Incremental Segmentation on Private Data without Catastrophic Forgetting

Tao HU
Peking University
taohu@pku.edu.cn

Abstract

Despite the success of image segmentation, convolutional neural networks are ill-equipped for incremental learning, i.e., adapting the original model trained on a set of classes to additionally segment new classes, without access of the original training data. They suffer from "catastrophic forgetting" — an abrupt degradation of performance on the old classes, when the training objective is adapted to the new classes. We present a method to address this issue, and learn image segmentation incrementally on private data whose annotations for the original classes in the new training set are unavailable. The key of our proposed solution is to balance the interplay between predictions on the new classes and distillation loss, it minimizes the discrepancy between responses for old classes on updated network via knowledge rehearsal. This incremental learning can be performed multiple times, for a new set of classes in each step, with a moderate drop in performance compared to the baseline network trained on the ensemble of data. We present image segmentation results on the PASCAL VOC 2012 and COCO datasets, on the ResNet and DenseNet architecture, along with a detailed empirical analysis of the approach.

1. Introduction

Lifelong or incremental learning[25, 1] is a key requirement for general artificial intelligent agents. Under this setting, the agent is required to acquire expertise on new tasks while maintaining its performance on previously learned tasks, ideally without the need to store large specialized models for each individual task. In the case of deep neural networks, the most common way of learning a new task is to fine-tune the network. However, as features relevant to the new task are learned through modification of the network weights, weights important for prior tasks might be altered, leading to deterioration in performance referred to as '*catastrophic forgetting*"[8]. In realistic application scene, older data annotations are sometimes unavailable due to the privacy policy, data rights, or deployed nature of the agent, therefore, direct fine-tuning is not a viable option for incremental learning in private data without catastrophic forgetting.

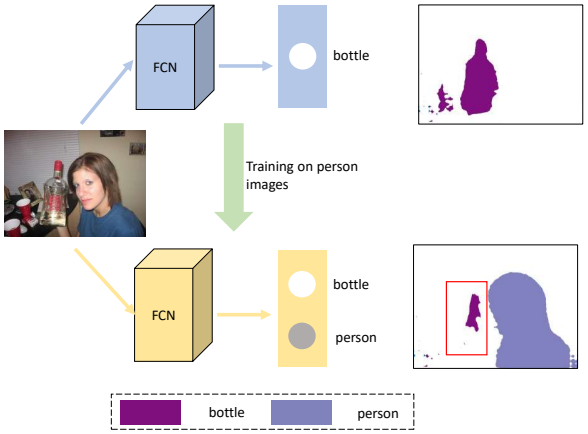

Figure 1: Catastrophic Forgetting. An image segmentation network is originally trained for one class, *bottle*(top). When the network is retrained with images of the new class *person*, it segments the person in the test image, but fails to segment the bottle(bottom).

For better understanding of this problem, consider the example in Figure 1. It illustrates a "catastrophic forgetting" phenomenon in image segmentation when incrementally adding a class, *person*. We can observe that an image segmentation network is originally trained for one class, *bottle*(top). When the network is retrained with images of the new class *person*, it segments the *person* in the test image, but fails to segment the *bottle*(bottom). In this paper, we propose a knowledge distillation method to alleviate this problem.

Without access of the training annotations for the old classes, we present a method not only adapting the old network to the new classes, but also ensuring the normal performance on the old classes. The hinge of our approach is to balance the interplay between predictions on the new classes and distillation loss, it minimizes the discrepancy

between responses for old classes on updated network via knowledge rehearsal. The overall approach is illustrated in Figure 2.

We use a frozen copy of the original segmentation network to compute the distillation loss. This loss is related to the concept of "knowledge distillation" proposed in[15]. Especially, we target the problem of image segmentation, which has the additional challenge of pixelwise classification unlike other attempts [20, 18, 19] limited to the image-level classification task. We demonstrate experimental results on the PASCAL VOC and COCO datasets using ResNet[14], DenseNet[17]-based Deeplab[3] as basic network architecture. Our results show that we can add new classes incrementally to an existing network without forgetting the original classes, when annotations for the original classes in the new training set are unavailable. Note that our framework is general and can be applied to any other FCN-based segmentation network.

2. Related Work

The problem of incremental learning has a long history in machine learning[33, 30]. One typical weakness is that the old training data must be kept for retraining, other work partially addresses these issues by learning classifiers without access to the ensemble of data[27]. Image segmentation is closely related with image classification, and can be seen as a pixelwise classification problem. The data annotations in image segmentation may not only contain one class, and these annotations can be unavailable due to the privacy policy, which are all very realistic problem. To solve this problem, our approach is aimed at learning the representation and classifiers jointly in an end-to-end pattern, without access of the old training data annotations.

Transfer Learning and Domain Adaptation. Incremental learning is closely related to transfer learning and domain adaptation methods. Transfer learning uses knowledge acquired from one task to help learn another(such as fine-tuning). Domain adaptation transfers the knowledge acquired for a task from a data distribution to other(but related) data. These paradigms, and in particular fine-tuning, a special case of transfer learning, are very popular in computer vision. CNNs learned for image classification[5, 32] are often used to train other vision tasks such as object detection[9, 10] and semantic segmentation[23, 3]. Recently, several works[4, 16, 29, 34] are proposed to utilize domain adaption in semantic segmentation to transfer the domain from synthesized dataset to real-life dataset, thus avoiding the tedious dataset labeling process.

Except transfer learning and domain adaption, an alternative to transfer knowledge from one network to another is distillation[15]. This is originally proposed to transfer knowledge between different neural networks — from a large network to a smaller one for efficient deployment. The method in [15] encouraged the large(old) and the small(new) networks to produce similar responses. It has found several applications in domain adaptation and model compression[35, 12]. Overall, transfer learning and domain adaptation methods require at least unlabeled data for both the tasks or domains and in its absence, the new network quickly forgets all the knowledge acquired in the source domain[8, 11]. In contrast, our approach addresses the challenging case where no training annotations are available for the original task(i.e., detecting objects belonging to the original classes), by building on the concept of knowledge distillation[15].

Learning without Forgetting. A few prior works and their variants, such as Learning without Forgetting(LwF)[20, 31, 25], iCaRL[26], Elastic Weight Consolidation(EWC)[18, 22], EWC Moment[19], PNN[28], PNN's extension PathNet[7] are aimed at training a network for multiple tasks sequentially.

LwF[20] use knowledge distillation for one of the classical vision tasks, image classification, formulated in a deep learning framework. When adding a new task, LwF preserves response of the network on older tasks by a distillation loss[15], where response targets are computed using data from the current task. As a result, LwF does not require the storage of older training data. However, LwF[20] is only evaluated in the simple task of image classification rather than more difficult tasks such as object detection, image segmentation.

iCaRL[26] addresses some of the drawbacks of LwF[20] with their incremental learning approach for image classification. They also use knowledge distillation and decouple the process into the classifier and the representation learning. Additionally, they rely on a subset of the original training data to preserve the performance on the old classes. In comparison, our approach is an end-to-end learning framework, where the representation and the classifier are learned jointly, and we do not use any of the original training samples to avoid catastrophic forgetting.

Another typical alternative to distillation is PNN[28], which grows the capacity of the network with new layers when new classes training, the downside to this method is the rapid increase in the number of new parameters to be learned.

EWC[18] is an intuitive approach that applies strong per-parameter regularization selectively so that the network will not "forget" the knowledge from old training data . It tries to minimize the change in weights that are important to previous tasks, specifically, through the use of a quadratic constraint that tries to ensure that they do not stray too far from their initial values. However, their evaluation is only limited on the easier task of image classification.

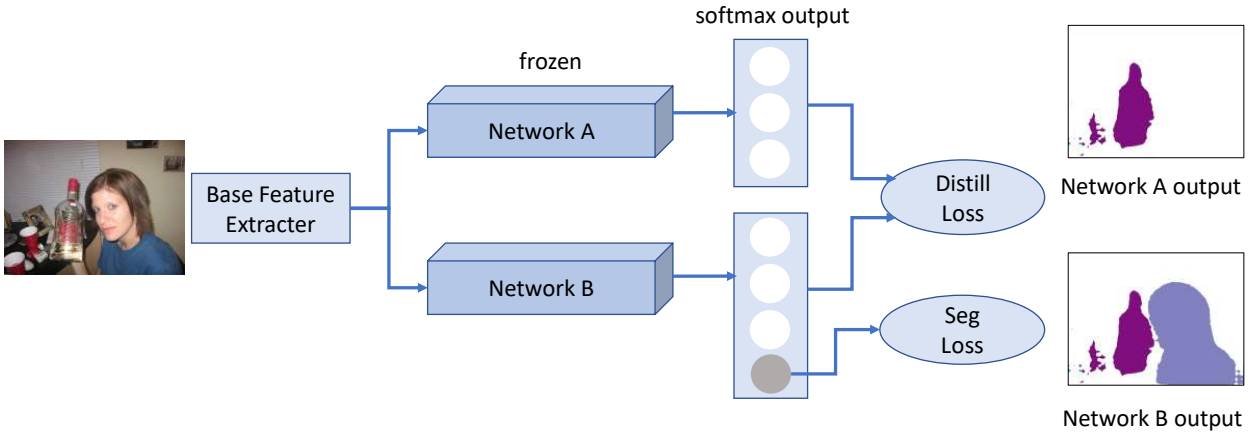

Figure 2: Our network Architecture. Network **A** is trained with old classes, Network **B** is trained with new classes. Our target is try to boost the performance both on the old classes and new classes. This figure is illustrated with 3 old classes and 1 new class. Best viewed in color.

Incremental-detector[31] is proposed to solve the "catastrophic forgetting" problem in object detection. It combines a detection loss(classification loss and bounding box regression loss) and knowledge distilling loss to realize an high-efficient incremental object detector. However, their application scene is only about object detection, our method mainly focuses on image segmentation.

In summary, none of the previous work addresses the problem of image segmentation incrementally, without utilizing previous training annotations.

3. Main idea

Our overall approach for incremental learning of a FCN model for image segmentation is illustrated in Figure 2. It contains a frozen copy of the original segmentation network(denoted by Network **A** in the figure). Network **B** in this figure is the adapted network for the new classes. It is obtained by increasing the number of outputs in the last layer of the original network, such that the new output layer includes the old as well as the new classes.

In order to avoid catastrophic forgetting, we constrain the learning process of the adapted network. We achieve this by incorporating a distillation loss, to preserve the performance on the old classes, as an additional term in the standard cross-entropy loss function. As we show in the experimental evaluation, the distillation loss is critical in preserving the performance on the old classes.

In the remainder of this section, we provide details of the image segmentation network, the loss function and the learning algorithm.

3.1. Image Segmentation Network

We use a popular framework for image segmentation–deeplab[3]. We adopt ResNet50 as our backbone in deeplab for demonstration. The network processes the whole image through a sequence of residual blocks. Dilated convolution with stride=2,4 is used in res3 and res4. A softmax layer for pixelwise classification(PASCAL VOC or COCO classes, for example, along with the background class). The loss function to train the image segmentation network is given by:

$$\mathcal{L}_{seg} = -\frac{1}{m \times n} \sum_{i=0}^{m} \sum_{j=0}^{n} y_{i,j}^{c} log x_{i,j}^{c} \tag{1}$$

where $c \in [0, |C_A|]$, $|C_A|$ is the total class number(including background). $y_{i,j}^{c}$ means the ground truth label in the position of (i,j), if the ground truth label is c ,$y_{i,j}^{c}$ =1, otherwise $y_{i,j}^{c}$ =0. $logx_{i,j}$ denotes the softmax output of position(i,j) in the c-th channel.

3.2. Dual-Network Learning

First, we train a Deeplab to segment the original set of classes C_A. We refer to this network as $\mathbf{A}(C_A)$. The goal now is to add a new set of classes C_B to this. We make two copies of $\mathbf{A}(C_A)$: one that is frozen to recognize classes C_A through distillation loss, and the second $\mathbf{B}(C_B)$ that is intended to segment the new classes C_B, which were not present or at least not annotated in the source images. We just change the last layer node number for new classes.The new weight is initialized randomly in the same way as the corresponding layers in Deeplab. Our goal is to train $\mathbf{B}(C_B)$

to recognize classes $C_A \cup C_B$ using only new data and annotations for C_B.

The distillation loss represents the idea of "keeping all the answers of the network the same or as close as possible". If we train $\mathbf{B}(C_B)$ without distillation, the performance on the old classes will degrade quickly. This is a manifestation of catastrophic forgetting. We compute the distillation loss by applying the frozen copy of $\mathbf{A}(C_A)$ to any new image. Even if no non-background class is segmented by $\mathbf{A}(C_A)$, the unnormalized logits(softmax output) carry enough information to "distill" the knowledge of the old classes from $\mathbf{A}_{C_A}$ to $\mathbf{B}(C_B)$. This process is illustrated in Figure 2. The distillation loss is written as:

$$\mathcal{L}_{dist} = -\frac{T^2}{mnc} \sum_{c=0}^{|C_B|} \sum_{i=0}^{m} \sum_{j=0}^{n} P_{i,j}^{c} log Q_{i,j}^{c} \tag{2}$$

$$Q_{i,j}^{c} = \frac{exp(z_{i,j}^{c}/T)}{\sum_{k=0}^{|C_B|} exp(z_{i,j}^{k}/T)} \tag{3}$$

where $|C_B|$ is the new class number(including background class). $P_{i,j}^{c}$ is the softmax output from Network $\mathbf{B}$, $Q_{i,j}^{c}$ is the softmax output from Network $\mathbf{A}$. T is temperature($T \geq 1$), with higher temperature, we can get "softer" probability distribution over classes. Since the softmax scores are now softened, the hidden information from the incorrect classes may become more evident to be distilled. $z_{i,j}^{c}$ is pre-softmax logits in the position of (i,j) of c-th channel .

The overall class $\mathcal{L}$ to train the model incrementally is a weighted sum of the distillation loss (2) and the standard segmentation cross-entropy loss (1) that is applied only to new classes C_B, where ground truth semantic label is available. In essence,

$$\mathcal{L} = \mathcal{L}_{seg} + \lambda \mathcal{L}_{dist} \tag{4}$$

where the hyperparameter λ balances the two losses. The choice of λ will be discussed in the later experiment.

The interplay between the two network $\mathbf{A}(C_A)$ and $\mathbf{B}(C_B)$ provides the necessary supervision that prevents the catastrophic forgetting in the absence of original training data used by $\mathbf{A}(C_A)$. After the training of $\mathbf{B}(C_B)$ is completed, we can add more classes by freezing the newly trained network and using it for distillation. We can thus add new classes sequentially. Since $\mathbf{B}(C_B)$ is structurally identical to $\mathbf{A}(C_A \cup C_B)$, the extension can be repeated to add more classes.

4. Experiment

In the following, we will mainly introduce dataset, implementation details, and different experimental results including the one-time addition of multiple classes, the multi-time addition of multiple classes.

4.1. Datasets and evaluation

We evaluate our method on the PASCAL VOC dataset and Microsoft COCO challenge dataset[21]. The PASCAL VOC 2012 segmentation benchmark[6] involves 20 foreground object classes and one background class. The original dataset contains 1,464 (train), 1,449 (val), and 1,456 (test) pixel-level labeled images for training, validation, and testing, respectively. The dataset is augmented by the extra annotations provided by [13], resulting in 10,582 (train-aug) training images. COCO on the other hand has 80K images in the training set and 40K images in the validation set for 80 object classes(which includes all the classes from VOC). COCO dataset is proposed used for instance segmentation, while it is utilized for image segmentation in our experiment. The performance is measured in terms of pixel intersection-over-union(IOU) averaged across all classes. Evaluations of the VOC and COCO are both done on the val split, while for COCO, we use the first 5000 images from the validation set.

4.2. Implementation details

We use SGD with Nesterov momentum[2] to train the network in all the experiments. We set the learning rate to 2.5e-4, decay to 5e-4, and momentum to 0.9. In the second stage of training, i.e., learning the extended network with new classes, we used a learning rate of 5e-4. The $\mathbf{A}(C_A)$ network is trained for 40K iterations on PASCAL VOC and for 100K iterations on COCO. The $\mathbf{B}(C_B)$network is trained the same number of iterations as $\mathbf{A}(C_A)$ when many classes are added at once.

We use pytorch[24] to develop our incremental learning framework. Each experiment begins with choosing a subset of classes to form the set C_A. Then, a network is learned only on the subset of the training set composed of all the images containing at least one object from C_A. Annotations for other classes in these images are ignored. With the new classes chosen to form the set C_B, we learn the network as described in Figure 2 with the subset of the training set containing at least one object from C_B. As in the previous case, annotations of all the other classes, including those of the original classes C_A, are ignored. Noticeably, the image number of $\mathbf{B}$ network always equal the total number of VOC. Even one image is all comprised of old classes, we still adopt this image for knowledge distillation from old classes.

In our later experiment we annotate "$|C_A|+|C_B|$" as the incremental segmentation process under old class C_A,new

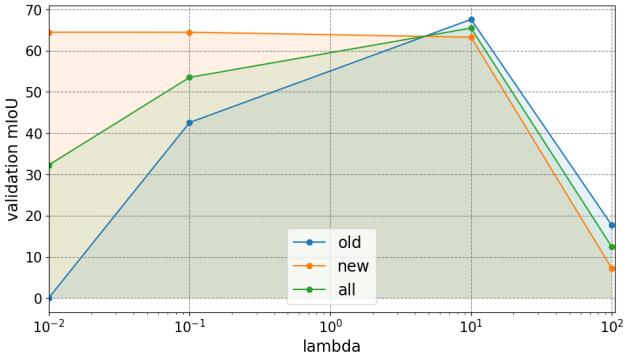

Figure 3: λ Setting Ablation Study

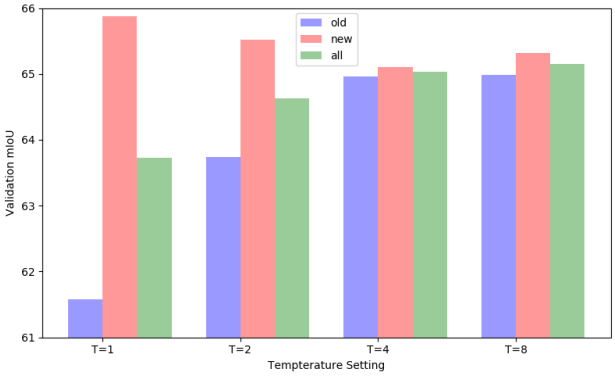

Figure 4: Temperature(T) Setting Ablation Study

class C_B. For example, "10+10" means we take 10 classes in alphabetical order from the VOC dataset as C_A, and the remaining one as the only new class C_B. "15+1+1+1+1+1" means that we take a network learned for 15 classes of VOC, train it for the 16th on the subset containing only this class, and then use the 16-class network as the frozen copy to then learn the 17th class. This is then continued until the 20th class.

4.3. One-time Addition of multiple classes

"10+10" addition for PASCAL VOC: In the first experiment we take 10 classes in alphabetical order from the VOC dataset as C_A, and the remaining one as the only new class C_B. We then train the **A**(1-10) network on the VOC trainaug subset containing any of the 10 classes, and the **B** (11-20) network is trained on the trainaug subset containing the new classes. A summary of the evaluation of these networks on the VOC val set is shown in Table 1, with the full results in Table 8.

When the network **B**(11-20) is trained without the distillation loss("w/o distillation" in the table), it can learn the 11-20 classes, but the performance decreases significantly on the old classes. As seen in Table 8, the mIoU on classes like "bird","cat" drops by over 40%. The same training procedure with distillation loss largely alleviates this catastrophic forgetting. Without distillation, the new network has 40% mIoU drop on the old classes compared to 66.7%. With distillation the performance is similar to that of the

Table 1: **PASCAL VOC validation mIoU.** Experiments demonstrates the addition of 10 classes to a pretrained network under various settings. Classes1-10 are the old classes, 11-20 are the new one. "share123" means share res1,res2,res3.

Method	old(10)	new(10)	all(20)
multiple-network	66.7	64.8	65.75
w/o distillation	27.33	61.20	44.27
T=8,share res1	63.9	64.7	64.3
T=8,share res12	64.99	64.99	**64.99**
T=8,share res123	63.84	62.51	63.18
T=8,share res1234	61.05	58.55	59.8
EWC[18]	66.7	40.4	53.5
A(1-20)	68.29	66.49	67.39

Table 2: **PASCAL VOC validation mIoU.** Experiments show one-time addition to a pretrained network. Classes1-15 are the old classes, 16-20 are the new one. "share123" means share res1,res2,res3.

Loss	Setting	old(15)	new(5)	all(20)
	multiple-network	68.24	56.18	65.23
MSE		25.59	57.4	33.54
	share res1	31.02	59.74	38.20
	share res12	32.72	60.19	39.59
	share res12,$\lambda = 10$	67.05	60.81	**65.49**
	share res123	45.59	60.92	46.36
	share res1234	22.43	54.13	30.36
KL	T=1	63.17	60.13	62.41
	T=8	67.55	59.82	65.62
	T=8,share res12	67.33	59.85	65.46
	T=8,share res12,$\lambda = 10$	68.88	58.30	**66.24**
	A(1-20)	68.9	62.85	67.39

Table 3: **PASCAL VOC and COCO validation mIoU.** Experiments show one-time addition to a pretrained network. Classes1-10 are the old classes, 11-20 are the new one.

Dataset	Method	old(10)	new(10)	all(20)
VOC	A(1-10)	66.7	–	–
	+B(11-20)	67.59	63.27	**65.48**
	A(1-20)	68.29	66.49	67.39
COCO	A(1-10)	53.77	–	–
	B(11-20)	51.64	50.24	**50.94**
	A(1-20)	55.33	55.33	55.33

Table 4: **COCO validation mIoU.** Experiments show one-time addition to a pretrained network. Classes1-40 are the old classes, 41-80 are the new one.

Method	old(40)	new(40)	all(80)
multiple-network	56.08	38.61	47.34
T=8	33.38	39.70	36.54
T=8,share res12	34.91	39.53	37.22
T=8, share res12, $\lambda = 10$	54.9	38.16	**46.53**
A(1-80)	53.69	39.59	46.64

old network **A**(1-10), but is lower for certain classes, e.g., "bus","chair".

We mainly compare the results in different distillation setting, feature sharing , and the balance between the cross-entropy loss and distillation loss. From Figure 4, we can find that large temperature such as T=8 is more helpful for the knowledge distillation process. For the feature sharing, we find feature sharing in res1, res2 will most enhance the overall segmentation result.

We also search the weight to balance the cross-entropy loss and distillation loss in Figure 3. We conclude that $\lambda = 10$ is the best choice for balancing those losses.

"15+5" addition for PASCAL VOC: Same with the previous setting, we take 15 classes in alphabetical order from the VOC dataset as C_A, and the remaining one as the only new class C_B. A summary of the evaluation of these networks on the VOC val set is shown in Table 2, with the full results in Table 9.

Under this setting, we mainly explore the different knowledge distillation approaches, an intuitive method is Mean Square Error(MSE), corresponding experiments are conducted to determine the best performance. Noticeably, the performance is unsatisfactory if we don't share any features between Network **A** and Network **B**. We can conclude that low-level feature sharing will stabilize the performance of knowledge distillation. The best result is 65.49 compared with our oracle 67.39.

On the other hand, we also conduct some experiments using KL distance to realize knowledge distillation. our best result with distillation temperature T=8, res12 feature sharing reaches 66.24, which is only 1.15 lower than our upper bound result in oracle. This also proves that KL distance is more suitable than MSE in knowledge distillation. In our later experiment, we will also use this setting to validate our algorithm.

"10+10" addition for PASCAL VOC and COCO: To validate our method for different data distribution, we choose PASCAL VOC and COCO as our default dataset. For COCO, we choose 20 classes that overlap with PASCAL VOC. therefore, the 20 classes in PASCAL VOC and COCO are overall the same, such as person, car, bike,... etc.

Table 5: **PASCAL VOC validation mIoU.** Experiments show one-time addition to a pretrained network. The backbone network is DenseNet121[17].

Network	Method	old	new	all
DenseNet121[17]	**A**(1-15)	68.73	–	–
	+**B**(16-20)	69.48	56.13	66.14
	A(1-20)	68.51	61.94	66.87
	A(1-10)	65.97	–	–
	+**B**(11-20)	67.17	62.13	64.65
	A(1-20)	67.42	66.32	66.87
DenseNet201[17]	**A**(1-15)	69.96	–	–
	+**B**(16-20)	70.56	60.03	67.93
	A(1-20)	69.81	62.66	68.02
	A(1-10)	67.48	–	–
	+**B**(11-20)	68.78	61.98	65.38
	A(1-20)	69.21	66.83	68.02
ResNet50[14]	**A**(1-15)	68.24	–	–
	+**B**(16-20)	68.88	58.3	66.24
	A(1-20)	68.9	62.85	67.39
	A(1-10)	66.7	–	–
	+**B**(11-20)	67.59	63.27	65.48
	A(1-20)	68.29	66.49	67.39
ResNet101[14]	**A**(1-15)	70.16	–	–
	+**B**(16-20)	70.67	61.31	68.33
	A(1-20)	70.56	68.24	69.40
	A(1-10)	67.57	–	–
	+**B**(11-20)	68.68	64.66	66.67
	A(1-20)	71.19	64.04	69.40

Table 6: **PASCAL VOC validation mIoU.** Experiments show multi-time addition to a pretrained network. The old classes are 1-15, the new classes are 16-20, these 5 classes are added sequentially.

Method	old	new	all
A(15)	68.24	–	–
+**B**(16)	69.56	50.41	68.35
A(1-16)	–	–	67.85
+**B**(16)(17)	67.78	69.86	67.91
A(1-17)	–	–	68.16
+**B**(16)(17)(18)	67.34	40.21	65.83
A(1-18)	–	–	67.11
+**B**(16)(17)(18)(19)	65.43	76.94	66.04
A(1-19)	–	–	67.82
+**B**(16)(17)(18)(19)(20)	65.74	64.99	65.70
A(1-20)	68.29	66.49	67.39
+**A**(16)+...+**A**(20)	68.24	52.36	64.27
A(1-20)	68.29	66.49	67.39

We take 10 classes in alphabetical order from the VOC

Table 7: **PASCAL VOC validation mIoU.** Experiments show multi-time addition to a pretrained network. The old classes are 1-10, the others are new classes. 11-15, 16-20 are added sequentially.

Method	old	new	all
A(10)	66.7	64.8	65.75
+**A**(11-15)	66.97	68.75	67.56
A(1-15)	–	–	68.24
+**A**(16-20)	67.65	59.08	65.51
A(1-20)	–	–	67.39
+**A**(11-20)	67.69	63.27	65.48
A(1-20)	–	–	67.39

dataset as C_A, and the remaining one as the only new class C_B. Table 3 shows a summary of the evaluation of these networks on the VOC val set.

In PASCAL VOC, the performance is 65.48 which is 1.91 lower than oracle. In COCO, the performance is 50.94 which is 4.39 less than oracle. The result in PASCAL VOC is more satisfactory than COCO due to the complexity of COCO dataset(more diverse object scale, more image number).

"40+40" addition for COCO: To further evaluate our architecture in complex dataset, we take 40 classes in alphabetical order from the COCO dataset as C_A, and the remaining one as the only new class C_B in Table 4. After feature sharing, the performance is 0.68 higher than our distillation baseline. Noticeably, the balance between cross-entropy loss and distillation loss plays an important role to boost the overall performance. Our result is only 0.11 lower than oracle that trains all 80 classes one time.

4.4. Multi-time addition of multiple classes

"15+1+1+1+1+1" incremental segmentation: In order to evaluate incremental learning of classes added sequentially, we update the frozen copy of the network with the one learned with the new class, and then repeat the process with another new class. For example, we take a network learned for 15 classes of VOC, train it for the 16th on the subset containing only this class, and then use the 16-class network as the frozen copy to then learn the 17th class. This is then continued until the 20th class. We denote this incremental extension as **B**(16)(17)(18)(19)(20). A summary of the evaluation of these networks on the VOC val set is shown in Table 6. Table 10 shows full intermediate evaluations after adding each class.

As we incrementally add 1 new class, we also compare with corresponding upper bound result. For example, when we complete +**B**(16)(17), we will compare it with **A**(1-17) to validate our knowledge distillation performance. From Table 6, the incremental segmentation always keeps a moderate drop compared with their corresponding upper bound in setting: "15+1","16+1","17+1","18+1","19+1","15+1+1+1+1+1".

"10+5+5" incremental segmentation: As before, we conduct experiment that sequentially adds 5 classes on the old 10 classes. A summary of the evaluation of these networks on VOC val set is illustrated in Table 7, with the full results in Table 11. We can find all of them only keep a moderate drop compared with their corresponding upper bounds in the setting: "10+5","10+5+5","10+10". This extensive experiment proves that our algorithm performs favorably in the extreme case of incremental settings.

4.5. Diverse Network Structure

Previous experiments are overall conducted on the ResNet50 backbone. To evaluate performance on other network structure, we demonstrate the result on Resnet50[14], Resnet101, DenseNet121[17], DenseNet201 in Table 5. The incremental segmentation setting includes "10+10", "15+5".

DenseNet Backbone Performance: Similar to ResNet, DenseNet is also comprised of 4 blocks. we share features in the lower 2 blocks. We can see that in "10+10" setting, the drop between our result and the upper bound is larger than the setting of "15+5". Because "10+10" is a more difficult task than "15+5" task, because it sequentially adds more new classes. Overall, the knowledge distillation result is satisfactory and the drop is within tolerable limits.

ResNet Backbone Performance: We conduct experiments in setting of "15+5", "10+10". In the setting "15+5", The gap between our result and the upper bound is only about 1. In a more difficult setting "10+10", The drop is nearly 2 in ResNet50 architecture, while reaches 3 in ResNet101.

4.6. Extra Analysis

Compare with multiple network. For the task of incremental learning, a direct solution is training multiple network for all data(including old data, new data). This method exists an obvious disadvantage — large capacity occupation. For example, if you incrementally add new data n times, the capacity occupation is also n times. On the contrary, the knowledge distillation method can realize capacity multiplexing, which greatly reduces the capacity demand. No matter how many times incremental data is added, the capacity demand still keeps the same.

We demonstrate the multiple network result in "10+10"(Table 1), "15+5"(Table 2), and "15+1+1+1+1+1"(Table 6). In the incremental setting

Table 8: **PASCAL VOC validation mIoU** under different settings when 10 classes are added at once.

Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mIoU
A(10)	81.55	36.10	79.45	54.8	74.93	85.38	78.17	83.46	28.46	64.89	–	–	–	–	–	–	–	–	–	–	66.7
+**B**(11-20),w/o distillation	35.13	14.25	35.4	15.5	23.23	46.4	36.43	40.30	3.3	23.36	48.34	75.24	54.32	68.33	74.32	47.21	69.99	36.40	76.54	61.32	44.27
+**B**(11-20),T=1	76.13	33.55	75.14	50.5	66.06	80.4	74.17	79.66	24.58	55.76	51.44	78.12	71.39	72.63	79.51	53.71	71.94	39.70	78.14	62.24	63.74
+**B**(11-20),T=8	81.48	35.15	79.1	53.85	71.24	82.48	77.74	82.86	25.00	60.88	48.84	77.84	69.46	73.54	78.60	52.3	72.29	37.94	76.76	65.65	65.15
+**B**(11-20),T=8, share res12,$\lambda = 10$	82.01	37.20	81.0	55.64	75.26	85.10	78.67	84.91	27.23	68.89	51.15	78.98	69.8	69.17	76.44	49.34	69.96	37.04	70.89	60.95	65.48
oracle	81.78	37.65	80.31	57.07	75.54	86.73	78.29	84.71	29.27	71.56	45.91	78.65	72.24	75.22	78.58	53.21	76.82	42.95	77.09	64.22	**67.39**

Table 9: **PASCAL VOC validation mIoU** under different settings when 5 classes are added at once.

Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mIoU
A(15)	82.79	37.46	81.16	55.24	75.97	85.79	79.11	83.86	28.19	65.3	47.11	75.64	71.69	75.33	78.95	–	–	–	–	–	68.24
+**B**(16-20),mse,share res1	48.26	24.53	0	0	0	67.92	63.66	73.84	1.72	0.37	0	57.47	0	56.32	70.99	51.57	68.22	37.63	78.29	63.8	38.23
+**B**(16-20),T=1	76.69	33.22	73.26	54.81	64.77	81.95	76.18	82.25	23.37	56.46	45.44	71.67	61.51	71.30	74.72	51.78	66.85	37.46	79.14	65.4	62.41
+**B**(16-20),T=8	80.53	35.60	79.82	56.59	72.80	85.88	79.08	84.84	27.56	65.30	48.81	77.66	67.46	73.77	77.59	50.06	68.80	40.75	77.24	62.3	65.62
+**B**(16-20),T=8,share res12, $\lambda = 10$	83.55	38.11	80.94	56.73	75.81	86.84	80.28	84.57	28.16	68.68	45.59	78.22	71.22	75.87	78.81	49.27	68	37.09	74.53	62.62	66.24
oracle	81.78	37.65	80.31	57.07	75.54	86.73	78.29	84.71	29.27	71.56	45.91	78.65	72.24	75.22	78.58	53.21	76.82	42.95	77.09	64.22	67.39

Table 10: **PASCAL VOC validation mIoU** under different settings when 5 classes are added sequentially.

Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mIoU
A(1-15)	82.79	37.46	81.16	55.24	75.97	85.79	79.11	83.86	28.19	65.3	47.11	75.64	71.69	75.33	78.95	–	–	–	–	–	68.24
+**B**(16)	83.33	37.38	81.83	56.46	76.15	88.3	80.17	85.00	30.75	71.86	45.59	78.80	72.52	75.61	79.4	50.41	–	–	–	–	68.34
+**B**(16)(17)	82.64	37.33	81.52	57.18	75.01	88.49	79.82	84.26	31.50	70.15	42.52	77.24	70.31	75.7	79.09	51.99	69.86	–	–	–	67.91
+**B**(16)(17)(18)	81.89	36.72	81.20	57.78	74.25	87.66	80.03	83.76	29.02	69.29	44.19	75.98	70.72	75.32	78.34	49.12	69.47	40.21	–	–	65.83
+**B**(16)(17)(18)(19)	82.31	36.21	80.40	57.2	72.97	85.94	80.46	82.91	28.36	68.69	44.89	75.56	68.08	74.56	77.87	49.68	71.43	40.36	76.95	–	66.04
+**B**(16)(17)(18)(19)(20)	81.85	35.7	79.25	55.46	71.95	84.9	80.53	82.13	27.57	68.45	42.93	76.43	67.22	74.78	77.63	50.92	70.97	41.23	79.14	64.99	65.7
oracle	81.78	37.65	80.31	57.07	75.54	86.73	78.29	84.71	29.27	71.56	45.91	78.65	72.24	75.22	78.58	53.21	76.82	42.95	77.09	64.22	67.39

Table 11: **PASCAL VOC validation mIoU** under different settings when 5 classes are added twice.

Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mIoU
A(1-10)	81.55	36.10	79.45	54.8	74.93	85.38	78.17	83.46	28.46	64.89	–	–	–	–	–	–	–	–	–	–	66.72
+**B**(11-15)	81.67	36.41	78.26	56.08	74.76	86.95	78.7	81.80	28.61	66.42	51.17	77.25	69.24	69.38	76.71	–	–	–	–	–	67.56
+**B**(15-20)	84.32	36.95	80.81	58.13	74.53	86.18	79.97	83.97	28.56	60.86	47.11	75.31	69.18	71.12	77.79	51.11	71.85	38.12	73.14	61.18	65.51
+**B**(11-20)	82.02	37.20	81.0	55.64	75.26	85.10	78.67	84.91	27.23	68.88	51.15	78.98	69.80	69.17	76.44	49.34	69.96	37.04	70.89	60.95	65.48
oracle	81.78	37.65	80.31	57.07	75.54	86.73	78.29	84.71	29.27	71.56	45.91	78.65	72.24	75.22	78.58	53.21	76.82	42.95	77.09	64.22	67.39

of "10+10", the result of our best knowledge distillation method is nearly the same with the result of multiple network. In the setting of "15+5", our result is even little better than multiple network training. Noticeably, in the extreme sequential adding experiment of "15+1+1+1+1+1", our distillation method performs favorably than the multiple network(65.7 vs 64.27). our method not only owns better result than multiple networks, but also has less capacity demand.

Comparing with other methods. We mainly compare our method with EWC[18], which is proposed in the task of image classification. We reproduced the result of EWC, then apply it in our incremental segmentation task. Our method of knowledge distillation is nearly 12 higher than EWC[18]. This proves our method's effectiveness in more complex task of image segmentation.

Is feature sharing important? In PASCAL VOC, The result of our knowledge distillation method is nearly the same with non-feature sharing method. However, in the complex dataset such as COCO, the knowledge distillation method is much better than non-feature sharing method. This indicates low-level feature sharing can do a great favor for enhancing the basic feature extraction process when we learning new knowledge without forgetting the knowledge obtained from old classes.

5. Conclusion

In this paper, we have presented an approach for incremental learning of image segmentation for new classes, without access to the data annotations corresponding to the old classes. We address the problem of catastrophic forgetting in this context, with a loss function that optimizes the performance on the new classes, in addition to preserving the performance on the old classes. Our extensive experimental analysis demonstrates that our approach performs well, even in the extreme case of adding new classes one by one. Part of our future work is applying the method of incremental image segmentation problem into the area of instance segmentation, and exploring how to deal with incremental learning with both image and labels missed.

References

[1] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. *CoRR, abs/1611.06194*, 2, 2016.

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] 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*, 40(4):834–848, 2018.

[4] Y. Chen, W. Li, and L. Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7892–7901, 2018.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[7] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

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

[9] R. Girshick. Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[10] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[11] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[12] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2827–2836. IEEE, 2016.

[13] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision*, pages 991–998. IEEE, 2011.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems*, 2014.

[16] J. Hoffman, D. Wang, F. Yu, and T. Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[19] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, pages 4655–4665, 2017.

[20] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[21] 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 *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[22] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *International Conference on Pattern Recognition*, 2018.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[25] A. Rannen Ep Triki, R. Aljundi, M. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. In *International Conference on Computer Vision*, pages 1320–1328, 2017.

[26] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[27] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool. Incremental learning of ncm forests for large-scale image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3654–3661, 2014.

[28] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *Advances in Neural Information Processing Systems*, 2016.

[29] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *IEEE Conference on Computer Vision and Pattern Recognition*, 3, 2017.

[30] J. C. Schlimmer and D. Fisher. A case study of incremental concept induction. In *AAAI*, volume 86, pages 496–501, 1986.

[31] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. *International Conference on Computer Vision*, 2017.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

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

[34] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[35] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *International Conference on Computer Vision*, pages 4068–4076. IEEE, 2015.