Large-scale deep class-incremental learning

Eden Belouadah

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE MINES-TÉLÉCOM ATLANTIQUE
BRETAGNE PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Eden BELOUADAH

Large-Scale Deep Class-Incremental Learning

Thèse présentée et soutenue à l'amphithéâtre 34 du bâtiment 862 de Nano-Innov, CEA-LIST, Palaiseau, le 29 novembre 2021 à 9h00
Unité de recherche : IMT Atlantique, CNRS/Lab-STICC/RAMBO et CEA-LIST, DIASI/SIALV/LASTI
Thèse N° : 2021IMTA0281

Rapporteurs avant soutenance :

Nicu SEBE	Professeur, University of Trento, Italie
David FILLIAT	Professeur, ENSTA Paris, Institut Polytechnique de Paris, France

Composition du Jury :

Présidente :	Céline HUDELOT	MDC/HDR, CentraleSupélec, France
Examinateurs :	Tinne TUYTELAARS	Professeure, Katholieke Universiteit, Belgique
	Symeon PAPADOPOULOS	Chercheur Senior, CERTH-ITI, Grèce
	Nicu SEBE	Professeur, University of Trento, Italie
	David FILLIAT	Professeur, ENSTA Paris, Institut Polytechnique de Paris, France
Dir. de thèse :	Ioannis KANELLOS	Professeur, IMT Atlantique, France
Encadrant :	Adrian POPESCU	Ingénieur Chercheur, CEA-LIST, France

LIST OF ABBREVIATIONS

CL	Continual Learning
CF	Catastrophic Forgetting
IL	Incremental Learning
AL	Active Learning
DL	Deep Learning
SL	Supervised Learning
FT	Fine Tuning
FR	Fixed Representation
NEM	Nearest Exemplars Mean
iCaRL[‡]	incremental Classifier and Representation Learning
BiC[‡]	Bias Correction
LUCIR[‡]	Learning a Unified Classifier Incrementally via Rebalancing
LwF[‡]	Learning without Forgetting
E2EIL[‡]	End-to-end Incremental Learning
REMIND[‡]	REplay using Memory INDexing
Deep-SLDA[‡]	Deep Streaming Linear Discriminant Analysis
SVM	Support Vector Machine
DFE	Deep Feature Extractor
MLP	Multi Layer Perceptron
CNN	Convolutional Neural Network
i.i.d.	Independent and identically distributed
IL2M[†]	Incremental Learning with Dual Memory
ScaIL[†]	Classifier weights scailing for Class-Incremental Learning
SIW[†]	Standardization of Initial Weights
TransIL[†]	Dataset Knowledge Transfer for Class-incremental Learning
DeeSIL[†]	Deep-Shallow incremental Learning
AdBiC[†]	Adaptive Bias Correction

[†]: proposed methods

[‡]: main baselines

LIST OF SYMBOLS

$\mathcal{T}$	total number of states
$\mathcal{K}$	exemplars memory
$\mathcal{I}$	secondary memory
$\mathcal{B}$	active learning (AL) budget
$\mathcal{Z}$	number of AL annotation iterations
$\mathcal{S}_t$	state number t
$\mathcal{M}_t$	Model in state $\mathcal{S}_t$
$\mathcal{F}_t$	feature extractor in state $\mathcal{S}_t$
$\mathcal{C}_t$	classification component in state $\mathcal{S}_t$
$\boldsymbol{\mathcal{W}_t}$	last layer weights matrix in state $\mathcal{S}_t$
$\boldsymbol{W_t^j}$	embedding of class j (a row from $\mathcal{W}_t$)
$\boldsymbol{\mathcal{B}_t}$	last layer bias vector in state $\mathcal{S}_t$
$\mathcal{X}^r$	reference dataset
$\mathcal{X}_t$	new classes data (target dataset) in state $\mathcal{S}_t$
X_t^j	set of images of class j in state $\mathcal{S}_t$ (a subset from $\mathcal{X}_t$)
F^j	set of feature vectors of all images of class j
$\boldsymbol{o_t}$	raw output scores vector in state $\mathcal{S}_t$
o_t^j	raw output score of class j in state $\mathcal{S}_t$
$o_t^{j'}$	rectified score of class j in state $\mathcal{S}_t$
$\boldsymbol{p_t}$	output probabilities vector in state $\mathcal{S}_t$
N_t	total number of classes recognized until state $\mathcal{S}_t$ included
P_t	number of new classes learned in state $\mathcal{S}_t$
$\mathcal{L}^c$	classification loss
$\mathcal{L}^d$	distillation loss
σ	softmax function
T	Temperature
$\hat{\sigma}$	softened softmax function (σ with $T = 2$)
t, k	indices for states
i	index for initial states

z	index for AL iteration
j, l	indices for classes
ϕ	parameters of the feature extractor
ψ	parameters of the classification layer
θ	all network parameters ($\theta_t = \{\phi_t, \psi_t\}$)
$\lvert \cdot \rvert$	number of elements (in the memory $\mathcal{K}$ for example)
A	accuracy
$\mathcal{A}$	Class-IL algorithm
y	ground-truth label
$\hat{y}$	predicted label
R	number of reference datasets
G_{IL}	aggregation measure for evaluation
α, β	calibration parameters
δ	Kronecker delta

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Context and background

Artificial Intelligence

In our complex world, many technological fields evolve at a tremendous speed that requires keeping pace with their development. This development was accompanied by a significant increase in the amount of data, especially with the advent of the era of big data [1]. For example, billions of images and content are added every day to social media such as Facebook, Instagram, and Twitter. Big companies (such as Google, Amazon, and Facebook) use this data to improve user's experience. Behind such services, intelligent systems are trained with huge data collected from different users on different platforms. Other examples include smart digital assistants such as *Alexa*, *Siri* and *Google Assistant* that rely on artificial intelligence to recognize the human voice and carry out what is requested of them. Simply put, artificial intelligence has become pervasive, and its integration in applications and services is likely to become even more prevalent in the future [93].

Neural Networks and Deep Learning

The most popular and efficient artificial intelligence systems today are based on artificial neural networks (ANNs), or simply *neural networks* (NNs). Their development is based on multidisciplinary work which includes contributions from computer science, mathematics, cognitive science, statistics, psychology, and neuroscience.

Neural networks simulate the working mechanism of biological neurons in two aspects:

— *The structure* - a neural network consists of a certain number of nodes (called *neurons*) connected together through artificial connections.

— *The behavior* - artificial neurons imitate biological neurons in how they generate and transmit signals between them.

Neural networks learn from examples, and the more relevant examples they encounter, the better is their experience, and thus their performance. The theoretical progress in AI, the development of computation power [10, 84], and the availability of large amounts of data lead to the emergence of Deep Learning. The latter refers to neural networks with a large number of layers. Deep neural networks require thus up to thousands or millions of data examples to learn efficiently. Many ANNs have been proposed after the formal neuron. The most popular architectures nowadays include: Convolutional Neural Networks (CNNs) [80], Recurrent Neural Networks (RNNs) [134, 63], and Transformer Neural Networks [165]. In this thesis, we are interested in image classification with deep convolutional neural networks.

Image classification with Convolutional Neural Networks

We focus on *Supervised Learning (SL)* and provide a simple illustration of it in Figure 1.1. Simply put, we focus on an automatic system that can classify a set of images into a specific number of classes that belong or not to the same visual field, for example, house, giraffe, car, lasagna, and phone. To do this, a set of images and their labels are showed several times to the AI system. The goal is to guide the system by indicating the number of errors it makes in order to improve the quality of its learning and correct its mistakes. This stage is called the learning stage. When it finishes, the system is provided with a set of images that it did not see in the learning phase. The goal of the system is to classify them according to the classes that it recognized during the learning stage. Here, the simplest way to evaluate the system is to count the number of correctly classified images. This stage is called the evaluation, testing, or inference stage. Note that we do not address in this thesis other types of learning (unsupervised and semi-supervised) [141].

Convolutional Neural Networks (CNNs) were introduced by LeCun et al. [79] in 1989 to solve the problems of the full connectivity of multi-layer perceptrons [113]. CNNs are based on convolutions, and the associated filters are equivalent to the values of the weights in MLPs. Here, the difference with MLPs is that a neuron operates on a group of input neurons (called *receptive-field*), where the convolution is computed. This sparsity in the weights helps to considerably reduce the number of learnable parameters, making CNNs more efficient to learn without overfitting the training set.

Two pioneering works which contributed to the current prevalence of CNNs in image classification are Ciresan's multi-column architecture [29], and Krizhevsky's AlexNet [75]. The latter won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [32, 135] in 2012. These works and the increasing attention given to the ILSVRC challenge led to further progress

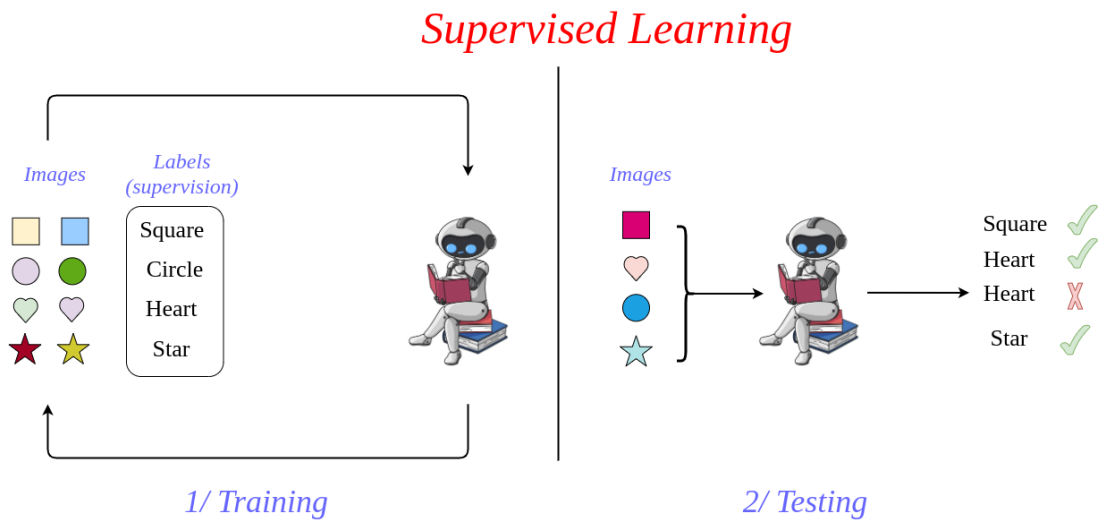


Figure 1.1 – Supervised Learning in image classification

in image classification. Among the latter important contributions, we cite the following CNN architectures: VGGNet [153], GoogleNet [159], ResNet [51], Wide ResNet [183], DenseNet [57], PolyNet [188], ResNext [177], Xception [28], DarkNet [129], etc. In this thesis we use a ResNet-18 architecture [51] containing 18 convolutional layers.

Class-Incremental Learning as a Continual Learning paradigm

While strong progress was achieved in artificial intelligence-related tasks, such as image classification, standard artificial agents are designed to be trained in a static manner. In contrast, real-world data is often dynamic [6] and artificial agents should be able to ingest new information without full retraining. In other words, it is not enough for AI systems to learn, but it is important that they learn continuously. This way of learning is called *Continual Learning (CL)* or *LifeLong Learning (LFL)*. It is interesting insofar as it avoids training from scratch when new data is ingested, thus reducing the computational requirements and memory footprint. The memory limitation is often encountered in robotics [83], embedding systems (such as mobile devices), and when regulation limits data storage and distribution (as is the case for health-related data). Some environment-friendly applications also use CL to reduce the carbon footprint [158, 149].

Figure 1.2 shows a simplified supervised continual learning process. Depending on the application domain, one CL cycle is called task [101, 7, 9], state [128, 174, 24], or experience [83].

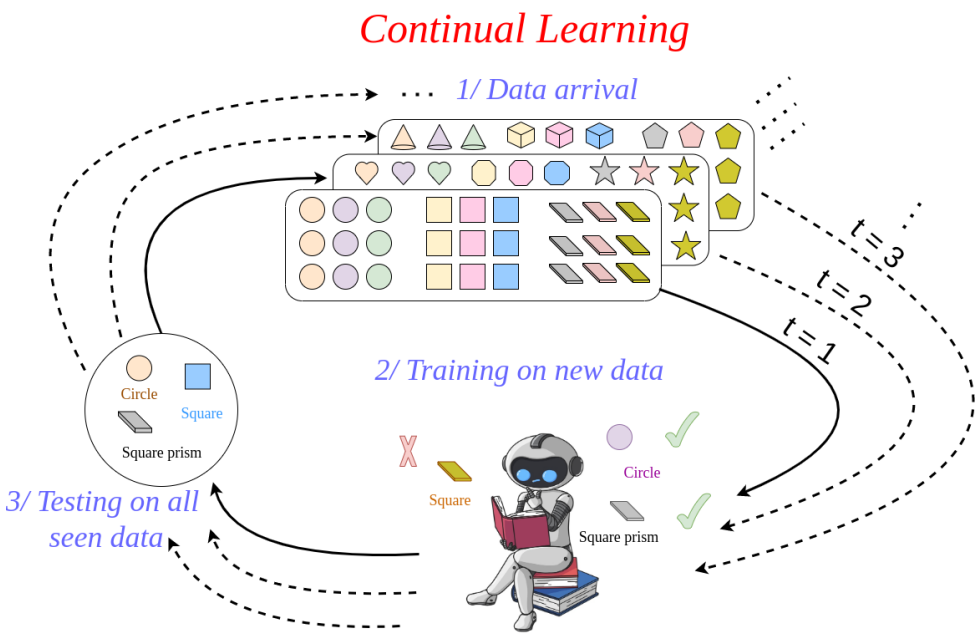


Figure 1.2 – Continual Learning life cycle.

The cycle starts by feeding the AI system with new data. The AI system uses it to update its knowledge in a supervised manner. After learning the new data, the system is evaluated on a separate testing set before being fed again with new data and starting a new life cycle. It is noteworthy that while the AI system receives tasks sequentially, it can access data from a single task while being evaluated on all tasks seen so far. In some cases [128, 174], a bounded memory of past data is allowed. A detailed discussion of the memory is provided in Chapter 2.

There are three main scenarios of continual learning [166]. Namely: *Task-incremental learning* [100, 9], *Domain-incremental Learning* [8, 47, 48] , and *Class-incremental Learning* [128, 189, 53]. The three areas are related but we briefly highlight the differences between them hereafter:

— **Task-Incremental Learning** - the task boundaries (number of new classes) are known, and the task-id is provided at inference time. Here, the system does not discriminate classes from different tasks but only those belonging to the same task. Equally important, tasks are semantically coherent since the system learns each time classes belonging to a different domain, such as letters, then planes, cars, and birds.

— **Class-Incremental Learning** - a CL scenario that is task-agnostic (the task-id is not known at inference time), but the task boundaries are known in the sense that past classes

are not repeated in future states. The system should discriminate between all classes seen so far, and not only new classes. Task-incremental learning can be seen as a more relaxed version of class-incremental learning, where task-id helps restraining the search space during inference. Equally important, class-incremental learning is more complicated than task-IL because states include classes that are mixed from a semantic point of view. Note that the task is called *state* in IL systems of this group, and this is how we call it in the rest of the thesis.

— **Domain-Incremental Learning** - this is the most challenging scenario, where the task boundaries are not known. Here, new data appears in a stream and should be ingested online. New streamed data can belong or not to past classes. Methods in this group usually visit each data example once during all the training process.

We are interested in class-incremental learning. This scenario is more challenging than task-incremental learning, but contrarily to domain-incremental learning, it assumes that data arrives in groups of classes with known boundaries. Figure 1.3 illustrates the difference between the three main areas of continual learning.

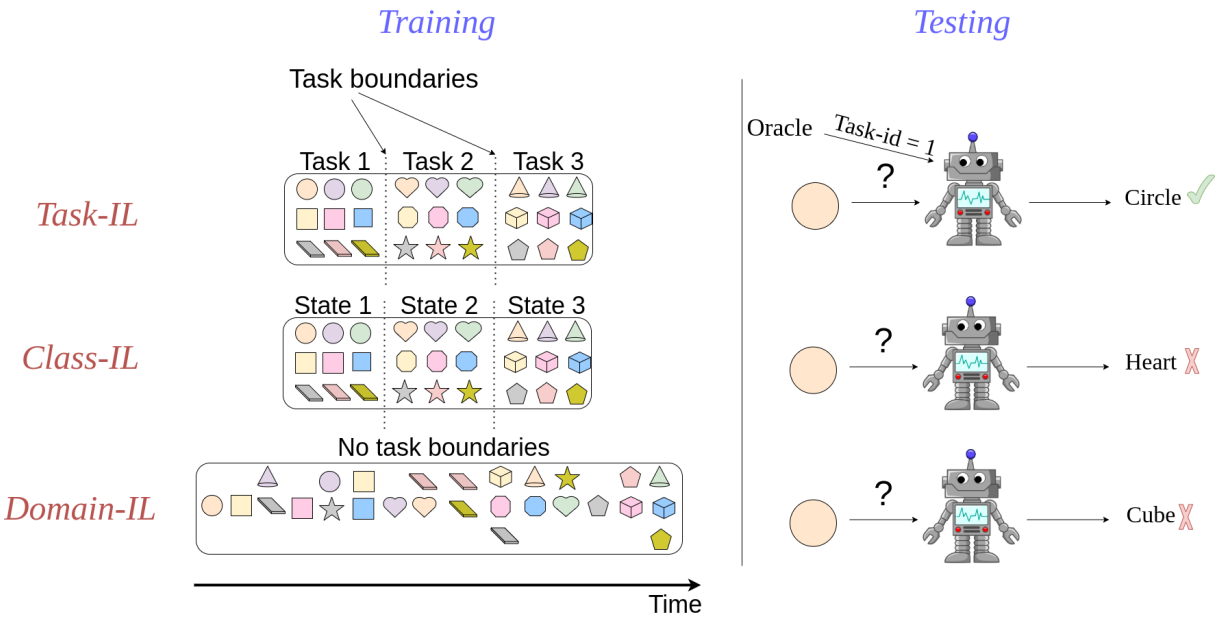


Figure 1.3 – Three main scenarios of continual learning.

1.2 Open challenges in Continual Learning

As already explained, continual learning consists of updating system capacities through a long-life cycle, where knowledge continuously appears over time. This task is easy if full retraining is allowed but becomes very challenging otherwise. The authors of [102] noticed what they called the *catastrophic interference* or *catastrophic forgetting*, a phenomenon from which suffer systems that are trained with back-propagation. Catastrophic forgetting (CF) is the tendency of neural networks to underfit (forget) past knowledge when trying to accommodate a new one. Later, CF was experimentally proved by [124] who studied the behavior of many back-propagation-based systems. One early work which brought attention to CF in the context of deep learning is [45]. The study consisted of an experimental investigation of the effects and the reasons behind CF in gradient-based systems.

Catastrophic forgetting remains the main problem that continual learning systems tackle. We exemplify its occurrence in image classification below. When new classes are ingested, the network weights are updated to minimize a loss on images of these classes. The consequence of this update is called *weight drift* because weights that were important for past classes in previous states have been changed [101]. The change in weights leads to a change in activations (output) of the network (*activations drift*). In class-incremental learning, the activation drift is called *recency bias*, and it refers to the tendency of IL systems to favor newly learned classes in each incremental state. This is because new classes are learned each time with all their data, while past ones have, in the best case, a limited number of exemplars to be replayed for them from the memory. Effects of recency bias were observed at the level of prediction scores [174] or weights matrix [189] of the classification layer. The effect of recency bias is a drastic drop in performance on past classes. A more detailed analysis of the effects of catastrophic forgetting is proposed in Chapters 3 and 4.

Along with catastrophic forgetting, many challenges make class-incremental learning difficult to tackle. In 2017, the authors of $iCaRL$ [128], an early and influential class-incremental learning system based on a deep learning backbone, rightfully note that there exists no satisfactory algorithm that can qualify as class-incremental. They frame three necessary properties of it: (1) be trainable from new stream data that occurs arbitrarily; (2) provide competitive performance for past classes when new ones are integrated, and (3) computational requirements and memory footprint should remain bounded. With the advances in class-incremental learning algorithms in the last five years, we revisit the properties proposed in [128] (marked with * below) and define six desirable properties that an ideal continual learning algorithm should fulfill:

— **Complexity*** - capacity to integrate new information with a minimal change in terms of the model structure. For a deep neural network, only the size of the classification layer should grow. Otherwise, the total number of model parameters is likely to increase strongly, especially at a large scale. Dynamic networks [136, 170, 7] fail to satisfy this challenge since the first two works expand the models capacity by increasing its architecture throughout the incremental process. The third work [7] adds one model per task, which is heavy at a large scale.

— **Memory*** - ability to work with or without a fixed-size memory of past classes. Naturally, algorithms that do not require past data are preferable, but this condition is hard to satisfy. Distillation-based approaches [86] require saving the previous (teacher) model to compute the loss. Bias-removal approaches [174, 189] require saving at least some floating values to make use of past knowledge. The performance of methods that do not use an exemplar memory is lower, especially if complexity growth is minimized.

— **Accuracy*** - performance for past and new classes should approach that of a non-incremental learning process that has access to all data at all times. This goal is not achieved by any continual learning approach because of the CF [102] problem. All continual learning algorithms struggle to reduce their gap with a non-incremental learning.

— **Timeliness** - delay needed between the occurrence of new data and its integration in the incremental models. Most works [128, 53, 24, 97, 96, 173, 152] do not consider the additional training time needed to incorporate the components that reduce the catastrophic forgetting. For instance, the NEM classifier in [128], the balancing FT step in [24], the time needed to free unimportant parameters in [96, 97], and the time needed to compute the sophisticated objective in [53]. On the contrary, fixed-representation-based approaches [47, 66, 48] are faster since no deep retraining is done incrementally.

— **Plasticity-Stability** - capacity not only to deal with new classes that are significantly different from the ones learned in the past but also to keep as much knowledge as possible from the past. This challenge is not evident because, on the one hand, several updates of the model worsen the catastrophic forgetting and increase its bias [109] towards new classes. On the other hand, not updating the model would lead to a problem of *intransigence* (the resistance to learning new classes [26]) that makes it not efficient to recognize future classes that belong to other visual domains. This problem is known under the plasticity-stability dilemma [104]. Task-incremental learning methods [7] are more robust against domain shift between incremental states.

— **Scalability** - the aptitude for learning a large number of classes, typically up to tens of

thousands, and ensure usability in complex real-world applications. Fixed Networks [96, 97] are not scalable because they continuously free unimportant weights and allocate them to learn new classes until the network cannot be compressed anymore. Similarly, methods that are highly dependent on exemplars memory suffer from catastrophic forgetting when the number of classes exceeds the memory size since more and more classes will not have any more representatives to be replayed for them.

In fact, the difficulty comes mainly from satisfying all properties jointly. It would be good to illustrate this with some examples. For instance, if complexity is stable, it is hard to maintain accuracy. If timeliness is very important and resources are scarce, plasticity will suffer because it is difficult to retrain.

Other challenges are related to the experimental protocol to assess the quality of incremental learning algorithms. We list below some of these challenges:

— **Data imbalance** - class-IL becomes more challenging if there is an important variation of the number of images per class [4]. Catastrophic forgetting is mainly due to the absence of images for past classes. When a memory of the past is allowed, class-IL is akin to an imbalanced learning since new classes are learned with all their data, while past classes are learned with the exemplars in the memory only. Adding more imbalance among new classes makes the training of the model more difficult.

— **Data annotation** - Most existing works assume that new data is entirely labeled [128, 35, 7, 174, 101]. This is not the case in real-life situations. For instance, only a small part of the images on the web are annotated. Data annotation is time and money expensive and for this reason active learning [184, 146, 144] and semi-supervised learning [85, 91, 77, 23, 138, 169] approaches saw the light. The former selects the most representative set of images to annotate by an expert while discarding the unlabeled images. The latter annotates the set of unlabeled images automatically. Active learning is still not widely used in class-incremental learning, while semi-supervised learning is now in fashion in class-IL. Note that the three domains can be combined for more efficiency. Assuming that new data is labeled relaxes the difficulty of class-incremental learning approaches but should be considered if we aim for real-life applications.

— **Size of class batches** - One challenge could be the number of classes added at each incremental state. Adding one class per state would add more rehearsals to fine-tuning-based approaches [128, 53, 24] and causing faster forgetting. This, however, does not have a negative effect on fixed-representations [47, 66, 48]. We vary this parameter in our experiments.

— **Selection of exemplars** - When a memory of the past is allowed, other challenges are encountered in class-IL regarding the criterion on which the exemplars are selected, and the number of exemplars to keep in the memory for each class. This challenge was largely debated in literature [128, 61, 89], and we provide our own discussion about it in Chapter 3.

In this thesis, we focus on approaches which: (1) keep the complexity of the model fixed, (2) work with and without a bounded memory of the past, (3) propose one method which favors stability, and four methods that balance stability and plasticity, (4) learn (but are not limited to) 1000 classes, (5) tackle data imbalance and non-availability of annotated data, and (6) vary the number of classes per incremental state. While focusing on one or several of these points, the ultimate goal of the proposed contributions is to increase the accuracy of our IL systems and reduce their gap with a standard learning from scratch.

1.3 Contributions overview

The focus of this work is on: **Large-scale** (systems that are able to learn at least 1000 classes) **Deep** (we use deep convolutional neural network) **class-incremental learning** (AI system where new data appears continuously in groups of classes). Naturally, class-incremental learning methods that we visit in this thesis can be deployed for any kind of classification, such as that of text [58] or sentiments [65].

In Chapter 2, we analyze relevant continual learning systems and focus on those that are usable in a class-incremental learning scenario. The next two chapters (3 and 4) are dedicated to our contributions to class IL with and without memory, respectively. We briefly describe these approaches hereafter.

Class-incremental learning with memory (Chapter 3):

1 – Deep-Shallow Incremental Learning (DeeSIL) (Section 3.2) - is a fixed-representation-based approach, which applies a standard transfer-learning scheme [44, 125] to an incremental learning scenario. A deep representation is learned using the training set of the first incremental state and then exploited as a feature extractor for the other classes learned incrementally. A convolutional neural network is used for features extraction, and a battery of Support Vector Machines (SVMs) [19] is used to learn new classes. When a memory of the past is allowed, it stores the SVMs negatives from past states. When no memory of the past is allowed, this method

is still usable by replacing the negatives of the memory with negatives from classes belonging to the same state as the class that the system is currently learning. This work was published in the TaskCV workshop of the *European Conference on Computer Vision* 2018 [12].

2 – Incremental Learning with Dual Memory (IL2M) (Section 3.3) - uses a secondary memory to store statistics of new classes in each state. In future states (when these classes become past classes), the stored statistics of past classes are used to rectify their predictions in order to make them more comparable to those of new classes. This work was published in the *International Conference on Computer Vision* 2019 [13].

3 – Classifier Weights Scaling for Class-Incremental Learning (ScaIL) (Section 3.4) - Since past classes are better learned in initial states in which they were encountered for the first time, we perform initial weights replay for them, and normalize updated weights matrix using statistics computed over new class weights of each state. This work was published in the *IEEE Winter Conference on Applications of Computer Vision* 2020 [14].

4 – Active class-incremental learning for imbalanced datasets (Section 3.6) - We assume that the new data is not entirely labeled and dataset is strongly imbalanced. This is a more challenging scenario of class-IL with memory. The main contributions of this section are: (1) the proposal of two acquisition functions that help us choosing the subset of training images to label by an expert (the rest of unlabeled images is discarded), and (2) the adaptation of the thresholding method from [21] to class-IL. The latter is efficient in tackling the bias of the network caused by data imbalance. This work was published in the IPCV workshop of the *European Conference on Computer Vision* 2020 [17].

5 – A Comprehensive Study of Class Incremental Learning Algorithms for Visual Tasks (Sections 3.5 and 4.4) - We conduct extensive experiments to assess the merits and limitations of popular class-incremental methods and variants of them. We notably study the effect of distillation loss [52], the herding mechanism to select past-class exemplars (in case a bounded memory is allowed), and the effect of not having a memory of the past. The study is divided in two parts, the one with memory is presented in Section 3.5 of Chapter 3, and the one without memory is in Section 4.4 of Chapter 4. The main conclusion is that none of the existing IL algorithms is always best in all configurations. This work was published in the *Elsevier's Neural Networks* journal [15].

Class-incremental learning without memory (Chapter 4):

This scenario is more challenging than the preceding one, but it is more interesting and useful in practice, especially when access to past data is impossible. We propose two methods that can be implemented on top of many IL algorithms to improve their performance.

1 – Standardization of Initial Weights for Class-Incremental without memory (SIW) (Section 4.2) - Sharing the same spirit as $ScaIL$, this approach also performs initial weights replay for past classes. At the same time, the normalization is done by standardizing weights vectors of all classes. This work was published in the *British Machine Vision Conference* 2020 [16].

2 – Dataset Knowledge Transfer for Class-Incremental Learning without Memory (TransIL) (Section 4.3) - We propose to transfer calibration parameters from reference to target datasets to make bias correction methods such as BiC [174] usable in a memoryless IL setting. Besides, we propose $AdBiC$, a refinement of BiC, which rectifies past class scores based on their initial IL state (in which they were encountered for the first time). This work was published in the *IEEE Winter Conference on Applications of Computer Vision* 2022 [155].

We conclude this work in Chapter 5. We present a brief discussion of the memory footprint required by our proposed approach. In addition, we present a diagram of IL methods before and after our proposed methods to help understand which type of IL scenarios our methods are more useful. We finally present some open directions for future research.

The full list of publications related to this thesis can be found in Appendix A. Datasets and implementation details are in Appendix B. All our codes and datasets details are publicly available to facilitate reproducibility [1]

1. https://github.com/EdenBelouadah/class-incremental-learning

CHAPTER 2

STATE OF THE ART OF CLASS-INCREMENTAL LEARNING

2.1 Introduction

Artificial agents that evolve in dynamic environments should be able to update their capabilities in order to integrate new data. Depending on the work hypotheses made, names such as continual learning [142, 152], lifelong learning [7, 116] or incremental learning (IL) [24, 51, 128] are used to describe associated works. The challenge faced in all cases is catastrophic forgetting [102], i.e., the tendency of a neural network to underfit past data when new ones are ingested. The effect of catastrophic forgetting can be alleviated by (1) increasing the model capacity to accommodate new knowledge, (2) storing exemplars of past classes in a bounded memory and replaying them in each new state, (3) using knowledge distillation to encourage the network to keep the same outputs for past classes in future states, or by (4) combining two or all of these components. Continual and lifelong learning algorithms usually increase model capacity and are tested in a setting in which a new task is added in each new state of the system. Most of the recent comparative studies [76, 116] provide good coverage of these two types of approaches, but few of them [101] give room to class incremental learning algorithms. Consequently, we focus on representative recent works from literature which tackle class IL [24, 48, 47, 53, 128, 174]. Their study is interesting because one early example of such work [128] is shown to outperform continual learning approaches when tested in a common experimental setting [76], while more recent works [24, 53, 174] have provided strong improvements compared to [128]. We propose in Section 2.2 a unified formulation of the class-incremental learning problem and use it to analyze different algorithms. The same formulation is subsequently used to present our methods in Chapters 3 and 4. In Sections 2.3.1, 2.3.2, and 2.3.3, focus is put on the components which differentiate algorithms one from another in order to facilitate the understanding of their advantages and limitations. Finally, we conclude the chapter by providing a global assessment of the characteristics of each group of IL methods.

2.2 Problem Formulation

We propose a unified formulation of class-incremental learning which builds on those introduced in [86, 128] and [24]. We note $\mathcal{T}$ the total number of states. The first one is called the initial state, and the $\mathcal{T}-1$ remaining states are incremental. A set of P_t new classes is learned in the t^{th} state. States do not overlap in classes, i.e., a class is initially learned with all its data only once during all the training process. A model $\mathcal{M}_1$ is initially trained on a dataset $\mathcal{X}_1 = \{X_1^1, X_1^2, ..., X_1^j, ..., X_1^{P_1}\}$, where P_1 is the number of classes learned in the first state. $X_t^j = \{x_j^1, x_j^2, ..., x_j^{n_j}\}$ is the set of n_j training examples for the j^{th} class, p_t^j is its corresponding classification probability in the state $\mathcal{S}_t$. We note N_t the set of all classes seen until the t^{th} state included. Thus, $N_1 = P_1$ initially, and $N_t = N_{t-1} + P_t = P_1 + P_2 + ... + P_{t-1} + P_t$ for subsequent states. $\mathcal{M}_t$ is updated with an IL algorithm $\mathcal{A}$ using $\mathcal{X}_t = \{(X_t^j, Y_t^j) : j \in [N_{t-1}, N_t]\}$ if no memory of the past is allowed and using $\mathcal{X}_t \cup \mathcal{K}$ if a past memory $\mathcal{K}$ is allowed. At each state, $\mathcal{M}_t$ is evaluated on all classes seen so far ($j \in [1, N_t]$). We mark in bold vectors and matrices.

Following [128, 24], we use in this thesis $P_1 = P_2 = ... = P_{\mathcal{T}}$. Some recent works [88, 53, 35, 89] train the first model $\mathcal{M}_1$ on half of the available classes, and divide the remaining classes evenly in the subsequent $\mathcal{T}-1$ states. The first group of methods tackle more challenging scenario of IL since the initial representation is learned with few classes.

Recent class-incremental learning algorithms are implemented using deep convolutional networks (DNNs) as backbone [128, 24]. While DNNs are end-to-end classification approaches, a part of the IL algorithms uses a separate classifier layer. In such cases, the model $\mathcal{M}_t$ includes two main components: a feature extractor $\mathcal{F}_t$ and a classification component $\mathcal{C}_t$. We note $\theta_t = \{\phi_t, \psi_t\}$ the set of all network parameters in state $\mathcal{S}_t$.

The feature extractor $\mathcal{F}_t$ is parameterized by weights ϕ_t and is defined as:

$$\begin{aligned} \mathcal{F}_t : X_t &\rightarrow \mathbb{R}^D \\ x &\mapsto \mathcal{F}_t(\phi_t; x) = \boldsymbol{f}(\boldsymbol{x}) \end{aligned} \tag{2.1}$$

where $\boldsymbol{f}(\boldsymbol{x})$ is a D-dimensional compact vectorial representation of the image x, called feature vector. Its size depends on the deep architecture. For example, for a ResNet-18 [51], $D = 512$. We note F_t^j the set of feature vectors of all images of a class j extracted with $\mathcal{M}_t$.

The classifier $\mathcal{C}_t$ is parameterized with weights ψ_t, and is usually defined as:

$$\begin{aligned} \mathcal{C}_t : \mathbb{R}^D &\rightarrow \quad\quad\quad\quad \mathbb{R}^{N_t} \\ \boldsymbol{f}(\boldsymbol{x}) &\mapsto \quad \mathcal{C}_t(\psi_t; \boldsymbol{f}(\boldsymbol{x})) = \boldsymbol{f}(\boldsymbol{x}) \times \boldsymbol{\mathcal{W}_t} + \boldsymbol{\mathcal{B}_t} = \boldsymbol{o_t} \end{aligned} \tag{2.2}$$

where:

— $\boldsymbol{o_t} = (o_t^1, o_t^2, ..., o_t^{N_t})$ is the vector of raw scores of size N_t providing the individual prediction scores for each class $j = 1, 2, ..., N_t$. We omit the x for simplicity.

— $\psi_t = \{\boldsymbol{\mathcal{W}_t}, \boldsymbol{\mathcal{B}_t}\}$

— $\boldsymbol{\mathcal{W}_t}$ and $\boldsymbol{\mathcal{B}_t}$ are the weights matrix and bias vector of the last fully connected layer of size (D, N_t) and N_t respectively.

$\boldsymbol{\mathcal{W}_t}$ is defined as:

$$\boldsymbol{\mathcal{W}_t} = \{\boldsymbol{W_t^1}, .., \boldsymbol{W_t^{N_0}}, \boldsymbol{W_t^{N_0+1}}, .., \boldsymbol{W_t^{N_1}}, ..., \boldsymbol{W_t^{N_{t-2}+1}}, .., \boldsymbol{W_t^{N_{t-1}}}, \boldsymbol{W_t^{N_{t-1}+1}}, .., \boldsymbol{W_t^{N_t}}\} \tag{2.3}$$

$\mathcal{C}_t$ can also be implemented using an external classifier. For instance, $iCaRL$ [128] and a variant of $LUCIR$ [53] use a Nearest-Class-Mean (NCM) as a classifier to compute class predictions based on the similarity of each test image to the average class feature computed from the available images in each state. Yet another choice [12] is to exploit a fixed deep representation to extract features for all incremental states and a set of linear SVMs to implement the classifier $\mathcal{C}_t$.

In a DNN [51], a softmax function σ is applied to transform raw scores $\boldsymbol{o_t} = (o_t^1, o_t^2, ..., o_t^{N_t})$ to probabilities $\boldsymbol{p_t} = (p_t^1, p_t^2, ..., p_t^{N_t})$ using Equation 2.4.

$$p_t^j = \sigma(o_t^j) = \frac{e^{o_t^j(x)}}{\sum_{l=1}^{N_t} e^{o_t^l(x)}} \tag{2.4}$$

The different types of final layers mentioned above will be compared in a common setting (Chapters 3 and 4) to assess their merits and limitations.

2.2.1 Past-class exemplars memory

The memory $\mathcal{K}$, which stores a partial view of past classes, is a central component of a majority of existing IL algorithms. The imbalance between past and new classes makes the model biased toward new classes and leads to an underfitting of past classes, a well-known effect of catastrophic forgetting [102]. Two types of memories are used by the community:

— **Growing memory -** [89, 56, 163, 35, 53] here, the imbalance in the number of images between past and new classes is stable through the incremental process. At the end of each state $\mathcal{S}_t$, the same number of images is kept for each class in the memory. The advantage

of this method is to keep the representation of past classes balanced, but it increases memory footprint linearly.

— **Bounded memory -** [89, 128, 24, 174, 189] here, the memory capacity is limited. At the end of each state $\mathcal{S}_t$, the number of images for each past class is reduced to accommodate images from new classes. Thus, the same number of images $|\mathcal{K}_t^j| = |\mathcal{K}|/N_t$ is kept for each class. The representation of past classes is thus degraded and the imbalance in favor of new classes grows throughout incremental states since the bounded memory $\mathcal{K}$ needs to be allocated to a larger number of past classes each time.

On the one hand, having a growing memory is challenging at the beginning of the incremental process where early classes cannot benefit from the full memory size. The scalability of this approach depends on the available memory resources. On the other hand, having a bounded memory helps early classes benefit from the full memory size, but this is problematic at a large scale where more and more classes do not have representatives for them in the memory. The scalability of this approach depends on the robustness of the class-incremental learning method used.

To make the class IL more realistic and challenging, we focus on scenarios where the memory of the past is bounded. There are many ways to select images to fill the memory, but few of them were studied more attentively:

— **Random selection -** consists in shuffling and selecting random images for each class. This method is intuitive and fast but remains a strong baseline.

— **Selection by herding -** it consists in selecting the most representative set of images for each class heuristically. Here, the order of exemplars in the memory matters because the less important exemplars are removed when space is needed in the memory. Herding is computationally expensive compared to a random selection.

— **Optimizable exemplars -** this method is explored in $Mnemonics$ [89]. Here the exemplars are parameterized and thus optimized in an end-to-end manner. At each new state, new exemplars are learned for new classes, and those of past classes are adjusted to fit the current data distribution. This method provides flexible and adaptable exemplars set but is computationally very expensive compared to the above methods.

The role of exemplar selection techniques is debated in literature [24, 89, 127, 174]. Notably, the authors of [24] and [89] report similar results with herding-based and random selection of exemplars. However, both of these works select exemplars statically using their similarity to the mean feature vector of the class. The original definition from [172] (also exploited in

[128]) selects each exemplar based on a dynamic mean computed at each step of the exemplars which were already selected. The advantage of the original definition is that it provides a better approximation of the actual class center compared to a static selection.

We run extensive experiments in Chapter 3 with both random and herding methods and show that herding is beneficial to improve incremental learning. The authors of [101] reach the same conclusion for long sequence tasks. Note that we also implemented the version of herding proposed in [24] and initial tests showed that its performance is indeed comparable to that of random sampling. We also tried other sampling strategies inspired from active learning approaches such as: entropy [148] and min margin [146] (also explored in *RWALK* [26]), core-set [144], and k-means [184]. Initial experiments showed that none of these techniques provide better performance than herding.

2.2.2 Knowledge distillation

Knowledge distillation (KD) [52] was originally designed to help a *student* model learn from output activations of a *teacher* model. It was later adopted in class-incremental learning by [86] to prevent the drift in the representation learned by the network while updating the model with data from new classes. The authors introduced a distillation term in the loss function with Equation 2.5

$$\mathcal{L} = \lambda \ \times \mathcal{L}^c + (1 - \lambda) \ \times \mathcal{L}^d \tag{2.5}$$

where $\mathcal{L}^c$ and $\mathcal{L}^d$ are classical cross-entropy and distillation terms, respectively. $\lambda \in [0, 1]$ is a hyper-parameter that provides the weight of each loss term.

Cross-Entropy Loss

In state $\mathcal{S}_t$, the cross-entropy (or classification) loss is computed for all past and new classes and is given by :

$$\mathcal{L}_t^c(x, y; \theta_t) = \sum_{(x,y) \ \in \ \mathcal{D}_t \cup \mathcal{K}} \sum_{j=1}^{N_t} -\delta_{y=j} \ log[p_t^j(x)] \tag{2.6}$$

where δ is the Kronecker delta.

Distillation Loss

In state $\mathcal{S}_t$, distillation loss is computed for past classes only and is given by:

$$\mathcal{L}_t^d(x) = \sum_{(x,y) \in \mathcal{D}_t \cup \mathcal{K}} \sum_{j=1}^{N_{t-1}} -\hat{\sigma}_{t-1}^j(x) \; log[\hat{\sigma}_t^j(x)] \tag{2.7}$$

where $\hat{\sigma}$ is the softened softmax applied on the raw scores predicted by the network. The softened score of the j^{th} class in state $\mathcal{S}_t$ is:

$$\hat{\sigma}_t^j(x) = \frac{e^{o_t^j(x)/T}}{\sum_{l=1}^{N_t} e^{o_t^l(x)/T}} \tag{2.8}$$

Compared to the softmax definition provided in Equation 2.4, a temperature value T is used for *temperature scaling* [52], that prevents occurrence of very high probabilities. Note that σ is equivalent to $\hat{\sigma}$ when $T = 1$.

$\mathcal{L}^d$ was originally introduced to improve performance when no memory of past classes is available [86]. The authors of [127] adapted it for the case when a bounded memory $\mathcal{K}$ is allowed. Different flavors of distillation were later proposed in [24, 53, 61, 174] in order to further improve the compromise between model stability and plasticity in class IL. We ran experiments with different formulations of distillation and report the main findings in the next chapters.

2.3 Class-incremental learning methods

After defining some fundamental concepts in class-incremental learning, we present the most popular methods from the state of the art in the following sections. Inspired by [76], we categorize these approaches in three main groups as shown in Figure 2.1, and we discuss approaches from each category hereafter.

Note that the categorization proposed in Figure 2.1 is not strict in the sense that an IL approach can belong to one or many categories. For instance, [174] belongs to both bias removal and regularization-based approaches. Alternatively, [66] belongs to both fixed-representation and pseudo-replay-based approaches.

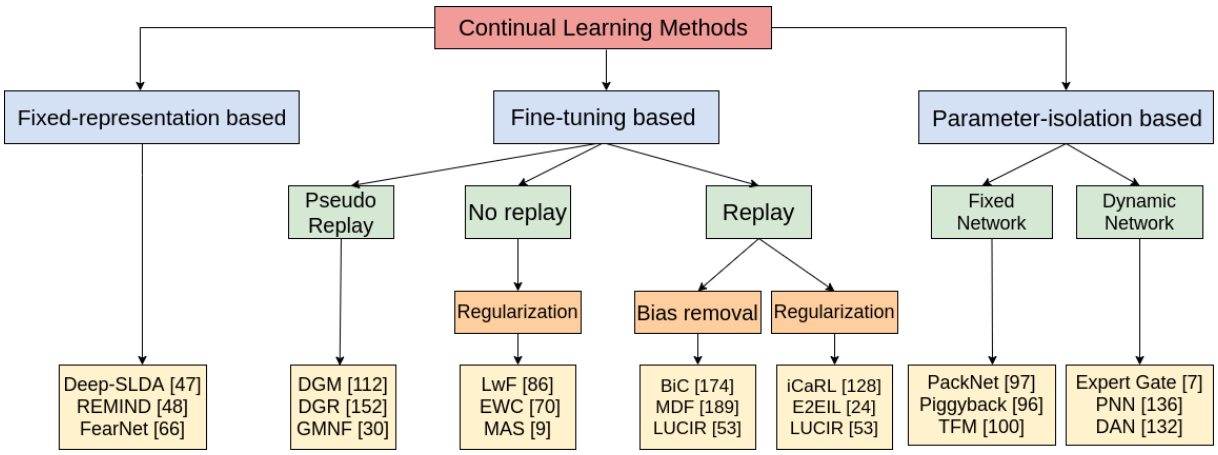


Figure 2.1 – State-of-the-art approaches divided into three main categories: (1) Fixed-representation based approaches where the model does not evolve through incremental states, (2) fine-tuning based approaches where the model is updated with new classes data and possibly with a pool of exemplars from past classes, and (3) parameter-isolation based approaches that either compress the network or expand its size in order to accommodate new classes. *Best viewed in color.*

2.3.1 Fine-tuning based approaches

They constitute the most popular type of approaches. Fine tuning consists of initializing the weights of the model with those of the previous one to benefit from previously learned knowledge and quickly start the learning process on new data. Fine-tuning-based approaches are the most popular in class-incremental learning. We distinguish between three main categories: (1) replay-free approaches that do not use a memory of the past, (2) replay-based approaches that use a bounded memory of the past, and (3) pseudo-replay-based approaches that generate past images instead of storing them in the memory.

2.3.1.1 Replay-free methods

This group of approaches can operate without the use of past exemplars memory. They are often based on regularization losses. We can distinguish between two categories of this group (that we do not show in Figure 2.1 for simplicity): (1) methods that aim to prevent *weight drift* of weights that are important to previous classes (known as *weight regularization methods*), and (2) methods that prevent *activation drift* that is more remarkable in the classification layer (known as *data regularization methods*).

Weight regularization methods

They compute the importance of each parameter in the network after each state. When learning the new set of classes, the more a weight is important, the more the network is penalized for its changes. Such works add an additional loss term (Equation 2.9).

$$\mathcal{L}_t^r(x, y; \theta_t) = \frac{1}{2} \sum_{w=1}^{|\theta_{t-1}|} \Omega_w (\theta_{t-1}^w - \theta_t^w)^2 \tag{2.9}$$

where $|\theta_{t-1}|$ is the number of parameters of the model $\mathcal{M}_{t-1}$, and Ω_w is the importance of the weight w.

Class IL methods in this category differ in how they compute the importance of the weights Ω. *Elastic Weights Consolidation (EWC)* [70] computes Ω as a diagonal approximation of the *Fisher Information Matrix (FIM)* , while in [87] authors rotate the parameter space in order provide a better approximation of the FIM. Alternatively, [81] build on continual learning approaches that use batch normalization layers and propose a quadratic penalty method with a Hessian approximation. Contrarily to [70, 87], authors approximate the FIM using a Krenocker factorization.

$PathInt$ [185] is another variant that maintains an online estimate of weights through the incremental process while consolidating those having high values when learning new tasks. This approach underestimates weights importance when fine tuning from a pre-trained model, while it overestimates them when learning the first model from scratch.

Memory Aware Synapses (MAS) [9] deploys a mechanism that identifies the most important weights in the model by looking at the sensitivity of the output function instead of the loss. When a new task arrives, changes to important weights are penalized. This method was initially designed to work in a task-incremental scenario but was later adapted for usage in domain-incremental learning [8].

Data regularization methods

They often use a distillation [52] term to reduce catastrophic forgetting [102]. The use of knowledge distillation in an IL context is similar to self-distillation [40, 179] in that it operates with the same network architecture for the teacher and the student. However, a notable difference arises from the fact that new data are incorporated progressively.

Learning without Forgetting (LwF) [86] is a pioneering work that does not require memory of past classes. It leverages knowledge distillation [52] to minimize the discrepancy between representations of past classes from the previous and current IL states. LwF first performs a

warm-up step that freezes the past parameters and trains only the new ones and then jointly trains all network parameters until convergence (Equation 2.7).

Learning without Memorizing (LwM) [34] is a distillation-based approach that does not need a memory of past classes. Instead, the authors propose an information preserving penalty using attention distillation loss that captures the changes in the classifier attention maps in order to preserve past knowledge. The loss function in LwM is defined in Equation 2.10.

$$\mathcal{L}^{LwM} = \mathcal{L}^c + \beta\mathcal{L}^d + \gamma\mathcal{L}^a \tag{2.10}$$

where β and γ are weighting parameters, $\mathcal{L}^c$ is the classification loss, $\mathcal{L}^d$ is the distillation loss, and $\mathcal{L}^a$ is the attention loss. The latter is defined in Equation 2.11:

$$\mathcal{L}^a(x;\theta_t) = \left\| \frac{\boldsymbol{Q_{t-1}(x)}}{\|\boldsymbol{Q_{t-1}(x)}\|_2} - \frac{\boldsymbol{Q_t(x)}}{\|\boldsymbol{Q_t(x)}\|_2} \right\|_1 \tag{2.11}$$

where $Q(x)$ is the attention map of image x generated with Grad-CAM algorithm [143]. Grad-CAM helps to localize the parts of the image that contribute to the prediction.

[186] propose *Deep Model Consolidation (DMC)*, another distillation based system. It trains two separate networks, one for new classes and one for past classes, and then combines them via a double distillation loss $\mathcal{L}^{dd}$. The latter helps to resolve the asymmetry in predictions between past and new classes. Here, an exemplars memory is not needed, but it is replaced by an unlabeled auxiliary data $\mathcal{X}^u$ to perform deep memory consolidation (Equation 2.12).

$$\mathcal{L}_t^{dd}(x_u;\theta_t) = \frac{1}{N_t}\sum_{j=1}^{N_t}(o_t^j(x_u) - \hat{o}_t^j(x_u))^2 \tag{2.12}$$

where x_u is an unlabeled example and $\hat{o}$ is the normalized version of o (Equation 2.13):

$$\hat{o}_t^j(x_u) = \begin{cases} o_{t-1}^j(x_u) - \frac{1}{N_{t-1}}\sum_l^{N_{t-1}} o_{t-1}^l(x_u) & \text{if } j \in [1, N_{t-1}] \\ o_t^j(x_u) - \frac{1}{N_t}\sum_{l=1}^{N_t} o_t^l(x_u), & \text{if } j \in]N_{t-1}, N_t] \end{cases} \tag{2.13}$$

Semantic Drift Compensation (SDC) [154] was proposed to estimate the semantic drift of past knowledge while learning new knowledge to compensate for it, to further improve performance. The drift is computed at the class-mean-embedding level. This approach is based on an NCM classifier that does not need exemplars storage since the past class-mean embeddings are estimated using new data only.

In [154], authors propose an approach that calibrates activation maps of the CNN in order to accommodate new knowledge. Calibration is done using spatial and channel-wise calibration

modules, and only the calibration parameters are trained at each new incremental state. This method does not require a past-class memory. However, the calibration parameters grow instead.

Recently, [101] proposed a simple yet efficient cross-entropy loss for vanilla fine tuning without memory. Instead of computing the loss on all classes (Equation 2.6), it is computed on scores of new classes only, using Equation 2.14

$$\hat{\mathcal{L}}_t^c(x, y; \theta_t) = \sum_{(x,y) \in \mathcal{D}_t \cup \mathcal{K}} \sum_{j=N_{t-1}+1}^{N_t} -\delta_{y=j} \, log[p_t^j(x)] \tag{2.14}$$

This loss helps back-propagating the error using weights of new classes only. These weights improve the encoding of the representation of the model in the current state where the model is updated with data from new classes.

Very recently, the authors of [56] proposed a novel framework where they compute and distill the *colliding effect* between past and new data. In addition, they propose to capture the *Incremental Momentum Effect* and remove what causes the forgetting of past classes. The proposed method is functional with and without a memory of exemplars. It can be applied on top of unified classifiers such as $LUCIR$ [93] and $PODNet$ [35]. Obtained results are promising, especially without memory. However, it is not clear how calibration parameters were optimized in this scenario.

2.3.1.2 Replay-based methods

Here, we discuss methods that rely on the loss function to regularize the weights update (known as *regularization-based methods*), and methods that address recency bias by removing the bias between past and new classes' scores (also known as *bias-removal-based methods*).

Regularization-based methods

Incremental Classifier and Representation Learning (iCaRL) [128] is a popular IL method that combines the use of distillation and a memory for past class exemplars. Classification is performed with a nearest-mean-of-exemplars method instead of the raw scores predicted by the network. This external classifier is deployed to reduce the prediction bias in favor of new classes, which occurs due to data imbalance between past and new classes. An iCaRL analysis [61] concludes that its most important components are the fixed-size memory and the distillation loss. The authors of [61] claim that the nearest-mean-of-exemplars classification seems to matter less, and they replace it with a *dynamic threshold moving* rule to remove bias generated with distillation loss among past classes. It is important to highlight the fact that the comparison of

$iCaRL$ with a vanilla fine tuning was not fair in [128] insofar as $iCaRL$ was implemented with a bounded memory of the past. In contrast, fine tuning did not use such a memory. We show in Chapter 3 that when a memory of the past is allowed, a vanilla fine tuning outperforms $iCaRL$ on large-scale datasets.

The authors of [24] present *End-to-End incremental learning (E2EIL)*, an IL algorithm that differs from $iCaRL$ mainly through the way prediction bias is reduced. The external classifier is replaced by a balanced fine-tuning step, where data from new classes is reduced to have equally distributed batches between past and new classes. This component has an important impact on performance and leads to a strong improvement compared to $iCaRL$. It was later adopted in [53] for its efficiency to tackle task imbalance during training. Sophisticated data augmentation was also used by the authors and has a small positive influence on results.

Many recent IL approaches focus on a more sophisticated tackling of catastrophic forgetting. The authors of *Multi-model and Multi-level Knowledge Distillation (M2KD)* [193] propose a loss that distills knowledge not only from the previous model but from all the past models where the classes have been learned for the first time. They also propose an additional distillation term that operates on the intermediate layers of the CNN in addition to the last fully connected one. The drawback of this method is that it requires saving all previous models, a strategy that is heavy at a large scale.

In [176], knowledge distillation is also combined with a bounded memory of the past. The authors deploy an algorithm to set a dynamic vector that corrects the bias induced by distillation loss among past classes and improves the representativeness of past image features.

Hou et al. [53] present *Learning a Unified Classifier Incrementally via Rebalancing (LUCIR)*, a method that gains much traction. $LUCIR$ is based on three main components: (1) *cosine normalization*, (2) *inter-class separation*, and (3) *less-forget constraint*. The first two contributions help reducing *recency bias* and are thus presented in the related section. The *less-forget constraint* from $LUCIR$ is defined in Equation 2.15:

$$\mathcal{L}_t^{lf}(x;\theta_t) = 1 - \frac{\langle \boldsymbol{o_{t-1}}(\boldsymbol{x}), \boldsymbol{o_t}(\boldsymbol{x}) \rangle}{\|\boldsymbol{o_{t-1}}(\boldsymbol{x})\|_2 \, \|\boldsymbol{o_t}(\boldsymbol{x})\|_2} \tag{2.15}$$

where $\langle ., . \rangle$ is the inner product (cosine similarity) function. The other two components of $LUCIR$ are described later in this chapter.

Further improvements of IL with distillation were obtained by adapting recent theoretical and empirical advances such as those described in [117] and [121]. For instance, $PODNet$ [35] relies on a spatial-based distillation loss that constrains the evolution of the representation of the model and multiple proxy vectors to flexibly represent learned classes. This approach is more

suitable for long runs of small incremental tasks.

The behavior of knowledge distillation was studied in [25]. The authors observed that KD works better when the *domain shift* is low, as is the case for class-incremental scenarios. However, it hurts performance when the domain shift is high, as is the case for task-incremental learning approaches, where each task often contains classes from a different domain.

Similarly to DMC [186], *Global distillation (GD)* [82] takes advantage of an external dataset to tackle catastrophic forgetting using triple distillation. This method has three main stages. First, a teacher model is trained on new data, then calibrated using the exemplars memory and the external dataset. Second, a triple distillation is applied using the teacher model, the previous model, and the ensemble model (used with external data only). Finally, a fine tuning is performed to tackle the recency bias problem. Alternatively, [54] propose an approach that trains a teacher model separately with new data while distilling knowledge using exemplars memory to preserve accuracy on past classes. The main difference with *Global Distillation* [82] is the absence of the external dataset.

Less-forgetting Learning (LFL) [64] freezes the last layer while penalizing the changes in activations of previous layers. This approach is problematic when the *domain shift* is large.

Riemanian Walk (RWALK) [26] is a regularization-based approach that does not use distillation. Instead, it combines $PathInt$ [185] and EWC [9] with an exemplar memory. Based on a KL-divergence, this method aims to make a compromise between *forgetting* and *intransigence* of the network.

Mnemonics Training [89] is built on top of herding-based approaches such as $iCaRL$ and $LUCIR$ to modify the herding procedure by parameterizing exemplars and making them optimizable. The network is then optimized in two manners: model-level and exemplar-level. The memory is thus adjusted incrementally to match the data distribution effectively, leading mnemonic exemplars to yield separation between classes.

Recently, feature transformation using a dedicated MLP is introduced in [60]. This approach only stores features instead of images to reduce the memory footprint of exemplars. Similar to $LUCIR$, this method applied features distillation combined with cosine normalization.

The authors of *Greedy Sampler and Dumb Learner (GDumb)* [122] question the generality of continual learning approaches. Authors claim that current CL approaches use simplifying assumptions that make CL unrealistic. They propose a simple algorithm that greedily samples balanced stored exemplars and trains the neural network from scratch using the exemplars only. $GDumb$ surprisingly outperforms most recent works from the state of the art when tested on a wide variety of CL scenarios. This work questions the real advances in continual learning

regarding the proposed methods, evaluation metrics and protocols.

Bias-removal-based methods

These methods tackle the *recency bias* problem that refers to the bias of the network towards classes of newer states [101]. Recent works [189, 174] hypothesize that due to the limited memory of the past, IL is akin to imbalanced learning where the network is trained with enough images for the new classes but only a few ones for past classes.

These works tackle catastrophic forgetting as an imbalance problem. *Bias Correction (BiC)* [174] is a recent approach that uses a classical knowledge distillation term and adds a linear layer after the prediction layer of the deep model to reduce the bias in favor of new classes. The new classes' outputs are rectified with Equation 2.16

$$BiC(\boldsymbol{o_t^k}) = \alpha_t \boldsymbol{o_t^k} + \beta_t \cdot \mathbf{1} \tag{2.16}$$

where $\boldsymbol{o_t^k}$ are the raw scores of new classes, α_t and β_t are the bias correction parameters, and $\mathbf{1}$ is a vector of ones.

The method needs a validation set to learn parameters and is effective as long as the size of the validation set is sufficient. We study this method in more detail and propose an improved version of it in Section 4.3 of Chapter 4.

Maintaining Discrimination and Fairness (MDF) [189] uses distillation loss to maintain discrimination between past classes. In addition, the rectification of class scores is done by aligning new class weights to those of past classes by multiplying each new class weight by the mean norm of past class weights and dividing it by the mean norm of new class weights, before to finally compute the prediction scores.

The authors of $LUCIR$ [53] propose, in addition to the less-forget constraint defined above, two techniques to tackle recency bias: (1) *cosine normalization* to balance the magnitudes of past and new class probabilities, and (2) *inter-class separation* to encourage the network to separate past and new class embeddings and actually implements a theoretical finding from [121].

The cosine layer replaces the last fully connected layer and is defined in Equation 2.17:

$$\mathcal{L}_t^{cos}(x, \theta_t) = \sum_{j=1}^{N_t} y_j \, log \frac{exp(\eta \langle \frac{\boldsymbol{f(x)}}{\|\boldsymbol{f(x)}\|_2}, \frac{\boldsymbol{W_t^j}}{\|\boldsymbol{W_t^j}\|_2} \rangle)}{\sum_{l=1}^{N_t} exp(\eta \langle \frac{\boldsymbol{f(x)}}{\|\boldsymbol{f(x)}\|_2}, \frac{\boldsymbol{W_t^l}}{\|\boldsymbol{W_t^l}\|_2} \rangle)} \tag{2.17}$$

where $\boldsymbol{f(x)}$ is the feature vector of image x, $\langle \cdot, \cdot \rangle$ is the cosine similarity, $\boldsymbol{W_t^j}$ is the classifi-

cation weight vector for the j^{th} class (called class embedding in this thesis), and η is a learnable parameter that controls the peakiness of the softmax distribution.

The *inter-class separation* is implemented through a *margin ranking loss* that separates the current feature vector with the feature vectors of the J most similar classes using Equation 2.18:

$$\mathcal{L}_t^{mr}(x) = \sum_{j=1}^{J} max\left(m - \langle \frac{\boldsymbol{f}(\boldsymbol{x})}{\|\boldsymbol{f}(\boldsymbol{x})\|_2}, \frac{\boldsymbol{W}^{\boldsymbol{y}}}{\|\boldsymbol{W}^{\boldsymbol{y}}\|_2} \rangle + \langle \frac{\boldsymbol{f}(\boldsymbol{x})}{\|\boldsymbol{f}(\boldsymbol{x})\|_2}, \frac{\boldsymbol{W}_t^{\boldsymbol{j}}}{\|\boldsymbol{W}_t^{\boldsymbol{j}}\|_2}, 0\right) \tag{2.18}$$

where m is the margin, $\boldsymbol{W}^{\boldsymbol{y}}$ is the ground truth class embedding of x, and $\boldsymbol{W}_t^{\boldsymbol{j}}$ is the embedding of the closest class j. Note that $LUCIR$ belongs to both *data-regularization-based* (for the less-forget constraint) and *bias-removal-based* approaches (for the cosine layer).

Bias-removal-based approaches provide the best results in the literature, especially when combined with a bounded memory of the past [101].

2.3.1.3 Pseudo-replay-based methods

These approaches do not store exemplars for past classes in the memory. Instead, they generate synthetic data to represent past classes in the current incremental state.

Using *Generative Adversarial Networks (GANs)* to generate past data holds promise since it reduces the memory footprint of algorithms. However, despite recent progress [43], generated images are still sub-optimal for IL. Since additional GAN models need to be created, the complexity in the number of parameters is fair but not optimal. The authors of [173] use a GAN to create artificial images for past classes. Generated and real examples are mixed to obtain slightly better performance than that of $iCaRL$ [128]. However, the accuracy drops significantly when relying exclusively on artificially generated images. Alternatively, *GAN Memory with No Forgetting* [30] is based on sequential style modulations to represent the past memory by forming a sequential targeted generative model. Here, the memory itself is designed as a form of lifelong learning.

In [112], authors propose *Dynamic Generative Memory (DGM)*, a continual learning framework that relies on an adversarial generative network with learnable plasticity between connections. Plasticity is represented by a parameter-level attention mechanism. Instead of storing images of past classes, DGM uses previous distributions to incrementally learn a single generator of past classes images.

Alternatively, [175] propose to use conditional generative adversarial networks (CGANs) to tackle class IL. The generator is conditioned on embeddings of past classes. Besides, it generates

pseudo exemplars in the form of perceptual convolutional features. The discriminator contains two heads and produces normalized embeddings that are discriminative for multi-category classification and indiscriminative for true and false example identification. The generator and the discriminator are trained alternatively for continuous incremental learning.

Recently, [157] proposed a hybrid approach that combines replay and pseudo-replay methods to tackle catastrophic forgetting. Authors store a small number of exemplars from the past and generate variations of them to be replayed in addition to new classes data. The main conclusion derived by the authors is that combined replay is efficient, especially for tiny memory buffers.

2.3.2 Fixed-representation based approaches

Fixed-Representation (FR) based methods do not update the deep representation for each incremental state and are less numerous in literature. They can be seen as a basic variant of fine-tuning-based methods. A fixed-representation method is briefly described in [128] and the results reported with it are poor. The experiments from the next chapters show that the finding from[128] is due to suboptimal usage of the method. In particular, the classification layer for past classes is needlessly relearned in each incremental state using only the exemplars of past classes. Since the representation is fixed, the stronger classifier weights learned initially with all past class data are reusable.

FearNet [66] is a biologically inspired fixed-representation method. Separate networks are used for long- and short-term memories to represent past and new classes. A decision mechanism is implemented to decide which network should be used for each test example. $FearNet$ is interesting, but its memory increases significantly with time since the algorithm stores detailed statistics for each class learned.

Deep Streaming Linear Discriminant Analysis (Deep-SLDA) [47] is an online approach based on SLDA [114] algorithm. The Network is trained on the first batch of classes and is frozen afterward. During training, a class-specific running mean vector and a shared covariance matrix are updated. The prediction is done by assigning the label to the closest Gaussian in the feature space defined by the class-mean vectors and covariance matrix.

REplay using Memory INDexing (REMIND) [48] is brain-inspired by the hippocampal indexing theory. The method is also based on an initial representation which is only partially updated afterward. The approach uses a vector quantization technique to store compressed intermediate representations of images, which are more compact than images themselves. The stored vectors are reconstructed and replayed for memory consolidation. $REMIND$ is designed for

online learning, where each training example is seen once during all the training process. We evaluate it in this thesis following a class-incremental protocol. Note that vector quantization is widely used in unsupervised incremental learning [94]. Here, the authors combine the *Adaptive Resonance Theory (ART)* with a variant of vector quantization to make a trade-off between *plasticity and stability* [104] during incremental online learning. This approach was designed to handle two- and high-dimensional data within the image classification framework. We tackle in this thesis supervised learning, and this type of approaches is not compatible with our experimental protocol.

2.3.3 Parameter-isolation based approaches

We distinguish between two types of approaches: (1) those where the architecture of the model grows to accommodate new knowledge (*dynamic networks*), and (2) those where the complexity of the model is constant (*fixed networks*).

2.3.3.1 Dynamic networks

Dynamic networks increase the size of deep models to include new knowledge. Wang et al. [170] introduced *Growing a Brain*, a method based on increasing representational capacity by widening or deepening the network. *Progressive Neural Networks (PNN)* [136] is an alternative approach that exploits several models during training to preserve knowledge from past tasks. Lateral connections between all models are learned to leverage prior knowledge of past features and thus reduce the effect of catastrophic forgetting. Recently, [133] propose an adaptive network that enables self-growth in a tree-like manner. It is based on features hierarchy reorganization whenever new tasks arrive.

Aljundi et al. [7] present a lifelong learning architecture based on a network of experts. A gating mechanism exploits training samples for each task to decide which expert to trigger at inference time. *Deep Adaptation Networks (DAN)* [132] is another model-growth based approach which adds additional parameters for each new task. The architecture is importantly augmented if a large number of new tasks arrives. The approach presented in [126] is based on several neural networks that share the majority of parameters and add modular adapters to connect the networks and specialize each one of them for a specific task.

Dynamically Expandable Networks (DEN) [181] automatically split and duplicate neurons as more tasks are encountered. Depending on the similarity between past and current tasks, the network might be expanded or not. The advantage of this approach is its applicability to

whatever generic neural network, including CNNs.

Self-Organizing Maps (SOMs) are online unsupervised learning algorithms that rely on approximate stochastic gradient techniques and can be adapted to class-incremental learning. *Neural Gas (NG)* networks [99] and its growing NG variant [39] are related to SOMs which are often exploited for incremental learning. *PROjection-PREdiction (PROPRE)* [42] is an incremental learner based on NG and SOMs. It implements an extra supervised read-out layer implemented as linear regression and a concept drift detection mechanism to make the SOM usable in an IL context.

NG and SOM are growing networks that were widely used for incremental semi-supervised clustering [39], multi-class online classification problems [11], and online semi-supervised vector quantization learning [151]. They thus need an adaptation to be used in class-incremental learning. *TOpology-Preserving knowledge InCrementer (TOPIC)* [162] is a recent work that exploits NG for class-incremental learning for visual datasets with a focus on few-shot learning. First, $TOPIC$ introduces an NG network to learn feature space topologies for knowledge representation. The network grows to learn new classes while also dealing with changes in the feature space due to deep model updates. This is achieved using a *min-max* loss that pushes new classes which share the same label to a new NG node while pulling the new nodes of different labels away from each other. Second, $TOPIC$ preserves past knowledge by stabilizing the topology of the NG network using an *anchor loss* term. Since $TOPIC$ focuses on the feature space, which encodes more semantic information than the raw classification scores, it is less affected by the bias induced by high new classes' raw scores. A topology-preserving network named $TPCIL$ is introduced in [163] to handle catastrophic forgetting. The network models the feature space using an Elastic Hebbian Graph, and the topology is maintained using a topology-preserving loss that constrains the neighborhood relationships in the graph when learning new classes. This approach augments the Hebbian graph by inserting vertices for each new class. The addition of nodes in $TOPIC$ and $TPCIL$ gradually increases the complexity of the architecture.

2.3.3.2 Fixed networks

Contrarily to dynamic networks, these methods do not modify the overall architecture of the model. $PackNet$ [97] is based on a pruning technique that identifies redundant parameters and uses them to train the network on new tasks. The approach cannot learn a large number of tasks since the network can not be strongly compressed without significant performance loss. $Piggyback$ [96] is a modified version of $PackNet$ that exploits network quantization to propose

masks for individual weights. It thus learns a large number of tasks with a single base network. The approach increases the model complexity because extra parameters are added each time to include new tasks.

Recently, [100] proposed *Ternary Feature Masks (TFM)*. This task-incremental method applies ternary masks to features of each layer in order to allow features to be used or not during the forward and backward pass. The authors also use a task-specific feature normalization to adjust previously learned features to better learn future tasks. TFM has a negligible memory overhead (3 bits per feature per task) but allows training only a fixed number of tasks since the neuron masks are disjoint between tasks.

Alternatively, *Adaptive Aggregation Networks (AAN)* [88] handle catastrophic forgetting at the network level. Stable and plastic blocks are built on top of residual blocks of a ResNet architecture [51] to preserve past knowledge and integrate new data. This approach makes use of exemplars memory, making it part of replay-based methods also.

In the same scope, [168] recently proposed a feature map transformation strategy with negligible additional network parameters to improve class separability. Model parameters are shared between global and task-specific parameters, and only the latters are updated at each IL state to improve training times.

2.4 Class-incremental learning evaluation

2.4.1 Datasets

Many datasets have been used by the community for class-IL, among them:

— **MNIST** [78] is designed for hand-written digit recognition. It contains 10 classes with 60000 training images and 10000 testing images.

— **CIFAR-10** [74] is designed for object recognition and includes 10 classes. Each class has 6000 training images and 1000 testing images.

— **CIFAR-100** [74] is the same than CIFAR-10, but it contains 100 classes regrouped in 20 categories. Each class has 500 training images and 100 testing images.

— **ILSVRC** [135] is a subset of 1000 leave classes from ImageNet [32] database. ILSVRC contains around 1.2 million images and is used in ImageNet challenges. The latter database follows WordNet hierarchy, thus, leave classes represent specific visual concepts.

— **Mini-ImageNet** is a randomly selected subset of 100 classes from ILSVRC [135].

— **VGGFACE2** [22] is a face recognition dataset including over 9000 identities in its full version.

— **Google Landmarks** [108] (**LANDMARKS** below) is a landmark recognition dataset which includes over 2 million images for over 30000 landmarks across the world.

— **FOOD-101** [20] is a dataset for fine-grained food recognition with 101 classes.

— **Caltech-UCSD Birds** [171] (also known as CUBS) contains 200 fine-grained bird classes with a total of 5994 images.

— **Oxford Flowers** [107] contains 102 flower species with 2040 training images and more than 6000 testing images.

— **MIT Scenes** [123] is a dataset for indoor scene classification. It contains 67 classes having each 80 training images and 20 testing images.

— **FGVC-Aircraft** [95] contains around 100 classes with a total of 10200 images.

— **Stanford Cars** [73] contains 196 classes of cars with a total of 16185 images, where the first half are used for training, and the second half is used for testing.

— **VOC Actions** [37] is used in VOC challenge for human action classification. It contains 10 classes of specific actions and one class for other types of actions. It contains more than 6000 multi-labeled images.

— **Street View House Numbers (SVHN)** [106] is designed for digit recognition. It contains more than 600000 images obtained with Google Street View.

— **iNaturalist** [59] contains 8000 fine-grained categories with highly imbalanced classes. This dataset contains about 450000 images and is useful to test the robustness of IL algorithms in real-world settings.

— **Places-365** [192] is a dataset for scene classification. It contains 365 classes with over 1.8 millions of images.

— **WikiArt** [137] artists classification dataset that contains a wide genre of painting styles. It contains more than 80000 images from 1119 artists. This dataset and also **Sketch dataset** [36] were used in [96] but is more suitable for domain-adaptation applications.

MNIST[78] and CIFAR-10 [74] are very small and easy datasets. Thus, we do not include them in our experiments because the focus here is on realistic and challenging datasets with a minimum of 100 classes. Note that Birds [171], Flowers [107], Scenes [123], Aircraft [95], Cars [73], Actions [37], and SVHN [106] are more popular in task-incremental learning, where

the new classes have different visual features each time [9, 7]. Among the mentioned datasets, the most used ones in the class IL are ILSVRC [135], Mini-ImageNet [135], and CIFAR-100 [74]. We are the first to use VGGFACE2 [22] and LANDMARKS [108] as large scale datasets, in addition to ILSVRC. The other datasets that we used include many subsets of 100 classes from ImageNet (different from ILSVRC), Places-365 [192] and FOOD-101 [20].

There exist other datasets that were proposed recently but are used in other continual learning scenarios, such as: Omniglot [140], CORe50 [90], OpenLORIS [150], and Stream-51 [130].

2.4.2 Metrics

Various evaluation metrics have been used in the state of the art. The most important and popular one is accuracy.

— **Accuracy (A) -** two variants of this measure are used: the top-1 accuracy and the top-5 accuracy. The former simply measures the percentage of correctly classified images among all the test set. Practically, the top-1 accuracy means that the target class corresponds exactly to the class predicted with the highest probability. Top-5 is a relaxed version of the former but shares with it the same spirit. The difference is that, in top-5 accuracy, an image is considered as correctly classified if the target class belongs to the five classes predicted with the highest probabilities. The top-5 accuracy was popularized during the *Imagenet LSVRC 2012* challenges [135], because the dataset is challenging and the goal was to relax the objective. Since the model is learned incrementally, the accuracy averaged over incremental states (excluding the first non-incremental one) was proposed by [24] (Equation 2.19).

$$A = \frac{1}{\mathcal{T}-1}\sum_{t=2}^{\mathcal{T}} A_t \tag{2.19}$$

where $\mathcal{T}$ is the total number of states, and A_t is the test accuracy of the model in state $\mathcal{S}_t$. Intuitively, the higher the value of A, the better. Note that some works [128, 174] average the accuracy over all states, including the first.

With the advances in continual learning approaches, other evaluation metrics saw the light but are less used than the accuracy. Among them, we mention:

— **Normalized Accuracy Metric** (Ω) **-** was initially proposed in [67] and used by other

works afterwards [47, 156, 46].

$$\Omega = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \frac{A_t}{A_{Joint,t}} \tag{2.20}$$

where $A_{Joint,t}$ is the accuracy of a training from scratch with all data seen until state $\mathcal{S}_t$ included. The normalized accuracy helps to compare different IL methods across datasets. Note that $\Omega \in [0, 1]$.

— **Forgetting (F) -** As proposed by [89], Forgetting corresponds simply to the difference in accuracy (%) between the last incremental state and the initial state on the test set of the initial state. The lower forgetting rate is better (Equation 2.21)

$$F = A^1_{\mathcal{T}} - A^1_1 \tag{2.21}$$

where $A^1_{\mathcal{T}}$ is the accuracy of the model in the last state ($\mathcal{S}_{\mathcal{T}}$) on classes initially encountered in the initial state, A^1_1 is the accuracy of the model in the initial state on classes learned in this state.

The formula presented in Equation 2.21 involves the first and the last states only. Another formula was proposed in [26], and it takes the average of forgetting over all states. Here, forgetting is defined as the difference between the maximum knowledge acquired when learning a set of classes (or task) in previous states, and the learned knowledge about the same task in the current state (Equation 2.22)

$$F_t = \frac{1}{\mathcal{T} - 1} \max_{k \in [1, \mathcal{T}-1]} (A^t_k - A^t_{\mathcal{T}}) \tag{2.22}$$

where t refers to the set of classes (task) we want to compute forgetting for, and A^t_k is the accuracy of the system on classes of state $\mathcal{S}_t$ in the state $\mathcal{S}_k$.

— **Backward Transfer (BWT) -** measures the influence of training the model in the current state $\mathcal{S}_t$, on the learning of classes seen in previous states ($\mathcal{S}_{k<t}$) [92]. A negative forgetting $F_t < 0$ implies a positive influence on previous classes (also known as Positive Backward Transfer (PBWT)), and vise versa (Negative Backward Transfer (NBWT)).

— **Forward Transfer (FWT) -** measures the influence of learning the current set of classes on training the model on future classes [92]. Ideally, continual learning systems apply both forward and backward transfer. The former aims at using past knowledge to easily learn new classes, and the latter aims at using new knowledge to reinforce previously

learned one [92].

In this thesis, we use top-1, and top-5 accuracy averaged on incremental states, as proposed by [24]. We later propose in Chapter 3, an aggregation accuracy-based method (called G_{IL}) that gives a global idea on the performance of IL algorithms when tested under diversified configurations, which vary in terms of test dataset, number of incremental states, and memory size (when allowed).

2.5 Conclusion

We map each IL group to the six IL properties from Section 1.2 in Table 2.1. We discuss the advantages and/or challenges related to each group-property pair to facilitate their comparison. We also provide a global assessment focusing on the application contexts in which each type of approach could be deployed.

As shown in this chapter and the one before (Chapter 1), the scope of incremental learning in the literature is large. We limit it in this thesis by proposing methods that:

— are designed for class-incremental learning (also called *task-agnostic*) - we discard methods that require task-id at the inference time (task-incremental learning). The goal is to tackle more challenging IL scenarios that are useful in real-life situations.

— add several classes at once - we discard methods that add one to few images per class (domain-incremental learning). These methods constitute a more challenging scenario, and we do not handle them in this thesis.

— keep a constant complexity of deep models - we discard all dynamic networks that increase the architecture of the model upon time

— do not need a large pre-trained model - we suppose that all methods start with the same initial model trained from scratch on the classes of the first state. Comparison between them is thus made on the incremental part only.

Consequently, experiments are conducted with approaches that are fit to work under these conditions, namely those based on fine tuning and fixed representations.

We further investigate the effect of allowing past exemplars memory or not. The majority of works from the state of the art use a memory because it helps to alleviate the effects of catastrophic forgetting considerably. In practice, when the AI system is allowed to store a limited number of past images, the memory can be of great benefit to update its capacities with new data.

	Parameter-isolation based	Fixed-Representation based	Fine-Tuning based
Complexity	In dynamic networks, the model evolves by adding parameters and weights to interconnect them [126, 136, 170] or small networks [7] to include new knowledge. This is not the case for fixed networks [96, 97] that do not augment the architecture of the model. The challenge of dynamic networks is to optimize the effect of model growth on performance [76, 126].	The model is fixed after the first non-incremental state. In a basic setting [127], the only parameters added are those needed for new classes weights. In a more advanced setting [66], additional parameters are needed to improve past classes' performance.	This group of IL methods is designed to work with a fixed structure of the backbone model. The number of parameters is only marginally affected by the modifications of the classification layer designed to reduce the imbalance between past and new classes [24, 53, 174].
Memory	Fixed networks do not require extra memory storage, while growth allows dynamic networks to deploy these methods without using an exemplar memory. Memory is allocated to additional model parameters and weights, which is a more parsimonious way to store information about past classes [9, 7, 133].	Fixed-representations do not update the model during the incremental learning process and thus have a very low dependency on the memory of past classes [47]. Class weights are learned when they are first encountered and can be used throughout all subsequent incremental states.	Performance of these methods is heavily dependent on the size of the past memory. However, storing a large number of past exemplars is contradictory to the IL global philosophy. Memory needs are reduced by exploiting knowledge distillation [24, 53, 61, 174] or by exploiting statistical properties of past states [189].
Accuracy	Performance is correlated with the amount of model growth allowed. If growth is limited, dynamic networks have lower performance compared to that of FT-based ones [76]. If significant growth is allowed [133], performance comes close to that of classical learning, but this is somewhat contradictory to the requirement to keep models' complexity close to constant.	Accuracy is lower compared to FT-based methods because the model is not updated incrementally. High performance can be obtained with fixed representations if the initial model is learned with a large dataset [47]. However, the existence of such a dataset is a strong assumption in incremental scénarios.	Recent approaches report strong performance gain compared to previous work such as [24, 61, 128] either through more sophisticated definitions of knowledge distillation [53] or through the casting of IL as an imbalanced learning problem [21], or a combination of both [174]. The gap with classical learning is narrowed if enough memory of past classes is allowed [53, 174].

	Parameter-isolation based	Fixed-Representation based	Fine-Tuning based
Timeliness	The complexity is generally similar to that of FT-based methods since retraining is needed for each incremental update [76]. For fixed networks, extra time is needed to determine important weights	Only the classifier weights layer needs to be trained, and new knowledge is integrated promptly [47].	New classes are not recognizable until retraining is finished to include them in the model. If applications are time-sensitive, an acceleration of the training process can be envisioned at the expense of result suboptimality.
Plasticity-Stability	Parameter-isolation-based methods are specifically designed to cope with different visual tasks [9, 96]. The challenge is to minimize the number of additional parameters needed to accommodate each new task [76].	Plasticity is limited since the representation is learned in the first state and then fixed. Performance drops significantly if the incremental tasks change a lot and the initial representation is not transferable anymore [125].	The model updates enable adaptation to new data as they are streamed into the system. If no memory is allowed, plasticity is too important, and this shift is controlled through knowledge distillation or imbalance handling.
Scalability	Fixed networks are not scalable. Dynamic networks scale well to new classes or tasks as long as the systems in which they are deployed have sufficient resources to support the underlying model growth for training and inference phases, as well as for its storage.	The dependence on the bounded memory is limited, and FR-based methods can include a huge number of classes. This is possible because class weights are learned in their initial state and reused later.	The size of the bounded memory determines the number of past classes for which exemplars can be stored and which are still recognizable when new ones are integrated. If the system constraints allow for this, the memory can be increased to keep the number of exemplars per class constant [53].

	Parameter-isolation based	Fixed-Representation based	Fine-Tuning based
Global assessment	Approaches in this group cope well with new data, are not or weakly dependent on a memory of the past, but are scalable only if dynamic. However, the complexity of the latter is a disadvantage since the model has to grow in order to integrate new knowledge. They also require retraining when new classes are added and timeliness is not optimal. Dynamic networks are usable when: model complexity can grow during the incremental process, streamed data vary a lot between incremental states, no storage is available for past data, and immediate use of updated models is not essential.	Fixed-representation methods inherit the advantages and disadvantages of transfer learning schemes. Model complexity is constant, and they can be updated promptly since only the classification layer is retrained. They have a low dependency on past memory and can scale up to a large number of classes. However, these algorithms depend heavily on the quality of the initial representation and have low plasticity. They are usable when: model complexity should stay constant, data variability is low, no storage is available for past memory, and updates are needed promptly.	Fine-tuning-based methods are adequate when we try to optimize the architecture complexity, and the plasticity [161] of representations. However, since they require network retraining when new classes are added, their timeliness is not optimal. Equally important, the bounded memory constraint makes them hard to scale because the memory will eventually become too small to represent past classes adequately. They are usable when: model complexity should stay constant, streamed data vary a lot between incremental states, storage is available for past data, and immediate use of updated models is not essential.

Table 2.1 – Analysis of the main groups of class-incremental learning algorithms with respect to their desirable properties. A global assessment with recommended use cases is also provided.

CHAPTER 3

CLASS-INCREMENTAL LEARNING WITH MEMORY

3.1 General Introduction

Class-incremental learning gained lots of attraction in the last decade [24, 48, 101, 128]. Most state-of-the-art approaches use a fixed-size memory of the past to alleviate the effects of catastrophic forgetting. The memory helps "refreshing" the representation of the model by replaying some examples from previous classes.

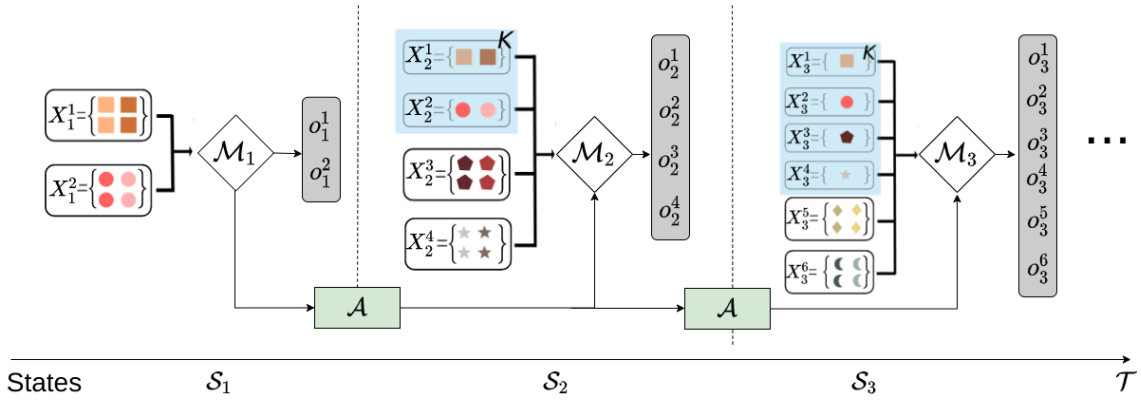


Figure 3.1 – A toy example of class-IL with memory where $\mathcal{T} = 3$ and $|\mathcal{K}| = 4$. $\mathcal{M}_t$ are models, $\mathcal{S}_t$ are states, X_t are sets of images, o_t are raw scores vectors, $\mathcal{A}$ is a class-IL algorithm. *Best viewed in color.*

In this chapter, we focus on a protocol where a bounded memory of the past is allowed. In Figure 3.1, we show a toy example with one initial state $\mathcal{S}_1$ and two incremental states ($\mathcal{S}_2$ and $\mathcal{S}_3$). In the initial state, a simple training from scratch is performed to learn the initial set of two classes. At each incremental state, another two new classes arrive with all their available data. These classes should be learned by the model $\mathcal{M}_t$ without forgetting past classes (learned

in previous states). An incremental learning algorithm $\mathcal{A}$ is used to update the model $\mathcal{M}_{t-1}$ ($t > 0$) with data of new classes. In this example, the past exemplars memory $\mathcal{K}$ contains 4 images. Note that the more incremental states are added, the worse the imbalance between past and new classes' data. This imbalance makes class-incremental learning akin to an imbalanced learning that worsens upon incremental states.

The main findings of this chapter are: (1) the bounded memory can be a good alternative to the widely used knowledge distillation on large scale datasets, and (2) the use of a bounded memory alone is not sufficient since the bias of the neural network is important towards new classes, especially at level of the classification scores. Further techniques should be deployed to remove this bias and will be discussed in detail in this chapter.

We propose three approaches called: (1) $DeeSIL$ (*Deep-Shallow Incremental Learning*), (2) $IL2M$ (*Incremental Learning with Dual Memory*) and (3) $ScaIL$ (*Classifier Weights Scaling for class Incremental Learning*).

$DeeSIL$ is based on a fixed representation and applies a transfer learning scheme to class-incremental learning. The model $\mathcal{M}_1$ is trained from scratch on classes of the first state and is frozen afterward. It is used in incremental states to extract features of new classes images where a set of *Support Vector Machines (SVMs)* [19] are used to incrementally learn new classes. It is important to mention that $DeeSIL$ is usable with and without memory of the past. $IL2M$ and $ScaIL$ are based on a fine-tuning backbone. Both methods aim to make classification scores of past and new classes comparable. The main difference arises in the part of the network in which each method interferes. $IL2M$ interferes at the end of the network to directly rectify past classes' scores, while $ScaIL$ interferes at the level of the last fully connected layer weights. We notably propose many simple yet efficient approaches to reduce the bias of the network toward new classes. These approaches are presented in the experiments subsections.

Furthermore, we conduct extensive experiments and study the merits and limitations of our methods as well as a range of class-incremental approaches that use an exemplars memory. We notably vary the number of states and the memory size, the two most important parameters in class-IL. Experiments show that no method is always the best in all configurations.

Finally, we propose in this chapter a combination between class-incremental learning, imbalanced learning, and active learning. Here, the contribution is not directly related to class-incremental learning. We propose new acquisition functions and a simple thresholding IL backbone that better handle data imbalance, and we apply them in an incremental learning protocol.

3.2 DeeSIL: Deep-Shallow Incremental Learning

3.2.1 Introduction

Typical deep learning pipelines are well adapted to solve tasks when all training data is available at all times, and there are loose constraints regarding time available for training. Under these conditions, augmenting the classification ability can simply be done by learning a new representation, either from scratch or via fine-tuning. However, when one or both of the above conditions are violated, adding new classes becomes non-trivial.

We introduce $DeeSIL$, an adaptation of a known transfer learning scheme [44, 71, 125] to incremental learning. In order to qualify as class-incremental and maximize flexibility, $DeeSIL$ includes two weakly correlated steps. First, a deep model $\mathcal{M}_1$ is trained in order to provide a fixed representation which is then used to learn independent shallow classifiers (SVMs) during the incremental phase. Instead of using the system memory $\mathcal{K}$ to keep positive examples, a set of negative features that are necessary to train classifiers incrementally is stored. This choice makes it possible to use all positive examples for training without violating the memory constraint.

Our hypothesis is that independent shallow learning over all positives compensates for the drawback related to the use of a fixed deep representation. Since no deep retraining is needed to increase system capacity, the approach is considerably less complex than its purely deep learning counterparts. The addition of a new class is done by training a shallow classifier, an operation that takes less than a minute on a single CPU. $DeeSIL$ is tested against three IL algorithms, including $iCaRL$ [128], the best such algorithm known to us in 2017. The ILSVRC 2012 dataset [135] (a subset from Imagenet [32]) is used for evaluation and results show significant improvement for the proposed method.

3.2.2 Proposed approach

We propose an adaptation for incremental learning of a well-known transfer learning scheme [71]. An overview of $DeeSIL$ is provided in Figure 3.2 with one initial state $\mathcal{S}_1$ and two incremental states $\mathcal{S}_2$ and $\mathcal{S}_3$. In the initial non-incremental state $\mathcal{S}_1$, we train the model $\mathcal{M}_1$ from scratch on the set of initial classes. Once the model is trained, we remove the last fully connected layer and use the trained feature extractor $\mathcal{F}_1^*$ as a *Deep Feature Extractor* (*DFE*) to extract features for new classes images in all subsequent states. Given an incremental state $\mathcal{S}_{t>1}$ and a set of images X_t^j for a class j to be learned, features F_1^j are extracted using the fixed deep representation provided by $\mathcal{F}_1^*$. Then a shallow binary classifier *SVM* is trained using F_1^j as

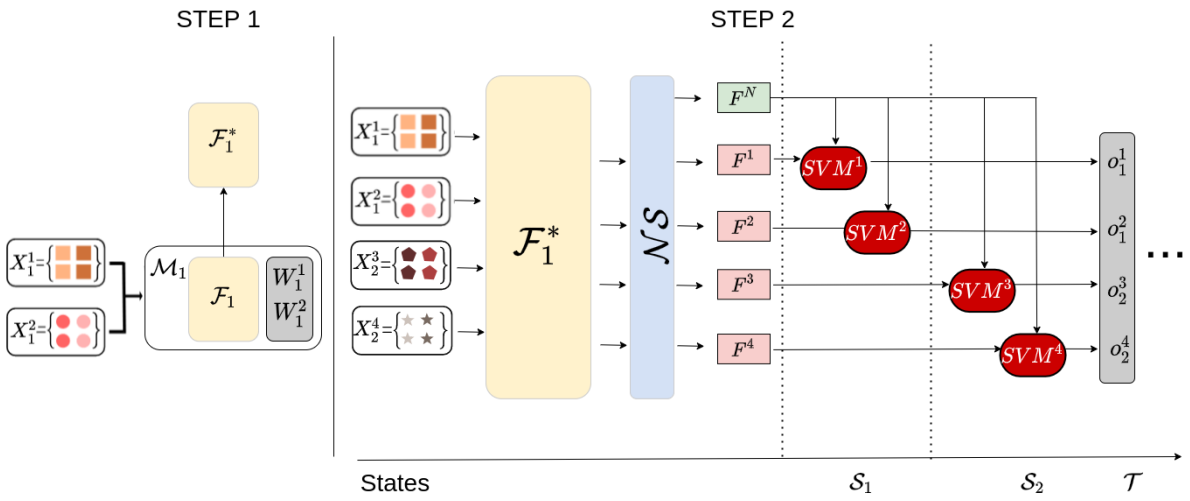


Figure 3.2 – Overview of $DeeSIL$. A toy example with one initial state and two incremental states is presented. Two classes are added at each incremental state. In the first step, the model $\mathcal{M}_1$ is trained from scratch on $N_1 = 2$ classes. Afterward, in the second step, the trained fixed representation $\mathcal{F}_1^*$ is used as a Deep Features Extractor (DFE) to extract features of all classes images as soon as they are available. X^j, F^j are sets of images and features for the j^{th} class. F^N is a set of negative features obtained using a negative selector ($\mathcal{NS}$) and common to all shallow classifiers that are added in a given state. SVM^j is a shallow classifier learned for the j^{th} class, and the output o^j is the associated prediction.

positives and F^N as negatives in order to predict the activations o_t^j of the class for test images. F^N replaces the memory $\mathcal{K}$ of the system, and it contains a constant number $|\mathcal{K}|$ of features, regardless of the state of the system (*i.e.* number of recognizable classes). F^N is generated by the negative selector ($\mathcal{NS}$) component, which is the main adaptation introduced in $DeeSIL$ to make a classical transfer learning pipeline [71] suitable for incremental learning. Given the initial state $\mathcal{S}_1$ (with N_1 recognizable classes), the following steps are needed to move to state $\mathcal{S}_t$ (with $N_{t-1} + P_t$ classes): (1) extract features for the P_t new classes, (2) update the pool of negatives F^N using $\mathcal{NS}$ component, and (3) train P_t shallow classifiers. Note that we use linear SVMs following common practice in transfer learning [44, 71, 125]. We further discuss steps (1) and (2) hereafter.

Deep features extractor (DFE)

In [128], each new state of the class-incremental algorithm depends on the representation learned in the preceding state. Here, deep features extraction and shallow classifier learning are separated. $DeeSIL$ thus implements a form of transfer learning which uses a fixed deep

representation. To evaluate the effect of the amount of training data and its visual proximity with the test data, we train three variants of *DFE*:

— IMAGENET-100 (IN100 below) - train only with the ILSVRC data of the initial state $\mathcal{S}_1$, a setting that is directly comparable with [128];

— IMAGENET-1000 (IN1000 below) - train with a larger dataset that has similar characteristics with the test set (ILSVRC) but has no common classes with it. One thousand classes are selected to form a diversified subset of ImageNet and thus increase universality (*i.e.* optimize their transferability toward new tasks) [161].

— FLICKR-1000 (FL1000 below) - train with a more challenging dataset which is obtained from weakly annotated Flickr group data and is visually more distant from the test set. Within each group, a semi-supervised reranking [27] is initially performed to remove a part of noisy images.

A greedy algorithm [33] which operates with classes' mean representations is used for dataset diversification in the last two variants. It picks at each iteration the class which is, on average, least similar to those already selected. Visual representations from IMAGENET-100 are used as a basis for the diversification process.

Negatives selection

In standard transfer learning [71], shallow classifiers are learned in a one-VS-rest fashion since all data is available at all times. Here, a selection is necessary to fit F^N features in the memory budget $|\mathcal{K}|$ for any state of the algorithm. We test three negative selection strategies:

— ind - (*independent*) following [44], F^N is composed of $|\mathcal{K}|$ image features taken from YFCC dataset [164]. The images are selected so as to represent frequent but diversified tags.

— $rand$ - a random and balanced sampling of image features from all past and current classes.

— div - diversified samples from all recognizable classes. The greedy algorithm implemented for dataset diversification is reused here at the image level.

For $rand$ and div, if $DeeSIL$ recognizes N_t classes in a given state $\mathcal{S}_t$, each class will have $\frac{|\mathcal{K}|}{N_t}$ representatives in F^N. Naturally, a class' own representatives are discarded from F^N when training its shallow classifier.

3.2.3 Experiments

$DeeSIL$ is tested using the ILSVRC 2012 dataset [135]. The evaluation protocol (order of classes, size of system states) is nearly identical to the one used for $iCaRL$ [128]. ILSVRC 2012 includes a total of 1000 classes, further split into 10 batches of 100 classes, which means that 10 distinct states of the class-incremental algorithms are tested. The test set is the same but, since we need to optimize the SVMs, we keep out 20 images for validation and train on remaining images. We use the best three systems from [128] as baselines: (1) $iCaRL$ - their contribution and the most influential IL algorithm known to us in 2017; (2) LwF - adaptation of *Learning without Forgetting* [86] to IL scenario and (3) Fixed Representation (FR) - training over a frozen initial network, except for the classification layer. We use a ResNet-18 as a backbone [51]. The feature vectors extracted with the *DFE* are L2-normalized before being fed into the SVMs. The memory size, which stores negatives, is $|\mathcal{K}| = 20000$, the same as in [128].

3.2.4 Results and discussion

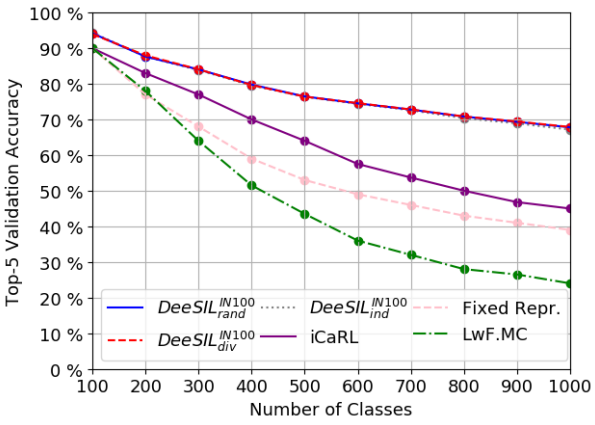


Figure 3.3 – Top-5 accuracy on ILSVRC for $DeeSIL$ variants obtained with three negative selection strategies.

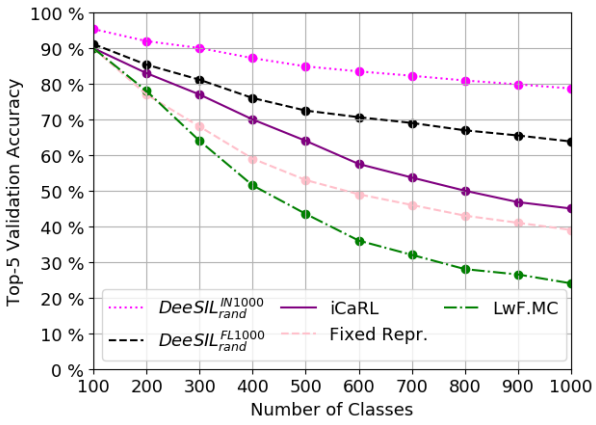


Figure 3.4 – Top-5 accuracy on ILSVRC for fixed deep representations obtained with larger datasets.

Effect of negatives selection

The results in Figure 3.3 show that all variants of $DeeSIL$, trained with $rand$, div and ind negatives selection outperform state-of-the-art systems. At scale, *i.e.* 1000 classes learned incrementally, performance increases from 45% for $iCaRL$ to 68% when $rand$ and div negatives

are exploited. This gain is consistent over all the states of the class-incremental evaluation, with a larger difference for large batches. $DeeSIL$ can be seen as a variant of Fixed Representation learning but differs from it by using all positives in the incremental phase. This leads to an even higher performance gain than in the case of $iCaRL$.

The three $\mathcal{NS}$ variants have rather performance and this finding shows that our method is robust w.r.t. the choice of negatives. Selecting negatives from the dataset ($rand$ and div) gives marginally better results (0.5 points gain) compared to the use of an independent negative set (ind) for 1000 classes. $rand$ being simpler to compute than div, $DeeSIL_{rand}$ will be used in further experiments.

Effect of the deep fixed representation

In Figure 3.4, we test the effect of using more data to obtain strong fixed representations. 1000 ImageNet classes and Flickr groups are used, respectively. Richer data compensates for the fact that features are transferred from classes that are different from the tested ILSVRC classes. This is especially the case for $DeeSIL^{IN1000}_{rand}$, which exploits a subset of ImageNet distinct from ILSVRC. Performance improvements of 10 and 33 points are obtained over $DeeSIL^{IN100}_{rand}$, the best configuration trained with the 100 initial ILSVRC classes and over $iCaRL$ respectively. $DeeSIL^{FL1000}_{rand}$, the version trained on non-curated Flickr data has lower performance than $DeeSIL^{IN1000}_{rand}$, but is close to $DeeSIL^{IN100}_{rand}$ and still well above the state of the art algorithms. The last result confirms the finding in [27] that it is possible to learn reasonable representations even with little or no manual data.

Memory and computational complexity

Beyond performance, it is also important to compare the complexity of $DeeSIL$ to that of $iCaRL$, the main baseline. ResNet-18, the basic deep architecture, is the same for both methods. Recognition capacity incrementation is done with linear SVMs. This entails the computation of a dot product per class, which is equivalent to adding a class in the final layer of a CNN.

Training is simpler in $DeeSIL$ since a single deep network training is needed at the beginning. In the incremental step, we only train shallow classifiers. Adding a single class typically takes less than 1 minute, distributed among deep features extraction and SVM training on an INTEL-Xeon-E5-2650-v2@2.60GHz CPU. For comparison, adding a batch of 100 new classifiers in $iCaRL$ takes approximately 32 hours on an NVIDIA Titan X GPU. Incremental learning is typically needed in low-resource contexts and, assuming that an initial deep representation is

available, $DeeSIL$ can be deployed even in the absence of a GPU. Equally important, due to the independent learning of shallow classifiers, $DeeSIL$ can seamlessly integrate batches of new classes of arbitrary size. In contrast, purely deep-learning-based algorithms need retraining, and this step is particularly long if one class is added at a time.

Usability of DeeSIL in IL without memory

Compared to $iCaRL$, the focus in $DeeSIL$ is shifted from positive to negative selection to fill in the memory of the system. As shown in the experiments, our algorithm is affected by catastrophic forgetting to a much lesser extent. The choice to select negatives is beneficial for scalability in terms of the number of learnable classes. Also, while the same number of items is stored in $iCaRL$ and $DeeSIL$, memory needs are lower in our case since we store $D = 512$ dimensional features instead of images of past classes. We remind that D depends on the deep architecture used.

Given a memory budget $|\mathcal{K}|$, $iCaRL$ can learn at most $|\mathcal{K}|$ classes since in the state $\mathcal{S}_{|\mathcal{K}|+1}$, not all known classes will be represented anymore. However, $DeeSIL$ can be easily deployed without a memory of the past, by replacing the negatives pool F^N with negatives from other classes belonging to the same state. In this case, $DeeSIL$ can learn an infinite number of classes with the condition that at least two classes are added in each new incremental state. Even if only one class appears at each incremental state, $DeeSIL$ can be deployed by using as negatives features of images coming from an independent dataset (like the YFCC dataset [164] above).

Comparison with an upper bound

It is interesting to evaluate the decrease in performance compared to a situation in which all training data is available at all times. ResNet-18 [51] top-5 accuracy on 1000 ILSVRC classes trained with all data is approximately 89%. $iCaRL$ halves this score while our best configurations with *DFE* based on 100 and 1000 classes lose only 22 and 12 points respectively. The gap could be further reduced if the feature extractors were more universal [161]. This could be achieved if $DeeSIL$'s initial training would be done with an even larger number of classes.

3.2.5 Conclusion

We revisit a known transfer learning scheme [44] for class-incremental learning. The proposed method achieves significantly better performance than existing algorithms [86, 128] while also being much faster to train and more scalable in terms of the number of learnable classes.

3.3 IL2M: Class-Incremental Learning with Dual Memory

3.3.1 Introduction

Previously, we proposed $DeeSIL$, an algorithm that applies transfer learning to tackle class-incremental learning. The main limitation of $DeeSIL$ is that its performance is heavily dependent on the quality of the fixed representation. If the latter is trained with few data, or with classes that are visually different from that of classes learned incrementally, $DeeSIL$ fails to continuously transfer knowledge to subsequent states. A solution for this is to fine-tune the model at each incremental state in order to update its representation with the new data. However, as shown in Section 3.1, the imbalance between past and new classes worsens upon time, a phenomenon that necessitates techniques to remove the bias of the network towards new classes.

In this section, we introduce *Incremental Learning with Dual Memory* ($IL2M$) that aims to partially reconcile the fine tuning and fixed representation based approaches. $IL2M$ uses vanilla fine tuning as a backbone to update deep models for each incremental state, as proposed in fine tuning approaches (Subsection 2.3.1). Similar to fixed representation methods (Subsection 2.3.2), $IL2M$ exploits class-related knowledge from the initial state in which they were learned across incremental states. Due to deep parameter updating when fine-tuning the model, initial class models cannot be fully reused in later states. Instead, $IL2M$ exploits past class statistics from their initial state to rectify their prediction scores in the current incremental state. This rectification is supported by two related hypotheses: (1) classes are best modeled when all their data are available, and (2) class prediction scores are higher on average when more training data are available. We illustrate the validity of these hypotheses in Figure 3.5. It plots the averaged predictions of past and new classes for the ILSVRC [135], VGGFACE2 [22] and LANDMARKS [108] datasets with $\mathcal{T} = 10$ states and memory sizes $|\mathcal{K}| = \{20000, 10000, 5000\}$.

The scores in Figure 3.5 confirm that vanilla fine tuning generates a prediction bias in favor of new classes. This bias is mainly due to the imbalance in favor of new classes, which appears in class IL. As a result, a large part of images from past classes is predicted as belonging to new classes (see Subsection 3.3.4 for a detailed analysis of error types). The comparison of the three subfigures of each dataset shows that the score gap between past and new classes is higher when the memory capacity is lower. For instance, the average difference over all incremental states for the ILSVRC dataset is 2.42, 4.02 and 6.45 for $|\mathcal{K}| = \{20000, 10000, 5000\}$ respectively. This is intuitive since the imbalance between past and new classes is higher for lower memories. The gap also tends to grow from left to right in each subfigure due to the increasing number of classes to fit in the bounded memory. For instance, the difference is 2.26, 4.16 and 4.67 for

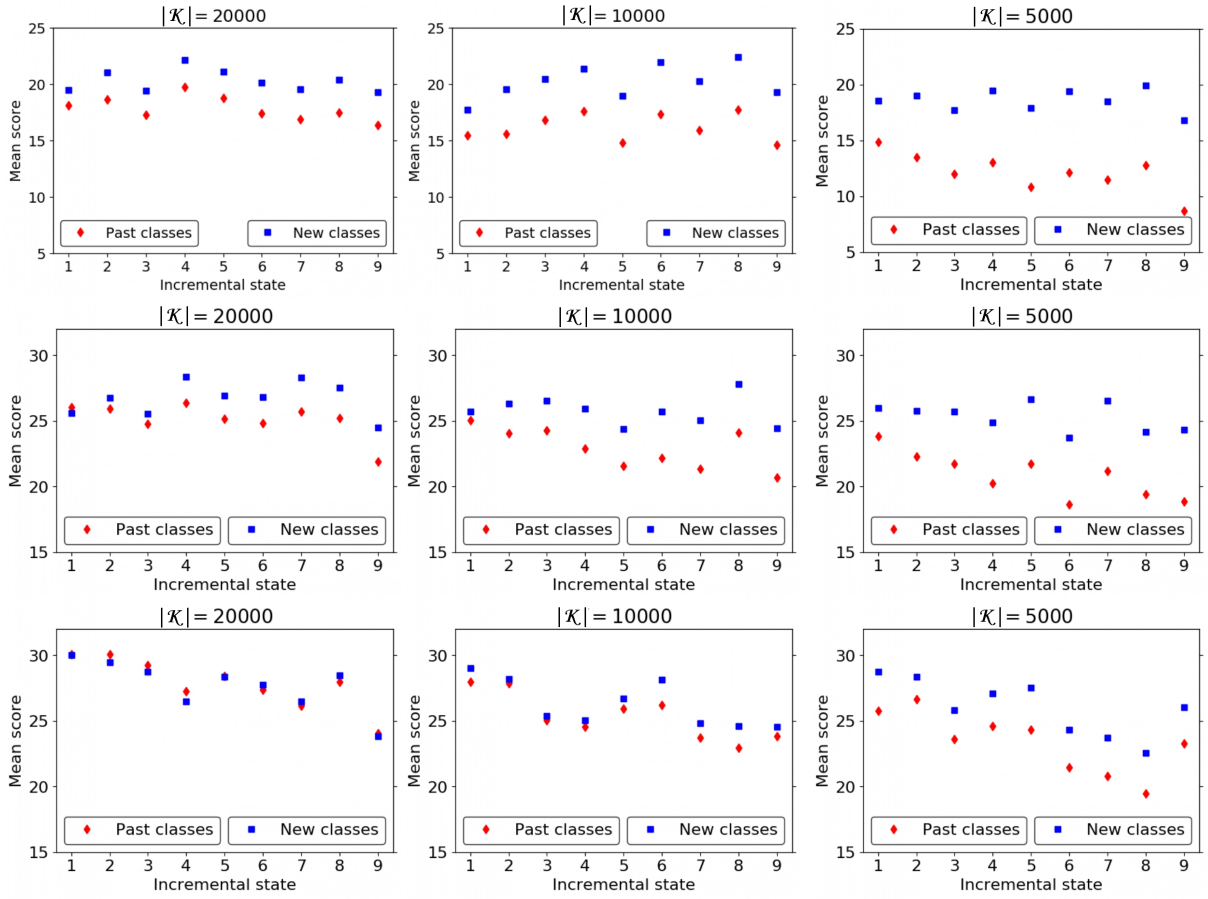


Figure 3.5 – Prediction scores for the ILSVRC, VGGFACE2, and LANDMARKS datasets with $\mathcal{T} = 10$ states and memory $|\mathcal{K}| = \{20000, 10000, 5000\}$ exemplars. We select the scores of the true class for train images and then average them for past and new classes. Only incremental states (from $\mathcal{S}_2$ to $\mathcal{S}_{10}$) are represented. The initial one does not include past classes and is not represented. *Best viewed in color.*

states $\mathcal{S}_2$, $\mathcal{S}_6$ and $\mathcal{S}_{10}$ with $|\mathcal{K}| = 10000$ exemplars.

The differences between predicted scores of past and new classes are much smaller than those of ILSVRC for VGGFACE2 and negligible for LANDMARKS when $|\mathcal{K}| = 20000$. This difference between scores is due to the fact that ILSVRC is a hard task, and it explains the very small contribution of $IL2M$ score rectification in this configuration (Subsection 3.3.4).

3.3.2 Proposed approach

Our method, called Incremental Learning with Dual Memory ($IL2M$) is summarized in Figure 3.6. with an example that includes an initial and two incremental states. $IL2M$ uses a

fixed DNN architecture and a bounded memory of the past. Our main contribution is to propose a secondary memory that stores initial class statistics in a very compact format. The introduction of this memory is based on the intuition that classes are best modeled when first learned, with all data available. Initial class statistics are reused in each subsequent incremental state to rectify the prediction scores of past classes. Rectification is necessary because class IL models are trained with imbalanced datasets in which past classes have fewer examples. Consequently, their prediction scores are generally lower than those of new classes.

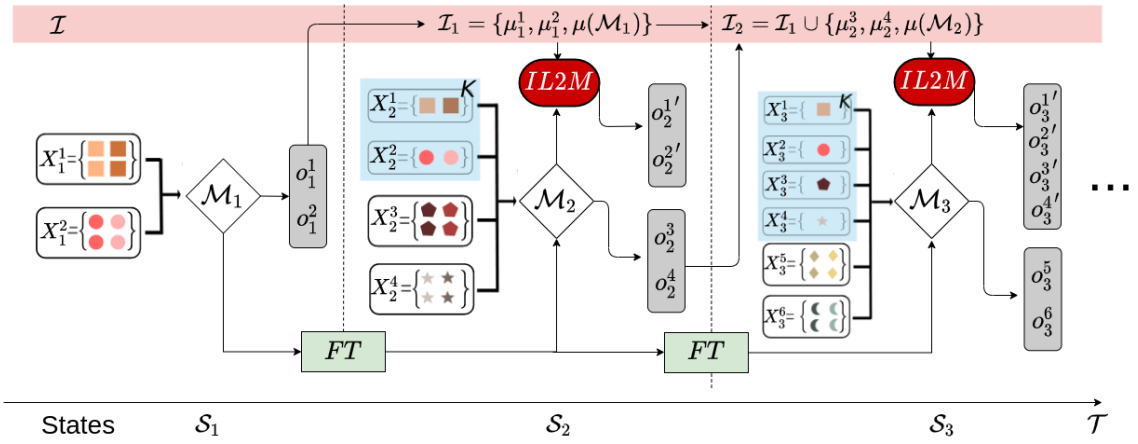


Figure 3.6 – Illustration of the proposed $IL2M$ training process. The deep models associated to the three states recognize 2, 4 and 6 classes respectively. The bounded memory includes $|\mathcal{K}| = 4$ image exemplars of past classes and is represented on light blue background. The number of class exemplars class stored in memory decreases when adding new classes to keep memory requirements constant. The IL training process is more and more prone to catastrophic forgetting because the dataset is increasingly imbalanced. The second memory $\mathcal{I}$, represented in light pink, stores statistics that are obtained when classes are initially learned. $IL2M$ makes these class statistics usable across different incremental states to rectify the raw prediction scores of past classes in order to make them more comparable to those of new classes. *Best viewed in color.*

To compensate for the bias toward new classes, we rectify predictions of past classes ($j = 1, \ldots, N_{t-1}$) using Equation 3.1 (check Section 2.2 for the used formal notations):

$$o_t^{j\prime} = IL2M(o_t^j) = \begin{cases} o_t^j \times \frac{\mu_i^j}{\mu_t^j} \times \frac{\mu(\mathcal{M}_t)}{\mu(\mathcal{M}_i)}, & \text{if } pred = new \\ o_t^j, & \text{otherwise} \end{cases} \tag{3.1}$$

with:

— i - the initial state in which the j^{th} class was learned

— t - the current incremental state

— o_t^j - the raw prediction the j^{th} class in the current state t

— μ_i^j and μ_t^j - the mean classification scores of the j^{th} class in states i and t obtained from all training data and the current exemplar set, respectively

— $\mu(\mathcal{M}_t)$ and $\mu(\mathcal{M}_i)$ - the model confidences in states t and i given by the averaged top1 prediction scores of all new training data.

In Equation 3.1, rectification is applied to past class predictions only if an image is initially predicted as belonging to a new class. This situation is the riskiest in terms of imbalance-driven errors in favor of new classes. Otherwise, we consider rectification unnecessary since a past class is directly predicted, and there is no prediction bias toward new classes. The effect of the rectification restriction to past images initially associated with a new class is studied in the ablation study from Subsection 3.3.4.

Since classes are initially learned in different incremental states, the following conditions need to be met for the proposed rectification to be useful in class IL:

1. the scores $o_t^{j'}$ for past classes in range $j = \{1, N_{t-1}\}$ and o_t^j from $\{N_{t-1}+1, N_t\}$ should be comparable;
2. the statistics stored in the memory $\mathcal{I}$ should be very compact in order to increase memory needs only marginally;
3. model level normalization should be introduced to limit the influence of combining the outputs of models learned in different incremental states.

The first condition is handled via the use of class-related statistics in the first term, which modifies o_t^j in Equation 3.1. More specifically, we use the means of the j^{th} class in its initial and current states i and t. The intuition here, supported by Figure 3.5, is that since the class is first learned with all training images in the state i when it was new, its mean prediction score μ_i^j is likely to be higher than μ_t^j. Consequently, this term of the equation generally increases $o_t^{j'}$ compared to o_t^j. The second condition listed above is related to the introduction of the secondary memory $\mathcal{I}$, which makes the $IL2M$ rectification possible. $\mathcal{I}$ includes a float value per class to store μ_i^j, and the induced memory requirement is negligible. As for the model-level knowledge, only one float per incremental state is needed to store $\mu(\mathcal{M})$.

The third condition is necessary since the averaged scores for new classes are not equivalent in the different incremental states which are combined. This is clear in Figure 3.5 where in ILSVRC, for instance, the new class mean scores for state 8 are higher than those of state 7

for $|\mathcal{K}| = 10000$. The last term of Equation 3.1 provides a global harmonization of the score rectification across the different states that are combined in $IL2M$.

The complexity of the supplementary arithmetic operations from Equation 3.1 is very low compared to the overall complexity of deep neural network architecture. For each class score rectification, a division and a multiplication are needed to introduce the second term. The division in the third term can be computed only once the training of the current incremental state is ready. This term is thus integrated through a simple multiplication. For 1000 past classes, $IL2M$ adds 1000 divisions and 2000 multiplications. This is to be compared to the tens to hundreds of millions of multiplications done in typical DNN architectures.

The rectification introduced here is an alternative to the NEM classification from $iCaRL$ [128] and to the balanced fine tuning step of end-to-end learning ($E2EIL$) from [24]. The three methods are compared in the following section.

3.3.3 Experiments

— **Datasets** (Appendix B) - ILSVRC [135], LANDMARKS [108], and VGGFACE2 [22].

— **Memory sizes** - We fix the number of states $\mathcal{T} = 10$ and vary the memory size to include $|\mathcal{K}| = \{20000, 10000, 5000\}$ exemplars.

— **Incremental states** - We fix the memory to $|\mathcal{K}| = 5000$ and test with $\mathcal{T} = \{5, 20\}$ in addition to $\mathcal{T} = 10$. The lowest memory size was selected since it is the most interesting configuration when memory budget is smallest.

— **Exemplar selection -** According to [24], herding has marginal effect. For sake of simplicity, we perform a random selection of exemplars. We will show in Sections 3.4 and 3.5 that herding is beneficial for $IL2M$ and other IL methods.

— **Evaluation measures -** Top-1 and Top-5 accuracy [135].

— **Baselines** (Chapter 2) - $iCaRL$ [128], $DeeSIL$ (Section 3.2), FT (vanilla fine tuning). In addition, we propose the following baselines:

➔ FT^{NEM} - a version of FT which uses the nearest-exemplars-mean classifier from [128] instead of the classification layer of the deep network. FT^{NEM} is a modified version of $iCaRL$ in which the distillation loss $\mathcal{L}_d$ is ablated.

➔ FT^{BAL} - a version of FT in which a balanced fine tuning is performed for classification after the initial imbalanced vanilla FT following [24]. FT^{BAL} is a modified version of $E2EIL$ [24] in which we again ablate $\mathcal{L}_d$. Note that original $E2EIL$

States	$\mathcal{T}=10$												$\lvert\mathcal{K}\rvert=5000$					
Dataset	ILSVRC				VGGFACE2				LANDMARKS				ILSVRC		VGGFACE2		LANDMARKS	
$\lvert\mathcal{K}\rvert$	$20k$	$10k$	$5k$	$0k$	$20k$	$10k$	$5k$	$0k$	$20k$	$10k$	$5k$	$0k$	$\mathcal{T}$=5	$\mathcal{T}$=20	$\mathcal{T}$=5	$\mathcal{T}$=20	$\mathcal{T}$=5	$\mathcal{T}$=20
$iCaRL$	35.1	33.6	32.9	20.8	66.8	65.3	64.4	26.1	68.9	66.9	65.6	27.0	32.7	29.6	74.1	49.5	73.8	52.6
$DeeSIL$	47.3	47.2	**47.0**	**46.5**	81.5	81.3	80.9	**80.0**	82.8	82.6	82.4	**81.2**	**50.9**	28.4	89.3	69.3	88.3	74.9
FT	51.1	42.3	32.2	18.3	91.1	87.6	82.0	20.8	93.2	90.1	84.7	21.0	35.4	36.8	85.7	83.3	85.4	84.1
FT^{NEM}	54.9	49.1	42.8	*NA*	91.1	87.6	84.2	*NA*	91.1	88.5	84.7	*NA*	44.1	**46.2**	87.4	**85.7**	83.4	84.4
FT^{BAL}	52.1	47.0	37.2	*NA*	91.5	88.6	82.1	*NA*	93.2	90.2	85.7	*NA*	44.7	41.6	87.7	83.9	88.2	84.8
$IL2M$	**56.4**	**50.8**	44.1	*NA*	**92.0**	**89.7**	**86.5**	*NA*	**93.4**	**90.8**	**86.9**	*NA*	44.9	42.0	**90.1**	**85.7**	**88.5**	**85.0**
$Joint$	73.0				97.0				97.1				73.0		97.0		97.1	

Table 3.1 – Top-1 average accuracy (%) for the different methods tested. *NA* stands for "Not Applicable". $Joint$ is the non-incremental upper-bound performance obtained with all data available for all classes. The available memory $\lvert\mathcal{K}\rvert$ (in thousand exemplars) and the number of states $\mathcal{T}$ are varied to the left and the right of the table. Each time, the other parameter is fixed. Following [24], accuracy is averaged only for incremental states (i.e. excluding the initial, non-incremental state). *Best results are in bold.*

[24] is not fully evaluated because the only available implementation uses non-free MathConvNet based on Matlab. However, a top-5 accuracy comparison of $E2EIL$ and FT^{BAL} for ILSVRC is clearly favorable to the latter method (69.4 vs. 77.52).

3.3.4 Results and discussion

Top-1 accuracy results

The comparison of the methods tested in Table 3.1 shows that $IL2M$ has the best performance in a wide majority of configurations with memory ($\lvert\mathcal{K}\rvert > 0$). Our method outperforms $iCaRL$ [128], $DeeSIL$, as well as FT (the vanilla fine tuning baseline) and its variants FT^{NEM} and FT^{BAL}, which use the classification components from [24] and [128].

Among tested baselines, FT consequently outperforms $iCaRL$ for $\mathcal{T} = 10$ and $\lvert\mathcal{K}\rvert = \{20000, 10000\}$. For $\lvert\mathcal{K}\rvert = 5000$, it is better for $\mathcal{T} = \{5, 20\}$ states and slightly falls behind for ILSVRC with $\mathcal{T} = 10$ states. Naturally, $iCaRL$ is better when no memory is allowed and distillation reduces catastrophic forgetting. The comparison of FT to $DeeSIL$ is also favorable for all settings where $\lvert\mathcal{K}\rvert > 0$, except for $\mathcal{T} = 5$ and ILSVRC with $\mathcal{T} = 10$ and $\lvert\mathcal{K}\rvert = \{5000, 10000\}$.

The detailed results for the three datasets with $\lvert\mathcal{K}\rvert = 10000$ and $\mathcal{T} = 10$ from Figure 3.7 confirm the above findings. $IL2M$ has the best performance for a wide majority of IL states. It is also interesting to see that our method provides good results for later incremental states. This is clear for ILSVRC, where $IL2M$ has similar performance with that of FT^{NEM} and $DeeSIL$

for states 7 to 9 and is better than them in earlier states. The gap between $iCaRL$ performance and all FT methods introduced here is large overall and clearly increases in later states for VGGFACE2 and LANDMARKS. This finding indicates that vanilla FT is a much better base for IL when the number of classes is large.

While our focus is on class IL with memory, we also present results without memory ($|\mathcal{K}| = 0$). Here distillation clearly has a positive effect and outperforms fine tuning, thus confirming the results from [128]. All methods derived from FT have the same performance because all score rectification methods rely on exemplars. $DeeSIL$ is the best method when $|\mathcal{K}| = 0$ because it has low dependence on memory. Except for $\mathcal{T} = 20$, its performance is better than that of $iCaRL$ by a consequent margin. This result is at odds with the conclusion of [128], where the authors found their fixed representation to be less effective than $iCaRL$. The difference is explained by the fact that fixed representations of past classes in [128] were learned only with exemplars from the current state. This restriction is unnecessary since the representation is fixed and each class can be learned the first time it is seen without violating memory requirements and then reused across IL states.

When compared to $Joint$, the upper-bound non-incremental learning, the results obtained by all incremental methods are lower in all configurations. This is particularly the case for ILSVRC, the hardest task among the three tested, where the gap reaches 16.6 top-1 accuracy points for $\mathcal{T} = 10$ states and $|\mathcal{K}| = 20000$. Naturally, this gap grows for all datasets when the memory is reduced. This finding confirms the conclusions of [24, 128] that class IL remains a hard problem if it operates under computational and memory constraints.

Top-5 accuracy results

In addition to the top-1 results, we provide top-5 results obtained by all methods to facilitate comparability with earlier works [24, 61, 128]. Overall, the results follow the same trend as top-1. It is noteworthy that the differences between the FT baseline and the methods built on top of it are globally lower than those with top-1 results. This is particularly true for the VGGFACE2 and LANDMARKS, the easiest datasets tested here, where the imbalance inherent to incremental learning matters less than in the case of ILSVRC. The smaller performance differences are explained by the fact that top-5 accuracy has a smoothing effect on results. $IL2M$ is still the best method in a majority of tested configurations. A first notable difference is that FT^{NEM} gives slightly better results for three configurations instead of one for top-1. A second difference is that $DeeSIL$ has best performance for all datasets with $|\mathcal{K}| = 5000$ and $\mathcal{T} = 5$. This is due to the fact that the initial representation is stronger when it includes a higher

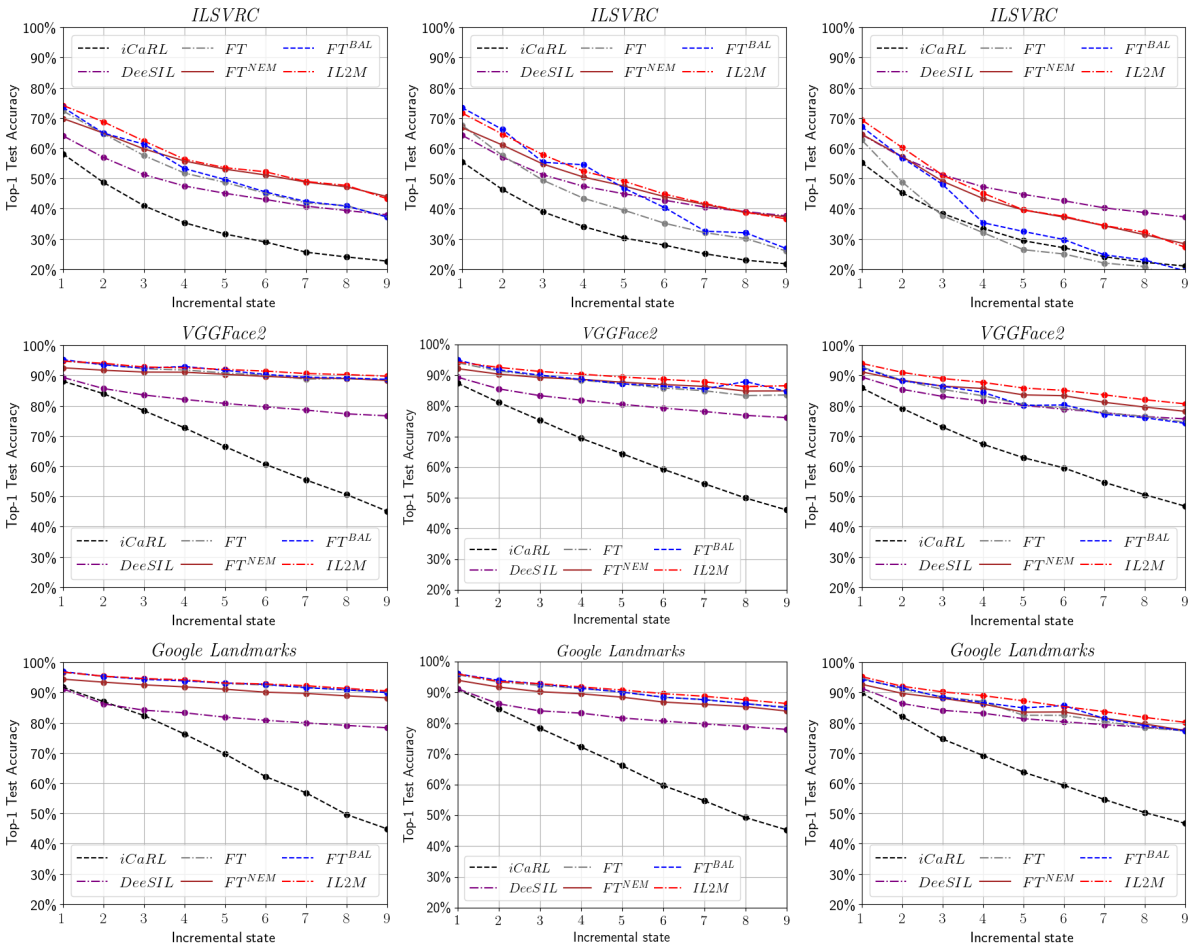


Figure 3.7 – Top-1 accuracy for object, face and landmark recognition with memory $|\mathcal{K}| = 10000$ and $\mathcal{T} = 10$ states. To be aligned with the results from Table 3.1, only the incremental states are represented. *Best viewed in color.*

number of classes. $DeeSIL$ has the best top-5 performance for ILSVRC with $|\mathcal{K}| = 5000$ and $IL2M$ comes second in this case.

Compared to $Joint$, the non-incremental training, the best class IL algorithms with $\mathcal{T} = 10$ and memory $|\mathcal{K}| = 20000$ loses 12.9, 2 and 1.5 top-5 points for ILSVRC, VGGFACE2, and LANDMARKS, respectively. This gap is rather small for VGGFACE2 and LANDMARKS, but more work is still needed for difficult tasks like ILSVRC. Naturally, the gap increases when the memory is reduced and the number of states increases. As expected, it becomes very important without memory. In this last case, which is not in focus here, the $DeeSIL$ baseline performs best for all three datasets.

States	$\mathcal{T}=10$												$\lvert\mathcal{K}\rvert=5000$					
Dataset	ILSVRC				VGGFACE2				LANDMARKS				ILSVRC		VGGFACE2		LANDMARKS	
$\lvert\mathcal{K}\rvert$	$20k$	$10k$	$5k$	$0k$	$20k$	$10k$	$5k$	$0k$	$20k$	$10k$	$5k$	$0k$	$\mathcal{T}$=5	$\mathcal{T}$=20	$\mathcal{T}$=5	$\mathcal{T}$=20	$\mathcal{T}$=5	$\mathcal{T}$=20
$iCaRL$	62.5	61.4	60.9	43.8	84.5	83.9	83.6	48.3	84.4	83.6	83.0	46.3	61.0	56.3	89.4	71.6	89.0	71.2
$DeeSIL$	74.5	74.3	**74.2**	**73.9**	92.6	92.6	92.5	**92.3**	94.2	94.1	94.0	93.6	**79.2**	69.0	**96.4**	87.2	**96.4**	90.3
FT	77.0	70.1	60.0	20.5	97.1	96.0	94.1	21.3	**97.6**	96.5	94.4	21.3	61.9	64.5	95.6	94.4	94.6	93.8
FT^{NEM}	**79.4**	74.5	69.6	*NA*	96.7	95.7	94.1	*NA*	96.8	95.8	93.9	*NA*	71.2	**71.4**	95.4	**94.6**	93.2	93.6
FT^{BAL}	77.5	73.4	65.0	*NA*	**97.2**	**96.2**	94.3	*NA*	97.5	96.5	94.6	*NA*	70.1	67.8	96.1	94.5	95.4	**94.0**
$IL2M$	78.3	**75.2**	71.2	*NA*	**97.2**	**96.2**	**94.9**	*NA*	**97.6**	**96.6**	**94.7**	*NA*	75.6	66.1	**96.4**	94.5	95.3	93.6
$Joint$	92.3				99.2				99.1				92.3		99.2		99.1	

Table 3.2 – Top-5 average accuracy (%) for the different methods tested. *NA* stands for "Not Applicable". $Joint$ is the non-incremental upper-bound performance obtained with all data available for all classes. Accuracy is averaged only for incremental states (excluding the initial, non-incremental state). The available memory $\lvert\mathcal{K}\rvert$ and the number of states $\mathcal{T}$ are varied to test their effect on the performance of the tested methods. *Best results are in bold.*

Effect of score rectification

$IL2M$, FT^{NEM} and FT^{BAL} all use vanilla FT with memory as IL backbone. The three methods differ in the way final classification scores are obtained. FT^{NEM} uses the NEM method from [128] as external classifier. FT^{BAL} classifier adds a balanced fine tuning step for classification following [24]. $IL2M$ notably exploits the content of a second memory to rectify scores. The results from Table 3.1 show that our method yields better performance than FT^{NEM} and FT^{BAL} for almost all configurations tested.

Equally important, $IL2M$ is useful for all memory sizes while this is not the case for NEM in FT^{NEM}, which actually hurts FT performance for LANDMARKS in three tested configurations. The balanced fine tuning in FT^{BAL} also improves performance for all memory sizes but to a lesser extent than $IL2M$. With lower memory, FT^{BAL} is more prone to catastrophic forgetting than $IL2M$ and FT^{NEM} because a larger extent of data needs to be dropped during balancing. It is noticeable that the usefulness of score rectification grows when exemplar memory is lower and the imbalance between past and new classes is consequently higher. For instance, $IL2M$ gains 5.3 and 11.9 top-1 accuracy points for ILSVRC with $\lvert\mathcal{K}\rvert = 20000$ and $\lvert\mathcal{K}\rvert = 5000$ exemplars respectively when $\mathcal{T} = 10$.

Effect of distillation

The results from Table 3.1 and Figure 3.7 show that the use of distillation loss is detrimental in class IL if at least a few exemplars per past class are allowed. The ablation of $\mathcal{L}_d$ in $iCaRL$

		Incremental states								
		$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$
$hybrid1$	$c(p)$	1075	1217	1442	1446	1435	1535	1483	1505	1591
	$e(p,p)$	600	2053	3756	5091	7406	9074	10580	11794	14156
	$e(p,n)$	3325	6730	9802	13463	16159	19391	22937	26701	29253
	$c(n)$	3562	3739	3558	3603	3673	3750	3584	3762	3641
	$e(n,n)$	1020	839	965	910	793	791	903	792	810
	$e(n,p)$	418	422	477	487	534	459	513	446	549
FT	$c(p)$	2621	4327	5730	6702	7600	7980	8576	9169	8746
	$e(p,p)$	194	690	1360	2203	3035	4016	4462	6100	5514
	$e(p,n)$	2185	4983	7910	11095	14365	18004	21962	24731	30740
	$c(n)$	4139	4314	4145	4155	4251	4319	4236	4376	4267
	$e(n,n)$	779	608	771	762	692	619	694	560	667
	$e(n,p)$	82	78	84	83	57	62	70	64	66

Table 3.3 – Top-1 analysis of $hybrid1$ the FT with distillation used as backbone for $iCaRL$ [128] and for vanilla FT using $\mathcal{T} = 10$ and $|\mathcal{K}| = 10000$. $c(\cdot)$ $e(\cdot,\cdot)$ stand for correct and erroneous predictions and p and n stand for past and new classes. For instance, $e(p,p)$ designates the number of past samples wrongly predicted as other past classes.

to obtain FT^{NEM} is beneficial for all datasets and memory sizes $|\mathcal{K}| = \{20000, 10000, 5000\}$ and $\mathcal{T} = 10$. The results presented here are at odds with the conclusion of [128] about the low performance of vanilla fine tuning in class IL with memory. That conclusion was based on a biased comparison of $iCaRL$ and FT since the first method used an exemplar memory and the second did not. Naturally, distillation is useful when no memory is allowed, the setting for which it was initially designed [86] and which is not in focus here. While we do not have a complete set of results for $E2EIL$ [24], we note that distillation is also harmful to this method on the ILSVRC dataset with $|\mathcal{K}| = 20000$. The original top-5 result reported in [24] is 69.4 while the modified FT^{BAL} version introduced here reaches 77.52.

In Table 3.3, we analyze the behavior of $hybrid1$, the version of FT with distillation which serves as a backbone for iCaRL [128] and of vanilla FT for ILSVRC with $|\mathcal{K}| = 10000$ images and $\mathcal{T} = 10$ states. The bias toward new classes ($e(p,n)$) is comparable for the two methods, although slightly higher when distillation is used. Consequently, data imbalance is not the main factor that explains the difference between the two methods. This difference comes mostly from the distribution of wrong classifications between past classes ($e(p,p)$). While distillation is assumed to preserve accuracy for past classes, the obtained results indicate that $hybrid1$ makes between two and three times more mistakes than vanilla fine tuning. A possible explanation for this situation is that distillation is usually assumed to be initialized with a strong model learned

on a large and balanced dataset [51]. This condition is not met in IL since the models from the previous state are trained on an imbalanced dataset.

The analysis from the previous two sections shows that data imbalance inherent to class IL with memory produces a classification bias toward new classes. In Table 3.4, we enrich the analysis by providing an analysis of error types before (FT) and after ($IL2M$) score rectification for all datasets with memory $|\mathcal{K}| = 10000$ and $\mathcal{T} = 10$ states.

Before rectification, the largest number of errors is of type $e(p, n)$, that is, test images of past classes mistaken for images of new classes. We will look closely at the incremental state $\mathcal{S}_{10}$ of ILSVRC, which includes 45000 and 5000 test images for past and new classes, respectively. 30740/45000 (68.31%) of test images of past classes were predicted as new, and only 8746/45000 (19.43%) images were correctly predicted. 4267/5000 (85.34%) of test images of new classes are predicted correctly, and only 66/5000 (1.32%) of them are assigned to past classes. These statistics further confirm the bias in favor of new classes and the need for score rectification.

After rectification with $IL2M$, the distributions of correct predictions and of errors changes quite significantly. For ILSVRC, there are significantly more correct predictions for past classes, accompanied by a lower performance for new classes. In state $\mathcal{S}_{10}$ of ILSVRC, correct predictions of past test images increase from 19.43% with FT to 32.86% with $IL2M$. The corresponding performance for new classes drops from 85.34% to 70.2%. $IL2M$ ensures a better performance balance between past and new classes. The errors of type $e(p, p)$, where images of a past class are mistaken for images of another past class, are increasingly frequent toward later incremental states. This covers a majority of cases for states from $\mathcal{S}_6$ to $\mathcal{S}_{10}$. The number of images of past classes predicted as new decreases significantly, and these errors cover only 21.32% of test images for past classes in the state $\mathcal{S}_{10}$ of ILSVRC.

Ablation study

We analyze the contribution of the $IL2M$ components in an ablation study with the ILSVRC dataset for $\mathcal{T} = 10$ states and memory $|\mathcal{K}| = \{20000, 10000, 5000\}$. We test the following changes on top of the FT baseline: $IL2M^1$ - activation of the first component of the rectification which works with class level means; $IL2M^2$ - activation of the second component which works with model level means; $IL2M^{1+2}$ - both mean based components are activated; $IL2M$ - full version in which we also add the restriction of rectifying past class scores only if an image is initially predicted as belonging to a new class (given by Equation 3.1).

The results from Table 3.5 indicate that each component has a positive effect compared to

			Incremental states								
Dataset			$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$
$ILSVRC$	FT	$c(p)$	2621	4327	5730	6702	7600	7980	8576	9169	8746
		$e(p,p)$	194	690	1360	2203	3035	4016	4462	6100	5514
		$e(p,n)$	2185	4983	7910	11095	14365	18004	21962	24731	30740
		$c(n)$	4139	4314	4145	4155	4251	4319	4236	4376	4267
		$e(n,n)$	779	608	771	762	692	619	694	560	667
		$e(n,p)$	82	78	84	83	57	62	70	64	66
	$IL2M$	$c(p)$	3223	5913	7744	9279	11233	11899	13115	13563	14791
		$e(p,p)$	433	2010	3374	5324	9177	11239	13984	16780	20614
		$e(p,n)$	1344	2077	3882	5397	4590	6862	7901	9657	9595
		$c(n)$	3940	3791	3815	3816	3484	3774	3552	3900	3510
		$e(n,n)$	666	409	582	553	352	361	398	347	341
		$e(n,p)$	394	800	603	631	1164	865	1050	753	1149
$VGGFACE2$	FT	$c(p)$	4619	8887	13114	17234	21279	25163	29084	32617	36893
		$e(p,p)$	62	275	580	898	1270	1638	2051	2649	3145
		$e(p,n)$	319	838	1306	1868	2451	3199	3865	4734	4962
		$c(n)$	4789	4814	4847	4868	4873	4879	4878	4868	4884
		$e(n,n)$	167	129	115	87	90	88	86	92	88
		$e(n,p)$	44	57	38	45	37	33	36	40	28
	$IL2M$	$c(p)$	4657	9122	13436	17780	22031	26232	30353	34024	38506
		$e(p,p)$	78	378	813	1382	1885	2601	3287	4039	4781
		$e(p,n)$	265	500	751	838	1084	1167	1360	1937	1713
		$c(n)$	4776	4762	4814	4810	4806	4802	4798	4802	4784
		$e(n,n)$	161	112	94	63	70	55	56	72	57
		$e(n,p)$	63	126	92	127	124	143	146	126	159
$LANDMARKS$	FT	$c(p)$	1894	3649	5423	7170	8847	10414	12070	13570	15093
		$e(p,p)$	31	85	174	329	516	643	858	1128	1437
		$e(p,n)$	75	266	403	501	637	943	1072	1302	1470
		$c(n)$	1937	1952	1957	1954	1969	1960	1963	1965	1960
		$e(n,n)$	49	32	32	37	18	22	27	24	29
		$e(n,p)$	14	16	11	9	13	18	10	11	11
	$IL2M$	$c(p)$	1907	3718	5493	7230	8951	10599	12245	13826	15358
		$e(p,p)$	45	107	218	384	587	834	1067	1462	1711
		$e(p,n)$	48	175	289	386	462	567	688	712	931
		$c(n)$	1934	1896	1935	1949	1944	1947	1955	1940	1922
		$e(n,n)$	42	30	29	33	16	19	21	18	28
		$e(n,p)$	24	74	36	18	40	34	24	42	50

Table 3.4 – Analysis of top-1 errors for (FT) and ($IL2M$) methods with memory $|\mathcal{K}| = 10000$ and $\mathcal{T} = 10$ states. p and n stand for past and new classes; c and e stand for correct and erroneous predictions. For instance $e(p,n)$ designates the number of wrong predictions of past classes as new ones.

	$\mathcal{T} = 10$				
$\|\mathcal{K}\|$	FT	$IL2M^1$	$IL2M^2$	$IL2M^{1+2}$	$IL2M$
20000	51.13	53.45	51.94	55.15	**56.37**
10000	42.29	47.64	43.63	49.57	**50.82**
5000	32.23	42.20	31.74	42.51	**44.05**

Table 3.5 – Top-1 average ILSVRC accuracy for different versions of $IL2M$ evaluated in the ablation study with $\mathcal{T} = 10$ states and memory $|\mathcal{K}| = \{20000, 10000, 5000\}$.

FT. The largest single contribution is the use of class means from the secondary memory $\mathcal{I}$ in $IL2M^1$. The gain is particularly interesting for the lower memory sizes, where the effect of catastrophic forgetting on FT is higher. The model level means have a small positive contribution for $|\mathcal{K}| = \{20000, 10000\}$ and a slight negative effect for $|\mathcal{K}| = 5000$. The final restriction of rectification has a moderate positive effect in all settings.

3.3.5 Conclusion

We introduce $IL2M$, a method designed for IL with memory. Extensive experiments show that $IL2M$ outperforms competitive algorithms which are either based on adapted fine tuning [24, 128] or fixed representations ($DeeSIL$). $IL2M$ gets better results than existing adapted fine-tuning-based methods for almost all configurations with memory and falls behind the fixed representation in a single case. The $IL2M$ ablation study from Subsection 3.3.4 shows that the obtained gain is mainly due to the use of the secondary memory $\mathcal{I}$ introduced here. The method has a negligible supplementary cost, both in terms of memory and computation. It is thus fitted for deployment in computationally constrained environments. Interestingly, the largest gains compared to FT, FT^{NEM} and FT^{BAL} are obtained for lower memory sizes. This makes $IL2M$ very interesting in real life since it reduces the memory requirements. We also find that, surprisingly, vanilla FT is a very effective baseline for class IL with memory. FT compares favorably with $DeeSIL$ and other existing algorithms [24, 128]. The ablation of the distillation component from $iCaRL$ and $E2EIL$ in FT^{NEM} and FT^{BAL} improves the performance of original methods. We test the proposed method and the baselines with three large-scale datasets and with different memory sizes. The reported results reduce the performance gap between incremental and non-incremental learning. However, this gap is still large, especially for the harder visual datasets like ILSVRC. We further investigate the reason behind the bias of the classification layer. We found that the classification weights matrix is the main reason behind the large scores gap between past and new classes. We deal with this problem in the next section.

3.4 ScaIL: Classifier Weights Scaling for Class-Incremental Learning

3.4.1 Introduction

Incremental learning algorithms strive to approach the performance of full learning, in which the entire training set is available for all classes at all times. In the previous section, we experimentally showed that when a bounded set of past exemplars is stored, a prediction bias toward new classes appears due to the data imbalance in their favor. This bias is illustrated in Figure 3.8(a) with the difference between mean raw predictions for past and new classes after incrementally fine tuning the $ILSVRC$ dataset [135] with $|\mathcal{K}| = 5000$ past exemplars. The average score difference in favor of new classes over all the incremental states is 6.45 points.

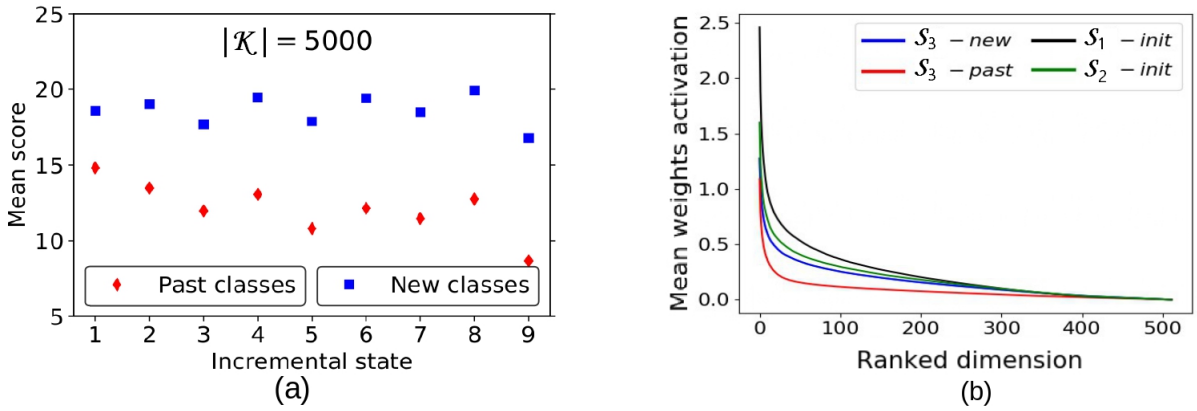


Figure 3.8 – (a) - Raw prediction scores of vanilla fine tuning for the ILSVRC dataset with $|\mathcal{K}| = 5000$ past exemplars and a total of $\mathcal{T} = 10$ states. Incremental states from 2 to 10 are represented. The initial state $\mathcal{S}_1$ is non-incremental and is not shown. (b) - Ranked mean weight activations of new and past classes in state $\mathcal{S}_3$ (blue and red) and mean weight activations of $\mathcal{S}_3$ past classes as initially learned in $\mathcal{S}_1$ (black) and $\mathcal{S}_2$ (green). *Best viewed in color.*

In $IL2M$, we directly rectify scores of past classes to make them more comparable to those of new classes. However, it would be interesting to dig deeper and understand what is happening. We further investigate the effect of catastrophic forgetting and discover that the bias is highly present in the weights matrix of the last fully connected layer. The prediction gap is due to the stronger activations of classifier weights for new classes compared to past classes, as illustrated by the blue and red curves from Figure 3.8(b). It is thus tempting to try to reshape the classification layers of past and new classes in order to make them more comparable.

We introduce $ScaIL$, a simple yet efficient method that reduces the bias in favor of new classes by exploiting the past classes embeddings as learned in their initial state with all class data available. Since past class classifiers are learned in different previous IL states, they are reshaped to be usable in the current state. Their scaling uses aggregate statistics from the current and initial states. In addition to the bounded exemplar memory $\mathcal{K}$, $ScaIL$ requires the use of a compact memory $\mathcal{I}$ which stores the classifier embeddings from the initial states of past classes. Similarly to $IL2M$ (Section 3.3), we simplify the deep model update across incremental states. The widely used distillation loss term [24, 49, 61, 128, 193] is again ablated here and model updates are done with vanilla fine tuning.

Evaluation is done with four public datasets and three values for the number of incremental states $\mathcal{T}$ and the exemplar memory $\mathcal{K}$, the two key components of class IL algorithms. $ScaIL$ is compared to strong baselines from literature and to new ones proposed in this chapter, and the obtained results indicate that it has the best overall performance.

3.4.2 Proposed Method

Initial weights replay

$ScaIL$ attempts to approximate full learning by exploiting past classifiers as learned in their initial state, with all images available. Since the deep models evolve during the incremental process, a transformation of the initial classifiers is needed for them to be usable in the current incremental state. $ScaIL$ is illustrated in Figure 3.9 with a toy example that includes an initial and two incremental states.

The main differences with existing IL algorithms which exploit a bounded memory are: (1) the introduction of a second memory $\mathcal{I}$ to store initial past class classifiers and (2) the ablation of the distillation loss. Note that the size of $\mathcal{I}$ is orders of magnitude smaller than that of $\mathcal{K}$ since it only stores hundreds of floating-point values per class instead of exemplar images. The immediate advantage of the method is that initial classifiers of past data are learned with all data. Initial classifiers learned with all images are stronger than the past classifiers learned only with exemplars in the current state. This is clearly visible in Figure 3.8(b) from the comparison of past classifiers weights as learned in the current third state (red) and the weights of the same classifiers learned in states $\mathcal{S}_1$ and $\mathcal{S}_2$ (black and green). We also note the activations of new classes become weaker as the incremental learning process advances. The new classes from $\mathcal{S}_1$ (black) are the strongest, followed by new classes from $\mathcal{S}_2$ (green) and those from $\mathcal{S}_3$ (blue).

The main challenge associated with $ScaIL$ is to combine classifiers originating from deep

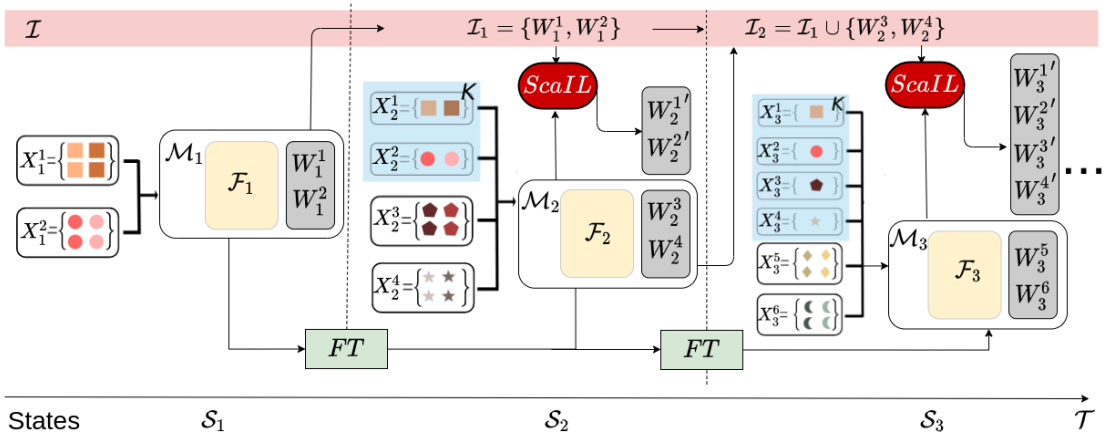


Figure 3.9 – Illustration of $ScaIL$. X_t^j is the data of the j^{th} class in state $\mathcal{S}_t$, $\mathcal{M}_t$ is the deep model of the state $\mathcal{S}_t$. W_i^j is the classifier of the j^{th} class in the initial state $\mathcal{S}_i$ in which the class was learned with all data, and $W_t^{j\prime}$ is the classifier of the j^{th} class in the state t after scaling it with $ScaIL$. We represent three states which recognize $2, 4$ and 6 classes respectively. The bounded memory (light blue) is fixed at $|\mathcal{K}| = 4$ past classes exemplars. As the training advances, the data imbalance between past and new classes grows due to the bounded memory $\mathcal{K}$ and the prediction bias in favor of new classes becomes more prominent. $ScaIL$ reduces this bias by making classifier weights of past and new classes more comparable by using a small memory $\mathcal{I}$ which stores initial classifiers W_i^j. In each IL state, $ScaIL$ replaces the raw classifiers of past classes provided by the model $\mathcal{M}_t$ by $W_t^{j\prime}$, a scaled version of W_i^j, the initial classifier. Since $ScaIL$ combines classifiers learned in different IL states, initial classifiers are reshaped using aggregate statistics from the current and the initial states. The classifiers for newly learned classes are left as learned by the current model $\mathcal{M}_t$. *Best viewed in color.*

models learned in different IL states. The reuse of initial classifiers in later incremental states is made possible by a fine tuning process with a memory of the past. This process results in partial preservation of the feature space even if the deep model evolves. In Subsection 3.4.5, we show that classifier reuse across states is impossible without memory during IL model updates.

Weights statistics computing

Once initial weights replay is performed, $ScaIL$ reshapes initial weights from $\mathcal{I}$ in order to make them comparable to those of newly learned classes in the feature space defined by the deep model of the current state. The scaling is based on weights statistics computed for new classes in each incremental state. We compute one mean vector per state and use it later for normalization. In Equation 3.2, we sort each new class embedding W_t^j based on the absolute

values of its elements.

$$\widehat{\boldsymbol{W}}_{\boldsymbol{t}}^{\boldsymbol{j}} = sort(|w_j^1|, |w_j^2|, ..., |w_j^d|, ..., |w_j^D|) \qquad ; j \in [N_{t-1}, N_t], d \in [1, D] \tag{3.2}$$

where $\widehat{\boldsymbol{W}}_{\boldsymbol{t}}^{\boldsymbol{j}}$ is the sorted version of the initial embedding of new class j in state $\mathcal{S}_t$. The use of absolute values is necessary since classifier weights activations can be positive or negative. Note that the matrix $\widehat{\boldsymbol{\mathcal{W}}}_{\boldsymbol{t}} = \{\widehat{\boldsymbol{W}}_{\boldsymbol{t}}^{\boldsymbol{N_{t-1}}}, \widehat{\boldsymbol{W}}_{\boldsymbol{t}}^{\boldsymbol{N_{t-1}+1}}, ..., \widehat{\boldsymbol{W}}_{\boldsymbol{t}}^{\boldsymbol{N_t}}\}$ is of dimensions (P_t, D), where $P_t = N_t - N_{t-1}$ is the number of new classes in $\mathcal{S}_t$ and D is the size of the feature vector.

Once the new classes embeddings are sorted descendingly based on their absolute values, we use Equation 3.3 to compute one mean vector per state that we use for normalization.

$$\mu_t^d = \frac{1}{P_t} \times \sum_{j=N_{t-1}}^{N_t} \widehat{w}_j^d \qquad d \in [1, D] \tag{3.3}$$

where μ_t (of dimension D) is the mean vector of the ranked new classes' embeddings in the state $\mathcal{S}_t$, and d is a dimension in the feature vector. Figure 3.8(b) shows that classifiers of each past state have different statistical distributions. To make class predictions from different states comparable, it is necessary to compute μ_t separately for each state. Note that each mean is computed using weights situated at the same rank for each classifier. For instance, μ_t^1 and μ_t^D will aggregate respectively the maximum and minimum weights of newly learned classes in $\mathcal{S}_t$.

Initial weights normalization

In state $\mathcal{S}_t$, $ScaIL$ transforms the past classifier weights as learned in their initial state using Equation 3.4.

$$w_j^{d\prime} = \frac{\mu_t^{R(d)}}{\mu_i^{R(d)}} \times w_j^d \tag{3.4}$$

$w_j^{d\prime}$ is the scaled version of w_j^d, the d^{th} dimension of the initial classifier W_i^j of the j^{th} past class. These weights are scaled using the ratio between the mean activation of new classes and that of past classes in their initial state. Each weight w_j^d is scaled using the mean activations of its corresponding rank, returned by function $R(\cdot)$, in the current and initial states $\mathcal{S}_t$ and $\mathcal{S}_i$. For instance, if the first weight ($d = 1$) of the embedding W_i^j is ranked 9^{th}, it will be scaled using the mean activations to the 9^{th} dimension of the mean ranked activations μ_t^9 and μ_i^9 respectively. This is done in order to preserve the relative importance of each classifier weight.

Figure 3.8(b) shows that $\mu_i^r > \mu_t^r$ for a given rank r. Consequently, $ScaIL$ scaling reduces the weights of the j^{th} class learned in its initial state to make it more comparable to classifiers

of new classes from the current state. The scaled classifier for each past class j of the current state $\mathcal{S}_t$ is written as $\boldsymbol{W}_t^{j\prime} = \{w_j^{1\prime}, w_j^{2\prime}, ..., w_j^{d\prime}, ..., w_j^{D\prime}\}$. The $ScaIL$ classification layer for $\mathcal{S}_t$ combines scaled classifiers for past classes and original classifiers for new classes. It can be written as $\boldsymbol{\mathcal{W}}_t' = \{\boldsymbol{W}_t'^{\mathbf{1}}, ..., \boldsymbol{W}_t'^{N_{t-1}}, \boldsymbol{W}_t^{N_{t-1}+1}, ..., \boldsymbol{W}_t^{N_t}\}$.

The features learned in $\mathcal{S}_t$ are fed into this scaled classification layer instead of the original one provided by $\mathcal{M}_t$. Note that only scores of the top-10 past classes are scaled as they code more information; the scores of the remaining past classes are set to zero. The choice of this value is experimental.

We illustrate the effect of $ScaIL$ on the prediction scores in Figure 3.10. Past classes have a slightly larger mean classification score in the first states and a lower one in subsequent states. While not completely aligned, the predictions of past and new classes in $ScaIL$ are much more balanced compared to those of raw fine tuning results from Figure 3.8(a).

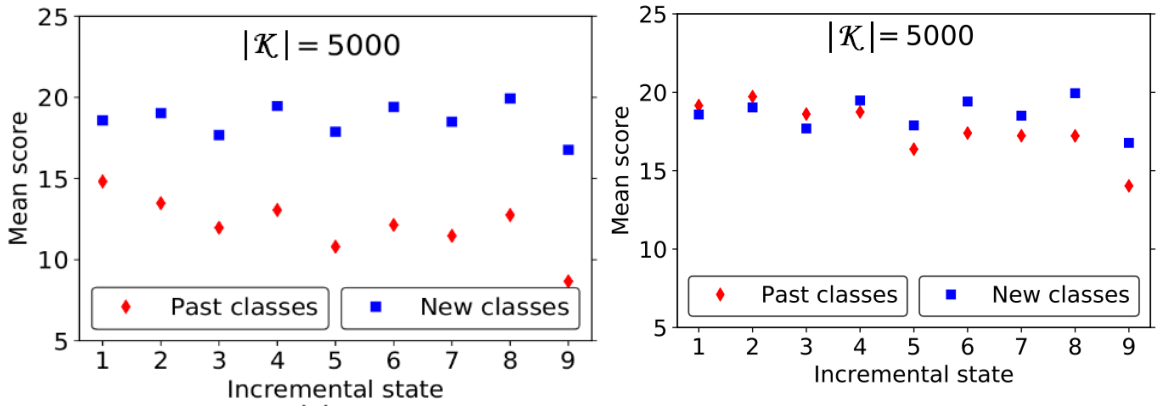


Figure 3.10 – Prediction scores before (left) and after (right) scaling for the ILSVRC dataset [135] with $|\mathcal{K}| = 5000$ exemplars and $\mathcal{T} = 10$ states. *Best viewed in color.*

3.4.3 Experiments

— **Datasets** (Appendix B) - ILSVRC [135], LANDMARKS [108], VGGFACE2 [22], and CIFAR-100 [74].

— **Memory sizes** - We fix the number of states $\mathcal{T} = 10$ and run experiments with a memory which amounts to approximately $2\%, 1\%, 0.5\%$ of the full training sets. Memory sizes are thus $|\mathcal{K}| = \{20000, 10000, 5000\}$ for ILSVRC, $|\mathcal{K}| = \{10000, 5000, 2500\}$ for VGGFACE2, $|\mathcal{K}| = \{8000, 4000, 2000\}$ for LANDMARKS and $|\mathcal{K}| = \{1000, 500, 250\}$ for CIFAR-100. Whenever a new incremental state is added, memory is updated using herd-

ing by inserting exemplars of new classes and reducing exemplars of past classes in order to fit the maximum size.

— **Incremental states** - We fix the memory to $|\mathcal{K}| = 0.5\%$ and test with $\mathcal{T} = \{20, 50\}$ in addition to $\mathcal{T} = 10$. The lowest memory size was selected since it is the most interesting configuration when memory budget is smallest.

— **Exemplar selection -** A herding mechanism [172], called Nearest-Exemplars-Mean (NEM) was introduced in $iCaRL$ for exemplar selection [128]. BiC uses the same herding mechanism. For this, we provide results for $ScaIL$ with and without herding. $ScaIL^{herd}$ is directly comparable with $iCaRL$ and BiC.

— **Evaluation measures -** Top-5 accuracy [135] and G_{IL} measure (Subsection 3.4.4).

— **Baselines** (Chapter 2) - $iCaRL$ [128], BiC [174], $DeeSIL$ (Section 3.2), FT (vanilla fine tuning), FT^{NEM} and FT^{BAL} from Subsection 3.3.3. In addition, we propose the following baselines:

➔ FT_{L2} - adds an L2-normalization layer to the raw classifier weights $\mathcal{W}_t$ given by model $\mathcal{M}_t$ to reduce bias in favor of new classes in current state $\mathcal{S}_t$.

➔ FT^{init} - the initial classifiers W_i^j of each past class replace the classifiers learned only with the past classes exemplars in $\mathcal{S}_t$. No transformation is applied to W_i^j. This is an ablation of the mean-related statistics from $ScaIL$.

➔ FT_{L2}^{init} - version of FT^{init} in which all classifiers are $L2$-normalized to make them more comparable.

3.4.4 G_{IL} evaluation measure

Since each algorithm is tested in a large number of configurations, we find it important to propose a summarized performance score. Inspired by works such as [127, 161], we propose a global score computed with Equation 3.5. G_{IL} measures the performance gap between each algorithm and an upper bound method. This upper bound is represented by $Joint$, a non-incremental learning with all data available.

$$G_{IL} = \frac{1}{C} \times \sum_{c=1}^{C} \frac{A(c) - A(Joint)}{A_{max} - A(Joint)} \tag{3.5}$$

where: C - number of tested configurations; $A(c)$ - top-5 score for each configuration (individual values of each row in Table 3.6); $A(Joint)$ - the upper-bound accuracy of the

dataset ($Joint$ in Table 3.6); A_{max} - the maximum theoretical value obtainable for the measure ($A_{max} = 100$ here).

G_{IL} estimates the average behavior of each algorithm with respect to the upper bound. The denominator is introduced to avoid a disproportionate influence of individual datasets in the aggregate score. G_{IL} is necessarily a negative number and the closer its value to zero, the better the method is. An ideal method, which reaches the upper bound value in all configurations, gives $G_{IL} = 0$. The proposal of aggregated measures is important for tasks that are evaluated in a large number of configurations [127, 161]. Building on previous work regarding such measures, the authors of [161] list eight criteria that should be met by global evaluation metrics when evaluating universal visual representations: (1) coherent aggregation, (2) significance, (3) merit bonus, (4) penalty malus, (5) penalty for damage, (6) independence to outliers, (7) independence to reference and (8) time consistency. They note that none of the global evaluation measures can fulfill all criteria simultaneously. However, their formulation, which inspired us to propose G_{IL} fulfills the maximum number of criteria.

While the IL context is different from that of universal representations, a majority of criteria from [161] are relevant here. The aggregation is easier in our work since the use of $Joint$ as a reference score is a natural upper bound for incremental learning algorithms. The aggregation of scores is natural in G_{IL} since all scores are compared to a single reference. The significance criterion, put forward in [127] is only implicitly modeled because configurations that give the largest gain contribute more to the global score. The merit bonus refers to the proportionality of the reward with respect to the reference method and is modeled through the denominator of Equation 3.5. The penalty for damage and the penalty malus are not applicable since all methods penalize the performance compared to the upper bound. The independence to outlier methods has a low effect in our case since it refers to the contributions of individual configurations. Since G_{IL} averages the contributions of a relatively large number of contributions, the risk related to outliers is rather reduced. Naturally, the more datasets and configurations are tested, the more robust the score will be. However, the computational resources needed for training in IL are large, and we consider that the use of four datasets with three memory sizes and three incremental learning splits gives a fair idea about the behavior of each algorithm. Time consistency is respected since methods are not compared to each other but only to a stable reference if the same deep model and data are used across time. The question remains whether datasets of different sizes should be given the same weights in the score, but weighting would further complicate the evaluation measure.

States	$\mathcal{T}=10$												$\lvert\mathcal{K}\rvert=0.5\%$								G_{IL}
Dataset	ILSVRC			VGGFACE2			LANDMARKS			CIFAR-100			ILSVRC		VGGFACE2		LANDMARKS		CIFAR-100		
$\lvert\mathcal{K}\rvert$	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	
$iCaRL^{herd}$	62.5	61.4	60.9	83.9	81.4	78.2	82.5	80.5	76.2	85.1	83.7	83.2	56.2	42.9	72.7	52.3	72.4	54.2	**73.2**	**55.7**	-16.75
BiC^{herd}	**85.5**	**82.8**	**79.7**	**97.3**	**96.6**	**95.7**	**97.9**	**97.3**	**96.6**	**88.8**	**87.6**	**83.5**	74.6	63.9	92.3	85.3	**94.7**	**90.5**	50.5	19.6	-4.03
$DeeSIL$	74.5	74.3	74.2	92.6	92.5	92.2	93.9	93.6	92.9	66.5	65.2	63.7	69.0	58.0	87.2	78.9	90.6	84.8	63.4	42.5	-7.10
FT	77.0	70.1	60.0	96.0	94.1	90.7	95.8	93.2	89.1	80.0	73.7	63.3	64.5	59.2	90.8	86.5	87.8	85.5	59.9	49.4	-6.40
FT^{NEM}	79.4	74.5	69.6	95.7	94.1	91.0	95.2	92.7	88.8	82.4	77.4	68.4	72.7	63.4	91.8	87.8	88.1	86.0	64.5	51.0	-6.01
FT^{BAL}	81.3	78.0	72.3	96.4	95.0	92.2	96.3	94.3	90.0	73.0	65.0	56.1	70.5	61.1	91.7	86.5	87.8	85.3	57.1	50.0	-5.98
FT_{L2}	81.4	77.6	72.1	96.5	95.1	92.4	96.2	94.4	91.4	81.8	77.2	69.1	73.4	66.7	92.8	88.8	89.8	87.1	63.2	49.9	-5.17
FT^{init}	68.9	66.5	61.2	95.9	95.3	94.5	96.5	95.0	92.7	79.3	77.3	73.7	53.4	39.0	**95.1**	90.3	90.6	87.5	60.7	40.1	-5.23
FT^{init}_{L2}	78.4	75.7	73.3	95.9	95.3	94.5	96.5	95.0	92.7	83.0	79.2	72.7	72.0	66.0	**95.1**	90.2	90.7	87.6	64.3	42.5	-4.67
$ScaIL$	81.0	78.2	75.1	96.4	95.6	94.5	96.9	95.3	92.7	84.6	81.1	74.9	73.9	68.3	94.5	90.5	90.7	88.2	67.9	47.7	-4.41
$ScaIL^{herd}$	82.0	79.8	76.6	96.5	95.8	95.2	97.3	96.0	94.0	85.6	83.2	79.1	**76.6**	**70.9**	95.0	**92.4**	92.6	90.4	69.8	51.0	**-3.71**
$Joint$	92.3			99.2			99.1			91.2			92.3		99.2		99.1		91.2		-

Table 3.6 – Top-5 average accuracy (%). Following [24], accuracy is averaged only for incremental states. The sizes of past memory $\mathcal{K}$ and number of states $\mathcal{T}$ are varied to evaluate the robustness of algorithms. $Joint$ is the non-incremental upper-bound performance obtained with all data available. The methods whose names include $herd$ exploit herding while the others are based on random exemplar selection. *Best results are in bold.*

3.4.5 Results and discussion

Confirming the conclusions of [128], $iCaRL$ has the best overall performance for CIFAR-100 in Table 3.6. For the three larger datasets, the FT consistently outperforms $iCaRL$. Overall, FT more than halves the gap with $Joint$ compared to $iCaRL$ ($G_{IL} = -6.40$ vs. $G_{IL} = -16.75$). The comparison to $E2EIL$ [24], which achieves 69.4% top-5 accuracy for ILSVRC with $|\mathcal{K}| = 2\%$ is equally favorable to FT [1]. Since one important difference between FT and existing IL methods is the use of distillation, we analyze its role separately later in this subsection.

The FT-based methods all have a positive contribution. FT^{NEM} and FT^{BAL} which are inspired by $iCaRL$ [128] and $E2EIL$ [24] improve over FT by less than 0.5 G_{IL} points. FT_{L2}, the L2-normalized version of the classifiers from the current IL state, provides a gain of 1.23 G_{IL} points compared to FT. Somewhat surprisingly, the direct concatenation of initial classifier weights from different states in FT^{init} also improves performance over FT by over 1 point. However, its performance for individual configurations is much more contrasted than that of FT_{L2}. FT^{init} has low results for the two object recognition datasets, which are on average more difficult than face and landmark recognition tasks. FT^{init}_{L2} adds $L2$-normalization to FT^{init} classifiers and ranks fourth among all methods tested, with 1.73 G_{IL} improvement over FT. The best overall result is obtained with $ScaIL^{herd}$, which improves FT performance by 2.69 points.

1. Note that a complete set of results is not presented for $E2EIL$ [24]. This method was not fully tested because we were not able to reproduce the results presented by the authors since the original implementation is based on Matlab, a non-free environment to which we do not have access.

The difference between $ScaIL$ and FT_{L2}^{init} in terms of G_{IL} is not large but still interesting. $ScaIL$ has the most stable behavior among all those tested. In fact, its performance on the three large datasets is most interesting for the smallest $\mathcal{K}$ values. This is the most challenging case and also the most interesting in practice since it requires a reduced memory for past data. The increase of the number of incremental state results in a drop of performance for all methods. With equal memory $\mathcal{K}$, the worst results are obtained for $\mathcal{T} = 50$ states, followed by $\mathcal{T} = 20$ and $\mathcal{T} = 10$. This finding confirms the results reported in [24] and [128]. It is probably an effect of a larger number of incremental rehearsal steps, which are applied for larger $\mathcal{T}$. Again, $ScaIL$ is the method that is the least affected by the change of the number of incremental states.

Contrarily to the conclusion of [24], the herding mechanism in $ScaIL^{herd}$ has positive effect compared to random selection of exemplars in $ScaIL$. Results show that, while BiC [174] is better for a lower number of incremental states ($\mathcal{T} = 10$), $ScaIL$ has better behavior for a larger number of states. Equally important, $ScaIL$ performance is less affected by the reduction of the memory size, and its performance is globally better for $|\mathcal{K}| = 0.5\%$, this leads to a better G_{IL} score for $ScaIL$. Finally, the need of BiC for a validation set to parametrize the bias correction layer makes it nonfunctional if no memory of the past is available.

The performance gap between $Joint$ learning and IL is naturally higher for more complex tasks, such as object recognition, compared to face and landmark recognition. For the last two tasks, classes have a more coherent visual representation and fewer examples are needed for a comprehensive representation of them. In the simplest configurations reported here ($\mathcal{T} = 10$, $|\mathcal{K}| = 2\%$), the best IL algorithms are less than three points behind $Joint$ for faces and landmarks. For such specialized tasks, incremental learning seems thus applicable in practice without a very significant performance loss. The situation is different for more complex tasks, such as object recognition, where significant progress is needed before IL algorithms approach the performance of classical learning.

An additional result concerns $DeeSIL$, the fixed representation method. Here, it is globally better than $iCaRL$, a finding which is at odds with the results originally reported in [128]. The difference is explained by the use of all data for each class, while past class training was unnecessarily restricted to $\mathcal{K}$ exemplars in [128]. FT outperforms $DeeSIL$ by less than 1 G_{IL} point. For $\mathcal{T} = 10$, $DeeSIL$ has very low dependence on the bounded memory size and could also be used in the absence of past exemplars memory. Naturally, its performance drops for larger $\mathcal{T}$ values because the initial model is learned with fewer classes but remains interesting.

Effect of distillation

The use of knowledge distillation in incremental learning with bounded memory was pioneered in iCaRL [128], which extends the work on IL without memory from [51]. Distillation was largely adopted afterwards [24, 49, 61, 68, 115, 193] as a way to reduce the effect of catastrophic forgetting. This adoption was based on one experiment presented in [128] which compared the performance of iCaRL and fine tuning only on the CIFAR-100 dataset and with single memory size. In Table 3.6, we report a similar finding for this dataset. For CIFAR-100, FT is probably less effective because it uses hard targets for loss minimization. These targets encode very sparse information for the small dataset available. In contrast, distillation exploits soft targets which encode more information [52] and is thus more fitted to work with small datasets. The results for $\mathcal{T} = 10$ states with different values of $|\mathcal{K}|$ support the above observation since the difference in favor of $iCaRL$ grows as $|\mathcal{K}|$ is reduced.

However, distillation hurts performance for all configurations tested for the three larger datasets, where FT has consequently better performance than $iCaRL$. The use of network outputs as soft targets for distillation was noted to produce a classification bias for past classes both in the original knowledge distillation paper [52] and in an incremental context [61]. A common assumption of distillation-based IL algorithms, first made in [51], is that the process starts with a powerful pretrained model which is trained on a large and balanced dataset. Under this condition, the soft targets used by the distillation loss are efficient to transfer knowledge to the next incremental state. Our hypothesis is that distillation tends to reinforce the errors due to data imbalance in the previous incremental state. In practice, if the distillation component is fed with soft targets whose predictions are wrong, it will push the classifier toward the wrong classes. To verify this hypothesis, we present an analysis of correct and erroneous predictions for past and new classes in Table 3.7 for vanilla fine tuning (FT) and fine tuning with distillation used as backbone in $iCaRL$ ($FT^{distill}$). We use $\mathcal{T} = 10$ states and $|\mathcal{K}| = 0.5\%$ exemplars for all datasets. For ILSVRC, the bias toward new classes (expressed by $e(p, n)$ errors) is similar with and without distillation. The correct predictions for new classes are also in a comparable range, although lower for $FT^{distill}$. This indicates that the data imbalance toward new classes has rather a comparable effect regardless of the use of distillation. The performance difference between the two methods is due mainly to confusion between past classes expressed by $e(p, p)$. They are roughly three times more frequent for $FT^{distill}$ compared to FT in Table 3.7. Equally important, while distillation is supposed to preserve accuracy for past classes, it clearly does not since the amount of correctly recognized past examples grows very steadily in $FT^{distill}$.

		Incremental states								
		$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$
ILSVRC										
FT	$c(p)$	2117	2995	3415	3875	3653	4451	4558	5003	3119
	$e(p,p)$	156	450	807	1363	1842	2710	2626	3932	2388
	$e(p,n)$	2727	6555	10778	14762	19505	22839	27816	31065	39493
	$c(n)$	4151	4322	4103	4141	4267	4304	4247	4378	4248
	$e(n,n)$	809	638	875	828	716	674	743	595	741
	$e(n,p)$	40	40	22	31	17	22	10	27	11
$FT^{distill}$	$c(p)$	850	1008	1355	1355	1195	1344	1419	1543	1562
	$e(p,p)$	472	1746	3700	4999	6904	8246	10771	13400	14556
	$e(p,n)$	3678	7246	9945	13646	16901	20410	22810	25057	28882
	$c(n)$	3645	3834	3597	3607	3744	3754	3605	3766	3662
	$e(n,n)$	1043	793	928	905	785	776	828	692	751
	$e(n,p)$	312	373	475	488	471	470	567	542	587
VGGFACE2										
FT	$c(p)$	4168	7718	11062	14293	15953	19614	21075	24690	24196
	$e(p,p)$	89	282	611	947	1354	2170	3203	3827	4929
	$e(p,n)$	743	2000	3327	4760	7693	8216	10722	11483	15875
	$c(n)$	4825	4834	4866	4865	4881	4879	4887	4874	4883
	$e(n,n)$	155	143	118	119	108	102	101	108	108
	$e(n,p)$	20	23	16	16	11	19	12	18	9
$FT^{distill}$	$c(p)$	1729	2109	1886	1787	1520	1657	1412	1199	1131
	$e(p,p)$	242	1455	2553	3360	4056	5766	6248	6506	7838
	$e(p,n)$	3029	6436	10561	14853	19424	22577	27340	32295	36031
	$c(n)$	4620	4637	4694	4740	4747	4714	4693	4685	4728
	$e(n,n)$	299	239	236	203	212	224	218	248	216
	$e(n,p)$	81	124	70	57	41	62	89	67	56
LANDMARKS										
FT	$c(p)$	1670	3072	4476	5550	6564	7626	8081	9303	10309
	$e(p,p)$	38	131	318	616	879	1005	1340	1961	2237
	$e(p,n)$	292	797	1206	1834	2557	3369	4579	4736	5454
	$c(n)$	1945	1970	1959	1956	1973	1966	1975	1973	1971
	$e(n,n)$	51	27	35	37	24	27	25	23	27
	$e(n,p)$	4	3	6	7	3	7	0	4	2
$FT^{distill}$	$c(p)$	901	1011	859	815	788	769	622	533	419
	$e(p,p)$	159	831	1770	2617	3194	3880	4708	5889	6744
	$e(p,n)$	940	2158	3371	4568	6018	7351	8670	9578	10837
	$c(n)$	1893	1893	1902	1910	1937	1913	1949	1926	1936
	$e(n,n)$	66	53	58	61	37	53	36	52	38
	$e(n,p)$	41	54	40	29	26	34	15	22	26
CIFAR-100										
FT	$c(p)$	366	614	675	605	686	950	779	692	467
	$e(p,p)$	10	181	312	288	641	974	835	732	601
	$e(p,n)$	624	1205	2013	3107	3673	4076	5386	6576	7932
	$c(n)$	791	873	886	866	848	859	834	888	915
	$e(n,n)$	196	114	103	131	146	127	159	104	80
	$e(n,p)$	13	13	11	3	6	14	7	8	5
$FT^{distill}$	$c(p)$	719	1160	1507	1706	1988	2195	2349	2404	2251
	$e(p,p)$	91	457	847	1210	1800	2551	2929	3499	3743
	$e(p,n)$	190	383	646	1084	1212	1254	1722	2097	3006
	$c(n)$	694	742	735	752	723	767	708	786	814
	$e(n,n)$	78	62	40	53	48	35	57	38	28
	$e(n,p)$	228	196	225	195	229	198	235	176	158

Table 3.7 – Top-1 correct and wrong classifications for vanilla fine tuning (FT) and fine tuning with distillation ($FT^{distill}$) for the four datasets with $\mathcal{T} = 10$ and $|\mathcal{K}| = 0.5\%$.

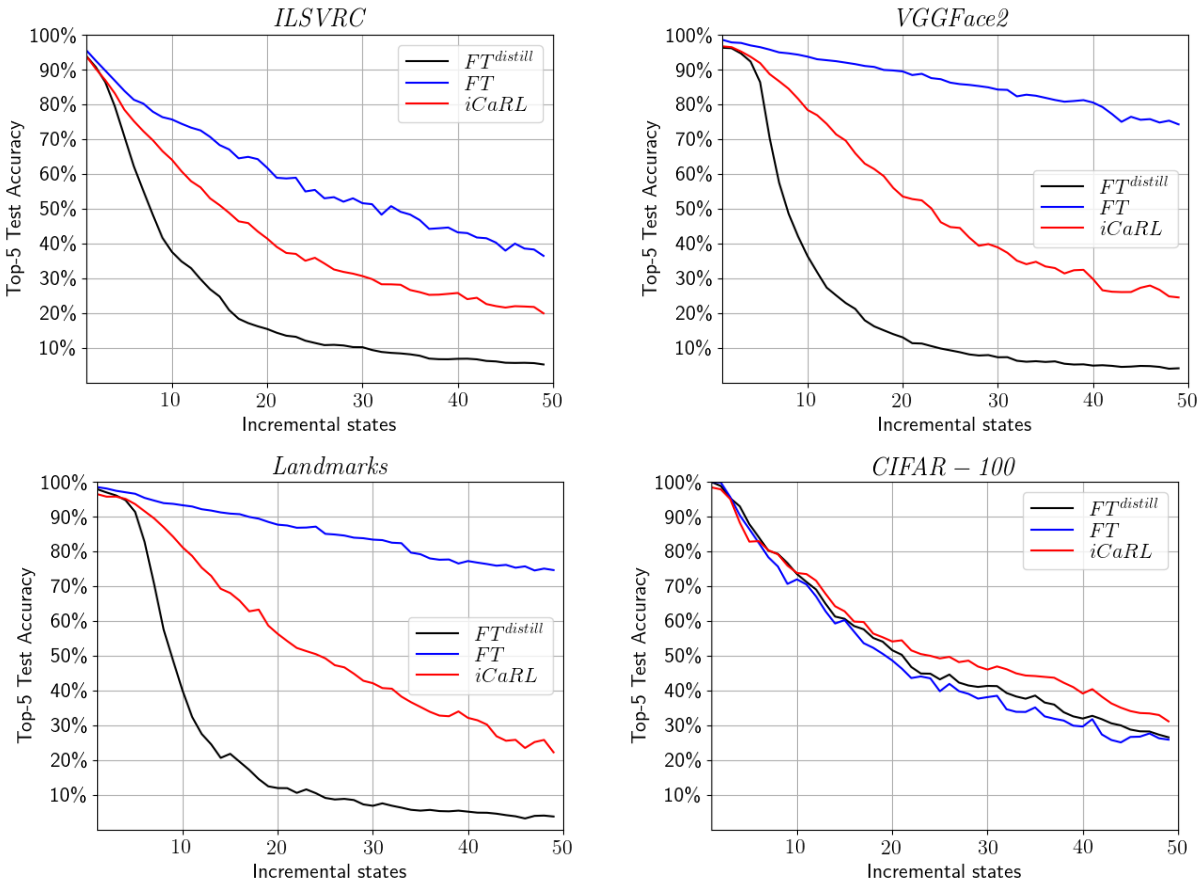


Figure 3.11 – Detailed Top-5 Test accuracy with $\mathcal{T} = 50$ and memory $|\mathcal{K}| = 0.5\%$. A comparison is done between FT, $FT^{distill}$ and $iCaRL$ to analyze the role of distillation.

In Figure 3.11, we provide detailed top-5 accuracy per incremental state for FT, $FT^{distill}$ and $iCaRL$ for $|\mathcal{K}| = 0.5\%$ and $\mathcal{T} = 50$ states. The largest value of $\mathcal{T}$ was chosen in order to observe the behavior with and without distillation for a small number of classes per incremental state. For the large datasets, the difference between FT and $FT^{distill}$ is small for initial incremental states, increases a lot afterward, and tends to decrease toward the end of the process but remains very large. This behavior is explained by the fact that, since past memory is tiny, the number of exemplars per class becomes very small toward the end. For instance, $\mathcal{K}$ includes 5000 images for ILSVRC and there will be only 5 exemplars per class in the last states. It is noticeable that rehearsal in FT still works with such a small number of exemplars. This finding provides further support to the results reported regarding the negative role of distillation at a large scale for imbalanced datasets when a memory of the past is available. Confirming the results from [128], distillation is indeed useful for CIFAR-100, where its performance is slightly better than that of FT. Also, the introduction of an external classifier in $iCaRL$ is clearly useful.

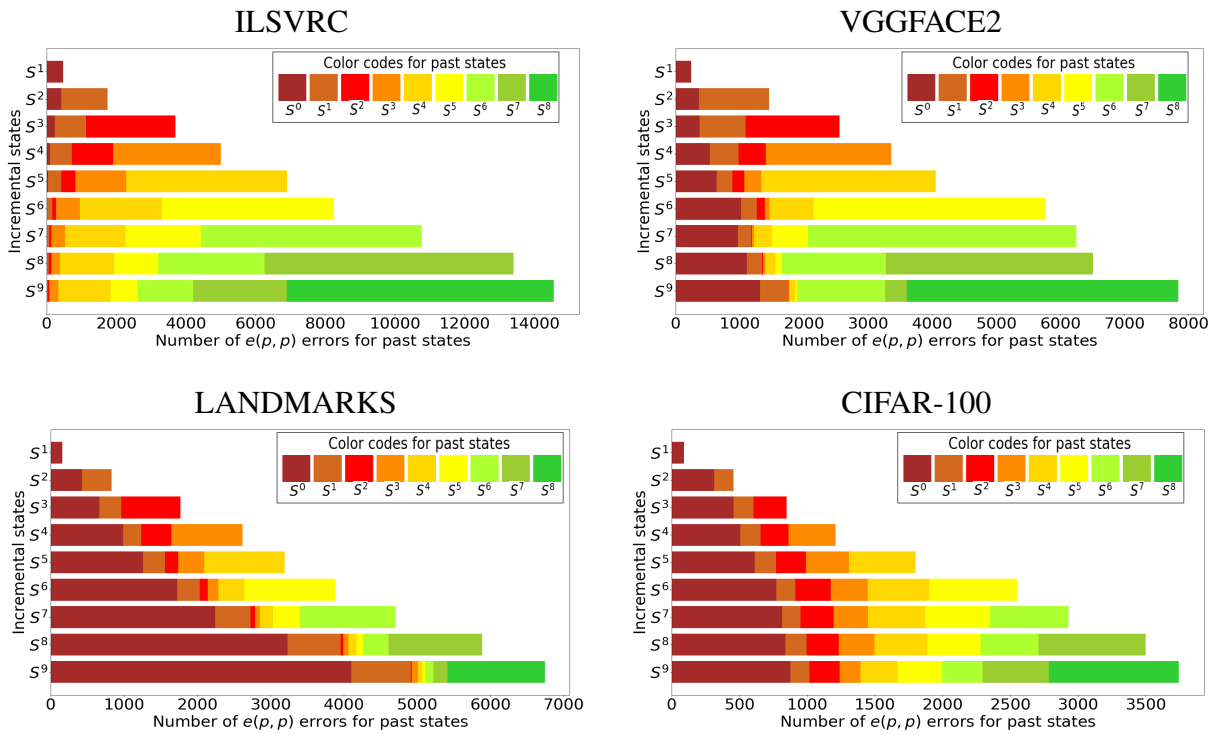


Figure 3.12 – Detail of past-past errors $e(p,p)$ for individual states of $FT^{distill}$ with $\mathcal{T} = 10$ and $|\mathcal{K}| = 0.5\%$. In each state, errors due to the latest past state are over-represented as a result of learning its associated state with an imbalanced training set. *Best viewed in color.*

In Figure 3.12, we present the distribution of $e(p,p)$ errors among individual past states for $FT^{distill}$. Since test data is balanced among states, the distribution of errors should also be approximately so. The $e(p,p)$ errors related to the last incremental state are overrepresented for all four datasets compared. However, the errors toward the first incremental state are also better represented for VGGFACE2 and even become dominant for LANDMARKS and CIFAR-100. This behavior is probably due to the fact that the initial state is stronger for easier tasks. In these cases, the model evolves to a lesser extent compared to ILSVRC, a more complex visual task.

Figure 3.12 shows that a majority of past test data for state $\mathcal{S}_t$ are predicted as belonging to classes that were new when first learned in $\mathcal{S}_{t-1}$. This result confirms that class imbalance has an important role for the distillation component of the loss, similarly to its influence on the classification component. We also notice that, except for $\mathcal{S}_6$, the number of errors grows for more recent past states. Along with imbalance, the number of rehearsals after the initial learning of the class also plays an important role in terms of distillation-related errors. Our findings indicate that vanilla FT is preferable to distillation-based FT as a backbone for large-scale IL with memory.

Results without memory

States	$\mathcal{T} = 10$			
Dataset	ILSVRC	VGGFACE2	LANDMARKS	CIFAR-100
LwF	43.80	48.30	46.34	**79.49**
FT	20.64	21.28	21.29	21.27
FT_{L2}	20.64	21.27	21.27	21.27
FT^{init}	**60.95**	**90.90**	**68.77**	55.05
FT^{init}_{L2}	51.57	76.84	61.42	47.48
$ScaIL$	21.96	23.06	22.31	33.49

Table 3.8 – Top-5 accuracy without memory ($|\mathcal{K}| = 0$) with $\mathcal{T} = 10$ states. We also present LwF [86], which is equivalent to $iCaRL$ [128] without memory. *Best results are in bold.*

Table 3.8 provides results obtained with fine tuning without memory for past classes ($|\mathcal{K}| = 0$) and with $\mathcal{T} = 10$ states. The accuracy drops significantly for FT since the network cannot rehearse knowledge related to past classes. Catastrophic forgetting is more severe and past classes become unrecognizable in the current state. The accuracy of FT without memory is mostly due to the recognition rate of new classes. When $\mathcal{T} = 10$, they represent between a half and a tenth of the total number of classes for states $\mathcal{S}_2$ and $\mathcal{S}_{10}$, the first and the last incremental state, respectively. The accuracy for past classes is close to random. Since $ScaIL$ depends heavily on the weights of past classes in the current state, its performance drops significantly. LwF [86] includes a distillation component that is clearly useful in the absence of memory. It outperforms FT and $ScaIL$ for all datasets by a very large margin. This finding reinforces the conclusions of [128] regarding the positive role of distillation in IL without memory.

3.4.6 Conclusion

We introduced $ScaIL$, a simple but effective IL algorithm that combines classifiers learned in different IL states to reduce catastrophic forgetting. It keeps the number of parameters of the network constant across IL states and requires a second memory whose size is negligible. $ScaIL$ provides an improvement over other methods and is also better than the new baselines. A consequent part of the performance improvement is due to the ablation of the distillation in IL algorithms. While widely used, we find that distillation is only useful for small-scale datasets. Our analysis indicates that a performance drop appears for large-scale datasets with memory when distillation is used. The drop is notably due to the inherently imbalanced character of datasets available in IL. In the next section, we will provide a comprehensive study of class-incremental learning methods with memory, including $DeeSIL$, $IL2M$, and $ScaIL$.

3.5 A Comprehensive study of Class-Incremental learning with memory

3.5.1 Introduction

The ability of artificial agents to increment their capabilities when confronted with new data is an open challenge in artificial intelligence. The main challenge faced in such cases is catastrophic forgetting, i.e., the tendency of neural networks to underfit past data when new ones are ingested. The first group of approaches tackles forgetting by increasing deep model capacity to accommodate new knowledge. The second type of approaches fixes the deep model size and introduces a mechanism whose objective is to ensure a good compromise between stability and plasticity of the model. While the first type of algorithms was compared thoroughly, this is not the case for methods that exploit a fixed size model. Here, we focus on the latter, place them in a common conceptual and experimental framework and propose the following contributions:

— propose a common evaluation framework which is more thorough than existing ones in terms of number of datasets, size of datasets, size of bounded memory, and number of incremental states

— examine the role of herding-based exemplar selection for past classes. Introduced in [172] and first used in an IL context by [128], its usefulness was questioned in [24, 61, 89] where it was reported to provide only marginal improvement compared to random selection. We run extensive experiments with the two selection algorithms and conclude that herding is useful for all methods tested.

— provide experimental evidence that it is possible to obtain interesting performance without the widely used knowledge distillation component [24, 53, 61, 89, 128, 174]. Instead, we use vanilla fine-tuning as a backbone for class IL with memory and model the problem as a case of imbalanced learning. The well-known thresholding method [21] is used to reduce the classification bias between past and new classes.

In this section, the focus is put on the components that differentiate algorithms one from another to facilitate the understanding of the advantages and limitations of each one of them. Moreover, we introduce promising combinations of components from different algorithms and assess their merits experimentally. In addition, we propose a thorough evaluation framework. Four public datasets designed for different visual tasks are used to test performance variability. Three splits in terms of the number of incremental states and three sizes for past memory are

tested to assess performance robustness for these IL parameters, which were previously identified as being the most important [24, 128]. We complement in Section 4.4 with experiments when no past memory is allowed because this setting has a strong influence on algorithm performance.

The main experimental finding here is that none of the existing class IL algorithms is better than the others in all experimental configurations. We find that both memory and incremental state sizes influence the relative performance of algorithms. Important differences arise notably if a bounded memory of past classes is allowed or not. We report such results in Chapter 4. These findings indicate that class-incremental learning remains an open research problem, and further research efforts should be dedicated to it.

3.5.2 Tested approaches

We compare recent class incremental algorithms and also adaptations of them, which combine components from different algorithms.

Fine-Tuning based IL algorithms

— ***iCaRL*** [128] exploits fine-tuning with distillation loss $\mathcal{L}^d$ to prevent catastrophic forgetting and a variant of Nearest-Class-Mean [103] to counter imbalance between past and new classes. The main difference with LwF is the introduction of a bounded memory to enable efficient replay.

— ***LUCIR*** [53] is based on fine-tuning with an integrated objective function. Authors propose the following contributions: (1) cosine normalization to balance magnitudes of past and new classifiers (2) less forget constraint to preserve the geometry of past classes, and (3) inter-class separation to maximize the distances between past and new classes. The combination of these contributions constitutes a more sophisticated take at countering catastrophic forgetting compared to the use of knowledge distillation from [24, 61, 128]. We experiment with two versions of this approach:

➔ $LUCIR^{NCM}$ - the original definition proposed by the authors where a Nearest-Class-Mean classifier is used.

➔ $LUCIR^{CNN}$ - the network outputs are used for classification.

— ***FT*** is the plain use of vanilla fine-tuning. The model $\mathcal{M}_t$ is initialized with the weights of the previous model $\mathcal{M}_{t-1}$ and only the cross-entropy loss $\mathcal{L}^c$ is used. FT constitutes

the simplest way to update models in incremental learning. It is heavily affected by catastrophic forgetting if a bounded memory is not available [128] but becomes an interesting baseline if memory is available (Section 3.3).

— ***FT***^{NEM} (Sections 3.3 and 3.4) is a version of FT which replaces the classifier $\mathcal{C}_t$ by a NEM classifier from [128]. FT^{NEM} is a modified version of $iCaRL$ in which the distillation loss $\mathcal{L}^d$ is ablated.

— ***FT***^{BAL} (Sections 3.3 and 3.4) is inspired by [24]. A vanilla FT is first performed, followed by a balanced FT. This second step aims to reduce bias between past and new classes by training on a version of $\mathcal{D}_t$, which stores the same number of images for past and new classes to obtain similar magnitudes for them. *FT*^{BAL} is also tributary to the fixed memory $\mathcal{K}$ because past exemplars are needed for the balancing step.

— ***BiC*** [174] adds a linear layer for bias removal, which is trained on a validation set separately from the rest of the model learned with cross-entropy and distillation losses. The objective of the supplementary layer is to reduce the magnitudes of predictions for new classes to make them more comparable to those of past classes. We note that $\mathcal{K}$ needs to be large enough to obtain reliable parameter estimations.

— ***ScaIL*** (Section 3.4) hypothesizes that the classification weights matrix $\mathcal{W}_t$ learned when classes were first streamed and learned with all data can be reused later. The main challenge is that deep models $\mathcal{M}_t$ are updated between incremental states. Normalization of the initial $\mathcal{W}_t$ is proposed to mitigate the effect of model updates and make past and new classes' predictions comparable.

— ***IL2M*** (Section 3.3) uses past classes' statistics to reduce the prediction bias in favor of new classes. Past classes' scores are modified using the ratio between their mean classification score when learned initially in the state $\mathcal{S}_i$ and in the current state $\mathcal{S}_t$. Furthermore, the ratio between the mean classification score over all classes in $\mathcal{S}_t$ and $\mathcal{S}_i$ is also used.

— ***FT***^{init}, ***FT***$^{init}_{L2}$, and ***FT***$^{init}_{L2+mc}$ (Sections 3.4 and 4.2) are methods built on top of FT in order to reduce the bias of the network towards new classes, where:

➔ $init$ - replaces the embeddings of past classes in the current state with their initial embeddings learned in the initial state with all available data.

➔ $L2$ - normalization that makes classifier weights more comparable across states.

➔ mc - state mean calibration defined as:

$$p_t^{j\prime} = p_t^j \times \frac{\mu(\mathcal{M}_t)}{\mu(\mathcal{M}_i^j)} \tag{3.6}$$

$\mu(\mathcal{M}_t)$ and $\mu(\mathcal{M}_i^j)$ - means of top-1 predictions of models learned in the current state and the initial state of the j^{th} class computed over their training sets.

The four components can be combined together, where first $init$ is applied, followed by $L2$ normalization, and finally, the state mean calibration mc. These methods are mostly interesting for IL without memory because they do not require the use of a past exemplars memory.

In addition we propose :

— ***FT***th, inspired by imbalanced learning [21], it implements fine tuning followed by threshold calibration (also known as threshold moving or post scaling). Thresholding adjusts the decision threshold of the model by adding a calibration layer at the end of the model during inference to compensate the prediction bias in favor of new classes:

$$p_t^{j'} = p_t^j \times \frac{|\mathcal{X}_t \cup \mathcal{K}|}{|X_t^j|} \tag{3.7}$$

where $\mathcal{K}$ is the bounded past classes' memory, $|X_t^j|$ is the number of training examples for the j^{th} class in the state $\mathcal{S}_t$, and $|\mathcal{X}_t^l \cup \mathcal{K}|$ is the total number of training examples in state $\mathcal{S}_t$. The memory $\mathcal{K}$ is needed to rectify past classes' scores.

Fixed-Representation based IL algorithms

Fixed-Representation (FR) [128] exploits the initial model $\mathcal{M}_1$ trained on the classes of $\mathcal{S}_1$ and freezes all its layers except the classification one in later incremental states. The frozen model is a limitation but also an advantage in that it allows the reuse of initial classifier layers, learned with all images throughout the entire incremental process. Unfortunately, the reuse of initial layers is not done in [128] and results are suboptimal. The method does not need a bounded memory for the update.

DeeSIL (Section 3.2) is a variant of FR in which the classification layer of DNNs is replaced by linear SVMs. *DeeSIL* is a straightforward application of a transfer learning [71, 125] scheme in an incremental context. The use of external classifiers is proposed because they are faster to optimize than an end-to-end FR.

REMIND [48] defines the model as $\mathcal{M}_t \equiv F(G(\cdot))$ where G is the fixed upper part of the network (the first 15 convolutional and 3 downsampling layers) and $F(\cdot)$ is the remaining layers (2 convolutional and 1 fully connected). Only $F(\cdot)$ is trained across incremental states in a streaming manner, while $G(\cdot)$ serves as a feature extractor. $REMIND$ relies on a Product Quantizer (PQ) [62] algorithm to store intermediate representations of images as compressed

	LwF^{init}_{*} (Sec.4.2)	$iCaRL$ [128]	$LUCIR$ [53]	FT (Sec.3.3)	FT^{NEM} (Sec.3.3-3.4)	FT^{BAL} (Sec.3.4)	BiC [174]	$ScaIL$ (Sec.3.4)	$IL2M$ (Sec.3.3)	FT^{th} [21]	FT^{init}_{*} (Sec.4.2)	FR [128]	$DeeSIL$ (Sec.3.2)	$REMIND$ [48]
(1)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×
(2)	✓	✓	✓	×	×	×	✓	×	×	×	×	×	×	×
(3)	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	×	×	×
(4)	NC	C	NC	NC	C	C	C	C	C	C	NC	NC	NC	NC

Table 3.9 – Main characteristics of tested approaches: **(1) model update** - indicates if the model is trained for each incremental state; **(2) distillation** - if this part of the loss is exploited to control catastrophic forgetting; **(3) bias removal** - is a separate component which is specifically dedicated to balancing scores between past and new classes, and **(4) memory usage** - if this component is compulsory (C) or not (NC). * refers to all methods built on top of $init$.

vectors for fast learning. The compact vectors are then reconstructed and replayed for memory consolidation. Note that compact vectors allow us to save much more past data than with raw images (for instance, all ILSVRC can fit in the memory when $|\mathcal{K}| = 20000$).

An overview of the tested algorithms and of their characteristics is presented in Table 3.9. The model update for each incremental state is widely used in existing approaches. Distillation is also used by a majority of algorithms from literature to counter the effect of catastrophic forgetting. Bias removal aims at balancing predictions for past and new classes. It is deployed either as a complement to distillation [24, 128, 174] or to replace it (Sections 3.3 and 3.4). Memory usage is compulsory for methods that rely heavily on exemplars of past classes. The dominant approach is based on model updating via fine-tuning to integrate new knowledge [24, 53, 174]. The performance of these algorithms depends heavily on the existence of a bounded memory of the past.

3.5.3 Experiments

— **Datasets** (Appendix B) - ILSVRC [135], LANDMARKS [108], VGGFACE2 [22], and CIFAR-100 [74].

— **Memory sizes** - We fix the number of states $\mathcal{T} = 10$ and run experiments with a memory which amounts to approximately $2\%, 1\%, 0.5\%$ of the full training sets. Memory sizes are thus $|\mathcal{K}| = \{20000, 10000, 5000\}$ for ILSVRC, $|\mathcal{K}| = \{10000, 5000, 2500\}$ for VGGFACE2, $|\mathcal{K}| = \{8000, 4000, 2000\}$ for LANDMARKS and $|\mathcal{K}| = \{1000, 500, 250\}$ for CIFAR-100.

— **Incremental states** - We fix the memory to $|\mathcal{K}| = 0.5\%$ and test with $\mathcal{T} = \{20, 50\}$ in addition to $\mathcal{T} = 10$.

— **Exemplar selection -** with and without herding (Subsection 3.4.3).

— **Evaluation measures -** Top-5 accuracy [135] and G_{IL} measure (Subsection 3.4.4).

3.5.4 Results and discussion

States	$\mathcal{T}=10$												$\lvert\mathcal{K}\rvert=0.5\%$								G_{IL}
Dataset	ILSVRC			VGGFACE2			LANDMARKS			CIFAR-100			ILSVRC		VGGFACE2		LANDMARKS		CIFAR-100		
$\lvert\mathcal{K}\rvert$	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	
iCaRL	79.3	76.5	71.0	96.0	95.3	93.9	95.1	94.0	91.8	66.5	56.1	47.9	55.9	45.0	88.5	78.2	86.8	82.4	35.5	35.4	-7.36
$LUCIR^{CNN}$	79.9	76.4	72.6	97.2	**96.9**	**96.5**	97.2	96.6	96.1	79.8	75.4	69.9	63.9	55.3	93.5	88.3	93.7	90.5	53.5	47.9	-4.13
$LUCIR^{NEM}$	80.5	80.0	79.4	96.2	96.0	95.7	95.4	94.9	94.4	82.6	80.8	78.8	73.6	66.3	92.7	87.9	91.9	89.8	69.0	63.0	-4.33
FT	79.4	74.4	65.9	96.4	94.5	91.3	96.6	94.7	91.4	82.4	77.9	70.7	69.4	64.3	91.6	89.2	90.9	89.0	64.3	54.8	-5.19
FT^{NEM}	81.4	79.0	75.0	96.4	95.4	94.0	96.1	94.6	92.6	85.1	81.7	76.0	76.5	69.0	94.0	91.1	91.9	89.9	68.8	55.9	-4.28
FT^{BAL}	84.0	80.9	76.5	97.0	95.7	92.4	96.9	95.3	92.2	80.0	74.0	69.0	75.9	67.1	92.3	89.5	91.2	88.9	62.9	54.2	-4.70
BiC	**85.5**	**82.8**	**79.7**	**97.3**	96.6	95.7	**97.9**	**97.3**	**96.6**	**88.8**	**87.6**	**83.5**	74.6	63.9	92.3	85.3	**94.7**	90.5	50.5	19.6	-4.03
ScaIL	82.0	79.8	76.6	96.5	95.8	95.2	97.3	96.0	94.0	85.6	83.2	79.1	76.6	70.9	**95.0**	**92.4**	92.6	90.4	69.8	51.0	-3.70
IL2M	80.9	78.1	73.9	96.7	95.4	93.4	96.5	94.7	92.5	81.8	77.0	71.2	70.9	60.6	92.5	88.4	90.8	88.1	61.5	51.0	-4.95
FT^{th}	84.3	82.1	78.3	97.2	96.3	94.8	97.2	95.8	94.0	86.4	83.9	79.1	**78.6**	**71.2**	94.3	91.6	92.9	**90.7**	**71.4**	**57.9**	**-3.62**
FT^{init}_{l2}	79.2	76.5	73.0	95.9	95.2	94.6	97.0	95.5	92.7	83.4	80.5	75.2	73.6	67.3	94.6	91.4	91.2	88.5	63.6	44.1	-4.43
FR	76.7	76.6	76.4	91.7	91.5	89.7	93.8	93.5	93.5	79.5	79.4	78.7	69.2	58.2	85.8	75.2	89.3	82.8	62.3	33.5	-7.62
DeeSIL	75.5	75.1	74.3	92.7	92.5	92.2	94.0	93.7	93.2	66.9	65.8	64.2	73.0	58.1	87.2	80.0	90.5	85.1	63.9	44.0	-6.92
REMIND	80.9	80.7	78.2	94.7	93.2	93.0	96.3	95.8	94.7	60.7	60.7	60.7	73.9	65.0	87.4	80.1	92.8	88.6	52.8	46.4	-6.02
Joint	92.3			99.2			99.1			91.2			92.3		99.2		99.1		91.2		-

Table 3.10 – Top-5 average incremental accuracy (%) for IL methods with herding using different values of $\mathcal{K}$ and $\mathcal{T}$. *Best results are in bold.*

Table 3.10 presents the performance of all algorithms tested in all experimental configurations when using herding for exemplar selection. Both the number of incremental states $\mathcal{T}$ and the bounded memory size $|\mathcal{K}|$ have a strong influence on results. The easiest configurations for all visual tasks are those including a large memory ($|\mathcal{K}| = 2\%$) and a small number of states ($\mathcal{T} = 10$). Inversely, the most difficult configuration combines a low memory ($|\mathcal{K}| = 0.5\%$) and a large number of states ($\mathcal{T} = 50$). This finding is intuitive insofar more exemplars for past classes enhance the quality of the replay for them, and a larger number of states makes IL more prone to catastrophic forgetting. However, the performance drop is more marked for the object recognition tasks (ILSVRC and CIFAR-100) compared to faces and landmarks recognition (VGGFACE2 and LANDMARKS). For instance, with $\mathcal{T} = 10$, the accuracy on ILSVRC drop for BiC is of 5.8 points when moving from $|\mathcal{K}| = 2\%$ to $|\mathcal{K}| = 0.5\%$ while the corresponding drop for VGGFACE2 is only of 1.6 points and the one for LANDMARKS is only of 1.3 points. The latter two tasks are simpler, and a smaller amount of exemplars can thus represent past classes.

The increase of $\mathcal{T}$, the total number of states, also has a detrimental effect on performance. For fine-tuning-based methods, the performance drop is explained by the fact that a larger number of retraining steps causes more information loss, and the effect of catastrophic forgetting is

increased. Also important, for methods like BiC, which need a validation set, the size of the latter becomes insufficient when $\mathcal{T}$ increases. This insufficiency is clearly illustrated by BiC results for $|\mathcal{K}| = 0.5\%$ and $\mathcal{T} = 50$. In this configuration, BiC performance drops more significantly than that of competing methods. The loss is most striking for CIFAR-100, the smallest of all datasets tested, where a performance of only 19.6% is obtained compared to 50.5% for $|\mathcal{K}| = 0.5\%$ and $\mathcal{T} = 20$. For fixed-representation methods, larger values of $\mathcal{T}$ decrease performance because the size of the first non-incremental state becomes smaller. Consequently, the fixed representation obtained from this state has lower generalization power and is less transferable to later states.

None of the methods is best in all configurations tested. On aggregate, the best results are obtained with FT^{th}, with a $G_{IL} = -3.62$ points loss compared to $Joint$, the classical learning upper-bound. The other methods with strong performance were all proposed recently: $ScaIL$ (Section 3.4) ($G_{IL} = -3.7$), BiC [174] ($G_{IL} = -4.03$) and $LUCIR^{CNN}$ [53] ($G_{IL} = -4.13$). The analysis of individual configurations shows that BiC has good performance in many of them. This method is best or second-best for the largest memory tested ($|\mathcal{K}| = 2\%$) and the smallest number of incremental states ($\mathcal{T} = 10$). However, its performance drops faster for the other values of $|\mathcal{K}|$ and $\mathcal{T}$. This is explained by its dependency on a validation set whose size becomes insufficient when $|\mathcal{K}|$ is low, and $\mathcal{T}$ is high.

The two $LUCIR$ variants have similar overall performance, with $LUCIR^{CNN}$ being globally better than $LUCIR^{NEM}$. This result confirms the original findings reported in [53]. The $iCaRL$ implementation from the same paper [53] has significantly lower performance than the two versions of $LUCIR$. The positive influence of inter-class separation and cosine normalization introduced in addition to standard knowledge distillation is thus confirmed.

Vanilla FT has lower performance than more recent methods but still much better than $iCaRL$, contrary to the comparison presented in [128]. However, the original comparison in that paper was biased since $iCaRL$ used memory while their version of FT was implemented without memory. All bias reduction methods applied to FT are beneficial, with FT^{th} being the best one followed closely by $ScaIL$. FT^{NEM}, which exploits the external classifier from [128] also has interesting performance and outperforms FT^{BAL} and $IL2M$. The lower performance of the last two methods is an effect of the fact that they are the most sensitive to memory reduction ($|\mathcal{K}| = 0.5\%$) and the growth of the number of states ($\mathcal{T} = 50$).

FR and $DeeSIL$, the fixed-representation based methods, behave worse than most FT-based approaches, with the only exception being that $DeeSIL$ is globally better than $iCaRL$. However, it is interesting to note that FR and $DeeSIL$ have a low dependency on memory size,

and their performance becomes competitive for $|\mathcal{K}| = 0.5\%$. In this latter setting, fine-tuning-based methods suffer more from catastrophic forgetting since memory becomes insufficient for an efficient replay of past classes. Globally, $DeeSIL$ has a better behavior than FR, especially for large-scale datasets. This confirms that the optimization of an external classifier is easier than that of the classification layer of a deep model. $REMIND$ performs better than FR and $DeeSIL$ for large datasets and has a better global score ($G_{IL} = -6.02$ VS. $G_{IL} = -7.62$ and $G_{IL} = -6.92$ respectively). However, its performance drops significantly compared to $DeeSIL$ when the number of states is increased from $\mathcal{T} = 10$ to $\mathcal{T} = 20$ and $\mathcal{T} = 50$.

For ILSVRC, $REMIND$ clearly outperforms many FT-based approaches such as $iCaRL$, $LUCIR$, $IL2M$ and FT variants except FT^{th} and FT^{BAL}. Streaming-based approaches like $REMIND$ have the advantage to run much faster than class-incremental-based approaches since they revisit each training example only once. However, the computational cost of $DeeSIL$ is still comparable to that of $REMIND$ since the SVMs training is fast. Equally important, $REMIND$ allows immediate evaluation since it learns the dataset images one by one. It is still usable in class incremental context since we can evaluate the model at the end of all training samples of each incremental state.

Role of exemplar selection

States	$\mathcal{T} = 10$												$\|\mathcal{K}\| = 0.5\%$								G_{IL}
Dataset	ILSVRC			VGGFACE2			LANDMARKS			CIFAR-100			ILSVRC		VGGFACE2		LANDMARKS		CIFAR-100		
$\|\mathcal{K}\|$	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	2%	1%	0.5%	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=20	$\mathcal{T}$=50	
iCaRL	77.9	73.0	65.3	95.3	93.8	91.1	93.9	91.4	87.4	64.5	53.4	43.9	51.3	40.9	84.3	73.2	81.9	76.4	32.6	33.4	-9.51
$LUCIR^{CNN}$	79.8	75.9	72.2	**97.3**	**97.0**	**96.6**	97.1	96.4	96.0	78.6	73.9	67.5	62.4	52.9	93.5	87.8	93.2	**89.2**	50.4	44.3	**-4.36**
FT	77.0	70.1	60.0	96.0	94.1	90.7	95.8	93.2	89.1	80.0	73.7	63.3	64.5	59.2	90.8	86.5	87.8	85.5	59.9	49.4	-6.40
BiC	**85.0**	**82.4**	**78.6**	**97.3**	96.8	96.1	**97.8**	**97.2**	**96.4**	**88.2**	**86.5**	**82.6**	72.1	59.9	92.0	82.9	**93.8**	88.1	54.2	18.1	-4.42
ScaIL	81.0	78.2	75.1	96.4	95.6	94.5	96.9	95.3	92.7	84.6	81.1	74.9	**73.9**	**68.3**	**94.5**	**90.5**	90.7	88.2	**67.9**	47.7	-4.41
IL2M	78.3	75.2	71.2	96.2	94.9	92.2	95.8	93.6	90.1	79.0	73.9	64.7	66.1	55.6	91.1	85.3	87.6	84.3	58.1	46.3	-6.22
FT^{th}	82.0	78.6	74.1	96.7	95.6	93.4	96.6	94.7	91.9	84.2	79.9	72.7	73.8	66.4	92.9	88.8	90.0	87.3	67.1	**52.7**	-4.85
FR	74.4	74.3	74.3	83.3	83.3	83.2	93.1	93.1	92.6	78.6	78.6	78.0	66.9	54.4	76.2	49.5	84.4	71.8	58.8	28.8	-12.41
DeeSIL	74.5	74.3	74.2	92.6	92.5	92.2	93.9	93.6	92.9	66.5	65.2	63.7	69.0	58.0	87.2	78.9	90.6	84.8	63.4	42.5	-7.09
Joint	92.3			99.2			99.1			91.2			92.3		99.2		99.1		91.2		-

Table 3.11 – Top-5 average incremental accuracy (%) for the main methods with random selection of exemplars for different values of $\mathcal{K}$ and $\mathcal{T}$. *Best results are in bold.*

As we mentioned, there is an ongoing debate concerning the effectiveness of herding-based versus random exemplar selection in IL [101, 24, 89]. We compare the two selection methods by providing results with random selection for the main algorithms evaluated here in Table 3.11. The obtained results indicate that herding has a positive effect on performance for most of the algorithms, albeit with a variable difference with respect to random selection. The best results in Table 3.11 are obtained with $LUCIR^{CNN}$, $ScaIL$ and BiC which have very close

G_{IL} performance. Among fine-tuning-based methods, $LUCIR^{CNN}$ and BiC are the methods that are least affected by the switch from herding to random exemplar selection. Both of these methods implement an end-to-end IL approach. $iCaRL$ has a more significant performance drop because it makes use of a NEM external classifier. This is a consequence of the fact that the classifiers are computed directly on the randomly selected exemplars.

Vanilla FT is more affected by the use of random selection than $LUCIR^{CNN}$ and BiC. The use of distillation for past data partly compensates for a poorer class representation with random exemplars. Since vanilla FT has lower performance, algorithms that build on it are also negatively affected. Among them, $ScaIL$ is the least affected because it exploits the initial classifiers of past classes. FT^{th} performance falls behind that of $LUCIR^{CNN}$, $ScaIL$, and BiC with random exemplar selection because exemplars have a more prominent role for learning past classes' representations. Thresholding with prior class probabilities is less efficient on poorer past class models.

The use of random selection has a small effect on FR and $DeeSIL$ because the exemplars are only used as negatives when new classifiers are trained. Their presence has a positive effect in that it allows slightly better separation between new and past classes across IL states. According to the authors of $REMIND$, many herding strategies were deployed based on distance from current example, number of times a sample has been replayed, and the time since it was last replayed, but all of the tested methods performed nearly the same as random selection, with higher computational time.

Role of knowledge distillation

In [49], authors hypothesize that distillation is useful when the teacher model is trained with a large and balanced dataset. This is not the case in IL due to the fact that the dataset progressively includes knowledge about more classes and that there is an imbalance between past and new classes. In spite of this observation, knowledge distillation is commonly used to tackle catastrophic forgetting [24, 49, 61, 68, 116, 127, 193]. Its use in IL with memory was encouraged by the experimental results presented in the influential $iCaRL$ paper [127]. There, the original comparison between FT and $iCaRL$ was not fair since the first method is implemented without memory, while the second exploits a memory of the past. The results reported in Table 3.10 for vanilla FT and methods built on top of it challenge the assumption that distillation loss $\mathcal{L}^d$ is necessary in IL with memory. These experiments show that FT is globally better since the G_{IL} score is over 2 points smaller than that of $iCaRL$. $iCaRL$ is more effective for ILSVRC and VGGFACE2 datasets only for a small number of incremental

states ($\mathcal{T} = 10$) and the smallest memory ($|\mathcal{K}| = \{1\%, 0.5\%\}$). FT^{NEM} is a version of $iCaRL$ without distillation. The results from Table 3.10 show that the use of the NEM classification layer further improves performance compared to vanilla fine-tuning. A detailed study of results without memory is presented in Section 4.4 of the next Chapter.

Additional experiment

IL Method	FT	FT^{th}	$iCaRL$	BiC	$LUCIR^{CNN}$	$LUCIR^{NCM}$	*TOPIC-AL*	*TOPIC-AL-MML*
Random	23.0	32.1	48.6	49.5	55.2	61.5	49.6	49.5
Herding	27.7	38.5	51.6	52.5	55.6	**62.6**		

Table 3.12 – Top-1 average incremental accuracy for IL methods for IMAGENET-100 with 60 initial classes and 8 incremental states containing each 5 classes ($\mathcal{T} = 9, P_1 = 60, P_{t\in[2,9]} = 5$). *The best result is in bold.*

We present a supplementary experiment that compares the performance of a very recent Neural Gas (NG) based approach to that of other methods. *TOpology-Preserving knowledge InCrementer* [162] relies on the NG network to preserve the feature space topology using a Hebbian learning [98]. Two variants of $TOPIC$ are tested: (1) *TOPIC-AL* - uses an *Anchor Loss* to stabilize the NG network in order to preserve past knowledge and (2) *TOPIC-AL-MML* - uses an *Anchor Loss* and also a *Min-Max Loss* to control the network growth while adapting it to new knowledge. They are compared to FT, FT^{th}, $iCaRL$, BiC, $LUCIR^{CNN}$, and $LUCIR^{NCM}$. Note that we use a single dataset because the authors of [162] did not provide their complete code and, while we tried to reproduce their results independently, that was not possible. Following [162], we use IMAGENET-100 [32], a subset of 100 classes extracted from the ILSVRC dataset, where each class contains 500 training images and 100 test images. To start with a good data representation, the first model $\mathcal{M}_1$ is trained on $P_1 = 60$ initial classes. The remaining 40 classes are divided in 8 incremental states containing each $P_t = 5$ new classes. The past classes' memory is set to $|\mathcal{K}_{t>1}| = 400 + 4 \times (N_t - P_1)$, where 400 images are divided equally between the first state classes, and 4 images per class are used for the classes that do not belong to the first state.

Table 3.12 provides top-1 accuracy of classical class IL approaches FT, FT^{th}, $iCaRL$, $LUCIR$ and BiC, and also results of $TOPIC$, the NG-based incremental learner, for IMAGENET-100. Results indicate that $LUCIR^{NCM}$ is the best approach, followed by $LUCIR^{CNN}$, BiC, $iCaRL$, $TOPIC$, FT^{th} and FT. The two variants of $TOPIC$ provide very similar performances with $TOPIC - AL$ being marginally better. These results are different from those

reported in [162], where $TOPIC$ was found to have better performance. This difference is probably partly explained by method parametrization choices and partly because herding was not exploited for $LUCIR$, BiC and $iCaRL$. We used the original parameters for the methods compared to $TOPIC$ in Table 3.12. The use of herding is beneficial for all methods compared to $TOPIC$, with important impact for FT^{th}, $iCaRL$ and BiC and lower impact for $LUCIR$ variants. $TOPIC$ is not affected by the use of herding since this selection is made by the NG component. Globally, the results show that, while interesting, the recent adaptation of neural gas approaches to class IL lags well behind the best methods tested in this chapter.

3.5.5 Conclusion

We conducted extensive experiments on recently proposed algorithms. The evaluation confirms that no algorithm is best in all configurations. When a memory is allowed, the best global result is obtained when IL is cast as a kind of imbalanced learning. This type of approach implements a vanilla FT backbone followed by a bias rectification layer. It is especially useful in the most challenging conditions (low memory and many IL states). If enough memory is available and the number of IL states is low, distillation-based approaches become competitive. The choice of the method will thus depend on the computation and storage capacities but also on the expected characteristics of the data stream, which needs to be processed. For fairness, we evaluated fixed-representation-based and fine-tuning-based methods with the same initial representation. If a larger pool of classes is available at the beginning of the process, the performance of fixed representations will be boosted because the initial representation generalizes better. However, fixed-representations work well only if the task does not change over time, as it is the case in the evaluated scenarios presented in [9]. The evaluation is done with four different datasets dedicated to distinct visual tasks. This setting can be reused and enriched to ensure robust testing of class IL algorithms. The comparison presented here shows that recently proposed approaches reduce the performance gap between non-incremental and incremental learning processes. However, it highlights a series of open problems which could be investigated in the future. First, handling class IL as an imbalanced learning problem provides very interesting results with [174] or without [13, 140] the use of a distillation component. Here, we introduced a competitive method where classification bias in favor of new classes is reduced by using prior class probabilities [21]. It would be interesting to investigate more sophisticated bias reduction schemes. Second, a more in-depth investigation of why distillation fails to work for large-scale datasets is needed. Finally, the results obtained with the herding-based selection of exemplars are better than a random selection for all tested methods.

3.6 Active Class Incremental Learning for Imbalanced Datasets

3.6.1 Introduction

Most existing IL algorithms assume that new data are readily labeled at the start of each incremental step. This assumption is strong since data labeling is a time-consuming process, even with the availability of crowdsourcing platforms. Two notable exceptions are presented in [8] and [120] where the authors introduce algorithms for self-supervised face recognition. While interesting, these works are applicable only to a specific task, and both exploit pretrained models to start the process. Also, minimal supervision is needed to associate a semantic meaning (i.e., person names) to the discovered identities. A second hypothesis made in incremental learning is that datasets are balanced or nearly so. In practice, imbalance occurs in a wide majority of real-life datasets but also in research datasets constructed in controlled conditions. Public datasets such as ImageNet [32], Open Images [72] or VGGFace2 [22] are all imbalanced. However, most research works related to ImageNet report results with the ILSVRC subset [135] which is nearly balanced.

These two hypotheses limit the practical usability of existing IL algorithms. We replace them with two weaker assumptions to make the incremental learning scenario more realistic. First, full supervision of newly streamed data is replaced by the possibility to annotate only a small subset of these data. Second, no prior assumption is made regarding the balanced distribution of new data in classes. We combine active and imbalanced learning methods to tackle the challenges related to the resulting IL scenario.

The main contribution of this work is to adapt the sample acquisition process, which is the core component of active learning (AL) methods, to incremental learning over potentially imbalanced datasets. A two phases procedure is devised to replace the classical acquisition process, which uses a single acquisition function. A standard function is first applied to a subset of the active learning budget in order to learn an updated model that includes a suboptimal representation of new data. In the second phase, a balancing-driven acquisition function is used to favor samples that might be associated with minority classes (i.e., those having a low number of associated samples). The data distribution in classes is updated after each sample labeling to keep it up-to-date. Two balancing-driven acquisition functions which exploit the data distribution in the embedding space of the IL model are introduced here. The first consists of a modification of the core-set algorithm [145] to restrain the search for new samples to data points that are closer to minority classes than to majority ones. The second function prioritizes samples that are close to the poorest minority classes (i.e., those represented by the minimum number of samples) and

far from any of the majority classes. The balancing-driven acquisition phase is repeated several times, and new samples are successively added to the training set in order to enable an iterative active learning process [147].

A secondary contribution is the introduction of FT^{th} (previously defined in Subsection 3.5.2). It is a backbone training procedure that considers incremental learning with memory as an instance of imbalanced learning. The widely used training with knowledge distillation [24, 53, 61, 128, 193] is consequently replaced by a simpler procedure which aims to reduce the prediction bias towards majority classes during inference [21]. Following the conclusions of this last work, initial predictions are rectified by using the prior class probabilities from the training set.

The proposed balancing-driven sample acquisition process is compared with a standard acquisition process, and results indicate that it has a positive effect for imbalanced datasets.

3.6.2 Related work

While we already discussed existing works from class-incremental learning in Chapter 2, we discuss here imbalanced and active learning areas and focus on those which are most closely related to our contribution.

Classical active learning is thoroughly reviewed in [147]. The first group of approaches exploits informativeness to select items for uncertain regions in the classification space. Uncertainty is often estimated with measures such as entropy [148], least confidence first [31] or min margin among top predictions [139]. Another group of approaches leverages sample representativeness computed in the geometric space defined by a feature extractor. Information density [147] was an early implementation of such an approach. *Core-set*, which rely on the classical *K-centers* algorithm to discover an optimal subset of the unlabeled dataset, was introduced in [145].

Recent active learning works build on the use of deep learning. The labeling effort is examined in [55] to progressively prune labels as labeling advances. An algorithm that learns a loss function specifically for AL was proposed in [180]. While very interesting, such an approach is difficult to exploit in incremental learning since the main challenge here is to counter data imbalance between new and past classes or among new classes. Another line of works proposes to exploit multiple network states to improve the AL process. *Monte Carlo Dropout* [41] uses softmax prediction from a model with random dropout masks. In [18], an ensemble approach that combines multiple snapshots of the same training process is introduced. These methods are not usable in our scenario because they increase the number of parameters due to the use of multiple models. We retain the use of the same deep model through incremental states to pro-

vide feature vectors and propose a stronger role for them during the sample acquisition process. Recently, [3] proposed a method which focuses on single-stage AL for imbalanced datasets. They exploit a pretrained feature extractor and annotate the unlabeled samples so as to favor minority classes.

Ideally, incremental updates should be done in a fully unsupervised manner [178] in order to remove the need for manual labeling. However, unsupervised algorithms are not mature enough to capture dataset semantics with the same degree of refinement and performance as their supervised or semi-supervised counterparts. Closest to our work are the self-supervision approaches designed for incremental face recognition [8, 120]. They are tightly related to unsupervised learning since no manual labeling is needed, except for naming the person. Compared to self-supervision, our approach requires manual labeling for a part of new data and has a higher cost. However, it can be applied to any class IL problem and not only to specific tasks such as face recognition as it is the case for [8, 120].

A comprehensive review [110] groups imbalanced object-detection problems in a taxonomy depending on their class imbalance, scale imbalance, spatial imbalance, or objective imbalance. The study shows the increasing interest of the computer vision community in the imbalanced problems for their usefulness in real-life situations.

3.6.3 Proposed approach

The proposed active learning adaptation to an incremental scenario (Figure 3.13) is motivated by the following observations:

— Existing acquisition functions ($\mathcal{AF}$s) were designed and tested successfully for active learning (AL) over balanced datasets. However, a wide majority of real-life datasets are actually imbalanced. Here, no prior assumption is made regarding the imbalanced or balanced character of the unlabeled data which is streamed in IL states. Unlike existing sample acquisition approaches which exploit a single $\mathcal{AF}$, we propose to split the process into two phases. The first phase uses a classical $\mathcal{AF}$ to-kick off the process. The second one implements an $\mathcal{AF}$ which is explicitly designed to target a balanced representation of labeled samples among classes.

— In IL, a single deep model can be stored throughout the process. This makes the application of recent ensemble methods [18] inapplicable. Following the usual AL pipeline, an iterative fine tuning of the model is implemented to incorporate labeled samples from the latest AL iteration.

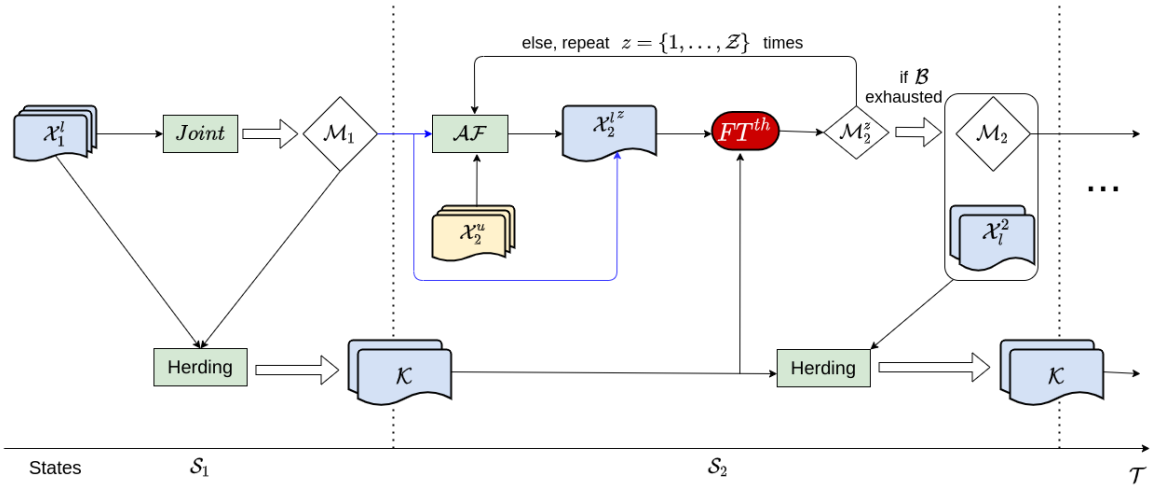


Figure 3.13 – Illustration of the proposed training process with one initial state $\mathcal{S}_1$, and one incremental state $\mathcal{S}_2$. The initial deep model $\mathcal{M}_1$ is trained from scratch on a fully-labeled dataset $\mathcal{X}_1^l$ using $Joint$ (a non-incremental learning). $\mathcal{M}_1$ and $\mathcal{X}_1^l$ are used to prepare the past class memory $\mathcal{K}$ for the next state using herding (a mechanism that selects the best representative past class images). State $\mathcal{S}_2$ starts with a sample acquisition function $\mathcal{AF}$ that takes the unlabeled set $\mathcal{X}_2^u$ and the model $\mathcal{M}_1$ as inputs. $\mathcal{AF}$ provides a part of the budget $\mathcal{B}$ annotated as $\mathcal{X}_2^{lz}$. The model $\mathcal{M}_1$ is then updated with data from $\mathcal{X}_2^{lz} \cup \mathcal{K}$ using FT^{th} (a fine tuning followed by a threshold calibration). The updated model $\mathcal{M}_2^z$ is again fed into the acquisition function $\mathcal{AF}$ with the rest of unlabeled examples from $\mathcal{X}_2^u$ to further annotate a part of the budget $\mathcal{B}$ and the model is updated afterward. This process is repeated $\mathcal{Z}$ times until $\mathcal{B}$ is exhausted. The model $\mathcal{M}_2$ is then returned with the annotated dataset $\mathcal{X}_2^l$ and the memory $\mathcal{K}$ is updated by inserting exemplars of new classes from $\mathcal{X}_2^l$ and reducing exemplars of past classes in order to fit its size. Note that the two blue arrows are applicable only in the first AL iteration (when $z = 1$). *Best viewed in color.*

— A memory $\mathcal{K}$ of past class samples is allowed and, following [174, 53], we model IL as an instance of imbalanced learning. The distillation component, which is central to most existing class IL algorithms [24, 61, 128, 174], is removed. Instead, we deal with imbalance by using a simple but efficient post-processing step that modifies class predictions based on their prior probabilities in the training set. The choice of this method is motivated by its superiority in deep imbalanced learning over a large array of other methods [21].

Problem Formulation

We build on the formulation we proposed in Section 2.2, and define some notations related to active learning:

— $\mathcal{B}$ - the labeling budget available for active learning

— $\mathcal{AF}$ - an acquisition function designed to optimize sample selection in active learning

— $\mathcal{Z}$ - the number of iterations done during active learning

— $\mathcal{X}_t^u$ - the unlabeled dataset associated to $\mathcal{S}_t$

— $\mathcal{X}_t^l$ - a manually labeled subset of $\mathcal{X}_t^u$,

We assume that all the samples are labeled in the first state. Active learning is deployed using $\mathcal{AF}(\mathcal{X}_t^u)$ to obtain $\mathcal{X}_t^l$, a labeled subset from $\mathcal{X}_t^u$. $\mathcal{X}_t^l$ data of the P_t new classes are available but only a bounded exemplar memory $\mathcal{K}$ for the N_{t-1} past classes is allowed. $\mathcal{M}_t$, the model associated to the state $\mathcal{S}_t$ is trained over the $\mathcal{X}_t^l \cup \mathcal{K}$ training dataset. An iterative AL process is implemented to recognize the set of classes $j \in [1, N_t]$

Active Learning in an Incremental Setting

We discuss the two phases of the adapted active learning process below. Classical sampling is followed by a phase that exploits the proposed balancing-driven acquisition functions.

Classical Sample Acquisition Phase. At the start of each IL state $\mathcal{S}_t$, an unlabeled dataset $\mathcal{X}_t^u$ is streamed into the system and classical AL acquisition functions are deployed to label $\mathcal{X}_t^l$, a part of $\mathcal{X}_t^u$, for inclusion in the training set. Due to IL constraints, the only model available at the beginning of $\mathcal{S}_t$ is $\mathcal{M}_{t-1}$, learned for past classes in the previous incremental step. It is used to extract the embeddings needed to implement acquisition functions. A number of acquisition functions were proposed to optimize the active learning process [147], with adaptations for deep learning in [18, 41, 145, 194]. Based on their strong experimental performance [18, 139, 145, 147], four $\mathcal{AF}$s are selected for the initial phase:

— **core-set sampling** [145] (*core* hereafter): whose objective is to extract a representative subset of the unlabeled dataset from the vectorial space defined by the deep embeddings. The method selects samples with:

$$x_{next} = \underset{x_u \in \mathcal{X}_t^u}{\operatorname{argmax}} \{ \min_{1 \leq k \leq n} \Delta(\boldsymbol{f}(\boldsymbol{x_u}), \boldsymbol{f}(\boldsymbol{x_k})) \} \tag{3.8}$$

where: x_{next} is the next sample to label, x_u is an unlabeled sample left, x_k is one of the n samples which were already labeled, $\boldsymbol{f}()$ is the feature vector extracted using $\mathcal{M}_{t-1}$ and Δ is the Euclidean distance between two embeddings.

— **random sampling** ($rand$ hereafter) : a random selection of images for labeling. While basic, random selection remains a competitive baseline in active learning.

— **entropy sampling** [147] (ent hereafter): whose objective is to favor most uncertain samples as defined by the set of probabilities given by the model.

$$x_{next} = \underset{x_u \in \mathcal{X}_t^u}{\operatorname{argmax}} \{ - \sum_{j=1}^{J} (p_t^j * \log(p_t^j)) \} \tag{3.9}$$

where p_t^j is the prediction score of x_u for the class j and J is the number of detected classes so far by AL.

— **margin sampling** [139] ($marg$ hereafter): selects the most uncertain samples based on their top-2 predictions of the model.

$$x_{next} = \underset{x_u \in \mathcal{X}_t^u}{\operatorname{argmax}} \{ max(p_t^1, .., p_t^j, .., p_t^J) - max_2(p_t^1, .., p_t^j, .., p_t^J) \} \tag{3.10}$$

where $max(\cdot)$ and $max_2(\cdot)$ provide the top-2 predicted probabilities for the sample x_u. This $\mathcal{AF}$ favors samples that maximize the difference between their top two predictions.

This acquisition phase is launched once at the beginning of each incremental state to get an initial labeled subset of the new data. This step is necessary to include the samples for the new classes in the trained model and initiate the iterative AL process.

Balancing-driven Sample Acquisition Phase. The second acquisition phase tries to label samples so as to tend toward a balanced distribution among new classes. The distribution of the number of samples per class is computed after each sample labeling to be kept up-to-date. The average number of samples per class is used to divide classes into minority and majority ones. These two sets of classes are noted $\mathcal{C}_t^{mnr}$ and $\mathcal{C}_t^{maj}$ for incremental state $\mathcal{S}_t$. Two functions are proposed to implement the balancing-driven acquisition:

— **balanced core-set sampling** ($b - core$ hereafter) is a modified version of $core$ presented in Equation 3.8. $b - core$ acts as a filter that keeps candidate samples for labeling only if they are closer to a minority class than to any majority class. We write the relative distance of an unlabeled image w.r.t. its closest minority and majority classes as:

$$\Delta_{\frac{mnr}{maj}}(x_u) = \min_{c_t^{mnr} \in \mathcal{C}_t^{mnr}} \Delta(\boldsymbol{f(x_u)}, \boldsymbol{\mu(c_t^{mnr})}) - \min_{c_t^{maj} \in \mathcal{C}_t^{maj}} \Delta(\boldsymbol{f(x_u)}, \boldsymbol{\mu(c_t^{maj})}) \tag{3.11}$$

where: x_u is an unlabeled sample, c_t^{mnr} and c_t^{maj} are classes from the minority and majority sets $\mathcal{C}_t^{mnr}$ and $\mathcal{C}_t^{maj}$ respectively, $f(x_u)$ is the feature vector of x_u extracted from the latest deep model available, $\mu(c_t^{mnr})$ and $\mu(c_t^{maj})$ are the centroids of minority and majority classes c_t^{mnr} and c_t^{maj} computed over the embeddings of their labeled samples.

The next sample to label is chosen by using the core-set definition from Equation 3.8 but after filtering remaining unlabeled samples with Equation 3.11:

$$x_{next} = \underset{x_u \in \mathcal{X}_t^u \text{ and } \Delta_{\frac{mnr}{maj}}(x_u) < 0}{\operatorname{argmax}} \{ \min_{1 \leq k \leq n} \Delta(\boldsymbol{f(x_u)}, \boldsymbol{f(x_k)}) \} \tag{3.12}$$

— **poorest class first sampling** ($poor$) is an acquisition function that gives priority to the class represented by the minimum number of labeled samples associated to it at a given moment during active learning. If there are several such classes, one of them is selected randomly. The method translates the hypothesis that samples which are close to a poor class and far from any majority class should be favored in order to achieve a more balanced distribution. The next candidate for labeling is selected with:

$$x_{next} = \underset{x_u \in \mathcal{X}_t^u}{\operatorname{argmin}} \{ \Delta(\boldsymbol{f(x_u)}, \boldsymbol{\mu(c_t^{poor})}) - \min_{\forall c_t^{maj} \in \mathcal{C}_t^{maj}} \Delta(\boldsymbol{f(x_u)}, \boldsymbol{\mu(c_t^{maj})}) \} \tag{3.13}$$

where c_t^{poor} is a minority class from $\mathcal{C}_t^{mnr}$ which has the lowest number of samples in the current labeled subset.

$poor$ is similar in spirit to $b - core$ but has a stronger drive towards balancing because an individual class with the poorest representation is targeted instead of samples that are close to any minority class.

In an iterative active learning scenario, the balancing-driven acquisition can be repeated several time until the AL budget $\mathcal{B}$ is exhausted.

Imbalance-driven Incremental Learning

The model update within each incremental state is inspired by a usual iterative AL approach [147] which includes a classical acquisition phase at the beginning and several balancing-driven iterations. For a total of $\mathcal{Z}$ active learning iterations in each state $\mathcal{S}_t$, intermediate models $\mathcal{M}_t^1$, ..., $\mathcal{M}_t^z$, ..., $\mathcal{M}_t^{\mathcal{Z}-1}$ are created while annotating $\mathcal{X}_t^{l^1}$, ..., $\mathcal{X}_t^{l^z}$, ..., $\mathcal{X}_t^{l^{\mathcal{Z}-1}}$ during the first $\mathcal{Z} - 1$ iterations before obtaining the final $\mathcal{M}_t$. The number of iterations $\mathcal{Z}$ and the size of each iteration can take different values. The choice of a particular setting is made empirically so

as to: (1) have enough new samples in the initial iteration in order for the new classes to be trainable in $\mathcal{M}_t^1$, i.e., the model $\mathcal{M}_t$ in the first iteration, (2) have enough candidates left for the balancing-driven iterations and (3) do not repeat the fine tuning process too many times to keep the incremental update timely. $\mathcal{M}_{t-1}$ is used to extract embeddings if $core$ is used in the initial AL iteration. Note that while iterative training increases the level of forgetting in IL, it is needed in AL to update model representation while annotating the images [147].

As we mentioned, we depart from the usual modeling of the IL problem [24, 53, 128, 174] which exploits knowledge distillation to counter catastrophic forgetting. Following the recent observation that a simpler fine tuning based approach gives interesting results (Section 3.3), we use an IL backbone inspired by imbalance learning results presented in [21]. This backbone is called fine tuning with thresholding (FT^{th} below), also known as threshold moving or post scaling [21]. Thresholding adjusts the decision threshold of the model. It consists of the addition of a calibration layer at the end of the model during inference to compensate for the prediction bias in favor of majority classes. This layer rectifies the class prediction p_t^j of the j^{th} class in the state $\mathcal{S}_t$ as already explained in Subsection 3.5.2 (Equation 3.7).

FT^{th} boosts the scores of classes with a lower number of associated samples. The method has the interesting property of dealing with the imbalance in IL in a uniform manner. It does not matter whether imbalance comes from the distribution of newly streamed data or from the fact that only a bounded memory of past classes is available. This stands in contrast with knowledge distillation which handles imbalance for past classes but not among new ones. FT^{th} is competitive against state-of-the-art algorithms. In a classical (i.e. fully supervised) IL setting, it has 59.59 top-1 accuracy for CIFAR-100, compared to $iCaRL$ [128] (57.35) and $LUCIR$ [53] (55.36). More results are provided in Subsection 3.5.4 (top-5) and also in the next section (top-1).

3.6.4 Experiments

Datasets

Experiments are run with four public datasets, out of which three are imbalanced and one is balanced. We provide the coefficient of variation $cv = \frac{\sigma}{\mu}$, with σ the standard deviation and μ the mean of the distribution of samples per class. cv provides information about the degree of imbalance associated with each dataset. The larger this value is, the more imbalanced the dataset will be.

The used datasets are: IMAGENET-100 ($cv = 0.7523$), FOOD-100 ($cv = 0.7940$), FACES-

100 ($cv = 0.7216$), and CIFAR-100 ($cv = 0$). More details are in Appendix B.

Incremental Learning Setting

We run the experiments with $\mathcal{T} = 10$ states for each dataset. This setting is classically used in class incremental learning [24, 128]. A total of $|\mathcal{K}|$ images of past classes are kept at any time during incremental learning. $\mathcal{K}$ approximates 2% of the full training sets. Memory sizes are thus $|\mathcal{K}| = 1000$ for IMAGENET-100 and CIFAR-100, $|\mathcal{K}| = 465$ for FACES-100 and $|\mathcal{K}| = 450$ for FOOD-100. At the end of each incremental state, memory is updated by inserting exemplars of new classes and reducing exemplars of past classes in order to fit its maximum size. Note that since $\mathcal{K}$ is bounded and the number of past classes grows, the imbalance in favor of new classes grows for later incremental states and the problem becomes more challenging. The exemplars are chosen using the herding mechanism introduced in [128]. The herding procedure consists in choosing the set of images that best approximates the real mean of the class.

Active Learning Process

Three active learning budgets are tested covering $\mathcal{B} = \{20\%, 10\%, 5\%\}$ of the unlabeled dataset $\mathcal{X}_t^u$ streamed in state $\mathcal{S}_t$. These different values are used to get a comprehensive view of the behavior of each configuration. Active learning is implemented with a usual iterative approach [145, 147] including $\mathcal{Z} = 4$ iterations, 40% of $\mathcal{B}$ are used for classical acquisition and three times 20% of $\mathcal{B}$ for balancing-driven acquisition (values were experimentally chosen). Classical and balancing-driven acquisition phases are independent of one another and we test all their combinations. For completeness, we include results with a baseline in which both phases are implemented with random sampling. Note that the proposed acquisition functions are non-deterministic and experiments are run five times for each configuration in order to have a robust estimation of its performance. To improve comparability of configurations which use the same initial $\mathcal{AF}$, the same initial models are used for all subsequent balancing-driven $\mathcal{AF}$s.

Upper Bound Methods

In addition to the active learning configurations, we present results with:

— sIL - usual supervised incremental learning in which all samples are labeled (equivalent to $\mathcal{B} = 100\%$).

— $Joint$ - classical non-incremental learning in which all samples are provided at once.

For comparability, sIL and $Joint$ are both trained using threshold calibration. sIL is an incremental upper bound for active learning configurations since it is fully supervised. $Joint$ is an upper bound for sIL since all the data are labeled and available at once. These upper bounds are useful insofar they provide information about the performance gap due to partial labeling of streamed data.

3.6.5 Results and discussion

FT$^{\mathbf{th}}$ in supervised mode - Instead of handling catastrophic forgetting [102] as previous works did [24, 53, 127, 174], we address IL with bounded past memory as an imbalanced learning problem. We use threshold calibration [21] to rectify scores in order to give more chances to minority classes to be selected during inference. The comparison to recent IL methods in supervised mode from Table 3.13 indicates that FT^{th} is competitive. It clearly outperforms $iCaRL$ [128] and $IL2M$ (Section 3.3) and is better than LUCIR [53] for three datasets out of four. We also provide the results of vanilla fine tuning before threshold calibration to underline the usefulness of thresholding. It has a positive effect for all four datasets, a finding which validates its usefulness in our scenario.

Dataset	FT	FT^{th} [21]	$LUCIR$ [53]	$iCaRL$ [128]	$IL2M$ (Section 3.3)
IMAGENET-100	54.80	**61.42**	60.77	52.40	57.68
FACES-100	69.11	73.26	**78.44**	60.48	70.33
FOOD-100	30.21	**34.79**	25.70	21.99	32.20
CIFAR-100	50.98	**59.59**	55.36	57.35	54.24

Table 3.13 – Top-1 average supervised IL accuracy (%). *Best results are in bold.*

Active Learning - The experimental results obtained with FT^{th} for the proposed active incremental learning scenario are presented in Table 3.14. The comparison of classical $\mathcal{AF}$s ($rand$ -$rand$ and $core-core$ in Table 3.14) indicates that random sampling outperforms the $core-set$ sampling in a majority of cases. This result is at odds with the one reported in [145] but is in line with the findings of [18, 41] that random sampling in AL is a strong baseline and is actually better than the recent core-set method from [136]. The authors of this last paper also report that random sampling has better performance for lower active learning budgets, which are studied

Dataset	$\mathcal{B}$	*rand*			*core*			*ent*			*marg*			*sIL*	*Joint*
		rand	*poor*	$b-core$	*core*	*poor*	$b-core$	*ent*	*poor*	$b-core$	*marg*	*poor*	$b-core$		
IMAGENET-100	20%	57.48 ±0.50	**58.65** ±0.23	58.08 ±0.40	56.85 ±0.23	57.25 ±0.84	57.46 ±0.52	45.07 ±0.58	56.53 ±0.27	56.23 ±0.22	54.13 ±0.66	56.39 ±0.53	56.26 ±0.45	61.42	72.48
	10%	52.61 ±0.45	**54.89** ±0.53	53.40 ±0.26	52.09 ±0.41	53.55 ±1.21	52.22 ±1.13	42.15 ±0.43	51.91 ±0.51	51.81 ±0.76	46.26 ±1.24	51.40 ±0.89	50.61 ±0.36		
	5%	47.72 ±0.69	**48.71** ±0.97	48.18 ±0.56	46.01 ±0.51	47.39 ±0.85	46.45 ±0.30	37.95 ±0.61	45.10 ±1.46	44.70 ±1.02	36.74 ±1.05	44.18 ±1.09	43.93 ±0.93		
FACES-100	20%	65.91 ±0.94	66.41 ±0.10	**67.24** ±0.36	66.41 ±0.66	66.46 ±0.46	66.94 ±1.37	48.62 ±0.95	63.30 ±0.33	64.99 ±0.80	59.51 ±1.17	62.85 ±1.75	65.27 ±0.53	73.26	93.62
	10%	58.40 ±0.71	**59.13** ±1.66	58.92 ±1.05	55.82 ±4.70	58.76 ±2.93	57.26 ±2.90	42.12 ±1.38	54.93 ±1.07	55.45 ±1.47	49.32 ±1.40	54.82 ±2.19	58.69 ±1.52		
	5%	48.38 ±1.27	50.09 ±2.30	**50.12** ±0.96	48.74 ±1.21	47.71 ±1.28	50.04 ±2.04	35.61 ±0.51	45.79 ±0.83	45.39 ±1.13	38.37 ±1.02	45.90 ±2.31	45.64 ±1.56		
FOOD-100	20%	28.67 ±0.42	**28.89** ±0.43	28.60 ±0.52	28.24 ±0.42	27.88 ±0.34	27.98 ±0.46	23.72 ±1.00	27.99 ±0.41	27.51 ±0.54	28.13 ±0.59	28.18 ±0.35	27.56 ±0.24	34.79	62.53
	10%	24.12 ±0.47	**24.17** ±0.56	24.07 ±0.68	22.91 ±0.63	23.46 ±0.12	23.07 ±0.31	19.41 ±0.96	22.32 ±0.73	22.25 ±0.64	23.35 ±0.64	22.68 ±0.81	22.84 ±0.52		
	5%	20.51 ±0.61	19.10 ±0.68	**20.63** ±0.46	19.22 ±0.36	19.17 ±0.58	18.79 ±0.64	16.80 ±0.75	18.66 ±0.31	18.57 ±0.47	18.62 ±0.48	18.79 ±0.75	18.41 ±0.82		
CIFAR-100	20%	**49.47** ±0.16	49.36 ±0.33	48.46 ±0.75	46.75 ±0.40	46.87 ±0.19	46.87 ±0.36	39.76 ±1.30	46.66 ±0.29	47.69 ±0.23	46.07 ±0.31	45.37 ±0.39	46.68 ±0.44	59.59	76.98
	10%	**45.49** ±0.61	45.23 ±1.17	44.83 ±0.29	41.76 ±0.54	42.60 ±0.77	42.04 ±0.77	34.87 ±0.66	42.64 ±0.50	43.76 ±0.55	39.92 ±0.43	40.94 ±0.31	41.82 ±0.35		
	5%	**41.58** ±0.29	40.69 ±0.23	39.69 ±0.46	35.23 ±0.64	37.70 ±0.46	35.72 ±0.67	31.74 ±0.74	37.68 ±0.34	38.02 ±0.66	31.88 ±0.58	35.96 ±0.42	35.69 ±0.64		

Table 3.14 – Top-1 average accuracy (%). Following [24], accuracy is averaged only for incremental states (i.e. excluding the initial, non-incremental state). Results are averaged over 5 runs for all AL configurations. sIL is the result obtained in a fully supervised IL scenario. $Joint$ is the non-incremental upper-bound performance obtained with all data available. *Best results for each active learning configuration (row) are in bold.*

here. Consequently, improving over random sampling for imbalanced datasets is an interesting result.

The results from Table 3.14 indicate that the balancing-driven acquisition phase is useful for all three imbalanced datasets and active learning budgets tested. The gains for IMAGENET-100 and FACES-100 are usually between 1 and 2 points compared to the classical acquisition processes implemented here ($rand$ - $rand$ or $core$ - $core$). The gains are low for FOOD-100, the third imbalanced dataset tested. This is probably due to the fact that FOOD-100 is a more difficult task, as shown by sIL. More labeled samples per class would probably be needed for efficient training.

$poor$ strategy is better than $b-core$ for IMAGENET-100, while more mixed results are

obtained for FACES-100 and FOOD-100 datasets. Interestingly, the best results are always obtained on top of a $rand$ initial sampling, even when $core$-$core$ baseline is better than a $rand$-$rand$ one, as it is the case for FACES-100 with $\mathcal{B} = 20\%$ and $\mathcal{B} = 5\%$.

When applied without balancing, ent and $marg$ have poorer performance compared to that of $rand$ and $core$. Balancing significantly improves results for both of uncertainty-based methods, but their overall performance still lags behind that of random followed by balancing. This reinforces the finding that a random selection is a competitive acquisition function in our active incremental learning over imbalanced datasets scenario.

The performance drop between active learning configurations and fully supervised IL naturally grows as $\mathcal{B}$ is reduced. The drop between sIL and the best AL configuration is of 3, 6 and 5 points for $\mathcal{B} = 20\%$ for IMAGENET-100, FACES-100, and FOOD-100 respectively. When the AL budget is reduced to only 5% of new data, the corresponding performance losses go to 12.5, 23 and 14 points. Even when as little as 5% of the new data is annotated, suboptimal models are trainable and usable if the IL system needs to be operational quickly.

While the focus is on imbalanced datasets, we also report results with CIFAR-100, a perfectly balanced dataset for completeness. In this case, the balancing-driven sampling has a slightly negative effect when applied over $rand$ and a slightly positive effect over $core$. It is, however, notable that $core$ lags consistently behind $rand$ for CIFAR-100. The best strategy for all $\mathcal{B}$ sizes is $rand$-$rand$, with $rand$ - $poor$ being a close second best configuration.

The gap between active IL and supervised IL is still notable, especially for smaller AL budgets. In practice, active IL is useful when the system needs to be operational quickly after new data are streamed but at the expense of suboptimal performance. If a longer delay is permitted, it is naturally preferable to annotate all new data before updating the incremental model. The gap is even higher between incremental and classical learning, even though FT^{th} has competitive performance compared to existing IL algorithms. Globally, our results provide further confirmation that the use of incremental learning vs. classical learning should be weighted depending on the time, memory and/or computation constraints associated to the AI system.

3.6.6 Conclusion

We proposed a more realistic incremental learning scenario that does not assume that streamed data are readily annotated and that they are evenly distributed among classes. An adaptation of the active learning sampling process is proposed in order to obtain a more balanced labeled subset. This adaptation has a positive effect for imbalanced datasets and a slightly negative effect for the balanced dataset evaluated here. Both proposed acquisition functions improve results

compared to a classical acquisition process. Also interesting, experiments show that the random baseline outperforms the $core-set$ function. The strong performance of random sampling indicates that this method should be consistently used as a baseline for future works in active incremental learning. As a secondary contribution, we introduce FT^{th}, an IL backbone that provides competitive results when compared to state-of-the-art methods.

The proposed approach brings the IL scenario closer to practical needs. It reduces the time needed for an IL system to become operational upon receiving new data. The obtained results are encouraging but further investigation is needed to reduce the gap between active and supervised IL. For instance: (1) use semi-supervised learning methods to automatically expand the labeled dataset and improve overall performance. While appealing, not all semi-supervised methods prove efficient in practice [111] and their usefulness for imbalanced datasets needs to be studied. (2) complement the proposed balancing-driven acquisition functions with a component that pushes the sampling process towards a better coverage of the manifold of each modeled class. This could be done, for instance, by taking inspiration from the herding mechanism [128] already used to select past exemplars. (3) render the IL scenario even more realistic by testing incremental steps of variable size to account for the fact that data might arrive at a variable pace and considering that newly streamed data might belong both to unseen and past classes.

3.7 General Conclusion

Due to the bounded memory of the past, a prediction bias in favor of new classes is observed in the classification layer. This bias leads the network to predict testing images as belonging to new classes, even if it is not necessarily the case. In this chapter, we proposed two methods to tackle catastrophic [102] forgetting while memory of the past is allowed. $Il2M$ (Section 3.3) operates at the end of the CNN, by directly rectifying prediction scores of past classes in order to increase them and make them more comparable to those of new classes. Bias rectification is based on class statistics that we save in a secondary memory and use later for calibration. $ScaIL$ (Section 3.4) takes one step backward and handles bias in the weights matrix of the last fully connected layer. $ScaIL$ performs initial embeddings replay for past classes, in order to take advantage of their weights in their initial states. A further normalization based on new class statistics is done to make past and new class weight magnitudes more comparable. Both $ScaIL$ and $Il2M$ ablate the widely used knowledge distillation [52], and results show that if at least a few exemplars per past class are allowed, distillation loss hurts performance at large scale, and simpler methods that handle IL as an imbalanced learning are favorable.

We conducted an extensive study on a range of popular IL approaches from the state of the art: (1) those based on fine tuning with distillation loss and/or bias removal layers [128, 24, 174, 53, 189], (2) and those based on fixed representations [48]. Experiments show that no approach is always better than the others. Depending on the application domain, each method can be adopted to handle one or many challenges (Section 1.2) related to class IL.

Unfortunately, most existing algorithms make two strong hypotheses that reduce the realism of the incremental scenario: (1) new data are assumed to be readily annotated when streamed and (2) tests are run with balanced datasets while most real-life datasets are imbalanced. In section 3.6, we discard these hypotheses and the resulting challenges are tackled with a combination of active and imbalanced learning. We introduce sample acquisition functions that tackle imbalance and are compatible with IL constraints. We also consider IL as an imbalanced learning problem and propose FT^{th}. This approach increases prediction scores of minority classes to give them more chances to be selected during inference. Results indicate that the proposed contributions positively reduce the gap between active and standard IL performance.

In the next chapter, we take a step further and tackle a more difficult scenario of class-incremental learning, when no memory of the past is allowed.

CHAPTER 4

CLASS-INCREMENTAL LEARNING WITHOUT MEMORY

4.1 General Introduction

Artificial agents are often deployed in applications that need to work under strong computational constraints and receive data sequentially. Incremental Learning (IL) algorithms are deployed to deal with such situations. Examples include (1) exploring robots that receive data in streams and that have a limited access to a memory or no memory, (2) disease classification systems that are not allowed to access past data for privacy issues and, (3) tweets analysis where new data arrives at a fast pace and should be handled in a timely manner.

The most important three characteristics that qualify an IL system to be effective and efficient are the low dependency on memory, the high accuracy on both past and new data, and the time needed to update the model to incorporate new data.

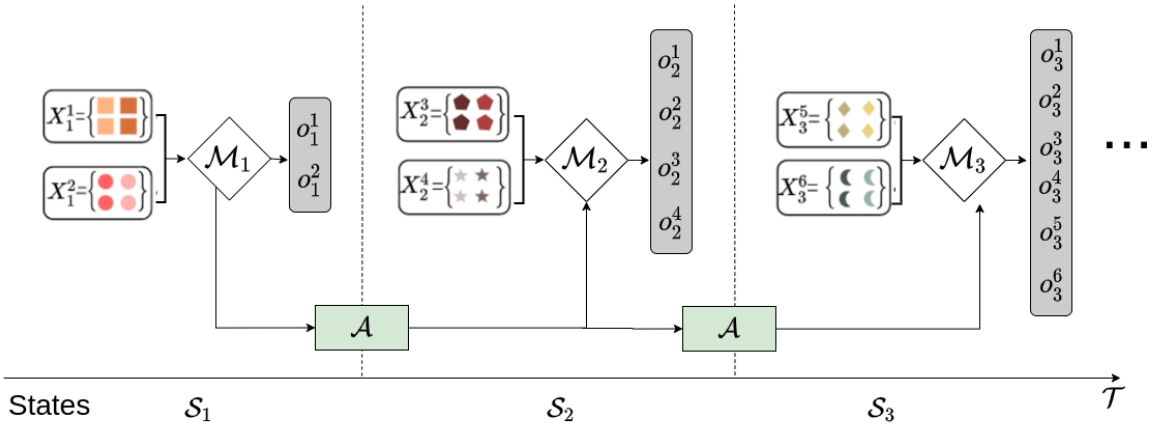


Figure 4.1 – A toy example of class IL without memory where $\mathcal{T} = 3$. $\mathcal{M}_t$ are models, $\mathcal{S}_t$ are states, X_t are sets of images, o_t are raw scores vectors, $\mathcal{A}$ is a memoryless class IL algorithm. *Best viewed in color.*

Memoryless class IL is understudied in literature but important in practice when past data are impossible to store (due to confidentiality issues, for example). In this chapter, we focus on class IL where no memory of the past is allowed. Figure 4.1, we show a toy example with one initial state $\mathcal{S}_1$ and two incremental states ($\mathcal{S}_2$ and $\mathcal{S}_3$). In the initial state, a simple training from scratch is performed to learn the initial set of two classes. At each incremental state, another two new classes arrive with all their available data. These classes should be learned by the model $\mathcal{M}_t$ without forgetting past classes (learned in previous states). An incremental learning algorithm $\mathcal{A}$ is used to update the model $\mathcal{M}_{t-1}$ ($t > 0$) with data of new classes only. Note that contrarily to the Figure 3.1, there is no memory $\mathcal{K}$ to store images from past classes.

The main findings of this chapter are: (1) knowledge distillation is effective when no memory of the past is allowed (this is not the case at large scale when a bounded memory is allowed), (2) the use of past class embeddings from their initial states is beneficial to tackle catastrophic forgetting. Accuracy of current class IL systems without memory is still low as the gap with a full training (with all class data) is large. More efforts are needed to improve current systems in real-life scenarios as the one studied in this chapter.

We propose two approaches designed for class IL without memory. The first method is called SIW (*Standardization of Initial Weights*). It performs initial weights replay for past classes, followed by standardizing all class weights and a state-level calibration. The second method is based on transferring calibration parameters between datasets. The main contributions of this method are first to enable the use of *Bias Correction* (BiC) [174] in a scenario without memory by transferring calibration parameters from reference datasets trained offline to target datasets. Second, we propose a fine-grained version of BiC that rectifies past classes scores based on states in which they were encountered with all their data for the first time. We call this method *Adaptive Bias Correction* ($AdBiC$).

Similarly to Chapter 3, we conduct a comprehensive study and highlight the merits and limitations of a range of class-incremental approaches that do not use a memory of the past. Experiments show that distillation is beneficial, while fixed representations are the best choice if the initial model is trained on a sufficiently rich dataset.

4.2 SIW: Standardization of Initial Weights for Class-Incremental Learning

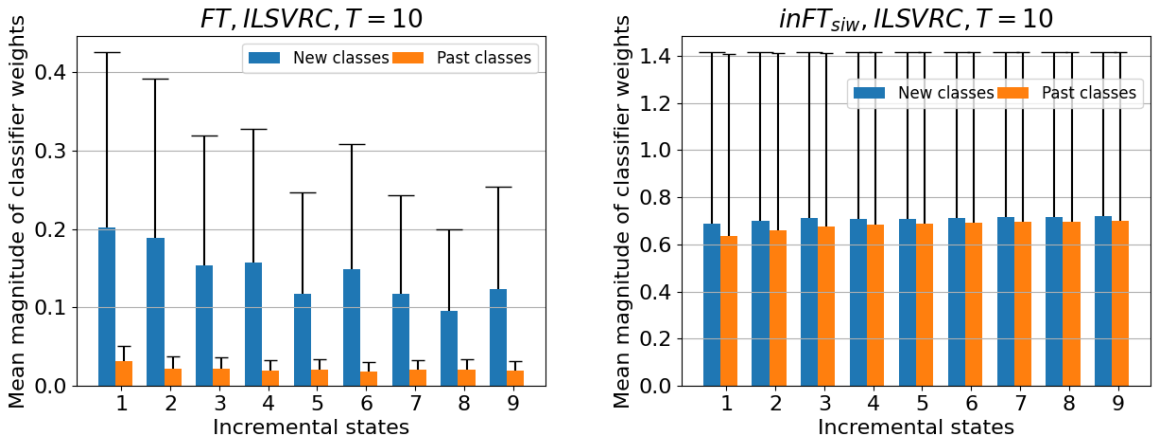
4.2.1 Introduction

Based on experiments conduction in Chapter 3, our main hypothesis is that catastrophic forgetting mainly affects the classification layer of deep models during vanilla fine tuning. With this in mind, we tackle memoryless class IL and provide an overview of the introduced method in Figure 4.4. Inspired by $ScaIL$ (Section 3.4), we propose to exploit initial classifier weights learned with all data of past classes. Initial weights provide a good representation of past classes but need normalization for use in later IL states because their magnitude varies significantly across IL states. Preliminary analysis indicates that the magnitude of classifiers tends to decrease for classes that are learned in later states. We exploit classifier standardization as a way to normalize initial classifiers. Normalization helps, but the prediction scores also change due to the variable performance of IL models. Consequently, we use the same state-level calibration of class predictions as in $IL2M$ (Section 3.3). We evaluate results with four public datasets and three values for the number of incremental states. Results show that our approach performs consequently better than competitive baselines for large-scale datasets.

We build on $ScaIL$ and [193] to: exploit information from the initial state of classifiers; ensure fairness for the predictions associated to past and new classes [53, 174]; use vanilla FT as a backbone to train deep models [5] . However, our approach differs through the method used for the normalization of initial classifier weights and the focus is on a large-scale memoryless IL. Note that many existing methods cannot operate in the absence of memory and become unusable in our setting. The following ones are usable in memoryless IL and will be used in the evaluation: LwF [86], the end-to-end version of $LUCIR$ [53], baseline methods that exploit initial classifiers without bounded memory from Section 3.4.

We hypothesize that it is possible to exploit initial classifier weights, learned when data is first streamed for each class to mitigate catastrophic forgetting in memoryless IL. CNN prediction scores are obtained from the combination of features provided by the penultimate layer of the model with classifier weights from the final layer. We analyze these two layers in an IL context to motivate our approach. The ILSVRC dataset with an initial and nine incremental states is used for both analyses.

In Figure 4.2(a), we present a summarized view of the magnitudes of classifiers for an IL baseline that exploits vanilla fine tuning. The absolute values of individual classifier dimen-



(a) (b)

Figure 4.2 – Mean magnitudes of classifier weights for new and past classes in memoryless incremental learning: (a) left - with vanilla fine tuning affected by catastrophic forgetting and (b) right - after standardization of initial classifiers. Mean magnitudes are computed only for incremental states and the first non-incremental state is excluded. Note that the reference state is the rightmost one in each figure.

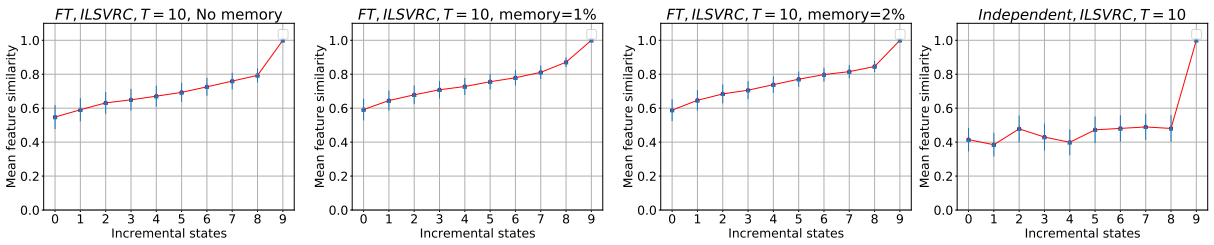
sions are aggregated to compute mean magnitudes for new and past classes, respectively. Past classes have much lower magnitudes because there are no exemplars to be replayed for them in memoryless IL. Since magnitudes are much higher for new classes, test examples will always be attributed to one of these classes, even if they belong to the past classes. This observation provides further support for the previous conclusion that vanilla FT is not directly usable in IL [24, 128].

The magnitudes of new classes in Figure 4.2(a) vary across incremental states, with a global tendency toward reduction in later states. A normalization of initial classifiers is thus needed to ensure fairness if they are replayed across incremental states as proposed here. New classifiers from previous states of Figure 4.2(a) are aggregated to represent past classes in each current state of Figure 4.2(b). Normalization makes statistical populations more comparable and it is obtained by applying standardization [38], a method which is discussed in detail in Subsection 4.2.2. The standardized classifiers, illustrated in Figure 4.2(b), have comparable magnitudes and become usable in memoryless IL.

A second important assumption of our approach is that features extracted from the penultimate layer of the current IL model are compatible with initial classifiers from previous states. This assumption holds if the current features keep a trace of what was learned before. We de-

sign a simple experiment that assesses the degree of similarity between features of the same images extracted in different incremental states. Features are extracted for test images of the initial non-incremental state using models learned in each incremental state. Features of test images extracted in the 9^{th} and last incremental states are used as a reference to illustrate the use of initial classifiers from previous states with its features. Cosine similarity between them and the features of the same images from each previous state is computed. The mean feature similarities between the last state and previous ones are presented in Figure 4.3.

To better situate similarities for memoryless IL (Figure 4.3(a)), we also present statistics for IL with bounded memory, including 1% and 2% of the dataset (Figure 4.3(b) and (c) respectively). We also provide similarities for independent training of incremental states where no fine tuning is used in Figure 4.3(d). Naturally, this last setting is not a valid IL approach and is shown only to illustrate a lower bound of feature similarity.



(a) (b) (c) (d)

Figure 4.3 – Mean feature similarities between incremental states for test images of the first state. Cosine similarities are computed for vanilla fine tuning as follows: (a) without memory, (b) with bounded memory of 1% of the dataset, (c) with bounded memory of 2%. Subfigure (d) plots lower-bound similarities for the case when individual states are learned independently, without fine tuning. The upper bound for similarity is 1 and would be obtained for the model, which is frozen after the initial state. However, such an approach has no plasticity and cannot incorporate knowledge related to data streamed during IL. The final incremental state ($\mathcal{S}_{10}$) is used as a reference to compute similarities with other states. The more distant two states are, the lower the similarity is likely to be.

The results from Figure 4.3 indicate that mean similarities obtained with fine tuning without and with memory are significantly higher than those obtained with independent training. This finding validates the fact that current features learned with vanilla FT keep a trace of what was learned in previous states. Mean similarities decrease with the distance between the current state and the initial one because forgetting is higher when more trainings are involved. The use of a bounded memory in Figure 4.3(b) and (c) provides better similarities compared to memoryless

IL in Figure 4.3(a). The effect is particularly visible for states such as $\mathcal{S}_9$ or $\mathcal{S}_8$ which are close to the reference one and becomes less important for more distant states such as $\mathcal{S}_3$ or $\mathcal{S}_2$. Note that features from the current IL state were used successfully with initial classifiers in $ScaIL$ (Section 3.4) if a bounded memory of the past was allowed. The comparison of feature similarities indicates that their use with adapted initial classifiers is more challenging without memory than with a bounded memory but still doable.

4.2.2 Proposed approach

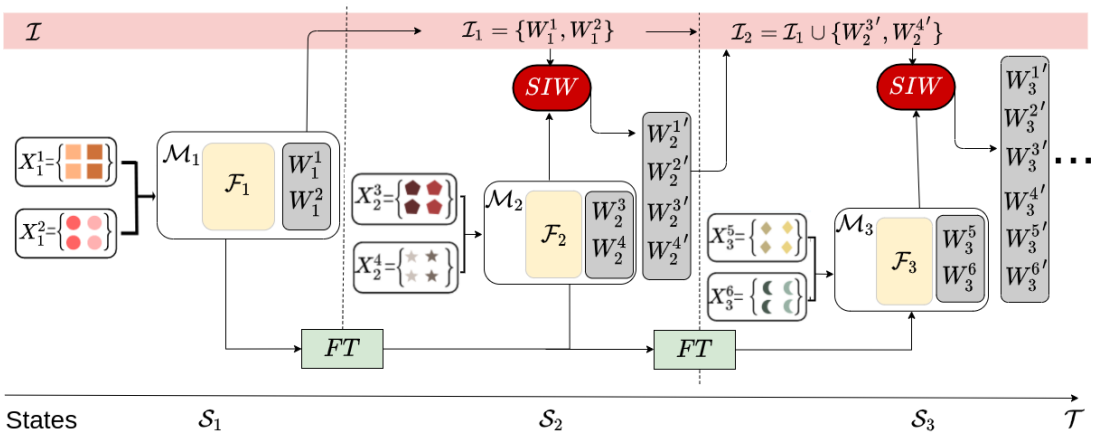


Figure 4.4 – Overview of the proposed method with an initial and two incremental states, and two classes per state. A deep model $\mathcal{M}_t$ is trained for each state $\mathcal{S}_t$. Each model includes a feature extractor $\mathcal{F}_t$ and a classification layer $\mathcal{W}_t$. In memoryless IL, models have access only to data for new classes and to the previous model for fine tuning (FT). Note that FT can be implemented in different ways: with a classical distillation component to counter catastrophic forgetting [24, 61, 128], with a more sophisticated tackling of forgetting [53] or using vanilla FT (Sections3.3 and 3.4). Existing end-to-end methods [24, 53] (in blue) perform recognition using the weights from $\mathcal{W}_t$, the classification layer of the current model which updates past classifiers. Our approach (in green) is different because it freezes classifiers learned initially ($init$) for each class and applies standardization (siw) to make them more comparable.

We propose to replay the initial classifiers of each class in order to mitigate catastrophic forgetting. Following the notations from Equation 2.3 given in Section 2.2, we write the classification layer made of initial classifier weights as:

$$\boldsymbol{\mathcal{W}_t^{init}} = \{\boldsymbol{W_1^1}, .., \boldsymbol{W_1^{N_1}}, \boldsymbol{W_2^{N_1+1}}, .., \boldsymbol{W_2^{N_2}}, ..., \boldsymbol{W_{t-1}^{N_{t-2}+1}}, .., \boldsymbol{W_{t-1}^{N_{t-1}}}, \boldsymbol{W_t^{N_{t-1}+1}}, .., \boldsymbol{W_t^{N_t}}\} \tag{4.1}$$

The analysis from Subsection 4.2.1 shows that the weights from Equation 4.1 need normalization to become comparable across states. We apply a standardization of initial weights (siw) to obtain a normalized version of the weights matrix:

$$\boldsymbol{\mathcal{W}}'_t = \{\boldsymbol{W_1^{1'}}, .., \boldsymbol{W_1^{N_1'}}, \boldsymbol{W_2^{N_1+1'}}, .., \boldsymbol{W_2^{N_2'}}, ..., \boldsymbol{W_{t-1}^{N_{t-2}+1'}}, .., \boldsymbol{W_{t-1}^{N_{t-1}'}}, \boldsymbol{W_t^{N_{t-1}+1'}}, .., \boldsymbol{W_t^{N_t'}}\} \tag{4.2}$$

Each dimension w'_d of a standardized classifier $\boldsymbol{W'}$ from Equation 4.2 is calculated using:

$$w'_d = \frac{w_d - \mu(\boldsymbol{W})}{\sigma(\boldsymbol{W})} \tag{4.3}$$

with: w'_d - the d^{th} dimension of $\boldsymbol{W'}$, w_d - the d^{th} dimension of an initial classifier $\boldsymbol{W}$ from Equation 4.1, $\mu(\boldsymbol{W})$ and $\sigma(\boldsymbol{W})$ are the mean and standard deviation of $\boldsymbol{W}$.

Standardization is useful if it is applied to the statistical populations which follow a normal distribution [38], which is the case for classifier weights from Equation 4.1. Figure 4.5 provides weights distribution of a random subset of classifier weights from the weights matrix $\boldsymbol{\mathcal{W}_t}$ defined in Equation 2.3. These examples illustrate the fact that the classifier weights follow a normal distribution. The use of standardization to normalize them is thus appropriate.

Assuming that the incremental process is in the t^{th} state, the final prediction score of a test image x for the j^{th} class learned initially in the i^{th} state (with $i \leq t$), is given by:

$$o_t^j(x) = (\boldsymbol{f_t(x)} \cdot \boldsymbol{W_t^{j'}} + b_j^i) \times \frac{\mu(\mathcal{M}_t)}{\mu(\mathcal{M}_i)} \tag{4.4}$$

with: $\boldsymbol{f_t(x)}$ - features of image x given by the extractor of the current model $\mathcal{M}_t$; $\boldsymbol{W_t^{j'}}$ - standardized classifier weights of the j^{th} class initially learned in the i^{th} state as given by Equation 4.2; b_j^i - the class bias value; $\mu(\mathcal{M}_t)$ and $\mu(\mathcal{M}_i)$ - means of top-1 predictions of models learned in the t^{th} and i^{th} states computed over their training sets.

The first term of Equation 4.4 is a version of the usual CNN prediction process in which the basic weights $\boldsymbol{W_t^j}$ from Equation 2.3 are replaced by the standardized initial weights $\boldsymbol{W_t^{j'}}$ from Equation 4.2. This term is referred to as siw in Section 4.2.4. The second term is the same that we use in $IL2M$ (Section 3.3), where we observed that a model level calibration is useful when combining information from different models. $\mu(\mathcal{M}_t)$ and $\mu(\mathcal{M}_t)$ are calculated by passing all training images available in each of the two states through the respective model. This term is referred to as mc in Section 4.2.4.

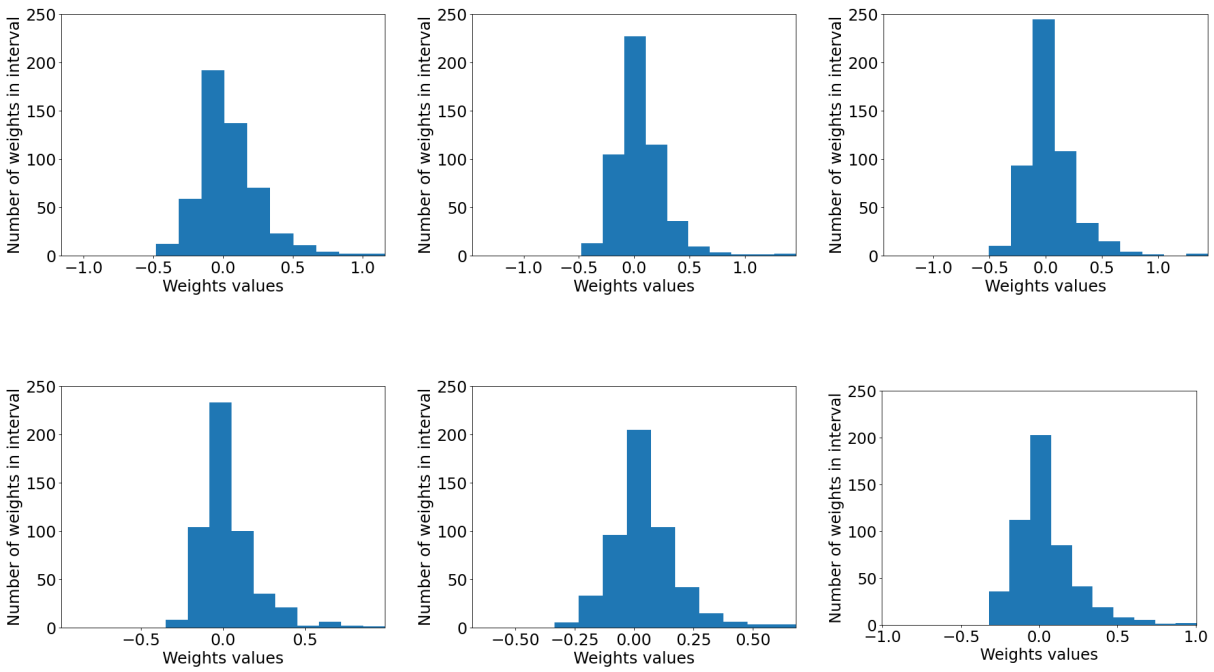


Figure 4.5 – Weights distribution of a random subset of classifier weights from the weights matrix $\mathcal{W}_t$ defined in Equation 2.3 (Section 2.2).

4.2.3 Experiments

— **Datasets** (Appendix B) - ILSVRC [135], LANDMARKS [108], VGGFACE2 [22], and CIFAR-100 [74].

— **Incremental states** - We use $\mathcal{T} = \{10, 20, 50\}$.

— **Evaluation measures -** Top-5 accuracy [135] and G_{IL} measure (Subsection 3.4.4).

— **Baselines -** LwF [86], $LUCIR$ [53], FT^{init}, and FT^{init}_{L2} (the last two baselines are the one we proposed in Subsection 3.4.3). We also try two supplementary normalization techniques defined below.

➔ *min-max normalization* - each dimension of the classifier is calculated using:

$$w'_d = \frac{w_d - min(\boldsymbol{W})}{max(\boldsymbol{W}) - min(\boldsymbol{W})} \tag{4.5}$$

➔ *mean normalization* - each dimension of the classifier is calculated using

$$w'_d = \frac{w_d - \mu(\boldsymbol{W})}{max(\boldsymbol{W}) - min(\boldsymbol{W})} \tag{4.6}$$

Note that our methods are based on FT^{init}: FT^{init}_{siw} exploits only the siw term from Equation 4.4, while FT^{init}_{siw+mc} exploits both the siw and the mc terms from Equation 4.4.

4.2.4 Results and discussion

Dataset	ILSVRC			VGGFACE2			LANDMARKS			CIFAR-100			G_{IL}
States	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	
LwF	45.3	37.6	27.1	53.3	42.6	30.8	58.8	49.2	35.2	79.5	65.3	39.0	-34.72
LwF^{init}	47.1	39.9	32.2	58.1	50.8	40.5	55.7	50.2	39.8	79.4	67.9	42.8	-31.97
LwF^{init}_{siw}	54.0	45.8	35.1	70.4	59.3	45.2	61.0	53.8	42.2	**80.0**	**68.8**	**44.6**	-28.06
LwF^{init}_{siw+mc}	40.2	44.7	33.8	67.5	56.5	42.0	54.6	48.0	37.2	78.6	67.5	43.8	-30.79
$LUCIR$	57.6	39.4	21.9	**91.4**	68.2	32.2	**87.8**	63.7	32.3	57.5	35.3	21.0	-24.75
$LUCIR^{mc}$	53.7	30.5	12.7	82.6	51.0	21.0	84.1	44.0	21.6	45.8	26.8	23.7	-32.18
FT	20.6	13.4	7.1	21.3	13.6	7.1	21.3	13.6	7.1	21.3	13.7	17.4	-54.91
FT^{init}	61.0	44.9	23.8	90.9	64.4	33.1	68.8	49.4	22.2	55.1	40.8	19.9	-28.99
FT^{init}_{L2}	51.6	43.3	34.5	76.8	66.8	55.1	61.4	52.5	39.2	47.5	39.3	22.5	-26.80
FT^{init}_{mc}	62.0	39.6	19.2	78.5	52.3	27.5	73.3	44.2	17.7	57.9	34.2	18.2	-32.75
FT^{init}_{L2+mc}	53.6	42.7	35.6	86.9	71.4	53.6	66.2	52.6	37.9	52.6	43.1	18.2	-25.02
FT^{init}_{siw}	61.6	51.9	39.9	84.0	80.6	61.9	75.1	62.6	**43.2**	56.0	41.8	22.5	-20.97
FT^{init}_{siw+mc}	**64.4**	**54.3**	**41.4**	88.6	**84.1**	**62.6**	79.5	**64.5**	**43.2**	59.7	44.3	18.4	**-19.38**
$Joint$	92.3			99.2			99.1			91.2			-

Table 4.1 – Top-5 average IL accuracy (%) for the different methods with $\mathcal{T} = \{10, 20, 50\}$ states. $Joint$ is a classical learning, with all data available. *Best results are in bold.*

The results from Table 4.1 show that FT^{init}_{siw+mc} provides a G_{IL} improvement of over 5 points compared to $LUCIR$ [53], the best baseline. The gain is even higher compared to FT^{init}_{L2+mc}, the second-best baseline which extends FT^{init}_{L2}. This difference in favor of FT^{init}_{siw+mc} underlines the fact that classifier weights standardization is more appropriate than $L2$ normalization for memoryless IL. A favorable comparison to two other normalization methods is provided in Table 4.2. Results show that the siw term from Equation 4.4 is useful both for FT^{init} and LwF

while mc is beneficial for FT^{init} but degrades results for LwF and $LUCIR$. Standardization of LwF^{init} weights has a positive effect for all datasets compared to LwF, especially for $\mathcal{T} = \{20, 50\}$ states. Moreover, even when mc works, its contribution is less important than that of siw.

The worst results by far are obtained with FT, both globally and for individual configurations. FT trains well for new classes but provides nearly random results for past classes that have no exemplars available for replay in memoryless IL. This confirms previous findings [128, 24] regarding the strong effect of catastrophic forgetting on vanilla fine tuning in memoryless IL.

The comparison of performance for different datasets indicates that distillation is useful at small scale but has lower utility or becomes even detrimental for large datasets. The usefulness of distillation for small datasets but not for large ones is an open problem. In [61], authors note that distillation induces confusions among past classes [52]. To better understand this phenomenon, we provide an analysis of the typology of errors in Table 4.3. Note that while FT runs do not perform well, normalization does help for them.

The LwF^{init}_{siw} version of LwF is the best method for CIFAR-100, with a large margin compared to all methods which are not based on LwF. The use of initial weights in LwF^{init} brings a G_{IL} improvement of nearly 3 points, and the addition of standardization in LwF^{init}_{siw} brings another $4\ G_{IL}$ points gain. $LUCIR$ has lower performance but rather comparable to that of FT^{init} based methods for CIFAR-100. The results for the three large datasets show that LwF has the second-lowest performance, after vanilla FT. The improvements over classical LwF brought by standardization are much more important for the three large datasets. $LUCIR$'s more sophisticated scheme for countering catastrophic forgetting is clearly useful compared to classical distillation from LwF. The removal of the distillation component and the use of initial classifier weights in FT^{init} gives globally better results than $LUCIR$ and LwF for the three large datasets. This behavior in a large-scale setting is mainly explained by the observation made in [53] regarding the high number of confusions among past classes when distillation is used. The use of siw and mc to FT^{init} is also beneficial, especially when $\mathcal{T}$ is big for large datasets.

The number of incremental states has an important effect on IL performance [24, 128]. The larger the number of states is, the more challenging the process will be. This is confirmed by the results with $\mathcal{T} = \{10, 20, 50\}$ states in Table 4.1 for all tested methods. Actually, as more incremental states are added, IL models are more prone to forgetting due to the increasing number of intermediate model updates, which causes information loss. Our approach does significantly better than existing methods for the three large datasets with $\mathcal{T} = 50$. Its performance reaches

41.4 for ILSVRC compared to only 27.1 and 21.9 for LwF and $LUCIR$.

Table 4.1 allows an ablation analysis of our approach. Such an analysis is important to understand the contribution of individual components to the final results. Vanilla FT is the backbone upon which we build. The use of initial raw weights for past classes in FT^{init} has a strong positive effect as it reduces the performance gap measured by G_{IL} by nearly a half (-28.99 vs. -54.91 for vanilla FT). The sole use of calibration in FT^{init}_{mc} has a negative effect since performance drops to -32.75. The introduction of standardization in FT^{init}_{siw} has a important positive effect since it brings an 8 points G_{IL} improvement over FT^{init}. Finally, the use of both terms from Equation 4.4 in FT^{init}_{siw+mc} has a light positive effect with a 1.6 points improvement over FT^{init}_{siw}. We note that FT^{init}_{siw} improves over FT^{init} for all individual configurations. The largest performance gains between the two methods are obtained for the three large datasets with the $\mathcal{T} = 50$, the highest number of incremental states tested. This is the most challenging setting since the effects of catastrophic forgetting are stronger for a larger number of IL states. The addition of state mean calibration has a positive, albeit smaller, effect in all individual configurations but two. It does not improve results for LANDMARKS with $\mathcal{T} = 50$ states and degrades them for CIFAR-100 with the same number of states. This is probably because there are only two classes per state in the latter configuration and the obtained statistics are not stable enough.

Results with other calibration methods

Table 4.2 provides results with mean and min-max normalization of weights in addition to $L2$ and siw. Standardization provides the best performance for all tested configurations. Mean calibration is second best and has a better performance compared to the L2-normalization already used in 3.4. Calibration with min-max is not effective and did not provide any good results.

Dataset	ILSVRC			VGGFACE2			LANDMARKS			CIFAR-100			G_{IL}
States	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	
$FT^{init}_{min-max}$	3.3	10.0	7.1	4.7	20.1	18.5	17.2	12.2	6.3	19.9	18.3	20.7	-55.52
FT^{init}_{mean}	54.1	49.4	38.0	69.7	78.4	58.6	72.8	61.1	41.3	52.9	38.1	21.0	-23.76
FT^{init}_{L2}	51.6	43.3	34.5	76.8	66.8	55.1	61.4	52.5	39.2	47.5	39.3	22.5	-26.80
FT^{init}_{siw}	**61.6**	**51.9**	**39.9**	**84.0**	**80.6**	**61.9**	**75.1**	**62.6**	**43.2**	**56.0**	**41.8**	**22.5**	**-20.97**
Joint		92.3			99.2			99.1			91.2		-

Table 4.2 – Top-5 average IL accuracy (%) for the min-max and mean normalization, with $\mathcal{T} = \{10, 20, 50\}$ incremental states. *Best results are in bold.*

Error analysis

Similarly to $Scail$ (Section 3.4), we provide in Table 4.3 top-1 correct and wrong classifications for: (1) FT - the simplest method tested, (2) $LUCIR$ - the best existing method (3) FT^{init}_{siw+mc} - the proposed method. The analysis is done for the large dataset ILSVRC, with $\mathcal{T} = 20$ states. $c(p)$ and $c(n)$ are the correct classification for past/new classes. $e(p,p)$ and $e(p,n)$ are erroneous classifications for test samples of past classes mistaken for other past classes and new classes respectively. $e(n,p)$ and $e(n,n)$ are erroneous classifications for test samples of new classes mistaken for past classes and other new classes respectively. Note that the percentages on the first three and last three lines of each table sum up to 100%. Since the number of test images varies across IL states, percentages are calculated separately for test images of past and new classes in each $\mathcal{S}_t$ to get a quick view of the relative importance of each type of errors. $c(p)$, $e(p,p)$, and $e(p,n)$ sum to 100% on each column, as do $c(n)$, $e(n,n)$, and $e(n,p)$.

The analysis shows that vanilla FT suffers from a total forgetting of the past classes since all their test images are wrongly classified. The effect of catastrophic forgetting is obvious in the way that 100% of past classes are mistakenly classified as belonging to new classes. Equally important, standardization of the initial weights not only reduces forgetting, but also considerably reduces the confusions among new classes.

The comparison of $LUCIR$ and FT^{init}_{siw+mc} shows that the first method is better at classifying test samples of new classes but has worse behavior for test samples of past classes. $LUCIR$ $c(p)$ scores are better for the first three iterations but fall behind those of FT^{init}_{siw+mc} afterward. Note that both methods are strongly affected by catastrophic forgetting toward the end of the incremental process, with top-1 accuracy at 6% and 11.8% for $LUCIR$ and FT^{init}_{siw+mc} respectively. This finding indicates that, while both distillation in $LUCIR$ and classifier weights replay FT^{init}_{siw+mc} have a slight positive effect, memoryless IL remains a very challenging task. It is also interesting that the distribution of errors is different. $LUCIR$ fails to ensure fairness between past and new classes since $e(p,n)$ are much more frequent than $e(p,p)$. FT^{init}_{siw+mc} is less biased toward new classes but produces a large number of confusions between past classes ($e(p,p)$).

Incremental states		$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	$\mathcal{S}_{11}$	$\mathcal{S}_{12}$	$\mathcal{S}_{13}$	$\mathcal{S}_{14}$	$\mathcal{S}_{15}$	$\mathcal{S}_{16}$	$\mathcal{S}_{17}$	$\mathcal{S}_{18}$	$\mathcal{S}_{19}$
FT	$c(p)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$e(p,p)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$e(p,n)$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	$c(n)$	87.8	87.28	90.48	91.4	90.44	87.92	89.64	88.12	87.24	89.68	89.72	90.16	90.6	89.8	87.84	92.4	89.56	89.28	87.52
	$e(n,n)$	12.2	12.72	9.52	8.6	9.56	12.08	10.36	11.88	12.76	10.32	10.28	9.84	9.4	10.2	12.16	7.6	10.44	10.72	12.48
	$e(n,p)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FT^{init}_{siw+mc}	$c(p)$	38.4	27.0	33.2	31.3	29.0	22.0	20.1	15.0	17.9	14.7	17.7	16.5	15.3	13.1	13.2	14.0	14.1	12.5	11.8
	$e(p,p)$	22.7	15.0	41.4	41.9	60.7	48.5	51.8	31.9	60.2	40.7	68.1	62.6	66.8	48.2	47.2	66.9	64.9	52.7	50.0
	$e(p,n)$	38.9	58.0	25.4	26.8	10.3	29.5	28.1	53.0	21.9	44.6	14.1	20.9	17.9	38.7	39.6	19.1	21.0	34.8	38.2
	$c(n)$	75.8	82.7	75.7	75.8	67.2	75.1	77.4	83.8	69.8	83.2	68.6	76.1	70.5	82.0	78.4	76.2	72.6	80.4	80.3
	$e(n,n)$	8.5	11.5	4.2	3.1	1.8	6.8	4.6	9.8	4.5	7.9	3.2	3.5	3.1	6.5	8.3	2.7	3.8	5.4	7.7
	$e(n,p)$	15.7	5.8	20.0	21.1	31.0	18.0	18.0	6.4	25.7	8.9	28.2	20.4	26.4	11.5	13.3	21.1	23.6	14.1	12.0
$LUCIR$	$c(p)$	66.1	46.9	33.5	26.7	23.2	19.0	15.1	13.3	11.8	9.9	9.1	8.3	7.9	8.0	7.5	6.7	6.5	6.0	6.0
	$e(p,p)$	4.2	10.1	14.7	20.4	26.6	25.8	24.1	27.5	27.9	28.1	28.3	28.8	29.8	31.5	29.3	30.8	29.6	29.6	30.6
	$e(p,n)$	29.8	42.9	51.8	52.9	50.2	55.3	60.8	59.2	60.3	62.0	62.6	63.0	62.3	60.5	63.2	62.5	63.9	64.4	63.4
	$c(n)$	78.3	79.7	82.2	82.2	82.4	78.2	82.6	81.5	79.0	84.5	82.7	83.4	84.1	82.9	81.2	86.2	82.8	83.3	81.2
	$e(n,n)$	16.0	15.5	13.5	11.4	12.2	15.2	12.3	13.0	14.5	11.4	11.9	11.9	11.2	11.5	14.2	9.0	11.7	12.1	13.8
	$e(n,p)$	5.6	4.8	4.4	6.4	5.3	6.6	5.2	5.5	6.5	4.1	5.4	4.8	4.6	5.6	4.6	4.8	5.5	4.6	5.0

Table 4.3 – Top-1 correct and wrong classification for FT, FT^{init}_{siw+mc} and $LUCIR$ for ILSVRC with $\mathcal{T} = 20$.

4.2.5 Conclusion

We proposed the reuse of normalized initial classifier weights to mitigate the effect of catastrophic forgetting in memoryless IL. A preliminary analysis showed that initial classifiers could be reused in later incremental states, but a normalization step is needed to make them comparable. We introduced a normalization component based on standardization which ensure fairness between classes learned in different IL states which are used together. Our method compares favorably to standard handling of catastrophic forgetting in [53, 86, 128]. Interestingly, our results indicate that distillation is only useful for small datasets and has a negative effect on larger datasets. In this latter case, the use of simpler vanilla fine tuning backbone is more appropriate. Note that the proposed method also improves the results obtained with classical distillation [86, 128]. However, the gap between classical learning remains important, and further efforts are needed towards reducing it.

In the next section, we present a novel method that tackles catastrophic forgetting in a memoryless setting. To the best of our knowledge, we are the first to provide a method that enables the use of the bias correction layer from [174] without having access to past data.

4.3 TransIL: Dataset Knowledge Transfer for Class-Incremental Learning

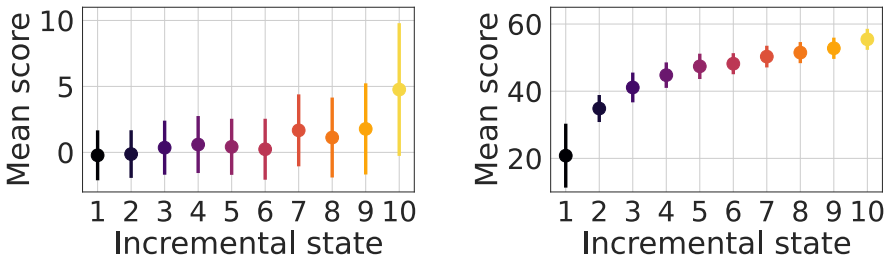
4.3.1 Introduction

Class-incremental learning without memory is challenging and generic since no storage of past samples is allowed. In the absence of memory, existing methods become variants of Learning without Forgetting [86] with different formulations of the distillation term. Importantly, bias correction methods become inapplicable without access to past classes samples.

Our main contribution is to enable the use of the bias correction methods, such as the BiC layer from [174], in class IL without memory. We focus on this approach because it is both simple and effective in IL with memory [15, 101]. Authors of BiC [174] use a validation set that stores samples of past classes to optimize parameters. Instead, we learn correction parameters offline on a set of reference datasets and then transfer them to target datasets. The method is thus abbreviated *TransIL*. The intuition is that, while datasets are different, optimal bias correction parameters are stable enough to be transferable between them. We illustrate the approach in Figure 4.7, with the upper showing the IL process with a reference dataset. A memory for the validation samples needed to optimize the bias correction layer is allowed since the training is done offline. The lower part of the figure presents the incremental training of a target dataset. The main difference with the standard memoryless IL training comes from the use of a bias correction layer optimized on the reference dataset. Its introduction leads to an improved comparability of prediction scores for past and new classes. Note that the proposed method is applicable to any class IL method since it only requires the availability of raw predictions provided by deep models $\mathcal{M}$.

The second contribution is to refine the definition of the bias correction layer introduced in [174]. The original formulation considers all past classes equally in the correction process. With [101], we hypothesize that the degree of forgetting associated with past classes depends on the initial state in which they were learned. Consequently, we propose *Adaptive BiC* ($AdBiC$), an optimization procedure that learns a pair of parameters per IL state instead of a single pair of parameters as proposed in [174]. We provide a comprehensive evaluation of the proposed method by applying it on top of four backbone class IL methods. Five target visual datasets with different numbers of IL states are used. An improvement of top-1 accuracy is obtained for almost all tested configurations. Importantly, the additional memory needs are negligible since only a compact set of correction parameters is stored.

4.3.2 Proposed approach



(a) LwF [86] (b) LUCIR [53]

Figure 4.6 – Mean prediction scores and associated standard deviations for CIFAR-100 classes grouped by state at the end of the IL process, with $\mathcal{T} = 10$ states, for LwF and $LUCIR$ methods. Results indicate that the degree of forgetting depends on the initial state in which classes were first learned. Bias correction with parameters optimized per state is thus needed to make class IL fairer.

The unavailability of past class exemplars when updating the incremental models leads to a classification bias toward new classes [174, 189]. We illustrate this in Figure 4.6 by plotting mean prediction scores per state for the CIFAR-100 dataset with $\mathcal{T} = 10$ states using $LUCIR$ and LwF, the two distillation based approaches tested here. Figure 4.6 confirms that recently learned classes are favored, despite the use of knowledge distillation to counter the effects of catastrophic forgetting. New classes, learned in the last state, are particularly favored. The predictions profiles for $LUCIR$ and LwF are different. $LUCIR$ mean predictions per state increase from earlier to latest states, while the tendency is less clear for LwF. Predictions of the latter also have a stronger deviation in each state. These observations make $LUCIR$ a better candidate for bias correction compared to LwF.

Adaptive Bias Correction layer

Among the methods proposed to correct bias, the linear layer introduced in [174] is interesting for its simplicity and effectiveness. This layer is defined in the t^{th} state as:

$$BiC(\boldsymbol{o}_t^k) = \begin{cases} \boldsymbol{o}_t^k & \text{if} \quad k \in [1,\ t-1] \\ \alpha_t \boldsymbol{o}_t^k + \beta_t \cdot \mathbf{1} & \text{if} \quad k = t \end{cases} \tag{4.7}$$

where $\boldsymbol{o}_t^k$ are the raw scores of classes first seen in the k^{th} state, obtained with $\mathcal{M}_t$; $(\alpha_t,\ \beta_t)$ are the bias correction parameters in the t^{th} state, and $\mathbf{1}$ is a vector of ones.

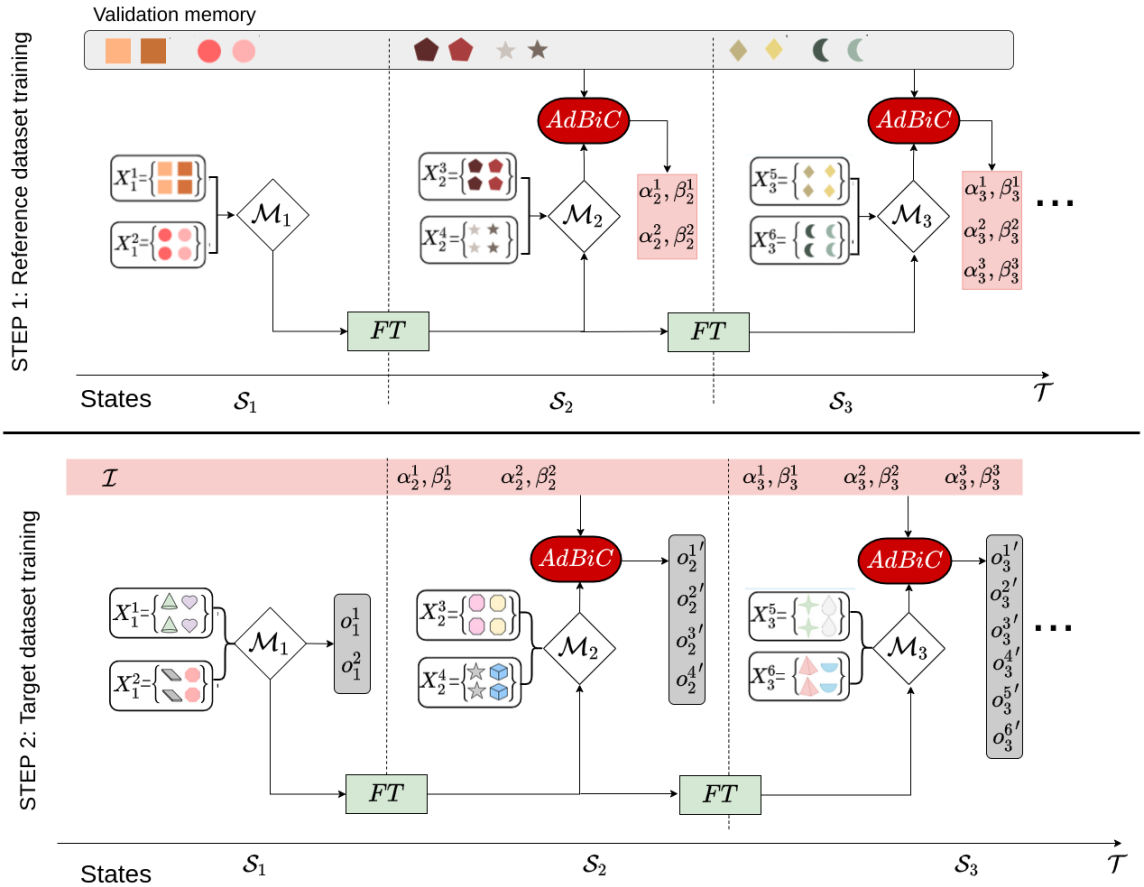


Figure 4.7 – Illustration of *TransIL*, our proposed method, depicting states from 1 to 3 for a reference (*top*) and a target (*bottom*) dataset. The model $\mathcal{M}$ is updated in each state with data from new classes. The class IL process is first launched offline on the reference dataset where $AdBiC$, our proposed bias correction layer, is trained using a validation memory (*in light grey*) which stores samples for past and new classes. Class IL is then applied to the target dataset but without class samples shared across states since a memory is not allowed in this scenario. The set of optimal parameters of $AdBiC$ obtained for the reference dataset (*light pink memory* $\mathcal{I}$) is transferred to the target dataset. This is the only information shared between the two processes and it has a negligible memory footprint. The transfer of parameters enables the use of bias correction for the target dataset. The final predictions obtained in the state $\mathcal{S}_3$ are improved compared to the direct use of $\mathcal{M}_3$ predictions since the bias in favor of new classes is reduced. *Best viewed in color.*

Equation 4.7 rectifies the raw predictions of new classes learned in the t^{th} state to make them more comparable to those of past classes. The deep model is first updated using $\mathcal{X}_t$ containing new classes for this state. The model is then frozen and calibration parameters (α_t and β_t) are optimized using a validation set that includes samples of new and past classes. We remind that

Equation 4.7 is not applicable in class IL without memory, the scenario explored here, because no samples of past classes are allowed.

Figure 4.6 shows that mean scores of classes learned in different incremental states are variable, which confirms that the amount of forgetting is uneven across past states. It is thus important to correct biases differently for classes that were learned in different IL states. To address this challenge, we define an adaptive version of BiC which rectifies predictions in the t^{th} state with:

$$adBiC(\boldsymbol{o_t^k}) = \alpha_t^k \boldsymbol{o_t^k} + \beta_t^k \cdot \mathbf{1} \; ; \quad k \in [1, t] \tag{4.8}$$

where α_t^k, β_t^k are the parameters applied in state $\mathcal{S}_t$ to classes first learned in state $\mathcal{S}_k$.

Differently from Equation 4.7, Equation 4.8 adjusts prediction scores depending on the state in which classes were first encountered in the IL process. Note that each α_t^k, β_t^k pair is shared between all classes first learned in the same state. These parameters are optimized on a validation set using the cross-entropy loss, defined for one data point $(\mathbf{x}, y)$ as:

$$\mathcal{L}(\boldsymbol{q_t}, y) = -\sum_{k=1}^{t} \sum_{j=1}^{|P_k|} \delta_{y=\widehat{y}} \; \log \left(q_{t,j}^k \right) \tag{4.9}$$

where y is the ground-truth label, $\widehat{y}$ is the predicted label, δ is the Kronecker delta, and $\boldsymbol{q_t}$ is the corrected softmax output of the sample via Equation 4.8, defined as:

$$\boldsymbol{q_t} = \sigma \left(\left[\alpha_t^1 \boldsymbol{o_t^1} + \beta_t^1 \cdot \mathbf{1} \; ; \; \ldots \; ; \alpha_t^t \boldsymbol{o_t^t} + \beta_t^t \cdot \mathbf{1} \right] \right) \tag{4.10}$$

where σ is the softmax function.

All α_t^k, β_t^k pairs are optimized using validation samples from classes in N_t. We compare $AdBiC$ over BiC for our class IL setting in the evaluation section and show that the adaptation proposed here has a positive effect.

Transferring knowledge between datasets

The optimization of α and β parameters is impossible in class IL without memory, since exemplars of past classes are unavailable. To circumvent this problem, we hypothesize that optimal values of these parameters can be transferred between reference and target datasets, noted $\mathcal{X}^r$ and $\mathcal{X}$ respectively. The intuition is that these values are sufficiently stable despite dataset content variability. We create a set of reference datasets and perform a modified class IL training for them using the procedure described in Algorithm 1. The modification consists

Algorithm 1: Optimization of calibration parameters

inputs : $\mathcal{A}, \mathcal{X}_t^r$ for $t \in [1, \mathcal{T}]$ ▷ *reference dataset*
randomly initialize $\mathcal{M}_1$; $\mathcal{M}_1^* \leftarrow$ train$(\mathcal{A}; \mathcal{M}_1, \mathcal{X}_1^r)$;
for $t = 2 \ldots \mathcal{T}$ **do**
  $\mathcal{M}_t^* \leftarrow$ update$(\mathcal{A}; \mathcal{M}_{t-1}^*, \mathcal{X}_t^r)$;
  $\alpha_t^k \leftarrow 1,\ \beta_t^k \leftarrow 0$ for each $k \in [1, t]$;
  foreach $(\mathbf{x}, y) \in \mathcal{X}_t^r$ ▷ *validation set*
  do
    $\mathbf{o_t} \leftarrow \mathcal{M}_t^*(\mathbf{x})$;
    for $k = 1 \ldots t$ **do**
      $\mathbf{o_t^k} \leftarrow adBiC(\boldsymbol{o_t^k}) = \alpha_t^k \boldsymbol{o_t^k} + \beta_t^k \cdot \mathbf{1}$;
    end
    $\mathbf{q_t} \leftarrow \sigma(\mathbf{o_t})$;
    loss $\leftarrow \mathcal{L}(\boldsymbol{q_t}, y)$;
    $(\alpha_t^1, \beta_t^1, \ldots, \alpha_t^t, \beta_t^t) \leftarrow$ optimize(loss) ;
  end
end

in exploiting a validation set that includes exemplars of classes from all incremental states. Validation set storage is necessary in order to optimize the parameters from Equation 4.8 and is possible since reference dataset training is done offline. Note that backbone incremental models for $\mathcal{X}^r$ are trained without memory in order to simulate the IL setting of target datasets $\mathcal{X}$. We then store bias correction parameters optimized for reference datasets in order to perform transfer toward target datasets without using a memory. For each incremental state, we compute the average of α and β values over all the reference datasets. The obtained averages are used for score rectification on target datasets. This transfer uses the procedure described in Algorithm 2. The memory needed to store transferred parameters is negligible since we need $2 \times (2 + 3 + \ldots + \mathcal{T}) = (\mathcal{T} + 2) \times (\mathcal{T} - 1)$ floats for each dataset and $\mathcal{T}$ value. For $\mathcal{T} = \{5, 10, 20\}$ states, we thus only store 28, 108 and 418 floating-point values respectively.

In Figure 4.8, we illustrate optimal parameters obtained across $R = 10$ reference datasets which are further described in Section 4.3.3. We plot α^k and β^k values learned after $\mathcal{T} = 10$ IL states, using LwF [86] and $LUCIR$ [53] methods. Mean and standard deviations are presented for past and current incremental states in the final state of the IL process. The parameter ranges from Figure 4.8 confirm that, while optimal values do vary across datasets, this variation is rather low and calibration profiles remain similar. This opens up the possibility of parameter

Algorithm 2: $AdBiC$ inference

inputs : $\mathcal{A}$, (α_s^k, β_s^k) averaged on reference datasets for each $s \in [\![1, S]\!]$, $k \in [\![1, s]\!]$

inputs : $\mathcal{X}_t$ for $s \in [1, \mathcal{T}]$ ▷ *target dataset*

randomly initialize $\mathcal{M}_1$; $\mathcal{M}_1^* \leftarrow$ train$(\mathcal{A}; \mathcal{M}_1, \mathcal{X}_1)$;

for $s = 2 \ldots S$ **do**

  $\mathcal{M}_t^* \leftarrow$ update$(\mathcal{A}; \mathcal{M}_{t-1}^*, \mathcal{X}_t)$;

  foreach $(\mathbf{x}, y) \in \mathcal{X}_t$ ▷ *test set*

  do

    $\mathbf{o_t} \leftarrow \mathcal{M}_t^*(\mathbf{x})$;

    for $k = 1 \ldots t$ **do**

      $\mathbf{o_t^k} \leftarrow adBiC(\boldsymbol{o_t^k}) = \alpha_t^k \boldsymbol{o_t^k} + \beta_t^k \cdot \mathbf{1}$;

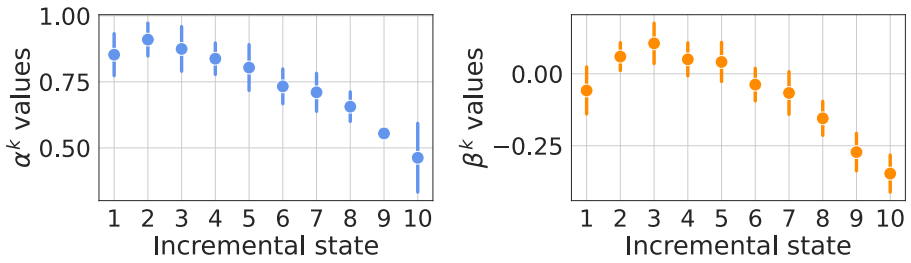
    end

    $\mathbf{q_t} \leftarrow \sigma(\mathbf{o_t})$;

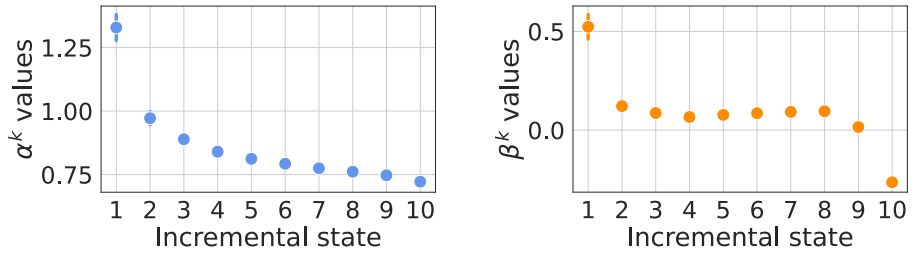
    $\hat{y} \leftarrow \underset{y \in [1, N_t]}{argmax}(\mathbf{q_t})$; ▷ *inference*

  end

end



(a) LwF [86]



(b) LUCIR [53]

Figure 4.8 – Averaged α^k (left) and β^k (right) values computed for $R = 10$ reference datasets using LwF and $LUCIR$, at the end of an incremental process with $\mathcal{T} = 10$ states.

transfer. When $R > 1$, a transfer function is needed to apply the parameters learned on a set of reference datasets to a target dataset. We transfer parameters using the averaged α_t^k and β_t^k values, obtained for the set of $\mathcal{X}^r$. In Subsection 4.3.4, we evaluate this transfer against an upper-bound oracle which selects the best $\mathcal{X}^r$ in each state.

The proposed approach adds a simple but effective linear layer to calibrate the predictions of backbone class-incremental methods. Consequently, it is applicable to any incremental learning method which works without memory. We test the genericity of the approach by applying it on top of four existing methods.

4.3.3 Experiments

— **Datasets** (Appendix B) - CIFAR-100 [74], IMN-100 [32], BIRDS-100 [32], FOOD-100 [20], and PLACES-100 [192].

— **Incremental states** - We use $\mathcal{T} = \{5, 10, 20\}$.

— **Evaluation measures -** Top-1 accuracy.

— **Baselines -** LwF [86], $LUCIR$ [53], $FT+$ [101], and SIW (corresponding to FT^{init}_{siw+mc} in Section 4.2).

We compare $AdBiC$ to BiC, the original linear layer from [174]. We also provide results with an optimal version of $AdBiC$, which is obtained via an oracle-based selection of the best-performing reference dataset for each IL state. This oracle is important as it indicates the potential supplementary gain obtainable with a parameter selection method more refined than the proposed one.

4.3.4 Results and discussion

Overall results

Table 4.4 provides results obtained with CIFAR-100, FOOD-100, IMN-100, and BIRDS-100. Results show that our method improves the performance of baseline methods for all but two of the configurations evaluated. The best overall performance before bias correction is obtained with LwF. This result confirms the conclusions of [101] regarding the strong performance of LwF in class IL without memory for medium-scale datasets. With $AdBiC$, $LUCIR$ performs generally better than LwF for $\mathcal{T} = 5$ and $\mathcal{T} = 10$, while LwF remains stronger with $\mathcal{T} = 20$ states. Results are particularly interesting for $LUCIR$, a method for which $AdBiC$ brings consistent gains (up to 16 accuracy points) in most configurations. Table 4.4 shows that $AdBiC$ also

Method	CIFAR-100			IMN-100			BIRDS-100			FOOD-100		
	$\mathcal{T}=5$	$\mathcal{T}=10$	$\mathcal{T}=20$	$\mathcal{T}=5$	$\mathcal{T}=10$	$\mathcal{T}=20$	$\mathcal{T}=5$	$\mathcal{T}=10$	$\mathcal{T}=20$	$\mathcal{T}=5$	$\mathcal{T}=10$	$\mathcal{T}=20$
LwF [86]	53.0	44.0	29.1	53.8	41.1	29.2	53.7	41.8	30.1	42.9	31.8	22.2
w/ BiC	54.0 + 1.0	45.5 + 1.5	30.8 + 1.7	54.7 + 0.9	42.5 + 1.4	31.1 + 1.9	54.6 + 0.9	43.1 + 1.3	31.8 + 1.7	43.4 + 0.5	32.6 + 0.8	23.8 + 1.6
w/ AdBiC	54.3 + 1.3	**46.4** + 2.4	**32.3** + 3.2	55.1 + 1.3	43.4 + 2.3	**32.3** + 3.1	55.0 + 1.3	44.0 + 2.2	**32.8** + 2.7	43.5 + 0.6	33.3 + 1.5	**24.7** + 2.5
w/ AdBiC + $\mathbb{O}$	54.9 + 1.9	47.3 + 3.3	32.6 + 3.5	55.9 + 2.1	44.2 + 3.1	33.1 + 3.9	55.8 + 2.1	44.8 + 3.0	33.3 + 3.2	44.0 + 1.1	34.2 + 2.4	25.3 + 3.1
LUCIR [53]	50.1	33.7	19.5	48.3	30.1	17.7	50.8	31.4	17.9	44.2	26.4	15.5
w/ BiC	52.5 + 2.4	37.1 + 3.4	22.4 + 2.9	54.9 + 6.6	36.8 + 6.7	21.8 + 4.1	56.0 + 5.2	37.7 + 6.3	20.6 + 2.7	49.9 + 5.7	31.5 + 5.1	17.2 + 1.7
w/ AdBiC	**54.8** + 4.7	42.2 + 8.5	28.4 + 8.9	**59.0** + 10.7	**46.1** + 16.0	27.3 + 9.6	**58.5** + 7.7	**45.4** + 14.0	27.3 + 9.4	**52.0** + 7.8	**37.1** + 10.7	17.7 + 2.2
w/ AdBiC + $\mathbb{O}$	55.5 + 5.4	43.6 + 9.9	31.2 + 11.7	59.4 + 11.1	46.6 + 16.5	29.7 + 12.0	59.0 + 8.2	46.0 + 14.6	28.8 + 10.9	52.6 + 8.4	38.2 + 11.8	21.0 + 5.5
SIW (Sec. 4.2)	29.9	22.7	14.8	32.6	23.3	15.1	30.6	23.2	14.9	29.4	21.6	14.1
w/ BiC	31.4 + 1.5	22.8 + 0.1	14.7 - 0.1	33.9 + 1.3	22.6 - 0.7	13.9 - 1.2	32.8 + 2.2	22.7 - 0.5	12.8 - 2.1	29.1 - 0.3	20.3 - 1.3	12.1 - 2.0
w/ AdBiC	31.7 + 1.8	24.1 + 1.4	15.8 + 1.0	35.1 + 2.5	24.5 + 1.2	15.0 - 0.1	33.0 + 2.4	25.2 + 2.0	15.3 + 0.4	30.9 + 1.5	21.3 - 0.3	14.5 + 0.4
w/ AdBiC + $\mathbb{O}$	32.8 + 2.9	25.0 + 2.3	16.5 + 1.7	36.4 + 3.8	25.7 + 2.4	16.1 + 1.0	34.4 + 3.8	26.2 + 3.0	16.3 + 1.4	31.5 + 2.1	22.6 + 1.0	15.1 + 1.0
FT+	28.9	22.6	14.5	31.7	23.2	14.6	29.7	23.3	13.5	28.7	21.1	13.3
w/ BiC	30.7 + 1.8	22.5 - 0.1	14.8 + 0.3	33.0 + 1.3	21.9 - 1.3	13.8 - 0.8	32.3 + 2.6	22.5 - 0.8	12.4 - 1.1	28.6 - 0.1	20.6 - 0.5	11.8 - 1.5
w/ AdBiC	31.9 + 3.0	23.6 + 1.0	15.0 + 0.5	34.9 + 3.2	23.7 + 0.5	15.7 + 1.1	34.0 + 4.3	25.0 + 1.7	14.2 + 0.7	30.8 + 2.1	22.2 + 1.1	14.2 + 0.9
w/ AdBiC + $\mathbb{O}$	32.5 + 3.6	24.6 + 2.0	15.9 + 1.4	35.7 + 4.0	24.9 + 1.7	16.2 + 1.6	34.5 + 4.8	25.7 + 2.4	15.4 + 1.9	31.3 + 2.6	22.7 + 1.6	14.5 + 1.2
Joint	72.7			75.5			80.9			71.03		

Table 4.4 – Average top-1 incremental accuracy using $\mathcal{T} = \{5, 10, 20\}$ states. Results are presented for each method without parameter transfer and with BiC and $AdBiC$ transfer. The gain (green) and loss (red) in accuracy obtained with parameter transfer are provided for each configuration. *Joint* is an upper bound obtained using a standard training with all data available. $\mathbb{O}$ denotes a choice of the reference dataset by oracle, in which the best reference dataset for each state is selected for transfer. Best results for each setting (excluding the oracle) are in bold. A graphical view of this table is provided in Figures 4.9 and 4.10. *Best viewed in color.*

improves the results of LwF in all configurations, albeit to a lesser extent compared to $LUCIR$. Interestingly, improvements for LwF are larger for $\mathcal{T} = 20$ states. This is the most challenging configuration since the model undergoes more rehearsal steps that cause more forgetting. $FT+$ [101] and SIW remove the distillation component for the class IL training process and exploit the weights of past classes learned in their initial state. $AdBiC$ improves results for these two methods in all but one configuration. However, their global performance is significantly lower than that of LwF and $LUCIR$, the two methods which make use of distillation. This result confirms the finding from Section 4.2 regarding the usefulness of the distillation term exploited by LwF and $LUCIR$ to stabilize IL training for medium scale datasets.

Results from Table 4.4 highlight the effectiveness of $AdBiC$ compared to BiC. $AdBiC$ has better accuracy in all tested configurations, with the most important gain over BiC obtained for $LUCIR$. It is also worth noting that $AdBiC$ improves results for SIW and $FT+$ in most configurations, while the corresponding results of BiC are mixed. The comparison of $AdBiC$ and BiC validates our hypothesis that finer-grained modeling of forgetting for past states is better than their uniform processing. It would be interesting to test the usefulness of $AdBiC$ in

the class IL with memory setting originally tested in [174].

We also compare $AdBiC$, which uses averaged α and β parameters, with the oracle selection of reference parameters (+ $\mathbb{O}$). The performance of $AdBiC$ is close to this upper bound for all tested methods, with a difference of less than one point in a majority of cases. This indicates that averaging bias correction parameters is an effective way to aggregate parameters learned from reference datasets. However, it would be interesting to investigate more refined ways to transfer parameters from reference to target datasets to further improve performance.

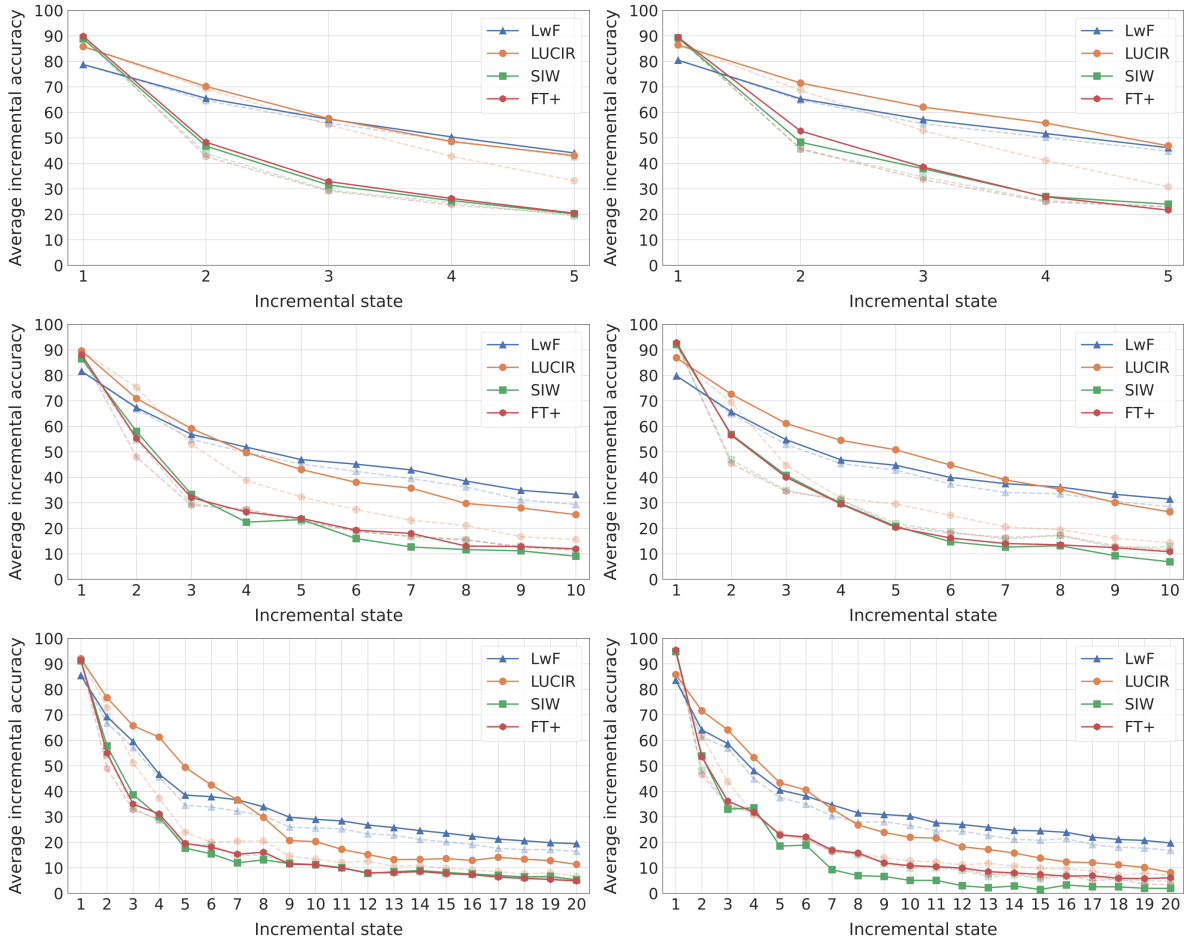


Figure 4.9 – Average accuracies in each state on CIFAR-100 ($left$) and IMN-100 ($right$) datasets with all backbone methods after $AdBiC$ correction, for $\mathcal{T} = 5$ (top), $\mathcal{T} = 10$ ($middle$) and $\mathcal{T} = 20$ ($bottom$) states. The accuracies without correction of the corresponding methods are provided in dotted lines (same colors). *Best viewed in color.*

The comparison of target datasets shows that the gain brought by $AdBiC$ is largest for IMN-100, followed by BIRDS-100, CIFAR-100 and FOOD-100. This is intuitive as IMN-100 has the closest distribution to that of reference datasets. BIRDS-100 is extracted from ImageNet

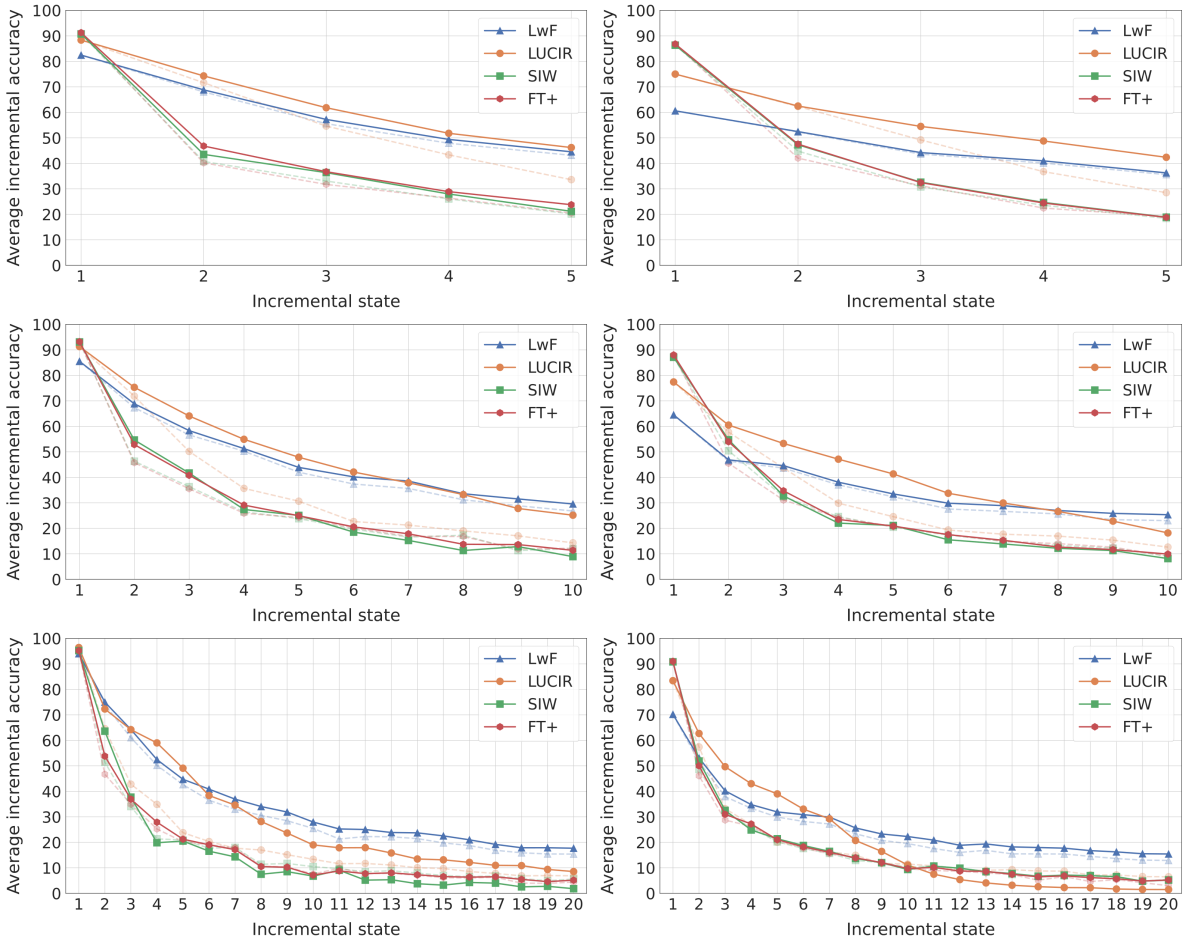


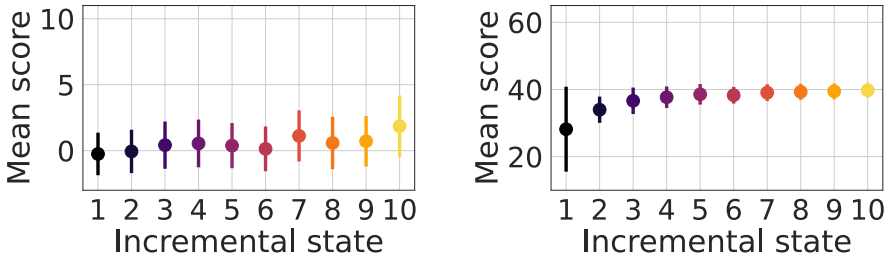
Figure 4.10 – Average accuracies in each state on BIRDS-100 ($left$) and FOOD-100 ($right$) datasets with all backbone methods after $AdBiC$ correction, for $\mathcal{T} = 5$ (top), $\mathcal{T} = 10$ ($middle$) and $\mathcal{T} = 20$ ($bottom$) states. The accuracies without correction of the corresponding methods are provided in dotted lines (same colors). *Best viewed in color*.

and, while topically different from reference datasets, was created using similar guidelines. The consistent improvements obtained with CIFAR-100 and FOOD-100, two datasets independent from ImageNet, show that the proposed transfer method is robust to data distribution changes. The performance gaps between IL results and $Joint$ are still wide, particularly for larger values of $\mathcal{T}$. This indicates that class IL without memory remains an open challenge.

Except for LwF, $AdBiC$ gains are generally larger for $\mathcal{T} = 5$ and $\mathcal{T} = 10$ compared to $\mathcal{T} = 20$. This result is consistent with past findings reported for bias correction methods [101, 174]. It is mainly explained by the fact that the size of validation sets needed to optimize $AdBiC$ parameters is smaller and thus less representative for larger values of $\mathcal{T}$. A larger number of states leads to a higher degree of forgetting. This makes the IL training process more challenging

and negatively affects the usefulness of the bias correction layer.

Effect of Adaptive Bias Correction



(a) LwF [86] (b) LUCIR [53]

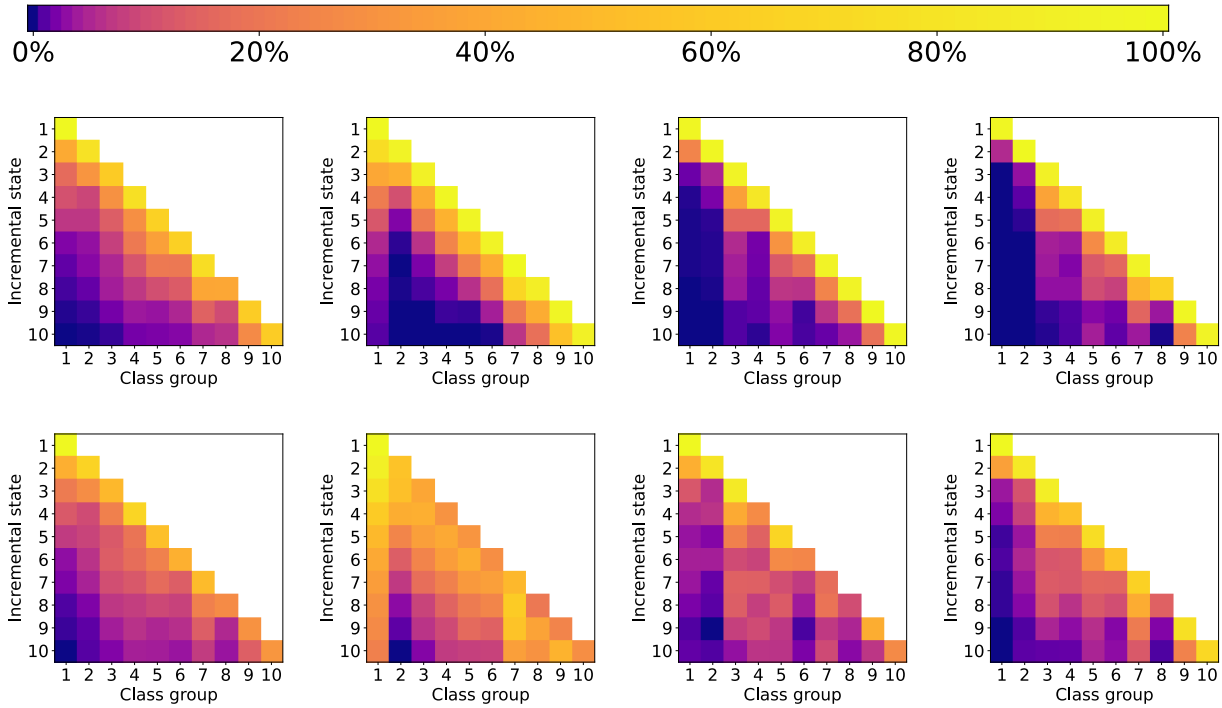
Figure 4.11 – Mean prediction scores for CIFAR-100 classes grouped by state at the end of the IL process for LwF and $LUCIR$ methods with $\mathcal{T} = 10$ states. Applying calibration parameters learned on reference datasets clearly reduces the imbalance of mean prediction scores and the bias toward recent classes, compared to raw scores from Figure 4.6.

Figure 4.11 provides a qualitative view of the effect of $AdBiC$ which complements numerical results from Table 4.4. It illustrates the predictions scores for LwF and $LUCIR$ after the application of bias correction with $AdBiC$. The correction is effective since the predictions associated to IL states are more balanced in Figure 4.11, compared to the raw predictions from Figure 4.6. The effect of calibration is particularly interesting for $LUCIR$, where mean prediction scores are balanced for states 3 to 10. We note that bias correction should ideally provide fully balanced mean prediction scores to give equal chances to classes learned in different states. Practical results show that some variation remains and is notably due to variable forgetting for past states and to the variable difficulty of learning different visual classes.

In Figure 4.12, we illustrate the effects of $AdBiC$ on state-wise accuracies, for all backbone IL methods evaluated in this work. Before adaptive correction (*top*), all methods provide strong performance on the last group of classes learned (represented by the diagonals). Their performance is generally poorer for past classes (under the diagonals). After correction (*bottom*), all methods perform better on past class groups (with a trade-off in accuracy on the last class group), resulting in higher overall performance.

Robustness of dataset knowledge transfer

We complement the results presented in Table 4.4 with two experiments that further evaluate the robustness of $AdBiC$. First, we test the effect of a different number of training images per



(a) LwF [86] (b) $LUCIR$ [53] (c) SIW (Sec.4.2) (d) $FT+$ [101]

Figure 4.12 – Accuracies per incremental state for each class group, for models trained with LwF, $LUCIR$, SIW and $FT+$ on CIFAR-100 for $\mathcal{T} = 10$ states, before (*top*) and after (*bottom*) $AdBiC$ correction. Each row represents an incremental state and each square the accuracy on a group of classes first learned in a specific state. In the first state, represented by the first rows of the matrices, models are only evaluated on the first-class group. In the second state, represented by the second rows, models are evaluated on the first two class groups, etc.

class for reference and target datasets. We remove 50% of training images for target datasets to test the transferability in this setting. The obtained results, presented in Table 4.8, indicate that performance gains are systematic for LwF and $LUCIR$, albeit lower compared to results in Table 4.4. Results are more mixed for SIW and $FT+$, but $AdBiC$ still has a positive effect in the majority of tested configurations. This experiment shows that the proposed dataset knowledge transfer approach is usable for reference and target datasets that have a different number of training samples per class. However, maintaining a low difference in dataset sizes is preferable in order to keep the transfer effective.

We provide, in Table 4.6, additional experiments with all and half of the training data on PLACES-100 dataset. Obtained results follow the same trend that those obtained on the other

datasets, despite the domain shift from ImageNet. This finding also confirms the robustness of $TransIL$ against domain variation.

Second, we assess the robustness of the method with respect to R, the number of available reference datasets. We vary R from 1 to 9 and perform transfer with ten random samplings for each R value. Results obtained on CIFAR-100 are reported in Table 4.8. Accuracy levels are remarkably stable for different values of R and significant gains are obtained even when using a single reference dataset. These results confirm that parameter transfer is effective even with few reference datasets, which is interesting considering that the computational cost of offline training is also reduced. Note that results on other datasets follow the same trend. We

Method	CIFAR-100			IMN-100			BIRDS-100			FOOD-100		
	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$
LwF [86]	41.3	33.3	23.3	45.6	33.5	23.8	44.6	34.0	23.2	29.5	23.3	17.3
w/ adBiC	42.1 + 0.8	34.8 + 1.5	25.0 + 1.7	46.7 + 1.1	35.3 + 1.8	25.6 + 1.8	45.5 + 0.9	35.4 + 1.4	25.2 + 2.0	29.9 + 0.4	24.3 + 1.0	18.7 + 1.4
LUCIR [53]	43.5	27.8	16.6	42.9	27.6	17.0	45.2	27.8	16.0	37.9	22.7	13.9
w/ adBiC	48.3 + 4.8	38.5 + 10.7	25.3 + 8.7	54.1 + 11.2	42.4 + 14.8	23.2 + 6.2	52.8 + 7.6	40.9 + 13.1	25.6 + 9.6	45.7 + 7.8	32.6 + 9.9	19.8 + 5.9
SIW [16]	31.7	21.6	13.7	32.1	22.7	14.4	29.7	22.8	14.1	28.4	18.7	13.5
w/ adBiC	33.7 + 2.0	22.5 + 0.9	14.0 + 0.3	35.0 + 2.9	22.6 - 0.1	12.2 - 2.2	32.1 + 2.4	23.7 + 0.9	13.5 - 0.6	29.9 + 1.5	16.9 - 1.8	13.3 - 0.2
FT+	30.4	21.5	12.9	31.2	22.2	12.0	29.2	22.8	12.2	27.4	18.2	11.6
w/ adBiC	32.0 + 1.6	21.4 - 0.1	13.4 + 0.5	34.8 + 3.6	21.2 - 1.0	13.7 + 1.7	31.9 + 2.7	23.0 + 0.2	13.6 + 1.4	28.8 + 1.4	16.2 - 2.0	12.2 + 0.6

Table 4.5 – Average top-1 IL accuracy with 50% of training images for target datasets. Gains are in green, losses are in red.

Method	PLACES-100			PLACES-100 (*halved*)		
	$\mathcal{T}=5$	$\mathcal{T}=10$	$\mathcal{T}=20$	$\mathcal{T}=5$	$\mathcal{T}=10$	$\mathcal{T}=20$
LwF [86]	43.3	35.1	25.9	35.4	27.7	21.5
w/ BiC	43.9 + 0.6	36.1 + 1.0	27.6 + 1.7	35.8 + 0.4	28.5 + 0.8	22.6 + 1.1
w/ AdBiC	**44.2** + 0.9	**36.6** + 1.5	**28.6** + 2.7	35.9 + 0.5	28.5 + 0.8	**23.6** + 2.1
w/ AdBiC + $\mathbb{O}$	44.6 + 1.3	37.5 + 2.4	29.3 + 3.4	36.5 + 1.1	29.2 + 1.5	24.3 + 2.8
LUCIR [53]	40.5	26.0	16.0	35.5	23.2	14.7
w/ BiC	42.6 + 2.1	29.9 + 3.9	18.0 + 2.0	38.3 + 2.8	26.9 + 3.7	16.5 + 1.8
w/ AdBiC	42.8 + 2.3	35.4 + 9.4	23.3 + 7.3	**40.5** + 5.0	**33.6** + 10.4	22.3 + 7.6
w/ AdBiC + $\mathbb{O}$	43.7 + 3.2	36.5 + 10.5	24.9 + 8.9	40.9 + 5.4	34.0 + 10.8	23.1 + 8.4

Table 4.6 – Average top-1 incremental accuracy using $\mathcal{T}=\{5,10,20\}$ states, for the PLACES-100 dataset with all and half of the training data. Gains obtained over the backbone method are given in green, and the best results for each setting in bold. *Best viewed in color*.

provide those of $LUCIR$ on FOOD-100 dataset in Table 4.7. Even though FOOD-100 dataset contains the largest domain shift with respect to reference datasets, $TranSIL$ shows stable results confirming its robustness.

$\mathcal{T} = 5$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	44.19	51.9 ± 0.4	52.0 ± 0.2	52.1 ± 0.2	52.0 ± 0.1	52.1 ± 0.1	52.0 ± 0.1	52.0 ± 0.1	52.0 ± 0.1	52.0 ± 0.1	52.0
$\mathcal{T} = 10$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	26.44	36.7 ± 0.7	36.9 ± 0.4	37.2 ± 0.4	37.2 ± 0.3	37.1 ± 0.2	37.0 ± 0.2	37.0 ± 0.1	37.1 ± 0.0	37.1 ± 0.1	37.1
$\mathcal{T} = 20$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	15.47	17.6 ± 1.2	17.5 ± 0.7	17.6 ± 0.7	17.8 ± 0.4	17.5 ± 0.3	17.7 ± 0.4	17.8 ± 0.3	17.6 ± 0.2	17.7 ± 0.1	17.7

Table 4.7 – Average top-1 incremental accuracy of $AdBiC$-corrected models trained incrementally on FOOD-100 with $LUCIR$, for $\mathcal{T} = \{5, 10, 20\}$ states, while varying the number R of reference datasets. For $R \leq 9$, results are averaged across 10 random samplings of the reference datasets (hence the std values). *Raw* is the accuracy of $LUCIR$ without bias correction.

4.3.5 Conclusion

We introduced a method that enables the use of bias correction methods for class IL without memory. This IL scenario is challenging because catastrophic forgetting is very strong in the absence of memory. The proposed method transfers bias correction parameters learned offline for reference datasets toward target datasets. Since reference dataset training is done offline, a validation memory that includes exemplars from all IL states can be exploited to optimize the bias correction layer. The evaluation provides comprehensive empirical support for the transferability of bias correction parameters. Performance is improved for all but two of the configurations tested, with gains of up to 16 top-1 accuracy points. Robustness evaluation shows that parameter transfer is efficient when only a few reference datasets are used for transfer. It is also usable when the number of training images per class in target datasets is different from that of available reference datasets. These last two findings are important in practice since the same reference datasets can be exploited in different incremental configurations. A second contribution relates to the modeling of the degree of forgetting associated with past states. While recency bias was already acknowledged [101], no difference was made between past classes learned in different IL states [174]. This is in part due to validation memory constraints which appear when the bias correction layer is optimized during the incremental process. Such constraints are reduced here since reference datasets training is done offline and a refined definition of the bias correction layer with specific parameters for each past state becomes possible. The comparison of the standard and of the proposed definition of the bias correction layer is favorable to the latter.

$\mathcal{T} = 5$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	53.0	54.3 ± 0.2	54.3 ± 0.2	54.3 ± 0.1	54.4 ± 0.1	54.3 ± 0.1	54.3 ± 0.1	54.3 ± 0.1	54.3 ± 0.1	54.3 ± 0.1	54.3
$\mathcal{T} = 10$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	44.0	46.2 ± 0.3	46.4 ± 0.2	46.4 ± 0.2	46.4 ± 0.2	46.4 ± 0.1	46.4 ± 0.1	46.5 ± 0.1	46.4 ± 0.1	46.4 ± 0.1	46.4
$\mathcal{T} = 20$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	29.1	31.8 ± 0.3	32.1 ± 0.1	32.1 ± 0.2	32.1 ± 0.1	32.2 ± 0.1	32.2 ± 0.1	32.3 ± 0.1	32.3 ± 0.1	32.3 ± 0.1	32.3

(a) LwF [86]

$\mathcal{T} = 5$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	50.1	54.7 ± 0.4	54.8 ± 0.3	54.8 ± 0.1	54.8 ± 0.1	54.8 ± 0.1	54.8 ± 0.1	54.8 ± 0.1	54.8 ± 0.1	54.8 ± 0.1	54.8
$\mathcal{T} = 10$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	33.7	42.0 ± 0.7	42.1 ± 0.3	42.2 ± 0.4	42.3 ± 0.3	42.2 ± 0.2	42.2 ± 0.2	42.2 ± 0.1	42.2 ± 0.1	42.2 ± 0.1	42.2
$\mathcal{T} = 20$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	19.5	27.5 ± 1.4	27.8 ± 0.7	27.8 ± 0.9	28.3 ± 0.4	28.5 ± 0.5	28.6 ± 0.6	28.5 ± 0.4	28.4 ± 0.3	28.4 ± 0.2	28.4

(b) LUCIR [53]

$\mathcal{T} = 5$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	29.9	31.6 ± 0.2	31.6 ± 0.2	31.6 ± 0.1	31.7 ± 0.2	31.7 ± 0.1	31.7 ± 0.1	31.7 ± 0.1	31.7 ± 0.1	31.7 ± 0.1	31.7
$\mathcal{T} = 10$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	22.7	23.8 ± 0.4	23.8 ± 0.2	23.9 ± 0.2	24.0 ± 0.2	23.9 ± 0.1	24.0 ± 0.1	24.1 ± 0.1	24.0 ± 0.1	24.1 ± 0.1	24.1
$\mathcal{T} = 20$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	14.8	15.7 ± 0.3	15.7 ± 0.2	15.7 ± 0.2	15.8 ± 0.1	15.8 ± 0.2	15.8 ± 0.1	15.8 ± 0.1	15.8 ± 0.1	15.8 ± 0.1	15.8

(c) SIW (Section 4.2)

$\mathcal{T} = 5$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	28.9	31.9 ± 0.2	32.0 ± 0.1	32.0 ± 0.1	32.0 ± 0.1	32.0 ± 0.1	32.0 ± 0.1	31.9 ± 0.1	32.0 ± 0.1	32.0 ± 0.1	31.9
$\mathcal{T} = 10$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	22.6	23.2 ± 0.4	23.5 ± 0.2	23.5 ± 0.2	23.6 ± 0.1	23.5 ± 0.2	23.6 ± 0.1	23.6 ± 0.1	23.6 ± 0.1	23.6 ± 0.1	23.6
$\mathcal{T} = 20$	Raw	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = 9$	$R = 10$
	14.5	14.8 ± 0.2	15.0 ± 0.1	15.0 ± 0.2	15.1 ± 0.1	15.0 ± 0.1	15.1 ± 0.1	15.1 ± 0.1	15.0 ± 0.1	15.0 ± 0.1	15.0

(d) FT+ [101]

Table 4.8 – Average top-1 incremental accuracy of $adBiC$-corrected models trained incrementally on CIFAR-100 with LwF, $LUCIR$, *SIW* and $FT+$, for $\mathcal{T} = \{5, 10, 20\}$ states, while varying the number R of reference datasets. For $R \leq 9$, results are averaged across 10 random samplings of the reference datasets. *Raw* is the accuracy of each method without bias correction.

4.4 A Comprehensive study of Class-Incremental learning without memory

4.4.1 Introduction

In many real-life applications, no memory of past classes is available. For instance, in medical data processing [167], this is often due to privacy issues. We complement the study conducted in Section 3.5 with a study of the behavior of the main algorithms which can be deployed in the absence of memory. Note that the following algorithms cannot be deployed: (1) all variants which exploit an external NEM classifier since exemplars are not available to build the classifiers for past classes; (2) BiC because it requires a validation set; (3) $ScaIL$ because it requires past exemplars for normalization; (4) $IL2M$ because the mean scores of past classes cannot be computed in the current state.

4.4.2 Tested approaches

Similarly to the study we conducted with memory in Section 3.5, we run experiments with the following works that are functional with or without memory: ***LUCIR***$^{\text{CNN}}$, ***FT***, ***FR***, ***DeeSIL***, ***REMIND***. In addition, we run experiments with:

— ***LwF*** [86] that uses distillation loss to encourage the model $\mathcal{M}_t$ to predict the same scores for past classes in the current state $\mathcal{S}_t$ than in the previous one. LwF constitutes the inspiration for all class IL algorithms [24, 53, 61, 174] and is equivalent to a version of $iCaRL$ without memory.

— ***Deep-SLDA*** [47] shares the same definition of $\mathcal{M}_t$ as $REMIND$, where $G(\cdot)$ is the first fifteen convolutional and three downsampling layers, and $F(\cdot)$ is the remaining two convolutional and one fully connected layers of a ResNet-18 [51]. At each incremental state, $Deep\text{-}SLDA$ updates a class-specific running mean vector and a running shared covariance matrix among classes. During inference, it assigns an image to the closest Gaussian in feature space defined by the mean class embeddings and the covariance matrix. $Deep\text{-}SLDA$ does not need to store past class data, and it is thus functional in the absence of memory.

On top of FT and LwF, we also apply $init$, followed either by $L2$ or SIW (Standardization of Initial Weights) normalization (Section 4.2), and finally by mc calibration. The other algorithms are not functional in a memoryless setting for the following reasons:

— $LUCIR^{NCM}$ [53] and FT^{NEM} (Section 3.3) - they need exemplars to compute past class-mean to deploy the NEM classifier.

— FT^{BAL} (Sections 3.3 and 3.4) - it needs past class exemplars to run the balancing phase. In the absence of memory, this approach becomes equivalent to FT and is heavily affected by catastrophic forgetting.

— *BiC* [174] since a validation set is needed to optimize the bias removal layer, this approach can only function if a memory is available.

— *ScaIL* and $IL2M$ (Sections 3.3 and 3.4) need a bounded memory to keep trace of past classes' statistics.

— FT^{th} - it needs a memory of the past to rectify past classes' scores.

Aligned with the analysis of different algorithm characteristics from Table 3.9, LwF performs model update and makes use of distillation loss to tackle catastrophic forgetting. This is not the case for $Deep\text{-}SLDA$ that is based on a fixed representation. Note that both methods do not apply any extra bias removal layer, and they do not require a memory from the past.

4.4.3 Experiments

— **Datasets** (Appendix B) - ILSVRC [135], LANDMARKS [108], VGGFACE2 [22], and CIFAR-100 [74].

— **Incremental states** - We test with $\mathcal{T} = \{10, 20, 50\}$.

— **Evaluation measures -** Top-5 accuracy [135] and G_{IL} measure (Subsection 3.4.4).

4.4.4 Results and Discussion

Role of knowledge distillation

When no memory is allowed for past classes, the experiments reported in Table 4.9 confirm those presented in [128]. There, LwF is clearly better than vanilla FT, and the usefulness of distillation is confirmed. Even without memory, the reuse of the embeddings of past classes from their initial states in FT^{init} is better than the sole use of knowledge distillation in LwF [86]. Only a more sophisticated scheme which combines distillation and an inter-class separation component in $LUCIR^{CNN}$ [53] outperforms FT^{init}, but stays way below FT^{init}_{siw+mc} that does not need distillation. Instead, it makes use of initial weights combined with standardization of all weights and mean state calibration.

Dataset	ILSVRC			VGGFACE2			LANDMARKS			CIFAR-100			G_{IL}
States	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	$\mathcal{T}$=10	$\mathcal{T}$=20	$\mathcal{T}$=50	
LwF	45.3	37.6	27.1	53.3	42.6	30.8	58.8	49.2	35.2	79.5	65.3	39.0	-34.72
LwF^{init}	47.1	39.9	32.2	58.1	50.8	40.5	55.7	50.2	39.8	79.4	67.9	42.8	-31.97
LwF^{init}_{L2}	24.5	39.7	32.0	57.1	50.7	40.5	52.1	50.5	40.0	79.5	68.1	43.3	-32.60
LwF^{init}_{siw}	54.0	45.8	35.1	70.4	59.3	45.2	61.0	53.8	42.2	**80.0**	**68.8**	44.6	-28.06
$LUCIR^{CNN}$	57.6	39.4	21.9	91.4	68.2	32.2	87.8	63.7	32.3	57.5	35.3	21.0	-24.75
FT	20.6	13.4	7.1	21.3	13.6	7.1	21.3	13.6	7.1	21.3	13.7	17.4	-54.91
FT^{init}	61.0	44.9	23.8	90.9	64.4	33.1	68.8	49.4	22.2	55.1	40.8	19.9	-28.99
FT^{init}_{L2}	51.6	43.3	34.5	76.8	66.8	55.1	61.4	52.5	39.2	47.5	39.3	22.5	-26.80
FT^{init}_{L2+mc}	53.6	42.7	35.6	86.9	71.4	53.6	66.2	52.6	37.9	52.6	43.1	18.2	-25.02
FT^{init}_{siw+mc}	64.4	54.3	41.4	88.6	84.1	62.6	79.5	64.5	43.2	59.7	44.3	18.4	-19.38
FR	74.0	66.9	49.2	88.7	83.0	54.4	**93.6**	88.1	71.2	73.1	54.8	27.4	-16.30
$DeeSIL$	**73.9**	**67.5**	53.9	**92.3**	**87.5**	75.1	**93.6**	**91.1**	**82.1**	65.2	63.4	32.3	**-9.22**
$REMIND$	62.2	56.3	44.4	86.8	81.4	69.2	84.5	79.6	69.0	52.7	40.5	25.7	-22.00
$Deep\text{-}SLDA$	70.3	64.5	**56.0**	90.2	85.4	**78.2**	89.3	86.4	81.3	68.9	64.4	**54.5**	-15.40
$Joint$		92.3			99.2			99.1			91.2		-

Table 4.9 – Top-5 average incremental accuracy (%) for IL methods without memory for past classes and different numbers of IL states. Best results are in bold.

In Table 4.10, we present the distribution of correct and erroneous predictions across incremental states to have a better understanding of the behavior of distillation in IL. Results are given for LwF [86], $LUCIR^{CNN}$ [53] and FT^{init}_{L2} which implement classical distillation, features-based distillation plus inter-class separation and the reuse of L2-normalized initial classifier weights, respectively. The authors of [61] noted that distillation induces a bias among past classes, which leads to confusion between their predictions. This finding is confirmed by the large number of past-past class confusions $e(p,p)$ associated to LwF in Table 4.10. When advancing in incremental states, the number of past classes increases. If an error is made while training the model $\mathcal{M}_1$ using the activations of $\mathcal{M}_1$ as soft targets, it will be passed on to all the subsequent incremental states. Consequently, the percentage of $e(p,p)$ errors in Table 4.10 increases in later incremental states. However, the percentage of $e(p,p)$ errors is smaller for LwF and $LUCIR^{CNN}$ compared to FT^{init}_{L2} indicating that distillation has a positive effect of past classes. The addition of the interclass separation in $LUCIR^{CNN}$ removes a part of $e(p,p)$, and the overall increases significantly $c(p)$, the number of correct predictions for past classes. We note that, since distillation operates on past class scores, the bias in favor of new classes is not handled by LwF and $LUCIR^{CNN}$. Consequently, $e(p,n)$ is the main type of error for the distillation-based methods. This is explained in Sections 3.3 and 3.4 by the fact that the model is biased towards new classes, leading to predict past images as belonging to new classes. This bias is caused by the fact that new classes are well learned with all their data.

Incremental states		$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$
ILSVRC										
$LUCIR^{CNN}$	$c(p)$	62.4	46.2	36.7	29.1	23.1	18.8	15.8	14.2	13.1
	$e(p,p)$	3.8	9.6	16.2	21.3	23.6	25.2	26.0	30.2	27.9
	$e(p,n)$	33.7	44.3	47.1	49.6	53.3	56.0	58.2	55.7	59.0
	$c(n)$	77.9	79.2	75.2	75.2	77.9	79.4	76.9	80.5	77.7
	$e(n,n)$	17.6	15.7	19.1	17.7	16.5	15.2	16.7	13.9	15.5
	$e(n,p)$	4.5	5.1	5.6	7.0	5.5	5.4	6.4	5.6	6.8
FT^{init}_{L2}	$c(p)$	5.4	17.1	15.6	16.6	15.3	16.6	13.7	13.1	14.9
	$e(p,p)$	0.6	12.8	10.5	28.2	20.8	47.0	27.6	28.5	55.0
	$e(p,n)$	94.0	70.1	73.9	55.2	63.8	36.4	58.8	58.4	30.1
	$c(n)$	83.7	85.5	81.2	79.9	82.4	78.5	80.9	82.3	74.4
	$e(n,n)$	16.1	11.1	16.5	12.0	13.6	7.2	11.9	10.9	5.9
	$e(n,p)$	0.2	3.4	2.3	8.1	4.0	14.4	7.2	6.7	19.7
LwF	$c(p)$	13.7	11.7	10.6	8.6	6.3	5.6	5.0	4.6	4.4
	$e(p,p)$	6.8	17.8	25.4	25.2	27.1	28.1	32.1	33.9	33.8
	$e(p,n)$	79.6	70.5	64.0	66.1	66.6	66.4	62.9	61.5	61.7
	$c(n)$	70.7	73.4	68.9	70.0	72.7	74.1	70.4	73.7	71.2
	$e(n,n)$	23.2	17.5	19.6	18.0	15.5	15.4	17.1	13.7	14.4
	$e(n,p)$	6.1	9.0	11.4	12.1	11.7	10.5	12.5	12.6	14.4
CIFAR-100										
$LUCIR^{CNN}$	$c(p)$	56.7	39.4	25.7	16.8	14.4	14.5	9.4	9.8	6.7
	$e(p,p)$	2.1	10.9	12.8	9.7	20.4	24.5	17.1	23.0	20.0
	$e(p,n)$	41.2	49.6	61.4	73.5	65.1	61.0	73.6	67.2	73.4
	$c(n)$	78.9	84.4	86.9	86.3	86.5	85.3	85.0	85.2	88.2
	$e(n,n)$	13.0	11.2	11.4	11.8	11.4	9.5	11.0	10.3	8.7
	$e(n,p)$	8.1	4.4	1.7	1.9	2.1	5.2	4.0	4.5	3.1
FT^{init}_{L2}	$c(p)$	0.8	6.7	9.9	8.5	8.1	5.4	7.8	7.5	5.8
	$e(p,p)$	0.0	4.8	9.4	10.6	25.6	12.3	23.2	38.5	19.2
	$e(p,n)$	99.2	88.5	80.7	80.9	66.3	82.3	69.0	54.0	75.0
	$c(n)$	86.7	89.2	87.8	88.2	85.2	88.1	84.0	85.3	90.7
	$e(n,n)$	13.2	9.5	9.2	9.1	8.9	10.2	8.9	6.1	4.6
	$e(n,p)$	0.1	1.3	3.0	2.7	5.9	1.7	7.1	8.6	4.7
LwF	$c(p)$	57.3	47.5	40.3	31.1	28.7	26.6	23.8	22.0	17.7
	$e(p,p)$	8.0	22.6	26.2	32.1	39.8	45.5	46.5	48.5	46.0
	$e(p,n)$	34.7	29.9	33.5	36.8	31.6	27.9	29.7	29.5	36.2
	$c(n)$	72.7	76.4	74.4	74.9	72.6	75.4	70.6	74.6	82.3
	$e(n,n)$	9.9	5.1	3.9	6.9	5.0	4.0	5.6	3.2	2.4
	$e(n,p)$	17.4	18.5	21.7	18.2	22.4	20.6	23.8	22.2	15.3

Table 4.10 – Top-1 correct and wrong classification for $LUCIR^{CNN}$, FT^{init}_{L2} and LwF for ILSVRC and CIFAR-100 with $\mathcal{T} = 10$ and $|\mathcal{K}| = 0$.

In Table 4.9, the comparison of LwF, $LUCIR^{CNN}$ and FT^{init}_{L2} for $\mathcal{T} = 10$ shows that LwF has the lowest and highest performance for ILSVRC and CIFAR-100 respectively. The detailed view in Table 4.10 gives further insights into the structure of results. In this table, $c(p)$ and $c(n)$ are the correct classification for past/new classes. $e(p,p)$ and $e(p,n)$ are erroneous

classifications for test samples of past classes mistaken for other past classes and new classes respectively. $e(n,p)$ and $e(n,n)$ are erroneous classifications for test samples of new classes mistaken for past classes and other new classes respectively. Since the number of test images varies across IL states, percentages are calculated separately for test images of past and new classes in each $\mathcal{S}_t$ to get a quick view of the relative importance of each type of errors. $c(p)$, $e(p,p)$, and $e(p,n)$ sum to 100% on each column, as do $c(n)$, $e(n,n)$, and $e(n,p)$.

We note that $c(p)$ is higher and $e(p,p)$ is lower for CIFAR-100 compared to ILSVRC, indicating that distillation is much more efficient at a smaller scale. We conclude that distillation is not always useful in IL. The performance of distilled networks depends on the size of the dataset, the number of incremental states, and the presence or not of the bounded memory of the past. It should be used only when the incremental task is known to be characterized by a favorable combination of these parameters.

Without memory, $iCaRL$ becomes LwF, the method which inspired more recent works using distillation in IL. LwF^{init}, LwF^{init}_{L2}, and LwF^{init}_{siw} test if the basic hypothesis of $ScaIL$ regarding the reuse of initial classifier weights applies to a method that integrates distillation. This use of initial weight leads to a 3 points G_{IL} gain compared to classical LwF. Further $L2$ normalization in LwF^{init}_{L2} is not efficient. However, standardization of weights in LwF^{init}_{siw} improves the results of LwF^{init} with 3.9 points. $LUCIR^{CNN}$ implements a more sophisticated scheme to counter catastrophic forgetting by adding cosine normalization and inter-class separation on top of knowledge distillation. The two additional components have a significant decisive role since they provide a 10 points gain compared to LwF. Vanilla FT has no component to counter catastrophic forgetting and it has the worst overall performance. The use of initial classifiers of past classes in FT^{init} provides a very consequent gain over simple FT. The application of $init$ is much more efficient for FT compared to LwF and even gives nearly the same results as LwF^{init}_{siw}, the best variant of LwF. The use of $L2$ normalization in FT^{init}_{L2} improves the results of FT^{init} with 2 G_{IL} points, while adding the mean state calibration mc in FT^{init}_{L2+mc} further gains another 1 G_{IL} points over FT^{init}_{L2}. The best fine tuning based approach without memory is FT^{init}_{siw+mc} from Section 4.2. This approach outperforms the other methods with a large margin.

Overall, the best results are obtained with the fixed-representation methods because their dependence on past exemplars is much lower compared to fine-tuning-based methods. In order, the best global score is obtained by $DeeSIL$, $Deep\text{-}SLDA$, FR and $REMIND$. As we mentioned, $DeeSIL$ is easier to optimize compared to FR and has a comparable accuracy variation with $Deep\text{-}SLDA$ for most tested configurations. However, $DeeSIL$ provides the best perfor-

mance when no memory is allowed. The performance of fixed-representation methods drops when the number of incremental states increases because the initial state includes a lower number of classes. This is notably the case for CIFAR-100, the smallest dataset tested, where the fixed-representations have lower performance compared to all variants of LwF for all tested T values. However, FR, $REMIND$, $Deep\text{-}SLDA$ and $DeeSIL$ have consequently better performance for ILSVRC, VGGFACE2, and LANDMARKS where their initial representations are trained with at least 20 classes.

The analysis of individual datasets shows that LwF variants have a strong performance for CIFAR-100, the smallest one among the four tested. LwF scales worse than $LUCIR^{CNN}$, which is better for the three larger datasets. The performance inversion is probably explained by the handling of inter-class separation in $LUCIR$. This indicates that knowledge distillation alone does not scale well because when the number of past classes increases, the confusions between them hamper the performance of the method.

4.4.5 Conclusion

When no memory is allowed, fixed-representation methods are globally much more competitive than fine-tuning ones while also being simpler and faster to deploy. They are particularly advantageous for large datasets, where distillation-based methods fail to scale-up. This finding is surprising insofar fixed-representation methods exploit a classical transfer learning scheme. They do not update models across incremental states and were considered less apt for usage in IL without memory [128]. We note that these methods work better than distillation-based IL algorithms even when initial representations are learned with a few dozens of classes.

4.5 General Conclusion

In this chapter, we showed that class IL becomes a more challenging task without a memory of the past because no images are replayed for past classes, leading to a stronger forgetting. We proposed SIW and $TransIL$, two IL methods that can be applied on top of many other methods. The former necessitates a backbone method where the classification layer is fully connected, and the latter necessitates a backbone method designed for bias correction. Obtained results are promising and more efforts are needed from the community to further improve IL without memory. This scenario is more interesting in real-life problems where the memory is likely to be unavailable.

CHAPTER 5

CONCLUSION

5.1 General discussion

In this thesis, we are interested in tackling the catastrophic forgetting problem for class-incremental learning. We defined in Chapter 1 six desirable properties that qualify an AI system to be class-incremental. Mainly its ability to: (1) correctly classify both past and new images, (2) be scalable, (3) integrate new classes promptly, (4) make a compromise between plasticity and stability, (5) keep the model complexity constant through the time and (6) work with or without a bounded memory of the past. These properties are hard to satisfy at the same time. Depending on the application domain, one or many properties can be discarded in order for the system to work. For instance, in health applications, it is often the case that data cannot be passed between incremental states for regulatory reasons. In this case, it is crucial to update the model's capacity without having access to data from the past. Alternatively, in embedding systems, both memory and computational requirements should be minimized. The scalability in such systems might not be as crucial as the timeliness to integrate new knowledge while keeping the resources bounded. When there is no significant domain shift between classes learned offline and classes learned incrementally, fixed representations are a good option because they keep the model stable across time. This can, for instance, be the case of face recognition models integrated into media applications. Here, the update needs to be very fast to follow the pace of news, and an initial model can ensure an efficient transfer of deep face representations. They are particularly interesting when learned on sufficiently large and diversified initial datasets [160, 161]. However, when the classes learned incrementally are visually different from the initial ones, the performance of fixed representations drops drastically, and the knowledge transfer fails. Ultimately, the choice between different types of incremental learning methods is driven by the concrete constraints of the envisioned application.

Contributions overview

In this thesis, we propose one fixed-representation-based method which works with and without memory of the past, and four fine-tuning-based methods, out of which two require a memory of the past, and the other two do not. While simple, our methods achieve competitive results compared to methods from the state of the art.

$DeeSIL$ [12] consists of (1) a deep feature extractor (*DFE*) trained on the classes of the first non-incremental state, and (2) a set of SVMs [19], where the number is equal to the number of classes learned in the incremental states. the *DFE* is used to extract features of images of all classes seen during all the incremental process, and the SVMs are trained using the extracted feature vectors. Each SVM is used to learn one class, where the positives are the feature vectors of the class, and the negatives are selected depending if we use a memory of the past or not:

— *When a memory of the past is allowed*, the negatives are selected among the feature vectors of past classes. This is interesting insofar as the features are much more compact then the images themselves.

— *When no memory of the past is allowed*, the negatives are selected among the feature vectors of classes learned in the same incremental state as the class that is currently being learned

$DeeSIL$ is useful in class-incremental learning when the data is shuffled to be i.i.d. However, it fails in task-incremental scenarios, where the incremental tasks are likely to be visually different from the first task. The other proposed FT-based approaches are presented briefly in the next two subsections.

When a memory is allowed, we found that the widely used distillation loss [128, 24, 53, 52, 61] is not necessary for large-scale IL systems. The conducted experiments showed that this distillation is useful for the medium-scale CIFAR-100 dataset. However, when tested on large datasets such as ILSVRC [135], Google Landmarks [108] and VGGFace2 [22] that contain 1000 classes, distillation leads to confusions among past classes. Practically, this means that the number of past testing images predicted as belonging to wrong past classes is large compared to the other type of errors. These results are at odds with the ones usually reported in class IL literature which originate from a biased comparison from [128]. There, the authors claimed that $iCaRL$ is more competitive than a vanilla fine tuning on ILSVRC. Their comparison was not fair as vanilla fine tuning was implemented without a memory of the past, while $iCaRL$ used such a memory. Our findings were later confirmed by [101] that did extensive experiments on a range of class IL systems with and without memory.

The use of a bounded memory of the past leads to an imbalanced learning that worsens as more incremental states are encountered. This is mainly because new classes are always learned with all their data, while fewer exemplars are replayed for past classes at each incremental state. This imbalance in data leads to *recency bias*, the tendency of neural networks to be biased in favor of the most recent task (or group of classes). In practice, recency bias refers to the imbalance in favor of new classes compared to past ones during inference. The experiments conducted in Chapter 3 show that the gap between the mean prediction scores of past and new classes is large and needs to be addressed. We thus propose:

— $Il2M$ [13] - aims to reduce the prediction bias by rectifying past classes' scores and make them more comparable to those of new classes. The scores rectification is done using scores statistics saved in a secondary memory.

— $ScaIL$ [14] - operates at the level of the weights matrix of the last fully connected layer of a CNN. We show that using past classes embeddings from the initial states in which they were learned for the first time is beneficial. We further add weights normalization based on classes statistics to make past and new classes embeddings comparable.

Both $IL2M$ and $ScaIL$ showed good performance, with the latter performing better in most cases. This is not surprising because $ScaIL$ handles catastrophic forgetting at a deeper level instead of directly rectifying prediction scores.

Without a memory of the past, our experiments confirmed that that distillation is beneficial for medium-scale and large-scale datasets. LwF [86] and $LUCIR$ [53] clearly outperform vanilla fine tuning based approaches. $DeeSIL$ is the best memoryless IL system that we proposed. It outperforms methods that use distillation (such as LwF[86]) or even more sophisticated objectives ($LUCIR$ [53]). It also outperforms online learning methods that are based on a fixed-representation ($Deep$-$SLDA$ [47] and $REMIND$ [48]). The latter two methods are interesting when the task boundaries are unknown, and the system can only see each training example once during all the training process. When evaluated online [47], these methods are competitive against the state of the art.

When using fine tuning as a backbone for class-IL, we propose two methods that can be implemented on top of memoryless IL methods to further improve their performance:

— SIW [16] - this approach is inspired by $ScaIL$. It makes use of past classes embeddings from their initial states. The main difference with $ScaIL$ is the normalization of the modified weights matrix. In SIW, we use a simple standardization of all classes embeddings since they follow a normal distribution. This approach largely improves results of a

vanilla fine tuning, LwF [86], and a memoryless version of $LUCIR$ [53].

— $TransIL$ [155] - adapts BiC [174] for usage in a memoryless IL setting. BiC shows very competitive performance against the most recent state-of-the-art methods. However, the dependency of this method on a validation set makes it unusable in incremental learning without memory since no exemplars of past classes are available. Instead, we propose to train BiC offline on reference datasets using a validation set and later transfer the calibration parameters in an online fashion to target datasets trained incrementally. The method achieves consequent gain compared to backbone methods, especially $LUCIR$ [53] and LwF [86].

The proposed methods are very useful when we cannot keep past images for memory issues or confidential restrictions. While the FT-based methods behave well compared to others from this category, they still lag behind fixed representations, such as $DeeSIL$. This finding is somewhat frustrating but also a recall that simple methods should be tried first.

Memory footprint of proposed methods

In Table 5.1, we present an overview of the additional storage (AS) of all our proposed methods when learning a total of $N_{\mathcal{T}} = 1000$ classes. This storage mainly concerns the secondary memory that we use to save statistics, embeddings, or calibration parameters.

Method	Additional Storage (AS) in float	AS for $N_{\mathcal{T}} = 1000$				
		$\mathcal{T} = 5$	$\mathcal{T} = 10$	$\mathcal{T} = 20$	$\mathcal{T} = 50$	$\mathcal{T} = 100$
$DeeSIL$	0	0	0	0	0	0
$IL2M$	$\mathcal{T} + N_{\mathcal{T}}$	4.02 KB	4.04 KB	4.08 KB	4.2 KB	4.4 KB
$ScaIL$	$N_{\mathcal{T}} \times D$	2.05 MB	2.05 MB	2.05 MB	2.05 MB	2.05 MB
SIW	$\mathcal{T} + N_{\mathcal{T}} \times D$	2.05 MB	2.05 MB	2.05 MB	2.05 MB	2.05 MB
$TransIL$ ($AdBiC$)	$R \times (\mathcal{T} + 2) \times (\mathcal{T} - 1)$	1.12 KB	4.32 KB	16.72 KB	101.92 KB	403.92 KB
$TransIL$ (BiC)	$2 \times R \times (\mathcal{T} - 1)$	320 B	720 B	1.52 KB	3.92 KB	7.92 KB

Table 5.1 – Additional Storage (AS) in floats of our proposed IL approaches with and without a bounded memory $\mathcal{K}$ of the past. In our experiments conducted in Chapters 3 and 4, we use a ResNet-18 architecture where $D = 512$ is the size of the feature vector at the penultimate layer. For $TransIL$, we use $R = 10$ reference datasets. If we save the mean values of calibration parameters, we divide the memory footprint of $TranSIL$ by R. Note that one floating-point occupies 4 Bytes in the disk.

$DeeSIL$ does not use a secondary memory. Therefore, it has no additional storage. In $IL2M$, we store one float for each class (its mean classification score) and one float by state

(the model's confidence). The additional footprint is thus negligible. For instance, if $\mathcal{T} = 100$, only 4.4 KB is needed for the secondary memory. In $ScaIL$, we save class embeddings in their initial states, where the memory needed does not depend on the number of incremental states. To learn 1000 classes, only 2 MB is needed as additional storage. SIW shares the same spirit as $ScaIL$ insofar as we save the class embeddings. In addition, we save one float by state (the model's confidence) to perform state calibration, which is negligible.

Finally, $TransIL$ stores calibration parameters whose memory footprint depends on (1) the bias correction method used (BiC or $AdBiC$), (2) the number of incremental states, and (3) if all α and β parameters are saved, or only their respective means. The worst case would be to use $AdBiC$ as a bias correction method and saving all calibration parameters. To learn 1000 classes while varying the number of states between 5 and 100 requires a small memory size (from 1.12 KB to 404 KB). Alternatively, the best case would be to use BiC [174] as a bias correction method and saving only the average values of calibration parameters. In this case, the memory footprint varies between 32 Bytes and 792 Bytes. In both best and worst cases, the memory used by $TransIL$ is negligible.

Knowledge-distillation based approaches [128, 24, 53, 174, 189] require saving both previous (teacher) and current (student) models. However, all our methods require saving the current model only (86 Megabytes for a ResNet [51] architecture), thus halving the storage needed to run other methods. For $DeeSIL$, we store one SVM per class in addition to the feature extractor. The size of one SVM is 8 Kilobytes and is equivalent to the increase in the CNN architecture when we add one neuron per class in the classification layer. Thus, $DeeSIL$ also keeps the memory of the model bounded.

Contrarily to SIW and $TransIL$, both $IL2M$ and $ScaIL$ need a memory of the past. The latter occupies 2.29 GB when having $\mathcal{K} = 20000$ exemplars for the ILSVRC [135] dataset. Note that this size corresponds to the size of the real image in the disk and can be reduced by compressing the images files. The memory is optional in $DeeSIL$. When no memory is allowed, no additional storage is needed for the past. When a memory is allowed, feature vectors are stored instead of images. The advantage of feature vectors is their limited size compared to images, as they contain, in the case of a ResNet-18 [51], 512 floating points. In such a case, the memory occupies 40 MB only. This size is negligible compared to the images memory used by other approaches, making $DeeSIL$ a good choice for applications where memory restrictions are imposed.

State-of-the-art schema after our proposed approaches

As we showed above, our proposed methods have a varying storage size for the model, exemplars memory, and secondary memory. This gives a wide choice to the user to select the method that is more suitable for the application domain. We bring the Figure 2.1 from Chapter 2 and enrich it with our proposed methods.

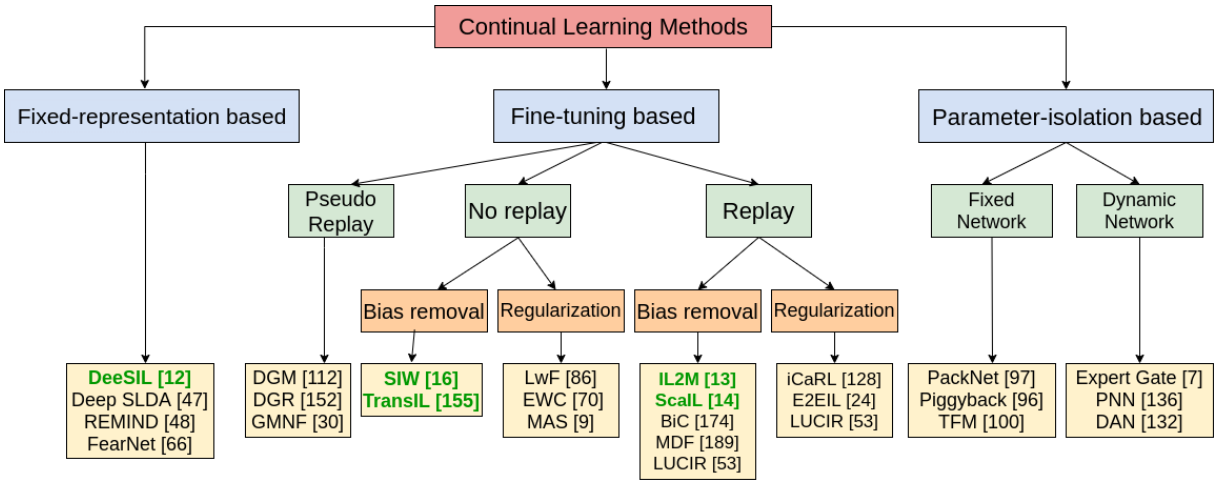


Figure 5.1 – State-of-the-art approaches after adding our proposed approaches (in red). To be compared to Figure 2.1. *Best viewed in color.*

To the best of our knowledge, $DeeSIL$ is one of the first methods that apply transfer learning on class-incremental learning. Other methods were released later and are based on a combination of transfer learning and bio-inspiration: $FearNet$ [66], $REMIND$ [48], and $DeepSLDA$ [47]. The first method combines TL with auto-encoders to generate past class data, while the two others use memory consolidation to learn online. $IL2M$ and $ScaIL$ are from the few works that handle class-incremental learning as an imbalanced learning, and combine the exemplars memory with bias-removal techniques to tackle catastrophic forgetting. SIW and $TransIL$ tackle the recency bias without having access to exemplars memory. Globally, our work is focused on proposing methods that ally simplicity and strong performance in class IL with and without memory. Their empirical comparison with many competing methods shows that this global objective is achieved since state-of-the-art performance, at the time of publication, is achieved in most cases while reducing the global memory footprint.

5.2 Directions for future research

The performance of incremental learning methods is getting closer to that of standard learning, where all data from all classes is assumed to be available at all times. However, the catastrophic forgetting problem is far from being solved, especially at a large scale. Most of the existing systems [128, 24, 53, 52, 61] use a memory of the past. Note that, in real-life scenarios, a memory is not always allowed. In addition, memory-dependent algorithms tend to do not scale well when more and more classes won't have representatives to be replayed for them when the maximum size of the memory is reached.

Class-incremental learning without memory is understudied in literature and we believe that more effort should be allocated to it. To the best of knowledge the only available such methods are those we proposed in addition to: *Learning without Memorizing* [34], *Learning without Forgetting* (LwF) [86], $Deep\text{-}SLDA$ [47], *Semantic Drift Compensation* (SDC) [182], and *Always Be Dreaming* (AbD) [156]. We could adapt other methods to a memoryless setting ($REMIND$ [48] and $LUCIR$ [53]), but this number of IL systems is negligible compared to those that rely on a memory of the past.

Handling class IL as an imbalanced learning problem provides very interesting results with or without the use of a distillation component ($IL2M$ (Section 3.3) $ScaIL$ (Section 3.4), BiC [174] and $AdBiC$ (Section 4.3)). It would be interesting to investigate more sophisticated bias reduction schemes to improve performance further. Also, a more in-depth investigation of why distillation fails to work for large-scale datasets is needed. The empirical findings reported in this thesis should be complemented with more theoretical analysis to improve the understanding of this phenomenon.

Last but not least, the results obtained with a herding-based selection of exemplars are better than a random selection for all methods tested. Further work in this direction could follow up on [89] and investigate in more depth which exemplar distribution is optimal for replay.

Finally, the evaluation scenario should be made more realistic by:

— dropping the strong hypothesis that new data are readily annotated when they are streamed

— using a variable number of classes in each incremental state. This case is more realistic as the data can be streamed in a random way

— working with imbalanced datasets which are more likely to occur in real-life applications than the controlled datasets tested until now by the community

— handling individual samples instead of classes, where samples of past and new classes are mixed (domain-incremental learning [50])

— aggregating works from class-incremental learning and new class discovery [190, 191] to make the whole process more realistic. Here, we would have systems that learn dynamically in a much more autonomous way. This is one step further beyond combining class-incremental learning with active learning

The above-mentioned items are interesting and should be addressed for seamless usage of IL in practice as in [105].

APPENDIX A

LIST OF PUBLICATIONS

Research that led to this thesis has appeared previously in the following publications. † refers to *equal contributions*.

A.1 Journal Papers

— Belouadah, E., Popescu, A. and Kanellos, I., 2020. **A Comprehensive Study of Class Incremental Learning Algorithms for Visual Tasks**. Neural Networks, t. 135, pp. 38-54.

— Aggarwal, U., Popescu, A., Belouadah, E. and Hudelot, C., 2020. **A Comparative Study of Calibration Methods for Imbalanced Class Incremental Learning**. Multimedia Tools and Applications.

A.2 Main Conference Papers

— Slim, H.†, Belouadah, E.†, Popescu, A., Onchis, D., "**TransIL: Dataset Knowledge Transfer for Class-Incremental Learning without Memory**". The IEEE Winter Conference on Applications of Computer Vision (WACV 2022).

— Belouadah, E., Popescu, A. and Kanellos, I., 2020. **Initial Classifier Weights Replay for Memoryless Class Incremental Learning**. British Machine Vision Conference (BMVC 2020).

— Belouadah, E. and Popescu, A., 2020. **ScaIL: Classifier Weights Scaling for Class Incremental Learning**. The IEEE Winter Conference on Applications of Computer Vision (WACV 2020). pp. 1266-1275.

— Belouadah, E. and Popescu, A., 2019. **IL2M: Class incremental learning with dual memory**. Proceedings of the IEEE International Conference on Computer Vision (ICCV 2019). pp. 583-592.

A.3 Workshop Papers

— Belouadah, E., Popescu, A., Aggarwal, U. and Saci, L., 2020. **Active Class Incremental Learning for Imbalanced Datasets**. Proceeding of the European Conference on Computer Vision workshops (W-ECCV 2020).

— Belouadah, E., Popescu, A., 2018. **DeeSIL: Deep-Shallow Incremental Learning**. Proceeding of the European Conference on Computer Vision workshops (W-ECCV 2018).

APPENDIX B

DATASETS AND IMPLEMENTATION DETAILS

B.1 Datasets

To facilitate reproducibility, we chose to perform the evaluation with public datasets.

DeeSIL (Section 3.2)

We use ILSVRC [135] dataset for our experiments. This dataset is described in details with datasets used to evaluate IL2M below.

IL2M (Section 3.3)

The datasets used in the evaluation are designed for three visual classification tasks: object, face and tourist landmark recognition. The main statistics of these datasets are provided in Table B.1.

— **ILSVRC** [135] is the well known subset of ImageNet used in the ILSVRC competitions and is reused here. The statistics from Table B.1 show that the training set is well balanced, with an average of 1231.2 images per class and a 70.2 standard deviation. The dataset is available for download from `http://image-net.org/download`.

— **VGGFACE2** [22] is a recent dataset focused on face recognition. It includes over 9000 unique identities. We selected the 1000 identities which have the largest number of associated images for the evaluation in order to have a dataset similar in size to ILSVRC. VGGFACE2 is well balanced and includes a mean of 491.7 images per class, with 49.4 standard deviation. The dataset includes loosely cropped face images and, following the usual face recognition pipeline, we extracted tighter crops before training and testing. Face detection was done using the publicly available MTCNN [187] framework. The

dataset is available for download from http://www.robots.ox.ac.uk/~vgg/data/vgg_face2/.

— **Google Landmarks** [108] (also called LANDMARKS) is a dataset built for tourist landmark recognition. It includes over 2 million images for over 30000 landmarks across the world. Again, we selected the 1000 landmarks which have the largest number of associated images for the evaluation. The selected train subset is more imbalanced than ILSVRC and VGGFACE2, with a mean number of 374.4 images per class and 103.8 standard deviation. The dataset is available for download from https://www.kaggle.com/google/google-landmarks-dataset

Dataset	#Train	#Test	#Classes	μ(train)	σ(train)
ILSVRC [135]	1,231,167	50,000	1,000	1231.2	70.2
VGGFACE2 [22]	491,746	50,000	1,000	491.7	49.4
LANDMARKS [108]	374,367	20,000	1,000	374.4	103.8

Table B.1 – Summary of the datasets used in $IL2M$ evaluation. μ is the mean number of train images per class and σ is the standard deviation of the number of train images per class

ScaIL (Section 3.4)

We use the same datasets used to evaluate $IL2M$, and we add CIFAR-100 (Table B.2):

— **CIFAR-100** [74]: object recognition dataset including 100 classes, with 500 training and 100 test images each.

Dataset	#Train	#Test	#Classes	μ(train)	σ(train)
CIFAR-100 [74]	50,000	10,000	100	500.00	0.00

Table B.2 – Main statistics for the CIFAR-100 evaluation dataset. Other datasets were described previously in Table B.1. μ is the mean number of train images per class and σ is the standard deviation of the number of train images per class

Class-incremental methods study (Section 3.5)

We use the same datasets used to evaluate $IL2M$ and $ScaIL$ described above.

Active class-incremental learning (Section 3.6)

— **IMAGENET-100** - dataset for fine grained object recognition consisting of a subset of 100 randomly selected leaf classes from ImageNet [32] which have at least 50 training images and are not present in the ILSVRC subset [135].

— **FACES-100** - face recognition dataset consisting of a subset of randomly selected 100 identities from VGG-Face2 [22] with at least 30 training images.

— **FOOD-100** - dataset built using 100 classes from Food-101 [20] for fine-grained food recognition [20]. Since the initial dataset is perfectly balanced, an imbalance induction procedure was applied by removing a variable number of training samples keeping at least 25 images per class.

— **CIFAR-100** - dataset for object recognition used in its original version [74] which is perfectly balanced.

The main statistics of experimental datasets are in Table B.3. We provide the coefficient of variation $cv = \frac{\sigma}{\mu}$, with σ the standard deviation and μ the mean of the distribution of samples per class. cv provides information about the degree of imbalance associated to each dataset. The larger this value is, the more imbalanced the dataset will be.

Dataset	Classes	Train	Test	μ(train)	σ(train)	cv
IMAGENET-100	100	50000	5K	500.0	376.17	0.7523
FACES-100	100	23237	5K	232.37	167.68	0.7216
FOOD-100	100	22374	10K	223.74	177.66	0.7940
CIFAR-100	100	50000	10K	500.0	0.0	0.0

Table B.3 – Dataset statistics. cv is the coefficient of variation defined as $cv = \frac{\sigma}{\mu}$. μ is the mean number of train images per class and σ is the standard deviation of the number of train images per class

SIW (Section 4.2)

We use the same datasets used to evaluate $IL2M$ and $ScaIL$ described above.

TransIL (Section 4.3)

Reference datasets. We use 10 reference datasets, each including 100 randomly chosen leaf classes from ImageNet [32], with a 500/200 train/val split per class. There is no intersection between these datasets, as each class appears only in one of them.

Target datasets. We test $TransIL$ with four target datasets. They were selected to include different types of visual content and thus test the robustness of the parameter transfer. The class samples from the target datasets are split into 500/100 train/test subsets respectively. There is no intersection between the classes from the reference datasets and the two target datasets, which are sampled from ImageNet. We describe target datasets hereafter:

— **CIFAR-100** [74] - object recognition dataset. It focuses on commonsense classes and is relevant for basic level classification in the sense of [131].

— **IMN-100** - subset of ImageNet [32] which includes 100 randomly selected leaf classes. It is built with the same procedure used for reference datasets and is thus most similar to them. IMN-100 is relevant for fine-grained classification with a diversity of classes.

— **BIRDS-100** - dataset built using 100 bird classes from ImageNet [32]. It is thus relevant for domain-specific fine-grained classification.

— **FOOD-100** - dataset built using 100 food classes from Food-101 [20]. It is also a fine-grained and domain-specific dataset. It is interesting insofar as it is independent from ImageNet.

— **PLACES-100** - Scene recognition dataset that contains 100 randomly chosen classes from the original Places-365 dataset [192].

We provide in Table B.4 the lists of classes contained in each of the target datasets we used for evaluation. Overall, IMN-100, the randomly sampled set of 100 leaf classes from Imagenet [32], is more diversified than CIFAR-100, which mostly contains animal classes. IMN-100 is visually varied between different types of objects, foods, animals, vehicles, clothes and events. CIFAR-100 contains, in addition to animals, some types of objects and vehicles. Note that even if IMN-100 and CIFAR-100 are both specialized in object recognition, they do not share the same data distribution. The BIRDS-100 dataset (extracted from ImageNet [32]) is finer-grained and more specialized than IMN-100 and CIFAR-100. PLACES-100 and FOOD-100 are target datasets which have a larger domain shift with ImageNet classes, and are thus useful to test the robustness of our method against domain variation.

Similarly to IMN-100, reference datasets are random subsets of ImageNet leaves. They contain various object types and are useful for knowledge transfer.

	Classes names
CIFAR-100	Apple, Aquarium fish, Baby, Bear, Beaver, Bed, Bee, Beetle, Bicycle, Bottle, Bowl, Boy, Bridge, Bus, Butterfly, Camel, Can, Castle, Caterpillar, Cattle, Chair, Chimpanzee, Clock, Cloud, Cockroach, Couch, Crab, Crocodile, Cup, Dinosaur, Dolphin, Elephant, Flatfish, Forest, Fox, Girl, Hamster, House, Kangaroo, Keyboard, Lamp, Lawn mower, Leopard, Lion, Lizard, Lobster, Man, Maple tree, Motorcycle, Mountain, Mouse, Mushroom, Oak tree, Orange, Orchid, Otter, Palm tree, Pear, Pickup truck, Pine tree, Plain, Plate, Poppy, Porcupine, Possum, Rabbit, Raccoon, Ray, Road, Rocket, Rose, Sea, Seal, Shark, Shrew, Skunk, Skyscraper, Snail, Snake, Spider, Squirrel, Streetcar, Sunflower, Sweet pepper, Table, Tank, Telephone, Television, Tiger, Tractor, Train, Trout, Tulip, Turtle, Wardrobe, Whale, Willow tree, Wolf, Woman, Worm
IMN-100	Bletilla striata, Christmas stocking, Cognac, European sandpiper, European turkey oak, Friesian, Japanese deer, Luger, Sitka spruce, Tennessee walker, Torrey pine, Baguet, Bald eagle, Barn owl, Bass guitar, Bathrobe, Batting helmet, Bee eater, Blue gum, Blue whale, Bones, Borage, Brass, Caftan, Candytuft, Carthorse, Cattle egret, Cayenne, Chairlift, Chicory, Cliff dwelling, Cocktail dress, Commuter, Concert grand, Crazy quilt, Delivery truck, Detached house, Dispensary, Drawing room, Dress hat, Drone, Frigate bird, Frozen custard, Gemsbok, Giant kangaroo, Guava, Hamburger bun, Hawfinch, Hill myna, Howler monkey, Huisache, Jennet, Jodhpurs, Ladder truck, Loaner, Micrometer, Mink, Moorhen, Moorhen, Moped, Mortarboard, Mosquito net, Mountain zebra, Muffler, Musk ox, Obelisk, Opera, Ostrich, Ox, Oximeter, Playpen, Post oak, Purple-fringed orchid, Purple willow, Quaking aspen, Ragged robin, Raven, Redpoll, Repository, Roll-on, Scatter rug, Shopping cart, Shower curtain, Slip-on, Spider orchid, Sports car, Steam iron, Stole, Stuffed mushroom, Subcompact, Sundial, Tabby, Tabi, Tank car, Tramway, Unicycle, Wagtail, Walker, Window frame, Wood anemone
BIRDS-100	American bittern, American coot, Atlantic puffin, Baltimore oriole, Barrow's goldeneye, Bewick's swan, Bullock's oriole, California quail, Eurasian kingfisher, European gallinule, European sandpiper, Orpington, Amazon, Barn owl, Black-crowned night heron, Black-necked stilt, Black-winged stilt, Black swan, Black vulture, Black vulture, Blue peafowl, Brambling, Bufflehead, Buzzard, Cassowary, Cockerel, Common spoonbill, Crossbill, Duckling, Eastern kingbird, Emperor penguin, Fairy bluebird, Fishing eagle, Fulmar, Gamecock, Golden pheasant, Goosander, Goshawk, Great crested grebe, Great horned owl, Great white heron, Greater yellowlegs, Greenshank, Gyrfalcon, Hawfinch, Hedge sparrow, Hen, Honey buzzard, Hornbill, Kestrel, Kookaburra, Lapwing, Least sandpiper, Little blue heron, Little egret, Macaw, Mandarin duck, Marsh hawk, Moorhen, Mourning dove, Muscovy duck, Mute swan, Ostrich, Owlet, Oystercatcher, Pochard, Raven, Red-legged partridge, Red-winged blackbird, Robin, Robin, Rock hopper, Roseate spoonbill, Ruby-crowned kinglet, Ruffed grouse, Sanderling, Screech owl, Screech owl, Sedge warbler, Shoveler, Siskin, Snow goose, Snowy egret, Song thrush, Spotted flycatcher, Spotted owl, Sulphur-crested cockatoo, Thrush nightingale, Tropic bird, Tufted puffin, Turkey cock, Weka, Whistling swan, White-breasted nuthatch, White-crowned sparrow, White-throated sparrow, White stork, Whole snipe, Wood ibis, Wood pigeon.

FOOD-100	Apple pie, Baby back ribs, Baklava, Beef carpaccio, Beef tartare, Beet salad, Beignets, Bibimbap, Bread pudding, Breakfast burrito, Bruschetta, Caesar salad, Cannoli, Caprese salad, Carrot cake, Ceviche, Cheese plate, Cheesecake, Chicken curry, Chicken quesadilla, Chicken wings, Chocolate cake, Chocolate mousse, Churros, Clam chowder, Club sandwich, Crab cakes, Creme brulee, Croque madame, Cup cakes, Deviled eggs, Donuts, Dumplings, Edamame, Eggs benedict, Escargots, Falafel, Filet mignon, Fish and chips, Foie gras, French fries, French onion soup, French toast, Fried calamari, Fried rice, Frozen yogurt, Garlic bread, Gnocchi, Greek salad, Grilled cheese sandwich, Grilled salmon, Guacamole, Gyoza, Hamburger, Hot and sour soup, Hot dog, Huevos rancheros, Hummus, Ice cream, Lasagna, Lobster bisque, Lobster roll sandwich, Macaroni and cheese, Macarons, Miso soup, Mussels, Nachos, Omelette, Onion rings, Oysters, Pad thai, Paella, Pancakes, Panna cotta, Peking duck, Pho, Pizza, Pork chop, Poutine, Prime rib, Pulled pork sandwich, Ramen, Ravioli, Red velvet cake, Risotto, Samosa, Sashimi, Scallops, Seaweed salad, Shrimp and grits, Spaghetti bolognese, Spaghetti carbonara, Spring rolls, Steak, Strawberry shortcake, Sushi, Tacos, Takoyaki, Tiramisu, Tuna tartare
PLACES-100	Airplane cabin, Amphitheater, Amusement arcade, Aqueduct, Arcade, Archaelogical excavation, Archive, Arena performance, Attic, Bamboo forest, Bar, Barn, Baseball field, Bazaar outdoor, Beach, Beach house, Beauty salon, Bedroom, Bookstore, Bus interior, Cafeteria, Castle, Chemistry lab, Church outdoor, Cliff, Corridor, Courthouse, Crevasse, Department store, Desert sand, Desert vegetation, Dining room, Dorm room, Drugstore, Elevator lobby, Elevator shaft, Entrance hall, Escalator indoor, Farm, Field cultivated, Field wild, Florist shop indoor, Food court, Fountain, Garage indoor, Gazebo exterior, Golf course, Hangar outdoor, Harbor, Hardware store, Hayfield, Heliport, Highway, Home theater, Hospital room, Hot spring, Hotel outdoor, Hunting lodge outdoor, Ice skating rink indoor, Junkyard, Kasbah, Kitchen, Lagoon, Lake natural, Marsh, Mosque outdoor, Oast house, Office cubicles, Pagoda, Park, Pavilion, Physics laboratory, Pier, Porch, Racecourse, Residential neighborhood, Restaurant, Rice paddy, Rock arch, Ruin, Sauna, Server room, Shed, Shopfront, Storage room, Sushi bar, Television room, Television studio, Throne room, Topiary garden, Tower, Tree house, Trench, Underwater ocean deep, Waiting room, Water park, Waterfall, Wet bar, Windmill, Zen garden

Table B.4 – Classes names of target datasets listed by alphabetical order

B.2 Implementation details

All our method are implemented in Pytorch [118] using a ResNet-18 architecture [51] with an SGD optimizer. Training images are processed using randomly resized 224×224 crops, horizontal flipping, and are normalized afterward. Given the difference in scale and the number of images between CIFAR100 and the other datasets, we found that a different parametrization was needed for this dataset. Note that the parameters' values presented below are largely

inspired by the original ones given in [51].

— **Joint**: Used to train the model on the first batch of classes, and also on the whole datasets to establish the upper bound. For ILSVRC, VGGFACE2 and LANDMARKS, $Joint$ is run for 120 epochs with $batch\ size = 256$, $momentum = 0.9$ and $weight\ decay = 0.0001$. The lr is set to 0.1 and is divided by 10 when the error plateaus for 10 consecutive epochs. For CIFAR100, $Joint$ is run for 300 epochs with $batch\ size = 128$, $momentum = 0.9$ and $weight\ decay = 0.0005$. The lr is set to 0.1 and is divided by 10 when the error plateaus for 60 consecutive epochs.

— **FT**: fine-tuning backbone (without distillation) used in $IL2M$, $ScaIL$, SIW. For ILSVRC, VGGFACE2 and LANDMARKS, IL models are trained for 35 epochs with $batch\ size = 256$, $momentum = 0.9$ and $weight\ decay = 0.0001$. The learning rate is set to $lr = 0.1/t$ at the beginning of each incremental state $\mathcal{S}_t$ and is divided by 10 when the error plateaus for 5 consecutive epochs. For CIFAR-100, IL models are trained for 70 epochs with $batch\ size = 128$, $momentum = 0.9$ and $weight\ decay = 0.0005$. The learning rate is set to $lr = 0.1/t$ at the beginning of each incremental state $\mathcal{S}_t$ and is divided by 10 when the error plateaus for 15 consecutive epochs.

— **FR**: fixed representation. The model of the first is trained using $Joint$ above, and incrementally, it is trained exactly as FT.

— **REMIND** [48], **Deep-SLDA** [47], **BiC** [174] and **LUCIR** [53] are run using the optimal parameters of the public implementations provided in the original papers.

— **iCaRL** [128] - is run using the code from [53] since it provides better performance than the original implementation. **LwF** [86] is run using the code from [128] (the only baseline implemented in TensorFlow [2]).

— **DeeSIL (Section 3.2)** - The deep feature extractor is trained in the same manner as $Joint$ described above. The SVMs in $DeeSIL$ are implemented using LinearSVC solver from Scikit-Learn toolbox [119]. The SVMs were optimized with 20 images per class for validation. We vary values of the regularization parameter between 0.0001 and 1000 and the optimal parameter was then used in each variant of the system.

— **IL2M (Section 3.3), ScaIL (Section 3.4), and SIW (Section 4.2)** - these methods are based on a fine-tuning without distillation backbone (FT), described above. Since the three methods are applied during inference only, we perform an end-to-end training of the deep neural network. Later, we extract testing features and store them on the disk to

be able to run the score calibration on a CPU. The same implementation process goes for all methods that are implemented on top of fine tuning, such as FT^{th}, FT^{init}..., etc.

— **TransIL (Section 4.3)** - The correction of raw output scores is done in the same way for all methods. After the extraction of raw scores and corresponding labels for models learned in each incremental state, batches are fed into a PyTorch [118] module which performs the optimization of *adBiC* parameters, or the transfer of previously learned parameters depending on the setting. Following [101], *adBiC* layers are implemented as pairs of parameters and optimized simply through backpropagation. Parameters α_s^k, β_s^k corresponding to each incremental state s are optimized for 300 epochs, with the Adam [69] optimizer and a starting learning rate of 10^{-3}. An L2-penalty is added to the loss given in Equation 4.9, with a lambda of $5 \cdot 10^{-3}$ for α parameters and $5 \cdot 10^{-2}$ for β parameters.

To facilitate reproducibility, dataset details and the code of all tested approaches and their adaptations are publicly available at: https://github.com/EdenBelouadah/class-\incremental-learning/

APPENDIX C

RÉSUMÉ EN FRANÇAIS

C.1 Introduction

Dans notre monde complexe, de nombreux domaines technologiques évoluent à une vitesse fulgurante qui nécessite de suivre leur rythme de développement. Ce dernier s'est accompagné d'une augmentation significative de la quantité de données, notamment avec l'avènement du big data [1]. En termes simples, l'intelligence artificielle (IA) est devenue omniprésente, et son intégration dans les applications de la vie quotidienne va devenir encore plus prévalente à l'avenir [93].

Les réseaux de neurones sont les systèmes les plus populaires en intelligence artificielle. Leur développement est basé sur plusieurs spécialités multidisciplinaire comme l'informatique, les mathématiques, la science cognitive, les statistiques, la psychologie et les neurosciences. Les réseaux neuronaux apprennent à partir d'exemples, et plus ils rencontrent d'exemples pertinents, meilleure est leur expérience, et donc leur performance. Dans cette thèse, nous nous intéressons à la classification d'images avec des réseaux de neurones convolutifs profonds [80].

Les agents artificiels standards sont conçus pour être entraînés de manière statique. En revanche, les données du monde réel sont souvent dynamiques [6] et les agents artificiels devraient être capables d'ingérer de nouvelles informations sans avoir à être entraînés à partir du zéro. En d'autres termes, il ne suffit pas que les systèmes d'IA apprennent, mais il est important qu'ils apprennent en permanence. Cette façon d'apprendre est appelée apprentissage incrémental (IL). Ce dernier est intéressant car il réduit les besoins en calcul et l'empreinte mémoire. La limitation de la mémoire est souvent rencontrée en robotique [83], dans les systèmes embarqués et dans le domaine médical pour des contraintes de confidentialité.

L'apprentissage incrémental serait facile si le réapprentissage à partir du zéro est autorisé, mais devient très difficile dans le cas contraire. Les auteurs de [102] ont remarqué ce qu'ils appellent l'oubli catastrophique, un phénomène dont souffrent les systèmes enraînés par rétropropagation. L'oubli catastrophique (CF) est la tendance des réseaux neuronaux à oublier les connaissances antérieures lorsqu'ils tentent d'en intégrer de nouvelles.

D'autres défis liés à l'apprentissage incrémental concernent:

— **La complexité -** Seulement la couche de classification doit augmenter pour pouvoir classifier les nouvelles classes.

— **La mémoire -** Le système doit être capable de fonctionner sans mémoire du passé, ou avec une mémoire limitée.

— **La précision -** Le système doit être performant non seulement sur les nouvelles classes, mais aussi sur les anciennes.

— **La rapidité -** Le temps nécessaire pour inclure les nouvelles classes doit être limité.

— **Le compromis entre la plasticité et la stablité -** Le système doit s'adapter à la distribution des nouvelles données et au même temps garder autant d'information que possible du passé.

— **Le passage à l'échelle -** Le système doit être capable d'apprendre un nombre illimité de classes.

Finalement, il existe d'autres défis tels que: la non-annotation des données nouvelles, le déséquilibre des données au sein des nouvelles classes, la taille des batches de classes..., etc.

Dans cette thèse, nous nous concentrons sur les approches qui gardent la complexité du modèle fixe et qui fonctionnent avec et sans mémoire limitée du passé. Nous proposons une méthode qui favorise la stabilité, et quatre méthodes qui équilibrent la stabilité et la plasticité du réseau. L'objectif de nos contributions est de réduire l'écart en performance par rapport à un apprentissage standard où toutes les données de toutes les classes sont disponibles.

C.2 Formalisation du problème

Nous proposons une formalisation unifiée de l'apprentissage incrémental qui s'appuie sur celles introduites dans [86, 128, 24]. On note $\mathcal{T}$ le nombre total d'états. Le premier est appelé état initial, et les $\mathcal{T} - 1$ états restants sont incrémentaux. Un ensemble de P_t nouvelles classes est appris dans l'état $\mathcal{S}_t$. Les états ne se chevauchent pas dans les classes, c'est-à-dire qu'une classe n'est apprise initialement avec toutes ses données qu'une seule fois pendant tout le processus d'apprentissage. Un modèle $\mathcal{M}_1$ est initialement appris sur un ensemble d'apprentissage $\mathcal{X}_1 = \{X_1^1, X_1^2, ..., X_1^j, ..., X_1^{P_1}\}$, où P_1 est le nombre de classes apprises dans le premier état. $X_t^j = \{x_j^1, x_j^2, ..., x_j^{n_j}\}$ est l'ensemble des n_j exemples d'apprentissage de la classe j, p_t^j est sa probabilité de classification dans l'état $\mathcal{S}_t$. On note N_t l'ensemble de toutes les classes rencontrées jusqu'à l'état $\mathcal{S}_t$ inclus. Ainsi, $N_1 = P_1$ initialement, et $N_t = N_{t-1} + P_t =$

$P_1 + P_2 + ... + P_{t-1} + P_t$ pour les états suivants. Le modèle $\mathcal{M}_t$ est mis à jour avec un algorithme d'apprentissage incrémental $\mathcal{A}$ en utilisant $\mathcal{X}_t = \{(X_t^j, Y_t^j) : j \in [N_{t-1}, N_t]\}$ si aucune mémoire du passé n'est autorisée et en utilisant $\mathcal{X}_t \cup \mathcal{K}$ si une mémoire du passé $\mathcal{K}$ est autorisée. À chaque état, $\mathcal{M}_t$ est évalué sur toutes les classes vues jusqu'ici ($j \in [1, N_t]$). Nous marquons en gras les vecteurs et les matrices.

Suivant [128, 24], nous utilisons dans cette thèse $P_1 = P_2 = ... = P_{\mathcal{T}}$. Certains travaux récents [88, 53, 35, 89] entraînent le premier modèle $\mathcal{M}_1$ sur la moitié des classes disponibles, et divisent les classes restantes de manière égale entre les états $\mathcal{T} - 1$ suivants. Ce dernier est un cas plus facile de l'apprentissage incrémental.

Les algorithmes récents d'apprentissage incrémental sont mis en œuvre en utilisant des réseaux de neurones profonds (DNNs) [128, 24]. Alors que les DNNs sont des approches de classification de bout en bout, une partie des algorithmes d'apprentissage incrémental utilise une couche de classification séparée. Dans ce cas, le modèle $\mathcal{M}_t$ comprend deux composants principaux: un extracteur de caractéristiques $\mathcal{F}_t$ et un classificateur $\mathcal{C}_t$. On note $\theta_t = \{\phi_t, \psi_t\}$ l'ensemble de tous les paramètres du réseau dans l'état $\mathcal{S}_t$.

$\mathcal{F}_t$ est paramétré par des poids ϕ_t et est défini par l'Equation 2.1. Le classificateur $\mathcal{C}_t$ est paramétré par des poids ψ_t, et est généralement défini par l'Equation 2.2. Il peut également être implémenté en utilisant un classificateur externe comme dans [12] et [128].

C.3 État de l'art

Les approches de l'état de l'art se divisent en trois catégories principales : (1) les approches basées sur une représentation fixe, dans lesquelles le modèle n'évolue pas dans les états incrémentaux, (2) les approches basées sur le fine tuning, dans lesquelles le modèle est mis à jour à l'aide des données des nouvelles classes et éventuellement d'une mémoire des classes précédentes, et (3) les approches basées sur l'isolation des paramètres, qui compressent le réseau ou augmentent son architecture afin d'accueillir de nouvelles classes.

— **Méthodes basées sur une représentation fixe** [48, 66, 47] - Les méthodes de représentation fixe héritent des avantages et des inconvénients des schémas d'apprentissage par transfert. La complexité des modèles est constante, et elles peuvent être mis à jour rapidement puisque seule la couche de classification est réentraînée. Elles sont peu dépendantes de la mémoire du passé et peuvent s'adapter à un grand nombre de classes. Cependant, ces algorithmes dépendent fortement de la qualité de la représentation initiale et ont une faible plasticité. Ils sont utilisables lorsque: la complexité du modèle est constante, la

variabilité des données est faible, aucun stockage n'est possible pour les classes passées et les mises à jour sont rapides.

— **Méthodes basées sur le fine tuning** [24, 128, 53, 174] - Ces méthodes sont adéquates lorsque nous essayons d'optimiser la complexité de l'architecture, et la plasticité [161] des représentations. Cependant, comme elles nécessitent un réentraînement du réseau lorsque de nouvelles classes sont ajoutées, leur vitesse d'exécution n'est pas optimale. Tout aussi important, la contrainte de mémoire limitée les rend difficilement évolutives, car la mémoire finira par devenir trop petite pour représenter adéquatement les classes passées. Elles sont utilisables lorsque: la complexité du modèle est constante, les données varient beaucoup entre les états incrémentaux, le stockage des données passées est possible et l'utilisation immédiate des modèles mis à jour n'est pas essentielle.

— **Méthodes basées sur l'isolement des paramètres** [7, 100, 9] - Les approches de ce groupe s'accommodent bien des nouvelles données, ne dépendent pas ou peu de la mémoire du passé, mais ne sont évolutives que si elles sont dynamiques. Cependant, la complexité de ces dernières est un inconvénient puisque le modèle doit croître afin d'intégrer de nouvelles connaissances. Elles nécessitent également un réentraînement lorsque de nouvelles classes sont ajoutées et la vitesse d'exécution n'est pas optimale. Les réseaux dynamiques sont utilisables lorsque: la complexité du modèle peut croître au cours du processus incrémental, les données transmises varient beaucoup entre les états incrémentaux, aucun stockage n'est disponible pour les données passées et l'utilisation immédiate des modèles mis à jour n'est pas essentielle.

C.4 Apprentissage Incrémental avec mémoire

DeeSIL: Deep-Shallow Incremental Learning

Cette méthode est une adaptation d'un schéma d'apprentissage par transfert connu [44, 71, 125] à l'apprentissage incrémental. $DeeSIL$ (illustrée dans la Figure 3.2) comporte deux étapes faiblement corrélées. Tout d'abord, un modèle profond $\mathcal{M}_1$ est entraîné afin de fournir une représentation fixe qui est ensuite utilisée pour entraîner des classificateurs linéaires de type SVM (Machines à Vecteur Support [19]) pendant la phase incrémentale. Au lieu d'utiliser la mémoire du système $\mathcal{K}$ pour conserver les exemples positifs, un ensemble de caractéristiques négatives nécessaires à l'apprentissage incrémental des classificateurs est stocké. Ce choix permet d'utiliser tous les exemples positifs pour l'entraînement sans violer la contrainte de mémoire.

Notre hypothèse est que l'apprentissage des SVMs sur tous les positifs compense l'inconvénient lié à l'utilisation d'une représentation profonde fixe. Puisqu'aucun réentraînement n'est nécessaire pour augmenter la capacité du système, l'approche est considérablement moins complexe que ses homologues à apprentissage profond pur. L'ajout d'une nouvelle classe se fait par l'entraînement d'un classificateur linéaire, une opération qui prend moins d'une minute sur un seul CPU.

IL2M : Incremental Learning with Dual Memory

L'inconvénient principal de $DeeSIL$ est que ses performances dépendent fortement de la qualité de la représentation fixe. Si cette dernière est entraînée avec peu de données, ou avec des classes visuellement différentes de celles des classes apprises en incrémental, $DeeSIL$ ne parvient pas à transférer continuellement les connaissances aux états suivants. Une solution à ce problème consiste à affiner le modèle à chaque état incrémental afin de mettre à jour sa représentation avec les nouvelles données. Cependant, le déséquilibre entre les anciennes et les nouvelles classes s'aggrave avec le temps, un phénomène qui nécessite des techniques pour supprimer le biais du réseau vers les nouvelles classes.

Dans cette section, nous présentons une méthode qui vise à concilier partiellement les approches basées sur le fine tuning et la représentation fixe. $IL2M$ (illustrée dans la Figure 3.6) utilise le fine tuning pour mettre à jour les modèles profonds à chaque état incrémental. Cependant, à cause de la mise à jour profonde des paramètres lors du fine tuning, les modèles initiaux ne peuvent pas être entièrement réutilisés dans les états ultérieurs. Au lieu de cela, $IL2M$ exploite les statistiques des classes passées de leur état initial pour rectifier leurs scores de prédiction dans l'état incrémental actuel. Cette rectification est soutenue par deux hypothèses connexes : (1) les classes sont mieux modélisées lorsque toutes leurs données sont disponibles, et (2) les scores de prédiction des classes sont en moyenne plus élevés lorsque davantage des données d'entraînement sont disponibles. Nous illustrons la validité de ces hypothèses dans la Figure 3.5. Elle représente les prédictions moyennes des classes passées et nouvelles pour trois datasets de grande échelle avec $\mathcal{T} = 10$ et $|\mathcal{K}| = \{20000, 10000, 5000\}$.

Les scores de la Figure 3.5 confirment que le fine tuning génère un biais de prédiction en faveur des nouvelles classes. Ce biais est principalement dû au déséquilibre en faveur des nouvelles classes, qui apparaît dans l'apprentissage incrémental. Par conséquent, une grande partie des images des anciennes classes est prédite comme appartenant aux nouvelles classes.

Notre contribution principale est de proposer une mémoire secondaire qui stocke les statistiques des classes initiales dans un format très compact. Les statistiques initiales des classes

sont réutilisées dans chaque état incrémental ultérieur pour rectifier les scores de prédiction des classes passées. La rectification est nécessaire car les modèles d'apprentissage incrémental sont formés avec des ensembles de données déséquilibrés dans lesquels les classes passées ont moins d'exemples. Par conséquent, leurs scores de prédiction sont généralement plus faibles que ceux des nouvelles classes.

Pour compenser le biais vers les nouvelles classes, nous rectifions les prédictions des classes passées en utilisant l'Équation 3.1 du Chapitre 3.

ScaIL: Classifier Weights Scaling for Class-Incremental Learning

Nous étudions plus profondément l'effet de l'oubli catastrophique et découvrons que le biais est fortement présent dans la matrice des poids de la dernière couche dense. Nous présentons $ScaIL$, une méthode simple et efficace (illustrée dans la Figure 3.9) qui réduit le biais en faveur des nouvelles classes en exploitant les vecteurs de poids des classes passées tels qu'ils ont été appris dans leur état initial avec toutes les données des classes disponibles. Puisque les poids des classes passées sont appris dans différents états, ils sont normalisés pour être utilisables dans l'état actuel.

La réutilisation des poids initiaux dans les états incrémentaux ultérieurs est rendue possible par un processus de fine tuning avec une mémoire du passé. Ce processus aboutit à une préservation partielle de l'espace des caractéristiques même si le modèle profond évolue. $ScaIL$ réctifie les poids initiaux stockés dans $\mathcal{I}$ afin de les rendre comparables à ceux des classes nouvelles dans l'espace des caractéristiques défini par le modèle actuel. La rectification est basée sur les statistiques de poids calculées pour les nouvelles classes dans chaque état. Nous calculons un vecteur moyen par état et l'utilisons ultérieurement pour la normalisation. Dans l'Équation 3.2, nous trions le vecteur de poids de chaque nouvelle classe en fonction des valeurs absolues de ses éléments.

L'utilisation de valeurs absolues est nécessaire puisque les activations des poids de classification peuvent être positives ou négatives. Une fois que les poids des nouvelles classes sont triés, nous utilisons l'Équation 3.3 pour calculer un vecteur moyen par état que nous utilisons pour la normalisation.

Pour rendre comparables les prédictions des classes de différents états, il est nécessaire de calculer μ_t séparément pour chaque état. Notez que chaque moyenne est calculée en utilisant des poids situés au même rang pour chaque vecteur.

Dans l'état $\mathcal{S}_t$, $ScaIL$ transforme les poids des classes passées tels qu'ils ont été appris dans leur état initial en utilisant l'Équation 3.4.

Chaque poids w_j^d est modifié en utilisant les activations moyennes de son rang, renvoyées par la fonction $R(\cdot)$, dans les états actuel et initial $\mathcal{S}_t$ et $\mathcal{S}_i$. Ceci est fait afin de préserver l'importance relative de chaque poids. Le poids réctifié pour chaque classe passée j de l'état actuel $\mathcal{S}_t$ s'écrit comme $\boldsymbol{W_t^{j\prime}} = \{w_j^{1\prime}, w_j^{2\prime}, ..., w_j^{d\prime}, ..., w_j^{D\prime}\}$. Notez que la couche de classification $ScaIL$ pour $\mathcal{S}_t$ combine des poids rectifiés pour les classes passées et des poids originaux pour les nouvelles classes.

C.5 Apprentissage Incrémental sans mémoire

SIW: Standardization of Initial Weights for Incremental Learning

Nous nous appuyons sur $ScaIL$ et [193] pour proposer SIW, une méthode (illustrée dans la Figure 4.4) qui: exploite les informations des états initiaux des classes; assure l'équité des prédictions associées aux classes passées et nouvelles [53, 174]; utilise le FT pour entraîner des modèles profonds [5] . Cependant, notre approche diffère par la méthode utilisée pour la normalisation des poids, et met l'accent sur un apprentissage incrémental sans mémoire à grande échelle. Nous supposons qu'il est possible d'exploiter les poids initiaux des classes, afin d'atténuer l'effet de l'oubli catastrophique dans un apprentissage incrémental sans mémoire. L'effet de l'oubli catastrophique peut être observé au niveau des magnitudes moyennes des poids des anciennes et nouvelles classes comme dans la Figure 4.2.

La Figure 4.2 montre que les classes passées ont des magnitudes beaucoup plus faibles car aucune mémoire n'est tolérée. Comme les magnitudes sont beaucoup plus élevées pour les nouvelles classes, les exemples de test seront toujours attribués à l'une de ces classes, même s'ils appartiennent aux classes antérieures. La figure montre aussi que l'ampleur des nouvelles classes varie selon les états incrémentaux, avec une tendance globale à la réduction dans les états ultérieurs. Une normalisation des poids initiaux est donc nécessaire pour assurer l'équité s'ils sont réutilisés à travers les états incrémentaux comme proposé ici. La Figure 4.2 montre que la normalisation des poids initiaux rend les populations statistiques plus comparables. Une deuxième hypothèse importante de notre approche est que les caractéristiques extraites de l'avant-dernière couche du modèle actuel sont compatibles avec les poids initiaux des états précédents. Cette hypothèse est valable si les caractéristiques actuelles gardent une trace de ce qui a été appris auparavant.

Nous réutilisons les poids initiaux des classes afin d'atténuer l'oubli catastrophique en appliquant l'Equation 4.1. Ensuite, nous normalisons les poids initiaux en utilisant l'Equation

4.2. La normalisation est utile si elle est appliquée aux populations statistiques qui suivent une distribution normale [38], ce qui est le cas pour les poids de l'Équation 4.1.

TransIL: Dataset Knowledge Transfer for Class-Incremental Learning

Notre principale contribution est de permettre l'utilisation des méthodes de correction de biais, comme la couche BiC de [174], dans un protocol sans mémoire. Nous nous concentrons sur cette approche car elle est à la fois simple et efficace [15, 101]. Les auteurs de BiC [174] utilisent un ensemble de validation qui stocke des échantillons de classes passées pour optimiser les paramètres. Au lieu de cela, nous apprenons les paramètres de correction hors ligne sur un ensemble de datasets de référence, puis nous les transférons aux datasets cibles. L'intuition est que, si les ensembles de données sont différents, les paramètres de correction de biais optimaux sont suffisamment stables pour être transférables entre eux. Nous illustrons l'approche dans la Figure 4.7, la partie supérieure montre le processus d'apprentissage incrémental avec un dataset de référence. Une mémoire pour les données de validation nécessaires à l'optimisation de la couche de correction de biais est prévue puisque l'apprentissage se fait hors ligne. La partie inférieure de la figure présente l'apprentissage incrémental d'un dataset cible. La principale différence avec l'apprentissage incrémental sans mémoire standard provient de l'utilisation d'une couche de correction de biais optimisée sur le dataset de référence. Son introduction conduit à une meilleure comparabilité des scores de prédiction pour les classes passées et nouvelles. Notez que la méthode proposée est applicable à n'importe quelle méthode d'apprentissage incrémental puisqu'elle ne nécessite que la disponibilité des prédictions brutes fournies par les modèles profonds $\mathcal{M}$.

La deuxième contribution consiste à affiner la définition de la couche de correction de biais introduite dans [174]. La formulation originale traite toutes les classes passées de manière égale dans le processus de correction. Comme [101], nous faisons l'hypothèse que le degré d'oubli associé aux classes passées dépend de l'état initial dans lequel elles ont été apprises. Par conséquent, nous proposons *Adaptive BiC* ($AdBiC$), une procédure d'optimisation qui apprend une paire de paramètres par état au lieu d'une seule paire de paramètres comme proposé dans [174].

Contrairement à BiC [174], l'Équation 4.8 ajuste les scores de prédiction en fonction de l'état dans lequel les classes ont été rencontrées pour la première fois. Notez que chaque paire α_t^k, β_t^k est partagée entre toutes les classes apprises pour la première fois dans le même état. Ces paramètres sont optimisés sur un ensemble de validation en utilisant l'erreur d'entropie croisée.

L'optimisation des paramètres α et β est impossible dans un protocol sans mémoire, puisque les exemplaires des classes passées ne sont pas disponibles. Pour contourner ce problème,

nous faisons l'hypothèse que les valeurs optimales de ces paramètres peuvent être transférées entre les datasets de référence et cibles, notés $\mathcal{X}^r$ et $\mathcal{X}$ respectivement. L'intuition est que ces valeurs sont suffisamment stables malgré la variabilité du contenu des ensembles de données. Le stockage de l'ensemble de validation est nécessaire afin d'optimiser les paramètres de l'Équation 4.8 et est possible puisque l'entraînement des datasets de référence est effectué hors ligne. Notez que les modèles incrémentaux de $\mathcal{X}^r$ sont entraînés sans mémoire afin de simuler l'apprentissage des datasets cibles $\mathcal{X}$. Nous stockons ensuite les paramètres de correction de biais optimisés pour les datasets de référence afin d'effectuer le transfert vers les datasets cibles sans utiliser de mémoire. Pour chaque état incrémental, nous calculons la moyenne des valeurs α et β sur tous les datasets de référence. Les moyennes obtenues sont utilisées pour la rectification du score sur les datasets cibles. La mémoire nécessaire au stockage des paramètres transférés est négligeable puisque nous avons besoin de $2\times(2+3+...+\mathcal{T}) = (\mathcal{T}+2)\times(\mathcal{T}-1)$ valeurs flottantes pour chaque dataset et chaque valeur de $\mathcal{T}$. Pour les états $\mathcal{T} = \{5, 10, 20\}$, nous ne stockons donc respectivement que 28, 108 et 418 valeurs flottantes.

C.6 Conclusions

— **Apprentissage incrémental avec mémoire -** Les principales conclusions sont les suivantes : (1) la mémoire bornée peut être une bonne alternative à l'erreur de distillation largement utilisée sur des datasets à grande échelle, et (2) l'utilisation d'une mémoire bornée seule n'est pas suffisante car le biais du réseau neuronal est important vers les nouvelles classes, notamment au niveau des scores de classification. D'autres techniques doivent être déployées pour supprimer ce biais.

— **Apprentissage incrémental sans mémoire -** Les principales conclusions sont les suivantes : (1) l'erreur de distillation est efficace lorsqu'aucune mémoire du passé n'est autorisée (ce n'est pas le cas à grande échelle lorsqu'une mémoire est autorisée), (2) l'utilisation des poids initiaux des classes passées à partir de leurs états initiaux est bénéfique pour lutter contre l'oubli catastrophique. La précision des systèmes actuels d'apprentissage incrémental sans mémoire est encore faible, car l'écart avec un apprentissage à partir de zéro (avec toutes les données disponibles) est important. Des efforts supplémentaires sont nécessaires pour améliorer les systèmes actuels dans des scénarios réels.

C.7 Perspectives

Les performances des méthodes d'apprentissage incrémental se rapprochent de celles de l'apprentissage standard, où l'on suppose que toutes les données de toutes les classes sont disponibles à tout moment. Cependant, le problème de l'oubli catastrophique est loin d'être résolu, surtout à grande échelle. La plupart des systèmes existants [128, 24, 53, 52, 61] utilisent une mémoire du passé. Notez que, dans les scénarios de la vie réelle, une mémoire n'est pas toujours autorisée. En outre, les algorithmes dépendant de la mémoire ont tendance à ne pas bien évoluer lorsque de plus en plus de classes n'auront pas de représentants pour elles lorsque la taille maximale de la mémoire est atteinte.

L'apprentissage incrémental sans mémoire est peu étudié dans la littérature et nous pensons que davantage d'efforts devraient y être consacrés. A notre connaissance, les seules méthodes de ce type disponibles sont celles que nous avons proposées en complément de: *Learning without Memorizing* [34], *Learning without Forgetting* (LwF) [86], $Deep\text{-}SLDA$ [47], *Semantic Drift Compensation* (SDC) [182], et *Always Be Dreaming* (AbD) [156]. Nous pourrions adapter d'autres méthodes à un cadre sans mémoire ($REMIND$ [48] et $LUCIR$ [53]), mais ce nombre de systèmes est négligeable par rapport à ceux qui reposent sur une mémoire du passé.

Traiter l'apprentissage incrémental comme un problème d'apprentissage déséquilibré donne des résultats très intéressants avec ou sans l'utilisation d'un composant de distillation. Il serait intéressant d'étudier des schémas de réduction de biais plus sophistiqués pour améliorer encore les performances. De même, une étude plus approfondie des raisons pour lesquelles la distillation ne fonctionne pas pour les ensembles de données à grande échelle est nécessaire. Les résultats empiriques rapportés dans cette thèse devraient être complétés par une analyse plus théorique pour améliorer la compréhension de ce phénomène.

Enfin, il convient de rendre le scénario d'évaluation plus réaliste en:

— abandonnant l'hypothèse que toutes les nouvelles données sont annotées

— utilisant un nombre variable de classes dans chaque état incrémental. Ce cas est plus réaliste car les données peuvent arriver de manière aléatoire.

— travaillant avec des jeux de données déséquilibrés qui sont plus susceptibles de se produire dans des applications réelles.

— manipulant des images individuelles au lieu de classes, où les images de passé peuvent être retrouvée dans les états futurs [50].

Les points mentionnés ci-dessus sont intéressants et devraient être abordés pour une utilisation transparente de l'apprentissage incrémental dans la vie pratique comme dans [105].

BIBLIOGRAPHY

[1] *25+ Impressive Big Data Statistics for 2021*, https://techjury.net/blog/big-data-statistics/#gref, Accessed: 2021-08-11.

[2] Martin Abadi et al., « TensorFlow: A system for large-scale machine learning », *in*: *CoRR* abs/1605.08695 (2016).

[3] Umang Aggarwal, Adrian Popescu, and Celine Hudelot, « Active Learning for Imbalanced Datasets », *in*: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020.

[4] Umang Aggarwal et al., « A comparative study of calibration methods for imbalanced class incremental learning », *in*: *Multimedia Tools and Applications* (2021), pp. 1–20.

[5] Hongjoon Ahn and Taesup Moon, *A Simple Class Decision Balancing for Incremental Learning*, 2020, URL: https://arxiv.org/abs/2003.13947.

[6] Luca Maria Aiello et al., « Sensing Trending Topics in Twitter », *in*: *IEEE Trans. Multim.* 15.6 (2013), pp. 1268–1282.

[7] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars, « Expert Gate: Lifelong Learning with a Network of Experts », *in*: *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[8] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars, « Task-Free Continual Learning », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 11254–11263.

[9] Rahaf Aljundi et al., « Memory Aware Synapses: Learning What (not) to Forget », *in*: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, ed. by Vittorio Ferrari et al., vol. 11207, Lecture Notes in Computer Science, Springer, 2018, pp. 144–161.

[10] ET Barron and Robert M Glorioso, « A micro controlled peripheral processor », *in*: *Conference record of the 6th annual workshop on Microprogramming*, 1973, pp. 122–128.

[11] Felipe Duque Belfort, Hansenclever de F. Bassani, and Aluizio F. R. Araujo, « Online incremental supervised growing neural gas », *in*: *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, IEEE, 2017, pp. 1034–1040.

[12] Eden Belouadah and Adrian Popescu, « DeeSIL: Deep-Shallow Incremental Learning », *in*: *TaskCV Workshop @ ECCV 2018.* (2018).

[13] Eden Belouadah and Adrian Popescu, « Il2m: Class incremental learning with dual memory », *in*: *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 583–592.

[14] Eden Belouadah and Adrian Popescu, « ScaIL: Classifier Weights Scaling for Class Incremental Learning », *in*: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020.

[15] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos, « A comprehensive study of class incremental learning algorithms for visual tasks », *in*: *Neural Networks* 135 (2021), pp. 38–54.

[16] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos, « Initial Classifier Weights Replay for Memoryless Class Incremental Learning », *in*: *British Machine Vision Conference (BMVC)*, 2020.

[17] Eden Belouadah et al., « Active Class Incremental Learning for Imbalanced Datasets », *in*: *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, ed. by Adrien Bartoli and Andrea Fusiello, vol. 12540, Lecture Notes in Computer Science, Springer, 2020, pp. 146–162.

[18] William H. Beluch et al., « The Power of Ensembles for Active Learning in Image Classification », *in*: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 9368–9377.

[19] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik, « A Training Algorithm for Optimal Margin Classifiers », *in*: *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, ed. by David Haussler, ACM, 1992, pp. 144–152.

[20] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool, « Food-101 – Mining Discriminative Components with Random Forests », *in*: *European Conference on Computer Vision*, 2014.

[21] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski, « A systematic study of the class imbalance problem in convolutional neural networks », *in*: *Neural Networks* 106 (2018), pp. 249–259.

[22] Qiong Cao et al., « VGGFace2: A Dataset for Recognising Faces across Pose and Age », *in*: *13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018, Xian, China, May 15-19, 2018*, 2018, pp. 67–74.

[23] Gustavo Carneiro and Jacinto C. Nascimento, « Incremental on-line semi-supervised learning for segmenting the left ventricle of the heart from ultrasound data », *in*: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, ed. by Dimitris N. Metaxas et al., IEEE Computer Society, 2011, pp. 1700–1707.

[24] Francisco M. Castro et al., « End-to-End Incremental Learning », *in*: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, 2018, pp. 241–257.

[25] Arslan Chaudhry et al., « Efficient Lifelong Learning with A-GEM », *in*: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[26] Arslan Chaudhry et al., « Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence », *in*: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, ed. by Vittorio Ferrari et al., vol. 11215, Lecture Notes in Computer Science, Springer, 2018, pp. 556–572.

[27] Xinlei Chen and Abhinav Gupta, « Webly Supervised Learning of Convolutional Networks », *in*: *International Conference on Computer Vision*, ICCV, 2015.

[28] François Chollet, « Xception: Deep Learning with Depthwise Separable Convolutions », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 1800–1807.

[29] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber, « Multi-column deep neural networks for image classification », *in*: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, IEEE Computer Society, 2012, pp. 3642–3649.

[30] Yulai Cong et al., « GAN Memory with No Forgetting », *in*: *CoRR* abs/2006.07543 (2020).

[31] Aron Culotta and Andrew McCallum, « Reducing Labeling Effort for Structured Prediction Tasks », *in*: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 2005, pp. 746–751.

[32] Jia Deng et al., « ImageNet: A large-scale hierarchical image database », *in*: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 2009, pp. 248–255.

[33] Thomas Deselaers et al., « Jointly optimising relevance and diversity in image retrieval », *in*: *ACM CIVR*, 2009.

[34] Prithviraj Dhar et al., « Learning without Memorizing », *in*: *CoRR* abs/1811.08051 (2018).

[35] Arthur Douillard et al., « PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning », *in*: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*, ed. by Andrea Vedaldi et al., vol. 12365, Lecture Notes in Computer Science, Springer, 2020, pp. 86–102.

[36] Mathias Eitz, James Hays, and Marc Alexa, « How do humans sketch objects? », *in*: *ACM Transactions on graphics (TOG)* 31.*4* (2012), pp. 1–10.

[37] M Everingham et al., « The pascal visual object classes challenge 2012 (voc2012) results (2012) », *in*: *URL http://www. pascal-network. org/challenges/VOC/voc2011/workshop/index. html*, 2011.

[38] David Freedman, Robert Pisani, and Roger Purves, *Statistics: Fourth International Student Edition*, W.W. Norton & Company, 2007.

[39] Bernd Fritzke, « A Growing Neural Gas Network Learns Topologies », *in*: *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, ed. by Gerald Tesauro, David S. Touretzky, and Todd K. Leen, MIT Press, 1994, pp. 625–632.

[40] Tommaso Furlanello et al., « Born-Again Neural Networks », *in*: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, ed. by Jennifer G. Dy and Andreas Krause, vol. 80, Proceedings of Machine Learning Research, PMLR, 2018, pp. 1602–1611.

[41] Yarin Gal, Riashat Islam, and Zoubin Ghahramani, « Deep Bayesian Active Learning with Image Data », *in*: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1183–1192.

[42] Alexander Gepperth and Cem Karaoguz, « Incremental learning with self-organizing maps », *in*: *12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization, WSOM 2017, Nancy, France, June 28-30, 2017*, ed. by Jean-Charles Lamirel, Marie Cottrell, and Madalina Olteanu, IEEE, 2017, pp. 153–160.

[43] Arnab Ghosh et al., « Multi-Agent Diverse Generative Adversarial Networks », *in*: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[44] Alexandru Lucian Ginsca et al., « Large-scale Image Mining with Flickr Groups », *in*: *Multimedia Modelling*, MM, 2015.

[45] Ian J. Goodfellow et al., « An Empirical Investigation of Catastrophic Forgeting in Gradient-Based Neural Networks », *in*: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, ed. by Yoshua Bengio and Yann LeCun, 2014.

[46] Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan, « Memory Efficient Experience Replay for Streaming Learning », *in*: *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, IEEE, 2019, pp. 9769–9776.

[47] Tyler L. Hayes and Christopher Kanan, « Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis », *in*: *CoRR* abs/1909.01520 (2019).

[48] Tyler L. Hayes et al., « REMIND Your Neural Network to Prevent Catastrophic Forgetting », *in*: *CoRR* abs/1910.02509 (2019).

[49] Chen He et al., « Exemplar-Supported Generative Reproduction for Class Incremental Learning », *in*: *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, 2018, p. 98.

[50] Jiangpeng He et al., « Incremental Learning in Online Scenario », *in*: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, IEEE, 2020, pp. 13923–13932.

[51] Kaiming He et al., « Deep Residual Learning for Image Recognition », *in*: *Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

[52] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean, « Distilling the Knowledge in a Neural Network », *in*: *CoRR* abs/1503.02531 (2015).

[53] Saihui Hou et al., « Learning a Unified Classifier Incrementally via Rebalancing », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 831–839.

[54] Saihui Hou et al., « Lifelong Learning via Progressive Distillation and Retrospection », *in*: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, ed. by Vittorio Ferrari et al., vol. 11207, Lecture Notes in Computer Science, Springer, 2018, pp. 452–467.

[55] Peiyun Hu et al., « Active Learning with Partial Feedback », *in*: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[56] Xinting Hu et al., « Distilling Causal Effect of Data in Class-Incremental Learning », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 3957–3966.

[57] Gao Huang et al., « Densely Connected Convolutional Networks », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 2261–2269.

[58] Yufan Huang et al., « Continual Learning for Text Classification with Information Disentanglement Based Regularization », *in*: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, ed. by Kristina Toutanova et al., Association for Computational Linguistics, 2021, pp. 2736–2746.

[59] *iNaturalist dataset: FGVC5 workshop at CVPR 2018." [Online]*, Available: https://www.kaggle.com/c/inaturalist-2018.

[60] Ahmet Iscen et al., « Memory-efficient incremental learning through feature adaptation », *in*: *European Conference on Computer Vision*, Springer, 2020, pp. 699–715.

[61] Khurram Javed and Faisal Shafait, « Revisiting Distillation and Incremental Classifier Learning », *in*: *CoRR* abs/1807.02802 (2018).

[62] Hervé Jégou, Matthijs Douze, and Cordelia Schmid, « Product Quantization for Nearest Neighbor Search », *in*: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.*1* (2011), pp. 117–128.

[63] MI Jordan, *Serial order: a parallel distributed processing approach. Technical report, June 1985-March 1986*, tech. rep., California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 1986.

[64] Heechul Jung et al., « Less-Forgetful Learning for Domain Expansion in Deep Neural Networks », *in*: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, ed. by Sheila A. McIlraith and Kilian Q. Weinberger, AAAI Press, 2018, pp. 3358–3365.

[65] Zixuan Ke, Hu Xu, and Bing Liu, « Adapting BERT for Continual Learning of a Sequence of Aspect Sentiment Classification Tasks », *in*: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, ed. by Kristina Toutanova et al., Association for Computational Linguistics, 2021, pp. 4746–4755.

[66] Ronald Kemker and Christopher Kanan, « FearNet: Brain-Inspired Model for Incremental Learning », *in*: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[67] Ronald Kemker et al., « Measuring Catastrophic Forgetting in Neural Networks », *in*: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, ed. by Sheila A. McIlraith and Kilian Q. Weinberger, AAAI Press, 2018, pp. 3390–3398.

[68] Dahyun Kim et al., « Incremental Learning with Maximum Entropy Regularization: Rethinking Forgetting and Intransigence », *in*: *CoRR* abs/1902.00829 (2019).

[69] Diederik P. Kingma and Jimmy Ba, « Adam: A Method for Stochastic Optimization », *in*: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, ed. by Yoshua Bengio and Yann LeCun, 2015.

[70] James Kirkpatrick et al., « Overcoming catastrophic forgetting in neural networks », *in*: *CoRR* abs/1612.00796 (2016).

[71] Simon Kornblith, Jonathon Shlens, and Quoc V. Le, « Do Better ImageNet Models Transfer Better? », *in*: *CoRR* abs/1805.08974 (2018).

[72] Ivan Krasin et al., « OpenImages: A public dataset for large-scale multi-label and multi-class image classification. », *in*: *Dataset available from https://storage.googleapis.com/openimages/web/in* (2017).

[73] Jonathan Krause et al., « 3d object representations for fine-grained categorization », *in*: *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 554–561.

[74] Alex Krizhevsky, *Learning multiple layers of features from tiny images*, tech. rep., University of Toronto, 2009.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks », *in*: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, ed. by Peter L. Bartlett et al., 2012, pp. 1106–1114.

[76] Matthias De Lange et al., « Continual learning: A comparative study on how to defy forgetting in classification tasks », *in*: *CoRR* abs/1909.08383 (2019).

[77] Alexis Lechat, Stéphane Herbin, and Frédéric Jurie, « Semi-Supervised Class Incremental Learning », *in*: *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*, IEEE, 2020, pp. 10383–10389.

[78] Yann LeCun et al., « Gradient-based learning applied to document recognition », *in*: *Proceedings of the IEEE* 86.*11* (1998), pp. 2278–2324.

[79] Yann LeCun et al., « Handwritten digit recognition with a back-propagation network », *in*: *Advances in neural information processing systems* 2 (1989).

[80] Yann LeCun et al., « Object recognition with gradient-based learning », *in*: *Shape, contour and grouping in computer vision*, Springer, 1999, pp. 319–345.

[81] Janghyeon Lee et al., « Continual Learning With Extended Kronecker-Factored Approximate Curvature », *in*: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, IEEE, 2020, pp. 8998–9007.

[82] Kibok Lee et al., « Incremental Learning with Unlabeled Data in the Wild », *in*: *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 29–32.

[83] Timothée Lesort et al., « Continual Learning for Robotics », *in*: *CoRR* abs/1907.00182 (2019).

[84] Ken Levine, « Core standard graphic package for the VGI 3400 », *in*: *ACM SIGGRAPH Computer Graphics* 12.*3* (1978), pp. 298–300.

[85] Yanchao Li et al., « Incremental semi-supervised learning on streaming data », *in*: *Pattern Recognit.* 88 (2019), pp. 383–396.

[86] Zhizhong Li and Derek Hoiem, « Learning Without Forgetting », *in*: *European Conference on Computer Vision*, ECCV, 2016.

[87] Xialei Liu et al., « Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting », *in*: *24th International Conference on Pattern Recognition, ICPR 2018, Beijing, China, August 20-24, 2018*, IEEE Computer Society, 2018, pp. 2262–2268.

[88] Yaoyao Liu, Bernt Schiele, and Qianru Sun, « Adaptive Aggregation Networks for Class-Incremental Learning », *in*: *arXiv preprint arXiv:2010.05063* (2020).

[89] Yaoyao Liu et al., « Mnemonics Training: Multi-Class Incremental Learning without Forgetting », *in*: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[90] Vincenzo Lomonaco and Davide Maltoni, « Core50: a new dataset and benchmark for continuous object recognition », *in*: *Conference on Robot Learning*, PMLR, 2017, pp. 17–26.

[91] Hui Ru Loo and Muhammad N. Marsono, « Online data stream classification with incremental semi-supervised learning », *in*: *Proceedings of the Second ACM IKDD Conference on Data Sciences, Bangalore, CoDS 2015, India, March 18-21, 2015*, ed. by Manish Gupta et al., ACM, 2015, pp. 132–133.

[92] David Lopez-Paz and Marc'Aurelio Ranzato, « Gradient Episodic Memory for Continual Learning », *in*: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, ed. by Isabelle Guyon et al., 2017, pp. 6467–6476.

[93] Stephen Lucci and Danny Kopec, *Artificial intelligence in the 21st century*, Stylus Publishing, LLC, 2015.

[94] Edwin Lughofer, « Extensions of vector quantization for incremental clustering », *in*: *Pattern Recognit.* 41.*3* (2008), pp. 995–1011.

[95] Subhransu Maji et al., « Fine-grained visual classification of aircraft », *in*: *arXiv preprint arXiv:1306.5151* (2013).

[96] Arun Mallya, Dillon Davis, and Svetlana Lazebnik, « Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights », *in*: *ECCV (4)*, vol. 11208, Lecture Notes in Computer Science, Springer, 2018, pp. 72–88.

[97] Arun Mallya and Svetlana Lazebnik, « PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning », *in*: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 7765–7773.

[98] Thomas Martinetz, « Competitive Hebbian learning rule forms perfectly topology preserving maps », *in*: *International conference on artificial neural networks*, Springer, 1993, pp. 427–434.

[99] Thomas Martinetz, Stanislav G. Berkovich, and Klaus Schulten, « 'Neural-gas' network for vector quantization and its application to time-series prediction », *in*: *IEEE Trans. Neural Networks* 4.*4* (1993), pp. 558–569.

[100] Marc Masana, Tinne Tuytelaars, and Joost van de Weijer, « Ternary Feature Masks: continual learning without any forgetting », *in*: *CoRR* abs/2001.08714 (2020), URL: https://arxiv.org/abs/2001.08714.

[101] Marc Masana et al., *Class-incremental learning: survey and performance evaluation on image classification*, 2021, arXiv: 2010.15277 [cs.LG].

[102] Michael Mccloskey and Neil J. Cohen, « Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem », *in*: *The Psychology of Learning and Motivation* 24 (1989), pp. 104–169.

[103] Thomas Mensink et al., « Distance-Based Image Classification: Generalizing to New Classes at Near-Zero Cost », *in*: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.*11* (2013), pp. 2624–2637.

[104] M Mermillod, A Bugaiska, and P Bonin, « The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. », *in*: *Frontiers in Psychology* 4 (2013), pp. 504–504.

[105] Fei Mi et al., « Generalized Class Incremental Learning », *in*: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, IEEE, 2020, pp. 970–974.

[106] Yuval Netzer et al., « Reading digits in natural images with unsupervised feature learning », *in*: (2011).

[107] Maria-Elena Nilsback and Andrew Zisserman, « Automated flower classification over a large number of classes », *in*: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, IEEE, 2008, pp. 722–729.

[108] Hyeonwoo Noh et al., « Large-Scale Image Retrieval with Attentive Deep Local Features », *in*: *ICCV*, IEEE Computer Society, 2017, pp. 3476–3485.

[109] Eirini Ntoutsi et al., « Bias in data-driven artificial intelligence systems - An introductory survey », *in*: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 10.*3* (2020).

[110] Kemal Oksuz et al., « Imbalance Problems in Object Detection: A Review », *in*: *CoRR* abs/1909.00169 (2019).

[111] Avital Oliver et al., « Realistic Evaluation of Deep Semi-Supervised Learning Algorithms », *in*: *Advances in Neural Information Processing Systems 31*, ed. by S. Bengio et al., Curran Associates, Inc., 2018, pp. 3235–3246.

[112] Oleksiy Ostapenko et al., « Learning to Remember: A Synaptic Plasticity Driven Framework for Continual Learning », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 11321–11329.

[113] Sankar K Pal and Sushmita Mitra, « Multilayer perceptron, fuzzy sets, classifiaction », *in*: (1992).

[114] Shaoning Pang, Seiichi Ozawa, and Nikola K. Kasabov, « Incremental linear discriminant analysis for classification of data streams », *in*: *IEEE Trans. Syst. Man Cybern. Part B* 35.*5* (2005), pp. 905–914.

[115] German Ignacio Parisi et al., « Continual Lifelong Learning with Neural Networks: A Review », *in*: *CoRR* abs/1802.07569 (2018).

[116] German Ignacio Parisi et al., « Continual Lifelong Learning with Neural Networks: A Review », *in*: *Neural Networks* 113 (2019).

[117] Wonpyo Park et al., « Relational Knowledge Distillation », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 3967–3976.

[118] Adam Paszke et al., « Automatic differentiation in PyTorch », *in*: *Advances in Neural Information Processing Systems Workshops*, NIPS-W, 2017.

[119] Fabian Pedregosa et al., « Scikit-learn: Machine Learning in Python », *in*: *CoRR* abs/1201.0490 (2012).

[120] Federico Pernici et al., « Memory Based Online Learning of Deep Representations From Video Streams », *in*: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[121] Mary Phuong and Christoph Lampert, « Towards Understanding Knowledge Distillation », *in*: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 5142–5151.

[122] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania, « GDumb: A simple approach that questions our progress in continual learning », *in*: *European Conference on Computer Vision*, Springer, 2020, pp. 524–540.

[123] Ariadna Quattoni and Antonio Torralba, « Recognizing indoor scenes », *in*: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 413–420.

[124] Roger Ratcliff, « Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. », *in*: *Psychological review* 97.2 (1990), p. 285.

[125] Ali Sharif Razavian et al., « CNN Features Off-the-Shelf: An Astounding Baseline for Recognition », *in*: *Conference on Computer Vision and Pattern Recognition Workshop*, CVPR-W, 2014.

[126] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi, « Efficient Parametrization of Multi-Domain Deep Neural Networks », *in*: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 8119–8127.

[127] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi, « Learning multiple visual domains with residual adapters », *in*: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 506–516.

[128] Sylvestre-Alvise Rebuffi et al., « iCaRL: Incremental Classifier and Representation Learning », *in*: *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[129] Joseph Redmon and Ali Farhadi, « YOLO9000: Better, Faster, Stronger », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 6517–6525.

[130] Ryne Roady et al., « Stream-51: Streaming classification and novelty detection from videos », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 228–229.

[131] Eleanor Rosch, « Principles of categorization », *in*: *Concepts: core readings* 189 (1999).

[132] Amir Rosenfeld and John K. Tsotsos, « Incremental Learning Through Deep Adaptation », *in*: *CoRR* abs/1705.04228 (2017).

[133] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy, « Tree-CNN: A hierarchical Deep Convolutional Neural Network for incremental learning », *in*: *Neural Networks* 121 (2020), pp. 148–160.

[134] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, *Learning internal representations by error propagation*, tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[135] Olga Russakovsky et al., « ImageNet Large Scale Visual Recognition Challenge », *in*: *International Journal of Computer Vision* 115.*3* (2015), pp. 211–252.

[136] Andrei A. Rusu et al., « Progressive Neural Networks », *in*: *CoRR* abs/1606.04671 (2016).

[137] Babak Saleh and Ahmed Elgammal, « Large-scale classification of fine-art paintings: Learning the right metric on the right feature », *in*: *arXiv preprint arXiv:1505.00855* (2015).

[138] Tobias Scheck, Ana Pérez Grassi, and Gangolf Hirtz, « A CNN-based Feature Space for Semi-supervised Incremental Learning in Assisted Living Applications », *in*: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2020, Volume 5: VISAPP, Valletta, Malta, February 27-29, 2020*, ed. by Giovanni Maria Farinella, Petia Radeva, and José Braz, SCITEPRESS, 2020, pp. 217–224.

[139] Tobias Scheffer, Christian Decomain, and Stefan Wrobel, « Mining the Web with Active Hidden Markov Models », *in*: *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, 2001, pp. 645–646.

[140] Jonathan Schwarz et al., « Progress & compress: A scalable framework for continual learning », *in*: *International Conference on Machine Learning*, PMLR, 2018, pp. 4528–4537.

[141] Nicu Sebe et al., *Machine Learning in Computer Vision*, vol. 29, Computational Imaging and Vision, Springer, 2005.

[142] Ari Seff et al., « Continual learning in generative adversarial nets », *in*: *arXiv preprint arXiv:1705.08395* (2017).

[143] Ramprasaath R. Selvaraju et al., « Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization », *in*: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE Computer Society, 2017, pp. 618–626.

[144] Ozan Sener and Silvio Savarese, « Active Learning for Convolutional Neural Networks: A Core-Set Approach », *in*: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[145] Ozan Sener and Silvio Savarese, « Active Learning for Convolutional Neural Networks: A Core-Set Approach », *in*: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[146] Burr Settles, *Active learning literature survey*, tech. rep., University of Winsconsin, 2010.

[147] Burr Settles, *Active learning literature survey*, tech. rep., University of Winsconsin, 2010.

[148] Claude E. Shannon, « A mathematical theory of communication », *in*: *ACM SIGMOBILE Mob. Comput. Commun. Rev.* 5.*1* (2001), pp. 3–55.

[149] Or Sharir, Barak Peleg, and Yoav Shoham, « The Cost of Training NLP Models: A Concise Overview », *in*: *CoRR* abs/2004.08900 (2020).

[150] Qi She et al., « OpenLORIS-Object: A robotic vision dataset and benchmark for lifelong deep learning », *in*: *2020 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2020, pp. 4767–4773.

[151] Yuan-Yuan Shen et al., « Online semi-supervised learning with learning vector quantization », *in*: *Neurocomputing* 399 (2020), pp. 467–478.

[152] Hanul Shin et al., « Continual Learning with Deep Generative Replay », *in*: *NIPS*, 2017, pp. 2994–3003.

[153] Karen Simonyan and Andrew Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition », *in*: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, ed. by Yoshua Bengio and Yann LeCun, 2015.

[154] Pravendra Singh et al., « Calibrating CNNs for Lifelong Learning », *in*: *Advances in Neural Information Processing Systems* 33 (2020).

[155] Habib Slim et al., « Dataset Knowledge Transfer for Class-Incremental Learning Without Memory », *in*: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[156] James Smith et al., « Always Be Dreaming: A New Approach for Data-Free Class-Incremental Learning », *in*: ICCV (2021).

[157] Miguel Solinas et al., « Beneficial Effect of Combined Replay for Continual Learning », *in*: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 2, Online Streaming, February 4-6, 2021*, ed. by Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, SCITEPRESS, 2021, pp. 205–217.

[158] Emma Strubell, Ananya Ganesh, and Andrew McCallum, « Energy and Policy Considerations for Modern Deep Learning Research », *in*: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 13693–13696.

[159] Christian Szegedy et al., « Going deeper with convolutions », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, IEEE Computer Society, 2015, pp. 1–9.

[160] Youssef Tamaazousti, Hervé Le Borgne, and Céline Hudelot, « MuCaLe-Net: Multi Categorical-Level Networks to Generate More Discriminating Features », *in*: *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[161] Youssef Tamaazousti et al., « Learning More Universal Representations for Transfer-Learning », *in*: *arXiv:1712.09708* (2017).

[162] Xiaoyu Tao et al., « Few-Shot Class-Incremental Learning », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12183–12192.

[163] Xiaoyu Tao et al., « Topology-Preserving Class-Incremental Learning », *in*: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIX*, ed. by Andrea Vedaldi et al., vol. 12364, Lecture Notes in Computer Science, Springer, 2020, pp. 254–270.

[164] Bart Thomee et al., « YFCC100M: The New Data in Multimedia Research », *in*: *Communication ACM* (2016), pp. 64–73.

[165] Ashish Vaswani et al., « Attention is all you need », *in*: *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[166] Gido M. van de Ven and Andreas S. Tolias, « Three scenarios for continual learning », *in*: *CoRR* abs/1904.07734 (2019).

[167] Ragav Venkatesan et al., « A Strategy for an Uncompromising Incremental Learner », *in*: *CoRR* abs/1705.00744 (2017).

[168] Vinay Kumar Verma et al., « Efficient Feature Transformations for Discriminative and Generative Continual Learning », *in*: *CoRR* abs/2103.13558 (2021).

[169] Liyuan Wang et al., « ORDisCo: Effective and Efficient Usage of Incremental Unlabeled Data for Semi-supervised Continual Learning », *in*: *CoRR* abs/2101.00407 (2021).

[170] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert, « Growing a Brain: Fine-Tuning by Increasing Model Capacity », *in*: *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[171] Peter Welinder et al., « Caltech-UCSD birds 200 », *in*: (2010).

[172] Max Welling, « Herding dynamical weights to learn », *in*: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, 2009, pp. 1121–1128.

[173] Yue Wu et al., « Incremental Classifier Learning with Generative Adversarial Networks », *in*: *CoRR* abs/1802.00853 (2018).

[174] Yue Wu et al., « Large Scale Incremental Learning », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 374–382.

[175] Ye Xiang et al., « Incremental Learning Using Conditional Adversarial Networks », *in*: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, IEEE, 2019, pp. 6618–6627.

[176] Yun Xiang et al., « Efficient Incremental Learning Using Dynamic Correction Vector », *in*: *IEEE Access* 8 (2020), pp. 23090–23099.

[177] Saining Xie et al., « Aggregated Residual Transformations for Deep Neural Networks », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 5987–5995.

[178] Jianwei Yang, Devi Parikh, and Dhruv Batra, « Joint Unsupervised Learning of Deep Representations and Image Clusters », *in*: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[179] Junho Yim et al., « A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 7130–7138.

[180] Donggeun Yoo and In So Kweon, « Learning Loss for Active Learning », *in*: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[181] Jaehong Yoon et al., « Lifelong Learning with Dynamically Expandable Networks », *in*: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[182] Lu Yu et al., « Semantic Drift Compensation for Class-Incremental Learning », *in*: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, IEEE, 2020, pp. 6980–6989.

[183] Sergey Zagoruyko and Nikos Komodakis, « Wide Residual Networks », *in*: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*, ed. by Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, BMVA Press, 2016.

[184] Lehel Csató Zalan Bodo Zsolt Minier, « Active Learning with Clustering », *in*: *Workshop on Active Learning and Experimental Design*, 2011.

[185] Friedemann Zenke, Ben Poole, and Surya Ganguli, « Continual Learning Through Synaptic Intelligence », *in*: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ed. by Doina Precup and Yee Whye Teh, vol. 70, Proceedings of Machine Learning Research, PMLR, 2017, pp. 3987–3995.

[186] Junting Zhang et al., « Class-incremental Learning via Deep Model Consolidation », *in*: *IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, 2020, pp. 1120–1129.

[187] Kaipeng Zhang et al., « Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks », *in*: *IEEE Signal Process. Lett.* 23.*10* (2016), pp. 1499–1503.

[188] Xingcheng Zhang et al., « PolyNet: A Pursuit of Structural Diversity in Very Deep Networks », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 3900–3908.

[189] Bowen Zhao et al., « Maintaining Discrimination and Fairness in Class Incremental Learning », *in*: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, IEEE, 2020, pp. 13205–13214.

[190] Zhun Zhong et al., « Neighborhood Contrastive Learning for Novel Class Discovery », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 10867–10875.

[191] Zhun Zhong et al., « OpenMix: Reviving Known Knowledge for Discovering Novel Visual Categories in an Open World », *in*: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 9462–9470.

[192] Bolei Zhou et al., « Places: A 10 million image database for scene recognition », *in*: *IEEE transactions on pattern analysis and machine intelligence* 40.*6* (2017), pp. 1452–1464.

[193] Peng Zhou et al., « M2KD: Multi-model and Multi-level Knowledge Distillation for Incremental Learning », *in*: *CoRR* abs/1904.01769 (2019).

[194] Zongwei Zhou et al., « Fine-Tuning Convolutional Neural Networks for Biomedical Image Analysis: Actively and Incrementally », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 4761–4772.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Apprentissage Incrémental Profond à Large Échelle

Mot clés : Apprentissage Incrémental, oubli catastrophique, classification d'image, réseaux de neurones convolutifs

Résumé : L'apprentissage incrémental (IL) permet l'adaptation d'agents artificiels à des environnements dynamiques dans lesquels les données sont présentées séquentiellement. Ce type d'apprentissage est nécessaire lorsque l'accès aux données passées est limité ou impossible, mais il est affecté par l'oubli catastrophique. Ce phénomène consiste en une chute drastique des performances des informations précédemment apprises lors de l'ingestion de nouvelles données. Une façon de résoudre ce problème est d'utiliser une mémoire limitée du passé pour rafraîchir les connaissances apprises précédemment. Actuellement, les approches basées sur la mémoire obtiennent les meilleurs résultats de l'état de l'art. Dans cette thèse, nous présentons plusieurs méthodes avec et sans mémoire du passé. Nos méthodes traitent l'oubli catastrophique soit (1) en calibrant les scores des classes passées et nouvelles à la fin du réseau, soit (2) en réutilisant les poids initiaux des classes passées, soit (3) en transférant les connaissances entre des datasets de référence et cibles. Nous étudions notamment l'utilité de la distillation largement utilisée et l'effet d'utiliser ou non une mémoire du passé. Des expériences approfondies contre des méthodes de l'état de l'art ont été menées afin de valider l'efficacité de nos méthodes.

Title: Large-Scale Deep Class-Incremental Learning

Keywords: Incremental learning, catastrophic forgetting, image classification, convolutional neural networks

Abstract: Incremental learning (IL) enables the adaptation of artificial agents to dynamic environments in which data is presented in streams. This type of learning is needed when access to past data is limited or impossible but is affected by catastrophic forgetting. This phenomenon consists of a drastic performance drop for previously learned information when ingesting new data. One way to tackle this problem is to use a limited memory of the past to refresh previously learned knowledge. Currently, memory-based approaches achieve the best state-of-the-art results. In this thesis, we present many methods with and without memory of the past. Our methods deal with catastrophic forgetting either by (1) calibrating past and new classes scores at the end of the network, or (2) performing initial class weights replay, or (3) transferring knowledge between reference and target datasets. We notably investigate the usefulness of the widely used knowledge distillation and the effect of enabling or not a memory of the past. Extensive experiments against a range of state-of-the-art approaches were conducted in order to validate the efficiency of our methods.