will probably lead to suboptimal results, but this was done to allow a comparison as close as possible to the experiments carried out in 4.3. In addition to this, it is important to highlight that the configuration adopted in 4.3. is well-suited for an incremental scenario with a full supervision approach while the one proposed in [11] is tailored for an offline setting with a weak supervision technique. As a consequence, an additional tuning of the parameters should be performed.

Task 15-5. If the incremental scenario is more challenging than an offline training due to the problem of *catastrophic forgetting*, it is immediately clear that switching to a weak supervision approach poses additional challenges caused by the significantly lower amount of data provided at training time. We observe a further drop in the performance, especially for the first 15 classes (on average 30%), while the last 5 labels seem to suffer less this new approach. The overall results show a different trend with respect to the ones in Table 3 since the best method appears to be LwF followed by the other techniques with similar performance.

Task 15-1. In this second task, as expected by the corresponding one of the previous step, performance are even lower due to the sequential addition of the last 5 classes. The already critical results obtained by the fully-supervised approach practically go to zero (less than 0.5%) with the exception of MiB which is the best method in this particularly demanding task.

	15-5			**15-1**		
Method	*1-15*	*16-20*	*all*	*1-15*	*16-20*	*all*
SEC-FT	0.000	0.210	0.052	0.000	0.012	0.003
SEC-LwF	0.086	0.225	0.121	0.000	0.002	0.000
SEC-ILT	0.055	0.087	0.063	0.000	0.002	0.001
SEC-MiB	0.008	0.195	0.055	0.000	0.039	0.010

Table 4. mIoU on Pascal-VOC 2012 dataset for different baselines

The obtained results are not satisfactory but it is also true that a drop in the performance was more than expected, so we hope the experiments we carried out in this especially challenging scenario will encourage future works to overcome the current issues. Our initial hypothesis has also been confirmed: a new tuning of both learning rate and hyper-parameters of the ICL methods we tested should be performed to improve results in this novel scenario.
As a last element that could have affected the outcomes, we found that the provided image-wise labels and localization cues can generate some inconsistencies in an incremental context: due to their extraction procedure, they may contain only a subset of the actual classes of the image which, if masked, could result in only background pixels.
In Appendix C. [5] it is possible to find some qualitative results of the four methods, their confusion matrices and the detailed mIoU per class on the validation set for the task 15-5.

5. Conclusions

We first studied the task of semantic segmentation in an offline scenario using BiSeNet architecture, then we moved to an incremental learning setting in which we followed MiB protocol assessing its superiority with respect to the other ICL baselines we tested. Lastly, we implemented the weak supervision approach of SEC protocol adapting it to the incremental context. During our work, we found several ideas for further developments: in particular, more advanced segmentation networks could be adopted and, to improve results in the incremental framework, additional approaches could be considered such as the use of exemplars or generative methods to generate samples belonging to previous classes. As for the most challenging scenario we studied of incremental weak supervision, improvements on both sides could be made: in addition to those related to the incremental component, it would be possible to enhance the weakly-supervised approach using better priors both for object sizes, shapes and materials or more recent works on this topic that overcome the limits of localization maps.

References

[1] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision, 2016.

[2] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns, 2018.

[3] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2016.

[5] 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, 2017.

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

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[8] 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.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Lecture Notes in Computer Science*, page 346–361, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[11] Alexander Kolesnikov and Christoph H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation, 2016.

[12] Alex Krizhevsky, I Sutskever, and G Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 01 2012.

[13] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials, 2012.

[14] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017.

[15] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation, 2016.

[16] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better, 2015.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[18] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[22] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation, 2018.

Appendix A.

Batch size	Crop size	Learning rate	Context Path	Data augmentation	Loss	mIoU Val	Precision Val	mIoU Train	Precision Train
32	480	$1 \cdot 10^{-2}$	ResNet18	Default	0.5723	0.628	0.902	0.867	0.948
32	480	$2 \cdot 10^{-2}$	ResNet18	Default	0.5203	0.608	0.896	0.870	0.949
32	480	$5 \cdot 10^{-2}$	ResNet18	Default	1.3585	0.305	0.822	0.686	0.884
32	480	$1 \cdot 10^{-3}$	ResNet18	Default	1.3945	0.586	0.896	0.778	0.917
32	480	$1 \cdot 10^{-4}$	ResNet18	Default	2.4374	0.477	0.873	0.558	0.852
32	344	$1 \cdot 10^{-1}$	ResNet18	Default	1.3161	0.344	0.840	0.645	0.876
32	344	$0.5 \cdot 10^{-2}$	ResNet18	Default	0.7126	0.607	0.901	0.843	0.940
32	344	$1 \cdot 10^{-2}$	ResNet18	Default	0.5749	0.604	0.900	0.855	0.944
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomResizedCrop(344, (0.25, 2.0))	0.7071	0.601	0.895	0.838	0.934
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomResizedCrop(344, (0.75, 2.0))	0.4717	0.590	0.896	0.872	0.954
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomResizedCrop(344, (1.25, 2.0))	0.4077	0.587	0.898	0.885	0.958
32	344	$1 \cdot 10^{-2}$	ResNet18	Default ColorJitter(0.5, 0.5, 0.5, 0)	0.6401	0.609	0.892	0.842	0.940
32	344	$1 \cdot 10^{-2}$	ResNet18	Default ColorJitter(0.5, 0.5, 0.5, 0.5)	0.7017	0.584	0.881	0.831	0.935
32	344	$1 \cdot 10^{-2}$	ResNet18	Default ColorJitter(1.5, 1.5, 1.5, 0)	0.9050	0.585	0.886	0.781	0.917
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomHorizontalFlip(0.75)	0.5610	0.602	0.899	0.860	0.946
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomResizedCrop(344, (0.75, 2.0)) RandomHorizontalFlip(0.75)	0.4647	0.581	0.896	0.875	0.955
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomRotation(15)	0.6389	0.634	0.905	0.837	0.939
32	344	$1 \cdot 10^{-2}$	ResNet18	Default ColorJitter(0.5, 0.5, 0.5, 0) RandomRotation(15)	0.7162	0.608	0.879	0.819	0.933
32	344	$1 \cdot 10^{-2}$	ResNet18	Default RandomVerticalFlip()	0.7234	0.595	0.895	0.817	0.931
32	344	$1 \cdot 10^{-2}$	ResNet18	Default Pad(4)	0.5773	0.611	0.900	0.854	0.945
32	344	$2 \cdot 10^{-2}$	ResNet18	Default	0.5280	0.589	0.895	0.862	0.946
32	344	$5 \cdot 10^{-2}$	ResNet18	Default	0.7426	0.492	0.878	0.803	0.927
32	344	$1 \cdot 10^{-3}$	ResNet18	Default	1.3674	0.589	0.897	0.774	0.916
32	344	$1 \cdot 10^{-4}$	ResNet18	Default	2.4034	0.459	0.875	0.580	0.856
64	320	$1 \cdot 10^{-2}$	ResNet18	Default	0.6856	0.587	0.897	0.840	0.939
64	320	$2 \cdot 10^{-2}$	ResNet18	Default	0.5620	0.587	0.892	0.854	0.944
64	320	$5 \cdot 10^{-2}$	ResNet18	Default	0.5259	0.577	0.893	0.854	0.944
64	320	$1 \cdot 10^{-3}$	ResNet18	Default	1.6165	0.526	0.889	0.720	0.899
64	320	$1 \cdot 10^{-4}$	ResNet18	Default	2.7650	0.334	0.860	0.455	0.827

Table 5. Results obtained by testing different sets of parameters with ResNet18 as Context Path, 30 epochs – BiSeNet offline

Batch size	Crop size	Learning rate	Context Path	Data augmentation	Loss	mIoU Val	Precision Val	mIoU Train	Precision Train
32	344	$1 \cdot 10^{-2}$	ResNet50	Default	0.3644	0.680	0.911	0.894	0.958
32	344	$1 \cdot 10^{-2}$	ResNet50	Default RandomScale((0.5, 2.0))	0.3706	0.673	0.911	0.892	0.957
32	344	$1 \cdot 10^{-2}$	ResNet50	Default RandomScale((0.75, 2.0)) RandomResizedCrop(344, (1.0, 1.0))	0.2497	0.625	0.904	0.914	0.968
32	344	$1 \cdot 10^{-2}$	ResNet50	Default ColorJitter(0.5, 0.5, 0.5, 0) RandomResizedCrop(344, (0.75, 2.0))	0.3258	0.652	0.905	0.896	0.961
32	344	$2 \cdot 10^{-2}$	ResNet50	Default	0.3594	0.655	0.907	0.894	0.957
32	344	$5 \cdot 10^{-2}$	ResNet50	Default	0.5991	0.549	0.881	0.828	0.936
32	344	$1 \cdot 10^{-3}$	ResNet50	Default	1.0174	0.660	0.910	0.833	0.937
32	344	$1 \cdot 10^{-4}$	ResNet50	Default	2.1278	0.569	0.893	0.685	0.888
16	480	$0.5 \cdot 10^{-2}$	ResNet50	Default	0.3786	0.705	0.915	0.898	0.959
16	480	$1 \cdot 10^{-2}$	ResNet50	Default	0.3662	0.690	0.914	0.898	0.959
16	480	$2 \cdot 10^{-2}$	ResNet50	Default	0.3990	0.673	0.902	0.886	0.955
16	480	$5 \cdot 10^{-2}$	ResNet50	Default	0.9810	0.440	0.863	0.741	0.907
16	480	$1 \cdot 10^{-3}$	ResNet50	Default	0.7453	0.683	0.913	0.861	0.947
16	480	$1 \cdot 10^{-4}$	ResNet50	Default	1.8854	0.625	0.901	0.724	0.900

Table 6. Results obtained by testing different sets of parameters with ResNet50 as Context Path, 30 epochs – BiSeNet offline

Batch size	Crop size	Learning rate	Context Path	Data augmentation	Loss	mIoU Val	Precision Val	mIoU Train	Precision Train
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default	0.2909	0.688	0.914	0.906	0.962
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default ColorJitter(0.5, 0.5, 0.5, 0)	0.3274	0.700	0.909	0.895	0.958
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default RandomRotation(15)	0.3310	0.721	0.919	0.891	0.958
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default ColorJitter(0.5, 0.5, 0.5, 0) RandomRotation(15)	0.3738	0.711	0.915	0.880	0.954
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default ColorJitter(0.3, 0.3, 0.3, 0.1) RandomRotation(15)	0.3713	0.716	0.909	0.879	0.954
16	344	$0.5 \cdot 10^{-2}$	ResNet101	Default RandomResizedCrop(344, (1.0, 1.0)) RandomScale((0.75, 2.0))	0.2004	0.675	0.911	0.925	0.971
16	344	$1 \cdot 10^{-2}$	ResNet101	Default	0.3013	0.689	0.906	0.902	0.961
16	344	$2 \cdot 10^{-2}$	ResNet101	Default	0.3247	0.660	0.905	0.896	0.959
16	344	$5 \cdot 10^{-2}$	ResNet101	Default	0.8090	0.493	0.867	0.765	0.916
16	344	$1 \cdot 10^{-3}$	ResNet101	Default	0.4107	0.679	0.913	0.880	0.954
16	344	$1 \cdot 10^{-4}$	ResNet101	Default	1.5079	0.647	0.905	0.750	0.910

Table 7. Results obtained by testing different sets of parameters with ResNet101 as Context Path, 30 epochs – BiSeNet offline

Appendix B.

Figure 5. Qualitative results on the 15-5 (top) and 15-1 (bottom) settings of Pascal-VOC 2012 dataset using different incremental methods. From left to right: image, FT, LwF, ILT, MiB and ground truth. Best viewed in color.

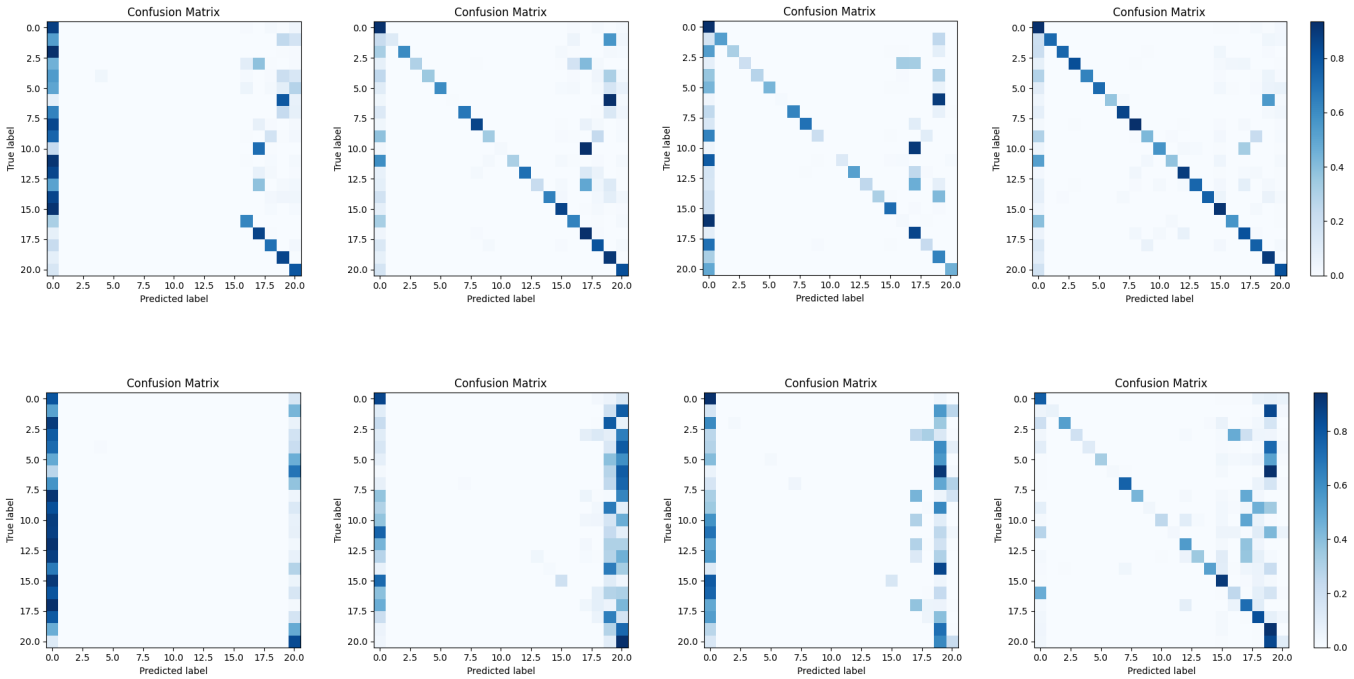

Figure 6. Confusion matrices on the 15-5 (top) and 15-1 (bottom) settings of Pascal-VOC 2012 dataset using different incremental methods. From left to right: FT, LwF, ILT and MiB.

		15-5				**15-1**			
class	Joint	FT	LwF	ILT	MiB	FT	LwF	ILT	MiB
background	97.6	81.4	91.3	92.6	91.3	72.1	74.9	86.7	76.9
aeroplane	83.6	0.0	12.1	54.2	71.2	0.0	0.0	0.1	6.3
bike	41.2	0.0	37.1	24.4	37.4	0.0	0.0	1.6	31.9
bird	86.8	0.0	29.6	20.9	75.9	0.1	0.1	0.0	18.6
boat	65.3	3.8	33.5	26.7	56.6	0.9	0.0	0.0	11.1
bottle	77.1	0.3	56.7	45.1	65.0	0.0	0.2	1.9	32.6
bus	92.6	0.0	0.5	1.4	36.6	0.0	0.0	0.0	1.7
car	85.3	0.0	66.4	65.3	77.6	0.0	0.8	3.9	70.9
cat	90.2	0.0	82.2	72.6	83.3	0.0	0.0	0.3	43.8
chair	36.3	0.0	29.8	20.5	33.3	0.0	0.0	0.0	3.1
cow	82.9	0.0	1.5	0.0	55.2	0.0	0.0	0.0	25.6
diningtable	46.0	0.0	31.7	13.2	35.6	0.0	0.0	0.0	2.3
dog	85.7	0.0	69.5	54.6	79.4	0.0	0.1	0.2	47.5
horse	82.1	0.0	22.4	26.0	70.1	0.0	3.1	0.0	34.9
motorbike	82.8	0.0	62.8	32.5	71.9	0.0	1.4	0.4	51.9
person	84.3	0.0	80.9	73.5	79.7	0.0	18.1	15.4	75.1
plant	56.3	17.5	19.6	1.1	21.9	0.0	0.0	0.0	7.4
sheep	80.4	23.3	20.5	18.2	39.7	0.0	2.3	9.5	11.8
sofa	46.6	26.8	25.4	18.8	27.9	0.0	1.7	3.3	11.1
train	83.5	32.1	28.7	22.3	43.6	0.0	3.2	7.8	8.9
tv/monitor	68.0	9.1	18.3	33.4	15.2	2.5	2.4	7.0	0.8
average	72.9	5.6	36.5	31.2	53.9	0.2	1.7	2.6	24.9

Table 8. mIoU per class in % on the Pascal-VOC 2012 dataset for different incremental class learning scenarios

Appendix C.

Figure 7. Qualitative results on the 15-5 setting of Pascal-VOC 2012 dataset using different incremental methods with weak supervision. From left to right: image, SEC-FT, SEC-LwF, SEC-ILT, SEC-MiB and ground truth. Best viewed in color.

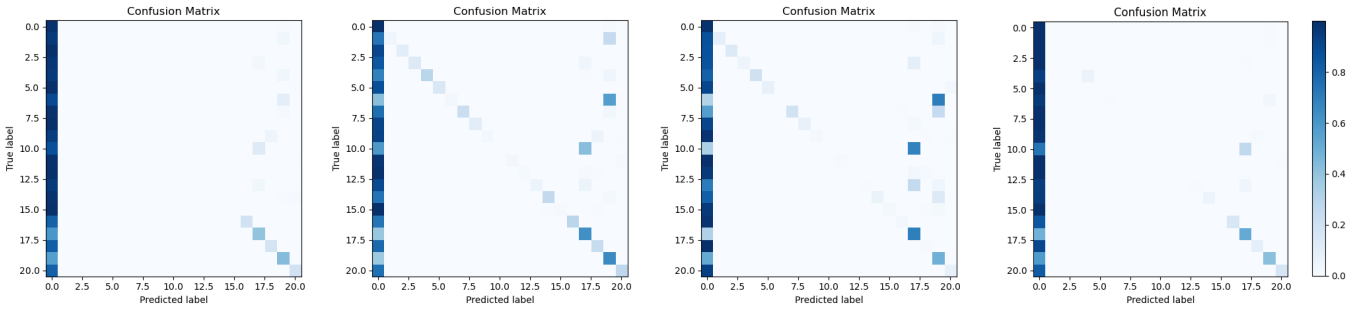

Figure 8. Confusion matrices on the 15-5 setting of Pascal-VOC 2012 dataset using different incremental methods with weak supervision. From left to right: SEC-FT, SEC-LwF, SEC-ILT and SEC-MiB.

	15-5			
class	SEC-FT	SEC-LwF	SEC-ILT	SEC-MiB
background	82.6	83.5	83.5	82.6
aeroplane	0.0	4.1	9.1	0.0
bike	0.0	8.9	9.4	0.1
bird	0.0	12.4	4.8	0.1
boat	0.0	18.7	14.9	5.6
bottle	0.0	13.5	6.9	0.4
bus	0.0	2.6	0.0	0.5
car	0.0	21.2	18.1	0.0
cat	0.0	10.1	7.1	0.0
chair	0.0	2.8	1.8	0.0
cow	0.0	0.0	0.0	0.0
diningtable	0.0	2.9	1.1	0.0
dog	0.0	1.4	0.3	0.1
horse	0.0	5.8	0.7	0.7
motorbike	0.0	23.9	6.2	4.8
person	0.0	1.1	2.3	0.0
plant	17.3	22.2	2.6	14.0
sheep	29.6	26.5	16.9	30.4
sofa	16.0	20.2	1.1	9.1
train	26.4	23.3	16.9	29.1
tv/monitor	15.8	20.2	5.9	15.1
average	5.2	12.1	6.3	5.5

Table 9. mIoU per class in % on the Pascal-VOC 2012 dataset for different weakly-supervised incremental class learning scenarios