Continual Learning for Computer Vision

Arthur Douillard

▶ **To cite this version:**

Arthur Douillard. Continual Learning for Computer Vision. Computer Vision and Pattern Recognition [cs.CV]. Sorbonne Université, 2022. English. NNT : 2022SORUS165 . tel-03872534

HAL Id: tel-03872534
https://theses.hal.science/tel-03872534

Submitted on 25 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SCIENCES SORBONNE UNIVERSITÉ

heuritech

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité **Informatique**

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Continual Learning for Computer Vision

Apprentissage Continu pour la Vision par Ordinateur

Présentée par

Arthur Douillard

Dirigée par

Pr. Matthieu CORD

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Présentée et soutenue publiquement le 13 juin 2022

Devant le jury composé de :

Dr. Diane Larlus *Principal Research Scientist, Naver Labs Europe*	Rapportrice
Pr. David Filliat *Professor, INRIA/ENSTA Paris*	Rapporteur
Pr. Tatiana Tommasi *Associate Professor, Politecnico di Torino*	Examinatrice
Dr. Karteek Alahari *Research Scientist, INRIA Grenoble*	Examinateur
Pr. Kévin Bailly *Associate Professor, Sorbonne Université*	Examinateur
Pr. Matthieu Cord *Professor, Sorbonne Université*	Directeur de thèse
Dr. Thomas Robert *Head of Research, Heuritech*	Invité

Arthur Douillard: *Continual Learnings for Computer Vision,* © 2022

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

I first review the existing methods based on regularization for continual learning. While regularizing a model's probabilities is very efficient to reduce forgetting in large-scale datasets, there are few works considering constraints on intermediate features. I cover in this chapter two contributions aiming to regularize directly the latent space of ConvNet. The first one, PODNet, aims to reduce the drift of spatial statistics between the old and new model, which in effect reduces drastically forgetting of old classes while enabling efficient learning of new classes. I show in a second part a complementary method where we avoid pre-emptively forgetting by allocating locations in the latent space for yet unseen future class.

Then, I describe a recent application of Class Incremental Learning (CIL) to semantic segmentation. I show that the very nature of Continual Semantic Segmentation (CSS) offer new specific challenges, namely forgetting on large images and a background shift. We tackle the first problem by extending our distillation loss introduced in the previous chapter to multi-scales. The second problem is solved by an efficient pseudo-labeling strategy. Finally, we consider the common rehearsal learning, but applied this time to CSS. I show that it cannot be used naively because of memory complexity and design a light-weight rehearsal that is even more efficient.

Finally, I consider a completely different approach to continual learning: dynamic networks where the parameters are extended during training to adapt to new tasks. Previous works on this domain are hard to train and often suffer from parameter count explosion. For the first time in continual computer vision, we propose to use the Transformer architecture: the model dimension mostly fixed and shared across tasks, except for an expansion of learned task tokens. With an encoder/decoder strategy where the decoder forward is specialized by a task token, we show state-of-the-art robustness to forgetting while our memory and computational complexities barely grow.

RÉSUMÉ

Depuis le début des années 2010 la recherche en apprentissage automatique a orienté son attention vers les efficaces réseaux de neurones profonds. Plus particulièrement, toutes les tâches de vision par ordinateur utilisent désormais des réseaux convolutionnels. Ces modèles apprennent à détecter des motifs d'abord simples (contours, textures) puis de plus en plus complexes jusqu'à apprendre le concept d'objets en particulier.

Malgré les grandes avancées dans le domaine des réseaux de neurones profonds, un problème important subsiste : comment apprendre une quantité croissante de concepts, à la manière d'un élève durant sa scolarité, sans oublier les précédentes connaissances. Ce problème d'apprentissage continu est complexe : si non traité, les réseaux de neurones oublient catastrophiquement. L'objectif de cette thèse était donc de résoudre de ce problème.

J'ai pu dans un premier temps développer plusieurs méthodes pour forcer un comportement similaire entre la version du modèle ayant appris de nouveaux concepts et sa précédente itération. Contrairement au reste de la littérature, qui imposait des contraintes sur le comportement final du modèle, je me suis intéressé aux représentations internes.

Dans un second temps, j'ai considéré l'apprentissage continu pour la tâche de segmentation sémantique. Cette tâche complexe possède des problèmes inédits dans un contexte continu en plus de l'oubli catastrophique. J'ai pu proposer plusieurs approches complémentaires pour les résoudre. Plus précisément : une nouvelle méthode de contraintes, une technique de pseudo-annotations et une manière efficace de révisions d'objets.

Et enfin, dans un troisième et dernier temps, je m'intéresse aux réseaux de neurones dynamiques, pouvant créer de nouveaux neurones à travers leur existence pour résoudre un nombre croissant de tâche. Les méthodes précédentes grandissent avec peu de contrôles, résultant en des modèles extrêmement lourd, et souvent aussi lents. Donc, en m'inspirant des récents *transformers*, j'ai conçu une stratégie dynamique avec un coût pratiquement nul, mais ayant malgré tout des performances à l'état-de-l'art.

REMERCIEMENTS

Je souhaite remercier tous ceux qui m'ont aidé à réaliser cette thèse.

Dans un premier temps, je dois Charles Ollion, et Tony Pinville : Matthieu pour m'avoir supervisé tout au long de ces trois années de thèse, pour m'avoir fait découvrir le monde de la recherche et avoir supporté mes avis tétus. J'ai beaucoup appris et j'en ressors grandis. Charles et Tony pour m'avoir fait confiance et permis de faire ce doctorat avec Heuritech. Et plus particulièrement Charles, pour avoir été le premier à m'initier au Deep Learning.

Je remercie aussi le jury de cette thèse pour avoir relu mon manuscrit et être m'accorder leur temps pour ma soutenance.

Je veux aussi remercier tous mes collègues dont leur discussions ont toujours été enrichissante aussi bien sur le plan du travail que de l'humain. Parmi les *Chordettes* : Corentin Dancette, Rémi Cadene, Alexandre Ramé, Antoine Saporta, Guillaume Couairon, Asya Grechka, Yifu Chen et Rémy Sun. Chez Heuritech : Thomas Robert (pour sa gentillesse infinie), Emilien Garreau, Antoine Hoorelbeke, Paul Morel, Florent Mercier, et Oscar Bouvier. Et enfin aussi Timothée Lesort, Fabio Cermelli, Eduardo Valle et Arnaud Dapogny pour leur collaboration. Plus largement je remercie aussi l'ensemble de l'équipe du MLIA et Heuritech pour leur support et la bonne ambiance qu'ils ont apporté.

Enfin, je veux remercier mes proches. A Jordan et Camille qui m'ont tant soutenu cette dernière année. A David, Hugo, Alexandre, Thibault et Marie-Anne, Benjamin, Florent, Sarasvati, Charlotte, et tous ceux avec qui j'ai pu passer de bons moments. A ma famille, Julie, mon père, ma mère, mes grand-parents pour leur soutien durant ces 26 ans de vie. J'insiste particulièrement sur l'éducation et la passion des sciences que m'ont inculqués ma mère et ma grand-mère, qui a été essentiel.

ACRONYMS

AI	Artificial Intelligence
ConvNet	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
ML	Machine Learning
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
LSC	Local Similarity Classifier
POD	Pooled Output Distillation
PODNet	Pooled Output Distillation Network
CIL	Class Incremental Learning
CL	Continual Learning
ZSL	ZeroShot-Learning
KL	Kullback-Leiber divergence
KD	Knowledge Distillation
NME	Nearest Mean Examplar
SVM	Support-Vectors Machine
CSS	Continual Semantic Segmentation
NLP	Natural Language Processing
AL	Active Learning
KD	Knowledge Distillation
GAP	Global Average Pooling
CML	Continual-Meta Learning
MCL	Meta-Continual Learning
OoD	Out-of-Distribution
RL	Reinforcement Learning

GAN	Generative Adversarial Network
mIoU	mean Intersection-over-Union
MHSA	Multi-Head Self-Attention

NOTATIONS

Total number of tasks	T
Current task	t
Classes of the current task	$\mathcal{C}^t$
Classes of the previous tasks	$\mathcal{C}^{1:t-1}$
Classes of the future tasks	$\mathcal{C}^{t+1:T}$
Cardinality of a set of classes	$\mathcal{N}^t = \text{card}(\mathcal{C}^t)$
Image at task t	$\mathbf{x}^t$
Ground-truth label at task t	y^t
Ground-truth segmentation map at task t	$\mathbf{y}^t$
Feature extractor at task t	$f^t(\cdot)$
Classifier at task t	$g^t(\cdot)$
Learnable parameters at task t	θ^t
Loss function	$\mathcal{L}$
Predicted label	$\hat{y}^t$
Predicted segmentation map	$\hat{\mathbf{y}}^t$
Intermediary spatial features at level l	$\mathbf{h}_\ell^t = f_\ell^t(\cdot),\ \ell \in \{1, \dots, L\}$
Final embeddings post-Global Average Pooling	$\mathbf{h}^t = f^t(\mathbf{x})$
c^{th} channel of a spatial tensor $\mathbf{x}$	$\mathbf{x}[c, :, :]$
w^{th} column of a spatial tensor $\mathbf{x}$	$\mathbf{x}[:, w, :]$
h^{th} row of a spatial tensor $\mathbf{x}$	$\mathbf{x}[:, :, h]$

CHAPTER 1

INTRODUCTION

> I believe that at the end of the century, the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.
>
> *Alan Turing*

In this thesis introduction, using layman terms, we describe Artificial Intelligence, and, in particular, one of its instances: Deep Learning. Then, we lay out the challenges of this thesis and our contributions.

The idea of thinking machines began in the previous century, from Karel Çapek's invention of the "*robot*" to the 1956's Dartmouth workshop passing by Turing & von Neumann's reflections. Despite suffering from multiple "AI winters" filled with disappointments and criticisms, Turing's prediction on the rising importance of Artificial Intelligence (AI) proved to be right as the first and the second decades of the XXI century saw the advent of respectively Machine Learning (ML) (Bishop 2006) and Deep Learning (DL) (I. Goodfellow et al. 2016), two major subfields of AI, related to statistical learning theories.

Providing **a definition of AI** is difficult, but its foremost domains, ML and DL, can be defined as statistical algorithms that can improve automatically through experience and the use of data. These methods are already ubiquitous: speech recognition enabling us to control devices remotely (Amodei et al. 2016), recommender systems proposing movies according to our taste (Töscher and Jahrer 2009), automatic translation (Vaswani et al. 2017), face recognition (Schroff et al. 2015), autonomous driving (P. Sun et al. 2020), *etc*. Less known but still useful applications comprise accelerated physics simulation (Breen et al. 2020), protein folding prediction (Jumper et al. 2021), molecule toxicity estimation (Advancing Translational Sciences 2014), data center cooling system (Evans et al. 2016), control of the magnetic coils of a nuclear fusion reactor (Degrave et al. 2022), *etc*.

AI is increasingly more important in our daily lives with some applications raising **ethical concerns**: face recognition biased towards some populations (Grother et al. 2019), loan grants (Anglekar et al. 2021), medical diagnosis (Larrazabal et al. 2020), justice advice (Russel 2020), biased chatbots (Sheng et al. 2019), *etc*. Therefore, we must pay a particular interest in the potential impact of this new technology. Towards this goal, people from diverse backgrounds must participate in the creation of such technology, and standardization bodies (Tommasi et al. 2021) should advise what types of AI systems can be used in which scenarios (Gebru 2019).

1.1 PhD Thesis Context

I now contextualize my thesis with relation to my sponsor, how it influenced our research, and what challenges we have aimed to tackle.

Heuritech This thesis was sponsored by the Parisian startup Heuritech [1] as a *CIFRE PhD*. The company analyzes social networks such as Instagram and Weibo, recognizes the clothes in pictures, estimates volumes of fine-grained types of garments, and finally forecasts future trends. The company's Computer Vision (CV) models must recognize an ever-growing number of entities from features (*e.g.* knitted, blue color, short cut) to brand models (*e.g.* Nike Air Max, Adidas Stan Smith, Puma Suede). This requirement leads to two problems: **(1)** the time spent to re-train a model is growing linearly, and **(2)** learning a new entity can incur a performance loss on previously learned entities.

Continual Learning is a field that emerged in the 1990s but saw renewed interest only very recently around the second half of the 2010s. The goal is to deal with datasets that evolve through time. This evolution can take many forms, including adding new entities to predict in a classification task (*e.g.* learning sneaker brands, then high-heel brands) or adding samples from new sources (*e.g.* commercial photoshoot then images from social media). Unfortunately, current State-of-the-Art models struggle to learn continually from new data without losing performance on previously seen data. This loss is so critical that the literature nicknamed it "*catastrophic forgetting*" (Robins 1995; French 1999). Multiple methods can reduce this forgetting, including rehearsal and constraints. Rehearsal involves reviewing previously learned knowledge, as a human student would *rehearse* the last semester's course. However, this rehearsal is often limited in order to reduce computational cost and because past data may not always be available for a variety

1. https://heuritech.com

of reasons, including privacy. On the other hand, constraints enforce the model to keep a similar *behavior* as it learns new concepts, but defining the optimal constraint is not trivial.

1.2 Contributions

This PhD thesis is structured around solving catastrophic forgetting in continual settings. We considered various specific situations and approaches declined in several chapters:

- Chapter 3: VISUAL FEATURE-BASED REGULARIZATIONS
  We first propose strategies to tackle catastrophic forgetting in deep neural neworks in the context of the image classification. Our main goal is to constrain the evolution of the network parameters during the continual training to be relatively rigid while also avoiding completely freezing the network. The work in this chapter has led to two conference publications:

  - Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle (2020). "PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*

  - Arthur Douillard, Eduardo Valle, Charles Ollion, and Matthieu Cord (2021c). "Insights from the Future for Continual Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*

- Chapter 4: CONTINUAL SEMANTIC SEGMENTATION
  Then, we extend our focus to the task of semantic segmentation. We show that this context brings new challenges, and we propose multiple complementary approaches to tackle them. The work in this chapter has led to one conference publication and one submission to a journal:

  - Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021a). "PLOP: Learning without Forgetting for Continual Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

  - Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021b). "Tackling Catastrophic Forgetting and Background Shift in Continual Semantic Segmentation". In: *Under review at TPAMI*

- Chapter 5: DYNAMIC STRATEGY WITH TRANSFORMERS
  Finally, in our last chapter, we consider the impact of the neural network architecture for continual learning and in particular propose using the recent Transformer. The work in this chapter has led to a conference publication:

  - Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord (2022). "DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

CHAPTER 2

RELATED WORK

In this chapter, we detail the related work needed to read this thesis. We first briefly explain the learning procedure in Deep Learning (DL) and how the data are structured. Then, we describe the main DL applications and architectures in Computer Vision (CV). Finally, we introduce the main topic of this thesis — Continual Learning— and showcase the challenges, benchmarks, and methods of this domain. The notations introduced in this chapter and through the thesis are listed in the Notations section.

2.1 Neural Network Learning

Neural Networks are based on the statistical learning theory (Vapnik 1999). In the *supervised setting*, the goal of a neural network is to learn the best mapping function $f : \mathcal{X} \to \mathcal{Y}$ between an input space $\mathcal{X}$ and an output space $\mathcal{Y}$. We consider a *loss* function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ that measures the disagreement between a ground-truth label y and the network's prediction $\hat{\mathbf{y}} = f(\mathbf{x})$. In the context of image classification, $\mathbf{x}$ is the image pixels, y is the *class* label of the object present in the image, and $\hat{\mathbf{y}}$ is a vector of probabilities, with one probability per class the model has to predict. The class with the highest predicted probability/confidence is chosen as the prediction. Given a *training dataset* $\mathbb{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, training a neural network consists in finding the set of parameters θ_* which minimizes the loss function:

$$\theta_* = \text{argmin}_\theta \left\{ \frac{1}{|\mathbb{D}|} \sum_{(\mathbf{x},y)\in\mathbb{D}} \mathcal{L}(f_\theta(\mathbf{x}), y) \right\} . \tag{2.1}$$

However, the optimal set of parameters θ_* minimizing the loss on the training set, can fail to predict the correct label on the testing set. Therefore, to avoid this *overfitting* phenomenon, regularization functions that limit the parameters space are used as additional constraints in addition to the classification loss $\mathcal{L}$.

In practice, the function f_θ learned by a neural network is composed of a succession of linear transformations and non-linear *activation* functions. For example, a

simple neural network, called Multi-Layer Perceptron (MLP), here with two layers, could be defined as:

$$\hat{\mathbf{y}} = f_\theta(\mathbf{x}) = \text{softmax}(\mathbf{W}_o\sigma(\mathbf{W}_h\mathbf{x} + \mathbf{b}_h) + \mathbf{b}_o)) \,, \tag{2.2}$$

with $\mathbf{W}_h \in \mathbb{R}^{H\times D}$, $\mathbf{b}_h \in \mathbb{R}^H$, $\mathbf{W}_o \in \mathbb{R}^{C\times H}$, $\mathbf{b}_o \in \mathbb{R}^C$ being the parameters θ of the network. σ is a hidden non-linear activation, often a Rectified Linear Unit (ReLU) ($\text{ReLU}(x) = \max(0, x)$), and $\hat{y} = \text{softmax}(\tilde{\mathbf{y}}) = {}^{e^{\tilde{\mathbf{y}}}}/_{\sum_i e^{\tilde{\mathbf{y}}_i}}$ is the final non-linear activation.

Likewise, in practice for image classification, the loss is often the categorical cross-entropy:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_c y_c \log \hat{y}_c \,, \tag{2.3}$$

with $\mathbf{y}$ a one-hot vector of the labels. Finally, to optimize the parameters of the neural network (such as Equation 2.1), we often use the mini-batch Stochastic Gradient Descent (SGD) algorithm or a variation thereof:

Algorithm 2.1 Procedure to optimize a neural network with gradient descent.

input: a dataset $\mathbb{D}$ with pairs of $(\mathbf{x}_i, \mathbf{y}_i)$
input: a loss function $\mathcal{L}$
input: a model function f_θ with θ the learned parameters
input: a learning rate η and a batch size b

1: **while** stopping criterion not satisfied **do**
2: $(\mathbf{x}, \mathbf{y}) \leftarrow$ sample mini-batch of size b from $\mathbb{D}$
3: Forward pass: $\hat{\mathbf{y}} \leftarrow f_\theta(\mathbf{x})$
4: Compute loss: $\mathcal{L} \leftarrow \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
5: Compute the gradients: $\delta \leftarrow \nabla_\theta \mathcal{L}$
6: Update all parameters: $\theta \leftarrow \theta - \eta\delta$
7: **end while**

The learning rate η controls the step size when updating the parameters in the direction of the gradient. The batch size b defines the number of images seen during one update. The *backpropagation* algorithm (Rumelhart et al. 1986) computes the gradients and the update of the parameters. In the case of image classification and segmentation, the main topics explored in this thesis, we discern the *feature extractor* from the *classifier* in the neural network. The former transforms the input space so that the latter linearly discriminates classes. From now on, the feature extractor will be denoted by f, and the classifier by g (see the notations). The model is then $g \circ f$.

The neural network in Equation 2.2 can be defined as *shallow* with only one hidden layer. By extension, an important success of later neural networks came

with training *deep* neural network with multiple hidden layers. More generally in this thesis, we will refer to them as Deep Learning (DL) models. The majority of DL models, in Computer Vision (CV), in Natural Language Processing (NLP), or even in Audio are based on the same structure (linear transformations and non-linear activations) and are trained with a form of gradient descent.

2.2 Deep Architectures for Computer Vision

A common type of architectures for computer vision is the Convolutional Neural Network (ConvNet). First defined by Fukushima (1980) and then trained with backpropagation by LeCun et al. (1999), it is a neural network architecture whose linear operators are *convolutions*. While handcrafted convolutions (Lowe 1999) rely on well-defined features, a ConvNet learns the convolution kernels to detect more complex patterns required to minimize the classification loss. In 2012, thanks to a large dataset and more efficient code working on Graphics Processing Units (GPUs) allowing training larger networks, Krizhevsky et al. (2012) won the ILSVC competition (Russakovsky et al. 2015) where they had to classify a large dataset —ImageNet— made of 1.28M training images and 1000 classes. From that point forward, multiple improvements were made to ConvNets (Ioffe and Szegedy 2015; K. He et al. 2016) and these methods have been applied not only to classification but also object detection (Ren et al. 2015), semantic segmentation (L.-C. Chen et al. 2018a), visual question answering (Antol et al. 2015; Ben-younes et al. 2017), image generation (I. J. Goodfellow et al. 2014), image editing (Grechka et al. 2021; Couairon et al. 2022), *etc*.

Most ConvNets follow a similar structure with blocks of convolutions and pooling. Often, the final spatial features are flattened by Global Average Pooling (GAP) and fed to a linear classifier predicting the classes probabilities. Figure 2.1 illustrates this general architecture.

Architectures: The 2010's decade saw major improvements to ConvNets, both in their architecture and in their training procedure. Srivastava et al. (2015) and K. He et al. (2016) propose residual connections between blocks likewise: $\mathbf{y} = \mathbf{x} + \sigma(\mathrm{Conv}(\mathbf{x}))$. By reducing the *vanishing gradient* problem (Hochreiter et al. 2001), it allows training deeper networks without accuracy degradation. This type of connection is now quasi-ubiquitous in all DL based architectures. Other architecture changes include using convolutions of different kernel sizes in parallel as in Inception (Szegedy et al. 2015), enabling a multi-scales view of the features. These architectures are depicted in Figure 2.2.

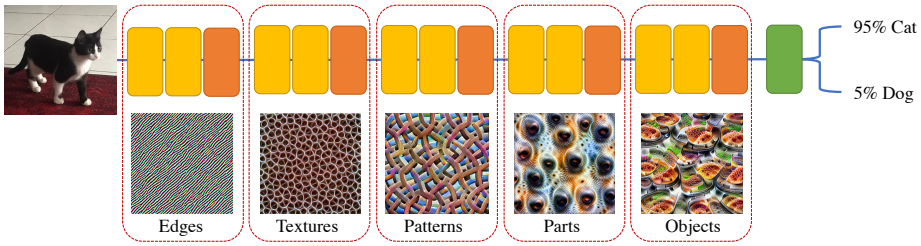


Figure 2.1. – **A Convolutional Neural Networks** extracts more complex patterns through its succession of convolutions. Yellow blocks are convolutions, **orange** blocks are poolings, and the unique **green** block is the classifier. Given an image, the ConvNet can assign to each possible class a probability, all summing to 1. The detected shapes grow in complexity with the depth of the network, from crude edges and textures, to objects. Detected patterns taken from Olah et al. (2017).

Regularizations: While the architecture design has an impact on the model performance, the training procedure has been shown to be essential in order to reach state-of-the-art results (Wightman et al. 2021). Modern procedures include improvement of SGD with an adaptive learning rate per layer such as Adam (Kingma and Ba 2014), an improved learning rate scheduling, stronger data augmentations (Mü and Hutter 2021; Hongyi Zhang et al. 2018; Zhong et al. 2017), and regularizations such as Dropout (Gal and Ghahramani 2016) and stochastic depth (G. Huang et al. 2016).

Transformers: While convolution-based neural networks dominated Computer Vision (CV) in the 2010's decade, in the last few years, the transformer architecture gained interest: it was originally designed for machine translation in NLP (Vaswani et al. 2017) with an encoder/decoder structure and a *self-attention* block between the words embeddings of a sentence. Each word is embedded into a high-dimensional vector named a "*token*". The self-attention operation of a transformer has a quadratic complexity w.r.t. the number of tokens: in NLP, it is manageable given a small sentence. However, when applying a transformer on images and considering each pixel as a token, the complexity is too important. Dosovitskiy et al. (2021), based on the encoder structure of BERT (Devlin et al. 2018), propose instead to consider a patch of pixels as a token, reducing significantly the number of tokens.

This architecture is illustrated in Figure 2.3: we concatenate a special learned token, called "*class token*", to the patch tokens. Moreover, we also sum these tokens with a position embedding to facilitate learning tokens position. Then, successive transformer blocks process all the tokens. Each block is made of LayerNorms

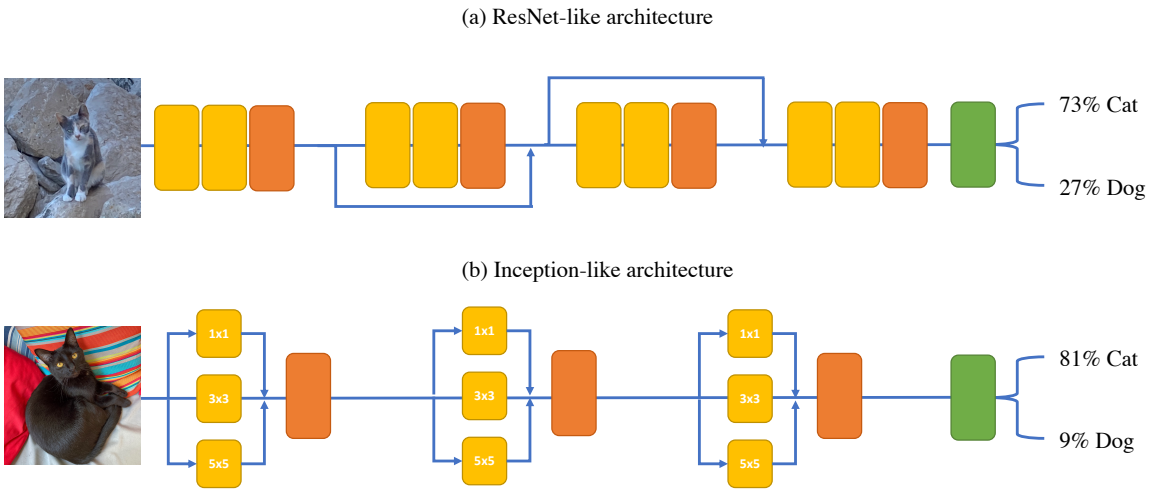


Figure 2.2. – **Different ConvNet architectures: (a)** illustrates a ResNet-like architecture where there are residual connections between blocks. Used by the vast majority of modern architectures, these connections reduce the vanishing gradient problem and thus enabling the training of deeper networks. **(b)** showcases an Inception-like architecture where at the same level convolutions with different kernel sizes are used. Each detects patterns of different scales.

(Ba et al. 2016), a self-attention block, a MLP, and residual connections. Thus, the self-attention block is:

$$\begin{aligned} Q &= W_q \mathbf{x}\,, \\ K &= W_k \mathbf{x}\,, \\ V &= W_v \mathbf{x}\,, \\ A &= \text{Softmax}\left(Q \cdot K^T / \sqrt{D/h}\right)\,, \\ O &= W_o A V + b_o\,, \end{aligned} \tag{2.4}$$

$\mathbf{x}$ are the N patch tokens and the class token, of shape (N, D), D being the embedding dimension. The patch tokens are linearly transformed three times in parallel into a **Q**uery, **K**ey, and **V**alue. An attention matrix A of shape (N, N) is computed from the query Q and the key K. Its i^{th} row contains the similarity between the i^{th} token with all other tokens. Finally, the multiplication between the attention matrix and the value matrix V averages all tokens according to their similarities. To extend the self-attention to its multi-heads variation (Multi-Head Self-Attention (MHSA)), we use several Query/Key/Value transformations, each used in a self-attention on a fraction of the embedding dimension.

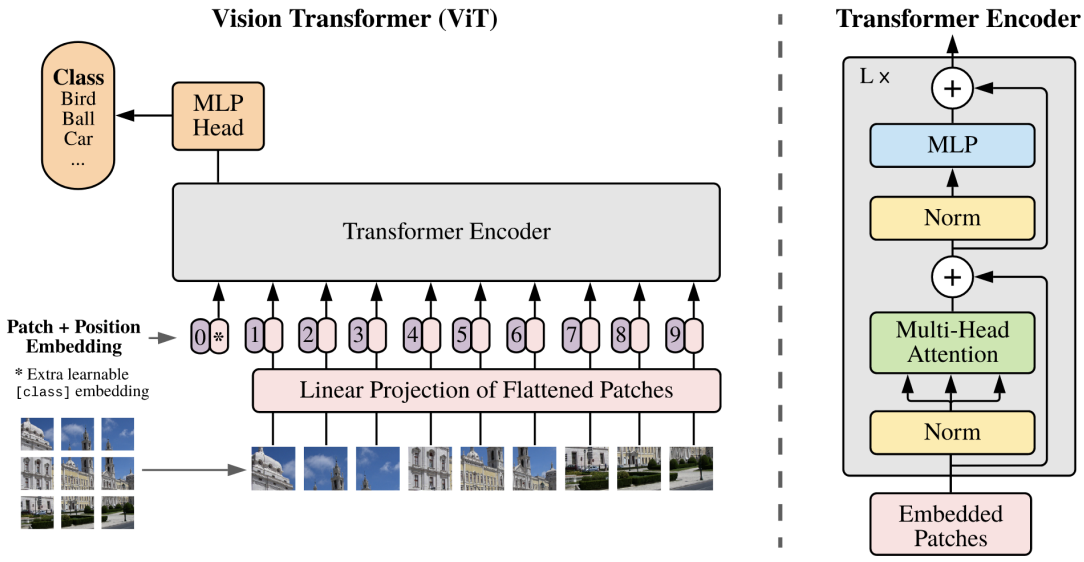


Figure 2.3. – **The Vision Transformer (ViT):** the image is cropped without overlap and projected using a convolution whose stride equals the kernel size. A learned "class token" is concatenated to the resulting patches, which are then summed with a position embedding. The encoder is made of multiple transformer blocks. Each block is made of two (Layer) Norm layers, a MLP with a single hidden layer, and the Multi-Head Attention block. Finally, only the special "class token" is used at the end, and fed to a classifier (here denoted as the "MLP Head"). Image from Dosovitskiy et al. (2021).

ViT (Dosovitskiy et al. 2021) is the seminal paper on vision transformer. However, its training was difficult and needed a large — and private — dataset called JFT300M. Later works, including DeiT (Touvron et al. 2021a) proposed an efficient optimization procedure enabling the training of transformers on smaller datasets such as ImageNet (Russakovsky et al. 2015). Finally, multiple works improved also the architecture itself, including CaiT (Touvron et al. 2021b), ConViT (d'Ascoli et al. 2021), and Swin (Z. Liu et al. 2021). Notably, PerceiverIO (Jaegle et al. 2021) proposed a general architecture whose output is adapted to different modalities using specific learned tokens, and whose computation is reduced using a small number of latent tokens.

2.3 Continual Learning

Usually, when training a ConvNet, we assume the dataset is immutable and *i.i.d.*: no new images nor new classes will be learned. The knowledge acquired on one dataset A can be *transferred* to another dataset B with different classes using **transfer learning** (Razavian et al. 2014). Then, the new model may be efficient on the classes of dataset B but cannot predict anymore the classes of dataset A.

Continual Learning aims to learn a continually changing dataset without forgetting the previous knowledge. The distribution of the dataset continually changes: *e.g.* at each time-step $t \in \{1, ..., T\}$, new classes or new samples from potentially new domains are added to the training dataset (Lomonaco and Maltoni 2017). We usually assume the test dataset evolves similarly. Multiple distribution drifts exist in Continual Learning (Moreron-Torresa et al. 2012; Lesort et al. 2021), and they have been called under various names in the literature. Given an input sample x and its ground-truth y (a label in image classification, or a segmentation map in semantic segmentation), the major drifts are:

- **Covariate drift**: when $p(x)$ changes, it happens with the introduction of new domains (Volpi et al. 2021).
- **Prior drift**: when $p(y)$ changes; Class Incremental Learning (CIL) happens with this kind of drift.
- **Conceptual drift**: when $p(y|x)$ changes. Seldom covered in the literature, it can be found in Continual Semantic Segmentation (CSS).

Learning an ever-growing dataset is possible by training from scratch a new model on the union of past and new data. However, for multiple reasons like privacy concerns, limited time, or small storage capacity (Vásquez et al. 2017), there is a restriction on the amount of previous data that can be kept. In the extreme case, a model only has access to new data but not old data. Evidently, if the initialization of the parameters is random, the model cannot predict the past data distribution. However, the new model parameters can be initialized using the previous parameters ($\theta^t := \theta_*^{t-1}$). Remark, from Equation 2.1, that the optimization procedure of the old $(t-1)$ and new (t) model is different. Given the average loss $L(f_\theta, \mathbb{D}) = {}^1/_{|\mathbb{D}|} \sum_{(\mathbf{x},y)\in\mathbb{D}} \mathcal{L}(f_\theta(\mathbf{x}), y)$:

$$\begin{aligned} \theta_*^{t-1} &= \text{argmin}_{\theta^{t-1}} \left\{ L(f_{\theta^{t-1}}, \mathbb{D}^{t-1}) \right\} , \\ \theta_*^{t} &= \text{argmin}_{\theta^{t}} \left\{ L(f_{\theta^{t}}, \mathbb{D}^{t}) \right\} , \end{aligned} \tag{2.5}$$

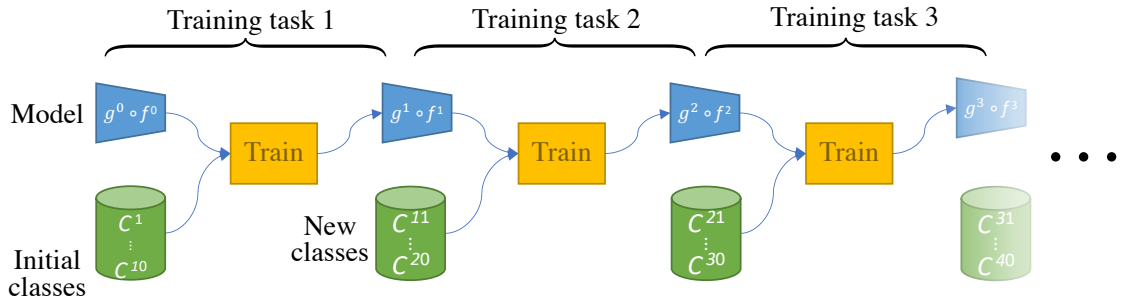


Figure 2.4. – **Training protocol for incremental learning**. At each training task we learn a new set of classes, and the model must retain knowledge about *all* classes. The model is allowed a *limited* memory of samples of old classes.

where the loss is minimized respectively with respect to the old ($\mathbb{D}^{t-1}$) and new dataset ($\mathbb{D}^t$). Evidently, the optimal parameters θ_* are different for the task t and $t-1$. This difference results in what we call a *forgetting*: θ_*^t is optimal for the new task t but is suboptimal for the task $t-1$, therefore performance on the previous task may be degraded ($L(f_{\theta^t}, \mathbb{D}^{t-1}) \gg L(f_{\theta^{t-1}}, \mathbb{D}^{t-1})$). This performance drop is actually so important that the literature names it a **Catastrophic Forgetting** (Robins 1995). It is particularly critical in the context of image classification where each new task brings new classes to be classified as described below.

Class-Incremental Example More concretely, a common benchmark in Class Incremental Learning (CIL) is learning the image classification CIFAR100 dataset (Krizhevsky and Geoffrey Hinton 2009), made of 100 classes, in multiple steps, each made of several new classes. During the first step $t = 1$, the model $g^1 \circ f^1$ learns the first 10 classes $\mathcal{C}^{t=1}$. During the second step $t = 2$, the model $g^2 \circ f^2$ is initialized from the learned parameters of the previous model $g^1 \circ f^1$ and learns the next 10 classes $\mathcal{C}^{t=2}$., and so on, until all 100 classes are learned at the last step $t = T$. After each step, we evaluate the model on the testing data of all seen classes $\mathcal{C}^{1:t}$. Figure 2.4 illustrates such continual protocol, and Figure 2.5 depicts the continual performance of a both trained from scratch on all data for each task and a continual model finetuned only on new data. The continual model over-predicts new classes (Davidson and Mozer 2020) and has a low overall accuracy due to a forgetting of old classes. This clearly illustrates how important is the catastrophic forgetting. While in this thesis we mainly tackle Class Incremental Learning (CIL) benchmarks, we describe other continual benchmarks in detail in the appendix (Section A.1).

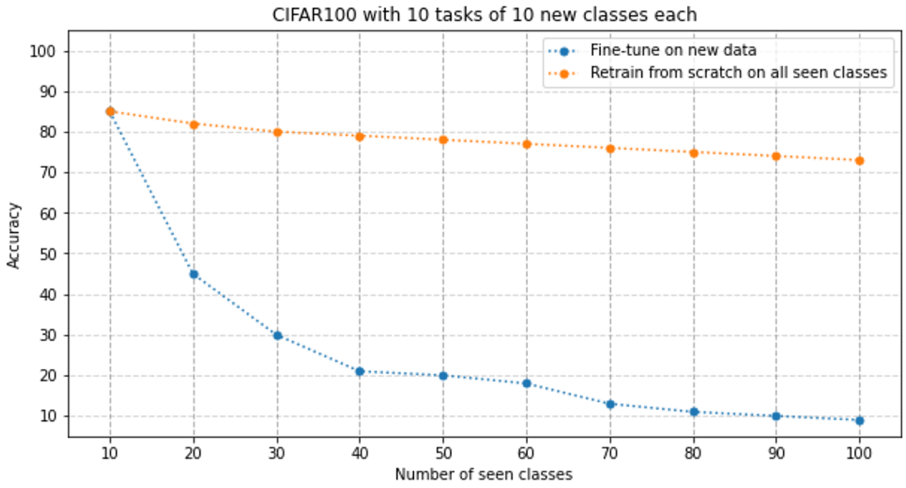


Figure 2.5. – **Illustration of the forgetting in Class-Incremental Learning**. orange line displays the accuracy of a model which is re-trained from scratch at each step on all previous training data $\mathcal{C}^{1:t}$. This model, usually called Joint, is considered as a reasonable upper bound. On the other hand, the blue line is a model finetuned solely on new classes $\mathcal{C}^t$ without access to previous classes $\mathcal{C}^{1:t-1}$. The gap between both models illustrates the catastrophic forgetting.

Single-Head vs Multi-Heads are the two main evaluation settings in Continual Learning (Chaudhry et al. 2018). In the former setting, a model has to classify samples among all seen classes $\mathcal{C}^{1:t}$, that could have been learned from any of the seen steps. On the other hand, in a multi-heads setting, a model knows at test-time the step/task identifier i of the samples. Thus, it only has to classify among the limited number of classes brought by a step ($\mathcal{C}^i$). Multi-Heads evaluation is closely related to multi-tasks learning. During this thesis, we focus on the Single-Head evaluation because it is more realistic as it is rarely possible to know from which step a sample comes from in a real-life setting. This setting is also more challenging because a unique classifier must discriminate among all tasks' classes instead of having a different classifier per task (Lesort et al. 2019b).

Metrics Multiple metrics exist in Continual Learning: the most common are the **final accuracy** and **average incremental accuracy**. The former measures the performance of the model on all tasks at the last step, the latter measures the average of performance on all seen tasks after each new task learned (Rebuffi

et al. 2017c). Practically, given $A_{i,t}$ the accuracy of the i^{th} task after learning the t^{th} task, the final accuracy is (assuming balanced tasks):

$$\text{Acc}_F = \frac{1}{T}\sum_{i=1}^{T} A_{i,T}\,, \tag{2.6}$$

and the average incremental accuracy:

$$\text{Acc}_a = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{t}\sum_{i=1}^{t} A_{i,t}\,. \tag{2.7}$$

Average incremental accuracy is somewhat more important than simply the final accuracy: a continual model should be good after every step because in a true continual setting, there is not a "final task".

Other metrics exist (Díaz-Rodríguez et al. 2018), such as **forgetting** (Chaudhry et al. 2018) which records how much a model has lost performance-wise on a task compared to the first time it has learned it. The interest of this metric is to be agnostic of the absolute performance of the model used.

Finally, metrics such as **speed** (*i.e.* the number of images processed per second) or used **capacity** (*i.e.* number of learned parameters) are important: Ramasesh et al. (2022) recently showed that the larger a model was the lower was the forgetting.

2.4 Methods to reduce forgetting

Multiple approaches exist to reduce forgetting in Continual Learning. The major ones are rehearsal of old data (Section 2.4.1), regularizations constraining the model's behavior (Section 2.4.2), and structural adaptations (Section 2.4.3).

2.4.1 Rehearsal Learning

The most efficient method to reduce forgetting is **rehearsal learning** where old samples will be seen alongside the new samples. The amount of old samples stored is extremely limited, otherwise, it would defeat the purpose of continual learning. Figure 2.6 illustrates how rehearsal learning happens in Continual Learning. During the first step, a model is trained on all available samples. Then, it stores a limited amount of those in a *memory*. During the second step, the model has access to new samples but also all samples stored in the memory. In Class-Incremental, an equal amount of samples per class is stored in memory. There are

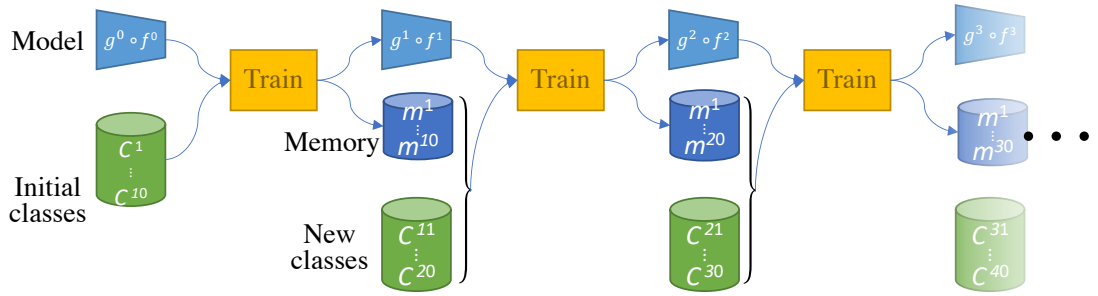


Figure 2.6. – **Training with a rehearsal memory**. After each task a fraction of the data is stored in a memory to be used in the next task. Rehearsal learning is the most efficient method to reduce forgetting, but unfortunately the memory capacity is often extremely limited.

two major approaches to determine this amount: Rebuffi et al. (2017c) propose to fully use a memory of size $\mathcal{M}$ among all $\mathcal{C}$, while Hou et al. (2019) instead kept fixed the number of samples stored per class to $\mathcal{M}/|\mathcal{C}^{1:T}|$.

Herding is the action of choosing which samples per class to store in the rehearsal memory. The most naive herding method is to randomly sample images. Despite its simplicity, it is quite competitive with more complex method (Castro et al. 2018), echoing similar results in Active Learning (AL) (Gal et al. 2017). Other herding methods include fetching samples close to the class mean in the feature space (Castro et al. 2018) or close to an incremental barycenter (Rebuffi et al. 2017c).

Sampling is an important but yet fewly investigated topic in Continual Learning. Most models mix all memory samples with new samples without any under- or over-sampling. Castro et al. (2018) propose to finetune for a few epochs, after training on a new step, on a balanced set of old and new classes samples. Chaudhry et al. (2019b) oversample tiny memory with as low as one sample per class, and show, in the context of Online Learning where models learn in only one epoch, that continual models still do not overfit. In the same context, Aljundi et al. (2019a) propose to over-sample the memory examples with the highest losses. In an imbalanced situation for Continual Learning, over- and under-sampling can be applied depending on the number of samples per class (C. D. Kim et al. 2020).

Efficient Storing is important for rehearsal learning: a bigger rehearsal memory leads invariably to less forgetting (Hou et al. 2019). Thus, multiple works consider

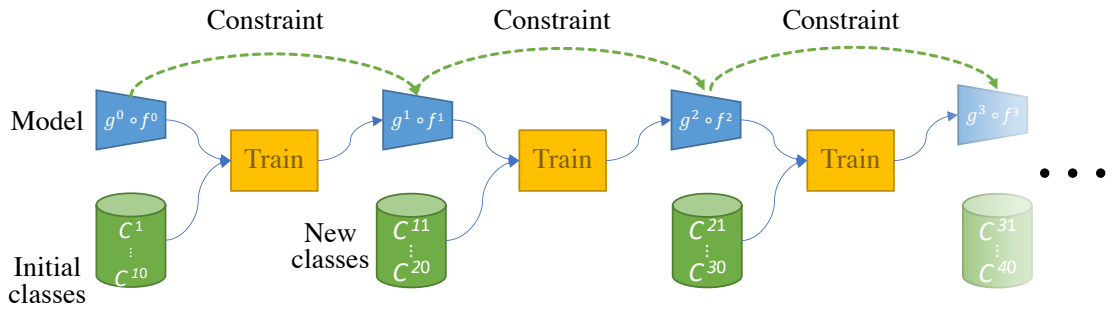


Figure 2.7. – **Constraining the new model based on the old model**. During each task, after the first one, the new model is constrained to be similar to the old model in order to reduce forgetting.

how to store more rehearsal samples given the same memory size: Hayes et al. (2020) compress intermediate features of memory samples with a lossless compression algorithm. Iscen et al. (2020) also store features but modify them through the training to handle the inherent internal covariate drift.

Pseudo-rehearsal does not need to store samples but instead generates pseudo-samples for rehearsal (Lesort et al. 2019a). The generation can be done with auto-encoders from intermediate features (Kemker and Kanan 2018; Ayub and Wagner 2021) or use Generative Adversarial Network (GAN) (Shin et al. 2017). Unfortunately, those methods have several drawbacks: they struggle to scale to large images, the generator size may be superior to a classic rehearsal memory size which would defeat the goal of using less storage, and finally, the generator may itself suffer from catastrophic forgetting (Zhai et al. 2019). Y. Liu et al. (2020) propose instead a method halfway between rehearsal and pseudo-rehearsal: the authors randomly sample real images, and then during continual training, slightly modify them via bi-level optimization (T. Wang et al. 2018) to minimize forgetting.

2.4.2 Regularization-based Approaches

A common and efficient way to reduce forgetting is to minimize the difference in behavior between the old and new models as illustrated in Figure 2.7. These constraints can be expressed through various forms and are described below.

2.4.2.1 Weight-based constraints

The most straightforward way to avoid completely forgetting is that the old and new models stay identical. While the model would be *rigid* (no forgetting), it is also not *plastic* (changing a lot) at all, and thus cannot learn any new tasks. Thus, a line of research proposed to constrain only a portion of the neurons:

$$\mathcal{L}(\theta^t, \theta^{t-1}) = \mathcal{L}_t(\theta^t) + \lambda \sum_i \Omega_i^{t-1} (\theta_i^t - \theta_i^{t-1})^2 \,, \tag{2.8}$$

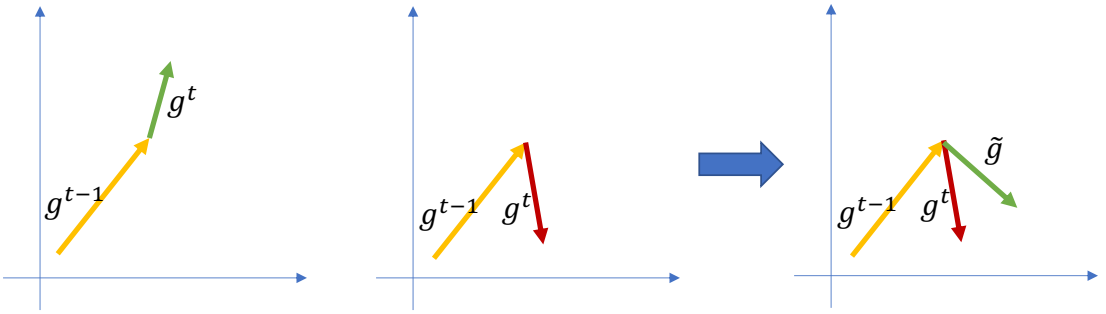
where $\mathcal{L}_t(\theta)$ is the loss at the current task t (*e.g.* the cross-entropy), θ_i^t and θ_i^{t-1} respectively the i^{th} neuron of the current and previous model, and Ω_i^{t-1} a neuron-wise importance factor. The intuition is that important neuron for the previous task $t-1$ should not change, while the others can be adapted to fit the new task t.

Kirkpatrick et al. (2017), followed by Zenke et al. (2017) and Chaudhry et al. (2018) propose to use the diagonal Fisher information matrix as importance factors. The motivation behind was that the posterior $p(\theta^{t-1}|\mathcal{D}^{t-1})$ must contain the information about which parameters are important to the previous dataset $\mathcal{D}^{t-1}$. This posterior can be approximated by a Gaussian distribution whose diagonal precision is given by diagonal Fisher information matrix. A higher value means a more important neuron for the previous task, and thus the constraint should be increased proportionally. Thus, a lower value, for a less important neuron, means that it can change drastically, which would facilitate learning new data. This strikes a balance between rigidity (not changing and thus not forgetting), and plasticity (changing, and thus learning new concepts). Note that Aljundi et al. (2018) instead use the sensitivity of the model when small perturbations are added to the neurons to measure their importance.

However, it is worth remarking that weight-based constraints are usually limited to the multi-heads setting where a task identifier is available at test time. Lesort et al. (2019b) show that in the single-head setting, they struggle to reduce forgetting and are significantly outperformed by the simple (but somewhat memory costly) rehearsal learning.

2.4.2.2 Gradient-Based methods

Lopez-Paz and Ranzato (2017) propose the GEM model that combines a constraint on the gradients and rehearsal learning. The algorithm requires that the loss on a given stored sample must not increase despite the model learning new classes. The authors, given a locality assumption, rewrite this formulation as en-



(a) Valid new gradient g^t (b) Invalid new gradient g^t (c) Correction of the new gradient g^t into $\tilde{g}$

Figure 2.8. – **GEM's gradient constraint** forcing updates to be in the same direction as the gradient w.r.t. old samples. In (a) the new gradient g^t is valid, while in (b) the new gradient violates the constraint of $\langle g^t, g_{t-1} \rangle \geq 0$. In (c), the invalid gradient g^t is projected to the closest valid alternative $\tilde{g}$.

forcing that the gradient of a new sample (g) to be in the same *direction* as the gradient of all stored old samples (g_i for all $i \in \mathcal{M}$):

$$\langle g, g_i \rangle \geq 0, \text{ for all } i \in \mathcal{M}\,, \tag{2.9}$$

with $\mathcal{M}$ the rehearsal memory. If the constraint is violated, the new gradient g is projected to the closest in L2 norm gradient that satisfies the angle constraint by minimizing a quadratic program. The constraint is illustrated in Figure 2.8. The drawback of this method is the computational cost that can grow prohibitively when the memory is too large. Chaudhry et al. (2019a) propose Averaged-GEM to speed up GEM: the authors do not constraint the gradient of individual memory samples but only the average of all memory samples. Aljundi et al. (2019c) also improved GEM's speed by selecting only a subset of the memory samples that maximize the feasible region.

Differently, but still constraining the gradients: Farajtabar et al. (2020)'s OGD forces the gradients of task t to be orthogonal to gradients of task $t-1$. They use the Gram-Schmidt procedure to orthogonalize the new gradients, allowing updates for the new task that minimally interfere with the performance of old tasks. Saha et al. (2021)'s GPM does likewise but uses instead a k-rank approximation of the SVD of the representation matrix.

2.4.2.3 Output-based constraints regularizations

Finally, the majority of Continual models that are benchmarked on large datasets (*e.g.* ImageNet (Deng et al. 2009)) use a combination of rehearsal learning (Section 2.4.1) and constraints on the model's outputs.

LwF (Z. Li and Hoiem 2016) and iCaRL (Rebuffi et al. 2017c) apply the Knowledge Distillation (KD) (Geoffrey Hinton et al. 2015) on the model's probabilities. It usually consists in minimizing the Kullback-Leiber divergence (KL) between the probabilities of the old and new models:

$$\mathcal{L}_{\text{KD}} = \text{KL}(\text{softmax}(\frac{\tilde{y}^{t-1}}{\tau}) \| \, \text{softmax}(\frac{\tilde{y}^{t}}{\tau}) \, , \tag{2.10}$$

where $\tilde{y}^{t-1} = g^{t-1}(f^{t-1}(\mathbf{x}))$ and $\tilde{y}^{t} == g^{t}(f^{t}(\mathbf{x}))$ are respectively the logits of the old and new model, and τ a *temperature* to soften the probabilities in order to give more importance to the model confidence in other classes than the top one. These probabilities, nicknamed *dark knowledge* by Geoffrey Hinton et al. (2015), contain additional information about the model which are useful to distil. Note that in the context of Class-Incremental, the new model predicts more classes than the old model, therefore, the KL is only applied on the logits common to both the old and the new models. The KD is sometimes also defined as the binary cross-entropy between the sigmoid-activated logits.

Constraining the probabilities is now so ubiquitous that most models include it in their base losses. On the other hand, a few models considered constraining intermediate outputs. MK2D (P. Zhou et al. 2019) uses the KD from both the final classifier and an auxiliary classifier similarly to the Inception network (Szegedy et al. 2015). Hou et al. (2019) maximize the cosine similarity between the embeddings produced by the GAP. Dhar et al. (2019) minimizes the L1 distance between the attention maps produced by GradCam (Selvaraju et al. 2017).

2.4.3 Structural Strategies

Multiple works also propose to adopt dynamic strategies where the configuration of the neural network evolves after each task. Critically, not only the number of neurons changes in the classifier g^t (to incorporate new classes to predict), but the feature extractor f^t's neurons count or organization will also differ from its previous iteration f^{t-1}.

Subnetworks The Lottery Ticket Hypothesis (Frankle and Carbin 2019) states that subnetworks made of a fraction of the neurons and connections of a larger

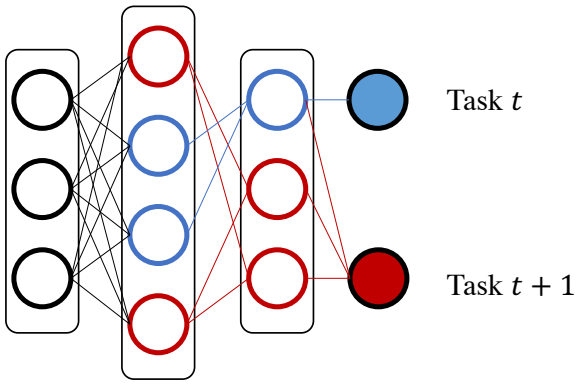


Figure 2.9. – **Task-specific subnetworks** that can be uncovered with a sparsity loss or learned masking. **Blue** neurons are dedicated to the task t, **red** neurons to the following task $t+1$, and **black** neurons are shared among all tasks. At test-time, a task identifier of the sample is required to select the right subnetwork path.

network, can reach excellent performance. Several Continual Learning models exploit that property by using a subnetwork per task. Those subnetworks can be uncovered via genetic algorithms (Fernando et al. 2017), via induced L1 sparsity (Golkar et al. 2019), or even learned masked Serrà et al. (2018) and Hung et al. (2019). Usually, these methods require a task identifier at test-time in order to select the right subnetwork (see *multi-heads* in Section 2.3). Later, Wortsman et al. (2020) propose to infer the task identifier by selecting the subnetwork with the lowest entropy. This subnetwork-based approach is illustrated in Figure 2.9.

Dynamically Expandable Networks A neural network can also be expanded through its continual training to accommodate the growing amount of tasks to solve. First, Rusu et al. (2016) propose to have one network per task, where the i^{th} network would depend both on the input and all previous networks' intermediate features. Unfortunately, memory consumption is quickly prohibitive with many tasks. Following works propose to only add blocks of parameters, and only when deemed necessary (Veniat et al. 2021). While these dynamic networks often require an identifier at test-time to select the right subset of parameters, recently, DER (Yan et al. 2021) removed this need by learning a classifier upon the concatenated features of all tasks: Their dynamic expansion of the representation adds a new feature extractor per task. All extractors' embeddings would then be concatenated and fed to a unified classifier, discarding the need for a task identifier at test-time. To limit an explosion in the number of parameters, they aggressively prune each model after each task using the HAT (Serrà et al. 2018) procedure. Unfortunately, the pruning is hyperparameter sensitive. Therefore, hyperparameters

are tuned differently on each experiment: for example, learning a dataset in 10 steps or in 50 steps use different hyperparameters. While being impracticable, it is also unrealistic because the number of classes is not known in advance in a true continual situation. Simple-DER (Zhuoyun Li et al. 2021) also uses multiple extractors, but its pruning method doesn't need any hyperparameters; the negative counterpart is that Simple-DER controls less the parameter growth.

Task Conditioning Rather than adding many new parameters, it is also possible to only add a few parameters that will adapt the existing network behavior for a task. Rebuffi et al. (2017a) propose to add a different residual per task: given a task identifier, the associated residual is used, and the features are modulated to best fit the given task. Instead of adapting the features, Wen et al. (2020) and Q. Sun et al. (2019) propose to share most of the weights across tasks, but have task-specific weights that directly modify the shared weights.

Mixture-of-Experts Mixture of experts (Masoudnia and Ebrahimpour 2014) have also been proposed, where multiple experts combine their decision. Aljundi et al. (2017) learn a gating system to use the right task-specific expert. Collier et al. (2020) stress the importance on sharing experts when tasks are similar.

Classifier Correction Forgetting happens in both the feature extractor and the classifier. Previously described rehearsal and regularization methods try to reduce it in both places. On the other hand, multiple works focus solely on the classifier. They remark that in Class Incremental Learning (CIL), the classifier is miscalibrated (Guo et al. 2017) where the model over-predicts new classes to the detriment of old classes. Belouadah and Popescu (2019) compensate the bias towards new classes by rectifying predictions of past classes using their recorded accuracies and confidences. Wu et al. (2019) learn a linear model on validation data to recalibrate the logits of the new classes. B. Zhao et al. (2020) normalizes the norm of the classifier weights associated with new classes so that their average norm becomes the same as that for old classes. Hou et al. (2019), aims for a similar result by replacing the dot product in the classifier with the cosine similarity, resulting in unit norm classifier weights.

2.5 Positioning

Continual Learning encompasses very different benchmarks and methods. In this PhD thesis, we tackle multiple types of continual drift using different approaches. All the proposed strategies consider the intermediary features of a

neural network to reduce catastrophic forgetting. We summarize below the three main chapters:

Feature-based Regularizations First in Chapter 3, we consider Class Incremental Learning (CIL) scenarios with a prior drift where new classes are continually added. In this setting, our approach consists in rehearsal learning (Section 2.4.1) and output-based regularizations (Section 2.4.2.3). Previous works reduced forgetting by constraining either the final probabilities of the classifier (Z. Li and Hoiem 2016) or the ultimate embedding of the feature extractor (Hou et al. 2019; Dhar et al. 2019). In a first section (Section 3.2), we propose instead a regularization applied on several intermediary levels of the features extractor. Moreover, few considerations have been made on how the regularization loss makes the model more rigid, leading to less forgetting on old classes but also impacting negatively the learning of new classes. Instead, we carefully define a regularization loss, through pooling, balance effectively the performance of old and new classes. In a second section (Section 3.3), we consider if the forgetting could be avoided preemptively. Aljundi and Tuytelaars (2019) consider allocating extra capacity implicitly in the learned feature space for future classes using an unsupervised sparsity loss. Differently, we choose to draw from weak metadata of the future unseen classes (Han et al. 2020) to estimate their representation and thus explicitly allocate capacity, reducing forgetting before it even happens.

Continual Semantic Segmentation Second, in Chapter 4, we tackle Continual Semantic Segmentation (CSS). In this benchmark, defined recently by Cermelli et al. (2020), images have a label per pixel, and only the current classes are labelized. Thus, both the prior and concept drifts happen where new classes are added but also the signification of a pixel can change through time. In this setting, all previous works only considered regularizations (Section 2.4.2.3) based on the final probabilities (Michieli and Zanuttigh 2019; Cermelli et al. 2020). Instead, we expand from our previous chapter to consider the multi-scale intermediary features. Moreover, no work directly tackled the partial labelization of the images (Cermelli et al. 2020). We propose for this challenge an uncertainty-based pseudo-labeling (Saporta et al. 2020), and a new rehearsal method (Section 2.4.1), optimized for segmentation.

Dynamic Transformers In this third and last chapter (Chapter 5), we only aim to solve the prior drift but using a method radically different from previous chapters: dynamic networks (Section 2.4.3). Previous dynamic networks usually expand their capacity by a large margin during the continual training to handle the growing amount of tasks to learn (Yan et al. 2021). To avoid a parameter

count unbounded growth, the models are usually pruned aggressively. The main drawback of these methods is that the pruning can still result in models too large and often need careful finetuning of hyperparameters. We propose in this chapter, a dynamic expansion based on the transformer architecture with almost no memory overhead contrary to concurrent works, that conditions (Perez et al. 2018) the intermediary features.

CHAPTER 3

VISUAL FEATURE-BASED REGULARIZATIONS

Chapter abstract

Regularization-based methods are powerful to reduce catastrophic forgetting, especially in challenging setting such as Class Incremental Learning (CIL) with single-head where task identity is not known at test-time. However, they often consider only constraining the final probability predictions of the network, weakly constraining the model's behavior and in turns falls short to forgetting when faced to large amount of tasks.

In this chapter, we present two methods exploiting the intermediate features to reduce forgetting. The first approach, nicknamed PODNet, aims to constrain similar statistics to avoid a representation drift at all levels of the network. We show that it scales particularly well on extreme scenarios where classes are learned one by one for a long succession of iterations, due to its balancing of rigidity (not forgetting old classes) - plasticity (learning new classes) tradeoff. The second approach, named Ghost, pre-emptively allocate capacity for future —yet to be seen— classes, avoiding forgetting before it even happens. We estimate those class locations in the representation space by drawing inspiration from the ZeroShot-Learning (ZSL) literature.

We evaluate our models on numerous benchmarks, from large image classification datasets (e.g. ImageNet), to zeroshot datasets with class attributes (e.g. AwA2), and perform ablations to validate the soundness of our methods.

The work in this section has led to a publication to two papers:

- *Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle (2020). "PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning". In:* Proceedings of the IEEE European Conference on Computer Vision (ECCV)
- *Arthur Douillard, Eduardo Valle, Charles Ollion, and Matthieu Cord (2021c). "Insights from the Future for Continual Learning". In:* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop

Contents

3.1 Introduction

Class Incremental Learning (CIL), where each task brings new classes, is among the most challenging settings of Continual Learning (CL) (refer to Section 2.3). When evaluated in single-head, where the task identity is not known at test-time, the majority of the methods rely on rehearsal learning where a limited amount of old data is replayed (Section 2.4.1). Furthermore, it is often combined to regularizations that aim to limit forgetting (Section 2.4.2.3). Regularization-based approaches defined in the literature have two drawbacks: (1) they focus solely on the model's output probabilities, disregarding all intermediary features, and as a result only weakly constrain a model's behavior; (2) the regularizations are designed to avoid forgetting of old classes (rigidity) to the detriment of giving enough slack to the continual model in order to learn new classes (plasticity). We present in this chapter two novel regularizations that consider the intermediary visual features.

Our first contribution, introduced in **PODNet** (Section 3.2), aims to **regularize the statistics shift at the intermediate feature-level to enforce a consistent representation** at multiple levels of the feature extractor across all steps of the continual training. We design this regularization to balance efficiently the rigidity (not forgetting old classes) and the plasticity (learning new classes) of continual models.

Our second contribution, nicknamed **Ghost** (Section 3.3), on the other hand, **avoid pre-emptively forgetting by regularizing the feature space** at locations where the model estimates future classes will be. The future classes representation

is produced using a generative model that exploits detailed attributes of said classes.

3.2 PODNet: reducing forgetting via intermediate feature statistics

3.2.1 Related Work

We design our model explicitly for a Class Incremental Learning (CIL) setting with a large number of classes. In this challenging setting, rehearsal learning is essential. We use the simple but efficient rehearsal method of Rebuffi et al. (2017c), refer to Section 2.4.1 for more information. Moreover, we limit the forgetting of our continual model using a regularization-based approach that constrain the model's output. Furthermore, we aim to explicitly learn a robust feature representation by drawing inspiration from the few-shot literature.

Regularization of the model's output Regularizations constraining the rate of change of the weights (Kirkpatrick et al. 2017) or the gradients (Farajtabar et al. 2020) have been proposed. They reduce effectively the forgetting of the feature extractors in a multi-heads setting where the task identity is known at test-time. However, in the more challenging and realistic setting of single-head, where classes from all tasks have to be predicted, these methods often fail to improve a naive finetuning (Lesort et al. 2019b). Therefore, We focused on regularizations applied on a model's output. LwF (Z. Li and Hoiem 2016) first used the Knowledge Distillation (KD) of Geoffrey Hinton et al. (2015): a KL divergence between the probabilities of the old and new models. While simple, this loss — with few variations— has been used by multiple following works (Rebuffi et al. 2017c; B. Zhao et al. 2020). Few works tried to constrain intermediate outputs of the model: LwM (Dhar et al. 2019) proposed to minimize the L2 distance between gradient-based attention maps produced by GradCam (Selvaraju et al. 2017). M2KD (P. Zhou et al. 2019) used a KD on both the final predictions and intermediate predictions from an auxiliary classifier similar to Inception (Szegedy et al. 2015). Hou et al. (2019) maximizes the cosine similarity between the final embeddings before the classifier. Finally, Zagoruyko and Komodakis (2016), in the context of model compression, investigated attention-based distillation for image classifiers, by pooling the intermediate features of convolutional networks into attention maps, then used in their distillation losses.

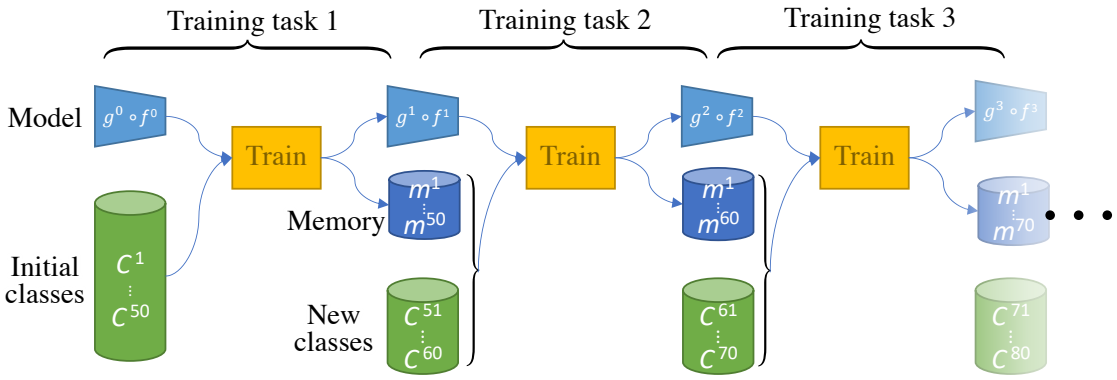


Figure 3.1. – **Training protocol for incremental learning**. At each training task we learn a new set of classes, and the model must retain knowledge about *all* classes. The model is allowed a *limited* memory of samples of old classes. In our experiments, the first task contain more classes than the following tasks (50 vs 10 on this figure).

Representation learning was already implicitly present in iCaRL (Rebuffi et al. 2017c): it introduced the Nearest Mean Examplar (NME) strategy which averages the outputs of the deep convolutional network to create a single proxy feature vector per class that are then used by a nearest-neighbor classifier predict the final classes. Hou et al. (2019) adopted this method and also introduced another, named CNN, which uses the output class probabilities to classify incoming samples, freezing (during training) the classifier weights associated with old classes, and then fine-tuning them on an under-sampled dataset. Hou et al. (2019), in the method called here UCIR, made representation learning explicit, by noticing that the limited memory imposed a severe imbalance on the training samples available for the old and for the new classes. To overcome that difficulty, they designed a metric-learning model instead of a classification model. That strategy is often used in few-shot learning (Gidaris and Komodakis 2018) because of its robustness when faced to few data. Because classical metric architectures require special training sampling (*e.g.* semi-hard sampling for triplets), Luo et al. (2018) chose instead to redesign the classifier's last layer of their model to use the cosine similarity.

3.2.2 Model

We follow the notations defined in the Notations. Our model is evaluated in the class incremental setting with rehearsal memory (Section 2.4.1) depicted Figure 3.1. Our strategy is made of two keys components: a distillation loss applied at the intermediate feature-level, and a local-similarity classifier.

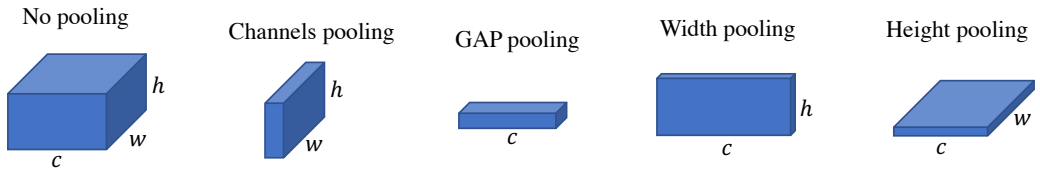


Figure 3.2. – **Different possible poolings**. The output from a convolutional layer $\mathbf{h}_\ell^t[c, w, h]$ may be pooled (summed over) one or more axes. The resulting loss considers only the pooled activations instead of the individual components, allowing more plasticity across the pooled axes.

3.2.2.1 POD: Pooled Outputs Distillation loss

Constraining the evolution of the weights is crucial to reduce forgetting. Each new task t learns a new (student) model, whose weights are not only initialized with those of the previous (teacher) model, but also constrained by a distillation loss. That loss must be carefully balanced to prevent forgetting (rigidity), while allowing the learning of new classes (plasticity).

To this goal, we propose a set of constraints we call **Pooled Outputs Distillation (POD)**, applied not only over the final embedding output by $\mathbf{h}^t = f^t(\mathbf{x})$, but also over the output of its intermediate layers $\mathbf{h}_\ell^t = f_\ell^t(\mathbf{x})$ (where by notation overloading $f_\ell^t(\mathbf{x}) \equiv f_\ell^t \circ \ldots \circ f_1^t(\mathbf{x})$, and thus $f^t(\mathbf{x}) \equiv f_L^t \ldots \circ f_\ell^t \circ \ldots f_1^t(\mathbf{x})$).

The convolutional layers of the network output tensors $\mathbf{h}_\ell^t$ with components $\mathbf{h}_\ell^t[c, w, h]$, where c stands for channel (filter), and $w \times h$ for column and row of the spatial coordinates. The loss used by POD may pool (sum over) one or several of those indexes, more aggressive poolings (Figure 3.2) providing more freedom, and thus, plasticity: the lowest possible plasticity imposes an exact similarity between the previous and current model while higher plasticity relaxes the similarity definition.

Pooling is an important operation in Computer Vision, with a strong theoretical motivation. In the past, pooling has been introduced to obtain invariant representations (Lowe 1999; Lazebnik et al. 2006). Here, the justification is similar, but the goal is different: as we will see, the pooled indexes are aggregated in the proposed loss, allowing *plasticity*. Instead of the model acquiring invariance to the input image, the desired loss acquires invariance to model evolution, and thus, representation. The proposed pooling-based formalism has two advantages: first, it organizes disparately proposed distillation losses into a neat, general formalism. Second, as we will see, it allowed us to propose novel distillation losses, with better plasticity-rigidity compromises. Those topics are explored next.

Pooling of convolutional outputs As explained before, POD constrains the output of each intermediate convolutional layer $\mathbf{h}_\ell^t = f_\ell^t(\cdot)$ (in practice, each stage of a ResNet (K. He et al. 2016)). All POD variants use the Euclidean distance of ℓ^2-normalize tensors, here noted as $\|\cdot - \cdot\|$. They differ on the type of pooling applied before that distance is computed. On one extreme, one can apply no pooling at all, resulting in the most strict loss, the most rigid constrains, and the lowest plasticity:

$$\mathcal{L}_{\text{POD-pixel}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) = \sum_{c=1}^{C}\sum_{w=1}^{W}\sum_{h=1}^{H} \left\|\mathbf{h}_\ell^{t-1}[c,w,h] - \mathbf{h}_\ell^t[c,w,h]\right\|^2 . \tag{3.1}$$

By pooling the channels, one preserves only the spatial coordinates, resulting in a more permissive loss, allowing the activations to reorganize across the channels, but penalizing global changes of those activations across the space,

$$\mathcal{L}_{\text{POD-channel}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) = \sum_{w=1}^{W}\sum_{h=1}^{H} \left\| \sum_{c=1}^{C} \mathbf{h}_\ell^{t-1}[c,w,h] - \sum_{c=1}^{C} \mathbf{h}_\ell^t[c,w,h] \right\|^2 ; \tag{3.2}$$

or, contrarily, by pooling the space (equivalent, up to a factor, to a Global Average Pooling), one preserves *only* the channels:

$$\mathcal{L}_{\text{POD-gap}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) = \sum_{c=1}^{C} \left\| \sum_{w=1}^{W}\sum_{h=1}^{H} \mathbf{h}_\ell^{t-1}[c,w,h] - \sum_{w=1}^{W}\sum_{h=1}^{H} \mathbf{h}_\ell^t[c,w,h] \right\|^2 . \tag{3.3}$$

Note that the only difference between the variants is in the position of the summation. For example, contrast equations Equation 3.1 and 3.2: in the former the differences are computed between activation pixels, and then totaled; in the latter, first the channel axis is flattened, then the differences are computed, resulting in a more permissive loss.

We can trade a little plasticity for rigidity, with less aggressive pooling by aggregating statistics across just one of the spatial dimensions:

$$\mathcal{L}_{\text{POD-width}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) = \sum_{c=1}^{C}\sum_{h=1}^{H} \left\| \sum_{w=1}^{W} \mathbf{h}_\ell^{t-1}[c,w,h] - \sum_{w=1}^{W} \mathbf{h}_\ell^t[c,w,h] \right\|^2 ; \tag{3.4}$$

or, likewise, for the vertical dimension, resulting in POD-height. Each of those variants measure the distribution of activation pixels across their respective axis. These two complementary intermediate statistics can be further combined:

$$\mathcal{L}_{\text{POD-spatial}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) = \mathcal{L}_{\text{POD-width}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) + \mathcal{L}_{\text{POD-height}}(\mathbf{h}_\ell^{t-1}, \mathbf{h}_\ell^t) . \tag{3.5}$$

$\mathcal{L}_{\text{POD-spatial}}$ is minimal when the average statistics over the dataset, on both width and height axes, are similar for the previous and current model. It brings the right balance between being too rigid (Equation 3.1) and being too permissive (Equation 3.2 and 3.3).

Constraining the final embedding After the convolutional layers, the network, by design, flattens the spatial coordinates, and the formalism above needs adjustment, as a summation over w and h is no longer possible. Instead, we set a flat constraint on the final embedding $\mathbf{h}^t = f^t(\mathbf{x})$:

$$\mathcal{L}_{\text{POD-flat}}(\mathbf{h}^{t-1}, \mathbf{h}^t) = \left\| \mathbf{h}^{t-1} - \mathbf{h}^t \right\|^2 . \tag{3.6}$$

Combining the losses, analysis The final POD loss combines the two components:

$$\mathcal{L}_{\text{POD-final}}(\mathbf{x}) = \frac{\lambda_c}{L-1} \sum_{\ell=1}^{L-1} \mathcal{L}_{\text{POD-spatial}} \left(f_\ell^{t-1}(\mathbf{x}), f_\ell^t(\mathbf{x}) \right) + \lambda_f \mathcal{L}_{\text{POD-flat}} \left(f^{t-1}(\mathbf{x}), f^t(\mathbf{x}) \right) . \tag{3.7}$$

The hyperparameters λ_c and λ_f are necessary to balance the two terms, due to the different nature of the intermediate outputs (spatial and flat).

As mentioned, the strategy above generalizes disparate propositions existing both in the literature of incremental learning, and elsewhere. When $\lambda_c = 0$, it reduces to the cosine constraint of *Less-Forget*, proposed by Hou et al. (2019) for incremental learning, which constrains only the final embedding. When $\lambda_f = 0$ and POD-spatial is replaced by POD-pixel, it suggests the Perceptual Features loss, proposed for style transfer (Johnson et al. 2016). When $\lambda_f = 0$ and POD-spatial is replaced by POD-channel, the strategy hints at the loss proposed by Zagoruyko and Komodakis (2016) to allow distillation across different networks, a situation in which the channel pooling responds to the very practical need to allow the comparison of architectures with different number of channels.

As we will see in our evaluations of pooling strategies (Table 3.4), what proved optimal was a completely novel idea, POD-spatial, combining two poolings, each of which flattens one of the spatial coordinates. That relatively rigid strategy (channels and one of the spatial coordinates are considered in each half of the loss) makes intuitive sense in our context, which is *small-task* incremental learning, and thus where we expect a slow drift of the model across a single task.

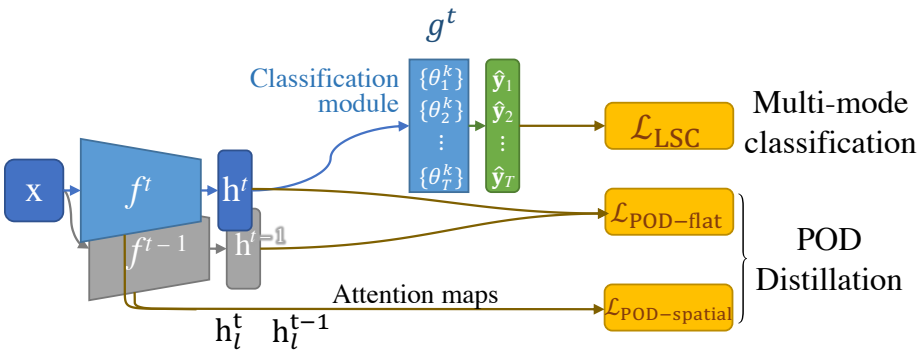


Figure 3.3. – **Overview of PODNet**: the distillation loss POD prevent excessive model drift by constraining intermediate outputs of the ConvNet f and the LSC classifier g learns a more expressive multi-modal representation.

3.2.2.2 Local Similarity Classifier

Hou et al. (2019) observed that the class imbalance of incremental learning has concrete manifestations on the parameters of the final layer on classifiers, namely the weights for the over-represented (new) classes becoming much larger than those for the underrepresented (old) classes. To overcome this issue, their method (called here UCIR) ℓ^2-normalizes both the weights and the activations, which corresponds to taking the cosine similarity instead of the dot product. For each class c, their last layer becomes

$$\hat{\mathbf{y}}_c = \frac{\exp\left(\eta \langle \theta_c, \mathbf{h} \rangle\right)}{\sum_i \exp\left(\eta \langle \theta_i, \mathbf{h} \rangle\right)}, \tag{3.8}$$

where θ_c are the last-layer weights for class c, η is a learned scaling parameter, and $\langle \cdot, \cdot \rangle$ is the cosine similarity.

However, this strategy optimizes a *global similarity*: its training objective increases the similarity between the extracted features and their associated weights. For each class, the normalized weight vector acts as a *single* proxy (Movshovitz-Attias et al. 2017), towards which the learning procedure pushes all samples in the class.

We observed that such global strategy is hard to optimize in an incremental setting. To avoid forgetting, the distillation losses (Section 3.2.2.1) tries to keep the final embedding $\mathbf{h}$ consistent through time so that the class proxies stay relevant for the classifier. Unfortunately catastrophic forgetting, while alleviated by current methods, is not solved and thus the distribution of $\mathbf{h}$ may change. The cosine classifier is very sensitive to those changes as it models a unique majority mode through its class proxies.

Local Similarity Classifier The problem above lead us to amend the classification layer during training, in order to consider multiple proxies/modes per class. A shift in the distribution of $\mathbf{h}$ will have less impact on the classifier as more modes are covered.

Our redesigned classification layer, which we call Local Similarity Classifier (LSC), allows for K multiple proxies/modes during training. Like before, the proxies are a way to interpret the weight vector in the cosine similarity, thus we allow for K vectors $\theta_{c,k}$ for each class c. The similarity $s_{c,k}$ to each proxy/mode is first computed. An averaged class similarity $\hat{\mathbf{y}}_c$ is the output of the classification layer:

$$s_{c,k} = \frac{\exp \langle \theta_{c,k}, \mathbf{h} \rangle}{\sum_i \exp \langle \theta_{c,i}, \mathbf{h} \rangle}, \qquad \hat{\mathbf{y}}_c = \sum_k s_{c,k} \langle \theta_{c,k}, \mathbf{h} \rangle . \tag{3.9}$$

The multi-proxies classifier optimizes the similarity of each sample to its ground truth class representation and minimizes all others. A simple cross-entropy loss would work, but we found empirically that the NCA loss (Goldberger et al. 2005; Movshovitz-Attias et al. 2017) converged faster. We added to the original loss a hinge $[\cdot]_+$ to keep it bounded, and a small margin δ to enforce stronger class separation, resulting in the final formulation:

$$\mathcal{L}_{\text{LSC}} = \left[-\log \frac{\exp\left(\eta(\hat{\mathbf{y}}_y - \delta)\right)}{\sum_{i \neq y} \exp \eta \hat{\mathbf{y}}_i} \right]_+ . \tag{3.10}$$

Weight initialization for new classes The incremental learning setting imposes detecting new classes at each new task t. New weights $\{\theta_{c,k} \mid \forall c \in C_N^t, \forall k \in 1...K\}$ must be added to predict them. We could initialize them randomly, but the class-agnostic features of the ConvNet f, extracted by the model trained so far offer a better prior. Thus, we employ a generalization of Imprinted Weights (Qi et al. 2018) procedure to multiple modes: for each new class c, we extract the features of its training samples, use a k-means algorithm to split them into K clusters, and use the centroids of those clusters as initial values for $\theta_{c,k}$. This procedure ensures mode diversity at the beginning of a new task and resulted in a one percentage point improvement on CIFAR100.

3.2.2.3 Complete model formulation

Our model has the classical structure of a convolutional network $f(\cdot)$ acting as a feature extractor, and a classifier $g(\cdot)$ producing a score per class. We introduced two innovations to this model: (1) our main contribution is a novel distillation loss (POD) applied all over the ConvNet, from the spatial features $\mathbf{h}_\ell$ to the final flat embedding $\mathbf{h}$; (2) as further refinement we propose that the classifier learns a

multi-modal representation that explicitly keeps multiple proxy vectors per class, increasing the model expressiveness and thus making it less sensible to shift in the distribution of $\mathbf{h}$. The final loss for current model $g^t \circ f^t$, i.e., the model trained for task t, is simply their addition $\mathcal{L}_{\{f^t;g^t\}} = \mathcal{L}_{\text{LSC}} + \mathcal{L}_{\text{POD-final}}$. The overall model is displayed in Figure 3.3.

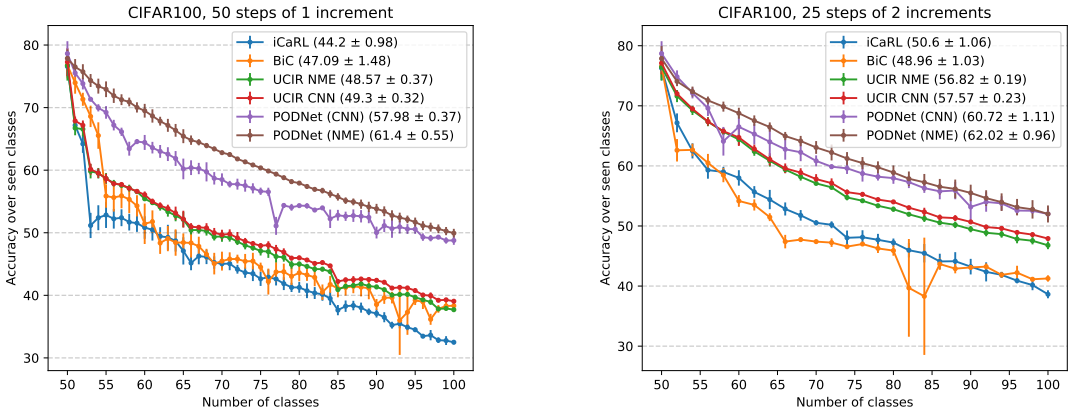
3.2.3 Experiment Results

We compare our technique (PODNet) with three state-of-the-art models. Those models are particularly comparable to ours since they all employ a sample memory with a fixed capacity. Both iCaRL (Rebuffi et al. 2017c) and UCIR (Hou et al. 2019) use the same inference method –NME, although UCIR also proposes a second inference method based on the classifier probabilities (called here UCIR-CNN). We evaluate PODNet with both inference methods for a small scale dataset, and the latter for larger scale datasets. BiC (Wu et al. 2019), while not focused on representation learning, is specially designed to be effective on large scale datasets, and thus provided an interesting baseline.

Datasets We employ three images datasets – extensively used in the literature of incremental learning – for our experiments: CIFAR100 (Krizhevsky and Geoffrey Hinton 2009), ImageNet100 (Deng et al. 2009; Hou et al. 2019; Wu et al. 2019), and ImageNet1000 (Deng et al. 2009). ImageNet100 is a subset of ImageNet1000 with only 100 classes, randomly sampled from the original 1000.

Protocol We validate our model and the compared baselines using the challenging protocol introduced by Hou et al. (2019): we start by training the models on half the classes (i.e., 50 for CIFAR100 and ImageNet100, and 500 for ImageNet1000). Then the classes are added incrementally in steps. We divide the remaining classes equally among the steps, *e.g.* for CIFAR100 we could have 5 steps of 10 classes or 50 steps of 1 class. Note that a training of 50 steps is actually made of 51 different tasks: the initial training followed by the incremental steps. Models are evaluated after each step on *all the classes seen until then*. To facilitate comparison, the accuracies at the end of each step are averaged into a unique score called *average incremental accuracy* (Rebuffi et al. 2017c) (Equation 2.7 in Figure 2.3). If not specified otherwise, the average incremental accuracy is the score reported in all our results. The protocol is also illustrated in Figure 3.1.

For CIFAR100 and ImageNet100, we ran all experiments thrice, varying the order of the classes. We report the averages and standard deviations in tables and



(a) 50 steps, 1 class / step

(b) 25 steps, 2 classes / step

Figure 3.4. – **Incremental Accuracy on CIFAR100** over three orders for two different step sizes. The legend reports the average incremental accuracy.

graphs. For ImageNet1000, whose models took much longer to train, we ran each experiment once.

Following Hou et al. (2019), for all datasets, and all compared models, we limit the memory M_{per} to 20 images per old class. For results with different memory settings, refer to Table 3.5.

Implementation details For fair comparison, all compared models employ the same ConvNet backbone: ResNet-32 for CIFAR100, and ResNet-18 for ImageNet. We remove the ReLU activation at the last block of each ResNet end-of-stage to provide a signed input to POD (Section 3.2.2.1). We implemented our method (called here PODNet) in PyTorch (Paszke et al. 2017). We compare both ours and UCIR's implementation of iCaRL. Results of UCIR come from the implementation of Hou et al. (2019). We provide their reported results and also run their code ourselves. We used our implementation of BiC (Wu et al. 2019) in order to compare with the same backbone. We sample our memory images using *herding selection* (Rebuffi et al. 2017c) and perform the inference with two different methods: the *Nearest-Mean-Examplars* (NME) proposed for iCarl, and also adopted on one of the variants of UCIR (Hou et al. 2019), and the "CNN" method introduced for UCIR (see Section 3.2.1). The training procedure is identical for both inference scheme, their difference lies at test-time: "CNN" uses the argmax of all classes probabilities. "NME" extracts features of all data (new classes data and rehearsal data), builds class prototypes by averaging features per class, and finally predictions using a K-NN. Please see Section A.2 in the appendix for the full implementation details.

	CIFAR100			
	50 steps	25 steps	10 steps	5 steps
New classes per step	1	2	5	10
*iCaRL** (Rebuffi et al. 2017c)	—	—	52.57	57.17
iCaRL	44.20± 0.98	50.60± 1.06	53.78± 1.16	58.08± 0.59
BiC (Wu et al. 2019)	47.09± 1.48	48.96± 1.03	53.21± 1.01	56.86± 0.46
*UCIR (NME)** (Hou et al. 2019)	—	—	60.12	63.12
UCIR (NME)	48.57± 0.37	56.82± 0.19	60.83± 0.70	63.63± 0.87
*UCIR (CNN)**	—	—	60.18	63.42
UCIR (CNN)	49.30± 0.32	57.57± 0.23	61.22± 0.69	64.01± 0.91
PODNet (NME)	**61.40± 0.68**	**62.71± 1.26**	**64.03± 1.30**	**64.48± 1.32**
PODNet (CNN)	**57.98± 0.46**	**60.72± 1.36**	**63.19± 1.16**	**64.83± 0.98**

Table 3.1. – **CIFAR100 quantitative experiments:** Average incremental accuracy for PODNet vs state of the art. We run experiments three times (random class orders) on CIFAR100 and report averages and standard deviations. Models with an asterisk * are reported directly from Hou et al. (2019). The initial task's size is 50 classes, the remaining 50 classes are learned incrementally.

3.2.3.1 Quantitative results

The comparisons with all the State-of-the-Art are tabulated in Table 3.1 for CIFAR100 and Table 3.2 for ImageNet100 and ImageNet1000. All tables show the average incremental accuracy for each considered models with various number of steps on the incremental learning run. The "New classes per step" row shows the amount of new classes introduced per task.

CIFAR100 We run our comparisons on 5, 10, 25, and 50 steps with respectively 10, 5, 2, and 1 classes per step. We created three random class orders to run each experiment thrice, reporting averages and standard deviations. For CIFAR100 only, we evaluated our model with two different kinds of inference: NME and CNN. With both methods, our model surpasses all previous State-of-the-Art models on all steps. Moreover, our model relative improvement grows as the number the steps increases, surpassing existing models by 0.82, 2.81, 5.14, and 12.1 percent points (*p.p.*) for respectively 5, 10, 25, and 50 steps. Larger numbers of steps imply stronger forgetting; those results confirm that PODNet manages to reduce drastically the said forgetting. While PODNet with NME has the largest gain, PODNet with CNN also outperforms the previous State-of-the-Art by up to 8.68*p.p.*. See Figure 3.4 for a plot of the incremental accuracies on this dataset. In the extreme setting of 50 increments of 1 class (Figure 3.4a), our model showcases large differences, with slow degradation ("*gradual forgetting*" (French 1999)) due to forgetting

	ImageNet100				Imagenet1000	
	50 steps	25 steps	10 steps	5 steps	10 steps	5 steps
New classes per step	1	2	5	10	50	100
iCaRL* (Rebuffi et al. 2017c)	—	—	59.53	65.04	46.72	51.36
iCaRL	54.97	54.56	60.90	65.56	—	—
BiC (Wu et al. 2019)	46.49	59.65	65.14	68.97	44.31	45.72
UCIR (NME)* (Hou et al. 2019)	—	—	66.16	68.43	59.92	61.56
UCIR (NME)	55.44	60.81	65.83	69.07	—	—
UCIR (CNN)*	—	—	68.09	70.47	61.28	64.34
UCIR (CNN)	57.25	62.94	67.82	71.04	—	—
PODNet (CNN)	**62.48** ± 0.59	**68.31** ± 2.45	**74.33** ± 0.93	**75.54** ± 0.26	**64.13**	**66.95**

Table 3.2. – **ImageNet quantitative experiments:** Average incremental accuracy, PODNet vs state of the art. Models with an asterisk * are reported directly from Hou et al. (2019). The initial task's sizes are respectively 50 and 500 classes for ImageNet100 and ImageNet1000. The remaining classes are learned incrementally.

throughout the run, while the other models show a quick performance collapse ("*catastrophic forgetting*") at the start of the run.

ImageNet100 We run our comparisons on 5, 10, 25, and 50 steps with respectively 10, 5, 2, and 1 classes per step. For both ImageNet100, and ImageNet1000 we report only PODNet with CNN, as the kNN-based NME classifier did not generalize as well to larger-scale datasets. With the more complex images of ImageNet100, our model also outperforms the State-of-the-Art on all tested runs, by up to 6.51*p.p.*.

ImageNet1000 This dataset is the most challenging, with much greater image complexity than CIFAR100, and ten times the number of classes as ImageNet100. We evaluate the models in 5 and 10 steps, and results confirm the consistent improvement of PODNet against existing arts by up to 2.85*p.p.*.

3.2.3.2 Further analysis & ablation studies

Ablation Studies Our model has two components: the distillation loss POD and the LSC classifier. An ablation study showcasing the contribution of each component is displayed in Table 3.3: each additional component improves the model performance. We evaluate every ablation on CIFAR100 with 50 steps of 1 new class each. The reported metric is the average incremental accuracy. The table

Classifier	POD-flat	POD-spatial	NME	CNN
Cosine			40.76	37.93
Cosine	✓		48.03	46.73
Cosine		✓	54.32	57.27
Cosine	✓	✓	56.69	55.72
LSC-CE	✓	✓	59.86	57.45
LSC			41.56	40.76
LSC	✓		53.29	52.98
LSC		✓	**61.42**	57.64
LSC	✓	✓	61.40	**57.98**

Table 3.3. – **Ablation study:** Comparison of the average incremental accuracy on CIFAR100 with 50 steps of the model when disabling parts of the complete PODNet's loss.

shows that our novel method of constraining the whole ConvNet is beneficial. Furthermore, applying only POD-spatial still beats the previous state of the art by a significant margin. Using both POD-spatial and POD-flat then further increases results with a large gain. We also compare the results with the Cosine classifier (Luo et al. 2018; Hou et al. 2019) against the Local Similarity Classifier (LSC) with NCA loss. Finally, we add LSC-CE: our classifier with multimode but with a simple cross-entropy loss instead of our modified NCA loss. This version brings to mind SoftTriple (Qian et al. 2019) and Infinite Mixture Prototypes (Allen et al. 2019), used in the different context of few-shot learning. The latter only considers the closest mode of each class in its class assignment, while LSC considers all modes of a class, thus, taking into account the intra-class variance. That allows LSC to decrease class similarity when intra-class variance is high (which could signal a lack of confidence in the class).

Spatial-based distillation We apply our distillation loss POD differently for the flat final embedding $\mathbf{h}$ (POD-flat) and the ConvNet's intermediate features maps $\mathbf{h}_\ell$ (POD-spatial). We designed and evaluated several alternatives for the latter whose results are shown in Table 3.4. Refer to Section 3.2.2.1 for their definition. In this table, all losses are with POD-flat ("*None*" is using only POD-flat). Note that We provide in the appendix (Section A.2) the same table without POD-flat.

Overall, we see that not using pooling results in bad performance (POD-pixels). Our final loss, POD-spatial, surpasses all others by taking advantages of the statistics aggregated from both spatial axis. For the sake of completeness, we also included losses not designed by us: GradCam distillation (Dhar et al. 2019) and Perceptual Style (Johnson et al. 2016). The former uses a gradient-based attention while the later – used for style transfer – computes a gram matrix for each channel.

Loss	NME	CNN
None	53.29	52.98
POD-pixels	49.74	52.34
POD-channels	57.21	54.64
POD-gap	58.80	55.95
POD-width	60.92	57.51
POD-height	60.64	57.50
POD-spatial	**61.40**	**57.98**
GradCam (Dhar et al. 2019)	54.13	52.48
Perceptual Style (Johnson et al. 2016)	51.01	52.25

Table 3.4. – **Comparison of distillation losses** based on intermediary features. All losses evaluated with POD-flat. We report the average incremental accuracy on CIFAR100 with 50 steps.

Forgetting and plasticity balance Forgetting can be drastically reduced by imposing a high factor on the distillation losses. Unfortunately, it will also degrade the capacity (its *plasticity*) to learn new classes. When POD-spatial is added on top of POD-flat (Table 3.4), we manage to increase the oldest classes' performance (+7 percentage points) while the newest classes' performance were barely reduced (-0.2*p.p.*). Because our loss POD-spatial constraints only statistics, it is less stringent than a loss based on exact pixels values as POD-pixel. The latter hurts the newest classes (-2*p.p.*) for a smaller improvement of old classes (+5*p.p.*). Furthermore, our experiments confirmed that LSC reduced the sensibility of the model to distribution shift, as the performance it brings was localized on the old classes.

Robustness of our model While previous results showed that PODNet improved significantly over the state-of-the-arts, we wish here to demonstrate here the robustness of our model to various factors. In Table 3.5, we compared how PODNet behaves against the baseline when the memory size per class M_{per} changes: PODNet improvements increase as the memory size decrease, up to a gain of 26.20*p.p.* with NME (resp. 13.42*p.p.* for CNN) with $M_{\text{per}} = 5$. Notice in our main experiments (Section 3.2.3.1), only 20 images per class are kept in the memory. Remark that when few images per class can be stored, a continual model must be rigid (to avoid forgetting old classes), while when numerous images can be stored, a continual model must be plastic (to learn efficiently new classes and stored examples). UCIR (Hou et al. 2019) and BiC (Wu et al. 2019) are respectively rigid- and plastic-oriented models, which can be seen from their good performance with respectively small and large rehearsal memory. On the other hand, PODNet excelled in all situations by efficiently trading rigidity and

M_{per}	5	10	**20**	50	100	200
iCaRL (Rebuffi et al. 2017c)	16.44	28.57	44.20	48.29	54.10	57.82
BiC (Wu et al. 2019)	20.84	21.97	47.09	55.01	62.23	**67.47**
UCIR (NME) (Hou et al. 2019)	21.81	41.92	48.57	56.09	60.31	64.24
UCIR (CNN)	22.17	42.70	49.30	57.02	61.37	65.99
PODNet (NME)	**48.37**	**57.20**	**61.40**	**62.27**	**63.14**	63.63
PODNet (CNN)	**35.59**	**48.54**	**57.98**	**63.69**	**66.48**	**67.62**

Table 3.5. – **Effect of the memory size** per class M_{per} on the models performance. Results from CIFAR100 with 50 steps, we report the average incremental accuracy.

plasticity with the POD loss. More experiments validating our model's robustness are provided in the appendix (Section A.2).

3.3 Ghost: avoiding pre-emptively forgetting via ghost features

Continual learning aims to perform equally well on past and present tasks. We present instead a challenging new setting, *prescient continual learning*, in which the model must perform well not only for present and past tasks, but also for *future* ones, both avoiding catastrophic forgetting (using a limited number of training samples for past classes), and giving the best possible estimates for the future classes (using no training samples at all). To make the setting possible, the model must know the classes and have some prior information about them: our desiderata is to make room for future classes to avoid interference with old classes once these new classes are actually learned. This could be done weakly by increasing the sparsity of the latent representation, but, given classes metadata, we present how to do it explicitly. In our setting, we know which labels we will encounter, but the training data for those labels arrive incrementally. For instance, in many real-world applications (*e.g.* fashion product classification), due to budget constraints, models are released incrementally, with partial classes and training data, despite the classes being known from the beginning, and being well-characterized by attributes.

We design a framework, nicknamed *Ghost*, which estimates the representation of future, yet unseen, classes using class attributes. We use these approximated representations to regularize the latent space so that forgetting could be avoided —or at least reduced— before even the introduction of disrupting new classes. This

is orthogonal to the proposed PODNet strategy where forgetting was reduced *after* a class introduction.

To develop our framework, we take inspiration from ZeroShot-Learning (ZSL) (Lampert et al. 2009; Y. Xian et al. 2019), which allows classifying examples from unseen classes by combining a vision model with an embedding possessing knowledge about the classes (*e.g.* a word embedding (Mikolov et al. 2013; Pennington et al. 2014) or an attribute matrix). Although several approaches exist for zero-shot learning, we will focus on generating a representation for the future classes (Bucher et al. 2017; Kumar Verma et al. 2018; Yongqin Xian et al. 2018). The framework of representation learning will allow us to integrate continual and zero-shot learning seamlessly, as we advance through the tasks and future classes become present classes, and then past classes. Moreover, we will be able to use *ghost features*, predicted features for the future classes, to make room in the representation space for future classes. All those goals all integrated into a simple, streamlined model due to a careful construction of the losses.

Our contributions are two-fold: (1) we propose a new challenging setting, *prescient* continual learning, where the model must perform well on past, present, and *future* classes; (2) we propose our *Ghost* model to address that setting, integrating continual and zero-shot learning into a coherent whole where future classes are pre-allocated, reducing their —yet not happened— forgetting.

3.3.1 Setting: Prescient Continual Learning

We propose an enriched experimental setting, prescient continual learning, that extend the class incremental setting used in Figure 3.3, in which each task is evaluated on *all classes* $\mathcal{C}^{1:T}$: past ($\mathcal{C}^{1:t-1}$), present ($\mathcal{C}^{t}$), and future ($\mathcal{C}^{t+1:T}$). In that challenging new setting, we must not only avoid the catastrophic forgetting of past classes (using the limited rehearsal training samples), but also give our best estimates for future classes (using no training samples at all). That will only be possible if we have some prior information about the classes, *e.g.* their hierarchy in a semantic network (like WordNet (Fellbaum 1998)), an associated word embedding (like Word2Vec (Mikolov et al. 2013)), or an attribute matrix. Such setting is illustrated in Figure 3.5. We will shorthand the set of past and present classes $\mathcal{C}^{1:t}$ as the *seen* classes, and the set of future classes $\mathcal{C}^{t+1:T}$ as the *unseen* classes. We denote individual samples by a superscript $\mathbf{x}^i$, the class label by a subscript $\mathbf{x}_c$, and on which parameters a loss is applied by a subscript $\mathcal{L}_\Theta$.

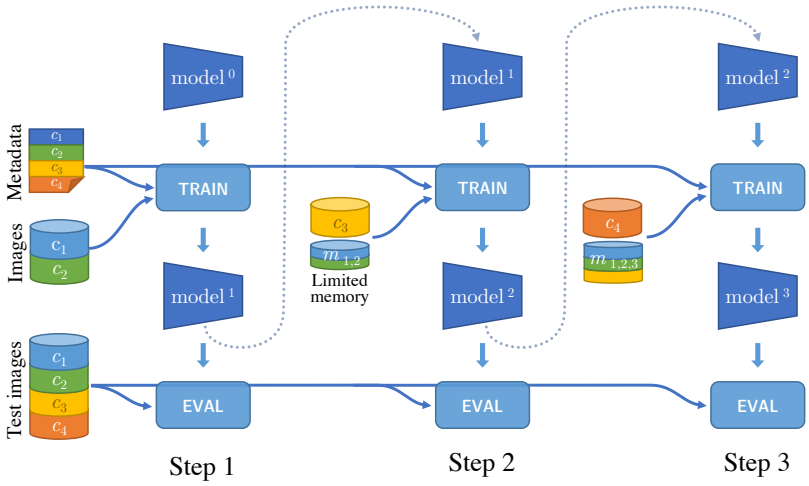


Figure 3.5. – **Prescient Continual Learning**. At each training task, we learn a new set of classes, but the model is evaluated on *all classes* — past, present, and future. The model has to avoid catastrophic forgetting of past classes (using a limited number of rehearsal training samples), as well as make a good guess for future classes (using no training samples at all).

3.3.2 Model

To address the setting described in the previous section, we propose our *ghost model*, comprising three components: a convolutional feature extractor f, a feature generator g, and a classifier *clf*. The feature extractor is the backbone of the model: it learns to extract a feature vector from actual samples that can be fed to the classifier. The generator learns the distribution of the features for all classes, aiming to generate plausible samples of features for the future classes. The classifier makes the final decision for all classes: past, present, and future. The classifier is trained on future classes with features sampled from the generator, which we call *ghost features* (since they must be "hallucinated" from the seen classes and some prior knowledge about the classes).

The base model, on which Ghost is built, is either PODNet (detailed in Section 3.2) or UCIR (Hou et al. 2019), both metric-based models.

3.3.2.1 Capacitating ghost model for future classes

The base model can deal with both present classes (with training samples constrained only by their availability in the training set) and past classes (with training samples severely constrained by the rehearsal memory). As discussed, the introduction of a distillation loss prevents catastrophic forgetting of the latter. We will now address *future classes*, with *no* training samples available. First, tak-

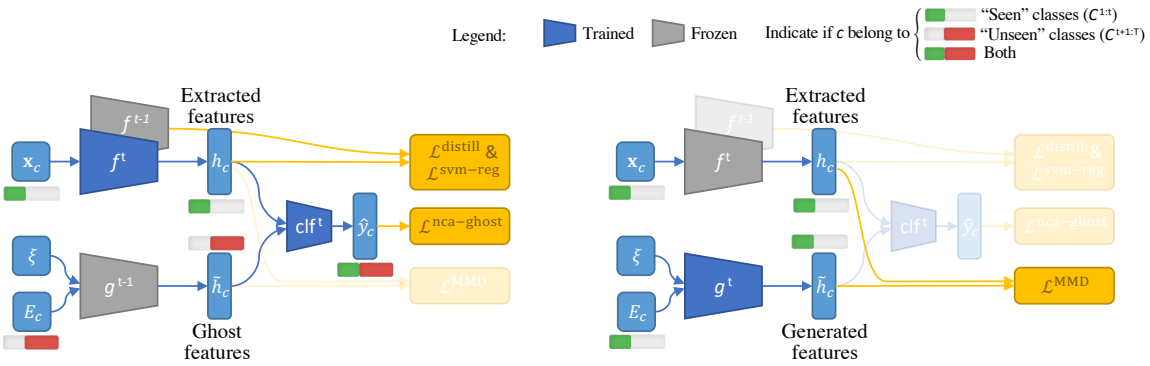
ing inspiration from zero-shot learning, we will use prior information about the classes to generate *ghost features*, plausible stand-ins for the unseen future classes' features. Next, we will adapt the classifier to incorporate those ghost features into the learning objective seamlessly. The representation learning framework will allow us to integrate the entire learning apparatus into one coherent loss.

Generator The generative model estimates the distribution of the unseen classes directly in terms of their features (instead of the input images). For the feature generation to work, we must have exploitable prior information about the classes, more precisely, we must be able to map the class labels c into a *class attribute space* that makes semantic sense. The exact way to perform that mapping will be data-dependent, but most often, either we will have an explicit set of attributes linked to each class (color, size, material, provenance, etc.), or we will be able to extract a latent semantic vector, using a technique like Word2vec (Mikolov et al. 2013; Pennington et al. 2014). The generator learns to link the attributes of the *seen* classes to the actual feature vectors extracted from the training samples of those classes. Thus, the first generator training must happen after the feature extractor (its ground-truth) is learned. The generator is fine-tuned after each task to handle distribution shift. Next, we ask the generator to draw random samples, using the attributes of the *unseen* classes, creating counterfeit features that we call *ghost features*. The strategy is agnostic to the generator model as long as it can be conditioned by class attributes. At present, as detailed in Section 3.3.3.2, we choose a Generative Moment Matching Network (Yujia Li et al. 2015): a shallow multi-layer perceptron conditioned by class attributes and a noise vector trained to minimize the Maximum Mean Discrepancy (Gretton et al. 2007; Gretton et al. 2012) between the estimated features $\tilde{\mathbf{h}}_c^t$ and the actual features $\mathbf{h}_c^t$ for each class c among the current classes $\mathcal{C}^t$, produced by the feature extractor f.

Complete classifier Remind that the parameters $\{\theta_c\,, \forall c \in \mathcal{C}^{1:t}\}$ on the representation-based classifier (Equation 3.9) may be interpreted as proxies for the classes $\mathcal{C}^{1:t}$. The base model for task t will, thus, learn $\mathcal{N}^{1:t} = |\mathcal{C}^{1:t}|$ such proxies, one for each of the seen classes. To extend the model for the unseen future classes, the complete classifier will learn $\mathcal{N}^{1:t} + \mathcal{N}^{t+1:T}$ proxies, which changes Equation 3.10 to:

$$\mathcal{L}_{\Theta_f, \Theta_{clf}}^{\text{nca-ghost}} = \left[-\log \frac{\exp\left(\hat{\mathbf{y}}_y - \delta\right)}{\sum\limits_{\substack{c \neq y \\ c \in \mathcal{C}^{1:t}}} \exp \hat{\mathbf{y}}_c + \sum\limits_{\substack{c \neq y \\ c \in \mathcal{C}^{t+1:T}}} \exp \hat{\mathbf{y}}_c} \right]_+ . \tag{3.11}$$

This classification loss maximizes the similarity $\hat{\mathbf{y}}_y$ (or, conversely, minimizes the distance) between sample feature $\mathbf{h}_y$ and correct class proxy θ_y in the numerator.



(a) Training of the classifier (b) Training of the generator

Figure 3.6. – **Procedure to train our model applied at each task/step**: (a) a complete classifier is learned with seen and unseen features ($\mathcal{L}^{\text{nca-ghost}}$). The feature extractor is protected from catastrophic forgetting ($\mathcal{L}^{\text{distill}}$), and constrained to separate seen classes from unseen/ghosts classes ($\mathcal{L}^{\text{svm-reg}}$). (b) Once a task is done, the generator is fine-tuned on the new latent space ($\mathcal{L}^{\text{MMD}}$) on seen classes. Notice that for the first and last tasks, the classifier does not use the ghost features.

In the denominator, the loss pushes away all wrong class proxies, from both seen and unseen (future) classes, by minimizing the similarities with $\mathbf{y}_c$, $\forall c \neq y$.

The participation of future classes in the classification loss has two effects. Most obviously, it allows the model to perform zero-shot-like guesses for those classes during test time. The representation-learning paradigm allows performing both continual and zero-shot learning seamlessly, as we advance through the tasks and future classes become present classes, and then past classes. Less evidently, but vitally important, the learning of proxies for the future classes makes room in the representation space for those classes, creating effective empty spaces that push away the actual features of the seen classes (due to the repulsive term in the denominator). As we advance through training, future classes become present, their ghost placeholders disappear, and they can neatly fit in the newly vacant region. Such a strategy reduces interference between classes throughout continual learning, and, as we will see in both visual and quantitative experiments, has long-range positive effects.

Naturally, the complete classifier has to be trained with samples from all classes. For seen classes, actual data is available from the training and rehearsal data. For unseen future classes data is not available, so we employ ghost features sampled from the generator. Note that ghost features are produced once per task by the generator and are kept fixed for the task duration.

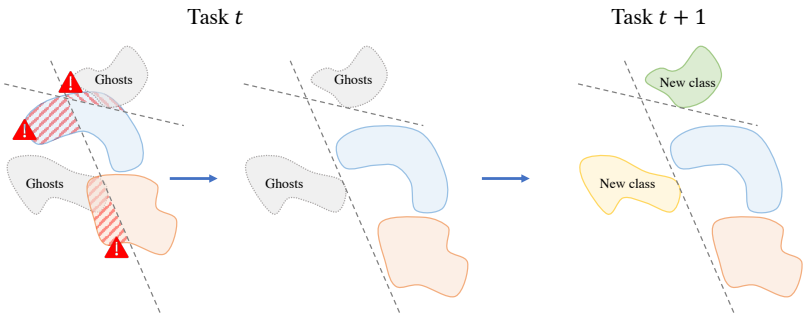


Figure 3.7. – **Latent-space regularization** establishes margin-based one-unseen-class vs all-seen-classes linear separations. Those separations are employed to directly condition the feature space, creating space for future unseen classes. In the following task, some unseen classes will become seen, and may occupy the feature space with less interference.

Latent-space regularization As explained above, our Ghost classification loss minimizes the intra-class distances and maximizes the inter-class distances. The loss enforces those constraints to all proxies regardless of whether they represent seen classes or not. We further promote an inter-class separation by optimizing the latent representation of seen classes to avoid overlapping with the representation of Ghosts. That loss constrains the features space directly and does not affect the proxies and the intra-class distances.

We based this regularization loss on SVM (Cortes and Vapnik 2015) for simplicity, but other methods could have similar behavior. To compute that loss, we learn binary one-unseen-class-Vs-all-seen-classes SVM classifiers, one for each unseen class. We employ a linear kernel, since the feature extractor and feature vector dimensionality (512) allows good linear separation, but more complex kernels could be used. Those SVM define hyperplanes $\mathbf{w}_c$ and biases b_c, $\forall c \in \mathcal{C}^{t+1:T}$, separating each unseen region from the mass of seen features $\mathbf{h}^{(i)}$ (Figure 3.7):

$$\mathcal{L}_{\Theta_f}^{\text{svm-reg}} = \frac{1}{\mathcal{N}^{t+1:T}} \sum_{c \in \mathcal{C}^{t+1:T}} [\mathbf{w}_c \cdot \mathbf{h}^t + b_c + \tau]_+ , \tag{3.12}$$

where $\mathbf{h}^t$ are seen features (classes in $\mathcal{C}^{1:t}$), $[\cdot]_+$ the hinge loss, and τ an additional margin (higher values of τ push seen features further away from the ghost regions, in practice, we set $\tau = 1$ to repel beyond the support vectors).

The margin-based regularization of Equation 3.12 refines the ghost classification loss of Equation 3.11. While the latter acts over the classifier conditioning the feature space indirectly via the action of the class proxies, the former acts directly over the latent/feature space and the feature extractor backbone that creates it.

The computational overhead of training several SVMs, detailed in the appendix (Section A.3), is negligible compared to the total training time.

Complete strategy All modules and losses fit neatly into the goal of learning continuously over seen and unseen classes. We train feature extractor (plus classifier) and generator in alternation. We train the latter to mimic the features of seen classes, and then ask it to extrapolate to unseen classes (ghost features). Ghost features allow us both to unify seen and unseen classes into a complete classifier ($\mathcal{L}^{\text{nca-ghost}}$), and to enforce early allocation in the feature space for unseen classes ($\mathcal{L}^{\text{svm-reg}}$). The complete loss, in addition to a distillation loss to counter-act catastrophic forgetting ($\mathcal{L}^{\text{distill}}$), is:

$$\mathcal{L} = \mathcal{L}^{\text{nca-ghost}}_{\Theta_f, \Theta_{clf}} + \lambda_1 \mathcal{L}^{\text{distill}}_{\Theta_f} + \lambda_2 \mathcal{L}^{\text{svm-reg}}_{\Theta_f}. \tag{3.13}$$

The model is illustrated in Figure 3.6 and Algorithm 3.1 is the algorithm of our model in pseudocode, showcasing a procedural view of the execution of one task.

Algorithm 3.1 Task procedure of the Ghost model

Require:
task id t
f^t, c^t, and g^t
Data from new task and rehearsal memory (X, y)

if $t = 1$ or $t = T$ **then**
  Train f^t and g^t with $\mathcal{L}^{\text{nca}}$ and $\mathcal{L}^{\text{distill}}$ on $(\mathbf{x}, y)$ and Ghost samples.
else
  Train f^t and g^t with $\mathcal{L}^{\text{nca-ghost}}$, $\mathcal{L}^{\text{svm-reg}}$ and $\mathcal{L}^{\text{distill}}$ on $(\mathbf{x}, y)$ and Ghost samples.
end if
Evaluate on all classes: $\mathcal{C}^{1:t} \cup \mathcal{C}^{t+1:T}$.
if $t \neq 1$ and $t < T - 1$ **then**
  Train g^t with $\mathcal{L}^{\text{MMD}}$ using attributes of seen classes $\mathbf{E}_c, \forall c \in \mathcal{C}^{1:t}$
  Generate Ghost samples for next task using attributes of unseen classes $\mathbf{E}_c, \forall c \in \mathcal{C}^{t+2:T}$
end if

3.3.3 Experiment Results

3.3.3.1 Pictorial experiments

Before running our full-scale experiments, we perform a set of experiments with a model that keeps the main components from the proposed model in a simplified form that will allow us to link quantitative performances to intuitive

visual plots of the feature spaces. For the experiments in this section we employ the MNIST (LeCun et al. 2010) dataset, with an initial task of 6 classes (digits 0 to 5), then two more tasks of two classes each; the feature extractor comprises two convolutional layers followed by a fully connected layer outputting a feature vector of only two dimensions — a purposeful choice to allow easy visualization of the feature space. Because MNIST classes have no attributes, we cannot apply zero-shot learning directly. Instead, we employ the features of actual images from the future classes instead of samples from the generator — which corresponds, in some ways, to have a perfectly calibrated generator. Those features are extracted once per task, and the feature extractor is never trained on unseen classes images. As we will see in Table 3.9, integrating the generator in our full-scale model outperforms using actual extracted features, so that necessary substitution does not exaggerate the abilities of this small-scale model. The losses used to train the small and the full-scale models are the same, but the SVM-based regularization was not employed since it made little sense for a 2D latent space.

The 2D feature space allows us to directly visualize the evolution of the feature space as the tasks progress, without the need for dimensionality reduction techniques that complicate the analysis (*e.g.* t-SNE (Maaten and G.E. Hinton 2008)). Figure 3.8 may be interpreted upfront: as the three tasks progress left-to-right, we see the evolution of the feature space on the base model (PODNet, on the top branch) and on the proposed model with ghost features (bottom branch). The base model presents a strong overlap between the initial classes, and the latter added 8 (orange) and 9 (dark purple). That comes partially from shape similarities ('8' is similar to '0' and '5'), partially from continual learning, and results in severe forgetting of old classes in favor of new ones. The proposed model organizes better the feature space, which is particularly visible between the second and third steps, where the early allocation of ghost zones for the future classes (displayed as empty black circles) is prominent. That better arrangement of the feature space increases the final accuracy from 44 to 66%, a 22 *p.p.* improvement. The small latent space of 2 dimensions explains the low performance for both models; a model that learns on all classes in one step (*i.e.* not in a continual setting) reaches only 69% of final accuracy. That said, our model shows a clear improvement both qualitatively in Figure 3.8 and quantitatively. More experiments of this kind can be found in the appendix (Section A.3).

3.3.3.2 Main experiments

Datasets & Protocols We perform our experiments on two datasets: AwA2 Y. Xian et al. (2019), with 50 animals categories, each with 85 attributes; and AP&Y Farhadi et al. (2009), with 32 everyday object classes, each with 64 attributes. We employ two experimental protocols: one typical for continual learning, following

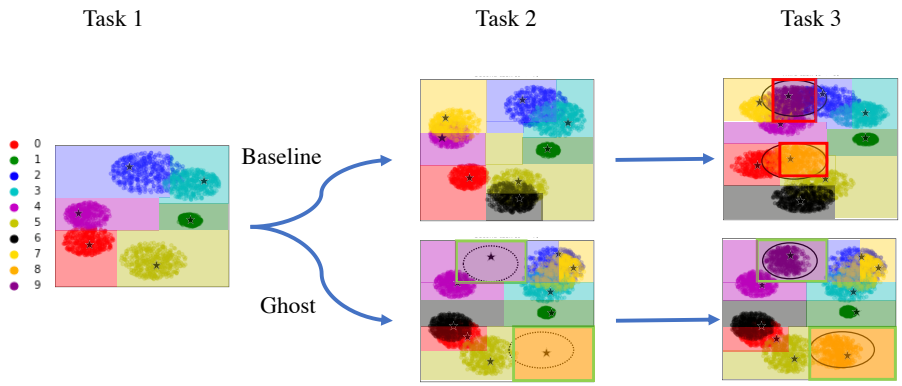


Figure 3.8. – **Small-scale PODNet** on MNIST with 3 steps (digits '0' to '5'; then '6' and '7'; then '8' and '9') with a features space of only two dimensions. The early incorporation of ghost features/proxies in the second task, denoted by dotted circles in the bottom row, enforces vacant space for those unseen classes. When filled in the third task (last column), there is less interference/overlap with previous classes. Such a strategy improves the final accuracy by 22 *p.p.*.

Hou et al. (2019), starting the first task with half the classes (i.e., 25 for AwA2, and 16 for aP&Y), then adding the remaining classes in evenly-sized tasks (5 tasks of 5 classes for AwA2, and 8 tasks of 2 classes for aP&Y); another inspired from zero-shot learning, following Y. Xian et al. (2019), starting with a standard selection of classes for each of the datasets (40 for AwA2, and 20 for aP&Y), and adding the remaining classes in small increments (5 tasks of 2 classes for AwA2, and 6 tasks of 2 classes for aP&Y). Our evaluation protocol is akin to the challenging and realist Generalized Zero-shot Learning (Scheirer et al. 2013; Chao et al. 2016) protocol with no information on whether a sample is from a seen or unseen class — but harsher, since classes are seen gradually, and training samples for past classes data are limited by rehearsal memory.

Base Models We evaluate our contributions on top of two different representation-based models, both based on ResNet18 (K. He et al. 2016) feature extractor backbones (with feature vector size of 512) and cosine classifiers. They differ on the distillation loss $\mathcal{L}_{\text{distil}}$ employed, the first model (PODNet) constraining the statistics of the intermediate features after each residual block (Section 3.2), and the second (UCIR) using Hou et al. (2019)'s distillation enforcing a cosine constraint on the final flat latent space. The former performs better than the latter, but both are improved by the innovations proposed with the Ghost framework. The cosine classifier has a single proxy/representative per class but could easily be generalized to multiple proxies. Implementation details are provided in the appendix (Section A.3).

Generator Following the work of Bucher et al. (2017) for zero-shot learning, our generator is a Generative Moment Matching Network (GMMN) (Yujia Li et al. 2015) $g^t(\xi, \mathbf{E}_c)$, which takes as inputs a Gaussian noise vector ξ and a class attributes vector $\mathbf{E}_c$, and outputs a sample from the estimated distribution of features for a class with the given attributes. In our experiments on AwA2 and AP&Y, the class attributes vectors are the average of the attributes for the training samples in the class. For each task t, the feature extractor f^t and the generator g^t are trained to minimize the Maximum Mean Discrepancy (MMD) (Gretton et al. 2007; Gretton et al. 2012) between the actual features of seen classes $\mathbf{h}_c = f^t(\mathbf{x}_c)\,, \forall c \in \mathcal{C}^{1:t}$ and their distribution on the generator $\tilde{\mathbf{h}}_c = g^t(\xi, \mathbf{E}_c)\,, \forall c \in \mathcal{C}^{1:t}$.

We train the generator and the main model (feature extractor and classifier) alternately, as shown in Figure 3.6. We always train the generator at the end of the task, after the feature extractor has adapted to the new distribution (Figure 3.6b). Then, we use the generator to produce ghost samples for the next task (unless we have reached the last task, with no unseen classes) which are fed to the feature extractor (Figure 3.6a).

When training the generator, we first extract features for all seen classes, with free access to the training samples for the present classes, but only a limited number (given by the rehearsal memory) for the past classes. For better numerical behavior, we scale each dimension of the extracted features to the interval [0, 1] before feeding them to the generator (and then re-scale the output of the generator back to the original intervals before feeding its samples to the classifier). The generator is trained to minimize the Maximum Mean Discrepancy (MMD) between the features of seen concepts $\mathbf{h}_c = f^t(\mathbf{x}_c)\,, \forall c \in \mathcal{C}^{1:t}$ and the representation it generates $\tilde{\mathbf{h}}_c = g^t(\xi, \mathbf{E}_c)\,, \forall c \in \mathcal{C}^{1:t}$:

$$\mathcal{L}^{\text{MMD}}_{\Theta_g} = \left\| \frac{1}{N}\sum_{i=1}^{N} \phi\big(\mathbf{h}_c^{(i)}\big) - \frac{1}{N}\sum_{j=1}^{N} \phi\big(\tilde{\mathbf{h}}_c^{(j)}\big) \right\|^2, \quad c \in \mathcal{C}^{1:t}\,, \tag{3.14}$$

with $\phi(\cdot)$ being a Gaussian kernel, and the superscript $\cdot^i$ denotes the i^{th} sample. The trained generator uses the attributes of the classes — which is the only information we have about them — to estimate sample features that we call *ghost features*. To better estimate the statistics, we use all the real features of a single seen class per batch. We denote the number of real features by N. The generator produces as many ghost features as real features.

Our main model has mechanisms to fight Catastrophic Forgetting, which we found were sufficient also to protect the generator. The feature extractor has an explicit distillation loss to prevent the problem, and since its output is used to train the generator, the latter is also protected. We considered applying a distillation loss to the generator, trying to minimize a drift between the produced

	AwA2 25 classes + 5 × 5 classes		aP&Y 16 classes + 8 × 2 classes	
	PODNet	UCIR	PODNet	UCIR
Baseline	62.92 ± 0.12	54.80 ± 0.40	58.64 ± 0.66	43.42 ± 0.21
+ $\mathcal{L}^{\text{nca-ghost}}$	68.31 ± 0.36	57.88 ± 0.27	62.08 ± 0.25	50.23 ± 0.29
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	68.46 ± 0.47	58.08 ± 0.46	62.73 ± 0.60	50.91 ± 0.56

Table 3.6. – **Continual Accuracy** on AwA2 and aP&Y for PODNet and UCIR.

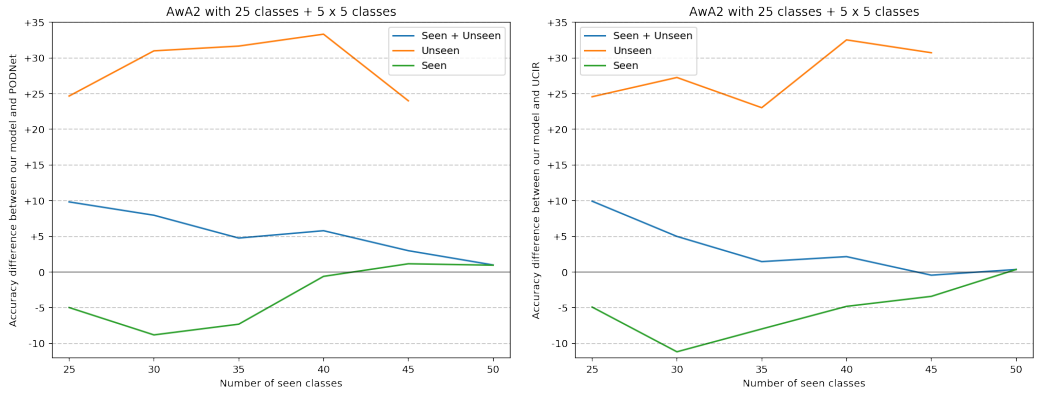
	AwA2 25 classes + 5 × 5 classes		aP&Y 16 classes + 8 × 2 classes	
	PODNet	UCIR	PODNet	UCIR
Baseline	77.63 ± 0.06	67.07 ± 0.81	57.80 ± 0.97	42.23 ± 1.34
+ $\mathcal{L}^{\text{nca-ghost}}$	78.70 ± 0.46	67.43 ± 0.08	62.47 ± 0.40	44.17 ± 1.48
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	79.08 ± 0.53	67.53 ± 0.45	63.30 ± 0.98	45.97 ± 0.26

Table 3.7. – **Final Accuracy** on AwA2 and aP&Y for PODNet and UCIR.

features of the previous and current generator. We measured such drift according to several metrics: cosine similarity, Kullback-Leiber divergence, L2 distance, and maximum mean discrepancy. The first metric, cosine similarity, gave the best results. However, as the generator was well protected, we gained at most 0.50 Continual Accuracy *p.p.* on AwA2.

Continual Learning with Future Classes For continual learning, it is usual to take into account the model's performance as it evolves. We adapt the traditional average incremental accuracy (Rebuffi et al. 2017c) (Equation 2.7 in Figure 2.3) to take into account *all* classes, including the future ones, and call that metric *continual accuracy*: the average of accuracy over all seen classes after each task. The results appear in Table 3.6, which shows, for the datasets and protocols explained in the top of this section, the performance for our two base models (Hou et al. 2019), and the improvements on those base models as we implement our proposed model, with and without the SVM latent-space regularization refinement. The ability to guess a future class' representation brings large improvements on both datasets, for both base models. The SVM-based regularization refinement also improves the results, by up to 0.65 *p.p.*.

Final Accuracy Once we reach the final task, the proposed model ability to guess future classes provides no advantage due to all classes being now seen. Still, as shown in Table 3.7 — where the metric is simply the accuracy at the final task of each run (Equation 2.6 in Figure 2.3) — the proposed method outperforms



(a) Ghost vs PODNet. (b) Ghost vs UCIR

Figure 3.9. – **Difference of accuracy** over all classes, only seen classes, and only unseen classes Ghost model vs base models on AwA2.

	AwA2 40 classes + 5 × 2 classes		aP&Y 20 classes + 6 × 2 classes	
	Continual	Final	Continual	Final
PODNet	82.84 ± 0.10	84.70 ± 0.10	67.57 ± 0.41	65.23 ± 0.50
+ $\mathcal{L}^{\text{nca-ghost}}$	84.99 ± 0.17	86.57 ± 0.49	68.80 ± 0.98	67.93 ± 1.24
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	84.47 ± 0.15	85.73 ± 0.40	69.02 ± 0.46	67.97 ± 0.60

Table 3.8. – **Further experiments** where the initial task size correspond to standard zero-shot seen classes Y. Xian et al. 2019. We report Continual and Final Accuracies for PODNet on AwA2 and aP&Y.

the baselines, due to a better organization of the feature space. Although the numerical advantages in this table are smaller than in the previous one, these results are consequential, showing that the ability of the proposed model of incorporating knowledge about the classes is useful beyond the zero-shot scenario. Again, the SVM-regularization refinement helps by up to 1.80 *p.p.*.

Model Evolution To showcase how the models evolve, plots contrasting the proposed methods with each base model (PODNet and UCIR) task by task appear in Figure 3.9. The plots show how, on early tasks, the main advantage of the proposed model is its ability to guess on the future classes, while on the final task no future classes remain, but the proposed model still keeps a (more modest, but definitive) advantage.

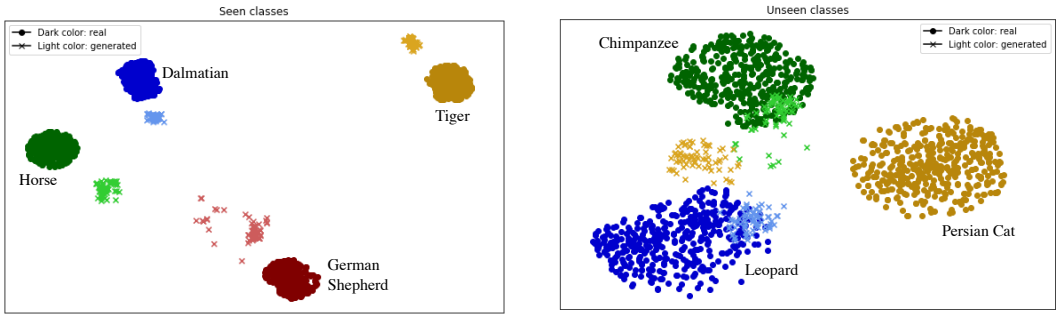
Zero-shot-like Initial Task Setting This set of experiments (Table 3.8) is intended for comparison with zero-shot learning benchmarks (Y. Xian et al. 2019),

	AwA2 25 classes + 5 × 5 classes		aP&Y 16 classes + 8 × 2 classes	
	Continual	Final	Continual	Final
Our model	68.46 ± 0.47	79.08 ± 0.53	62.73 ± 0.60	63.30 ± 0.98
with real features	67.65 ± 0.50	78.83 ± 0.31	61.88 ± 0.52	61.70 ± 0.26
Partial oracle	72.94 ± 0.25	84.60 ± 0.28	63.81 ± 0.29	68.03 ± 1.42
Full oracle	—	95.40 ± 0.02	—	97.40 ± 0.30

Table 3.9. – **Comparison of generated ghost features vs. actual features** extracted from future classes' samples with PODNet on AwA2 and aP&Y.

which always use the same split of seen/unseen classes for a given dataset. Our first task in the continual learning contains the classes defined in zero-shot benchmark as seen, and we learn next, in small increment, the remaining classes, i.e., those defined in the zero-shot benchmark as unseen. Because the initial task is larger than previously, fewer future classes remain, and the base models have better performance. Still, the proposed method improves both base methods in both datasets significantly. The setting proposed is different from the — markedly less challenging — setting appearing in Kankuekul et al. (2012) and Xue et al. (2017), where the set of unseen classes is fixed, and only more seen classes are added incrementally, without any sample limitations given by rehearsal memory.

Generator Validation The generator approximates the feature extractor for the unseen future classes. To validate its effectiveness, we replace the generated features with the actual features from the future images. This form of "cheating", of course, is not possible in *actual* real-world scenarios, but serves as a metric. Table 3.9 shows the comparison, with the surprising result that generated features performed better than the actual features from samples (respectively first and second row). Note that the latter are extracted once per task. We hypothesize it explains the score difference because the feature extractor was never adapted for the unseen classes distribution. The "oracle" experiments in the third and fourth rows in Table 3.9 establish an upper bound for what we could achieve by "cheating" around the experimental protocol restrictions. The partial oracle from third row is the same model as the second row, fine-tuning the feature extractor with samples coming from the future. The full oracle of the fourth line uses all images from all classes without any restriction in a single task. Despite the partial oracle has full access to real future data, we stress that our model's performance with generated future data is close to this upper bound.



(a) Generator interpolation for *seen* classes.

(b) Generator extrapolation for *unseen* classes.

Figure 3.10. – **t-SNE of the latent space**. Dark colors indicate real features, while lighter colors denote their generated homologous. Real features extracted with f^t, and ghost features sampled from g^t. Generation in (a) is both well-located and tightly bound because the GMMN was trained to approximate those seen classes. In (b), the generator is asked to extrapolate to unseen classes only from their attributes, resulting in more spread features — still surprisingly, in general, well-located. Notice that the even the real features in (b) gets more spread, since the feature extractor was never trained on those classes.

Ghost visualization We inspect the ghost samples created by our generator with visualization in Figure 3.10: we compare the features extracted from the actual images with their generated homologous, produced by the GMMN on the AwA2 dataset. To allow visualization, we reduce the dimensionality of the features to two with t-SNE. In Figure 3.10a, we compare, at task t, the actual features (in dark colors) to generated features (in light colors) for seen classes. The generated features are, in most cases, near the clusters of real features, and all clusters — real and generated are tightly bound. We then compare in Figure 3.10b the real and generated features on unseen classes. Compared to Figure 3.10a, even the real features show a bigger spread, due, we believe, to the feature extractor f^t not having met those unseen images. The generated features are also more spread but are still, in general, reasonably placed, indicating the ability of the generator to extrapolate — at least partially — the features for unseen classes from their attributes.

3.4 Conclusion

In this chapter, we covered our works on regularizations of the feature space. We proposed to regularize intermediate features instead of model's outputs as

done by the vast majority of the literature. In this context, we tackle catastrophic forgetting via two approaches.

The first approach, **PODNet, reduces the drift between the old and new models by constraining statistics of the intermediate feature**s. We show that a naive constraint balances poorly the plasticity-rigidity trade-off. Thus, we propose carefully designed pooling to directly exploit the spatial nature of the images. By constraining long-range statistics of the horizontal and vertical axes, we reduce drastically forgetting, especially in challenging settings with a large number of tasks with few classes per task. Furthermore, we design a robust metric-based classifier that exploits the multi-modality of the classes.

The second approach, **Ghost, avoids pre-emptively forgetting by estimating the future locations of yet unseen classes**. This model, an extension of PODNet when class attributes are available, incorporates capabilities of zero-shot learning into the continual learning model in a seamless way using the paradigm of representation learning. We refined that model with a novel SVM-based regularization loss acting over the feature space to reinforce exclusion zones, reserved for future classes.

CHAPTER 4

CONTINUAL SEMANTIC SEGMENTATION

Chapter abstract

In Class Incremental Learning (CIL), most of the literature focuses on image classification. In ***Continual Semantic Segmentation (CSS)****, the continual setting is the same (incremental addition of new classes), but the implications are wildly different. In Semantic Segmentation, an image has one label per pixel. Therefore, an image could contain pixels from old, current, and future classes.*

In this chapter, we present a new approach for Continual Semantic Segmentation: We first highlight the main challenges of this domain: an important ***catastrophic forgetting*** *linked to the higher complexity of segmentation images, and a* ***background shift*** *where images are partially labeled: the new classes' ground-truth labels are present but all other classes are unlabeled despite their presence in an image. To tackle the background shift, we design an uncertainty-based hard pseudo-labeling loss and an efficient object rehearsal method. To reduce the catastrophic forgetting of old classes, we also propose a multi-scale distillation loss inspired by our previous POD.*

We evaluate our model on multiple datasets (e.g. VOC, ADE, Cityscapes), propose new benchmarks, and perform extensive quantitative and qualitative ablations validating the efficiency of our strategy.

The work in this section has led to the writing of two papers:

- *Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021a). "PLOP: Learning without Forgetting for Continual Semantic Segmentation". In:* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
- *Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021b). "Tackling Catastrophic Forgetting and Background Shift in Continual Semantic Segmentation". In:* Under review at TPAMI

Contents

4.1 Introduction

Semantic segmentation aims to assign a label to each pixel of an image. It allows the prediction of multiple objects in the same image, and moreover, their exact position and shape. This task recently flourished (Tao et al. 2020; Hang Zhang et al. 2020; L. Chen et al. 2018) with larger datasets with thousands of fully annotated images (B. Zhou et al. 2017; Neuhold et al. 2017), increased computational power, and larger attention (H. Wang et al. 2020). Unfortunately, the recent research in this area is often impracticable for real-life applications: they mostly **need fully annotated data and require to be retrained from scratch if a new class is added to the dataset.** Ideally, one would wish to regularly expand a dataset, only adding and labeling new classes and updating the model in accordance. This setup, referred here as **Continual Semantic Segmentation (CSS)**, has emerged very recently for specialized applications (Ozdemir et al. 2018; Ozdemir and Goksel 2019; Tasar et al. 2019) before being proposed for general segmentation datasets (Michieli and Zanuttigh 2019; Cermelli et al. 2020).

In particular, in this chapter, we argue that two problems arise when performing CSS with Convolutional Neural Networks (ConvNets). The first one, inherited from continual learning, is **catastrophic forgetting** (Robins 1995), already well

detailed in the previous chapters including Chapter 2. One of the most efficient methods to avoid forgetting is rehearsal learning (Section 2.4.1) where we store a few images from previous tasks. Unfortunately, this solution has difficulty in the context of segmentation where multiple classes can be in the same image, and where the storage cost is high and images are partially labeled. The second challenge, specific to CSS, is the semantic shift of the background class. In a traditional semantic segmentation setup, all object categories are predefined, and the "background" class contains pixels that do not belong to any of these classes. However, in CSS, the background contains pixels that do not belong to any of the *current* classes. Thus, for a specific learning step, the background can contain old classes as well as future classes, not yet seen by the model. Thus, if nothing is done to distinguish pixels belonging to the real background class from old class pixels, this **background shift** phenomenon risks exacerbating the catastrophic forgetting even further (Cermelli et al. 2020). This issue also has an impact on the selection of the old data we want to store. Because some currently learned classes are annotated as background in the old data, this may degrade the performance of these classes if one naively treat them as background to fine-tune the current model.

We tackle the first challenge of catastrophic forgetting by designing a constraint enforcing a similar behavior between the old and current models. Specifically, we leverage intermediary representations of the convolutional networks to ensure that similar patterns are extracted through time. **This feature-based constraints, called Local POD, fully exploits the global and local scale necessary to semantic segmentation through a multi-scale design**. The second challenge, background shift, is greatly alleviated by **a confidence-based pseudo-labeling strategy to retrieve old class pixels within the background**. For instance, if a current ground truth mask only distinguishes pixels from class `sofa` and `background`, our approach allows assigning old classes to background pixels, *e.g.* classes `person`, `dog` or `background` (the semantic class). We name PLOP the model exploiting those two contributions. We then propose **an extension called PLOPLong that aims to excel on long continual learning scenarios**. This new model exploits cosine normalization to adapt the classifier and the Local POD resulting in improving robustness to the discrepancy between old and new classes. Moreover, PLOPLong features a modified batch normalization which reduces the sensitivity of the model to moving statistics seen across tasks in continual learning. Finally, we are the first to investigate rehearsal learning in the frame of CSS. We propose a baseline approach that is based upon rehearsing complete images. However, in practice, this seemingly classic rehearsal is not as trivial in our context as the labeling is partial: hence, once again, we need to complete the rehearsed images after each step by filling the background with the missing old classes. While significantly improving performances, we show that such approach has two main

drawbacks. First, it is memory intensive, as the whole images shall be rehearsed at each CSS step. Second, despite a large number of pixels in the images, we argue that the images contain few, sparse useful information. Consequently, we design **a novel rehearsal method, that we named "Object Rehearsal", that consists in selecting only non-regular objects-centered patches as candidates for rehearsal**. Those objects, belonging to old classes, are combined into the images of new classes *via* careful image editing. We empirically show that this new rehearsal method surpasses classic rehearsal with pseudo-labeling, while being up to 146x times more memory efficient.

From a practical point of view, our proposed methods (PLOP, PLOPLong, and Object Rehearsal) showed three important results. First, we achieve the state-of-the-art performance on several challenging datasets. Secondly, we propose several novel scenarios to further quantify the performances of CSS methods when it comes to long term learning, class presentation order, and domain shift. Last but not least, we show that our model contributions largely outperform every CSS approach in these scenarios.

To sum it up, our contributions are four-folds:

- We propose a multi-scale spatial distillation loss to better retain knowledge through the continual learning steps, by preserving long- and short-range spatial statistics, avoiding catastrophic forgetting.

- We introduce a confidence-based pseudo-labeling strategy to identify old classes for the current background pixels and deal with background shift.

- We propose PLOPLong, a carefully designed refinement of our method for dealing with long CSS scenarios. The extension comes from an adaptation of PLOP's classifier and Local POD distillation as well as batch re-normalization for better handling of both catastrophic forgetting and background shift, respectively.

- We design a novel memory-efficient Object rehearsal learning procedure that consists in storing and carefully pasting objects through selective erasing of foreground objects. It results in better performance for a fraction of the memory cost imposed by classic rehearsal.

Additionally, we show that our PLOP significantly outperforms state-of-the-art approaches in existing scenarios and datasets for CSS, as well as in several newly proposed challenging benchmarks on new datasets. Furthermore, we show that PLOPLong leads to superior performances on longer CSS scenarios. Last but not least, we prove that CSS models' performance can be greatly improved

when rehearsal learning is an option. In such cases, the proposed Object rehearsal allows reaching high accuracies with a small memory footprint.

4.2 Related Work

Continual Semantic Segmentation is a relatively young field that started getting traction following Michieli and Zanuttigh (2019) and Cermelli et al. (2020). However, this field is at the intersection of many popular topics. Therefore, we start this section with an overview of recent advances in segmentation. We then follow with a more in-depth discussion of existing approaches to CSS. For a thorough discussion of Continual Learning, please refer to Chapter 2.

Semantic Segmentation methods based on Fully Convolutional Networks (FCN) (Long et al. 2015; Sermanet et al. 2014) have achieved impressive results on several segmentation benchmarks (Everingham et al. 2015; Cordts et al. 2016; B. Zhou et al. 2017; Caesar et al. 2018). These methods improve the segmentation accuracy by incorporating more spatial information or exploiting contextual information specifically. Atrous convolution (L.-C. Chen et al. 2018a; Mehta et al. 2018) and encoder-decoder architecture (Ronneberger et al. 2015; Noh et al. 2015; Badrinarayanan et al. 2017) are the most common methods for retaining spatial information. Examples of recent works exploiting contextual information include attention mechanisms (Yuan and J. Wang 2018; Hengshuang Zhao et al. 2018; Fu et al. 2019; Z. Huang et al. 2019; Yuan et al. 2020; Tao et al. 2020; Hang Zhang et al. 2020), and fixed-scale aggregation (H. Zhao et al. 2017; L.-C. Chen et al. 2018a; L. Chen et al. 2018; Hang Zhang et al. 2018).

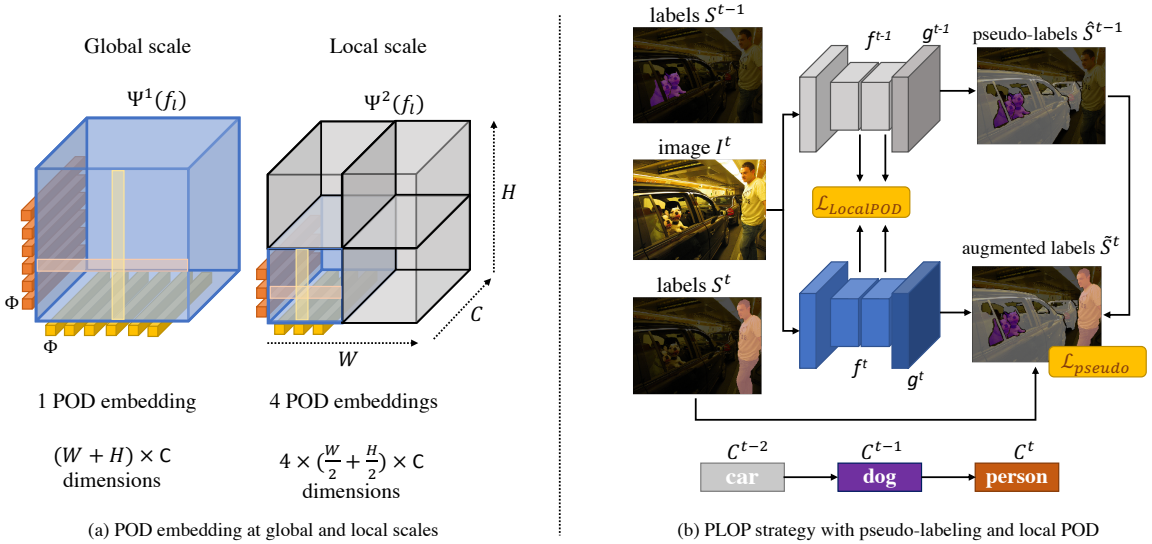
Continual Semantic segmentation Despite enormous progress in the two aforementioned areas respectively, segmentation algorithms are mostly used in an offline setting, while continual learning methods generally focus on image classification. Recent works extend existing continual learning methods (Z. Li and Hoiem 2016; Hou et al. 2019) for specialized applications (Ozdemir et al. 2018; Ozdemir and Goksel 2019; Tasar et al. 2019) and general semantic segmentation (Michieli and Zanuttigh 2019). The latter considers that the previously learned categories are properly annotated in the images of the new dataset. This is an unrealistic assumption that fails to consider the background shift: pixels labeled as background at the current step are semantically ambiguous, in that they can contain pixels from old classes (including the real semantic background class, which is generally deciphered first) as well as pixels from future classes. Cermelli et al. (2020) propose a novel classification and distillation losses. Both handle the

background shift by summing respectively the old logits with the background logits and the new logits with the background. We argue that a distillation loss applied to the model output is not strong enough for catastrophic forgetting in CSS. Furthermore, their classification loss does not preserve enough discriminative power w.r.t the old classes when learning new classes under background shift. We introduce our PLOP framework that solves more effectively those two aspects. Yu et al. (2020) proposed to exploit an external unlabeled dataset in order to do self-training with a pseudo-labeling loss; we show that our model, while not designed with this assumption in mind, can outperform their performance. Cermelli et al. (2021) create a novel setting of continual *few-shots* segmentation, we implement their method in our setting and draw inspiration from it to further improve PLOP. Michieli and Zanuttigh (2021) draw inspiration from the metric learning literature to conceive a model for continual segmentation that exploits prototypes updated with an exponential moving average of the mean batch features.

Positioning: Contrary to previous works in continual segmentation (Michieli and Zanuttigh 2019; Cermelli et al. 2020) which reduced slightly forgetting through a distillation of the probabilities, we propose a stronger constraint based on global and local statistics extracted from intermediary features. Moreover, background shift is often not considered (Michieli and Zanuttigh 2019) or only weakly tackled (Cermelli et al. 2020), while we propose to eliminate it through segmentation maps completion with pseudo-labeling. Finally, none of the work proposed rehearsal methods for CSS, while we propose a non-trivial method based on image rehearsal and further improve it with a more data-efficient method based with object rehearsal.

4.3 PLOP and PLOPLong models

The model description is organized as follows: we first detail the continual protocol and the notations. Then, we tackle the issue of catastrophic forgetting by designing an adapted distillation loss, and we alleviate the background shift by proposing an uncertainty-based pseudo-labeling. Drawing ideas from the continual learning literature, we propose an extension of PLOP specialized for long-range continual training that we nickname PLOPLong. Finally, we detail the limits of rehearsal learning in segmentation, propose a naive adaptation to the problem, and then deliver our carefully designed method.



(a) POD embedding at global and local scales

(b) PLOP strategy with pseudo-labeling and local POD

Figure 4.1. – Local POD details and the complete PLOP strategy. (a) Local POD consists in POD embeddings compute at multiple scale. The global scale aggregates statistics across the whole features maps while the local scale focuses on finer details. (b) The model incrementally learns new classes (*e.g.* `car`, `dog`, `person`). Only the current class (`person`) is labeled while previous classes are folded into the background. We use the previous model $g^{t-1} \circ f^{t-1}$ to generate pseudo-labels $\hat{\mathcal{S}}^{t-1}$ regarding the old classes to alleviate this ambiguity, and complete the labels $\mathcal{S}^t$ which are then used as ground-truth in $\mathcal{L}_{\text{pseudo}}$. The Local POD distillation is applied at multiple levels of the features extractors f^{t-1} and f^t.

4.3.1 Framework and notations

For the notations, please refer to the Notations. The slight difference with previous chapters, is that instead of working on pair of a classification image with a single label $(\mathbf{x}, y)$, we will work on pairs of image and segmentation maps where each pixel has a class $(\mathbf{x}^t, \mathbf{y}^t)$. Here, the task identifier t is even more important because the segmentation map $\mathbf{y}$, for a same image, will evolve through the tasks. Indeed, in the considered benchmarks (detailed later in Section 4.5), an image is only labeled for the current classes $\mathbf{y}^t$. The previous $\mathcal{C}^{1:t-1}$ or future classes $\mathcal{C}^{t+1:T}$ the image may contain are not labeled, and considered as the special class "*background*". However, at test-time, a model at step t must be able to discriminate between all the classes that have been seen so far, *i.e.* $\mathcal{C}^{1:t}$.

This leads us to identify two major pitfalls in CSS: the first one, inherited from continual learning, is catastrophic forgetting (Robins 1995). It suggests that a network will completely forget the old classes $\mathcal{C}^{1:t-1}$ when learning the new ones $\mathcal{C}^t$.

Furthermore, catastrophic forgetting is aggravated by the second pitfall, specific to CSS, that we call background shift: at step t, the pixels labeled as background are indeed ambiguous, as they may contain either old (including the real background class, predicted in $\mathcal{C}^1$) or future classes.

We define our model at step t as a composition of a feature extractor $f^t(\cdot)$ (a ResNet 101 (K. He et al. 2016) backbone and a classifier $g^t(\cdot)$. The output predicted segmentation map can then be written $\hat{\mathbf{y}}^t = g^t \circ f^t(\mathbf{x})$. We denote the intermediate features at each layer of the feature extractor $\mathbf{h}_l^t = f_l^t(\cdot)\,,\ l \in \{1, \dots L\}$. Finally, we denote the set of learnable parameters of f^t and g^t as Θ^t.

4.3.2 Overcoming catastrophic forgetting in CSS with local distillation

In this section, we propose to tackle the issue of catastrophic forgetting in continual learning in general and in CSS in particular. An effective method for doing so involves setting constraints between the old $(g^{t-1} \circ f^{t-1})$ and current $(g^t \circ f^t)$ models. These constraints aim at enforcing a similar behavior between both models and in turn reduce the loss of performance on old classes. A common such constraint is based on applying knowledge distillation Geoffrey Hinton et al. (2015) and Z. Li and Hoiem (2016) between the predicted probabilities of both models. When applied to CSS, such distillation loss must be carefully balanced to find a good trade-off between rigidity (*i.e.* too strong constraints, resulting in

not being able to learn new classes) and plasticity (*i.e.* enforcing loose constraints, which can lead to catastrophic forgetting of the old classes).

In Chapter 3, more precisely in Section 3.2, we designed POD, short for Pooled Output Distillation. Rather than solely constraining the output model probabilities, POD enforces consistency between intermediary statistics of both models. Recall (Equation 3.5), for a feature map $\mathbf{x}$, we define a POD embedding $\Phi(\cdot)$ as:

$$\Phi(\mathbf{x}) = \left[\frac{1}{W} \sum_{w=1}^{W} \mathbf{x}[:, w, :] \middle\| \frac{1}{H} \sum_{h=1}^{H} \mathbf{x}[:, :, h] \right] \in \mathbb{R}^{(H+W)\times C}, \tag{4.1}$$

where $[\cdot \,\|\, \cdot]$ denotes concatenation over the channel axis. The POD embedding is thus computed as the concatenation of the $H \times C$ width-pooled slices and the $W \times C$ height-pooled slices of $\mathbf{x}$ and captures long-range statistics across the whole features maps. The POD distillation loss then consists in minimizing the $\mathcal{L}_2$ distance between POD embeddings computed at several layers $l \in \{1, \dots, L\}$, w.r.t the current model parameters Θ^t:

$$\mathcal{L}_{\text{pod}}(\Theta^t) = \frac{1}{L} \sum_{l=1}^{L} \left\| \Phi(\mathbf{h}_l^t) - \Phi(\mathbf{h}_l^{t-1}) \right\|^2. \tag{4.2}$$

POD yielded state-of-the-art results in continual learning for image classification, especially when large numbers of tasks are considered, a case where the aforementioned plasticity-rigidity trade-off becomes even more crucial. Another interest arises in the context of CSS: the long-range statistics computed across an entire axis (horizontal or vertical) which reminds concurrent work on attention for segmentation (H. Wang et al. 2020; Zilong Huang et al. 2020; Park and Heo 2020) which aim to enlarge the receptive field through global attention/statistics (H. Wang et al. 2020). In the frame of classification, it is, to a certain extent, necessary to discard spatial information through global pooling. However, semantic segmentation requires the preservation of both long-range and short-range statistics, making a distillation loss such as POD suboptimal for that purpose.

Following this reflection, and inspired by the multi-scale literature (Lazebnik et al. 2006; K. He et al. 2014), we design a distillation loss, called Local POD retaining the long-range spatial statistics while also preserving the local information. The proposed Local POD consists in computing the width and height statistics at different scales $\{1/2^s\}_{s=0\dots S}$, as illustrated in Figure 4.1 (a). At a given level l of the feature extractor, s^2 POD embeddings are computed per scale s and concatenated:

$$\Psi^s(\mathbf{x}) = \left[\Phi(\mathbf{x}_{1,1}^s) \| \dots \| \Phi(\mathbf{x}_{s,s}^s) \right] \in \mathbb{R}^{(H+W)\times C}, \tag{4.3}$$

where $\forall i = 1 \dots s$, $\forall j = 1 \dots s$, $\mathbf{x}^s_{i,j} = \mathbf{x}[:, iW/s : (i+1)W/s, jH/s : (j+1)H/s]$ is a sub-region of the embedding tensor $\mathbf{x}$ of size $W/s \times H/s$. Then, we concatenate (along the channel axis) the Local POD embeddings $\Psi^s(\mathbf{x})$ of each scale s to form the final embedding:

$$\Psi(\mathbf{x}) = \left[\Psi^1(\mathbf{x}) \| \dots \| \Psi^S(\mathbf{x})\right] \in \mathbb{R}^{S \times (H+W) \times C} . \tag{4.4}$$

Similarly to POD, we compute Local POD embeddings for every layer $l \in \{1, \dots, L\}$ of both the old and current models. The resulting loss is thus:

$$\mathcal{L}_{\text{LocalPod}}(\Theta^t) = \frac{1}{L} \sum_{l=1}^{L} \left\| \Psi(f^t_l(I)) - \Psi(f^{t-1}_l(I)) \right\|^2 . \tag{4.5}$$

Thus, notice that the first scale of Local POD ($1/2^0$) is similar to the original POD and models long-range dependencies across the entire image. The subsequent scales ($s = 1/2^1, 1/2^2 \dots$), enforce short-range dependencies. Thus, the proposed Local POD tackles the problem of catastrophic forgetting by modeling and preserving long and short-range statistics between the old and current models, throughout the CSS steps.

4.3.3 Pseudo-labeling to fix background shift

In addition to catastrophic forgetting, a successful CSS approach shall handle the background shift problem, thus shall take into account the ambiguity of pixels labeled as background at each step. We propose a pseudo-labeling strategy that "*completes*" the ambiguous background labels. Pseudo-labeling (D.-H. Lee 2013) is commonly used in domain adaption for semantic segmentation (Vu et al. 2019; Yunsheng Li et al. 2019; Zou et al. 2018; Saporta et al. 2020) where a model is trained to match both the labels of a source dataset and the pseudo-labels (usually obtained using the same predictive model, in a self-training fashion) of an unlabeled target dataset. In this case, the knowledge acquired on the source dataset helps the model to generate labels for the target dataset. In the frame of CSS, at each step, we use the predictions of the old model to decipher previously seen classes among the ambiguous background pixels, as illustrated in Figure 4.1. The pseudo-labeling relies on the previous model which can be uncertain for some pixels due to inherent bias to the optimization and because of the forgetting. Therefore, in order to avoid propagating errors through incorrect pseudo-labels, we filter out the most uncertain ones based on an adaptive entropy-based threshold.

Formally, let $\mathcal{N}^t = \text{card}(\mathcal{C}^t)$ the cardinality of the current classes, excluding the background class. Let $\hat{\mathbf{y}}^t \in \mathbb{R}^{W,H,1+\mathcal{N}^{1:t}}$ denotes the predictions of the current model (which includes the real background class, all the old classes as well as the current ones). We define $\tilde{\mathbf{y}}^t \in \mathbb{R}^{W,H,1+\mathcal{N}^{1:t}}$ the target at step t, computed using the one-hot ground-truth segmentation map $\mathbf{y}^t \in \mathbb{N}^{W,H,1+\mathcal{N}^t}$ at step t as well as pseudo-labels extracted using the old model predictions $\hat{\mathbf{y}}^{t-1} \in \mathbb{R}^{W,H,1+\mathcal{N}^{1:t-1}}$ as follows:

$$\tilde{\mathcal{S}}^t(w,h,c) = \begin{cases} 1 \text{ if } \mathbf{y}^t[c_{bg},w,h] = 0 \text{ and } c = \underset{c' \in \mathcal{C}^t}{\arg\max}\, \mathbf{y}^t[c',w,h]) \\ 1 \text{ if } \mathbf{y}^t[c_{bg},w,h] = 1 \text{ and } c = \underset{c' \in \mathcal{C}^{1:t-1}}{\arg\max}\, \hat{\mathbf{y}}^{t-1}[c',w,h] \\ 0 \text{ otherwise} \end{cases} \tag{4.6}$$

In other words, in the case of non-background pixels we simply copy the ground truth label. Otherwise, we use the class predicted by the old model $g^{t-1}(f^{t-1}(\cdot))$. This pseudo-label strategy allows assigning each pixel labeled as background his real semantic label if this pixel belongs to any of the old classes. However, pseudo-labeling all background pixels can be unproductive, *e.g.* on uncertain pixels where the old model is likely to fail. Therefore, we only keep pseudo-labels where the old model is deemed "*confident*" enough. Equation 4.6 is modified to take into account this uncertainty:

$$\tilde{\mathcal{S}}^t(w,h,c) = \begin{cases} 1 \text{ if } \mathbf{y}^t[c_{bg},w,h] = 0 \text{ and } c = \underset{c' \in \mathcal{C}^t}{\arg\max}\, \mathbf{y}^t[c',w,h] \\ 1 \text{ if } \mathbf{y}^t[c_{bg},w,h] = 1 \text{ and } c = \underset{c' \in \mathcal{C}^{1:t-1}}{\arg\max}\, \hat{\mathbf{y}}^{t-1}[c',w,h] \text{ and } u < \tau_c \\ 0 \text{ otherwise}, \end{cases} \tag{4.7}$$

By notation abuse, u is function $u(\mathbf{y}^t(w,h))$ that measures the uncertainty of the current model given a pixel $\mathbf{x}[:,w,h]$. τ_c denotes a class-specific uncertainty threshold. Hence, in the case where the old model is uncertain ($u \geq \tau_c$) about some pixels, they will be ignored in the final classification loss. Our framework is agnostic to the type of uncertainty used, but in practice we define it as the entropy. Therefore, we use for u the current model's per-pixel entropy $u(\mathbf{y}^t[:,w,h]) = -\sum_{c \in C^{1:t}} \mathbf{y}^t[c,w,h] \log \mathbf{y}^t[c,w,h]$. Likewise, the class-specific threshold τ_c is computed from the median entropy of the old model over all pixels of $\mathcal{D}^t$ predicted the class c for all $c \in \mathcal{C}^{1:t-1}$ as proposed by Saporta et al. (2020). Consequently, the cross-entropy loss with pseudo-labeling of the old classes can be written as:

$$\mathcal{L}_{\text{pseudo}}(\Theta^t) = -\frac{\nu}{WH} \sum_{w,h}^{W,H} \sum_{c \in \mathcal{C}^t} \tilde{\mathbf{y}}(w,h,c) \log \hat{\mathbf{y}}^t[c,w,h]) \ . \tag{4.8}$$

We reduce the normalization factor WH proportionally to the number of discarded pixels. To avoid giving disproportional importance to the pixels belonging to new classes (which are not discarded), we introduce in Equation 4.8 an adaptive factor ν, which is the ratio of accepted old classes pixels over the total number

of such pixels. *i.e.* if most of the image is uncertain, the overall importance of the image relative to other images in the batch is reduced. The overall behavior of our pseudo-labeling is illustrated in Figure 4.1.

We call our final model PLOP (standing for Pseudo-labeling and LOcal Pod). PLOP's final loss is a weighted combination of Equation 4.5 and Equation 4.8:

$$\mathcal{L}(\Theta^t) = \underbrace{\mathcal{L}_{\text{pseudo}}(\Theta^t)}_{\text{classification}} + \lambda \underbrace{\mathcal{L}_{\text{localPod}}(\Theta^t)}_{\text{distillation}}, \tag{4.9}$$

with λ a hyperparameter. PLOP, while already very competitive, can face difficulties when dealing with long continual settings, *i.e.* for which the number of steps grows larger. For this reason, we propose PLOPLong, an extension of PLOP for dealing with such cases.

4.3.4 PLOPLong: a specialization for long settings

While the proposed method already achieves satisfying results in traditional CSS scenarios, it might struggle to deal with longer settings due to two major drawbacks, namely specialization of the classifier towards the recent classes, and the shifts of the batch normalization layers.

First, we observe that in CSS the classifier weights tend to be specialized for the last classes to the detriment of older classes (Hou et al. 2019). Several solutions exist that aim at correcting this bias (Wu et al. 2019; Belouadah and Popescu 2019; B. Zhao et al. 2020; Luo et al. 2018), and we choose the cosine normalization. In practice, we replace the classifier with a cosine classifier (Luo et al. 2018), where the final inner product is discarded in favor of cosine similarity. By doing so, all class weights –both old and new– have a constant magnitude of 1, which drastically reduces the bias towards new classes. The classifier g^t is in segmentation a pointwise (1×1 kernel) convolution which does not alter the spatial organization but maps the ch features channels to $\mathcal{C}^{1:t}$ channels, one per class to predict. This pointwise convolution can be seen as a fully-connected layer that is applied independently to each pixel. Therefore, the classifier g^t has $\{\theta_c^t \in \mathbb{R}^{ch} | \forall c \in \mathcal{C}^{1:t}\}$. The cosine normalization can then be expressed as:

$$\hat{\mathcal{S}}(w,h,c) = \frac{\alpha \langle \theta_c^t, \mathbf{h}(w,h) \rangle}{\|\theta_c^t\|^2 \|\mathbf{h}[:,w,h]\|^2} \tag{4.10}$$

with α a learned scalar parameter initialized to 1 and helping the convergence, and $\mathbf{h} \in \mathbb{R}^{W \times H \times ch}$ the final features embedding before the classifier. First used in continual learning by Hou et al. (2019), this classifier has more recently be adopted by continual few-shot segmentation (Cermelli et al. 2021) or multi-modes

continual learning (in PODNet, see Section 3.2.2.2). The new cosine classifier weights that correspond to the new classifier can then be initialized with weight imprinting (Qi et al. 2018) as recently adapted for segmentation by Cermelli et al. (2021).

Furthermore, we propose to exploit the cosine normalization for intermediary features: the comparison between the Local POD embeddings of the previous model f^{t-1} with the current f^t can be too constrained. We relax the constraints by imposing not a low Euclidean distance between both embeddings, but rather a high cosine similarity, which allows us better plasticity to learn new classes. We need to alter Equation 4.5 by replacing the $\Phi(\cdot)$ operator by $\bar{\Phi}(\cdot) = {}^{\Phi(\cdot)}/_{|\Phi(\cdot)\|^2}$. However, note that the features from level l given to level $l+1$ are not normalized. The normalization only happens for the Local POD embeddings.

Second, an almost ubiquitous component of modern deep computer vision models is the Batch Normalization (Ioffe and Szegedy 2015). It normalizes internal representation with the batch mean $\mu_{\mathbb{B}}$ and batch standard deviation $\sigma_{\mathbb{B}}$ (resp. running mean μ and running std σ) during training (resp. testing). Formally for a batch normalization layer taking an input $\mathbf{x}$ and producing an output $\mathbf{y}$:

$$\mathbf{y} = \frac{\mathbf{x} - \mu_{\mathbb{B}}}{\sigma_{\mathbb{B}}} \cdot \gamma + \beta \tag{4.11}$$

with γ and β two learned parameters. The important drawback of this normalization layer is its assumption that the data is sampled *i.i.d.* which is not the case in continual segmentation where multiple shifts (Moreron-Torresa et al. 2012; Lesort et al. 2021), happen in the training tasks, different from the testing tasks. Drawing inspiration from the domain incremental (Lomonaco et al. 2020) and continual few-shots segmentation (Cermelli et al. 2021) literature, we choose to replace the batch normalization by batch *re*normalization (Ioffe 2017), i.e.:

$$\mathbf{y} = \frac{\mathbf{x} - \mu_{\mathbb{B}}}{\sigma_{\mathbb{B}}} \cdot r + d, \text{ where } r = \frac{\sigma_{\mathbb{B}}}{\sigma}, \; d = \frac{\mu_{\mathbb{B}} - \mu}{\sigma}. \tag{4.12}$$

Intuitively, batch renormalization avoids the discrepancy between training and testing of the batch normalization. Furthermore, following Cermelli et al. (2021), we freeze during training the statistics (μ and σ) after the first task to avoid harmful statistics drifts.

All these improvements further increase performance for long series of tasks: (1) Cosine classifier reduces the increased bias between recent and old classes. (2) A cosine-based Local POD relax the constraints in order to correctly learn the bigger number of new classes. (3) Frozen BatchReNorm reduces the inherent drift of statistics that grow larger as the number of tasks increases. In what follows, we denote PLOPLong the model obtained from PLOP by adding all three

improvements. We empirically saw that these contributions marginally improve PLOP taken individually, but when used altogether provided an major boost of performance in long-range scenarios.

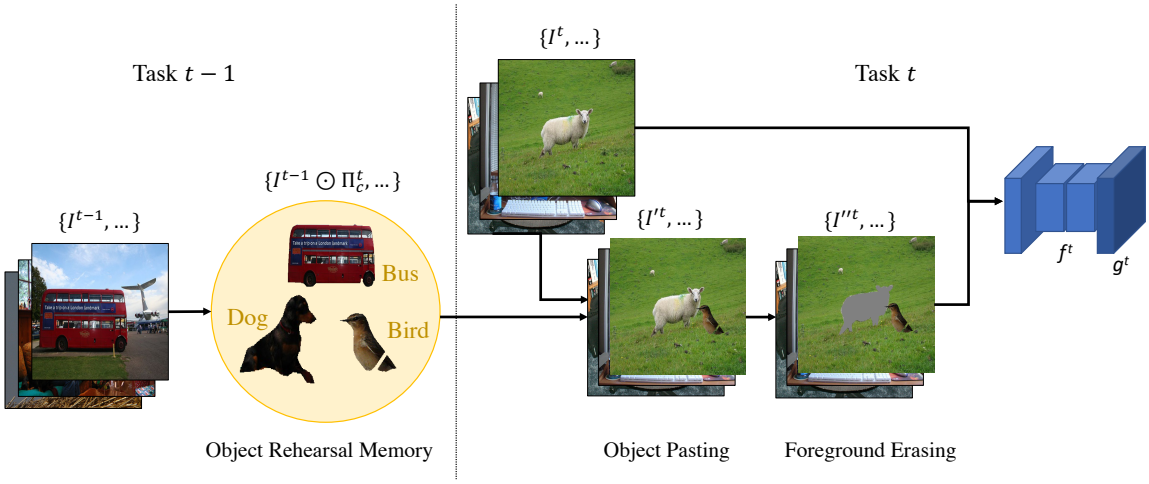


Figure 4.2. – **Our Object Rehearsal strategy**. In task $t-1$, we select from $\{\mathbf{x}^{t-1}, \ldots\}$ a limited amount of objects (here `bus`, `bird`, and `dog`), which will be then mixed in the images $\{\mathbf{x}^t, \ldots\}$ from the current task t; after pasting, the other present objects are erased. Finally, the current model $g^t \circ f^t$ will be given the concatenation of the original images and the augmented images $\{\mathbf{x}''^t, \ldots\}$. Our Object Rehearsal allows to generate a wider diversity of images, resulting in higher accuracies while being up to 146x more memory efficient.

4.4 Object Rehearsal

Neither the proposed PLOP nor PLOPLong did make use of previously seen data $\{\mathcal{D}^1, \ldots, \mathcal{D}^{t-1}\}$ when considering step t. In this section, we explore how to further improve CSS performance if a model is now allowed to rehearse a limited amount of previous data. We first consider a traditional approach, namely Image Rehearsal. We show that such a naive approach cannot work well in the frame of CSS and propose innovative adaptations to make it work. Then, we highlight the drawbacks of this rehearsal method and propose a novel approach named Object Rehearsal, more effective both in terms of mean Intersection-over-Union (mIoU) and memory consumption.

4.4.1 Image Rehearsal

We first consider rehearsing a limited amount of images from previous tasks during the current task. Image rehearsal is a well-studied problem in continual image classification. However, it was rarely applied to other tasks. Contrary to image classification, in CSS, images can have multiple labels if several objects (*e.g.* `car`, `sky`) are present in the image. Moreover, as we described previously in Section 4.3.1, the segmentation maps are partially labelized. For a given task t, when rehearsing an image stored from task $t-i$, we only have labels of classes C^{t-1}. Because of these two problems, we cannot naively apply image rehearsal on CSS. To tackle the former problem, we propose to select M images for each class $c \in C^t$, resulting in $M \times \mathcal{N}^t$ images for step t. We ensure that the selected images are unique: as multiple classes can co-exist in the same image, the resulting amount of sampled classes may be above $M \times \mathcal{N}^t$. We select the images with a simple random selection, which has been proved to generally perform as good as more elaborate methods (Castro et al. 2018). The memory footprint will grow until all tasks but the last are seen, resulting in a total amount of $M \times \mathcal{N}^{1:T-1}$ stored images. To address the second problem (partial labelization, see also Section 4.3.3), we fill the background using our proposed hard pseudo-labeling (see Section 4.3.3) which *completes* the segmentation maps with old classes $\mathcal{C}^{1:t-i}$ as well as new classes $\mathcal{C}^{t-i+1:t}$.

4.4.2 Object Rehearsal

The main drawback of image rehearsal is that CSS images are usually large (from 512×512 to 1024×2048) yet they are sparsely informative, as a significant part of the images consists in background pixels (Lin et al. 2017) (*e.g.* 63% of Pascal-VOC (Everingham et al. 2015) pixels) or belongs to a majority class (*e.g.* 32% of Cityscapes (Cordts et al. 2016) pixels are `roads`).

To address both problems, instead of storing whole images from the previous tasks $\{1, ..., t-1\}$, we propose to store an informative portion that we will mix with the images of the current task t. Image mixing is popular for classification (Hongyi Zhang et al. 2018; Yun et al. 2019; Dabouei et al. 2020; Verma et al. 2019; B. Li et al. 2021; Rame et al. 2021b) yet, to the best of our knowledge, sees limited use for semantic segmentation (Fang et al. 2019; Olsson et al. 2021; J. Zhang et al. 2021; Tranheden et al. 2021; Ghiasi et al. 2020), and has never been considered to design memory-efficient rehearsal learning systems. Formally, given an image $\mathbf{x}$

and the corresponding ground truth segmentation maps $\mathbf{y}^t$, we define a binary mask Π_c such that $\forall c \in \mathcal{C}^t$:

$$O_c = I \odot \Pi_c \text{ where } \Pi[:, w, h] = \begin{cases} 1, \text{if} \mathbf{y}^t[:, w, h] = c \\ 0, \text{otherwise} \end{cases}, \tag{4.13}$$

with O_c the selected object for class c. By nature, this patch is extremely sparse and can be efficiently stored on disk by modern compression algorithms (**ISO)10918**). The total memory footprint at task t is thus $M \times \mathcal{N}^{1:t}$ objects.

Then, when learning a task $t > 1$, the model will learn on both the task dataset $\cup_i^N (\mathbf{x}_i^t, \mathbf{y}_i^t)$ and the object memory $\cup_{c \in \mathcal{C}^{1:t-1}} \cup_i^M O_{c,i}$, with N being the total number of images in the task dataset. The latter will augment the former through object pasting. We augment each object by applying an affine transformation matrix (Fang et al. 2019):

$$\mathbf{T} = \begin{bmatrix} z\cos\alpha & z\sin\alpha & 0 \\ -z\sin\alpha & z\cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.14}$$

with z a zoom factor, and α an in-plane rotation angle. Note that we do not translate the object as its original position is often a good prior: indeed, objects are usually located at the same location (*e.g.* pedestrian on the left and right sidewalk, car on the middle road, etc.) We abuse the notation by denoting $T(.)$ the application of this transformation matrix. Once the data augmentation is done on the object (and its mask), we paste it on an image $\mathbf{x}^t$ that refers to the current task:

$$\mathbf{x}'^t = \mathbf{x}^t \odot (\mathbf{1} - T(\Pi_c)) + T(O_c) \odot T(\Pi_c)\,. \tag{4.15}$$

The pasting can result in local incoherence where the pixel of a `cow` is pasted on top or next to the pixel of a `television`. Naively the object borders can be smoothed into the image with a Gaussian filter. Unfortunately, it results in imprecise contours, which are important in segmentation (Y. Chen et al. 2020).

This approach has already been envisioned as a form of data augmentation in semantic segmentation, although the gains were small and to avoid the noise induced by this pasting, the batch size is prohibitively large (up to 512) (Ghiasi et al. 2020). We propose to reduce interference between the pasted object and the destination images by **selective erasing** of the surrounding pixels. Indeed, given a binary matrix $\Xi(\Pi, S)$ of the same dimension as I and Π:

$$\mathbf{x}''^t = \mathbf{x}'^t \odot (\mathbf{1} - \Xi) + \kappa \odot \Xi\,. \tag{4.16}$$

We replace the pixels erased according to the mask Ξ with a RGB color $\kappa \in \mathbb{R}^3$. This RGB vector could be chosen through in-painting (Fang et al. 2019) or be

Dataset	# classes	Background?	# train	# test	Image size
Pascal-VOC (Everingham et al. 2015)	20	✓	10k	1.5k	512×512
Cityscapes (Cordts et al. 2016)	19	✗	3k	0.5k	512×1024
ADE20k (B. Zhou et al. 2017)	150	✗	20k	2.0k	512×512

Table 4.1. – **Description of the three datasets** considered in this paper. For datasets without explicit background class, one is created based on unlabeled pixels.

Dataset	Setting	Mode	# tasks	# base classes	# classes / inc. task
Pascal-VOC	19-1	class	2	19	1
	15-5	class	2	15	5
	15-1	class	6	15	1
	10-1	class	11	10	1
Cityscapes	14-1	class	6	14	1
	11-5	domain	3	11	5
	11-1	domain	11	11	1
	1-1	domain	21	1	1
ADE20k	100-50	class	2	100	50
	50-50	class	2	50	50
	100-10	class	6	100	10
	100-5	class	11	100	5

Table 4.2. – **Description of the 12 different benchmarks** evaluated in this paper. For some of these, each task brings new classes while, for others, it comes with new domains.

random noise, but in practice, we choose a constant color gray. We also update the segmentation maps accordingly:

$$\mathbf{y}''^{t} = \boldsymbol{y}'^{t} \odot (\mathbf{1} - \Xi) + 255 \odot \Xi\,, \tag{4.17}$$

where 255 is a dummy class id used in segmentation to ignore some pixel labels which will not be counted in the classification loss Equation 4.8. We illustrate our rehearsal strategy in Figure 4.2.

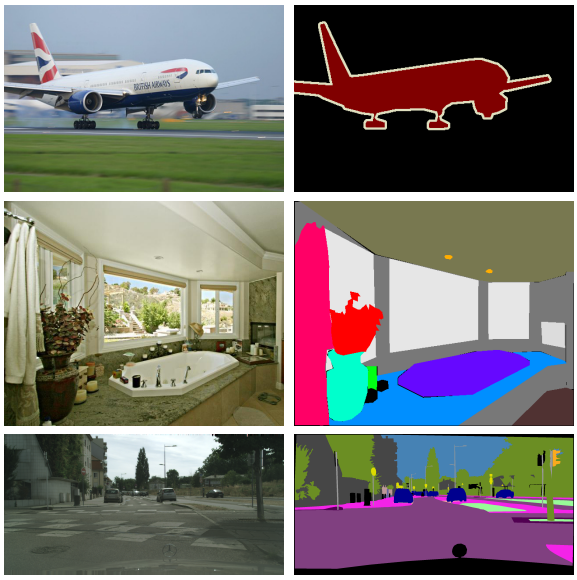


Figure 4.3. – **Dataset visualization:** of an example image and its segmentation maps for Pascal-VOC, ADE20k, and Cityscapes.

4.5 Experiments

4.5.1 Datasets, Protocols, and Baselines

To ensure fair comparisons with state-of-the-art approaches, we follow the experimental setup of Cermelli et al. (2020) for datasets, protocol, metrics, and baseline implementations. Although, we also propose to evaluate on new datasets and on more challenging protocols. Furthermore, we explore in advanced experiments, for the first time, how rehearsal can improve performance in CSS. We provide the full implementation details in the appendix (Section A.4).

Datasets: We evaluate our model on three datasets, summarized in Table 4.1: Pascal-VOC (Everingham et al. 2015), Cityscapes (Cordts et al. 2016) and ADE20k (B. Zhou et al. 2017). VOC contains 20 classes, 10,582 training images, and 1,449 testing images. Cityscapes contains 2975 and 500 images for train and test, respectively. Those images represent 19 classes and were taken from 21 different cities. ADE20k has 150 classes, 20,210 training images, and 2,000 testing images. All ablations and hyperparameters tuning were done on a validation subset of the training set made of 20% of the images. For all datasets, we use random resize and crop augmentation (scale from 80% to 110%), as well as random horizontal flip during training time. The final image size for Pascal-VOC and ADE20k is 512×512 while it is 512×1024 for Cityscapes.

CSS protocols: Cermelli et al. (2020) introduced the *Overlapped* setting where only the current classes are labeled vs a background class $\mathcal{C}^t$. Moreover, pixels can belong to any classes $\mathcal{C}^{1:t-1} \cup \mathcal{C}^t \cup \mathcal{C}^{t+1:T}$ (old, current, and future). Those two constraints make this setting both challenging and realistic, as in a real setting there is not any oracle method to exclude future classes from the background. In all our experiments, we respect the Overlapped setting. While the training images are only labeled for the current classes, the testing images are labeled for all seen classes. We evaluate several CSS protocols for each dataset, *e.g.* on VOC 19-1, 15-5, and 15-1 respectively consists in learning 19 then 1 class ($T = 2$ steps), 15 then 5 classes (2 steps), and 15 classes followed by five times 1 class (6 steps). The last setting is the most challenging due to its higher number of steps. Similarly, on Cityscapes 14-1 means 14 followed by five times 1 class (6 steps) and on ADE 100-50 means 100 followed by 50 classes (2 steps). We provide a summary of all these settings in Table 4.2.

Metrics: In Figure 2.3, we defined the **final** and **average** incremental accuracies. In the context of semantic segmentation, we generalize those metrics to the mean Intersection-over-Union (mIoU). Specifically, we compute mIoU after the last step T for the initial classes $\mathcal{C}^1$, for the incremented classes $\mathcal{C}^{2:T}$, and for all classes $\mathcal{C}^{1:T}$ (*final*). These metrics respectively reflect the robustness to catastrophic forgetting (the model rigidity), the capacity to learn new classes (plasticity), as well as its overall performance (trade-of between both). We also introduce the *avg* metric (short for *average*), novel in CSS, which measures the average of mIoU scores measured step after step, integrating performance over the whole continual learning process. We stress that all four metrics are important, and we should not disregard one for the other: a model suffering no forgetting but not able to learn anything new is useless.

Baselines: We benchmark our model against the latest state-of-the-arts CSS methods ILT (Michieli and Zanuttigh 2019), MiB (Cermelli et al. 2020), GIFS (Cermelli et al. 2021), and SDR (Michieli and Zanuttigh 2021). Note that while GIFS was created by Cermelli et al. (2021) for continual *few-shots* segmentation, we adapt it for the more general task of CSS. Unless stated otherwise, all the results are excerpted from the corresponding papers. We also evaluate general continual models based on weight constraints (PI (Zenke et al. 2017), EWC (Kirkpatrick et al. 2017), and RW (Chaudhry et al. 2018)) and knowledge distillation (LwF (Z. Li and Hoiem 2016) and LwF-MC (Rebuffi et al. 2017c)). Moreover, unless explicitly stated otherwise, all the models (ours included), do not use rehearsal learning.

Method	**19-1** (2 tasks)				**15-5** (2 tasks)				**15-1** (6 tasks)			
	0-19	20	*final*	*avg*	0-15	16-20	*final*	*avg*	0-15	16-20	*final*	*avg*
Fine Tuning†	6.80	12.90	7.10		2.10	33.10	9.80		0.20	1.80	0.60	
PI† (Zenke et al. 2017)	7.50	14.00	7.80		1.60	33.30	9.50		0.00	1.80	0.50	
EWC† (Kirkpatrick et al. 2017)	26.90	14.00	26.30		24.30	35.50	27.10		0.30	4.30	1.30	
RW† (Chaudhry et al. 2018)	23.30	14.20	22.90		16.60	34.90	21.20		0.00	5.20	1.30	
LwF† (Z. Li and Hoiem 2016)	51.20	8.50	49.10		58.90	36.60	53.30		1.00	3.90	1.80	
LwF-MC† (Rebuffi et al. 2017c)	64.40	13.30	61.90		58.10	35.00	52.30		6.40	8.40	6.90	
ILT† (Michieli and Zanuttigh 2019)	67.10	12.30	64.40		66.30	40.60	59.90		4.90	7.80	5.70	
ILT (Michieli and Zanuttigh 2019)	67.75	10.88	65.05	71.23	67.08	39.23	60.45	70.37	8.75	7.99	8.56	40.16
MiB† (Cermelli et al. 2020)	70.20	22.10	67.80		75.50	49.40	69.00		35.10	13.50	29.70	
MiB (Cermelli et al. 2020)	71.43	23.59	69.15	73.28	**76.37**	49.97	**70.08**	**75.12**	34.22	13.50	29.29	54.19
SDR◇ (Michieli and Zanuttigh 2021)	71.30	23.40	69.0		76.30	50.20	70.10		47.30	14.70	39.50	
GIFS (Cermelli et al. 2021)	57.88	32.82	56.69	67.05	23.61	16.43	21.90	50.97	59.36	13.89	48.53	61.43
PLOP	**75.35**	37.35	**73.54**	**75.47**	75.73	**51.71**	**70.09**	**75.19**	65.12	21.11	54.64	67.21
PLOPLong	74.75	**39.68**	73.08	74.32	75.95	48.31	69.37	73.58	**72.00**	**26.66**	**61.20**	**70.02**

Table 4.3. – **Pascal-VOC 2021 quantitative experiments** in mIoU (%). †: results excerpted from Cermelli et al. (2020), ◇ from Michieli and Zanuttigh (2021). Other results come from re-implementation.

Method	**14-1** (6 tasks)			
	1-14	15-19	*final*	*avg*
MiB (Cermelli et al. 2020)	55.11	12.91	44.56	49.76
PLOP	56.59	13.07	45.71	51.28
PLOPLong	**58.60**	**15.04**	**47.71**	**54.31**

Table 4.4. – **Cityscapes quantitative experiments** on Cityscapes 14-1 in mIoU (%).

4.5.2 PLOP and PLOPLong experiments

4.5.2.1 Quantitative Evaluations

Pascal VOC 2012: Table 4.3 shows quantitative experiments on VOC 19-1, 15-5, and 15-1. Both the proposed PLOP and PLOPLong outperform state-of-the-art approaches, MiB (Cermelli et al. 2020), SDR (Michieli and Zanuttigh 2021), and GIFS (Cermelli et al. 2021) on all evaluated settings by a significant margin. For instance, on 19-1, the forgetting of old classes (1-19) is reduced by 4.39 percentage points (*p.p.*) while performance on new classes is greatly improved (+13.76 *p.p.*), as compared to the best performing method so far, MIB (Cermelli et al. 2020). On 15-5, our model is on par with our re-implementation of MiB, and surpasses the original paper scores (Cermelli et al. 2020) by 1 *p.p.*. On the most challenging 15-1 setting, general continual models (EWC and LwF-MC) and ILT all have very low mIoU. While MiB shows significant improvements, PLOP still outperforms it by a wide margin. Furthermore, PLOP also significantly outperforms the best performing approach on this setting, GIFS Cermelli et al. (2021) (+6.11 *p.p.*). Moreover,

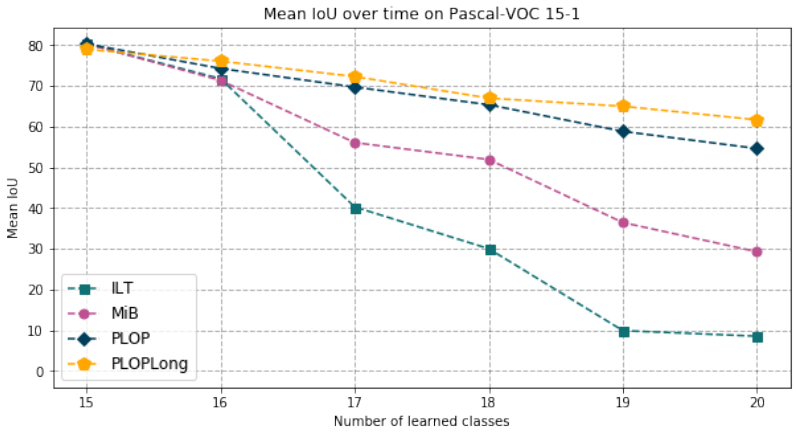


Figure 4.4. – mIoU evolution on Pascal-VOC 2012 15-1. While MiB's mIoU quickly deteriorates, PLOP and PLOPLong's mIoU remains high, due to improved resilience to catastrophic forgetting.

Method	**100-50** (2 tasks)				**50-50** (3 tasks)				**100-10** (6 tasks)			
	0-100	101-150	*final*	*avg*	0-50	51-150	*final*	*avg*	0-100	101-150	*final*	*avg*
ILT (Michieli and Zanuttigh 2019)	18.29	14.40	17.00	29.42	3.53	12.85	9.70	30.12	0.11	3.06	1.09	12.56
MiB (Cermelli et al. 2020)	40.52	**17.17**	**32.79**	**37.31**	45.57	**21.01**	29.31	38.98	38.21	11.12	29.24	35.12
PLOP	**41.87**	14.89	**32.94**	**37.39**	**48.83**	**20.99**	**30.40**	**39.42**	**40.48**	**13.61**	**31.59**	**36.64**

Table 4.5. – **ADE20k quantitative experiments** in mIoU (%).

mIoU for the joint model is 77.40%, thus PLOP narrows the gap between CSS and joint learning on every CSS scenario. The average mIoU is also improved (+5.78 *p.p.*) compared to GIFS, indicating that each CSS step benefits from the improvements related to our method. This is echoed by Figure 4.4, which shows that while mIoU for both ILT and MiB deteriorates after only a handful of steps, PLOP's mIoU remains very high throughout, indicating improved resilience to catastrophic forgetting and background shift. Last but not least, while PLOP performs better than PLOPLong on short setups (*e.g.* 19-1, 15-5), PLOPLong performs better, however, on longer CSS benchmarks such as 15-1 (+2.81 *p.p.*).

Cityscapes: We also validate our method on Cityscapes in Table 4.4. In order to design a setting similar to Pascal-VOC 15-1 (6 tasks), we design the 14-1 setting. Also, we simulate a background class by folding together the unlabeled classes. Here again, PLOP performs slightly better than MIB (+2.48 *p.p.* on the old classes, +0.16 *p.p.* on the new ones, +1.15 *p.p.* on average). Moreover, PLOPLong performs significantly better (+3.59 *p.p.* on the old classes, +2.13 *p.p.* on the new ones, +3.15 *p.p.* on average), indicating better robustness to both catastrophic forgetting and background shift, especially when considering long continual learning setups. To more precisely assess this phenomenon, we investigate the performance of both methods when dealing with longer task learning sequences.

Method	VOC 10-1 (11 tasks)			
	0-10	11-20	*final*	*avg*
ILT (Michieli and Zanuttigh 2019)	7.15	3.67	5.50	25.71
MiB (Cermelli et al. 2020)	12.25	13.09	12.65	42.67
PLOP	44.03	15.51	30.45	52.32
PLOPLong	**61.06**	**18.56**	**40.83**	**58.62**

Table 4.6. – **Pascal-VOC 2012 10-1** in mIoU (%).

Method	ADE 100-5 (11 tasks)			
	0-100	101-150	*final*	*avg*
ILT (Michieli and Zanuttigh 2019)	0.08	1.31	0.49	7.83
MiB (Cermelli et al. 2020)	36.01	5.66	25.96	32.69
PLOP	**39.11**	**7.81**	**28.75**	**35.25**

Table 4.7. – **ADE20k 100-5** in mIoU (%).

ADE20K: Table 4.5 shows experiments on ADE 100-50, 100-10, and 50-50. This dataset is notoriously hard, as the joint model baseline mIoU is only 38.90%. ILT has poor performance in all three scenarios. PLOP shows comparable performance with MiB on the short setting 100-50 (only 2 tasks), improves by 1.09 *p.p.* on the medium setting 50-50 (3 tasks), and significantly outperforms MiB with a wider margin of 2.35 *p.p.* on the long setting 100-10 (6 tasks). In addition to being better on all settings, PLOP showcased an increased performance gain on longer CSS (*e.g.* 100-10) scenarios, due to increased robustness to catastrophic forgetting and background shift.

Longer Continual Trainings: We argue that CSS experiments should push towards more steps (Wortsman et al. 2020; Lomonaco et al. 2020; Castro et al. 2018) to quantify the robustness of approaches w.r.t. catastrophic forgetting and background shift. We introduce two novel and much more challenging settings with 11 tasks, almost twice as many as the previous longest setting. We report results for VOC 10-1 in Table 4.6 (10 classes followed by 10 times 1 class) and ADE 100-5 in Table 4.7 (100 classes followed by 10 times 5 classes). The second previous State-of-the-Art method, ILT, has a very low mIoU (< 6 on VOC 10-1 and practically null on ADE 100-5). Furthermore, the gap between PLOP and MiB is even wider compared with previous benchmarks (*e.g.* $\times 3.6$ mIoU on VOC for mIoU of base classes 1-10), which confirms the superiority of PLOP when dealing with the long continual processes. Moreover, in such a case, PLOPlong really shines, bringing significant improvements (+17.03 *p.p.* on the old classes, +3.05 *p.p.* on the

new classes, +10.42 *p.p.* on average) due to a combination of cosine normalization (both on the classifier and Local POD) and its frozen BatchReNormalization.

4.5.2.2 Models Introspection

We compare several distillation and classification losses on VOC 15-1 to stress the importance of the components of PLOP and report results in Table 4.8. All comparisons are evaluated on a validation set made with 20% of the training set, therefore results are slightly different from the main experiments.

Distillation comparisons: Table 4.8a compares different distillation losses when combined with our pseudo-labeling loss. ILT (Michieli and Zanuttigh 2019) used the naive knowledge distillation of Geoffrey Hinton et al. (2015). CSC (Park and Heo 2020) constrained spatial and channel correlation of intermediary features. This work bears similarity with POD and Local POD, but it's much more computationally intensive as it computes pixel-by-pixel correlation and not as performant. Finally, UNKD introduced in Cermelli et al. (2020) performs better than the Knowledge Distillation (KD) of Geoffrey Hinton et al. (2015), but not at every step (as indicated by the *avg.* value), which indicates instability during the training process. POD (Section 3.2), improves the results on the old classes, but not on the new classes (16-20). In fact, due to too much plasticity, POD model likely overfits and predicts nothing but the new classes, hence a lower mIoU. Finally, Local POD leads to superior performance (+20 *p.p.*) w.r.t. all metrics, due to its integration of both long and short-range dependencies. This final row represents our full PLOP strategy.

Classification comparisons: Table 4.8b compares different classification losses when combined with our Local POD distillation loss. Cross-Entropy (CE) variants perform poorly, especially on new classes. UNCE, introduced in (Cermelli et al. 2020), improves by merging the background with old classes, however, it still struggles to correctly model the new classes, whereas our pseudo-labeling propagates more finely information of the old classes, while learning to predict the new ones, dramatically enhancing the performance in both cases. This penultimate row represents our full PLOP strategy. Also, notice that the performance for pseudo-labeling is very close to *(Oracle) Pseudo-good* (where the incorrect pseudo-labels are removed), which may constitute a performance ceiling of our uncertainty measure. A comparison between these two results illustrates the relevance of our entropy-based uncertainty estimate. *(Oracle) Pseudo-corrected* is similar to the previous oracle but instead fix the incorrect pseudo-labels; note that not all labels are present as some pixels were not even pseudo-labelized due to their high un-

Distillation loss	0-15	16-20	*final*	*avg*
ILT (Michieli and Zanuttigh 2019)	19.91	5.49	16.48	49.43
CSC (Park and Heo 2020)	25.49	4.72	20.48	44.97
Knowledge Distillation (Geoffrey Hinton et al. 2015)	29.72	4.42	23.69	49.18
UNKD (Cermelli et al. 2020)	34.85	5.26	27.80	46.39
POD	43.94	4.82	34.62	53.35
Local POD (Equation 4.5)	**63.06**	**17.92**	**52.31**	**65.71**

(a) **Distillation loss ablations:** Pseudo loss (Equation 4.8) with different distillation losses.

Classification loss	0-15	16-20	*final*	*avg*
CE only on new	12.95	2.54	10.47	47.02
CE	33.80	4.67	26.87	50.79
UNCE (Cermelli et al. 2020)	48.46	4.82	38.62	53.19
Pseudo (Equation 4.8)	**63.06**	**17.92**	**52.31**	**65.71**
(Oracle) Pseudo-good	*63.69*	*23.35*	*54.09*	*66.05*
(Oracle) Pseudo-corrected	*66.88*	*16.88*	*54.98*	*66.50*
(Oracle) CE + all labels	*71.45*	*10.78*	*57.00*	*67.04*

(b) **Classification loss ablations:** Local POD loss (Equation 4.5) with different classification losses.

Table 4.8. – **Comparison studies** on Pascal-VOC 2012 15-1 on a validation subset of 20% of the training set.

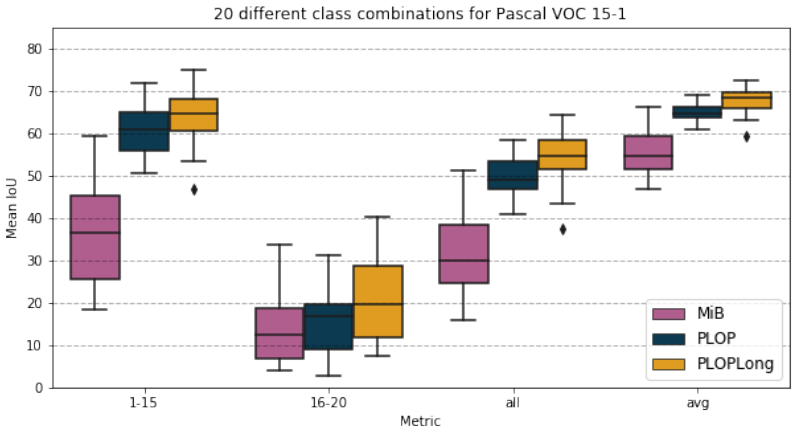


Figure 4.5. – **Robustness to class ordering:** Boxplots of the mIoU of initial classes (1-15), new (16-20), all, and average for 20 random class orderings. PLOP is significantly better and more stable than MiB. PLOPLong further improves upon PLOP by better retaining old class information.

certainty. Finally, *(Oracle) CE + all labels* has access to all labels of previous and current classes $\mathcal{C}^{1:t}$.

4.5.2.3 Robustness to class ordering

Continual learning methods may be prone to instability. It has already been shown in related contexts (D. Kim et al. 2019) that class ordering can have a large impact on performance. Unfortunately, in real-world settings, the optimal class order can never be known beforehand: thus, the performance of an ideal CSS method should be as invariant to class order as possible. In all experiments done so far, this class order has been kept constant, as defined by Cermelli et al. (2020). We report results in Figure 4.5 as boxplots obtained by applying 20 random permutations of the class order on VOC 15-1. We report in Figure 4.5 (from left to right) the mIoU for the old, new classes, all classes, and average over CSS steps. In all cases, PLOP surpasses MiB in terms of avg mIoU. Furthermore, the standard deviation (*e.g.* 10% vs 5% on *all*) is always significantly lower, showing the excellent stability of PLOP compared with existing approaches. Moreover, PLOPLong, while having more variance on the new classes, performs better overall as compared to PLOP, not to mention MiB. This is due to the fact that PLOPLong, due to improved distillation loss (at prediction and Local POD levels) and batch re-normalization that allows to better retain information of the old classes, which again is more conspicuous on longer CSS scenarios.

Method	Rehearsal	**15-1** (6 tasks) **Memory** (Mb) ↓	**Time** (Hours) ↓	0-15	16-20	*final*	*avg*
PLOP	—	0	1.8	65.12	21.11	54.64	67.21
PLOPLong	—	0	1.8	72.00	26.66	61.20	70.02
Yu et al. (2020)	Unlabeled COCO	20,000	7.0	71.40	40.00	63.60	
PLOP	Unlabeled COCO	20,000	1.4	72.57	45.08	66.03	71.85
PLOP	Unlabeled VOC	2,000	1.4	75.32	52.59	69.91	75.21
PLOPLong	Partial VOC	2.2	2.6	74.14	38.87	65.74	72.02
PLOPLong	Partial VOC	22	2.6	**74.18**	43.22	66.81	**72.48**
PLOPLong	Object VOC	0.26	2.7	73.32	42.86	66.07	72.21
PLOPLong	Object VOC	2.6	2.7	73.79	**45.78**	**67.12**	**72.42**
Joint model	—	—	—	79.10	72.60	77.40	—

Table 4.9. – **Pascal-VOC quantitative experiments with rehearsal** on Pascal-VOC 2012 15-1 overlap in mIoU (%). We only consider the time overhead spent after the first task whose computation overhead is similar for all methods.

4.5.3 Object Rehearsal experiments

4.5.3.1 Quantitative Evaluations

Pascal-VOC 15-1: We now allow CSS models to store information from the previous steps and classes: in such a case, the overall model efficiency can be understood as to what extent it allows to find a good trade-off between its accuracy (as measured by the aforementioned standard metrics) and the memory footprint of the stored images or objects. Methods such as HRHF (Zilong Huang et al. 2021) can not easily be understood in these terms as they do not store data, strictly speaking, but rather use deep inversion (Yin et al. 2019) techniques to generate synthetic data, which comes at a high time requirement: thus, we exclude this method from our comparisons. We first consider the challenging Pascal-VOC 15-1 setting in Table 4.9. We use PLOP and PLOPLong as baselines with 0 memory overhead, as both models only use data from the current task. Moreover, we compare with Yu et al. (2020), where an unlabeled external dataset such as COCO (Lin et al. 2014) is used through pseudo-labeling to improve performance on Pascal-VOC, as both datasets present significant overlap in terms of classes and domains. We reimplemented their method and also considered PLOP in this configuration. Using the external COCO provide PLOP an important gain of mIoU for both old classes (+7 *p.p.*) and new classes (+24 *p.p.*). Furthermore, PLOP with COCO is significantly more performant than Yu et al. (2020)'s model (+5.43 *p.p.*) despite the latter was designed explicitly to use such unlabeled external dataset. Notice also how PLOPLong, without any kind of rehearsal, remains equivalent to Yu et al. (2020) in terms of mIoU on 0-15 (72.00% vs 71.40%), despite the latter using a large

pool of data (+20Go). Perhaps counter-intuitively the gain is located on the new classes (26.66% vs 40.00%) as the rehearsal effect has a regularizing effect leading to less over-predicting of the recent classes. A drawback of using COCO is that the visual domain is not exactly the same as VOC, therefore we also considered using PLOP with a rehearsal of the unlabeled VOC. Without surprise, it results in a much better overall performance (+3.88 *p.p.* compared to using COCO). Another setup that we consider is the image rehearsal paradigm, where, at each step, we keep a number of (randomly selected) images along with their segmentation map. These segmentation maps are, however, incomplete, due to the nature of the CSS problem (see Section 4.4.2): hence, we refer to this setting as partial VOC. We consider two amounts of images to keep: 10 images per class (22 Mb) and 1 image per class (2.2 Mb). PLOPLong largely benefits from this rehearsal, most notably on new classes (16-20); with a gain up to 16.56 *p.p.*. Finally, we compare our novel Object Rehearsal denoted by "*object VOC*" where we store either one object per class (0.26 Mb) or 10 objects per class (2.6 Mb). As shown on Table 4.9, The proposed object rehearsal, in addition to being significantly more memory efficient than whole image rehearsal (8.5× less space used), is equivalent or better in terms of mIoU; especially for new classes (2.56 *p.p.*). The ensemble of our experiments proves that rehearsal, when the partial or missing labeling is carefully handled, can provide important performance gain. Furthermore, our novel object rehearsal manages to strike the best trade-off with a minimal memory overhead without impacting its performance.

Cityscapes: We propose more experiments with Object Rehearsal in Table 4.10 where we apply our model on Cityscapes 14-1. For the rehearsal, we sample either 10 images or objects per class. Cityscapes images are particularly large (even resized to 512×1024) and "empty" (most of it is road and sky). Consequently, the memory overhead is extremely important compared object rehearsal (117 vs 0.8). We show that both our novel image and object rehearsal improve performance of the already competitive PLOPLong, and with object rehearsal providing the large gain (+9 *p.p.* in *all*).

t

4.5.3.2 Object Rehearsal Introspection

Table 4.11 draws a comparison between several rehearsal alternatives in CSS. We split methods according to three criterions: "*type*" denotes whether we store a whole image, an object, or a patch. For both object and patch, we only rehearse a small amount of the images: in object, only the pixel's objects are used while for a patch, we use the pixel's object and the close surrounding pixels that contain background information. We insert the rehearsal data according to "*mixing*", either by

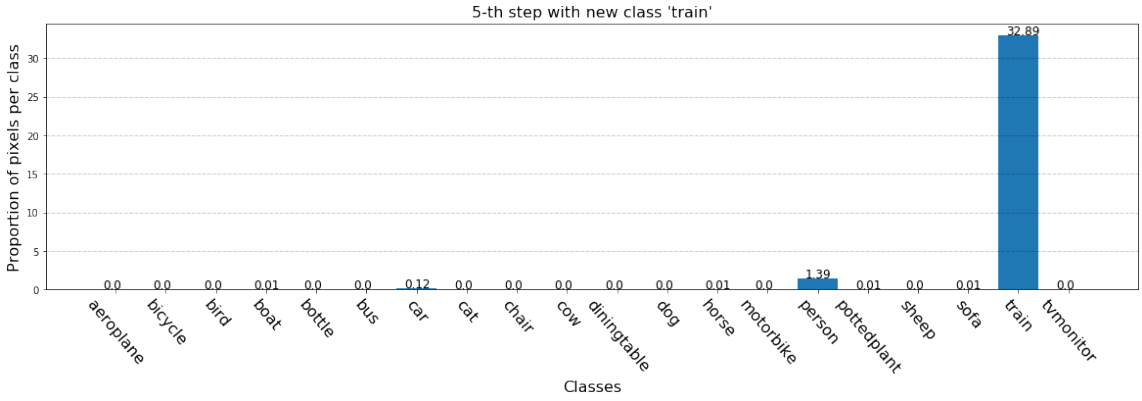


Figure 4.6. – **Pixel distribution per class** during the 5^{th} step of Pascal-VOC 15-1.

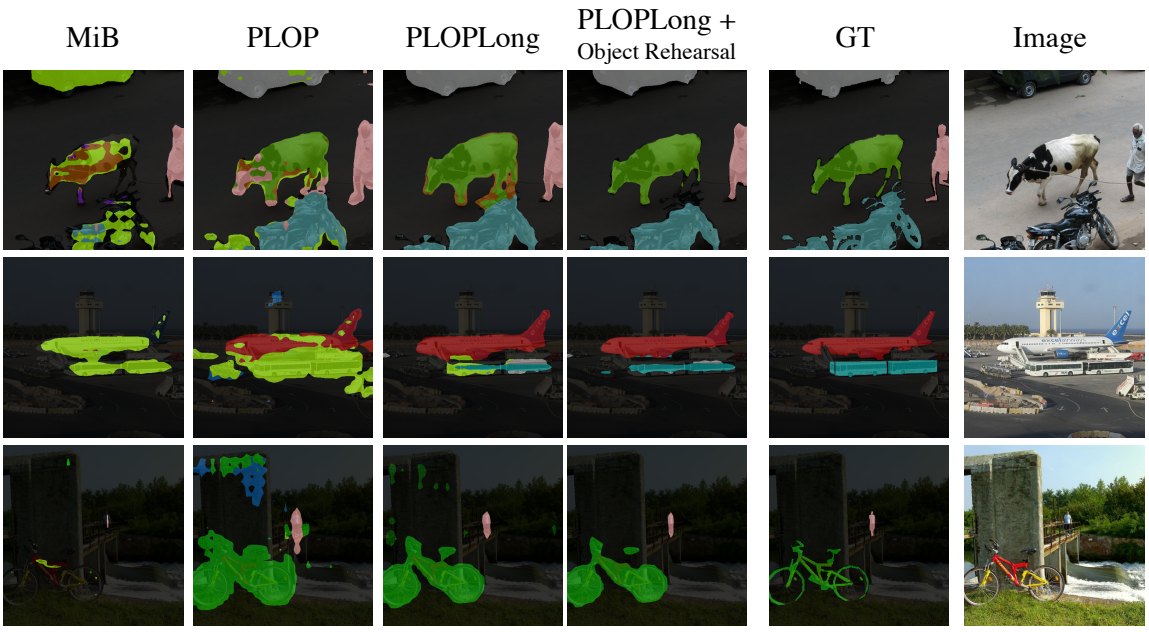


Figure 4.7. – **Visualization of the predictions** of MiB, PLOP, PLOPLong, and PLOPLong with Object Rehearsal on three test images at the 6^{th} and final step on VOC 15-1 scenario. The first image contains `car`, `cow`, and `person`; the second `plane` and `bus`, and the third `bicycle` and `person`. While MiB does not manage to predict the correct classes, and tends to overpredict the most recent ones (*e.g.* `train` in light green). PLOP mostly grasp the correct classes, though sometimes with imprecision. PLOPLong and *a fortiori* PLOPLong + Object rehearsal captures all existing classes, with the latter predicting almost perfect segmentation masks, compared to the ground-truth.

Method	Rehearsal	Memory	**14-1** (6 tasks) 1-14	15-19	*final*	*avg*
MiB (Cermelli et al. 2020)	—	0	55.11	12.91	44.56	49.76
PLOP	—	0	56.59	13.07	45.71	51.28
PLOPLong	—	0	58.60	15.04	47.71	54.31
PLOPLong	Partial	117.0	**58.93**	19.55	49.09	**55.74**
PLOPLong	Object	0.8	57.82	**23.13**	**49.15**	54.80

Table 4.10. – **Cityscapes quantitative experiments with rehearsal** on Cityscapes 14-1 overlap in mIoU (%).

Type	Mixing	Erase	Memory ↓	**15-1** (6 tasks) *final*	*avg*	
Image	Mixup	—	22.20	61.77	69.88	I
	—	—		66.81	**72.48**	II
Patch	Pasting	All	4.50	55.45	66.35	III
	Pasting	—		63.41	70.75	IV
	Pasting	Foreground		66.28	71.66	V
Object	Mixup	—	**2.60**	63.25	70.91	VI
	Mixup	Foreground		64.45	71.65	VII
	Pasting	All		52.26	65.97	VIII
	Pasting	—		63.12	70.52	IX
	Pasting	Foreground		**67.12**	**72.42**	X

Table 4.11. – **Rehearsal alternatives** on Pascal-VOC 2012 in mIoU (%). Object/Patch-based methods with 10 objects/patches per class, and Image-based with 10 images per class. All experiments done with PLOPLong.

pasting (see Section 4.4.2), by mixing pixels according to mixup (Hongyi Zhang et al. 2018) rule, or in the case of image rehearsal, no mixing is done. Finally, we consider two erasing methods: either all pixels are erased (*All*), or only pixels belonging to non-background classes are erased (*Foreground*). Note that for the latter method, it includes classes detected via pseudo-labeling. For all methods, we randomly select 10 images/patches/objects per class for rehearsal. Without surprise, Patch (III-V) and Object-based (VI-X) rehearsal are more data-efficient than images (I and II) (4.50 and 2.60 vs 22.20 Mb). Mixing the rehearsed data with mixup (VI and VII) is less efficient than direct pasting (III, IV, and VIII-X) because segmentation, contrary to classification, requires sharp boundaries (Y. Chen et al. 2020). We considered rehearsing each patch/object without mixing, but because their shape may vary, no batching is possible resulting in a slower training. Therefore, we investigate mixing patches/objects while erasing all others

pixels (III and VIII). This does not work, which is probably linked to the altered statistics of the batch normalization. On the other hand, we found that a selective erasing that remove foreground objects (V, VII, and X) while keeping the background of the destination image proved to be very effective (63.12% to 67.12% *all* mIoU for object). This confirmed our intuition that pasting in segmentation is a delicate operation that may lead to confusion in the network, particularly on object boundaries. The erased pixels are replaced by gray pixels. We considered more complicated approaches such as in-painting (Fang et al. 2019) or textures filling (Mallikarjuna et al. 2006), but were slightly less effective than our simpler solution. Overall, through careful design, we manage to get the best performance using our novel object rehearsal which was also the most memory-efficient.

Rehearsal alleviates pseudo-labeling's limitation: All previous methods, PLOP and PLOPLong included, did not consider rehearsal-learning. While popular in image classification, it has not been explored for continual segmentation. An attentive reader may wonder if rehearsal learning can provide benefits given the fact that our pseudo-labeling can uncover "hidden" old classes in the images. Even a perfect pseudo-labeling (considered in Table 4.8b) may fail in some particular data situation where the class distribution of a step is almost Dirac. We show in Figure 4.6 the distribution of pixels per class at the 5^{th} step of Pascal-VOC 15-1. The new class to learn, `train`, is abundant but also almost alone but the class `person`. In this case, no amount of pseudo-labeling can uncover previous classes like `bicycle` or `cat`. Our proposed rehearsal learning is complementary to our pseudo-labeling loss where the former can reduce the weakness for the latter in some cases.

4.5.4 Visualization

Figure 4.7 shows the predictions at the 6^{th} and final step of VOC 15-1 for four different models: MiB, PLOP, PLOPLong, and PLOPLong with Object Rehearsal. MiB struggles to even find the correct classes: in the first image the `car` is predicted as `train`, in the second the `plane` as `train`, and in the third no classes are detected. On the other hand, PLOP always manages to pick up the present classes, although sometimes with imperfection. PLOPLong further refines results, and the addition of Object Rehearsal produces a sharp and near perfect masks compared to the ground-truths (GTs). We showcased the harmful effect of background shift in Figure 4.8 where the class `person` is learned at 1^{st} step and the class `train` at 5^{th} step. MiB completely forgets the former class and overpredicts the latter. The background shift is efficiently mitigated with pseudo-labeling showed in the

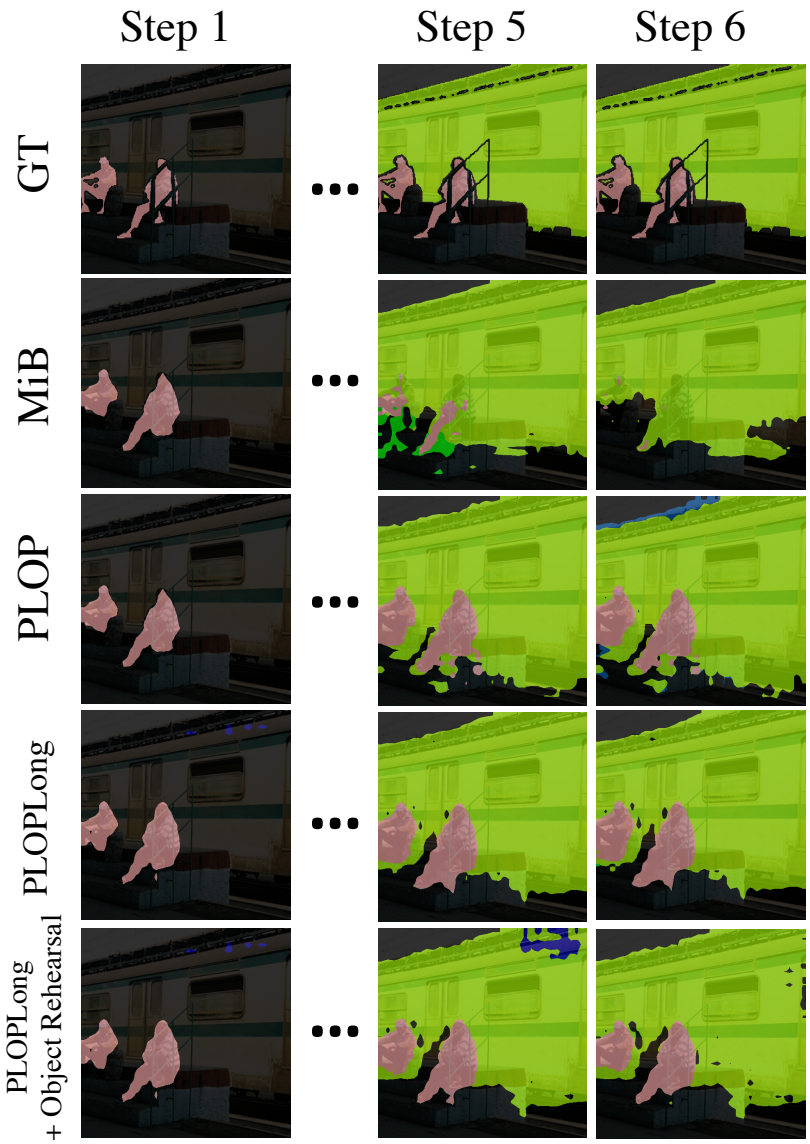


Figure 4.8. – **Visualization of the predictions with a background shift** of MiB, PLOP, PLOPLong, and PLOPLong with Object Rehearsal across time in VOC 15-1 on a test set image. At steps 1-4 only class `person` has been seen. At step 5, the class `train` is introduced, causing dramatic background shift. While MiB overfits on the new class and forget the old class, PLOP is able to predict both classes correctly. PLOPLong and PLOPLong + Object Rehearsal further refine the predicted masks, resulting in much sharper boundaries.

third row of PLOP. Our efficient proposition of object rehearsal allows further refinement of the predicted masks resulting in sharper boundaries.

4.6 Conclusion

Continual Semantic Segmentation (CSS) is an emerging but challenging computer vision domain. In this chapter, we highlighted two major issues in CSS: catastrophic forgetting and background shift. To deal with the former, we proposed Local POD, a multi-scale distillation scheme that preserves both long and short-range spatial statistics between pixels. This leads to an effective balance between rigidity and plasticity for CSS, which in turn alleviates catastrophic forgetting. We then tackled background shift with an efficient uncertainty-based pseudo-labeling loss. It completes the partially-labeled segmentation maps, allowing the network to efficiently retain previously learned knowledge. Afterward, we showed that carefully designed structural changes to the model could improve performance on long CSS scenarios, namely a cosine normalization adaptation of the classifier and Local POD followed by a modified batch normalization. Finally, we proposed to introduce rehearsal learning to CSS, one based on partially-labeled whole image rehearsal, and the other –much more memory-efficient– consisting in object rehearsal. The latter further refines our already effective model performance and enables real world applications of CSS with a stringent memory constraint. We evaluated the proposed PLOPLong, with or without object rehearsal, on three datasets and over twelve different benchmarks. Finally, we qualitatively validate our model through extensive ablations in order to better understand the performance gain.

CHAPTER 5

DYNAMIC STRATEGY WITH TRANSFORMERS

Chapter abstract

Continual models aim to learn an ever-growing number of tasks. Dynamic architectures expand their parameters to tackle this increasing complexity. However, existing approaches often require a task identifier at test-time, need complex tuning to balance the growing number of parameters, and barely share any information across tasks. As a result, they struggle to scale to a large number of tasks without significant overhead.

In this chapter, we propose a transformer architecture based on a dedicated encoder/decoder framework. Critically, the encoder and decoder are shared among all tasks. Through a dynamic expansion of special tokens, we specialize each forward of our decoder network on a task distribution. Our strategy scales to a large number of tasks while having negligible memory and time overheads due to strict control of the expansion of the parameters. Moreover, as a result, this efficient strategy doesn't need any hyperparameter tuning to control the network's expansion.

We evaluate our model on multiple datasets and show the interest of our parameter expansion strategy, striking the right balance between performance and overhead. We also extensively ablate our model to highlight each component contribution.

- *Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord (2022). "DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion". In:* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Contents

5.1 Introduction

Recent works expand the network architectures or re-arrange their internal structures using dynamic strategies (see Section 2.4.3). Unfortunately at test-time, they require to know the task to which the test sample belongs — in order to know which parameters should be used. More recently, DER (Yan et al. 2021) and Simple-DER (Zhuoyun Li et al. 2021) discarded the need for this task identifier by learning a single classifier on the concatenation of all produced embeddings by different subsets of parameters. Yet, these strategies induce an important memory overhead when tackling a large number of tasks, and thus need complex pruning as post-processing.

To improve the ease of use of continual learning frameworks for real-world applications, we aim to design a dynamically expandable representation (almost) "for free" by having the three following properties:

1. **limited memory overhead** as the number of tasks grows,
2. **limited time overhead** at test time,
3. **no setting-specific hyperparameters** for improved robustness when faced to an unknown (potentially large) number of tasks.

To this end, we leverage the computer vision transformers (refer to Section 2.2). Transformers (Vaswani et al. 2017) offer a very interesting framework to satisfy the previously mentioned constraints. Indeed, we build upon this architecture to design a **encoder/decoder strategy**: the encoder layers are shared among all

members of our dynamic network; the unique decoder layer is also shared, but its forward pass is specialized by a **task-specific learned token** to produce task-specific embeddings. Thus, the memory growth of the dynamic network is extremely limited: only a 384d vector per task, validating property #1. Moreover, this requires no hyperparameter tuning (property #3). Finally, the decoder is explicitly designed to be computationally lightweight (satisfying property #2). We nicknamed our framework, DyTox, for **DYnamic TOken eXpansion**.

Our strategy is robust to different settings, and can easily scale to a large number of tasks with minimal time and memory overheads. In particular, we validate the efficiency of our approach on CIFAR100, ImageNet100, and ImageNet1000 for multiple settings. We also perform ablations confirming the soundness of our dynamic strategy. In this chapter, we focus on continual dynamic networks (refer to Section 2.4.3) that expand their parameters coupled with rehearsal learning (Section 2.4.1). Moreover, to satisfy our desiderata listed previously, we base our framework on the recent Transformer architecture (see Section 2.2).

Positioning with respect to PODNet Note that while our methods PODNet (Section 3.2) and DyTox (presented in this chapter) are both designed for class-incremental image classification, they differ in terms of both objective and benchmark: PODNet aims to constrain the features drift to reduce forgetting, while DyTox conditions features to a particular task. Moreover, PODNet, a metric-based model, requires a larger initial task size to perform well. Therefore, PODNet was evaluated when half of the dataset's classes are seen in the first step. On the other hand, DyTox is considered here in the setting of Yan et al. (2021) where all steps, including the first one, bring an equal number of new classes.

5.2 DyTox transformer model

We evaluate our framework in the Class Incremental Learning (CIL) setting (Section 2.3) where each task brings an equal number of new classes. We also use rehearsal learning (Section 2.4.1). The Figure 5.1 displays our DyTox framework, which is made of several components (SAB, TAB, and Task Tokens) that we describe in the following sections.

5.2.1 Background

The vision transformer (Dosovitskiy et al. 2021) has three main components: the patch tokenizer, the encoder made of Self-Attention Blocks (SABs), and the

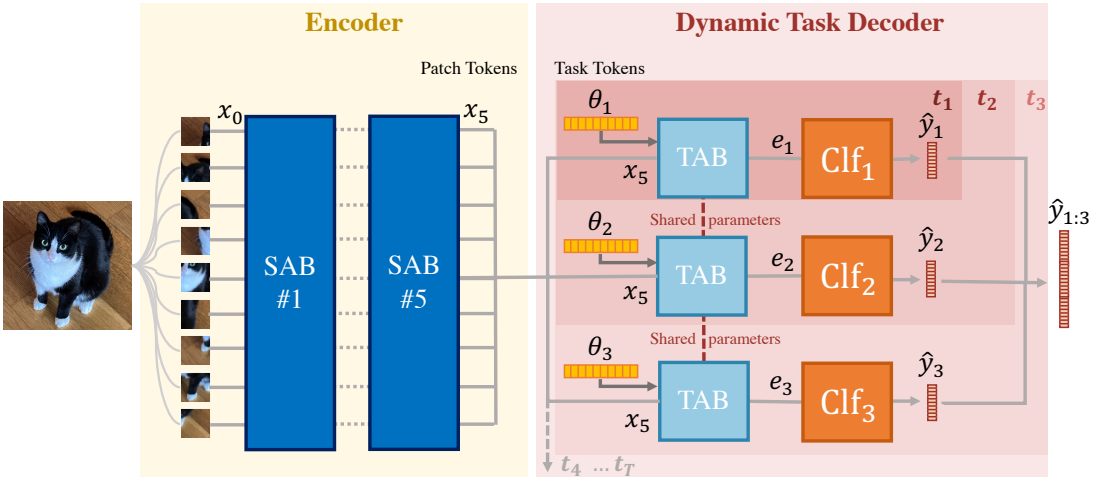


Figure 5.1. – **DyTox transformer model**. An image is first split into multiple patches, embedded with a linear projection. The resulting patch tokens are processed by 5 successive Self-Attention Blocks (SAB) (Section 2.2). For each task ($t = 1 \dots T$), the processed patch tokens are then given to the Task-Attention Block (TAB) (Section 5.2.2): each forward through the TAB is modified by a different task-specialized token θ_t for $t \in \{1 \dots T\}$ (Section 5.2.3). The T final embeddings are finally given separately to independent classifiers Clf_t each predicting their task's classes C^t. All $|C^{1:T}|$ logits are activated with a sigmoid. For example, at task $t = 3$, one forward is done through the SABs and three task-specific forwards through the unique TAB.

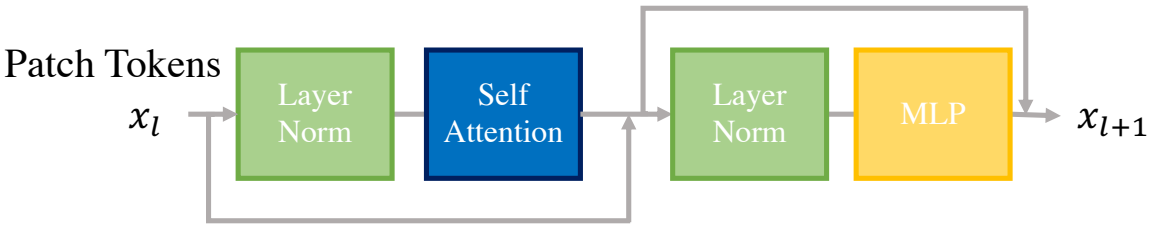


Figure 5.2. – **The Self-Attention Block (SAB)** combines a Self-Attention (SA), two Layer Norms, and one MLP with a single hidden layer. As in a ResNet, two shortcuts are used with element-wise addition.

classifier. We described in the related work (Section 2.2) the general architecture. A SAB is depicted in Figure 5.2.

Importantly, recall that in the original vision transformer ViT (Dosovitskiy et al. 2021), a learned vector called the "*class token*" is appended to the patch tokens after the tokenizer. This special class token, when processed after all the SABs, is given to a linear classifier with a softmax activation to predict the final probabilities. However, more recent works, as CaiT (Touvron et al. 2021b), propose instead to introduce the class token only at the ultimate or penultimate SAB to improve classification performance.

5.2.2 Task-Attention Block (TAB)

Contrary to previous transformer architectures, we don't have a class token, but rather what we nicknamed "**task tokens**"; the learned token of the i^{th} task is denoted θ_i. This special token will only be added at the last block. To exploit this task token, we define a new attention layer, that we call the Task-Attention. It first concatenates the patch tokens x_L produced by the ultimate SAB with a task token θ_i:

$$z_i = [\theta_i, x_L] \in \mathbb{R}^{(\mathbb{N}+1)\times\mathbb{D}}. \tag{5.1}$$

This is then given to the Task-Attention (TA), inspired by the Class-Attention of Touvron et al. (Touvron et al. 2021b):

$$\begin{aligned} Q_i &= W_q\theta_i\,, \\ K_i &= W_k z_i\,, \\ V_i &= W_v z_i\,, \\ A_i &= \text{Softmax}\left(Q_i \cdot K_i^T/\sqrt{d/h}\right)\,, \\ O_i &= W_o A_i V_i + b_o \in \mathbb{R}^{1\times\mathbb{D}}\,, \end{aligned} \tag{5.2}$$

with d being the embedding dimension, and h the number of attention heads (Vaswani et al. 2017). Contrary to the classical Self-Attention, the Task-Attention defines its query (Q_i) only from the task-token θ_i without using the patch tokens x_L. The Task-Attention Block (TAB) is then a variation of the SAB where the attention is a Task-Attention (TA):

$$\begin{aligned} c' &= c + \mathrm{TA}\left(\mathrm{Norm}_1\left(z\right)\right) , \\ c'' &= c' + \mathrm{MLP}\left(\mathrm{Norm}_2\left(c'\right)\right) . \end{aligned} \tag{5.3}$$

Overall, our new architecture can be summarized by the repetition of SA Blocks $\{\mathrm{SAB}_l\}_{l=1}^{L}$ (defined in Equation 2.4) ended by a single TA Block TAB (defined in Equation 5.3):

$$e_i = \mathrm{TAB} \circ \left([\theta_i, \mathrm{SAB}_{l=L} \circ \ldots \mathrm{SAB}_{l=1}(x_0)]\right) \in \mathbb{R}^D . \tag{5.4}$$

The final embedding e_i is fed to a classifier clf made of a Norm_c and a linear projection parametrized by $\{W_c, b_c\}$:

$$\tilde{y}_i = \mathrm{Clf}(e_i) = W_c \, \mathrm{Norm}_c(e_i) + b_c \, . \tag{5.5}$$

5.2.3 Dynamic task token expansion

We defined in the previous section our base network, made of a succession of SABs and ended by a single TAB. As detailed, the TAB has two inputs: the patch tokens x_L extracted from the image and a learned task-token θ_i. We'll now detail how our framework evolves in a continual situation at each new step.

During the first step, there is only one task token θ_1. At each new step, we propose to expand our parameter space by creating a new task token while keeping the previous ones. Thus, after t steps, we have t task tokens ($\theta_i \, \text{for} \, i \in \{1 \ldots t\}$). Given an image x — belonging to any of the seen tasks $\{1 \ldots t\}$ — our model tokenizes it into x_0, and processes it through the multiple SABs: this outputs the patch tokens x_L. Finally, our framework does as many forward passes through the TAB as there are tasks: critically, each TAB forward passes is executed with a different task token θ_i, resulting in different task-specific forwards, each producing the task-specific embeddings e_i (see Figure 5.1):

$$\begin{aligned} e_1 &= \mathrm{TAB}([\theta_1, x_L]) , \\ e_2 &= \mathrm{TAB}([\theta_2, x_L]) , \\ &\ldots \\ e_t &= \mathrm{TAB}([\theta_t, x_L]) . \end{aligned} \tag{5.6}$$

Algorithm 5.1 DyTox's forward pass at step t

Input: x_0 (initial patch tokens), y (ground-truth labels)
Output: $\hat{y}_{1:t}$ (predictions for all classes of $\mathcal{C}^{1:t}$)

$x_L \leftarrow \text{SAB}_{l=L} \circ ... \text{SAB}_{l=1}(x_0)$ ▷ Section 5.2.1
for $i \leftarrow 1\,;\ i \leq t\,;\ i{+}{+}$ **do**
  $e_i \leftarrow \text{TAB}([\theta_i, x_L])$ ▷ Section 5.2.2
  $\hat{y}_i \leftarrow \text{Clf}_i(e_i)$ ▷ Section 5.2.3
end for
$\hat{y}_{1:t} \leftarrow [\hat{y}_1, \ldots, \hat{y}_t]$

Rather than concatenating all embeddings $\{e_1, e_2, \ldots, e_t\}$ together and feeding them to one classifier, we leverage **task-specific classifiers**. Each classifier clf_i is made of a Ba et al. (2016)'s Layer Norm (Norm_i) and a linear projection parametrized by $\{W_i, b_i\}$, with $W_i \in \mathbb{R}^{\mathcal{C}^i \times D}$ and $b \in \mathbb{R}^{\mathcal{C}^i}$. It takes as input its task-specific embedding e_i and returns:

$$\hat{y}_i = \text{Clf}_i(e_i) = \sigma(W_i \,\text{Norm}_i(e_i) + b_i)\,, \tag{5.7}$$

the predictions for the classes $y_i \in \mathcal{C}^i$, where $\sigma(x) = {}^1\!/\!_{(1+e^{-x})}$ is the sigmoid activation. In comparison with the softmax activation, the element-wise sigmoid activation reduces the overconfidence in recent classes. Consequently, the model is better calibrated, which is an important attribute of continual model (Belouadah and Popescu 2019; Wu et al. 2019; B. Zhao et al. 2020). The loss is the binary-cross entropy. The independent classifiers paradigm coupled with the sigmoid activation and binary cross-entropy loss exclude explicitly a late fusion (Ramachandram and Taylor 2017) of the task embeddings resulting in more **specialized classifiers**.

The overall structure of the DyTox strategy is illustrated in Figure 5.1. We also show in Algorithm 5.1 the pseudo-code of a forward pass at test-time after having learned the task t. Critically, the test image can belong to any of the previously seen tasks $\{1 \ldots t\}$. Our dynamic task token expansion is more efficient than a naive parameter expansion that would create a new copy of the whole network for each new task. (1) Our expansion is limited to a new task token per new task, which is only $d = 384$ new parameters. This is small compared to the total model size ($\approx$ 11 million parameters). The **memory overhead is thus almost null**. (2) The computationally intensive blocks (*i.e.*, the SABs) are executed only once despite learning multiple tasks. In contrast, the TAB has as many forwards as there are tasks. Though, this induces minimal overhead because the **Task-Attention has a linear complexity w.r.t the number of patches** while the Self-Attention is quadratic. Therefore, the time overhead is sub-linear. We quantitatively show this in Section 5.3.

Context The current transformer paradigm starting from BERT (Devlin et al. 2018) and continuing with ViT (Dosovitskiy et al. 2021) is based on a encoder+classifier structure. Differently, our dynamic framework strays is a resurgence of the encoder/decoder structure of the original transformer (Vaswani et al. 2017): the encoder is shared (both in memory and execution) for all outputs. The decoder parameters are also shared, but its execution is task-specific with each task token, with each forward akin to a task-specific expert chosen from a mixture of experts (Masoudnia and Ebrahimpour 2014). Moreover, multi-tasks text-based transformers have natural language tokens as an indicator of a task (Raffel et al. 2019) (*e.g.* "summarize the following text"), in our context of vision we used our defined task tokens as indicators.

Losses Our model is trained with three losses: (1) the classification loss $\mathcal{L}_{\text{clf}}$, a binary-cross entropy, (2) a knowledge distillation (Geoffrey Hinton et al. 2015) $\mathcal{L}_{\text{kd}}$ applied on the probabilities, and (3) the divergence loss $\mathcal{L}_{\text{div}}$. The distillation loss helps to reduce forgetting. It is arguably quite naive, and more complex distillation losses (Selvaraju et al. 2017; Hou et al. 2019) could further improve results. The divergence loss, inspired from the "auxiliary classifier" of DER (Yan et al. 2021), uses the current last task's embedding e_t to predict $(|\mathcal{C}^t|+1)$ probabilities: the current last task's classes $\mathcal{C}^t$ and an extra class representing all previous classes that can be encountered via rehearsal. This classifier is discarded at test-time and encourages a better diversity among task tokens. The total loss is:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{clf}} + \alpha\mathcal{L}_{\text{kd}} + \lambda\mathcal{L}_{\text{div}}\,, \tag{5.8}$$

with λ a hyperparameter set to 0.1 for **all** experiments. α correspond to the fraction of the number of old classes over the number of new classes $\frac{|C^{1:t-1}|}{|C^{1:t}|}$ as done by B. Zhao et al. (2020). Therefore, α is automatically set; this removes the need to finely tune this hyperparameter.

5.2.4 Improved Continual Training Procedure

We nicknamed our model described previously DyTox. In this section, we propose two modifications of the training procedure aimed at improving the continual performance.

DyTox+ We introduce a new efficient training procedure for continual learning. Using MixUp (Hongyi Zhang et al. 2018), we linearly interpolate new samples with existing samples. The interpolation factor $\lambda \sim \text{Beta}(\alpha, \alpha)$ is sampled with $\alpha = 0.8$: the pixels of two images are mixed ($x = \lambda x_1 + (1-\lambda)x_2$) as their labels

($y = \lambda y_1 + (1 - \lambda) y_2$). MixUp was shown to have two main effects: (1) it diversifies the training images and thus enlarges the training distribution on the vicinity of each training sample (Chapelle et al. 2001) and (2) it improves the network calibration (Guo et al. 2017; Thulasidasan et al. 2019), reducing the overconfidence in recent classes. Thus, MixUp has shared motivation with the sigmoid activation. When DyTox is combined with this MixUp procedure, nicknamed as DyTox+, the forgetting is significantly reduced as shown in experiments.

DyTox++ We nicknamed DyTox+ our model when combined with a novel continual procedure based on MixUp (Hongyi Zhang et al. 2018). We now refine DyTox+ into DyTox++ by adding a new component during the training: the Sharpness-Aware Minimizer (SAM) (Foret et al. 2021). Indeed, **aiming for wider minima** is particularly important in continual learning (Kirkpatrick et al. 2017; Verwimp et al. 2021). This is because sharp task-specific minima lead to overspecialization to a particular task and consequently to a forgetting of all other tasks. Weights constraints as EWC (Kirkpatrick et al. 2017) or second-order optimization (J. Lee et al. 2020) have similar motivations. SAM estimates the worst closest parameters during a first forward/backward pass, and then optimizes the loss w.r.t. to them during a second forward/pass. In consequence, DyTox++ optimizes the loss not w.r.t. the current parameters but w.r.t. a region of possible parameters leading to wide local minima that span across multiple tasks. In practice, we used the Adaptive SAM (ASAM) (Kwon et al. 2021), an extension of SAM that is more robust to hyperparameters.

5.3 Experiments

5.3.1 Benchmarks & implementation

Benchmarks & Metrics We evaluate our model on CIFAR100 (Krizhevsky and Geoffrey Hinton 2009), ImageNet100 and ImageNet1000 (Deng et al. 2009) (descriptions in the supplementary materials) under different settings. The standard continual scenario in ImageNet has 10 steps: thus we add 10 new classes per step in ImageNet100, and 100 new classes per step in ImageNet1000. In CIFAR100, we compare performances on 10 steps (10 new classes per step), 20 steps (5 new classes per step), and 50 steps (2 new classes per step). In addition to the top-1 accuracy, we also compare the top-5 accuracy on ImageNet. We report the "*Avg*" accuracy which is the average of the accuracies after each step as defined by (Rebuffi et al. 2017c). We also report the final accuracy after the last step ("*Last*").

Hyperparameter	CIFAR	ImageNet
# SAB	5	
# TAB	1	
# Attentions Heads	12	
Embed Dim	384	
Input Size	32	224
Patch Size	4	16

Table 5.1. – **DyTox's architectures** for CIFAR and ImageNet. The only difference between the two architectures is the patch size, as the image sizes vary between datasets.

Finally, in our tables, "*#P*" denotes the parameters count in million after the final step.

Implementation details As highlighted in Table 5.1, our network has the same structure across all tasks. Specifically, we use 5 Self-Attention Blocks (SABs), 1 Task-Attention Block (TAB). All 6 have an embedding dimension of 384 and 12 attention heads. We designed this shallow transformer to have a comparable parameters count to other baselines, but also made it wider than usual "tiny" models (Dosovitskiy et al. 2021; Touvron et al. 2021a; Touvron et al. 2021b). We tuned all hyperparameters for CIFAR100 with 10 steps on a validation set made of 10% of the training set, and then kept them fixed for all other settings, ImageNet included. The only difference between the two datasets is that ImageNet images are larger; thus the patch size is larger, and overall the base transformer has slightly more parameters on ImageNet than on CIFAR (11.00M vs 10.72M) because of a bigger positional embedding. We use the attention with spatial prior (introduced by ConViT (d'Ascoli et al. 2021)) for all SABs, which allows training transformers on a small dataset (like CIFAR) without pretraining on large datasets or complex regularizations. Following previous works (Rebuffi et al. 2017c; Yan et al. 2021), we use for all models (baselines included) 2,000 images of rehearsal memory for CIFAR100 and ImageNet100, and 20,000 images for ImageNet1000. The Continuum library (Douillard and Lesort 2021) provides the implementations of the continual scenarios. Our network implementation is based on the DeiT (Touvron et al. 2021a) code base, which itself extensively uses the timm library (Wightman 2019). The code is released publicly [1]. The full implementation details are in the appendix (Section A.5).

1. https://github.com/arthurdouillard/dytox.

Methods	#P	ImageNet1000 10 steps top-1 Avg	top-1 Last	top-5 Avg	top-5 Last
ResNet18 joint	11.68	-	-	-	89.27
Transf. joint	11.35	-	73.58	-	90.60
LwF-MC (Rebuffi et al. 2017c)	11.68	-	-	48.45	25.06
E2E (Castro et al. 2018)	11.68	-	-	72.09	52.29
Simple-DER (Zhuoyun Li et al. 2021)	28.00	66.63	59.24	85.62	80.76
iCaRL (Rebuffi et al. 2017c)	11.68	38.40	22.70	63.70	44.00
BiC (Wu et al. 2019)	11.68	-	-	84.00	73.20
WA (B. Zhao et al. 2020)	11.68	65.67	55.60	86.60	81.10
DER w/o P (Yan et al. 2021)	116.89	68.84	60.16	88.17	82.86
DER† (Yan et al. 2021)	-	66.73	58.62	87.08	81.89
DyTox	11.36	**71.29**	**63.34**	**88.59**	**84.49**

Table 5.2. – **Results on the ImageNet-1000 dataset**, learned with 10 steps of respectively 10 and 100 new classes. E2E and Simple-DER results come from their respective papers, and used a different class ordering. Other results come from Yan et al. (2021). The † symbol means that Yan et al. (2021) needed setting-sensitive hyperparameters. Moreover, its reported parameters count was an average over all steps (Yan et al. (2021) reported 14.52M on ImageNet1000): the final parameters count (necessarily higher) was not available.

5.3.2 Quantitative results

ImageNet We report performances in Table 5.2 and in Table 5.3 for respectively the ImageNet-1000 and ImageNet-100 dataset. The † marks the DER with setting-specific pruning, and DER w/o P is for the DER without pruning. Critically, on the larger-scale ImageNet1000, DyTox systematically performs best on all metrics despite having lower parameters count. Specifically, DyTox reaches 71.29% in "Avg" top-1 accuracy, and 63.34% in "Last" top-1 accuracy. This outperforms the previous state-of-the-art DER w/o P (68.84% in "Avg", 60.16% in "Last") which has 10 ResNet18 in parallel and 116.89M parameters. Compared to the pruned DER†, DyTox has a +4.56 *p.p.* in top-1 and a +1.51 *p.p.* in top-5 for the "Avg" accuracy. In ImageNet100, DyTox reaches 69.10% and outperforms DER† by +3.04 percentage points (*p.p.*) in "Last" top-1 accuracy. Though, DyTox and DER w/o P somehow perform similarly in "Avg" accuracy, DyTox+ and DyTox++ reach state-of-the-art performance. Specifically, DyTox++, using our full improved continual training procedure, improves the "Avg" top-1 accuracy of DER by 3.58 *p.p.*.

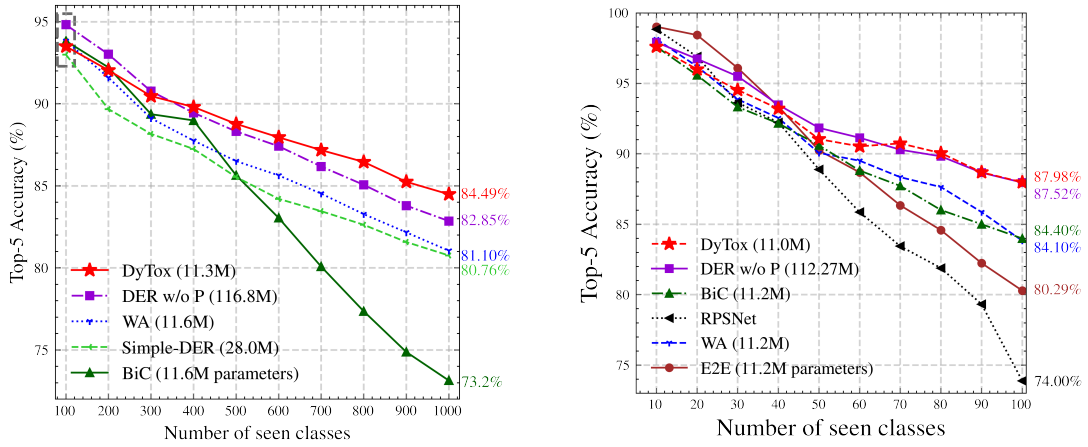
Methods	#P	top-1		top-5	
		Avg	Last	Avg	Last
ResNet18 joint	11.22	-	-	-	95.1
Transf. joint	11.00	-	79.12	-	93.48
iCaRL (Rebuffi et al. 2017c)	11.22	-	-	83.60	63.80
RPSNet (Rajasegaran et al. 2019)	-	-	-	87.90	74.00
E2E (Castro et al. 2018)	11.22	-	-	89.92	80.29
BiC (Wu et al. 2019)	11.22	-	-	90.60	84.40
WA (B. Zhao et al. 2020)	11.22	-	-	91.00	84.10
DER w/o P (Yan et al. 2021)	112.27	77.18	66.70	93.23	87.52
DyTox	11.01	77.15	69.10	92.04	87.98
DyTox+	11.01	79.22	69.06	93.72	88.82
DyTox++	11.01	**80.76**	**72.46**	**94.40**	**90.10**

Table 5.3. – **Results on ImageNet-100** with 10 steps of 10 new classes each. WA and DER w/o P results are reported from Yan et al. (2021). DyTox+ uses MixUp (Hongyi Zhang et al. 2018) in addition of the DyTox strategy, DyTox++ further adds a sharpness-aware minimizer (Foret et al. 2021).

All models evolutions on ImageNet-1000 and ImageNet-100 are illustrated in Figure 5.3: DyTox constantly surpasses previous state-of-the-art models — despite having a comparable performance at the first step and fewer parameters.

DyTox is able to scale correctly while handling seamlessly the parameter growth by sharing most of the weights across tasks. In contrast, DER had to propose a complex pruning method; unfortunately, this pruning required different hyperparameter values for different settings. Despite this, the pruning in DER† is less efficient when classes diversity increase: DER† doubles in size between ImageNet100 and ImageNet1000 (Yan et al. (2021) report 7.67M *vs.* 14.52M) while handling the same amount of tasks (10). Note that these parameter counts reported for DER† in Yan et al. (2021) are in fact averages over all steps: the final parameters count (necessarily higher) was not available and thus is not reported in our tables. Simple-DER also applies pruning but without hyperparameter tuning; while simpler, the pruning is also less efficient and induces larger model (28.00M parameters).

CIFAR100 Table 5.4 shows results for all approaches on CIFAR100. The more steps there are, the larger the forgetting is and thus the lower the performances are. Those settings are also displayed in Figure 5.4 after each task. In every setting, DyTox is close to DER w/o P for much fewer parameters (up to 52x less). Critically, DyTox is significantly above other baselines: *e.g.* DyTox is up to +25% in "Last" accuracy in the 50 steps setup. Note that our improved continual training



(a) **ImageNet1000** (b) **ImageNet100**

Figure 5.3. – **Performance evolution on ImageNet-{100, 1000}.** The top-5 accuracy (%) is reported after learning each task. Our model DyTox (in **red**) reaches state-of-the-art performance while using significantly fewer parameters than concurrent models. Note that at the initial step before the continual process begins, our model has performance comparable to other baselines: the performance gain is achieved by reducing catastrophic forgetting.

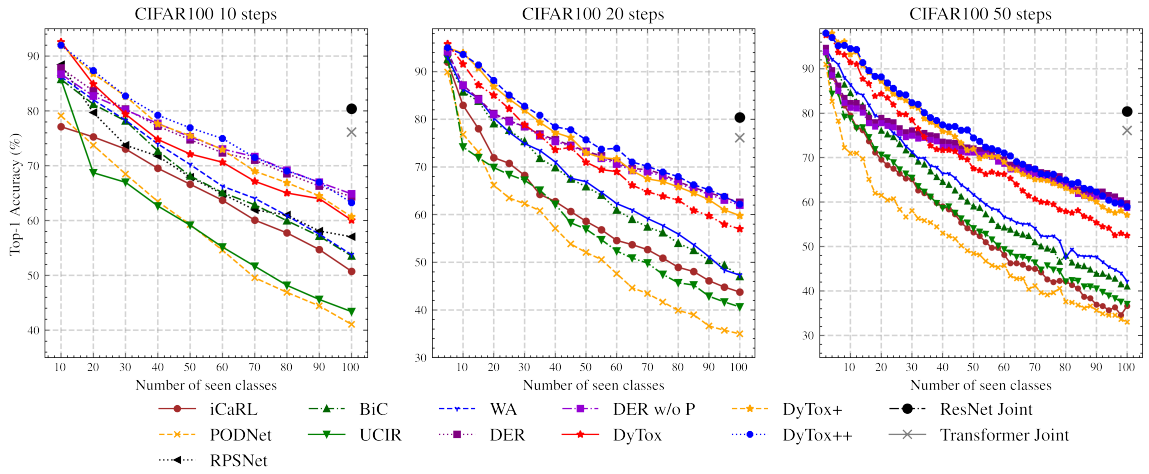


Figure 5.4. – **Performance evolution on CIFAR100**. The top-1 accuracy (%) is reported after learning each task. **Left** is evaluated with 10 steps, **middle** with 20 steps, and **right** with 50 steps.

procedure, with DyTox+ and DyToX++, further increases the results in all settings. Notably DyTox++ increases the "Àvg" accuracy in the 50 setup over DER by +3.4 *p.p.*. Remark also that DER "Avg" accuracy degrades by 2.59 *p.p.* between the easiest 10 steps setting and the hardest 50 steps setting. In comparison, DyTox++ only loses 1.65 *p.p.*, proving a better robustness to forgetting as the number of

Methods	10 steps			20 steps			50 steps		
	#P	**Avg**	**Last**	**#P**	**Avg**	**Last**	**#P**	**Avg**	**Last**
ResNet18 Joint	11.22	-	80.41	11.22	-	81.49	11.22	-	81.74
Transf. Joint	10.72	-	76.12	10.72	-	76.12	10.72	-	76.12
iCaRL (Rebuffi et al. 2017c)	11.22	65.27 ± 1.02	50.74	11.22	61.20 ± 0.83	43.75	11.22	56.08 ± 0.83	36.62
UCIR (Hou et al. 2019)	11.22	58.66 ± 0.71	43.39	11.22	58.17 ± 0.30	40.63	11.22	56.86 ± 0.83	37.09
BiC (Wu et al. 2019)	11.22	68.80 ± 1.20	53.54	11.22	66.48 ± 0.32	47.02	11.22	62.09 ± 0.85	41.04
WA (B. Zhao et al. 2020)	11.22	69.46 ± 0.29	53.78	11.22	67.33 ± 0.15	47.31	11.22	64.32 ± 0.28	42.14
PODNet	11.22	58.03 ± 1.27	41.05	11.22	53.97 ± 0.85	35.02	11.22	51.19 ± 1.02	32.99
DER w/o P (Yan et al. 2021)	112.27	75.36 ± 0.36	**65.22**	224.55	74.09 ± 0.33	62.48	561.39	72.41 ± 0.36	59.08
DER†	-	74.64 ± 0.28	64.35	-	73.98 ± 0.36	**62.55**	-	72.05 ± 0.55	**59.76**
DyTox	10.73	73.66 ± 0.02	60.67 ± 0.34	10.74	72.27 ± 0.18	56.32 ± 0.61	10.77	70.20 ± 0.16	52.34 ± 0.26
DyTox+	10.73	**75.54** ± 0.10	62.06 ± 0.25	10.74	**75.04** ± 0.11	60.03 ± 0.45	10.77	**74.35** ± 0.05	57.09 ± 0.13
DyTox++	10.73	**77.10** ± 0.08	64.53 ± 0.08	10.74	**76.57** ± 0.18	**62.44** ± 0.22	10.77	**75.45** ± 0.19	58.76 ± 0.28

Table 5.4. – **Results on CIFAR100** averaged over three different class orders. Baselines results are come from Yan et al. (2021). The † symbol means that Yan et al. (2021) needed setting-sensitive hyperparameters. Moreover, its reported parameters count was an average over all steps: the final parameters count (necessarily higher) was not available. DyTox+ uses MixUp (Hongyi Zhang et al. 2018) and DyTox++ uses both MixUp and SAM (Kwon et al. 2021).

tasks increases. Remark that the results in this table of our PODNet, presented in Chapter 3, can be explained because of the slightly different setting: PODNet was designed for situation where half of the dataset's classes were learned in a single step, and thus providing a better initialization. PODNet, a metric-based model (as also UCIR), excelled in those situations, but struggle when the initial step contains few classes (as it is the case in this chapter) due to slower initial learning. Arguably, in a real-life scenario, a model should be pretrained on a large dataset and therefore this "*weakness*" of PODNet won't materialize.

5.3.3 Model introspection on CIFAR100

Memory overhead We only add a vector of size $d = 384$ per task; thus, the overhead in memory (not considering the growing classifier which is common for all continual models) is only of $+0.004\%$ per step. Even in the challenging setting of CIFAR100 with 50 tasks, our memory overhead is almost null ($+0.2\%$).

Computational overhead The vast majority of the computation is done in the SABs, thus shared among all tasks. The dynamical component of our model is located at the ultimate TAB. Moreover, the Task-Attention, contrary to the Self-Attention, has a time complexity linear in terms of tokens and not quadratic reducing the time overhead to an acceptable sub-linear amount. Overall, for each new task, one forward pass is only 2.24% slower than at the previous task. Furthermore, the procedure can be accelerated by doing a single forward pass through

	Joint (1 step)	50 steps	
Training	**Last** (↑)	**Last** (↑)	**Forgetting** (↓)
DyTox	76.12	52.34	33.15
DyTox+	77.51+1.39	57.09+4.75	31.50-1.65
DyTox++	77.91+0.40	58.76+1.67	30.47-1.03

Table 5.5. – **"Last" accuracy and forgetting** on CIFAR100 for the joint (1 step, no continual) and 50 steps settings.

the TAB with a masked attention (Vaswani et al. 2017): the query is the concatenation of all task tokens, then we mask the attention logits corresponding to an interaction between task tokens. For an almost equivalent result (modulo numerical imprecision), a new task only increases the time spent in a forward pass by 1.09%.

Training procedure introspection Our DyTox+ and DyTox++ strategies really reduce catastrophic forgetting and does not just improve raw performances. This is shown in Table 5.5, where we compare DyTox vs DyTox+ vs DyTox++ strategies on CIFAR100. In the joint setting, our model slightly benefits from both MixUp and ASAM: the gain is limited (+1.79 *p.p.*). On the other hand, those two methods greatly improve the extreme continual setting of 50 steps (+6.42 *p.p.*). This shows that the gain is not due to absolute improvements of the model performance. Moreover, using the forgetting measure of Chaudhry et al. (2018), we compare how much a model has forgotten relatively to its previous tasks. This metric is therefore agnostic to absolute performance improvements. DyTox had a forgetting of 33.15%, DyTox+ of 31.50%, and DyTox++ of 30.47%: a total reduction of 2.68 *p.p.*. This validates our novel training procedures that are particularly efficient for continual learning. The computational overhead of ASAM is lower than more complex second-order methods, but it still doubles the number of forward and backward passes. For this reason, we didn't evaluated DyTox++ on the large ImageNet1000. However, future works could consider the promising Look-SAM Anonymous 2021 to reduce the time overhead.

		Knowledge Distillation	Finetuning	Token Expansion	Divergence Classifier	Indendepent Classifiers	Avg	Last
DyTox	Transformer						60.69	38.87
		✓					61.62	39.35
		✓	✓				63.42	42.21
	Dynamic	✓	✓	✓			67.30	47.57
		✓	✓	✓	✓		68.28	49.45
		✓	✓	✓	✓	✓	70.20	52.34

Table 5.6. – **Ablations** of the different key components of our DyTox architecture. We report the average accuracy and the last accuracy on CIFAR100 for the setting with 50 steps.

Model ablations We ablate the importance of the different components of DyTox in Table 5.6. We add on the base transformer a naive knowledge distillation (Geoffrey Hinton et al. 2015) and a finetuning (Castro et al. 2018) applied after each task on a balanced set of new data and rehearsal data. Finally, our DyTox strategy exploits directly the very nature of transformers (separated task information from the pixels information) to tackle catastrophic forgetting with three components: (1) a task token expansion, (2) a divergence classifier, and (3) independent classifiers. All three greatly improve over the baseline transformer ($42.21\% \rightarrow 52.34\%$ in "Last") while having almost no memory overhead ($+0.2\%$). The divergence classifier improves the diversity between task tokens: we observed that the minimal Euclidean distance between them increases by 8%. Moreover, we also remarked that having independent classifiers reduces the forgetting defined by Chaudhry et al. (2018) by more than 24%.

5.4 Conclusion

In this chapter, we covered our work on dynamic architectures. While in previous chapters, we aimed to constrain the visual features, we decided here to condition the features to specific tasks. With DyTox, a new dynamic strategy for continual learning based on transformer architecture, all tasks share a common encoding produced by self-attention layers. Then, task-specific tokens are used to produce task-specialized embeddings through a new task-attention layer. This architecture allows to dynamically process new tasks with very little memory

overhead and does not require complex hyperparameter tuning. Our experiments show that our framework scales to large datasets and an important number of tasks efficiently while using significantly fewer parameters than concurrent dynamic strategies.

CHAPTER 6

CONCLUSION

We now summarize the contributions of this thesis and offer some future directions of Continual Learning.

6.1 Contributions

During this thesis, we aim to learn an increasing number of classes with Deep Learning (DL) architectures for Computer Vision (CV) without forgetting. We design multiple methods to achieve this goal, with a particular interest on how the visual features of a continual model evolve through time. First, in Chapter 3, we investigate how to constrain features while satisfying a rigidity-plasticity trade-off. Then, in Chapter 4, we explore continual approaches for semantic segmentation. Finally, in Chapter 5, we exploit the transformer architecture in a dynamic framework to condition features per task.

Visual Features regularization We study in Chapter 3 Class Incremental Learning (CIL) for image classification. In this setting, regularization constraining a model's output is the most common approach. We challenge this paradigm by outlining two drawbacks: it balances poorly the rigidity (not forgetting old classes) vs plasticity (learning new classes) trade-off. Moreover, constraining intermediary visual features is a stronger regularization. Then, we design two feature-based regularizations: (1) PODNet minimizes the drift between statistics of the visual features between the old and new models. The design of this method explicitly reduces forgetting while letting enough slack to efficiently learn new classes. (2) Our second approach, Ghost, avoids forgetting before it even happens by pre-allocating areas of the latent space for future classes by drawing inspiration from the zeroshot literature. For this second approach, we propose the Prescient Continual Learning setting where detailed attributes of each class are available. This is a reasonable assumption in the fashion context of Heuritech, the company sponsoring this PhD.

Continual Semantic Segmentation We explore Continual Semantic Segmentation (CSS) in Chapter 4. We highlight the two main challenges: an important catastrophic forgetting linked to the higher complexity of segmentation images, and a background shift where only classes of the current task are labeled. To reduce the catastrophic forgetting of old classes, inspired by our previous POD, we present a multi-scale distillation loss that constrains local regions of visual features. Then, to tackle the background shift, we design an uncertainty-based hard pseudo-labeling loss. We show that despite its usefulness, our pseudo-labeling can fail for particular situations, and complement it with an efficient object rehearsal method.

Dynamic Strategy with Transformers Finally, in Chapter 5, we propose to use the recent transformer architecture with a dynamic strategy in Class Incremental Learning (CIL) for image classification. Previous dynamic networks, that expand their parameters as the number of learned tasks increases, struggle to limit their memory and time overheads. We propose instead to share a common encoding produced by self-attention layers, and to condition the features for each task using task-specific tokens. This architecture allows us to dynamically process new tasks with very little memory overhead even when faced to a large number of tasks.

6.2 Future Work

We now discuss some future directions of our work, both with respect to the data/benchmark and to the architecture/optimization process.

Data & Benchmarks

Time Limitation Current works in the literature, including this thesis, focus on using no rehearsal data or at least very few. The common justifications are about private data that cannot be kept, or embedded device with little storage. In many situations, those constraints are realistic. However, in other situations, given large data centers, storage capacity is less a problem. In that case, the main constraint is time: continually learn new data should be significantly faster than retraining from scratch. Thus, a new setting with large-scale rehearsal memory for continual learning would impose a time budget (Veniat and Denoyer 2018).

Universal Representation A major cause of forgetting in deep neural networks is that they learn spurious features (Lesort 2022), useful for the current task, but

that may cause interference with future tasks. Ideally, a network should learn invariants (Arjovsky et al. 2019; Rame et al. 2021a) that generalizes better to new distributions. Recently, it has been proposed that self-supervision could learn more class-agnostic features and in turns drastically reduce forgetting (Gallardo et al. 2021). Universal features, trained in self-supervision, with a metric-based approach, on a wide diversity of modalities (RGB, depth, *etc.*), will enable better continual models. First, they would forget less because the need to adapt the representation to new tasks is reduced. Second, an adaptation to new tasks will be extremely quick, where simply tuning a task token in a Dytox-like strategy would result in good performances.

Architecture & Optimization

Deep Architectures for continual learning is a fewly explored topic as most works focus on a fixed MLP or a ResNet-18. Initial findings remark that larger networks forget less (Ramasesh et al. 2022), especially when scaling the width (Mirzadeh et al. 2022). Going further than simply scaling, one may wonder if the structure of current architectures, optimized for *i.i.d.* training, have inherent flaws regarding continual learning. More particularly, Mixture-of-Experts (MoE) is an interesting direction for continual learning (L. Caccia et al. 2022): large models could learn universal features that could be conditioned with task-specific experts. The usage of experts does not necessarily need to be exclusive to a particular task: in fact a form of compositionality could be obtained where each expert specializes a some concept. This compositionality could speed up the learning of new tasks with little forgetting without requiring an important adaptation of the parameters.

Second-order optimization methods can help deep neural networks to find wide local minima, which could encompass multiple task local minima (J. Lee et al. 2020). As a result, the parameter drift between the optimal weights of a task t and $t+1$ can be minimal, and in turn avoid the bulk of the forgetting. We briefly explore this direction with the Sharpness-Aware-Minimizer (Foret et al. 2021) in this thesis, but more work could be done on this topic. An important drawback of current second-order methods is obviously the higher computational cost they incur, thus faster alternative methods also seeking flat wide minima such as Cha et al. (2021)'s SWAD could be explored.

Learning differently The vast majority of deep architectures, in all domains, are trained with gradient descent (including fancier optimizers as Adam or LAMB). The major drawback of this update rule is the "*tug-of-war*" (Hadsell et al. 2020) where the gradient from each task pulls the solution towards its optimum. Dif-

ferent update rules, more closely inspired from the biological brains, can be an axis of investigation, as Hebbian learning for continual learning (Thangarasa et al. 2020). Moreover, instead of designing explicitly an update rule, it could be meta-learned. This meta-learning could aim faster remembering of previous forgotten tasks (X. He et al. 2019; M. Caccia et al. 2020).

APPENDIX

A.1 Specific variations of Continual Learning

While Class Incremental Learning (CIL), detailed in Chapter 2, is the most common benchmark in Continual Learning, there are multiple variations of benchmark regardless of which kind of shift is involved.

Multiple labels The main task in Continual Learning is classification of a single class per sample, however it can also be expanded to multiple classes per samples, *e.g.* object detection (Shmelkov et al. 2017) and semantic segmentation (Michieli and Zanuttigh 2019; Cermelli et al. 2020). The latter has seen recently interest from the community for its concrete application: hand labeling in segmentation is extremely costly, and continual segmentation propose to labelize only the new classes in an image, reducing greatly the labeling cost. In that situation, a segmentation maps (made of one label per pixel) will only be partially labelized: new classes are labelized, but old classes are assumed to be background. Moreover, our model may have encountered new classes in the past, when they were themselves considered as background. It's a case of concept shift, where the conditional distribution $p(y|x)$ changes through time. I detail in Chapter 4 the existing benchmark in Continual Semantic Segmentation (CSS) and describe how we tackled this problem.

Online Learning & Task drift detection In Continual Learning, a model learns for multiple epochs for each task. On the other hand, in Online Learning, also called Stream Learning, there are no notions of tasks nor epochs: a model must learn on a stream of samples incoming one by one, and which cannot be replayed by epochs (Aljundi et al. 2019d). The methods to reduce forgetting described in Section 2.4 can still be applied in Online Learning. Modified versions of rehearsal learning, often inspired by reservoir sampling (Knuth 1997), are often used (Hayes et al. 2019; Aljundi et al. 2019b). Multiple regularization methods (Section 2.4.2) needs to do some computation between tasks. For example, weight-based regularization (Section 2.4.2.1) must compute the task-specific importance weights.

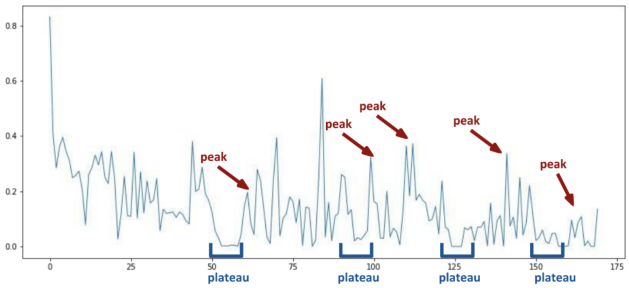


Figure A.1. – **Task-free detection of drift in the input distribution** by recording the plateau in the loss followed by a peak. y-axis is the loss value, and x-axis the update steps. Image from Aljundi et al. (2019b).

Because in Online Learning, there are no clear task separation, a heuristic must determine when doing this computation. Aljundi et al. (2019b), working a stream of images from soap operas, proposed to analyze the loss surface to find drift in the distribution: at some point the model is experienced enough, and the loss starts to plateau. When a drift happens, the loss will usually peak. This is a sign of task-free drift of the distribution as illustrated in Figure A.1.

Continual-Meta Learning Continual Learning aims to not forget. However, we –as humans– often forget, but we can also re-learn what was lost quicker than the first time. The goal of Meta-Continual Learning (MCL) is therefore to recover as quickly as possible –sample wise– the original performance on past tasks (X. He et al. 2019). As the name implies, meta-learning methods, that aims to *learn how to learn*, such as the MAML model (Finn et al. 2017) are used to that end. Then, MCL has been extended to a more general framework where the model also has to adapt quickly to new Out-of-Distribution (OoD) tasks (M. Caccia et al. 2020). Note that Meta-Continual Learning (MCL) is not to be confused with Continual-Meta Learning (CML) where in that case meta-learning is only used during pretraining to provide better model initialization.

Zeroshot Continual Learning In Computer Vision, ZeroShot-Learning (ZSL) (Lampert et al. 2009; Y. Xian et al. 2019) aims to classify classes that were never seen before. To do so, models usually exploit an external knowledge source as a word2vec embedding (Mikolov et al. 2013) trained on Wikipedia or an attribute matrix. Several works have proposed to unify both Continual Learning and ZSL where the future classes that haven't been seen yet must be classified (Lopez-Paz and Ranzato 2017; Wei et al. 2020; Gautam et al. 2020).

Natural Language Processing Continual Learning can be applied to all modalities. After Computer Vision (CV), the most common one is Natural Language Processing (NLP). NLP saw its "*ImageNet moment*" with the advent of Transformers (Vaswani et al. 2017), and more recently with multi-tasks learning (Raffel et al. 2019). Continual NLP Biesialska et al. (2020) aims naturally to learn multiple tasks, but in a consecutive fashion with no –or few– replay of the old tasks data. Applications can be similar to CV with addition of new classes (Masson d'Autume et al. 2019) or new domains (*e.g.* medical corpus, fiction, tweets, etc.) (Gerald and Soulier 2021).

Reinforcement Learning Reinforcement Learning (RL) (Sutton and Barto 1998) more often than not needs support from Continual Learning (Khetarpal et al. 2020): for example as an agent evolves in an environment, it usually needs rehearsal learning (also known as episodic memory) (Mnih et al. 2013). Multiple methods originally developped for continual learning in Computer Vision (CV) can also be applied for RL (Lesort et al. 2020).

A.2 Details on PODNet

We provide here more details on our PODNet model which was presented in Section 3.2.

We also compared our model against baselines with a more flexible memory $M_{\text{total}} = 2000$ (Rebuffi et al. 2017c; Wu et al. 2019), and with various initial task size (by default it is 50 on CIFAR100). In the former case (Table A.1), models benefit from a larger memory per class in the early tasks. In the later case (Table A.2), models initialization is worse because of a smaller initial task size. In these settings very different from Section 3.2.3.1, Pooled Output Distillation Network (PODNet) still outperformed significantly the compared models, proving the robustness of our model.

A.2.1 Implementation details

For all datasets, images are augmented with random crops and flips. For CIFAR100, we additionally change image intensity by a random value in the range [-63, 63]. We train our model for 160 epochs for CIFAR100, and 90 epochs for both ImageNet100 and ImageNet100, with a SGD optimizer with momentum of 0.9. For all datasets, we start with a learning rate of 0.1, a batch size of 128, and cosine annealing scheduling. The weight decay is $5 \cdot 10^{-4}$ for CIFAR100, and $1 \cdot 10^{-4}$ for

	Nb. steps	
Loss	50	10
iCaRL (Rebuffi et al. 2017c)	42.34	56.52
BiC (Wu et al. 2019)	48.44	55.03
UCIR (Nearest Mean Examplar (NME)) (Hou et al. 2019)	54.08	62.89
UCIR (CNN)	55.20	63.62
PODNet (NME)	**62.47**	**64.60**
PODNet (CNN)	**61.87**	**64.68**

Table A.1. – **Evaluation of an easier memory constraint:** ($M_{\text{total}} = 2000$).

	Initial task size				
Loss	10	20	30	40	**50**
iCaRL (Rebuffi et al. 2017c)	40.97	41.28	43.38	44.35	44.20
BiC (Wu et al. 2019)	41.58	40.95	42.27	45.18	47.09
UCIR (NME) (Hou et al. 2019)	42.33	40.81	46.80	46.71	48.57
UCIR (CNN)	43.25	41.69	47.85	47.51	49.30
PODNet (NME)	**45.09**	**49.03**	**55.30**	**57.89**	**61.40**
PODNet (CNN)	**44.95**	**47.68**	**52.88**	**55.42**	**57.98**

Table A.2. – **Varying initial task size:** with $M_{\text{per}} = 20$, and followed by 50 to 90 tasks made of a single class.

ImageNet100 and ImageNet1000. For CIFAR100 we set model hyperparameters $\lambda_c = 3$ and $\lambda_f = 1$, while for ImageNet100 and 1000 we set $\lambda_c = 8$ and $\lambda_f = 10$. Our model uses POD-spatial and POD-flat except when explicitly stated otherwise. Following Hou et al. (2019), we multiply both losses by the adaptive scaling factor: $\lambda = \sqrt{N/T}$ with N being the number of seen classes and T the number of classes in the current task.

For POD-spatial, before sum-pooling we take the features to the power of 2 element-wise. The vector resulting from the pooling is then L2 normalized.

A.2.2 Number of proxies per class

While our model's expressiveness increases with more proxies in $\mathcal{L}_{\text{LSC}}$, it remains fairly stable for values between 5 and 15, thus, for simplicity, we kept it fixed to 10 in all experiments.

In initial experiments, we had the following pairs for the number of clusters (k) and average incremental accuracy (acc): k=1, acc=56.80%; k=2, 57.14%; k=4, acc=57.40%; k=6, acc=57.46%; k=8, acc=57.95%, and k=10, acc=57.98% — i.e., a 1.18 p.p. improvement moving from k=1 to k=10. On ImageNet100, with 10 steps/tasks (increments of give classes per task), moving from k=1 to k=10 improved 1.51 p.p. on acc.

A.2.3 Reproducibility

Code Dependencies The Python version is 3.7.6. We used the PyTorch (Paszke et al. 2017) (version 1.2.0) deep learning framework and the libraries Torchvision (version 0.4.0), NumPy (Oliphant 2006) (version 1.17.2), Pillow (version 6.2.1), and Matplotlib (Hunter 2007) (version 3.1.0). The CUDA version is 10.2. Initial experiments were done with the data loaders library Continuum (Douillard and Lesort 2021). The code is released publicly [1]. We provide all configuration files necessary to reproduce results, including seeds and class ordering.

Datasets description I provide below extensive details on the content of the three datasets considered for PODNet: CIFAR100, ImageNet100, and ImageNet1000.

CIFAR100 contains 32×32-pixel images in 100 classes, with 50k images for training and 10k for testing.

IMAGENET100 contains 224×224-pixel images in 100 classes, with ∼128k images for training and ∼5k for testing.

1. github.com/arthurdouillard/incremental_learning.pytorch

IMAGENET1000 contains 224×224-pixel images in 1000 classes, with ∼1.28m images for training and ∼50k for testing.

Spatial-based distillation I displayed the differences of performance between spatial-based distillation in Table 3.2.3.2 (Table 3.4) when combined with POD-flat. In this appendix, I also detail in Table A.3 the same spatial-loss without POD-flat. The ranking between distillation losses is ostensibly the same. Notice that POD-spatial —and its sub-components POD-width and POD-height– are the only losses barely affected by POD-flat's absence. Note that all alternative losses were tuned on the validation set to get the best performance, including those from external papers. Still, our proposed loss, POD-spatial, outperforms all, both with and without POD-flat.

Loss	NME	CNN
None	41.56	40.76
POD-pixels	42.21	40.81
POD-channels	55.91	50.34
POD-gap	57.25	53.87
POD-width	61.25	57.51
POD-height	61.24	57.50
POD-spatial	**61.42**	**57.64**
GradCam (Dhar et al. 2019)	41.89	42.07
Perceptual Style (Johnson et al. 2016)	41.74	40.80

Table A.3. – **Comparison of distillation losses** based on intermediary features. All losses evaluated without POD-flat. We report the average incremental accuracy on CIFAR100 with 50 steps.

A.3 Details on Ghost

We provide here more details on our Ghost model which was presented in Section 3.3.

A.3.1 Overhead of SVMs training

Training the SVMs for $\mathcal{L}^{\text{svm-reg}}$ introduces a computational overhead. To minimize it, we limit the number of features per class to 500. Moreover, as we advance towards later tasks, fewer unseen classes remain, and thus we have fewer SVMs to train. Overall, an experiment on AwA2, with our setting of 25 classes + 5 × 5

classes, takes 5 hours to train. We observed that our SVM-based regularization extends that time by less than 5 minutes on average, an overhead of less than 2%, which we deemed acceptable. For reference, the SVMs were trained on a machine with 10 CPU cores of 3.90GHz each.

A.3.2 Implementation Details

For all datasets and settings, we set the classification margin $\delta = 0.6$, and the SVM latent-space regularization additional margin $\tau = 1$. We train the feature-extractor-and-classifier pipeline for 90 epochs with an SGD optimizer, learning rate of 0.1, cosine scheduling, and weight decay of 10^{-4}. We train the generator for 1200 epochs, with an Adam optimizer and a learning rate of 10^{-5}. Finally, following Hou et al. (2019) and our work on PODNet (Section 3.2), we fine-tune the classifier for 60 epochs (with the feature extractor frozen and a small learning rate of $10^{-}4$) at the end of every task (except the last one). We found useful to balance the bias towards the seen classes against the unseen classes. With the POD distillation (Section 3.2), we set $\lambda_1 = 3$ for AwA2, and $\lambda_1 = 15$ for aP&Y; with the Less-Forget distillation of Hou et al. (2019), we set $\lambda_1 = 4$ for both datasets. We always set $\lambda_2 = 10^{-3}$, moreover we apply it on L2-normalized features. Finally, we do not reinitialize the models between tasks: f^t results from training f^{t-1} on task t, etc. On the rehearsal memory limitation, we follow the strict setting of Hou et al. (2019), keeping only $s = 20$ training images per past class.

A.3.3 Datasets details

We train our model on three datasets: MNIST, AwA2, and aP&Y. Baselines and our Ghost models are run on the exact same data/class splits, with the exact same preprocessing.

MNIST This dataset has ten classes: handwritten digits ranging from '0" to '9'. It has a training set of 60,000 images and a test set of 10,000 images. We used for validation set, a subset of 10,000 examples of the training set. Images are in black&white (one channel) and of dimension 28×28. We convert the pixels values to the range [0, 1] and then normalize by the mean and standard deviation of the training dataset.

AwA2 This dataset has 50 animals classes. It has a training set of 29,857 images and a test set of 7,465 images. We used for validation set a subset of 8,000 images of the training set. Images are in RGB color. We convert the pixel values to the

range [0, 1] and normalize by the mean and standard deviation of the training dataset. Train images are randomly cropped to a square of 224×224 and are randomly flipped horizontally. Test images are resized to 256×256 and then center cropped to 224×224.

aP&Y This dataset has 32 classes of everyday objects. It has a training set of 12,269 images and a test set of 3,068 images. We used for validation set a subset of 4,000 images of the training set. Images are in RGB color. We convert the pixel values to the range [0, 1] and normalize by the mean and standard deviation of the training dataset. Train images are randomly cropped to a square of 224×224 and are randomly flipped horizontally. Test images are resized to 256×256 and then center cropped to 224×224.

A.3.4 Reproducibility

Code Dependencies The Python version is 3.7.6. We used the PyTorch Paszke et al. (2017) (version 1.2.0) deep learning framework and the libraries Torchvision (version 0.4.0), NumPy Oliphant (2006) (version 1.17.2), Pillow (version 6.2.1), and Matplotlib Hunter (2007) (version 3.1.0). The CUDA version is 10.2. Experiments on MNIST were done with the data loaders library Continuum Douillard and Lesort (2021). The code is released publicly [2].

Hardware & Training duration We ran our experiments on 3 Titan Xp GPUs with 12 Go of VRAM each. Each experiment had access to 10 CPU cores of 3.90 GHz each, and used at most 3 Go of RAM and 8 Go of VRAM. A single experiment run on AwA2 took on average 5 hours and, on aP&Y, 3 hours. We ran each experiment thrice with different random seeds (1, 2, and 3).

A.4 Details on PLOP

We provide here more details on our work done in Chapter 4.

A.4.1 Algorithm view of Local POD

In Algorithm A.1, we summarize the algorithm for the proposed Local POD. The algorithm consists in three functions. At first, the function `Distillation`,

2. github.com/arthurdouillard/incremental_learning.pytorch

loops over all L layers onto which we apply Local POD. Second, `LocalPOD`, computes the L2 distance (L.26) between POD embeddings of the current (L.19) and old (L.20) models. It loops over S different scales (L.14) and Φ computes the POD embedding given two features maps subsets (L.19-20). $\|=$ denotes an in-place concatenation.

Algorithm A.1 Local POD algorithm

1: **function** DISTILLATION(f^t, f^{t-1}, x, S)
2: $loss \leftarrow 0$
3: **for** $l \leftarrow 0;\ l < L;\ l{+}{+}$ **do**
4: $\mathbf{h}_l^t \leftarrow f_l^t(\mathbf{x})$
5: $\mathbf{h}_l^{t-1} \leftarrow f_l^{t-1}(\mathbf{x})$
6: $loss \leftarrow loss + \text{LocalPOD}(\mathbf{h}_l^t, \mathbf{h}_l^{t-1}, S)$
7: **end for**
8: **return** $\frac{loss}{L}$
9: **end function**
10:
11: **function** LOCALPOD($\mathbf{h}^t$, $\mathbf{h}^{t-1}$, S)
12: $\mathbf{P}^t \leftarrow [\,]$
13: $\mathbf{P}^{t-1} \leftarrow [\,]$
14: **for** $s \leftarrow 0;\ s < S;\ s{+}{+}$ **do**
15: $w \leftarrow {}^{W}/_{2^s}$
16: $h \leftarrow {}^{H}/_{2^s}$
17: **for** $i \leftarrow 0;\ i < W - w;\ i{+}{=}w$ **do**
18: **for** $j \leftarrow 0;\ j < H - h;\ j{+}{=}h$ **do**
19: $\mathbf{p}^t \leftarrow \Phi(\mathbf{h}^t$`[i:i+w, j:j+h]`$)$
20: $\mathbf{p}^{t-1} \leftarrow \Phi(\mathbf{h}^{t-1}$`[i:i+w, j:j+h]`$)$
21: $\mathbf{P}^t \| = \mathbf{p}^t$
22: $\mathbf{P}^{t-1} \| = \mathbf{p}^{t-1}$
23: **end for**
24: **end for**
25: **end for**
26: **return** $\|\mathbf{P}^t - \mathbf{P}^{t-1}\|^2$
27: **end function**

A.4.2 Reproducibility

Datasets: We evaluate our model on three datasets Pascal-VOC (Everingham et al. 2015), ADE20k (B. Zhou et al. 2017), and Cityscapes (Cordts et al. 2016). VOC contains 20 classes, 10,582 training images, and 1,449 testing images. ADE20k has 150 classes, 20,210 training images, and 2,000 testing images. Cityscapes contains

2975 and 500 images for train and test, respectively. Those images represent 19 classes and were taken from 21 different cities. All ablations and hyperparameters tuning were done on a validation subset of the training set made of 20% of the images. For Cityscapes and ADE20k, we resize the images to 512×512, with a center crop. An additional random horizontal flip augmentation is applied at training time. Cityscapes images are resized to 512×1024.

Implementation Details: As in Cermelli et al. (2020), we use a Deeplab-V3 (L.-C. Chen et al. 2017) architecture with a ResNet-101 (K. He et al. 2016) backbone pretrained on ImageNet (Deng et al. 2009) for all experiments and models but SDR (Michieli and Zanuttigh 2021) which used a Deeplab-V3+ (L.-C. Chen et al. 2018b). For all datasets, we set a maximum threshold for the uncertainty measure of Equation 4.7 to $\tau = 1e-3$. We train our model for 30, 60, and 30 epochs per CSS step on Pascal-VOC, ADE20k, and Cityscapes, respectively, with an initial learning rate of $1e-2$ for the first CSS step, and $1e-3$ for all the following ones. Note that for Cityscapes, the first step is longer with 50 epochs. We reduce the learning rate exponentially with a decay rate of $9e-1$. We use SGD optimizer with $9e-1$ Nesterov momentum. The Local POD weighting hyperparameter λ is set to $1e-2$ and $5e-4$ for intermediate feature maps and logits, respectively. Moreover, we multiply this factor by the adaptive weighting $\sqrt{|C^{1:t}|/|C^t|}$ introduced by Hou et al. (2019) that increases the strength of the distillation the further we are into the continual process. For all feature maps, Local POD is applied before ReLU, with squared pixel values, as in Zagoruyko and Komodakis (2016) or PODNet (Section 3.2). We use 3 scales for Local POD: 1, $1/2$, and $1/4$, as adding more scales experimentally brought diminishing returns. For PLOPLong only, we L2-normalize all POD embeddings before distilling them as also done by Section 3.2. Furthermore, for PLOPLong, the gradient norm-clipping is set at 1.0 for Pascal-VOC and 2.0 for Cityscapes. We use a batch size of 24 distributed on two 12Go Titan Xp GPUs. Contrary to many continual models, we do not have access to any task id in inference, therefore our setting/strategy has to predict a class among the set of all seen classes

Classes ordering details: For all quantitative experiments on Pascal-VOC 2012 and ADE20k, the same class ordering was used across all evaluated models. For Pascal-VOC 2012 it corresponds to [1, 2, ..., 20] and ADE20k to [1, 2, ..., 150] as defined by Cermelli et al. (2020). For class-incremental Cityscapes to [1, 2,...,19] . In this case, because Cityscapes does not have `background` class, it only appears as "unlabeled" for past and future classes. For continual-domain cityscapes, the order of the domains/cities is the following: `aachen`, `bremen`, `darmstadt`, `erfurt`, `hanover`, `krefeld`, `strasbourg`, `tubingen`, `weimar`, `bochum`, `cologne`, `dusseldorf`, `hamburg`, `jena`, `monchengladbach`, `stuttgart`, `ulm`, `zurich`, `frankfurt`, `lindau`, and `munster`.

In Figure 4.5, we showcased a boxplot featuring 20 different class orders for Pascal-VOC 2012 15-1. For the sake of reproducibility, we provide details on these orders:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
[12, 9, 20, 7, 15, 8, 14, 16, 5, 19, 4, 1, 13, 2, 11, 17, 3, 6, 18, 5]
[9, 12, 13, 18, 2, 11, 15, 17, 10, 8, 4, 5, 20, 16, 6, 14, 19, 1, 7, 3]
[13, 19, 15, 17, 9, 8, 5, 20, 4, 3, 10, 11, 18, 16, 7, 12, 14, 6, 1, 2]
[15, 3, 2, 12, 14, 18, 20, 16, 11, 1, 19, 8, 10, 7, 17, 6, 5, 13, 9, 4]
[7, 13, 5, 11, 9, 2, 15, 12, 14, 3, 20, 1, 16, 4, 18, 8, 6, 10, 19, 17]
[12, 9, 19, 6, 4, 10, 5, 18, 14, 15, 16, 3, 8, 7, 11, 13, 2, 20, 17, 1]
[13, 10, 15, 8, 7, 19, 4, 3, 16, 12, 14, 11, 5, 20, 6, 2, 18, 9, 17, 1]
[3, 14, 13, 1, 2, 11, 15, 17, 7, 8, 4, 5, 9, 16, 19, 12, 6, 18, 10, 20]
[1, 14, 9, 5, 2, 15, 8, 20, 6, 16, 18, 7, 11, 10, 19, 3, 4, 17, 12, 13]
[16, 13, 1, 11, 12, 18, 6, 14, 5, 3, 7, 9, 20, 19, 15, 4, 2, 10, 8, 17]
[10, 7, 6, 19, 16, 8, 17, 1, 14, 4, 9, 3, 15, 11, 12, 2, 18, 20, 13, 5]
[7, 5, 3, 9, 13, 12, 14, 19, 10, 2, 1, 4, 16, 8, 17, 15, 18, 6, 11, 20]
[18, 4, 14, 17, 12, 10, 7, 3, 9, 1, 8, 15, 6, 13, 2, 5, 11, 20, 16, 19]
[5, 4, 13, 18, 14, 10, 19, 15, 7, 9, 3, 2, 8, 16, 20, 1, 12, 11, 6, 17]
[9, 12, 13, 18, 7, 1, 15, 17, 10, 8, 4, 5, 20, 16, 6, 14, 19, 11, 2, 3]
[3, 14, 13, 18, 2, 11, 15, 17, 10, 8, 4, 5, 20, 16, 6, 12, 19, 1, 7, 9]
[7, 5, 9, 1, 15, 18, 14, 3, 20, 10, 4, 19, 11, 17, 16, 12, 8, 6, 2, 13]
[3, 14, 6, 1, 2, 11, 12, 17, 7, 20, 4, 5, 9, 16, 19, 15, 13, 18, 10, 8]
[1, 2, 12, 14, 6, 19, 18, 17, 5, 20, 8, 4, 9, 16, 10, 3, 15, 13, 11, 7]
```

In the 15-1 setting, we first learn the first fifteen classes, then increment the five remaining classes one by one. Note that the special class `background` (0) is always learned during the first task.

Hardware and Code: For each experiment, we used two Titan Xp GPUs with 12 Go of VRAM each. The initial step $t = 1$ for each setting is common to all models, therefore we re-use the weights trained on this step. All models took less than 2 hours to train on Pascal-VOC 2012 15-1, and less than 16 hours on ADE20k 100-10. We distributed the batch size equally on both GPUs. All models are implemented in PyTorch (Paszke et al. 2017) and run with half-precision for efficiency reasons with Nvdia's APEX library using O1 optimization level. Our code base is based on Cermelli et al. (2020)'s code based that we modified to implement our strategy. Our code is released publicly [3].

A.4.3 Additional Experiments

Model ablation: Table A.4 shows the construction of our model component by component on Pascal-VOC 2012 in 15-5 and 15-1. For this experiment, we train our model on 80% of the training set and evaluate on the validation set made of

3. github.com/arthurdouillard/CVPR2021_PLOP

Model	**15-5** (2 tasks)		**15-1** (6 tasks)	
	all	*avg*	*all*	*avg*
CE	13.85	46.91	3.99	19.37
Pseudo	66.19	73.07	19.74	44.48
Pseudo + Local POD	70.29	75.13	50.41	64.95
νPseudo + Local POD	**71.43**	**75.70**	**52.31**	**65.71**

Table A.4. – Ablations of PLOP on the Pascal-VOC 2012 dataset in 15-5 and 15-1. Scores are measured on a validation subset made of 20% of the training set.

the remaining 20%. We report the mIoU at the final task ("*all*") and the average of the mIoU after each task ("*avg*"). We start with a crude baseline made of solely cross-entropy (CE). Pseudo-labeling by itself increases by a large margin performance (eg. 3.99 to 19.74 for 15-1). Applying Local POD reduces drastically the forgetting leading to a massive gain of performance (eg. 19.74 to 50.41 for 15-1). Finally, our adaptive factor ν based on the ratio of accepted pseudo-labels over the number of background pixels further increases our overall results (eg. 50.41 to 52.31 for 15-1). The interest of ν arises when PLOP faces hard images where few pseudo-labels will be created due to an overall high uncertainty. In such a case, current classes will be over-represented, which can in turn lead to strong bias towards new classes (*i.e.* the model will have a tendency to predict one of the new classes for every pixel). The ν factor therefore decreases the overall classification loss on such images, and empirical results confirm its effectiveness.

A.5 Details on DyTox

We provide here more details on our PODNet model which was presented in Chapter 5.

A.5.1 Experimental details

Datasets We use three datasets: CIFAR100 (Krizhevsky and Geoffrey Hinton 2009), ImageNet100, and ImageNet1000 (Deng et al. 2009). CIFAR100 is made of 50,000 train RGB images and 10,000 test RGB images of size 32×32 for 100 classes. ImageNet1000 contains 1.2 million RGB train images and 50,000 validation RGB images of size 224×224 for 1000 classes. ImageNet100 is a subset of 100 classes from ImageNet1000. We follow PODNet (Section 3.2) and DER (Yan et al. 2021)

and use the same 100 classes they've used. Note that while we considered PODNet in a setting where the initial step contained half of all the classes, while for DyTox we consider that each step brings an equal amount of classes.

Implementation For all datasets, we train the model for 500 epochs per task with Adam (Kingma and Ba 2014) with a learning rate of $5e^{-4}$, including 5 epochs of warmup. Following UCIR (Hou et al. 2019), PODNet (Section 3.2), and DER (Yan et al. 2021), at the end of each task (except the first) we finetuned our model for 20 epochs with a learning rate of $5e^{-5}$ on a balanced dataset. In DyTox, we applied the standard data augmentation of DeiT (Touvron et al. 2021a) but we removed the pixel erasing (Zhong et al. 2017), MixUp (Hongyi Zhang et al. 2018), and CutMix (Yun et al. 2019) augmentations for fair comparison. In contrast, in DyTox+ we used a MixUp (Hongyi Zhang et al. 2018) with beta distribution $\beta(0.8, 0.8)$. During all incremental tasks ($t > 1$), the old classifiers Clf_i, $i < t$ and the old task tokens θ_i, $i < t$ parameters are frozen. During the finetuning phase where classes are rebalanced (Castro et al. 2018; Hou et al. 2019; Yan et al. 2021), these parameters are optimized, but the SABs are frozen. Our code is released publicly [4].

Hyperparameter tuning In contrast with previous works (Yan et al. 2021), we wanted stable hyperparameters, tuned for a single setting and then applied on all experiments. This avoids optimizing for the number of tasks, which defeats the purpose of continual learning (Farquhar and Gal 2018). We tuned hyperparameters for DyTox using a validation subset made of 10% of the training dataset, and this only on the CIFAR100 experiment with 10 steps. We provide in Table A.5 the chosen hyperparameters. Results presented in Chapter 5 shows that those hyperparameters reach state-of-the-art on all other settings and notably on ImageNet.

Baselines E2E (Castro et al. 2018) and Simple-DER (Zhuoyun Li et al. 2021) results come from their respective papers. All other baseline results are taken from the DER paper (Yan et al. 2021). We now further describe their contributions. iCaRL (Rebuffi et al. 2017c) uses a knowledge distillation loss (Geoffrey Hinton et al. 2015) and at test-time predicts using a k-NN from its features space. E2E (Castro et al. 2018) learns a model with knowledge distillation and applies a finetuning after each step. UCIR (Hou et al. 2019) uses cosine classifier and euclidean distance between the final flattened features as a distillation loss. BiC (Wu et al. 2019) uses a knowledge distillation loss and also re-calibrates (Guo et al. 2017) the logits of the new classes using a simple linear model trained on validation data. WA (B. Zhao et al. 2020) uses a knowledge distillation loss and re-weights at each

4. github.com/arthurdouillard/dytox

Hyperparameter	Range	Chosen value
Learning rate	$1e^{-3}$, $5e^{-4}$, $1e^{-4}$	$5e^{-4}$
Epochs	300, 500, 700	500
λ	0.05, 0.1, 0.5	0.1
CIFAR's patch size	4, 8, 16	4
ImageNet's patch size	14, 16	16

Table A.5. – **Hyperparameters** that were tuned from the codebase of Touvron et al. (2021a). We ran a gridsearch on CIFAR100 10 steps on a validation set made of 10% of the training set, and kept fixed the chosen hyperparameters for all experiments (any number of steps and any datasets).

epoch the classifier weights associated to new classes so that they have the same average norm as the classifier weights of the old classes. PODNet (Section 3.2) uses a cosine classifier and a specific distillation loss (POD) applied at multiple intermediary features of the ResNet backbone. RPSNet (Rajasegaran et al. 2019) uses knowledge distillation and manipulates subnetworks in its architecture, following the lottery ticket hypothesis (Frankle and Carbin 2019). DER (Yan et al. 2021) creates a new ResNet per task. All ResNets' embeddings are concatenated and fed to a unique classifier. ResNets are pruned using HAT (Serrà et al. 2018) masking procedure. Note that DER pruning has multiple hyperparameters that are set differently according to the settings. Furthermore, the reported parameters count, after pruning, in (Yan et al. 2021) is an average of the count over all steps: the final parameters count (necessarily higher) wasn't available. Finally, Simple-DER (Zhuoyun Li et al. 2021) is similar to DER, with a simpler pruning method which doesn't require any hyperparameter tuning.

A.5.2 Parameter sharing of the TAB

Previous dynamic methods as DER (Yan et al. 2021) and Simple-DER (Zhuoyun Li et al. 2021) shared no parameters between tasks until the final classifier. We proposed instead with DyTox to share the encoder (SABs) and the decoder (TAB) parameters across tasks, leading to a minimal memory overhead while also sharing common information between tasks. In Table A.6, we compare the impact of sharing the TAB per task — and only maintain different tokens per task. In the first row, a different TAB is created per task, while in the second row the same TAB is used — which is the DyTox strategy. A different TAB per task leads to better results (56% *v.s.* 52% in "Last" accuracy) because the network can be more diverse with each TAB specialized to its associated task. This increased diversity

TAB parameter sharing?	#P	Avg	Last
✗	97.59	72.20	56.00
✓	10.77	70.20	52.34

Table A.6. – **Investigation of the parameter sharing of TAB**. We report the "Avg" accuracy and the "Last" accuracy for the 50 steps setting on CIFAR100. The second row corresponds to DyTox.

	Joint (1 steps)	50 steps	
Patch size	**Last** (↑)	**Last** (↑)	**Forgetting** (↓)
4	76.12	52.34	33.15
8	67.65	43.93	35.44
16	50.15	31.49	33.20

Table A.7. – **Patch size effect on continual** for the joint (1 step, no continual) and 50 steps settings on CIFAR100. We choose a patch size of 4 for our main experiments: yet, it has only few impact on forgetting.

has a drawback: the memory overhead is too important (97M *v.s.* 10M parameters). We find in practice that DyTox strikes a good balance between memory overhead and continual performance.

A.5.3 Patch size effect on forgetting

A key component of the transformer architecture is the patch tokenizer. The number of patch tokens in an image is determined by the patch size: a larger patch size means less tokens, and vice-versa. We wondered about the effect of the patch size on forgetting and tested three different kind of patch sizes in Table A.7. Echoing results in vision transformers (Dosovitskiy et al. 2021; Touvron et al. 2021a), a smaller patch size (4 vs. 8 and 16) performs best in a joint training. However, the forgetting defined by Chaudhry et al. (2018) is relatively similar, with 33.15% for a patch of size of 4, and 33.20% for a patch size of 16. Therefore, we argue that the transformer architecture is hardly sensitive to the patch resolution w.r.t its forgetting in continual learning.

A.5.4 ResNet backbone

DyTox is made of two main components: the SABs and the unique TAB. The TAB structure, taking in input both patch tokens and a task token, is crucial to our strategy. Yet, the SAB could be of any kind of features extractor, based on

Encoder	**#P**	**Avg**	**Last**
ResNet	10.68	68.53	50.05
SABs	10.77	70.20	52.34

Table A.8. – **Hybrid network** on CIFAR100 50 steps. While the features extractor is made of SABs in DyTox, here we instead use a modified ResNet18. Our framework still works well with a convolution-based approach.

	CIFAR100 Top-1		ImageNet100 Top-5	
Task decoder	Avg	Last	Avg	Last
Residual Adapters	70.00	52.38	91.25	85.00
FiLM	69.42	54.05	89.49	81.40
TAB *(ours)*	70.20	52.34	92.04	87.98

Table A.9. – **Alternative task conditioner** on CIFAR100 50 steps and ImageNet100 10 steps. While the simpler Residual Adapters (Rebuffi et al. 2017b) and FiLM (Perez et al. 2018) perform similarly to our TAB on CIFAR100, they forget significantly more on the complex ImageNet100.

convolutions or attentions. Following the hybrid network proposed in ablations by Dosovitskiy et al. (2021), we tried to replace the collection of SABs by a ResNet18. The final features of the ResNet, before global pooling, of shape $(W \times H \times D)$ can be seen as $W \times H$ tokens of D dimension. We made a few modifications to this ResNet to boost its performance, namely removed the fourth and ultimate layer, and added a pointwise convolution with 504 output channels (so it can be divided by the 12 attention heads of the TAB), a batch normalization (Ioffe and Szegedy 2015), and a ReLU activation. These simple modifications are sufficient for our proof of concept, and thus we also didn't tune deeply this model. We display in Table A.8 the comparison of the two backbones on CIFAR100 50 steps: (1) with ResNet, and (2) with SABs (DyTox). Performances are slightly lower than DyTox with SABs, however, they are still significantly higher than previous state-of-the-art like WA (B. Zhao et al. 2020), especially in "Last" accuracy. Moreover, the parameters count is comparable to DyTox. This experiment shows that our DyTox framework, while designed with a transformer backbone in mind, is also efficient on non-token-based architectures such as a ResNet.

A.5.5 Alternative task decoders

We investigate here other approaches for *conditioning* features to the different tasks. Residual Adapters (Rebuffi et al. 2017b) adds a different residual branch

made of a pointwise convolution for each domain the model is learned (*e.g.* CIFAR then ImageNet then SVHN). This model needs the task/dataset/domain identifier at test-time to determine which residual to use. For VQA task (Antol et al. 2015), FiLM (Perez et al. 2018) proposes to modify the visual features using the the textual query.

We adapt these two feature conditioning strategies for our transformer backbone architecture. We perform a global token pooling after the ultimate SAB, and apply for each learned task, a residual adapter or a FiLM. Residual adapter in our case is a MLP, and FiLM parameters are directly learned. As for DyTox, we forward each task-specific embedding to the respective task-specific classifier. We showcase the continual performance in Table A.9 on CIFAR100 50 steps and ImageNet100 10 steps. On the former dataset, smaller and easier, the residual adapters and FiLM have similar performance as our TAB approach. On the other hand, as soon as the task complexity increases with the more detailed ImageNet100 dataset, FiLM and Residual adapter based conditioning strategies forget significantly more than our complete DyTox framework: TAB outperform the Residual Adapters by +2.98 *p.p.* in "Last" top-5 accuracy and FiLM by +6.58 *p.p.*.

BIBLIOGRAPHY

Advancing Translational Sciences, National Center for (2014). *Tox21 Data Challenge*. https://tripod.nih.gov/tox21/challenge/index.jsp (cit. on p. 1).

Aljundi, Rahaf, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars (2018). "Memory Aware Synapses: Learning what (not) to forget". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 17).

Aljundi, Rahaf, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, and Laurent Charlin (2019a). "Online Continual Learning with Maximally Interfered Retrieval". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 15).

Aljundi, Rahaf, Punarjay Chakravarty, and Tinne Tuytelaars (2017). "Expert gate: Lifelong learning with a network of experts". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 21).

Aljundi, Rahaf, Klaas Kelchtermans, and Tinne Tuytelaars (2019b). "Task-free continual learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 109, 110).

Aljundi, Rahaf, Min Lin, Baptiste Goujaud, and Yoshua Bengio (2019c). "Gradient based sample selection for online continual learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 18).

Aljundi, Rahaf, Min Lin, Baptiste Goujaud, and Yoshua Bengio (2019d). "Online continual learning with no task boundaries". In: *arXiv preprint library* (cit. on p. 109).

Aljundi, Rahaf and Tinne Tuytelaars (2019). "Selfless Sequential Learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 22).

Allen, Kelsey, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum (2019). "Infinite Mixture Prototypes for Few-shot Learning". In: *International Conference on Machine Learning (ICML)* (cit. on p. 38).

Amodei, Dario, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu (2016). "Deep Speech 2: End-to-

End Speech Recognition in English and Mandarin". In: *International Conference on Machine Learning (ICML)* (cit. on p. 1).

Anglekar, Sumegh, Urvee Chaudhari, Atul Chitanvis, and Radha Shankarmani (2021). "Machine Learning Based Risk Assessment Analysis for SMEs Loan Grant". In: *International Conference on Communication information and Computing Technology (ICCICT)* (cit. on p. 2).

Anonymous, Anonymous (2021). "Sharpness-Aware Minimization in Large-Batch Training: Training Vision Transformer in Minutes". In: *Under review at the International Conference on Learning Representations (ICLR)* (cit. on p. 101).

Antol, Stanislaw, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh (2015). "VQA: Visual Question Answering". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 7, 125).

Arjovsky, Martin, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz (2019). "Invariant Risk Minimization". In: *arXiv preprint library* (cit. on p. 107).

Ayub, Ali and Alan R. Wagner (2021). "EEC: Learning to Encode and Regenerate Images for Continual Learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 16).

Ba, Jimmy, Jamie Ryan Kiros, and Geoffrey Hinton (2016). "Layer Normalization". In: *Advances in NeurIPS 2016 Deep Learning Symposium* (cit. on pp. 9, 93).

Badrinarayanan, V., A. Kendall, and R. Cipolla (2017). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 59).

Belouadah, Eden and Adrian Popescu (2019). "IL2M: Class Incremental Learning With Dual Memory". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 21, 66, 93).

Ben-younes, Hedi, Rémi Cadene, Matthieu Cord, and Nicolas Thome (2017). "MUTAN: Multimodal Tucker Fusion for Visual Question Answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 7).

Biesialska, Magdalena, Katarzyna Biesialska, and Marta R. Costa-jussà (2020). "Continual Lifelong Learning in Natural Language Processing: A Survey". In: *International Conference on Computational Linguistics (COLING)* (cit. on p. 111).

Bishop, Christopher (2006). *Pattern Recognition and Machine Learning*. Springer (cit. on p. 1).

Breen, Philip G, Christopher N Foley, Tjarda Boekholt, and Simon Portegies Zwart (2020). "Newton versus the machine: solving the chaotic three-body problem using deep neural networks". In: *Monthly Notices of the Royal Astronomical Society* (cit. on p. 1).

Bucher, Maxime, Stephane Herbin, and Frederic Jurie (2017). "Generating Visual Representations for Zero-Shot Classification". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshop* (cit. on pp. 41, 49).

Caccia, Lucas, Jing Xu, Myle Ott, Marc'Aurelio Ranzato, and Ludovic Denoyer (2022). "On Anytime Learning at Macroscale". In: *arXiv preprint library* (cit. on p. 107).

Caccia, Massimo, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, and Laurent Charlin (2020). "Online Fast Adaptation and Knowledge Accumulation: a New Approach to Continual Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 108, 110).

Caesar, Holger, Jasper R. R. Uijlings, and Vittorio Ferrari (2018). "COCO-Stuff: Thing and Stuff Classes in Context". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 59).

Castro, Francisco M., Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari (2018). "End-to-End Incremental Learning". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 15, 69, 76, 97, 98, 102, 121).

Cermelli, Fabio, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo (2020). "Modeling the Background for Incremental Learning in Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 22, 56, 57, 59, 60, 72–79, 83, 109, 118, 119).

Cermelli, Fabio, Massimiliano Mancini, Yongqin Xian, Zeynep Akata, and Barbara Caputo (2021). "Prototype-based Incremental Few-Shot Segmentation". In: *Proceedings of the British Machine Vision Conference (BMVC)* (cit. on pp. 60, 66, 67, 73, 74).

Cha, Junbum, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park (2021). "SWAD: Domain Generalization by Seeking Flat Minima". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 107).

Chao, Wei-Lun, Soravit Changpinyo, Boqing Gong, and Fei Sha (2016). "An empirical study and analysis of generalized zero- shot learning for object recognition in the wild". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 48).

Chapelle, Olivier, Léon Bottou, and Vladimir Vapnik (2001). "Vicinal risk minimization". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 95).

Chaudhry, Arslan, Puneet Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr (2018). "Riemannian Walk for Incremental Learning: Understanding Forget-

ting and Intransigence". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 13, 14, 17, 73, 74, 101, 102, 123).

Chaudhry, Arslan, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny (2019a). "Efficient Lifelong Learning with A-GEM". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 18).

Chaudhry, Arslan, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H.S. Torr, and Marc'Aurelio Ranzato (2019b). "On Tiny Episodic Memories in Continual Learning". In: *International Conference on Machine Learning (ICML) Workshop* (cit. on p. 15).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille (2018a). "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on pp. 7, 59).

Chen, Liang-Chieh, George Papandreou, Florian Schroff, and Hartwig Adam (2017). "Rethinking Atrous Convolution for Semantic Image Segmentation". In: *arXiv preprint library* (cit. on p. 118).

Chen, Liang-Chieh, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam (2018). "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 56, 59).

Chen, Liang-Chieh, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam (2018b). "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 118).

Chen, Yifu, Arnaud Dapogny, and Matthieu Cord (2020). "SEMEDA: Enhancing Segmentation Precision with Semantic Edge Aware Loss". In: *Pattern Recognition* (cit. on pp. 70, 83).

Collier, Mark, Effrosyni Kokiopoulou, Andrea Gesmundo, and Jesse Berent (2020). "Routing Networks with Co-training for Continual Learning". In: *International Conference on Machine Learning (ICML) Workshop* (cit. on p. 21).

Cordts, M., M. Omran, S. Ramos, T. Reheld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 59, 69, 71, 72, 117).

Cortes, Corinna and Vladimir Vapnik (2015). "Support-vector Networks". In: *Machine Learning* (cit. on p. 45).

Couairon, Guillaume, Asya Grechka, Jakob Verbeek, Holger Schwenk, and Matthieu Cord (2022). "FlexIT: Towards Flexible Semantic Image Translation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 7).

d'Ascoli, Stéphane, Hugo Touvron, Matthew Leavitt, Giulio Morcos Ari annd Biroli, and Levent Sagun (2021). "ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases". In: *arXiv preprint library* (cit. on pp. 10, 96).

Dabouei, Ali, Sobhan Soleymani, Fariborz Taherkhani, and Nasser M. Nasrabadi (2020). "SuperMix: Supervising the Mixing Data Augmentation". In: *arXiv preprint library* (cit. on p. 69).

Davidson, Guy and Michael C. Mozer (2020). "Sequential mastery of multiple visual tasks: Networks naturally learn to learn and forget to forget". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 12).

Degrave, Jonas, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller (2022). "Magnetic control of tokamak plasmas through deep reinforcement learning". In: *Nature* 602.7897, pp. 414–419. URL: https://doi.org/10.1038/s41586-021-04301-9 (cit. on p. 1).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 19, 34, 95, 118, 120).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)* (cit. on pp. 8, 94).

Dhar, Prithviraj, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa (2019). "Learning Without Memorizing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 19, 22, 27, 38, 39, 114).

Díaz-Rodríguez, Natalia, Vincenzo Lomonaco, David Filliat, and Davide Maltoni (2018). "Don't forget, there is more than forgetting: new metrics for Continual Learning". In: *Advances in Neural Information Processing Systems (NeurIPS) Workshop* (cit. on p. 14).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *Pro-*

ceedings of the International Conference on Learning Representations (ICLR) (cit. on pp. 8, 10, 89, 91, 94, 96, 123, 124).

Douillard, Arthur, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021a). "PLOP: Learning without Forgetting for Continual Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 3, 55).

Douillard, Arthur, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021b). "Tackling Catastrophic Forgetting and Background Shift in Continual Semantic Segmentation". In: *Under review at TPAMI* (cit. on pp. 3, 55).

Douillard, Arthur, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle (2020). "PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 3, 25).

Douillard, Arthur and Timothée Lesort (2021). "Continuum: Simple Management of Complex Continual Learning Scenarios". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop* (cit. on pp. 96, 113, 116).

Douillard, Arthur, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord (2022). "DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 4, 87).

Douillard, Arthur, Eduardo Valle, Charles Ollion, and Matthieu Cord (2021c). "Insights from the Future for Continual Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop* (cit. on pp. 3, 25).

Evans, Richard, Jim Gao, and Deepmind (2016). *DeepMind AI Reduces Google Data Centre Cooling Bill by 40%*. https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40 (cit. on p. 1).

Everingham, Mark, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman (2015). "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision (IJCV)* (cit. on pp. 59, 69, 71, 72, 117).

Fang, Hao-Shu, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu (2019). "InstaBoost: Boosting Instance Segmentation via Probability Map Guided Copy-Pasting". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 69, 70, 84).

Farajtabar, Mehrdad, Navid Azizan, Alex Mott, and Ang Li (2020). "Orthogonal Gradient Descent for Continual Learning". In: *Internaltional Conference on Artificial Intelligence and Statistics (AISTATS)* (cit. on pp. 18, 27).

Farhadi, A., I. Endres, D. Hoiem, and D.A. Forsyth (2009). "Describing Objects by their Attributes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 47).

Farquhar, Sebastian and Yarin Gal (2018). "Towards Robust Evaluations of Continual Learning". In: *International Conference on Machine Learning (ICML) Workshop* (cit. on p. 121).

Fellbaum, Christiane (1998). *WordNet: An Electronic Lexical Database*. Bradford Books (cit. on p. 41).

Fernando, Chrisantha, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra (2017). "PathNet: Evolution Channels Gradient Descent in Super Neural Networks". In: *arXiv preprint library* (cit. on p. 20).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *International Conference on Machine Learning (ICML)* (cit. on p. 110).

Foret, Pierre, Ariel Kleiner, Hossein Mobahi, and Benhnam Neyshabur (2021). "Sharpness-Aware Minimization for Efficiently Improving Generalization". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 95, 98, 107).

Frankle, Jonathan and Michael Carbin (2019). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 19, 122).

French, Robert (1999). "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* (cit. on pp. 2, 36).

Fu, Jun, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu (2019). "Dual Attention Network for Scene Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 59).

Fukushima, Kunihiko (1980). "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* (cit. on p. 7).

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *International Conference on Machine Learning (ICML)* (cit. on p. 8).

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani (2017). "Deep Bayesian Active Learning with Image Data". In: *International Conference on Machine Learning (ICML)* (cit. on p. 15).

Gallardo, Jhair, Tyler L Hayes, and Christopher Kanan (2021). "Self-Supervised Training Enhances Online Continual Learning". In: *Proceedings of the British Machine Vision Conference (BMVC)* (cit. on p. 107).

Gautam, Chandan, Sethupathy Parameswaran, Ashish Mishra, and Suresh Sundaram (2020). "Generalized Continual Zero-Shot Learning". In: *arXiv preprint library* (cit. on p. 110).

Gebru, Timnit (2019). "Oxford handbook on AI ethics book chapter on race and gender". In: *arXiv preprint library* (cit. on p. 2).

Gerald, Thomas and Laure Soulier (2021). "Continual Learning of Long Topic Sequences in Neural Information Retrieval". In: *European Conference on Information Retrieval (ECIR)* (cit. on p. 111).

Ghiasi, Golnaz, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph (2020). "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation". In: *arXiv preprint library* (cit. on pp. 69, 70).

Gidaris, Spyros and Nikos Komodakis (2018). "Dynamic Few-Shot Visual Learning without Forgetting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 28).

Goldberger, Jacob, Geoffrey E Hinton, Sam T. Roweis, and Ruslan R Salakhutdinov (2005). "Neighbourhood Components Analysis". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 33).

Golkar, Siavash, Michael Kagan, and Kyunghyun Cho (2019). "Continual Learning via Neural Pruning". In: *Advances in Neural Information Processing Systems (NeurIPS) Workshop* (cit. on p. 20).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 7).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press (cit. on p. 1).

Grechka, Asya, jean Francois Goudou, and Matthieu Cord (2021). "MAGECally invert images for realistic editing". In: *Proceedings of the British Machine Vision Conference (BMVC)* (cit. on p. 7).

Gretton, A., K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola (2007). "A kernel method for the two-sample-problem". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 43, 49).

Gretton, A., K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola (2012). "A kernel two-sample test". In: *Journal of Machine Learning Research* (cit. on pp. 43, 49).

Grother, Patrick, Mei Ngan, and Kayee Hanaoka (2019). *Face Recognition Vendor Test: Part 3: Demographic Effects*. https://doi.org/10.6028/NIST.IR.8280 (cit. on p. 2).

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger (2017). "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 21, 95, 121).

Hadsell, Raia, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu (2020). "Embracing change: Continual learning in deep neural networks". In: *Trends in cognitive sciences* (cit. on p. 107).

Han, Kai, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman (2020). "Automatically Discovering and Learning New Visual Categories with Ranking Statistics". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 22).

Hayes, Tyler L., Nathan D. Cahill, and Christopher Kanan (2019). "Memory Efficient Experience Replay for Streaming Learning". In: *IEEE Internaltional Conference on Robotics and Automation (ICRA)* (cit. on p. 109).

Hayes, Tyler L., Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan (2020). "REMIND Your Neural Network to Prevent Catastrophic Forgetting". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 16).

He, Kaiming, X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 7, 30, 48, 62, 118).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 63).

He, Xu, Jakub Sygnowski, Alexandre Galashov, Andrei A. Rusu, Yee Whye Teh, and Razvan Pascanu (2019). "Task Agnostic Continual Learning via Meta Learning". In: *arXiv preprint library* (cit. on pp. 108, 110).

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network". In: *Advances in Neural Information Processing Systems (NeurIPS) Workshop* (cit. on pp. 19, 27, 62, 77, 78, 94, 102, 121).

Hochreiter, S., Y. Bengio, P. Frasconi, and J. Schmidhuber (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. Ed. by IEEE Press. Kremer, S. C.; Kolen, J. F. (eds.) (cit. on p. 7).

Hou, Saihui, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin (2019). "Learning a Unified Classifier Incrementally via Rebalancing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 15, 19, 21, 22, 27, 28, 31, 32, 34–40, 42, 48, 50, 59, 66, 94, 100, 112, 113, 115, 118, 121).

Huang, Gao, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger (2016). "Deep Networks with Stochastic Depth". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 8).

Huang, Z., X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu (2019). "CCNet: Criss-Cross Attention for Semantic Segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 59).

Huang, Zilong, Wentian Hao, Xinggang Wang, Mingyuan Tao, Jianqiang Huang, Wenyu Liu, and Xian-Sheng Hua (2021). "Half-Real Half-Fale Distillation for Class-Incremental Semantic Segmentation". In: *arXiv preprint library* (cit. on p. 80).

Huang, Zilong, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang (2020). "CCNet: Criss-Cross Attention for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 63).

Hung, Steven C.Y., Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen (2019). "Compacting, Picking and Growing for Unforgetting Continual Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 20).

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* (cit. on pp. 113, 116).

Ioffe, Sergey (2017). "Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 67).

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 7, 67, 124).

Iscen, Ahmet, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid (2020). "Memory-Efficient Incremental Learning Through Feature Adaptation". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 16).

Jaegle, Andrew, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira (2021). "Perceiver IO: A General Architecture for Structured Inputs & Outputs". In: *arXiv preprint library* (cit. on p. 10).

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual losses for real-time style transfer and super-resolution". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 31, 38, 39, 114).

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis (2021). "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873,

pp. 583–589. URL: https://doi.org/10.1038/s41586-021-03819-2 (cit. on p. 1).

Kankuekul, Pichai, Aram Kawewong, Sirinart Tangruamsub, and Osamu Hasegawa (2012). "Online Incremental Attribute-base Zero-shot Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 52).

Kemker, Ronald and Christopher Kanan (2018). "FearNet: Brain-Inspired Model for Incremental Learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 16).

Khetarpal, Khimya, Matthew Riemer, Irina Rish, and Doina Precup (2020). "Towards continual reinforcement learning: A review and perspectives". In: *arXiv preprint library* (cit. on p. 111).

Kim, Chris Dongjoo, Jinseo Jeong, and Gunhee Kim (2020). "Imbalanced Continual Learning with Partitioning Reservoir Sampling". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 15).

Kim, Dahyun, Jihwan Bae, Yeonsik Jo, and Jonghyun Choi (2019). "Incremental Learning with Maximum Entropy Regularization: Rethinking Forgetting and Intransigence". In: *arXiv preprint library* (cit. on p. 79).

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 8, 121).

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell (2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* (cit. on pp. 17, 27, 73, 74, 95).

Knuth, Donald E. (1997). *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Third. Boston: Addison-Wesley (cit. on p. 109).

Krizhevsky, Alex and Geoffrey Hinton (2009). "Learning multiple layers of features from tiny images". In: *Technical Report* (cit. on pp. 12, 34, 95, 120).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 7).

Kumar Verma, Vinay, Gundeep Arora, Ashish Mishra, and Piyush Rai (2018). "Generalized Zero-Shot Learning via Synthesized Examples". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 41).

Kwon, Jungmin, Jeongseop Kim, Hyunseo Park, and In Kwon Choi (2021). "ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep

Neural Networks". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 95, 100).

Lampert, C. H., H. Nickisch, and S. Hermeling (2009). "Learning to detect unseen object classes by between-class attribute transfer". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 41, 110).

Larrazabal, Agostina J., Nicolás Nieto, Victoria Peterson, Diego H. Milone, and Enzo Ferrante (2020). "Gender imbalance in medical imaging datasets produces biased classifiers for computer-aided diagnosis". In: *Proceedings of the National Academy of Sciences* (cit. on p. 2).

Lazebnik, Svetlana, Cordelia Schmid, and Jean Ponce (2006). "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories". In: *Object Categorization: Computer and Human Vision Perspectives, Cambridge University Press* (cit. on pp. 29, 63).

LeCun, Yann, Corinna Cortes, and CJ Burges (2010). "MNIST handwritten digit database". In: *ATT Labs* (cit. on p. 47).

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio (1999). "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision* (cit. on p. 7).

Lee, Dong-Hyun (2013). "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *International Conference on Machine Learning (ICML) Workshop* (cit. on p. 64).

Lee, Janghyeon, Hyeong Gwon Hong, Donggyu Joo, and Junmo Kim (2020). "Continual Learning with Extended Kronecker-factored Approximate Curvature". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 95, 107).

Lesort, Timothée (2022). "Continual Feature Selection: Spurious Features in Continual Learning". In: *arXiv preprint library* (cit. on p. 106).

Lesort, Timothée, Massimo Caccia, and Irina Rish (2021). "Understanding Continual Learning Settings with Data Distribution Drift Analysis". In: *Findings of CLVISION at IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop* (cit. on pp. 11, 67).

Lesort, Timothée, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat (2019a). "Generative models from the perspective of continual learning". In: *International Joint Conference on Neural Networks* (cit. on p. 16).

Lesort, Timothée, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez (2020). "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges". In: *Information Fusion* (cit. on p. 111).

Lesort, Timothée, Andrei Stoian, and David Filliat (2019b). "Regularization Shortcomings for Continual Learning". In: *arXiv preprint library* (cit. on pp. 13, 17, 27).

Li, Boyi, Felix Wu, Ser-Nam Lim, Serge Belongie, and Kilian Q. Weinberger (2021). "On Feature Normalization and Data Augmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 69).

Li, Yujia, Kevin Swersky, and Richard Zemel (2015). "Generative Moment Matching Networks". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 43, 49).

Li, Yunsheng, Lu Yuan, and Nuno Vasconcelos (2019). "Bidirectional learning for domain adaptation of semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 64).

Li, Z. and D. Hoiem (2016). "Learning without Forgetting". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 19, 22, 27, 59, 62, 73, 74).

Li, Zhuoyun, Changhong Zhong, Sijia Liu, Ruixuan Wang, and Wei-Shi Zheng (2021). "Preserving Earlier Knowledge in Continual Learning with the Help of All Previous Feature Extractors". In: *arXiv preprint library* (cit. on pp. 21, 88, 97, 121, 122).

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár (2017). "Focal loss for dense object detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 69).

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick (2014). "Microsoft COCO: Common Objects in Context". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 80).

Liu, Yaoyao, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun (2020). "Mnemonics Training: Multi-Class Incremental Learning without Forgetting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 16).

Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo (2021). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 10).

Lomonaco, Vincenzo and Davide Maltoni (2017). "CORe50: a New Dataset and Benchmark for Continuous Object Recognition". In: *Annual Conference on Robot Learning (CORL)* (cit. on p. 11).

Lomonaco, Vincenzo, Davide Maltoni, and Lorenzo Pellegrini (2020). "Rehearsal-Free Continual Learning over Small Non-I.I.D. Batches". In: *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop (cit. on pp. 67, 76).

Long, J., E. Shelhamer, and T. Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 59).

Lopez-Paz, David and Marc'Aurelio Ranzato (2017). "Gradient Episodic Memory for Continual Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (cit. on pp. 17, 110).

Lowe, David G. (1999). "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 7, 29).

Luo, Chunjie, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang (2018). "Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks". In: *International Conference on Artificial Neural Networks* (cit. on pp. 28, 38, 66).

Maaten, L.J.P. van der and G.E. Hinton (2008). "Visualizing High-Dimensional Data Using t-SNE". In: *Journal of Machine Learning Research* (cit. on p. 47).

Mallikarjuna, P, Alireza Tavakoli Targhi, Mario Fritz, Eric Hayman, Barbara Caputo, and Jan-Olof Eklundh (2006). "The kth-tips2 database". In: *Computational Vision and Active Perception Laboratory, Stockholm, Sweden* (cit. on p. 84).

Masoudnia, Saeed and Reza Ebrahimpour (2014). "Mixture of experts: a literature survey". In: *Artificial Intelligence Review* 42.2, pp. 275–293 (cit. on pp. 21, 94).

Masson d'Autume, Cyprien de, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama (2019). "Episodic Memory in Lifelong Language Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 111).

Mehta, Sachin, Mohammad Rastegari, Anat Caspi, Linda G. Shapiro, and Hannaneh Hajishirzi (2018). "Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 59).

Michieli, Umberto and Pietro Zanuttigh (2019). "Incremental Learning Techniques for Semantic Segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshop* (cit. on pp. 22, 56, 59, 60, 73–78, 109).

Michieli, Umberto and Pietro Zanuttigh (2021). "Continual Semantic Segmentation via Repulsion-Attraction of Sparse and Disentangled Latent Representations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 60, 73, 74, 118).

Mikolov, Tomas, Kai Chen, Greg S. Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint library* (cit. on pp. 41, 43, 110).

Mirzadeh, Seyed Iman, Arslan Chaudhry, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar (2022). "Wide Neural Networks Forget Less Catastrophically". In: *arXiv preprint library* (cit. on p. 107).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). "Playing Atari with Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems (NeurIPS) Workshop* (cit. on p. 111).

Moreron-Torresa, Jose G., Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera (2012). "A unifying view on dataset shift in classification". In: *Pattern Recognition* (cit. on pp. 11, 67).

Movshovitz-Attias, Yair, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh (2017). "No Fuss Distance Metric Learning Using Proxies". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 32, 33).

Mü, Samuel G. and Frank Hutter (2021). "TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 8).

Neuhold, Gerhard, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder (2017). "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 56).

Noh, H., S. Hong, and B. Han (2015). "Learning Deconvolution Network for Semantic Segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 59).

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill*. https://distill.pub/2017/feature-visualization (cit. on p. 8).

Oliphant, Travis E (2006). *A guide to NumPy*. Trelgol Publishing USA (cit. on pp. 113, 116).

Olsson, Viktor, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson (2021). "ClassMix: Segmentation-Based Data Augmentation for Semi-Supervised Learning". In: *Proceedings of the IEEE Winter Conference on Application of Computer Vision (WACV)* (cit. on p. 69).

Ozdemir, Firat, Philipp Fuernstahl, and Orcun Goksel (2018). "Learn the new, keep the old: Extending pretrained models with new anatomy and images". In: *International Conference on Med- ical Image Computing and Computer-Assisted Intervention* (cit. on pp. 56, 59).

Ozdemir, Firat and Orcun Goksel (2019). "Extending pretrained segmentation networks with additional anatomical structures". In: *International journal of computer assisted radiology and surgery* (cit. on pp. 56, 59).

Park, Sangyong and Yong Seok Heo (2020). "Knowledge Distillation for Semantic Segmentation Using Channel and Spatial Correlations and Adaptive Cross Entropy". In: *Sensors* (cit. on pp. 63, 77, 78).

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic Differentiation in PyTorch". In: *Advances in Neural Information Processing Systems (NeurIPS) Workshop* (cit. on pp. 35, 113, 116, 119).

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (cit. on pp. 41, 43).

Perez, Ethan, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville (2018). "FiLM: Visual Reasoning with a General Conditioning Layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (cit. on pp. 23, 124, 125).

Qi, Hang, Matthew Brown, and David G. Lowe (2018). "Low-Shot Learning With Imprinted Weights". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 33, 67).

Qian, Qi, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin (2019). "SoftTriple Loss: Deep Metric Learning Without Triplet Sampling". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 38).

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2019). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* (cit. on pp. 94, 111).

Rajasegaran, Jathushan, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Ming-Hsuan Yang (2019). "An Adaptive Random Path Selection Approach for Incremental Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 98, 122).

Ramachandram, Dhanesh and Graham W. Taylor (2017). "Deep Multimodal Learning: A Survey on Recent Advances and Trends". In: *IEEE Signal Processing Magazine* (cit. on p. 93).

Ramasesh, Vinay Venkatesh, Aitor Lewkowycz, and Ethan Dyer (2022). "Effect of scale on catastrophic forgetting in neural networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 14, 107).

Rame, Alexandre, Corentin Dancette, and Matthieu Cord (2021a). "Fishr: Invariant Gradient Variances for Out-of-distribution Generalization". In: *arXiv preprint library* (cit. on p. 107).

Rame, Alexandre, Remy Sun, and Matthieu Cord (2021b). "MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks". In: *arxiv* (cit. on p. 69).

Razavian, Ali Sharif, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson (2014). "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop* (cit. on p. 11).

Rebuffi, Sylvestre-Alvise, Hakan Bilen, and Andrea Vedaldi (2017a). "Learning multiple visual domains with residual adapters". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 21).

Rebuffi, Sylvestre-Alvise, Hakan Bilen, and Andrea Vedaldi (2017b). "Learning multiple visual domains with residual adapters". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 124).

Rebuffi, Sylvestre-Alvise, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert (2017c). "iCaRL: Incremental Classifier and Representation Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 13, 15, 19, 27, 28, 34–37, 40, 50, 73, 74, 95–98, 100, 111, 112, 121).

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 7).

Robins, Anthony (1995). "Catastrophic Forgetting, Rehearsal and Pseudorehearsal". In: *Connection Science* (cit. on pp. 2, 12, 56, 62).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)* (cit. on p. 59).

Rumelhart, David, Geoffrey Hinton, and Ronald Williams (1986). "Learning representations by back-propagating errors". In: *Nature* (cit. on p. 6).

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* (cit. on pp. 7, 10).

Russel, Jesse (2020). "Machine Learning Fairness in Justice Systems: Base Rates, False Positives, and False Negatives". In: *arXiv preprint library* (cit. on p. 2).

Rusu, Andrei A, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell (2016). "Progressive neural networks". In: *arXiv preprint library* (cit. on p. 20).

Saha, Gobinda, Isha Garg, and Kaushik Roy (2021). "Gradient Projection Memory for Continual Learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 18).

Saporta, Antoine, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2020). "ESL: Entropy-guided Self-supervised Learning for Domain Adaptation in Semantic

Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop* (cit. on pp. 22, 64, 65).

Scheirer, W. J., A. Rocha, A. Sapkota, and T. E. Boult. (2013). "Toward Open Set Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on p. 48).

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 1).

Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra (2017). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 19, 27, 94).

Sermanet, Pierre, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun (2014). "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 59).

Serrà, Joan, Dídac Surís, Marius Miron, and Alexandros Karatzoglou (2018). "Overcoming catastrophic forgetting with hard attention to the task". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 20, 122).

Sheng, Emily, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng (2019). "The Woman Worked as a Babysitter: On Biases in Language Generation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (cit. on p. 2).

Shin, Hanul, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim (2017). "Continual Learning with Deep Generative Replay". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 16).

Shmelkov, Konstantin, Cordelia Schmid, and Karteek Alahari (2017). "Incremental learning of object detectors without catastrophic forgetting." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 109).

Srivastava, Rupesh K., Klaus Greff, and Jürgen Schmidhuber (2015). "Training very deep networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 7).

Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov (2020). "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 1).

Sun, Qianru, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele (2019). "Meta-Transfer Learning for Few-Shot Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 21).

Sutton, R. S. and A. G. Barto (1998). "Introduction to Reinforcement Learning". In: *MIT Press, Cambridge, MA, USA, 1st edition, ISBN 0262193981* (cit. on p. 111).

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 7, 19, 27).

Tao, Andrew, Karan Sapra, and Bryan Catanzaro (2020). "Hierarchical Multi-Scale Attention for Semantic Segmentation". In: *arXiv preprint library* (cit. on pp. 56, 59).

Tasar, O., Y. Tarabalka, and P. Alliez (2019). "Incremental Learning for Semantic Segmentation of Large-Scale Remote Sensing Data". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (cit. on pp. 56, 59).

Thangarasa, Vithursan, Thomas Miconi, and Graham W. Taylor (2020). "Enabling Continual Learning with Differentiable Hebbian Plasticity". In: *International Joint Conference on Neural Networks* (cit. on p. 108).

Thulasidasan, Sunil, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak (2019). "On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 95).

Tommasi, Tatiana, Silvia Bucci, Barbara Caputo, and Pietro Asinari (2021). "Towards Fairness Certification in Artificial Intelligence". In: *arXiv preprint library* (cit. on p. 2).

Töscher, Andreas and Michael Jahrer (2009). "The BigChaos Solution to the Netflix Grand Prize". In: (cit. on p. 1).

Touvron, Hugo, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou (2021a). "Training data-efficient image transformers and distillation through attention". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 10, 96, 121–123).

Touvron, Hugo, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou (2021b). "Going deeper with image transformers". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 10, 91, 96).

Tranheden, Wilhelm, Viktor Olsson, Juliano Pinto, and Lennart Svensson (2021). "DACS: Domain Adaptation via Cross-Domain Mixed Sampling". In: *Proceedings of the IEEE Winter Conference on Application of Computer Vision (WACV)* (cit. on p. 69).

Vapnik, Vladimir (1999). "An overview of statistical learning theory". In: *IEEE transactions on neural networks* (cit. on p. 5).

Vásquez, Alex, Arnaud Dapogny, Kévin Bailly, and Véronique Perdereau (2017). "Sequential recognition of in-hand object shape using a collection of neural forests". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (cit. on p. 11).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 1, 8, 88, 92, 94, 101, 111).

Veniat, Tom and Ludovic Denoyer (2018). "Learning time/memory-efficient deep architectures with budgeted super networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 106).

Veniat, Tom, Ludovic Denoyer, and Marc'Aurelio Ranzato (2021). "Efficient Continual Learning with Modular Networks and Task-Driven Priors". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 20).

Verma, Vikas, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio (2019). "Manifold Mixup: Better Representations by Interpolating Hidden States". In: *International Conference on Machine Learning (ICML)* (cit. on p. 69).

Verwimp, Eli, Matthias De Lange, and Tinne Tuytelaars (2021). "Rehearsal revealed: The limits and merits of revisiting samples in continual learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop* (cit. on p. 95).

Volpi, Riccardo, Diane Larlus, and Grégory Rogez (2021). "Continual Adaptation of Visual Representations via Domain Randomization and Meta-learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 11).

Vu, Tuan-Hung, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez (2019). "ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 64).

Wang, Huiyu, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen (2020). "Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on pp. 56, 63).

Wang, Tongzhou, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros (2018). "Dataset distillation". In: *arXiv preprint library* (cit. on p. 16).

Wei, Kun, Cheng Deng, and Xu Yang (2020). "Lifelong Zero-Shot Learning". In: *International Joint Conference on Artificial Intelligence* (cit. on p. 110).

Wen, Yeming, Dustin Tran, and Jimmy Ba (2020). "BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 21).

Wightman, Ross (2019). *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models (cit. on p. 96).

Wightman, Ross, Hugo Touvron, and Hervé Jégou (2021). "ResNet strikes back: An improved training procedure in timm". In: *arXiv preprint library* (cit. on p. 8).

Wortsman, Mitchell, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi (2020). "Supermasks in Superposition for Continual Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 20, 76).

Wu, Yue, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu (2019). "Large Scale Incremental Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 21, 34–37, 39, 40, 66, 93, 97, 98, 100, 111, 112, 121).

Xian, Y., C. H. Lampert, B. Schiele, and Z. Akata (2019). "Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on pp. 41, 47, 48, 51, 110).

Xian, Yongqin, Tobias Lorenz, Bernt Schiele, and Zeynep Akata (2018). "Feature Generating Networks for Zero-Shot Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 41).

Xue, Nan, Yi Wang, Xin Fan, and Maomao Min1 (2017). "Incremental Zero-Shot Learning Based On Attributes for Image Classification". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (cit. on p. 52).

Yan, Shipeng, Jiangwei Xie, and Xuming He (2021). "DER: Dynamically Expandable Representation for Class Incremental Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 20, 22, 88, 89, 94, 96–98, 100, 120–122).

Yin, Hongxu, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz (2019). "Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion". In: *Proceedings of the IEEE Winter Conference on Application of Computer Vision (WACV)* (cit. on p. 80).

Yu, Lu, Xialei Liu, and Joost van de Weijer (2020). "Self-Training for Class-Incremental Semantic Segmentation". In: *arXiv preprint library* (cit. on pp. 60, 80).

Yuan, Yuhui, Xilin Chen, and Jingdong Wang (2020). "Object-Contextual Representations for Semantic Segmentation". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 59).

Yuan, Yuhui and Jingdong Wang (2018). "OCNet: Object Context Network for Scene Parsing". In: *arXiv preprint library* (cit. on p. 59).

Yun, Sangdoo, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo (2019). "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 69, 121).

Zagoruyko, Sergey and Nikos Komodakis (2016). "Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 27, 31, 118).

Zenke, Friedemann, Ben Poole, and Surya Ganguli (2017). "Continual Learning Through Synaptic Intelligence". In: *International Conference on Machine Learning (ICML)* (cit. on pp. 17, 73, 74).

Zhai, Mengyao, Lei Chen, Fred Tung, Jiawei He, Megha Nawhal, and Greg Mori (2019). "Lifelong GAN: Continual Learning for Conditional Image Generation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 16).

Zhang, Hang, Kristin J. Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal (2018). "Context Encoding for Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 59).

Zhang, Hang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola (2020). "ResNeSt: Split-Attention Networks". In: *arXiv preprint library* (cit. on pp. 56, 59).

Zhang, Hongyi, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz (2018). "mixup: Beyond Empirical Risk Minimization". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on pp. 8, 69, 83, 94, 95, 98, 100, 121).

Zhang, Jiawei, Yanchun Zhang, and Xiaowei Xu (2021). "ObjectAug: Object-level Data Augmentation for Semantic Image Segmentation". In: *arXiv preprint library* (cit. on p. 69).

Zhao, Bowen, Xi Xiao, Guojun Gan, Bin Zhang, and Shutao Xia (2020). "Maintaining Discrimination and Fairness in Class Incremental Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 21, 27, 66, 93, 94, 97, 98, 100, 121, 124).

Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia (2017). "Pyramid Scene Parsing Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 59).

Zhao, Hengshuang, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia (2018). "PSANet: Point-wise Spatial Attention Network for Scene Parsing". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 59).

Zhong, Zhun, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang (2017). "Random Erasing Data Augmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (cit. on pp. 8, 121).

Zhou, Bolei, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba (2017). "Scene Parsing Through ADE20K Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 56, 59, 71, 72, 117).

Zhou, Peng, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S. Davis (2019). "M2KD: Multi-model and Multi-level Knowledge Distillation for Incremental Learning". In: *arXiv preprint library* (cit. on pp. 19, 27).

Zou, Yang, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang (2018). "Unsupervised domain adaptation for semantic seg- mentation via class-balanced self-training". In: *Proceedings of the IEEE European Conference on Computer Vision (ECCV)* (cit. on p. 64).