SORBONNE UNIVERSITÉ

MASTER IMA - IMAGE PROCESSING

Context incremental learning for semantic segmentation of indoor scene with mobile robots

Author:
Jean-Paul LAM

Supervisor:
Dr. Daniela PAMPLONA

Internship from March to August 2021 (6 months) at Unité d'Informatique et d'Ingénierie des Systèmes (U2IS) from the École Nationale Supérieure de Techniques Avancées (ENSTA-Paris)

Abstract

Incremental learning aims to update a model without all the data used from the start, reducing the constraint of memory, time, and money and avoiding privacy issues. Our goal in this thesis is to apply and see where incremental deep learning techniques stand today in semantic segmentation for robotics.

Unlike classification and object segmentation, semantic segmentation hasn't been extensively studied in the incremental case. Also, to the best of our knowledge, no similar work has been published in robotics settings.

We first explain encoder-decoder, which is the leading deep learning architecture for semantic segmentation, before reviewing the latter's state-of-the-art in the case of incremental learning. We try different experiments and settings by applying reviewed methods on a robotic dataset, providing a baseline for future works.

While we show similar results with similar settings between robotics and "natural" indoor/outdoor environment, the task is generally more difficult in a realistic scenario where a model learns by roaming different rooms sequentially. We called this setting context incremental learning. Notably, we record drops performance up to 50%. Several points can explain this behavior: from the sequential nature of the robotic dataset used to the difference in the setting and learning process. Context incremental learning delivers new specific problems.

Overall, we conclude that while a high margin exists between incremental and offline learning for semantic segmentation, more and more researches are done in this field notably in the past three months (As of today, 27th September 2021). Specific techniques need to be developed to tackle the incremental, robotic, and context incremental settings.

Contents

Chapter 1

Introduction

1.1 Segmentation

Segmentation is the task of partitioning an image into different parts/groups of pixels. It is a critical low-level task in numerous practical applications: in medical imaging to create 3d reconstructions, video surveillance to detect pedestrians or for face detection, autonomous driving to navigate, robotics to interact with objects ... It is generally a new meaningful representation to analyze an image.

Two types of segmentation mainly exist in image processing, namely semantic and instance segmentation. If one needs the position of a said object to interact or analyze, one will need instance segmentation. On the other hand, understanding a scene requires "only" a semantic segmentation, e.g., moving a robot.

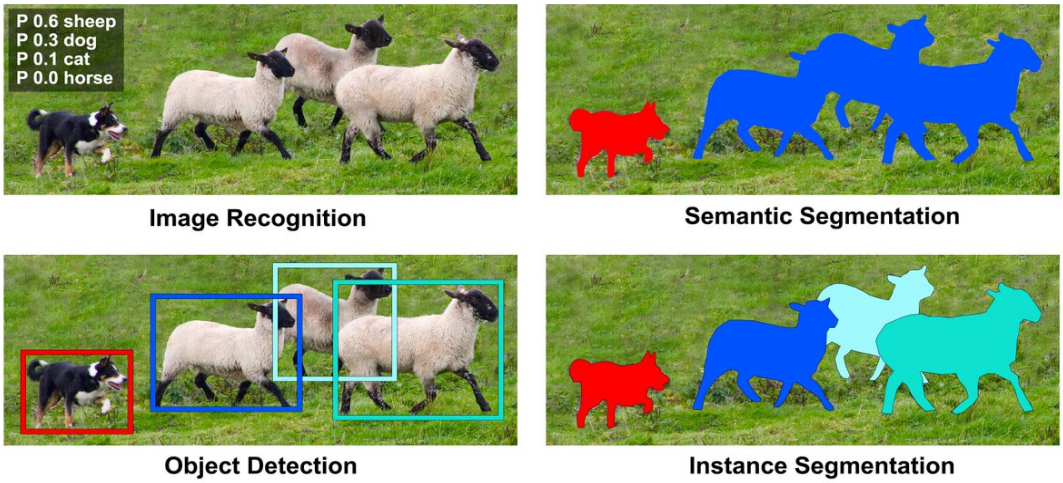


FIGURE 1.1: Segmentation related task. *Image from ai-pool*

In this thesis, we will focus on the latter. The former is more complex, precise and can be seen as the combination of object detection and semantic segmentation. It doesn't enter for now in our scope, but it remains a possible future goal, especially to get a robot to interact with objects. Fig.1.1 shows the different tasks on the same image.

1.2 Incremental learning

In many applications, the data is fully available a priori, and retraining a similar model using all the data, new and old, is not time and money efficient, especially

when the network contains hundreds of millions of parameters and the dataset hundreds of thousands of elements.

Incremental learning consists of learning per batch of data at a time without necessarily new classes to learn. Few similar terms are sometimes grouped with incremental learning, for example, continual/continuous/lifelong learning, where one model learns during a lifetime. It also has to be differentiated from online learning, where a model learns as the data are coming, and also from sequential learning, where data are correlated in time. Incremental learning can then be online, but the inverse can't be. Nonetheless, the terms can still be debated in the continual AI community, and subtle differences are yet to be strictly defined.

Incremental learning techniques can be split among three main categories :

- Replay-based where one keeps old data or generate ones
- Regularization-based where one tries to constraint the learning
- Parameter isolation-based where one wants to link parameters to the tasks. For example, one can add and learn new branches in a neural network at each new task while freezing old ones

A lot of sub-branches exist, as explained in [1]. In our case, we will mainly study the regularization one, which is also the most studied. The other two types don't suit the robotic setting due to the memory and power constraints of the robot.

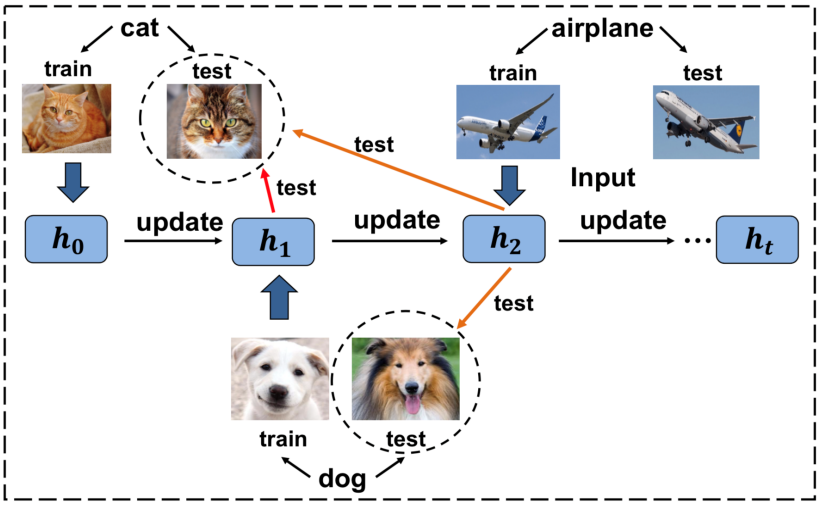


FIGURE 1.2: Class-incremental learning. *Image from purdue.edu*

Class-Incremental learning can be defined by learning new similar "classes" of a task step by step, allowing an update of the network/method without reusing directly previous data. Note that if previous data are out of the question in these settings, we will see that one can use a public dataset, labeled or unlabeled, even generated data to execute incremental learning.

Fig.1.2 shows the basic functioning in classification. In the particular case of semantic segmentation, the functioning is similar to a particularity. While classifying image results in a single output logit, semantic segmentation contains different classes within the image. State-of-the-art techniques presented in section 3 "reused" images by masking labels of classes from previous learning, resulting in learning only the new ones.

As powerful and practical as it sounds, the main problem raised is catastrophic forgetting, defined as losing all knowledge from previous learning resulting in a

good score in the new task learned but rapid degrading scores for previous tasks. Indeed, fine-tuning a neural network with new data without old ones bias the weights towards the classification of the new data.

1.3 Robotics

As mentioned before, segmentation is a critical low-level task for environment perception. It gives robots an easier representation of their environment to navigate and interact. Few results from offline learning in an indoor setting can be seen in Fig.1.3.

We placed ourselves in a realistic and real case scenario, using a robot that moves around a house. Note that the images are purely from indoors, and the robot is a mobile one and not an "industrial" ones. Also, although a robot is used, it is first controlled by a human to obtain data for supervised learning. It is an essential remark regarding active learning in robotics, also called the decision-making process (to not be confused with active learning in machine learning for efficiency data selection). This setting also limits memory and computational power. Thus the choice of the architecture used.

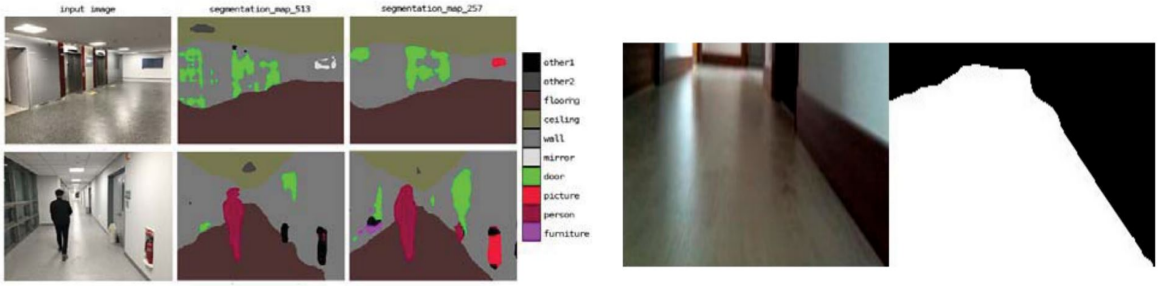


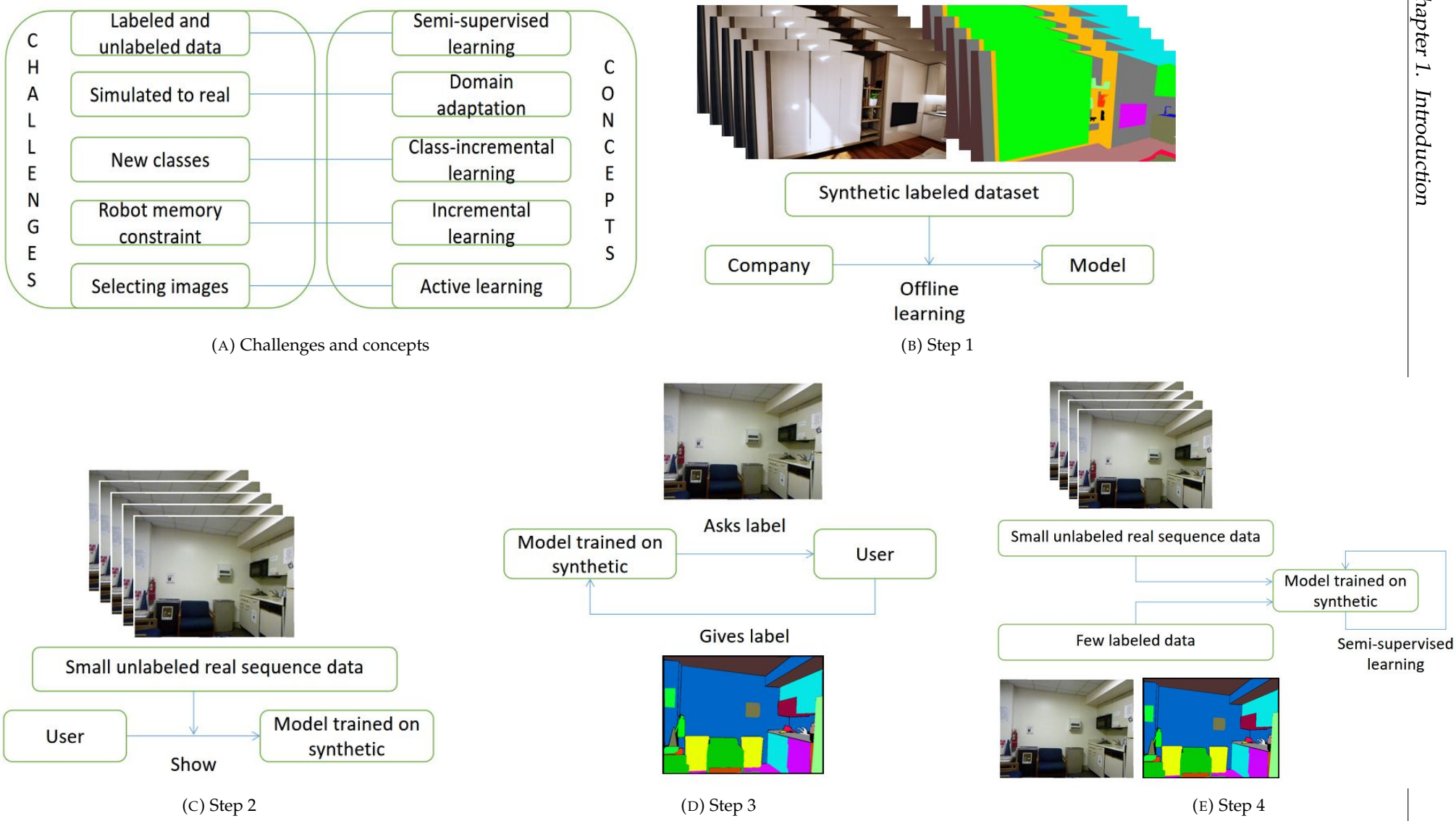
FIGURE 1.3: Examples results on indoors. *Images from [3] and [4]*

Synthetic datasets are more and more used, notably in autonomous driving and robotics. As labeling data manually is time and money-consuming. The simulation to real data is more and more studied. A famous example is the use of the game grand theft auto (GTA) for autonomous driving.

Few central problematics arise at the use of synthetic datasets: can generated data cover enough real cases? Do they have to be realistic? If the emergence of ray-tracing seems to be a leap forward, it still requires powerful computational power. Also, as synthetic data don't show the issues from the image sensor or digital camera in terms of the acquisition, are they suitable for learning? These challenges are studied respectively as outliers, technical and domain adaptation issues.

In our specific case of indoors learning, incremental learning is not enough suitable word description. Hence the use of the word "context". It emphasizes first the sequential nature of learning from one room to another and second that the network will learn on a set of new classes instead of only one class at a time, like in the presented papers in section 3. To the best of our knowledge, it is the first studied case of context incremental learning.

Fig.1.4a summarizes the different research domains which tackle the different challenges to develop context incremental learning in robotics. Fig.1.4b, 1.4c, 1.4d, 1.4e show a pipeline example of a real case scenario. This thesis will only study a modified first step where a company wants to update its model trained on synthetic data.



(A) Challenges and concepts

(B) Step 1

(C) Step 2

(D) Step 3

(E) Step 4

FIGURE 1.4: Ideal pipeline aimed. Steps 2,3 and 4 are repeated until all rooms are seen

Chapter 2

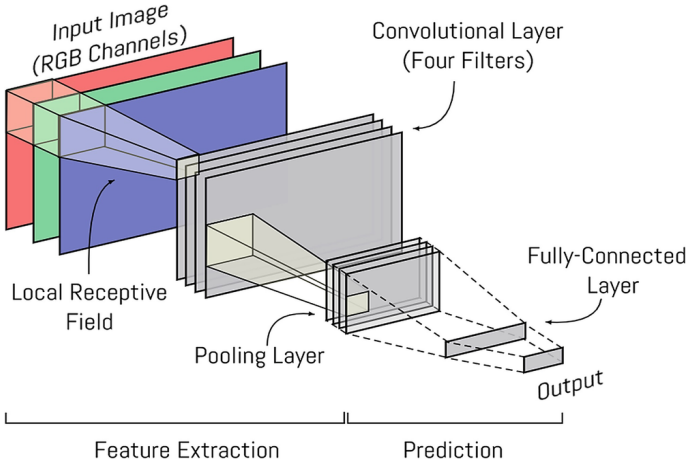
A deep learning architecture for segmentation

2.1 Convolutional neural network (CNN)

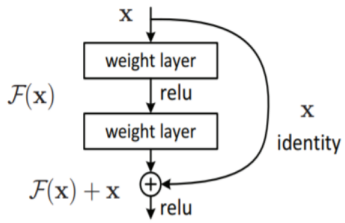
2.1.1 Definitions

Fig.2.1a shows the typical convolution layers of a CNN, which are often stacked one to another. We specified the diverse parts that compose them.

- Convolution in image processing consists of multiplying term by term and summing a defined matrix sliding to an image along the width and height axis. The goal is to get spatial information and highlight characteristics in an image depending on the values of the defined matrix.



(A) Classical convolutional layers



(B) Residual block

FIGURE 2.1: Convolutional layers.
Images from neurohive and link.springer

- In deep learning, the values of a convolution (more often named weights) are learned by gradient descent identically as for fully connected layers.

  One needs to fix a few hyper-parameters: mainly the number of filters and the kernel/filter size. There are some rules for fixing both, which impact the time and space complexity and the potential of the network, but it remains an open problem. Also, the kernel size affects the receptive field, which is the number of pixels from the input image which contribute to the current pixel.

The primary use of convolution in deep learning remains for the reasons above and reduces the number of weights.

- Pooling is a sub-sampling method and similar to convolution but with a more flexible processing part. A simple argmax or mean at each computation is often chosen. It is mainly used to reduce the size of an image and the number of parameters for the next convolution. It can be seen as a regularization method as two similar images are even "more similar" after a pooling layer.

- Batch normalization consists of normalizing the inputs by computing mean, and standard deviation learned gradually from training. They are placed before the convolution layer, accelerating the training by putting the same scale within the features.

- Activation functions are applied after the convolution layer to get non-linearity in the process. The ReLU function defined by x if x>0 else 0 is often used due to its weak need for computational power.

  A lot of variants of the ReLU have been studied, like leaky ReLU. Other common activation functions like tanh or sigmoid are avoided due to their behavior in the extreme sides of their function where the gradients are exponentially small, thus not allowing efficient network updates. This phenomenon is called the vanishing gradient.

2.1.2 Resnet

Winner of the ImageNet competition in 2015, ResNet is a deep neural network architecture developed by [5] resting upon a residual block. It remains a competitive network and is still used as a baseline in computer vision research.

A residual block uses one or multiple layers of convolution and, if willing, batch normalization, pooling, and activation function, then summed to the output of these the input itself. The main idea is that learning the difference between the input and output is easier than learning to predict the output from scratch.

It allowed to reduce the vanishing gradient problem, i.e. train deeper neural network and ease the training as less exploration is done in the features space. (where more epochs are needed to recover between close manifolds).

This straightforward but efficient addition can be seen in Fig.2.1b

2.2 Encoder-decoder

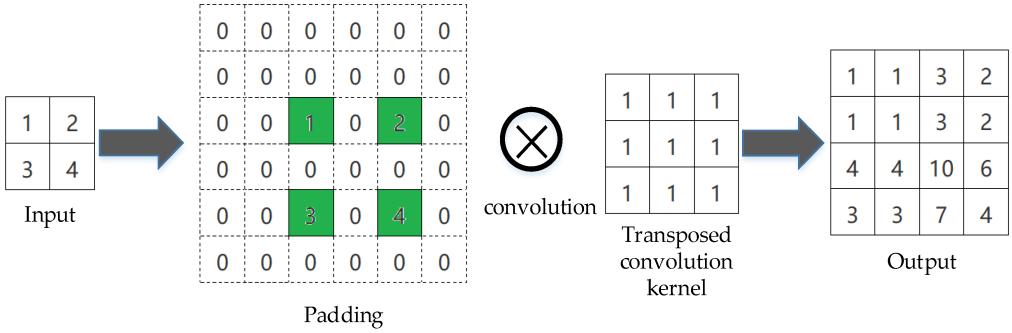
2.2.1 Definition

State-of-the-art deep learning methods for semantic segmentation rests on different methods: encoder-decoder architectures, e.g. the deeplab architectures or fully convolutional network, to name a few.

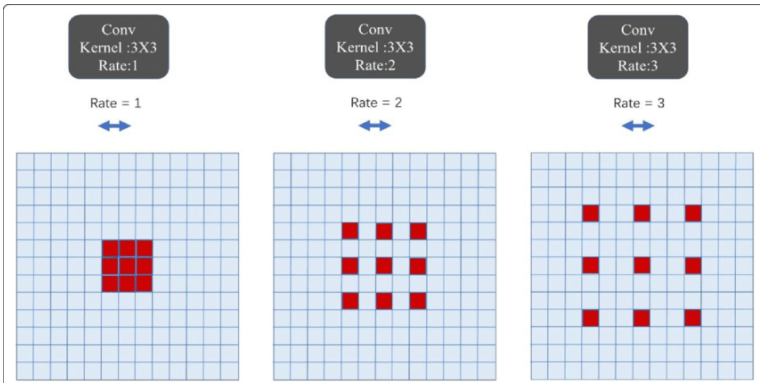
Encoders, as their name said, give another representation of an input. The result is often a compressed representation of the input for memory and features representation purposes. The decoder then can output and decipher the encoded information regarding to the task learned.

The size of the compressed representation is smaller than his input, thus depending on the task, e.g. denoising, auto-encoder, or segmentation ... there is a need to upsample the encoded element to the exact size of the first input.

One can use transpose convolution showed in Fig.2.2a. The latter also named deconvolution, is identified as a convolution as explained in section 2.1.1., except that the input is padded between the pixels to allow the upsampling of the input.



(A) Transpose convolution. *Image from MDPI*



(B) Atrous convolution. *Image from Research-gate*

FIGURE 2.2: Types of convolution

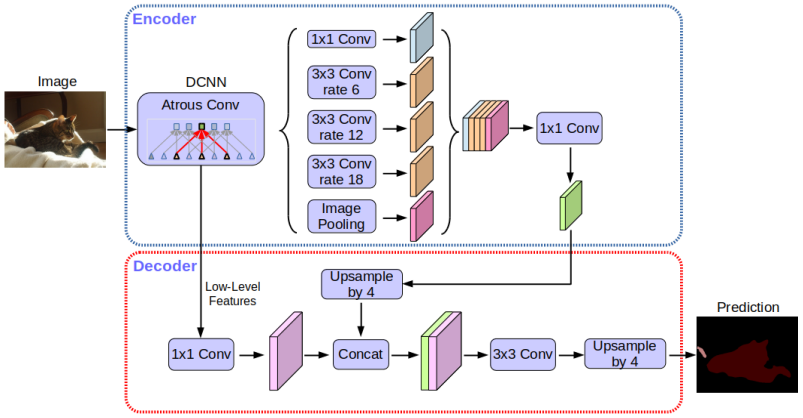


FIGURE 2.3: DeepLabv3. *Image from [6]*

2.2.2 DeepLabv3

DeepLabv3 developed by [6] is a widely used encoder-encoder architecture for semantic segmentation. The main key element is the atrous convolution which is a convolution with spaced defined matrix depending on a hyperparameter called rate.

It is a convolution with holes only between elements of the matrix. See Fig.2.2b. The point is to get a larger receptive field that raises exponentially with respect to the input while still getting a linear growth in terms of the number of parameters.

Note that the "typical" convolution is an atrous convolution with a specific rate of 1. See Fig.2.2b. It shouldn't be confused with transpose convolution in Fig.2.2a, which adds padding around the elements of the input for upsampling purposes.

Using different rate values simultaneously to the same input allows getting different levels of details and context. In DeepLabv3, these outputs are then concatenated to be the input of a fully connected layer for classifying each pixel. Fig.2.3 shows the DeepLabv3 architecture.

2.3 Metrics

The task of semantic segmentation can be defined as classifying each pixel of an image. The general loss function used in semantic segmentation is the same as in the classification task: the cross-entropy. Hence the other designation: dense prediction.

Simple accuracy can be used to evaluate the network but is inadequate. By looking at the mathematical formulation,

$$\frac{TP+TN}{TP+TN+FP+FN} \text{ where T=True, F=False, N=Negative, P=Positive}$$

one can realize that the measure can be relatively misleading when one reports a high case of true negatives, thus a high accuracy.

The intersection over union (IoU), also known as the Jaccard index, is a more meaningful metric. It is defined mathematically by $\frac{TP}{TP+FP+FN}$ making up the accuracy metric problem.

Chapter 3

State-of-the-art in incremental learning for semantic segmentation

Incremental learning has been revisited since the breakthrough of deep learning in the 2012 ImageNet competition. Since then, incremental learning for classification and object detection has been extensively studied.

The case for semantic segmentation may be the most recent task in incremental learning for image processing. To the best of my knowledge, the first paper around the subject since the new deep learning era dated from September 2019.

Some terms have to be described in the case of incremental learning : Settings 20-1, 16-5, 16-1, respectively means that we are in the cases of training first with 20, 16, and 16 classes and incrementally learn 1, 5, and 1 classe(s) at a time until there are no more new classes to learn (in this example, it was 21). Some papers used the term 16-5s to denote 16-1, using the letter 's' to emphasize the sequential nature of the training.

A summary of the state-of-the-art can be seen in Fig.3.5.

3.1 Knowledge distillation

[8] applied the method of [7] for incremental learning in semantic segmentation. The method called distillation loss is one of the most simple techniques to retain knowledge. It is a constraint often applied on features space or output layer, e.g. putting a L1 or L2 regularizer to prevent the weights to change too much between a network at time T and its T+1 network counterpart. They gained 6, 10, and 20% in mean IoU compared to finetuning in the 20-1, 16-5, 16-1 settings with the Pascalvoc2012 dataset (See chapter 4.1.1. for a description of this dataset).

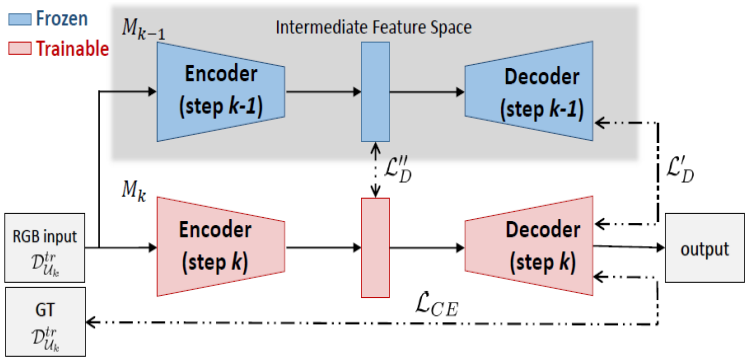


FIGURE 3.1: Classic knowledge distillation. *Image from [8]*

Fig.3.1 shows a schematic representation of the knowledge distillation with M_{k-1} and M_k the networks at times k-1 and k. L''_D the L2 loss between their features and L'_D the L2 loss between their probabilities outputs. L_{CE} is the classical cross-entropy.

[11] tests almost any combination possible between adding or not features distillation, output distillation, freezing all, some layers, or not at all the encoder in a complete paper. They conclude with no obvious better combination, as improvements depend on the setting.

Nonetheless, [9] mentioned that there is a bias towards old classes as this latter intervenes not only on the cross-entropy loss but also on the distillation loss when one uses knowledge distillation in outputs layers. They rectified it by applying cross-entropy loss only on the new classes. Each pixel then is counting once in the total loss, in the cross-entropy (L_{CE}) if it concerns a new class or in the loss between probabilities outputs (L'_D) in the case of an old class. It results in a slight improvement of a few percent comparing to [8] on the cityscapes dataset (a driving dataset).

3.2 Background shift

(A) Step 0 (B) Step 6 (C) Step 7

FIGURE 3.2: Example of background-shift with the evolution of the ground-truth function of the step

A crucial point raised by [10] is what they called the background shift. Background of an image at time T may contain new classes from images at time T+1 but still labeled as background at time T. Similarly, the background of an image at time T+1 may have old classes. An example can be seen in Fig.3.2. A problem arises as we tell the network that some pixels are at time T from a particular class and at T+1 from the background or vice-versa, confusing the network.

Masking the background pixels could be a simple solution, but [10] proposed to sum over the output of the old network to retain information from the pixels. They then revisited the cross-entropy and distillation losses.

Let $L_{CE}^{\Theta^t}(x,y) = -\frac{1}{|I|}\sum_{i\in I} log q_x^t(i,y_i)$ be the usual crossentropy. Instead of q_x^t the probabilities output they use with Y^{t-1} the old classes

$$\tilde{q}_x^t(i,c) = \begin{cases} q_x^t(i,c) & \text{if c} \neq \text{b} \\ \sum_{k\in Y^{t-1}} q_x^t(i,k) & \text{if c=b} \end{cases} \tag{3.1}$$

Let $L_{KD}^{\Theta^t}(x,y) = -\frac{1}{|I|}\sum_{i\in I}\sum_{c\in Y^{t-1}} q_x^{t-1}(i,c) log q_x^t(i,y_i)$ be the usual knowledge distillation on outputs probabilities. Instead of q_x^t they use with C^t the new classes

$$\hat{q}_x^t(i,c) = \begin{cases} q_x^t(i,c) & \text{if c} \neq \text{b} \\ \sum_{k\in C^t} q_x^t(i,k) & \text{if c=b} \end{cases} \tag{3.2}$$

3.3 Self-training

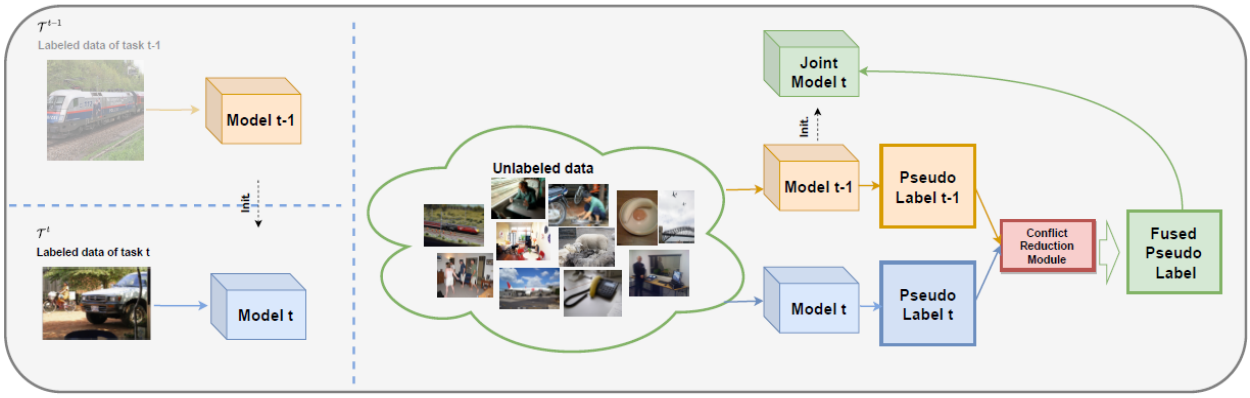


FIGURE 3.3: Self-training pipeline. *Image from [12]*

The second central concept in incremental learning for semantic segmentation is self-training, in particular, used in [12] and [13]. Self-training is a semi-supervised learning method. One can link it closely to knowledge distillation as it compares the outputs of one's model at time T-1 and T.

One of the main self-training points is to generate pseudo-labels. Chronologically by following the Fig.3.3:

- one trains a model T-1 on the new data to get a model T
- pseudo labels are created from the two networks T-1 and T with a conflict reduction mechanism solving the "conflict" between the outputs of the two models
- the model T-1 is retrained with the fused pseudo-labels resulting in knowledge conservation to obtain the "final" model T

The conflict reduction mechanism defined by [12] works in the following way: the fused pseudo labels follow the network, which doesn't predict the background. In the case of the two networks T-1 and T not predicting the background, the fused pseudo label selects the highest probability.

It is a flexible method as the data used for fused pseudo labels can be labeled or unlabeled, and [12] denotes the gains function of the size of this dataset. Notably, even a small fraction of a dataset can lead to few percentages of IoU gains. [12] recommends the Fréchet inception distance to measure the similarity between two datasets to choose the self-training dataset but also shows that using a semantically different dataset can lead to IoU gain.

3.4 PLOP a spatial and multiscale network

[13] is the paper chosen to be the base of this internship for the following reasons:

- the results are on par with the state-of-the-art if not is the state-of-the-art
- code is publicly available in GitHub
- it regroups the differents concepts presented above

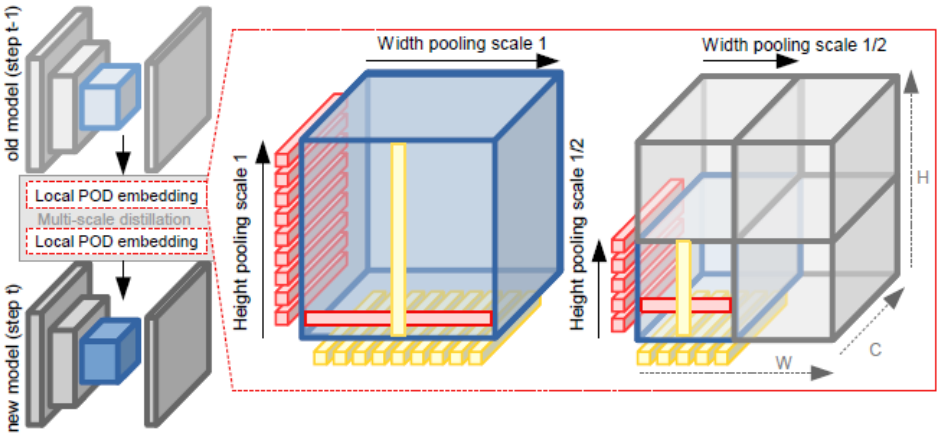


FIGURE 3.4: Local POD embedding for knowledge distillation. *Image taken from [13]*

New elements are added with :

- spatial characteristics linked to the distillation part. The knowledge distillation comes from a local pool outputs distillation (POD) embedding where new features are created from features layers produced by the model. POD is composed of the concatenation of vertical and horizontal slices of the features layers. See the left part in the red box of the Fig.3.4
- multiscale characteristics coming from the concatenation part above are applied on different scales to retain short to long spatial dependencies. See the right part in the red box of the Fig.3.4
- the use of pseudo-labels but for the background-shift. Unlike the description of [12] in section 3.3, the network is not trained two times to get the pseudo-labels. The pseudo-label is applied simultaneously during the training of the new class pixels on the pixels labeled in the background. These pixels can be actual background pixels or pixels masked at a said step due to background shift. The network at time T-1 output a label for the training of the network at time T retaining information
- a threshold based on entropy in the conflict reduction mechanism to prevent pseudo-label with low confidence

Title	Nature	Main idea	Positive notes	Negative notes
Incremental learning techniques For segmentation (ILT)	Knowledge Distillation	Cross-entropy loss on all classes Froze encoder Distillation loss on output layer And/or in features space	Simplicity Applicable to existing networks	Reusing old labels in new images Bias towards old classes No background-shift
Class-incremental learning For semantic segmentation reusing neither old data nor old labels (CIL)	Knowledge Distillation	CE loss only over new pixels classes Distillation loss in all other parts of the image Adding weights pixel-wise, the more uniform The soft output is, the bigger the weight	Reduce low performance on small/difficult objects By getting a bigger gradient	Cityscapes used No background-shift
Modeling the background for incremental learning in segmentation (MIB)	Background shift + Knowledge Distillation	CE Loss : if GT not background then sum on old classes Distillation loss : same trick Initializing the new classifiers uniformly with the background weights in previous step	Taking in account the uncertainty over the background class	
A few guidelines for incremental few-shot segmentation (iFSS)	Few-shot learning + Classes prototypes + Knowledge Distillation	Compute classes prototypes with cosine classifier New classifier initialization strategy wrt prototype Naive fine-tuning Batch renormalization Knowledge distillation with l2 constraint	Batch renormalization is good for sequential data And for small batches	Setting a bit different Harder to compare the results
Knowledge distillation for incremental learning in semantic segmentation (ILT2)	Knowledge Distillation	Same as ILT Froze encoder on first layers Softmax with temperature Distillation on ASPP dilation layers	Temperature works well with one class added at a time Combination exhaustivity of Knowledge distillation techniques	Weak improvements
Self-training for Class-incremental semantic segmentation (StCIL)	Self Training	Self-training with pseudo labels : Finetune copy of old model on new data only Generate pseudo-labels on unlabeled data with the two models Conflict reduction scheme to fuse the predictions Retrain the old model on pseudo-labels Soften output overconfidence by maximizing self entropy loss Choice of unlabeled dataset with Fréchet inception distance	Self-training can complement other methods Several datasets to consider for the pseudo-labels giving more possibilities/potential More unlabeled data, better it is More epochs on self-training, better it is	Complexity in time and space Double training Need storage for unlabeled dataset
PLOP : Learning without forgetting for continual Semantic segmentation (PLOP)	Self Training + Knowledge Distillation	Multi-scale local distillation Self-training with pseudo labels Different method for conflict reduction	Similar as above Better results on Ade20k (similar to RobotriX)	Similar as above
Continual semantic segmentation Via repulsion-attraction of sparse and disentangles Latent representations (SDR)	Classes prototypes + Knowledge Distillation	Prototypes matching with Frobenius norm GT downsampled to match latent dimensions Contrastive learning with repulsion and attraction losses Features sparsity loss by minimizing an energy Using knowledge distillation from MiB	Sparsity and matching domains	Weaker improvements than self-training papers Complexity in time and space

FIGURE 3.5: Table summarizing state-of-the-art

Chapter 4

Experiments

4.1 Datasets

4.1.1 Pascalvoc 2012

The Pascalvoc 2012 dataset comprises almost 10'000 images for 21 classes with images from diverse vehicles to indoors objects by way of animals and person. An example of [16] segmented by [6] can be found in Fig.4.1. This dataset is the primary used one in chapter 3 by the presented papers.

Current state-of-the-art results can be found in Table.4.1. While the non-incremental setting had reached the 90% milestone, the incremental setting has a long way to go, whether it be in 16-5, 16-1, or the 11-1 settings with respectfully 70.09%, 54.64% and 30.45%.

	EfficientNet L2	DeepLabv3 +ResNet101
offline	90.5	77.4
16-5	NA	70.09
16-1	NA	54.64
11-1	NA	30.45

TABLE 4.1: Mean IoU from paperswithcode

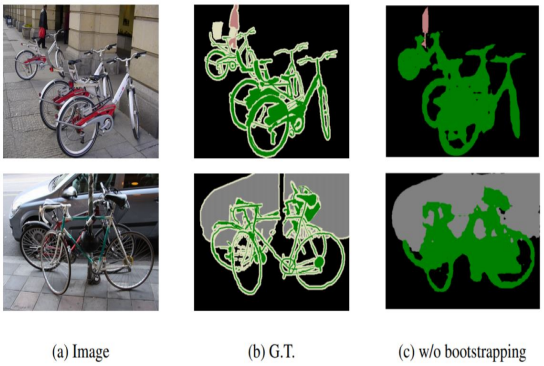


FIGURE 4.1: Images from [6]

4.1.2 RobotriX

Description

The dataset consists of around eight million images distributed between 8 rooms for 39 classes. However, only half of it is downloadable, and only 90'000 images from 3 rooms for 24 classes are usable due to missing labels in the data. Fig.4.2 presents an image from each of the three rooms available.

The images are hyper-realistic and with a 1920x1080 (1080p) resolution. They are also sequential; thus, the images are highly correlated in time and not independent and identically distributed. This point has to be taken into account during training.

Five points of view are available, but only the vision from the robot has been kept for this thesis. Note that a human controls this point of view from the robot in the training set. It is an essential remark regarding active learning in robotics, not to mix up with active learning in deep learning. With few tools delivered by [17],

one can generate images and labels for segmentation and object detection using a VR headset.

The high resolution implies fined details and increases learning and inference time, which is unsuitable for household robots. That is why we used diverse pre-processing like cropping and resizing to sacrifice the less details as possible. One can even argue that a smaller image size acts as a regularization method to prevent overfitting to these same details. In our case, we will go from 1080p to 480p, which is a 480x640 resolution.

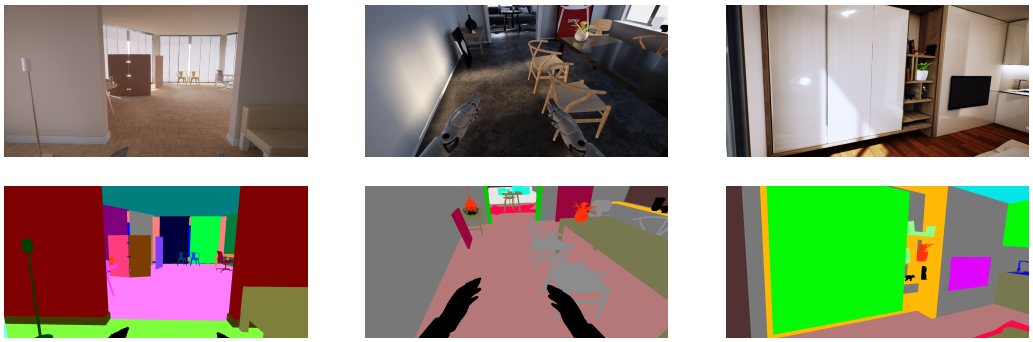
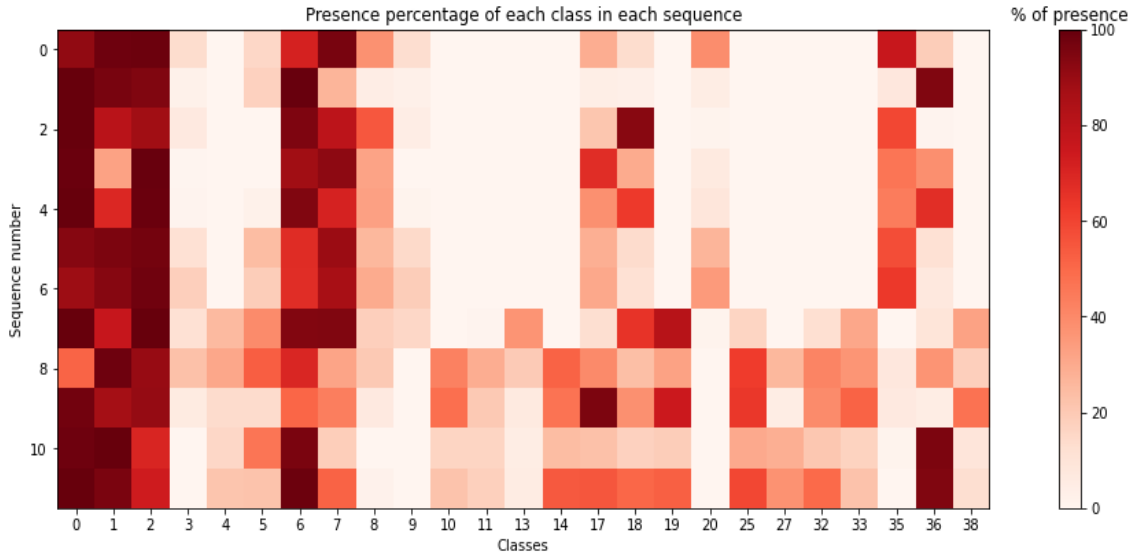


FIGURE 4.2: 1080p images from 3 differents rooms of RobotriX [17]
First row: rgb images. Second row: semantic segmentation.

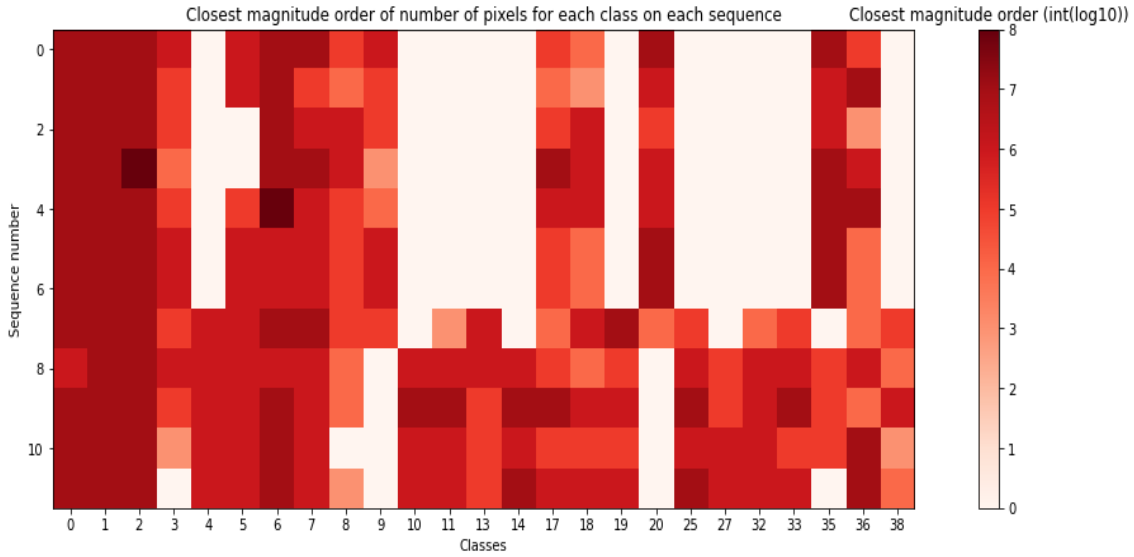
Data analysis

Fig.4.3a and 4.3b show that the available dataset is unbalanced with respect to the classes' presence and the number of pixels of each class. We mean by"present" if at least one pixel of the said class appears in an image. It's an important remark that we will develop later on. We can notice that some classes are present in each of the rooms, e.g., the class "wall" while some objects are exclusive to a room, e.g. "kitchen stuff".



(A) Classes presence on each sequence

FIGURE 4.3: Data analysis



(B) Number of pixels of each class for each sequence

FIGURE 4.3: Data analysis

4.2 Replicating PLOP results

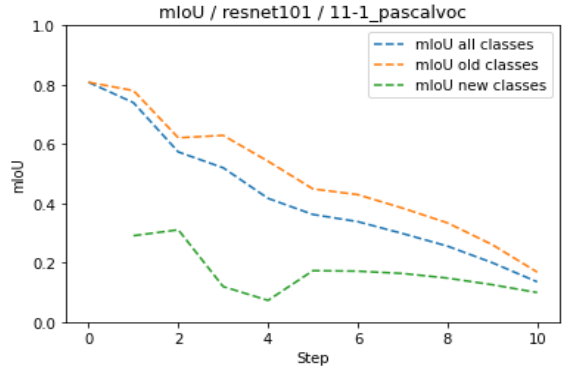
4.2.1 Implementation details

The replication has been done with a Titan Z with dual GPUs of 8Go ram each. Like the authors of PLOP, we run the experiments with Nvidia's Apex library to get half-precision and in-place batch normalization library for computation and memory efficiency.

However, the PLOP authors used two Titan Xp with 12 Go of VRAM each and a batch size of 24 equally distributed to the GPUs. To mimic this training as much as possible, we get a "virtual" batch size of 24 by running several batches of sizes 12,8 or 6 and accumulated the gradients before taking an optimization step.

The hyperparameters used are the same as in [13].

4.2.2 Difficulties in 11-1 setting



(A) 11-1 Replicated results

Table 3: Mean IoU on Pascal-VOC 2012 10-1.

	VOC 10-1 (11 tasks)			
Method	0-10	11-20	*all*	*avg*
ILT [54]	7.15	3.67	5.50	25.71
MiB [8]	12.25	13.09	12.65	42.67
PLOP	**44.03**	**15.51**	**30.45**	**52.32**

(B) 11-1 PLOP original results from [13]

FIGURE 4.4: Pascalvoc 2012 11-1 original and replicated results

In all figures, we called "old classes" the classes learned at the first step (step 0 actually) and "new classes" the incrementally learned classes since step 0. For example, in the 11-1 setting at step 5, we referred to the 11 first classes as "old" and 12 to 16 as "new".

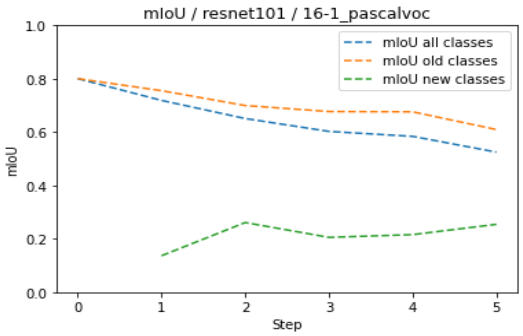
We couldn't reproduce the results in the 11-1 setting with a mIoU around 18% in Fig.4.4a against 30.45% in Fig.4.4b. The reason may reside in the in-place batch normalization affected by the "virtual" batch size, as the normalization parameters are updated with smaller batch sizes. It is unlikely as the images are diversified enough and not sequential, but we could not think of another reason.

Nonetheless, the 55% mIoU in the 16-1 setting has been successfully replicated. See Fig.4.5a and 4.5b. We suppose from now on the proper functioning of the code.

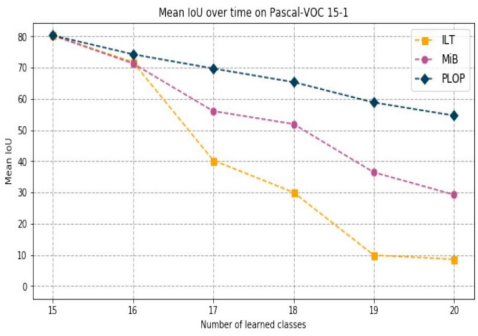
4.2.3 The mean IoU problem

One can argue that the mean IoU may not represent the performance well as it does not analyze the strengths and weaknesses in the case of incremental learning.

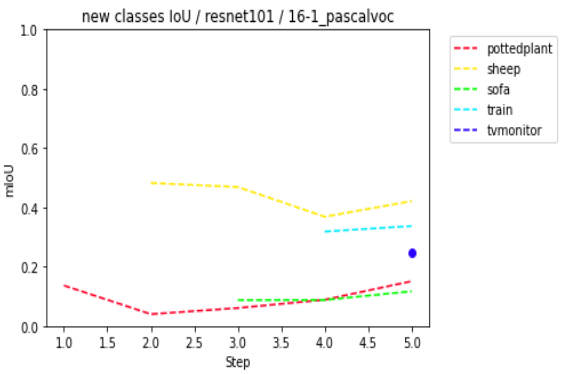
For example, some steps can deliver good mean IoU but with high variance between the classes. In that case, it is more than interesting to analyze the conditions of such results, e.g. it can be the nature of a class similar to previous ones or completely different which may allow easier or more complicated learning.



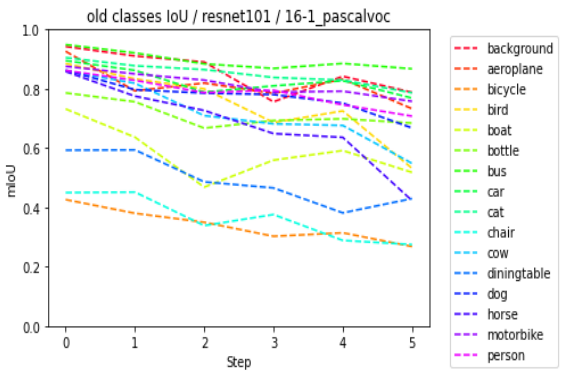
(A) 16-1 Replicated results



(B) 16-1 PLOP original results from [13]



(C) 16-1 Replicated results - new classes only



(D) 16-1 Replicated results - old classes only

FIGURE 4.5: Pascalvoc 2012 16-1 original and replicated results

Moreover, some classes are challenging to learn, i.e., with a close to 0 IoU. It might be interesting to avoid learning the said class as the training modified the

weights and thus the results in other classes. A method to detect this situation could be helpful to enhance results. Naively one can train a new class and discard it if necessary.

Interestingly, Fig.4.5c and 4.5d show that in the case of PLOP, some new classes have a very low IoU, around 0.2, while the old classes remain relatively high. This type of behavior results in "good enough" results in terms of mean IoU. But for comparison, if we let the network trained on the first 16 classes with a mIoU of 80% as it is for the next five steps, it results theoretically on a mean IoU on 21 classes of around $\frac{80}{100} * \frac{16}{21} = 61\%$ which is higher to the 55% of the PLOP results but with the five new classes at around 0%.

4.3 Incremental learning on RobotriX

4.3.1 Implementation details

The experiments have been done with an RTX 2080 ti with 11Go of VRAM, which is an efficient graphic card today. We use the automatic mixed precision (amp) package from PyTorch instead of the apex library for mixed-precision as amp is more flexible and part of PyTorch; thus, PyTorch compatibility and updates are guaranteed.

After few experiments with similar or even identical hyperparameters (see the following subsection), we notice similar results between the PLOP method on Pascalvoc2012 and RobotriX. Thus supposing that the PLOP authors have fine-tuned the hyperparameters on Pascalvoc 2012, we hypothesis that they translate on the RobotriX one.

Therefore, we didn't think that a brute grid search was necessary and only tune parameters one by one to see their effects.

From these experiments, we also decided to raise for subsequent experiments the learning rate for incremental classes learned from 0.001 to 0.01 as 0.001 was too precautious to avoid divergence.

4.3.2 Naive experiments with Pascalvoc2012-like settings

Settings

We first try naive experiments to see how the model stands in similar conditions as Pascalvoc2012 experiments. We use every 10th image of the data for memory efficiency and due to the high framerate of the sequence giving relatively similar images. We split randomly with 70/15/15 ratios for train/validation/test sets.

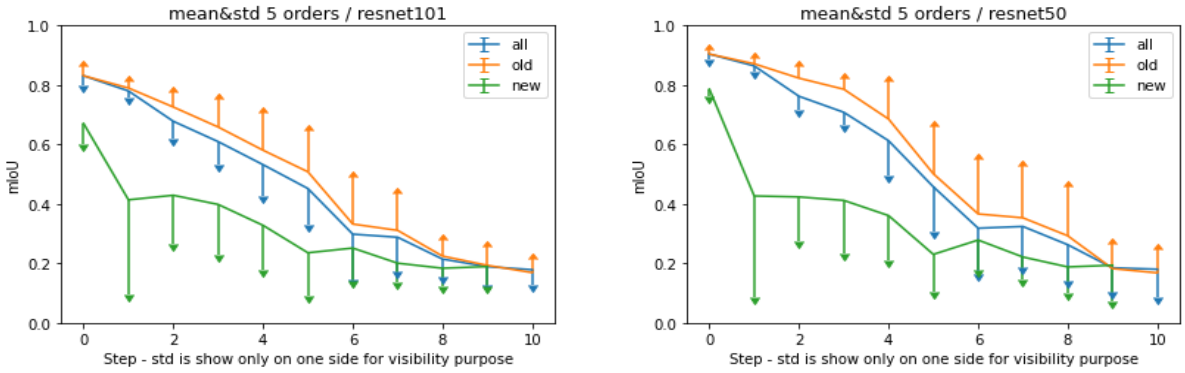
It is a naive experiment as the robotrix dataset comprises sequences at 60fps, meaning that every 10th image is relatively similar. It is a golden rule in machine learning that train and test sets don't overlap.

We first used a 512x512 centered cropping in Figure 4.6., then a 1280x960 cropping and 640x480 resize for any other figures in this report. We chose these processing because first, the center is more important than other image parts. Second, the 480p resize retains context and reduces training and inference time. The resize is done with nearest neighbors interpolation for two main reasons :

- we decided to process on the fly to avoid keeping all different sized images
- the ground truth segmented image also needs to be processed. It is easier to do so than with a more precise but more complex method as bilinear or bicubic interpolations where one needs to decide for each pixel his segmented part.

First results

We use five different classes orders to see how robust the method is. In Fig.4.6a, the curves show the mean results and arrows their standard deviation. For visibility purposes, the standard deviation has only arrows directed to one side. While the training of 10 first classes shows expected high results, we see that they rapidly degrade the function of new classes learned despite being a relatively easy setting. It shows the level of difficulty of incremental learning. Qualitative results can be seen in Fig.4.7 with notably the last two steps predictions manifesting catastrophic forgetting.



(A) Resnet101 (B) Resnet50

FIGURE 4.6: mIoU. 512x512 centered images. Random sampling

As expected, we get at the beginning a low standard deviation for the mean IoU because one new class shouldn't have much impact against the ten old ones but also because of the knowledge distillation.

Moreover, the standard deviation for the new class at the first step is surprisingly low before raising at each step. It then goes back to similar firsts values. In the end, we see the standard deviation is low, whether for old or new classes. It shows the robustness to classes order of the methods.

We also compare the ResNet101 and ResNet50 as backbones of the encoder with a 22 million parameters difference with the Fig.4.6a and 4.6b.Although they give similar performances, one will use the more extensive network possible in a real setting as it leaves more potential to learn new classes. In this thesis, we limit ourselves to the ResNet50 to able more experiments. Another point is that even if ResNet50 gives unexpectedly better overall performances, we can notice that the drop is also higher, which must be considered if one expects numerous new classes.

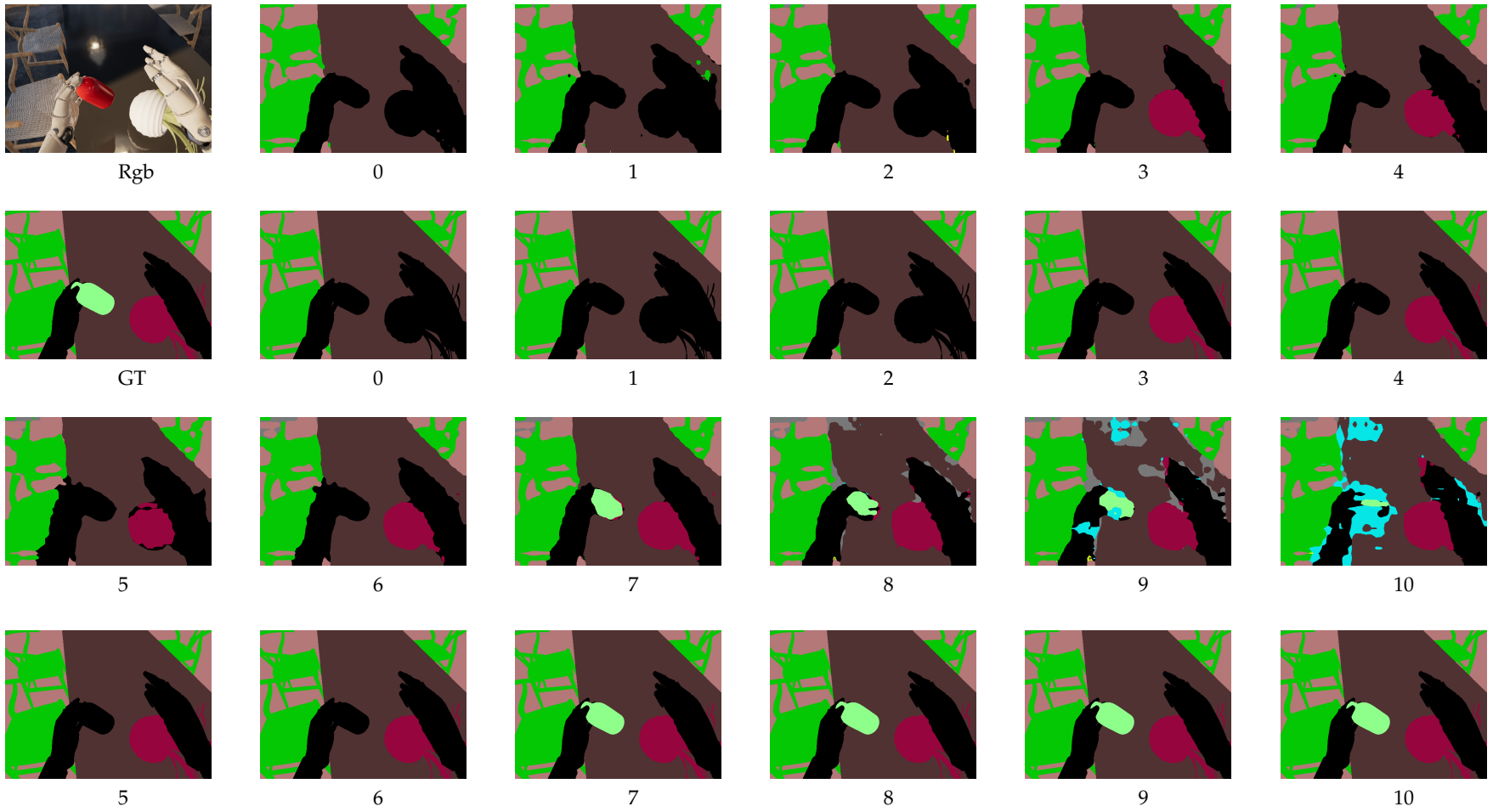


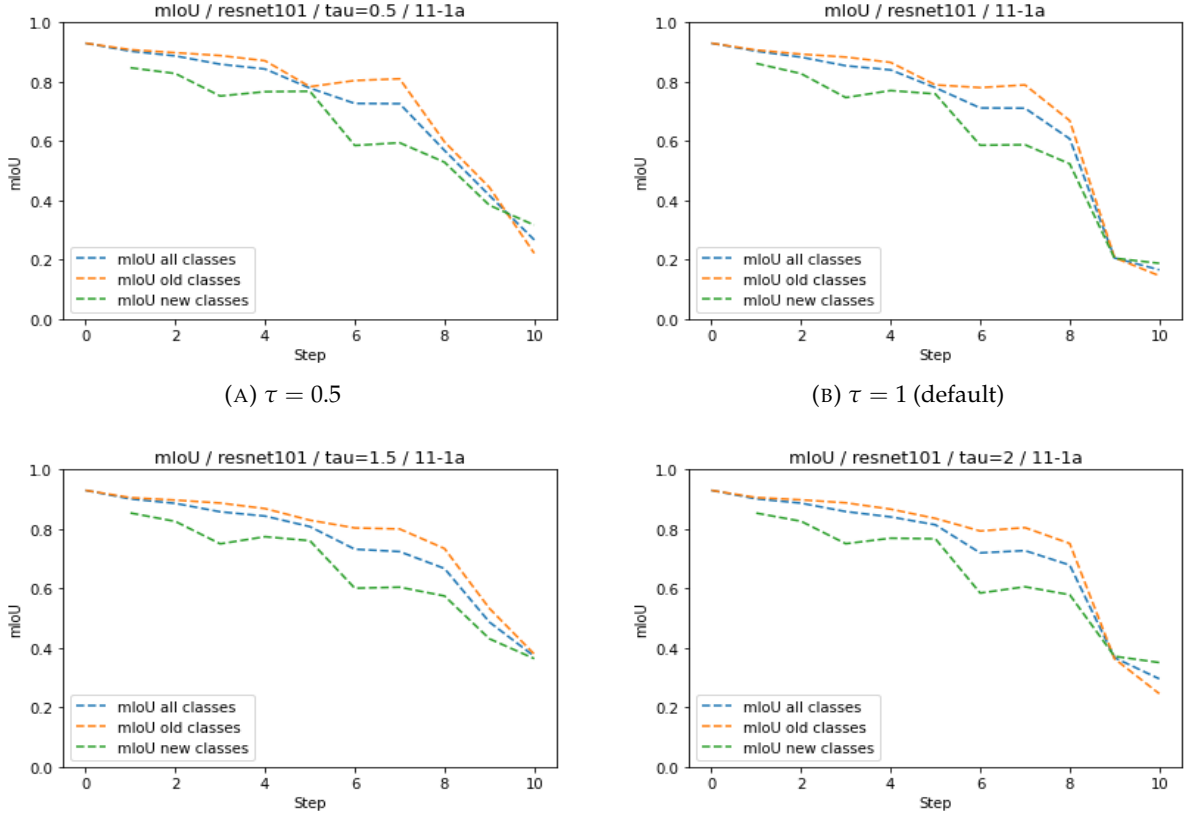
FIGURE 4.7: Evolution of predictions and GT function of step.
Random sampling. ResNet50.
First and third rows : rgb image and predictions. Second and fourth rows : ground truth

4.3.3 Hyperparameters

As mentioned in section 4.2.1, we fine-tuned the hyper-parameters mainly to see the behavior of the network. We chose the following ones that we suppose are the most interesting: temperature in the softmax, freezing the backbone, and changing the backbone.

Temperature in softmax

Fig.4.9b shows the effects of temperature. While the temperature with $\tau = 1.5$ is better in this case, it shows random results in other experiments. We decided not to go further and let the hyperparameter be as it is.



(A) $\tau = 0.5$

(B) $\tau = 1$ (default)

(C) $\tau = 1.5$

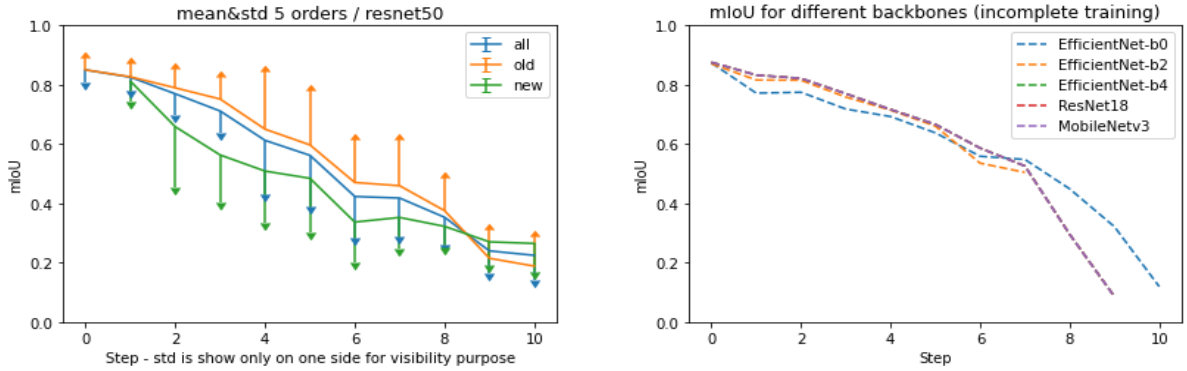
(D) $\tau = 2$

FIGURE 4.8: Temperature effect. ResNet101. 480p images. Random sampling

Different backbones

Following the relatively good results using a ResNet50, we try different backbones as it is relatively easy and fast to try. We don't aim to fine-tune any hyper-parameters. We use the lighter ResNet18 and different state-of-the-art architectures in the mobile and the less than 50 million parameters categories. Results in Fig4.9b show similar behavior from the different backbones until the seventh step before dropping significantly.

Last, we try a different direction with classical unsupervised learning: incremental principal components analysis (PCA) and independent components analysis (ICA). While we apply PCA directly on the images of the first step, we perform incremental ICA on patches uniformly extracted in the images and update at each step. Results were not promising, getting mIoU close to 10% at the first steps. Due to the number of hyper-parameters to fine-tune: features size, padding size, sampling rate, we didn't go further.



(A) ResNet50

(B) Dfferent backbones without fine-tuning any hyper-parameters

FIGURE 4.9: Different backbones. 480p images. Random sampling

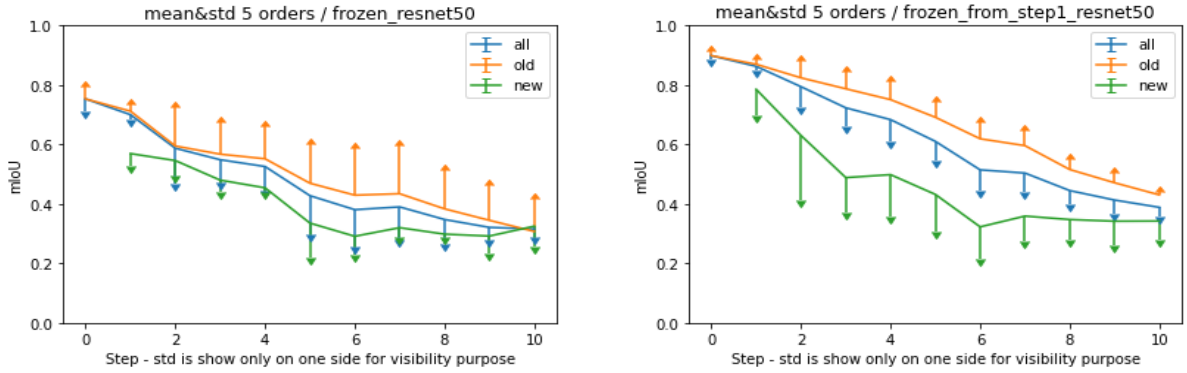
Freezing the encoder

Fig.4.10a shows results with with frozen encoder parameters. As we are using a ResNet pre-trained on ImageNet classification, one can consider letting these parameters as they are.

The thought behind is that not only does it lightens the memory and time complexity in training, but it also prevents overfitting as the features trained on ImageNet are more diversified than ones trained directly on RobotriX. This is actually our case as we see an strong increase from the fine-tuned one at 22% in Fig.4.9a to 32% in Fig.4.10a.

We also try to freeze the encoder only at the incremental part and fine-tuning it on the first step. Results in Fig.4.10b show relatively lower standard deviation and better absolute overall results with an increase from the completely frozen encoder at 32% to 38%. However, if the results are overall better, we can notice that the drop is also more important as it goes down from 90% in the first step to 38%, i.e., a 52% drop. The previous training with frozen encoder from the start went from 78% to 32%, i.e., a 46% drop. Depending on the number of classes one may encounter in the future, freezing from the beginning might be more advantageous in the long term.

We have decided for the rest of the experiments to use a pre-trained network that we fine-tune at the first step before freezing it for the incremental part. However, with hindsight, freezing the encoder from the start for the context incremental learning (section 4.4) would have been adequate. By fine-tuning on the first context/room, we might be biased towards the specificities of the said room, which could affect future training and performance.



(A) Encoder frozen from start

(B) Encoder frozen at incremental learning steps

FIGURE 4.10: Freezing encoder effect. ResNet50. 480p images. Random sampling

4.4 Context incremental learning on RobotriX

We move to a more realistic and challenging setting by learning a room instead of a class at a time. It means that :

- we feed the network all the sequences of a room at each step. Instead of randomly sampling all the images that concern the said classes within all sequences in all rooms. Similarly, the test set is composed of sequence(s). Thus the network sees fewer data.

  To illustrate this, supposing a class like "book" is present in the living room and the bedroom. Previously, we would use all images with "book" from the two rooms for training. From now, the training will only see the "books" in the room selected as old classes are not relearned between the two rooms. Due to the weak performance of some classes, we choose in section 4.4.2 to select few classes to relearn

- we don't learn one new class at a time but all new classes from the new room which can be substantial

We reused the best hyper-parameters found in the naive experiments. (Section 4.3.3). We still considered the "old" classes as the classes learned at the first step and the incremental part as "new" classes. For example, the room classes at step 2 are "new" at step 2 and stay as "new" in the following steps. We named first context the step 0, the second context step 1 ...

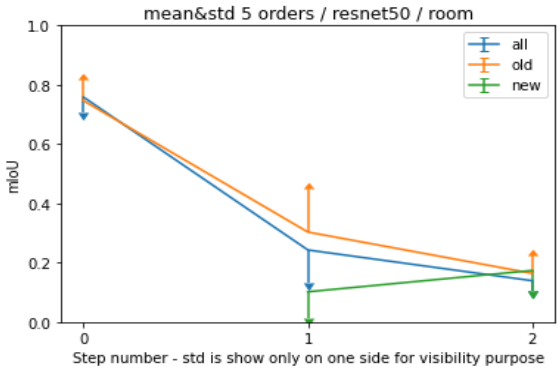
4.4.1 First results

Initial step

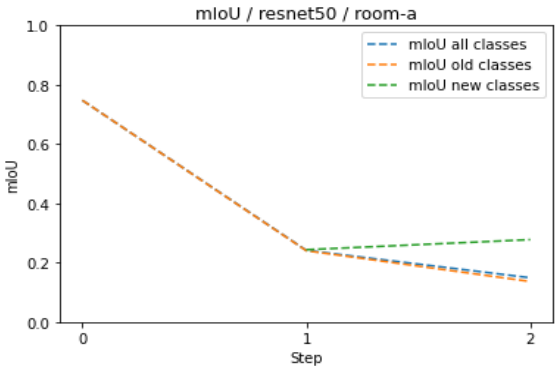
For step 0, quantitative results in Fig.4.11a are slightly lower but still on par with the naive experiments with a mIoU around 80%, and qualitative results in Fig.4.12 show overall good predictions.

We go through the detailed IoU of each order. We notice, for example, in room-a, the first order, which uses sequence one as the test set, that two classes have abysmally low IoU. Fig.4.11c shows that they concern the classes book and bottle.

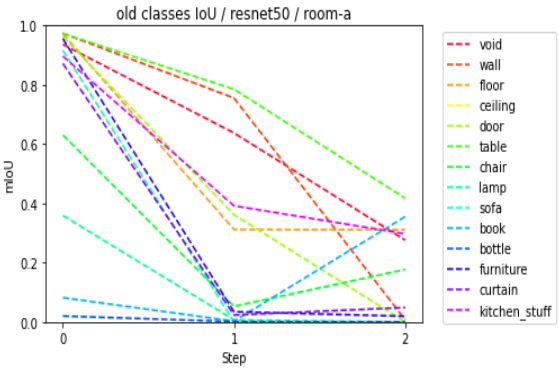
The classes' presence seems on par with others in Fig.4.3a with book and bottle being the classes 17 and 18. However, Fig.4.3b shows that the magnitude of the order of the number of pixels in the test sequence is low, with only around thousands of pixels concerning the class book in the test set, which is composed of 700 images. Due to lack of data and that sequences are biased to different classes, we use cross-validation to validate the results. Fig.4.11d show much higher IoU for book and bottle (with blue colors), showing the necessity of cross-validating with unbalanced sequences set.



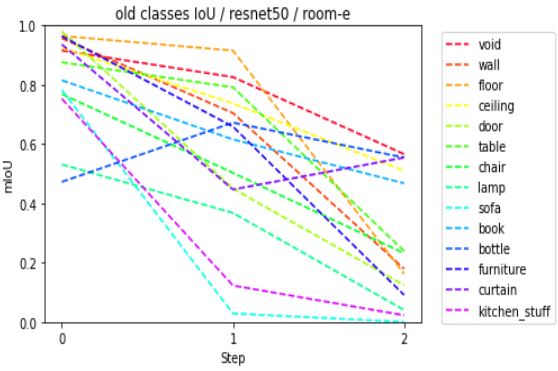
(A) Mean and std of 5 orders without overlap



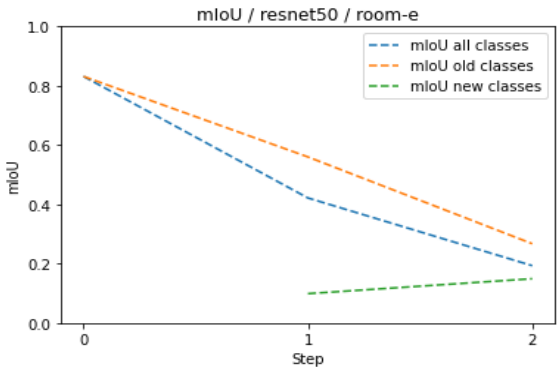
(B) Order 1 without overlap



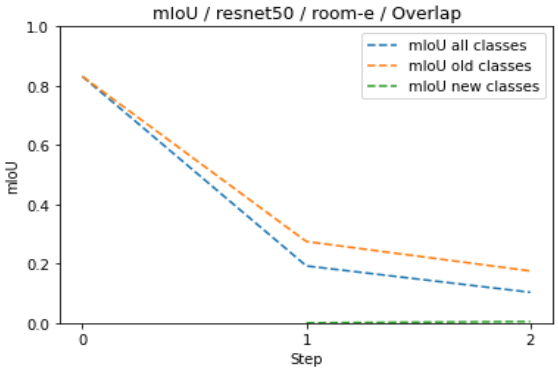
(C) Old classes. Order 1 without overlap



(D) Old classes. Order 5 without overlap



(E) Order 5 without overlap



(F) Order 5 with overlap

FIGURE 4.11: ResNet50. Encoder frozen at incremental steps. 480p images. Context setting

Context incremental steps

We see a high drop in mean IoU after learning the second room with step 1. This drop is much higher than the one from Pascalvoc in Table.4.1 where five new classes are learned jointly as they see a drop of 7% from 77% to 70%. In our case, we see in Fig.4.11a a consequent drop of around 50% with a standard deviation of around 5%. Seeing in Fig.4.11e relatively strong performances, a drop higher than 50% happened in other rooms. Qualitative results in step 1 of Fig.4.12 show heavy catastrophic forgetting but in a particular way. While a whole floor is wrongly predicted, boundaries are more or less rightly estimated. The second context incremental step shows similar behavior, pushing further the catastrophic forgetting.

We think of two main explanations:

- we need to remind of the difference of training setting in the first bullet of the introduction of section 4.4: the network in context incremental learning sees less data than in incremental learning. Using the same example, the class "book" will only be learned in the context of a bedroom but not in the living room or vice-versa
- images are reused in incremental learning setting, at least in some way in the state-of-the-art papers presented in section 3. The network sees the same set of images again in a different form. Indeed, at each step, images have their pixels concerning old classes masked, but they are still selected for training if they contain one pixel of the new class. Therefore, the use of self-training, for example allows a stronger knowledge distillation

To summarize: our setting does not let the network see either classes in different rooms as they are masked between steps or the previous room again as we do not reuse old data in any way.

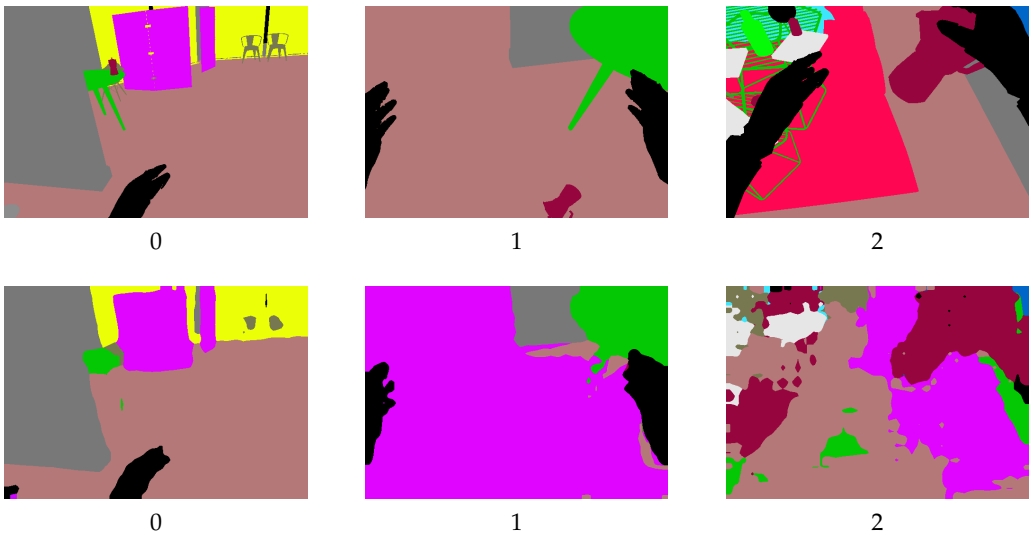
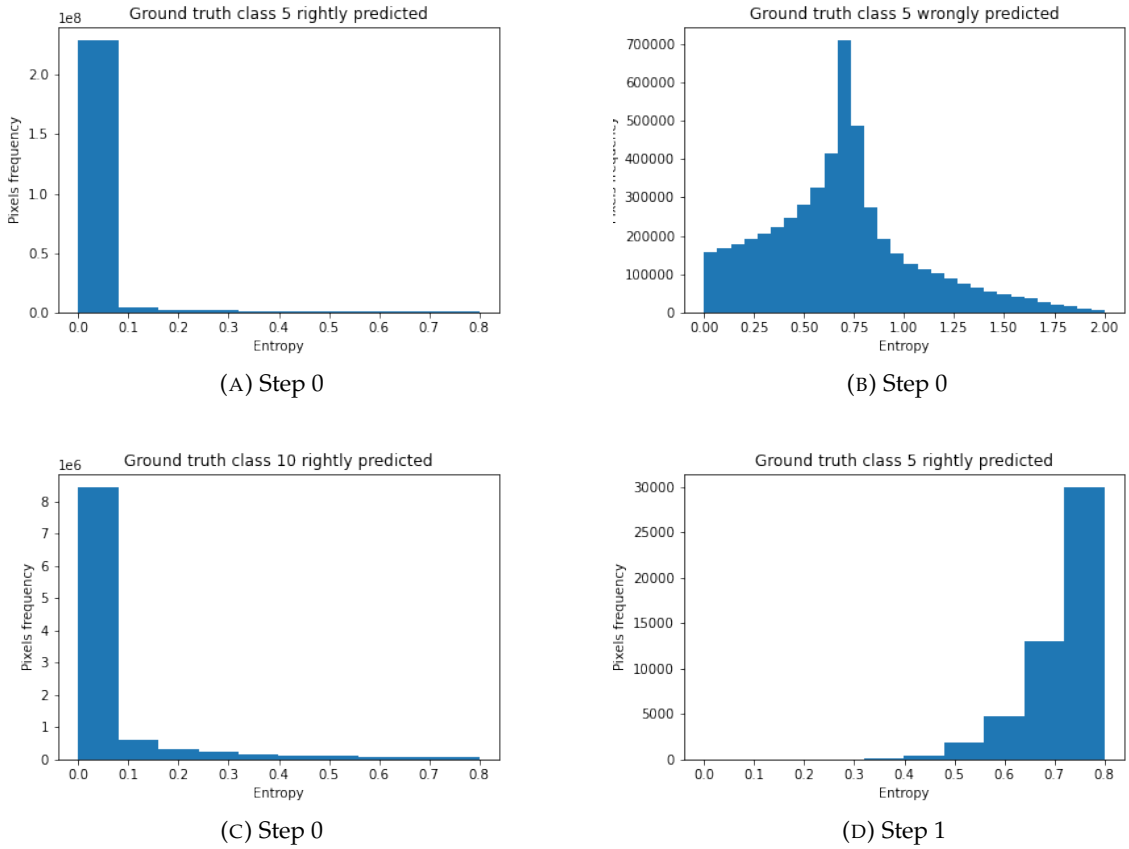


FIGURE 4.12: Evolution of predictions and GT function of step.
Context incremental setting. ResNet50.
First row: ground truths. Second row: predictions

4.4.2 Overlap

Following the high drop after learning a new context, we decided to let the network see old labels in new rooms. Here again, we still considered the "old" classes as the classes learned at the first step and the "new" classes as the incremental ones. For example, an "old" class relearned in future steps is still considered an "old" class.

We rely on the entropies of the network predictions to guide the choice of the labels kept for the new context. Fig.4.13 shows for the pixels of classes 5 and 10 the histogram of entropies of the network predictions in the step 0 training set. Correct and wrong predictions from the network are separated. [18] shows that neural networks tend to show overconfidence which is our case.



(A) Step 0

(B) Step 0

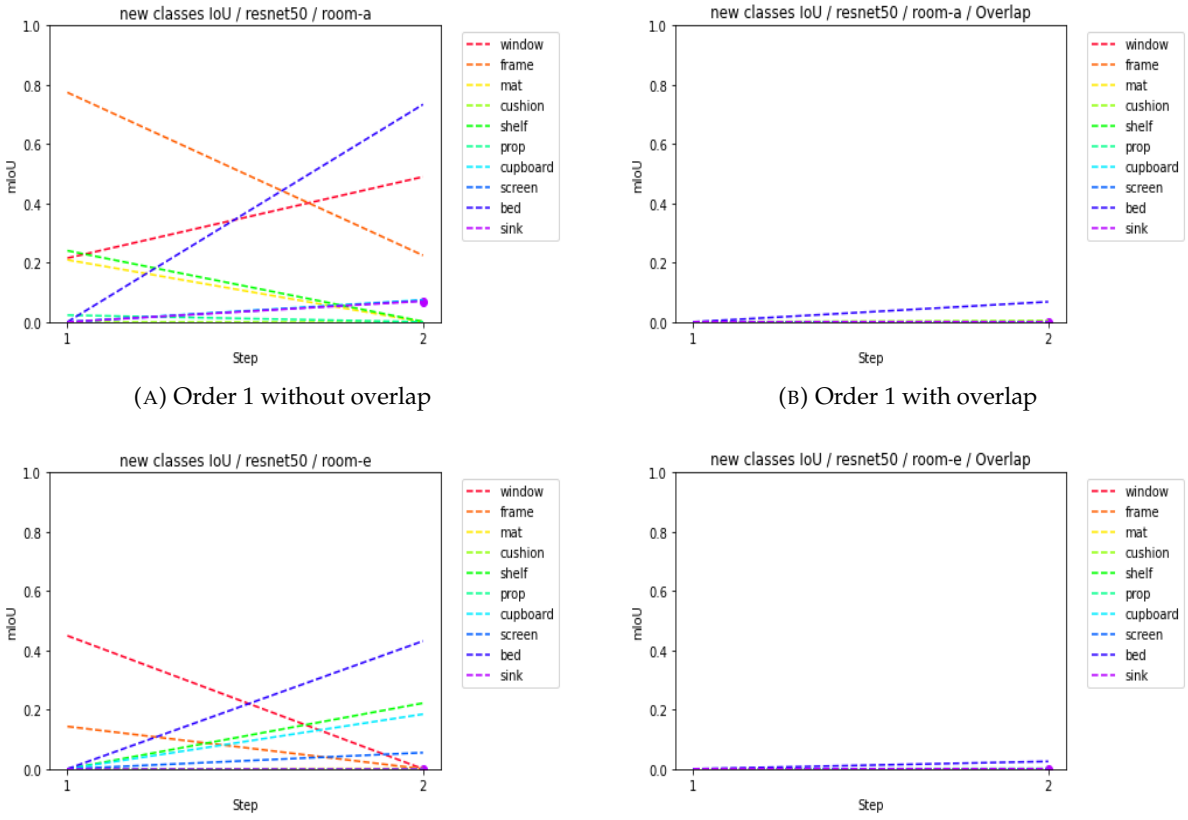
(C) Step 0

(D) Step 1

FIGURE 4.13: Histogram of entropies in the training set

Notably, the histogram of correct predictions seems to follow an exponential distribution. Therefore we can compute his mean parameter $\mathbb{E}[X] = \frac{1}{\lambda}$ and decided to define it as the threshold to keep learning a class or not. After using it to select the old classes to relearn at step 1, it added 5 old classes. We then try two thresholds to select the classes at step 2. One is the same threshold as before and the other a dynamic one as we recomputed the threshold but this time on the step 1 train set. The fixed threshold was too low for the step 1. It results in relearning all old classes. We discarded it and use the dynamic one. We went from a mean entropy of 0.1 on step 0 to 1.4 on step 1, relearning 5 old other classes again though the distribution is not exponential anymore. See Fig.4.13d for the entropies at step 1 for the class 5. Notice the drop in the number of correct predicted pixels compare to the first context.

However, we don't see any mIoU improvements. Results are even worse as we go down from around 40% at step 1 in Fig.4.11e to 20% at step 1 in Fig.4.11f. In any case, not only knowledge distillation doesn't seem to be powerful enough to keep previous knowledge, it also prevents the learning of the new classes as we can see in Fig.4.14 which compares the IoU of the new classes learn with and without relearning some classes. The results is irrevocably one-sided.



(A) Order 1 without overlap

(B) Order 1 with overlap

(C) Order 5 without overlap

(D) Order 5 with overlap

FIGURE 4.14: New classes. ResNet50. Encoder frozen at incremental steps. 480p images. Context setting

Chapter 5

Conclusion and future work

In this thesis, we applied state-of-the-art incremental learning techniques for semantic segmentation in the case of robotics. Due to the recentness of the domain and the dataset, we incrementally raise the difficulty of the setting, going from naive experiments to a more realistic scenario. We tried different methods to enhance results.

Notably, in the naive setting, where one incrementally learns one new class at a time with a random-sampling within all images when splitting the data, we went from ResNet101 to ResNet50, a 22 million parameters difference, without losing significant performance and freezing the encoder lead to a considerable increase of 16% mIoU from 22% to 38%.

In the more challenging context-to-context setting where one learns all the new classes of a room, we encounter new problems specific to the setting. To name a few, the split of sequence for train and test sets can bias the results due to the difference of information between sequences, e.g., some classes are predominant in a sequence but almost non-visible in others. Also, testing is more than ever bias towards the predominance and size of classes due to the sequential nature of the dataset. It pushes us to cross-validate our results. We also get an issue about object class inter context as the setting does not allow us to learn through different contexts.

To mitigate these effects, we decide to let the network continue to learn some of the old classes. We propose a threshold based on entropy for automatic selection. However, adding classes to learn in an incremental step leads to complex learning as knowledge distillation prevents substantial updates. It results in a worst-case scenario in modifying the initial weights without learning the new classes.

From a setting point of view, the final goal would be to train a network on a synthetic dataset from a company and adapt this network to the actual house of a user. This adds the challenge of domain adaptation for future work. Furthermore, where one can expect the company to get labels for his dataset and thus used supervised learning, the user can't be expected to do so. This setting adds the problem of semi-supervised learning, even active learning if we ask the user to label some of the unlabeled data to ease the training.

Nonetheless, this domain is more and more studied. Notably, more and more works have appeared since June 2021. We can expect the emergence of more effective methods to tackle this particular task as many different directions have and can be taken for future works. Unsupervised learning is an interesting take to separate the networks' weights, and while weights selection seems to lack scalability, it could fit the robotics settings.

Bibliography

[1] De Lange, Matthias and Aljundi, Rahaf and Masana, Marc and Parisot, Sarah and Jia, Xu and Leonardis, Ales and Slabaugh, Gregory and Tuytelaars, Tinne. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv:1909.08383*, 2019.

[2] Russ Tedrake. Robot Manipulation: Perception, Planning, and Control (Course Notes for MIT 6.881). *manipulation.csail.mit.edu*, 2021.

[3] Kim, Wonsuk and Seok, Junhee. Indoor semantic segmentation for robot navigating on mobile. *Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2018.

[4] Teso-Fz-Betoño, Daniel and Zulueta, Ekaitz and Sánchez-Chica, Ander and Fernandez-Gamiz, Unai and Saenz-Aguirre, Aitor. Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot. *10.3390/math8050855*, 2020.

[5] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[6] Chen, Liang-Chieh and Papandreou, George and Schroff, Florian and Adam, Hartwig. Rethinking atrous convolution for semantic image segmentation. *Proceedings of the European Conference on Computer Vision*, 2018.

[7] Hinton, Geoffrey and Vinyals, Oriol and Dean, Jeff. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[8] Michieli, Umberto and Zanuttigh, Pietro. Incremental learning techniques for semantic segmentation. *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop*, 2019.

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

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

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

[12] Yu, Lu and Liu, Xialei and van de Weijer, Joost. Self-Training for Class-Incremental Semantic Segmentation. *arXiv:2012.03362*, 2020.

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

[14] Cermelli, Fabio and Mancini, Massimiliano and Xian, Yongqin and Akata, Zeynep and Caputo, Barbara. A few guidelines for incremental few-shot segmentation. *arXiv:2012.01415*, 2020.

[15] Michieli, Umberto and Zanuttigh, Pietro. Continual Semantic Segmentation via Repulsion-Attraction of Sparse and Disentangled Latent Representations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[16] Everingham, Mark and Van Gool, Luc and Williams, Christopher KI and Winn, John and Zisserman, Andrew. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 2010.

[17] Garcia-Garcia, Alberto and Martinez-Gonzalez, Pablo and Oprea, Sergiu and Castro-Vargas, John Alejandro and Orts-Escolano, Sergio and Garcia-Rodriguez, Jose and Jover-Alvarez, Alvaro. The robotrix: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[18] Thulasidasan, Sunil and Chennupati, Gopinath and Bilmes, Jeff and Bhattacharya, Tanmoy and Michalak, Sarah. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *arXiv:1905.11001*, 2019.