Mitigating catastrophic forgetting via feature transfer and knowledge consolidation for deep class-incremental learning

Quentin Ferdinand

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

ENSTA BRETAGNE

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES BRETAGNE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Sciences et technologies de l'information et de la communication - Informatique*

Par

Quentin FERDINAND

Mitigating catastrophic forgetting via feature transfer and knowledge consolidation for deep class-incremental learning

Thèse présentée et soutenue à Brest, le 22 novembre 2023
Unité de recherche : Lab-STICC (UMR CNRS 6285)

Rapporteurs avant soutenance :

Céline HUDELOT Professeur, CentraleSupélec
Sébastien LEFEVRE Professeur, Université Bretagne Sud

Composition du Jury :

Président :	Romain BILLOT	
Examinateurs :	Romain BILLOT	Professeur, IMT Atlantique
	David FILLIAT	Professeur, ENSTA Paris
Dir. de thèse :	Benoit CLEMENT	Professeur, ENSTA Bretagne
Co-encadrant :	Panagiotis PAPADAKIS	Maitre de Conférence, HDR, IMT Atlantique

Invité(s) :

Gilles LE CHENADEC Maitre de Conférence, ENSTA Bretagne
Quentin OLIVEAU Ingénieur, Naval Group

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial intelligence

BiC incremental bias correction

CIL Class-incremental learning

CNN Convolutional neural network

DER Dynamically expendable representation

FT Fine tuning

GPU Graphics processing unit

i.i.d. independent and identically distributed

iCarL incremental classifier and representation learning

IL Incremental learning

KD Knowledge distillation

KL Kullback-leibler divergence

ML Machine learning

MLP Multi layer perceptron

NEM Nearest exemplar mean

SGD stochastic gradient descent

WA Weight alignment

List of Symbols

$\mathcal{T}$ total number of incremental steps

t current incremental step

$\mathcal{D}$ dataset used for supervised training

$\mathcal{D}_t$ incremental dataset available at step t

$\mathcal{D}_{new}$ dataset containing all the data about new classes

$\mathcal{M}_t$ Rehearsal memory

$\Phi_\theta^t(\cdot)$ model trained during incremental step t parameterized by θ

$\phi(\cdot)$ feature extractor of the model

$\mathcal{H}(\cdot)$ classification layer of the model

$\mathcal{L}$ Loss function

$\mathcal{L}_{KD}$ knowledge distillation loss

$\mathcal{L}_{CE}$ Cross-entropy loss

INTRODUCTION

In recent years, the field of Artificial intelligence (AI), and specifically Machine learning (ML), has undergone remarkable evolution, demonstrating impressive achievements across various domains. Nevertheless, as the range of applications expands, a significant challenge remains: continual adaptation. This thesis centers on the field of incremental learning, revolving around the development of intelligent systems capable of performing complex tasks while continually adapting to new challenges and data. This introduction will first offer an overview of the context as well as the scope of our thesis, followed by a presentation of our main research contributions.

Context

Naval Group. This thesis is performed in the context of a Cifre PhD sponsored by Naval Group. Naval Group, formerly known as DCNS (Direction des Constructions Navales), is a major French defense and naval engineering company. The company designs, builds, and maintains submarines, surface ships, and associated systems and infrastructure. It is known for its expertise in naval architecture, shipbuilding, and marine engineering.

Industrial Motivation. Naval Group's interest in incremental learning lies in its application to boat recognition. Many ships nowadays use on-board cameras for marine surveillance and Naval Group develops automatic systems that classify any boat captured by the cameras. Current systems, however, as will be explained in more details later on, are not adaptable at all and need to be trained with an overwhelming amount of data before being deployed. Understanding incremental learning is the first step towards making fully adaptable models able to be trained on the fly, and adapt themselves even after being deployed.

Class-incremental learning. Often referred to as continual learning or even lifelong learning, incremental learning constitutes a research domain focused on training machine learning models able to adapt to evolving data and tasks as they emerge over time.

Class-incremental learning, a prominent sub-field within incremental learning, specifically addresses the challenge of training image classification models to handle the continuous addition of new classes or categories via the addition of new data. However, current state-of-the-art machine learning systems for image classification are primarily static; they require initial training to recognize specific categories of images before they can effectively classify them. Consequently, when faced with new data and categories, as is the case in class-incremental learning, these systems necessitate retraining on the new images to adapt. Furthermore, training them solely on this new data leads to a drastic loss of performances on the previous classes which is a famous problem of incremental learning known as "**catastrophic forgetting**" [1], [2].

Contributions

The classification models studied in this thesis employ feature extraction techniques to discern crucial patterns in images for classification purposes. This thesis is organized to first analyse catastrophic forgetting and its influence on these extracted features, and explore strategies to mitigate its effects. On this basis, the contribution of this thesis consists in developing new approaches that improve upon the state-of-the-art and which are distributed in specific chapters as follows :

— **Chapter 2 : Experimental analysis of the open challenges faced by incremental methods**

The first year of this thesis was dedicated to extensive experimentation on catastrophic forgetting and its impact on the features extracted by the model, and on the different incremental methods used to alleviate its effects. These experiments revealed a defect in the features extracted by the models trained incrementally and laid the foundations for the methods developed in the sequel of this thesis.

— **Chapter 3 : Joint incremental and contrastive learning**

Attempting to solve the feature issues highlighted by our previous experimentation, a new approach is introduced for training models jointly with contrastive and incremental methods. In this chapter, the presented approach makes use of contrastive losses to improve the features extracted by the network while relying on the incremental knowledge distillation loss to alleviate catastrophic forgetting at a classifier level. This work led to the following publication :

1. Ferdinand Quentin, Clement Benoit, Oliveau Quentin, Le Chenadec Gilles, and

Papadakis Panagiotis (2022). "Attenuating catastrophic forgetting by joint contrastive and incremental learning". In the 3rd Workshop on Continual Learning in Computer Vision. Workshop of the Conference on Computer Vision and Pattern Recognition 2022 (CVPR).

— **Chapter 4 : FECIL : Feature Expansion and enhanced Compression for Incremental Learning**

By combining the emerging concept of dynamic networks and an improved model compression method that we developed, we proposed a method for extracting better incremental features than our contrastive-based method while also incurring lower computational overhead. This work is being finalized for an article submission in parallel with the redaction of the manuscript.

Chapter 1

Related works

In this chapter, we will begin by providing a detailed description of state-of-the-art machine learning methods for image classification. Following that, a comprehensive overview of the class-incremental learning problem and its challenges when applied to these specific image classification systems will be presented. Lastly, an in-depth explanation of the state-of-the-art techniques used to alleviate catastrophic forgetting in the class incremental learning setting under consideration in this thesis will be provided.

1.1 Machine learning for image classification

This thesis revolves around two main themes, machine learning and more specifically Incremental learning (IL) and image classification.

Machine learning encompasses various approaches, each tailored to distinct learning scenarios. For this thesis and image classification problems in general, supervised learning is the preferred solution due to the large amount of labeled data easily obtainable leading to high performance of models trained this way. Supervised learning is a well-established paradigm where models are trained on labeled datasets, meaning the input data is paired with corresponding true output labels. The goal is to learn a mapping function that can make predictions or classifications on new, unseen data accurately. Generalization capability is a critical aspect of supervised learning as it determines a model's ability to perform well on unseen or new data, which is essential for practical applications. Models that exhibit strong generalization can make accurate predictions or classifications beyond the training data they were exposed to. This capability ensures that the model can effectively handle real-world scenarios and adapt to various conditions.

While traditional machine learning models, such as decision trees or linear regression, may still be suitable for simple tasks and situations with limited data, neural networks have become the go-to choice when dealing with complex, large-scale, and high-dimensional data. In fact, neural networks and particularly deep neural networks, have

gained significant popularity in recent years due to their remarkable capacity for generalization. They excel at learning complex and hierarchical patterns in data, making them well-suited for tasks involving unstructured or high-dimensional data, such as images, text, and speech. Their superior generalization capability, coupled with advancements in training techniques and network architectures, has positioned them as the dominant approach in supervised learning.

1.1.1 Artificial neural networks

Artificial neurons are the fundamental building blocks of artificial neural networks. Inspired by their biological counterparts, these computational units receive input signals, compute a weighted sum of the inputs and output an activation value. Mathematically, they operate by computing the sum of the dot product between inputs and a weight vector, and add a bias term to this sum. This activation of the neuron is then passed through a non-linear activation function to produce the final output of the neuron. The choice of activation function can vary and introduces non-linearity into the neuron's output, allowing it to model complex relationships in the data.

Figure 1.1 illustrates the operation of an artificial neuron. Considering m inputs $\{x_i\}_{i=1}^m$, the neuron's weights $\{\omega_i\}_{i=1}^m$, the bias term b, and an activation function $f(\cdot)$ the equation of a neuron's output is the following :

$$y = f(\sum_{i=1}^{m}(x_i \cdot \omega_i) + b) \tag{1.1}$$

Artificial neural networks represent interconnected layers of these artificial neurons. These layers are often referred to as dense or fully connected layers where each neuron is linked to each neuron of the previous layer. Simple neural networks are generally organized in an input layer, followed by one or more "hidden" layers, until the final output layer. Information flows through these layers, with each layer's neurons transforming and processing the data. Due to the non-linear activation function of each neuron, chaining them together layer after layer allows to model increasingly complex functions. This function, which the neural network models, is entirely defined by the weights $\{\omega_i\}_{i=1}^m$ used by each neuron. For image classification problems this function ideally takes either specific features extracted beforehand or directly all pixels of an image as input and outputs the correct label of the input image.

Initially, the favored approach to extract meaningful features for classification involved

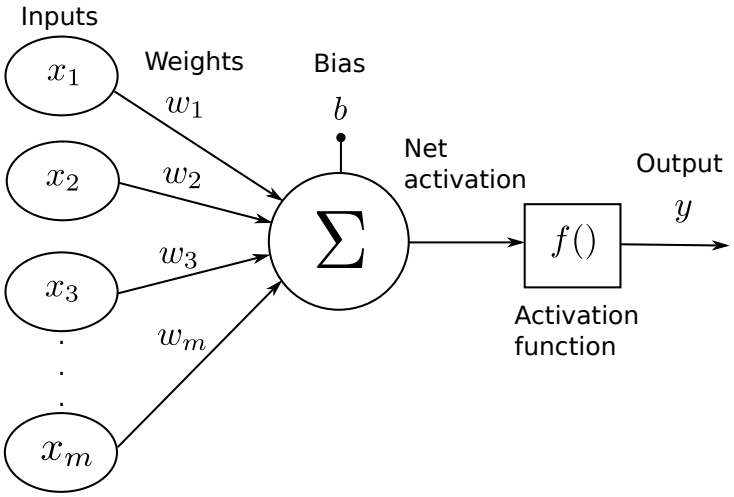


Figure 1.1 – Schema illustrating each component of an artificial neuron.

utilizing specific algorithms depending on the problem and features needed. However, with the emergence of deep learning and the advent of convolutional neural networks, a notable shift has occurred towards working directly with all of the image pixels.

1.1.2 Convolutional neural networks

As illustrated in figure 1.3, Convolutional neural networks (CNNs) are deep neural networks with many layers and weights that are specifically designed to deal with images. They employ convolutional layers that apply filters to extract local patterns and features from the input data, capturing spatial hierarchies. This features are then input into a classification head, or classifier, which consists of a standard neural network, usually of one or two layers. This classification head then outputs one value called an output logit for each considered classes representing the CNN's prediction about the input image's probability of belonging to each class.

Specifically, convolution layers apply a convolution operation to their inputs with a specific kernel composed of the layer's weights before introducing non-linearity via an activation function. This convolution operation amounts to cutting the input in small portions and computing the weighted sum of inputs with a specific filter, the convolution kernel, to this small image (see figure 1.2). Depending on the size and weights of the convolution kernel and activation function used, different features will be extracted from the input. In most CNN architectures many kernels are used per layers to extract different features from this small portion of the image that are then passed to the kernels of the

next layer. This process results in features initially quite abstract, representing horizontal and vertical edges in the input for example, that become more and more complex and end up extracting the presence of eyes and other discriminative features.

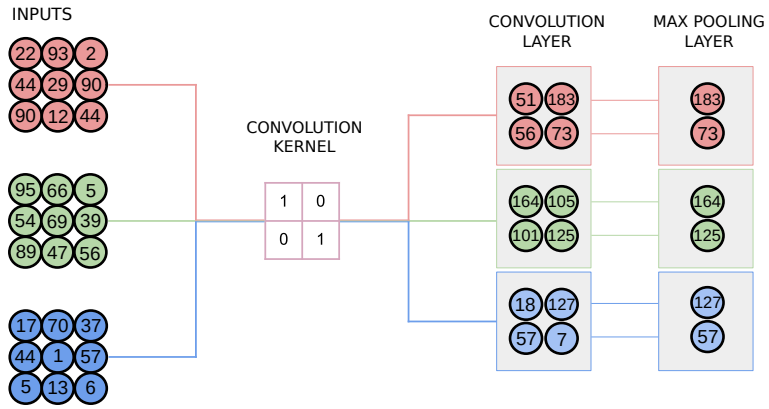


Figure 1.2 – schema representing the operation of convolution and pooling layers. Here the convolution kernel is of size 2x2 with a stride 1x1, and the pooling layer represented is a 2x1 maximum pooling layer.

Convolution layers, however, increase the dimensionality of the input, which can increase significantly the feature extraction computation time, therefore many CNN include pooling layers after each block of convolution layers. As illustrated in figure 1.2, pooling layers downsample the inputs by moving a convolution filter across them, and taking the maximum value (max pooling) or the average (average pooling) of small portions of the input. For example, in figure 1.2 the represented maximum pooling layer employs a filter of size 2 to cut in half the number of dimensions of the convolution layer outputs.

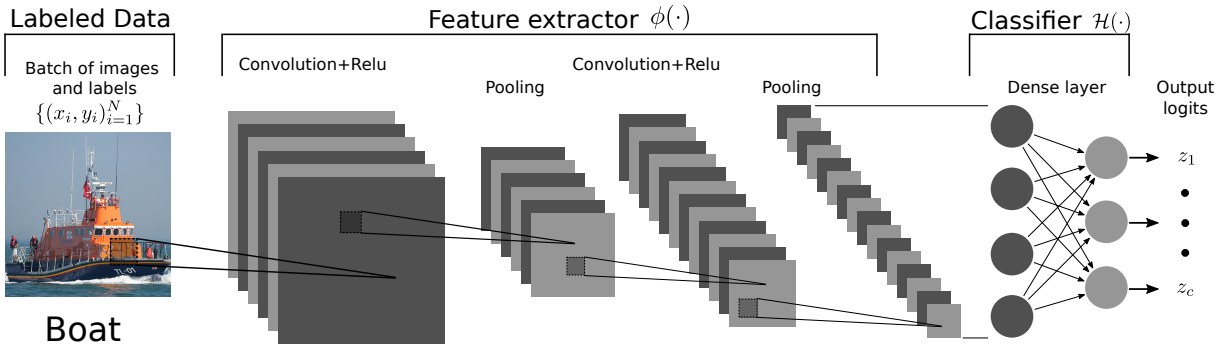


Figure 1.3 – Schema of a convolutional neural network. Multiple convolution and pooling layers extract features from the input images, these features are then fed into a fully connected layer that will output the final prediction of the model.

1.1.3 Supervised learning

For image classification tasks, deep convolutional neural networks are typically trained in a supervised way. Supervised learning consists in training a neural network with a labeled dataset where the label of each instance corresponds to the ground truth supervision that is used for training. A neural network can be considered as a function $\Phi_\theta : \mathcal{X} \mapsto \mathcal{Y}$ defined by its weights θ that maps an input space $\mathcal{X}$ into an output $\mathcal{Y}$.

Considering an image classification problem with C different categories and a labeled dataset $\mathcal{D} = \{(x_i, y_i)_{i=1}^N\}$, where N is the total amount of data, x_i a 3-dimensional matrix of values within $[0, 255]$ corresponding to the pixels of an RGB image and $y_i \in [0, C[$ its corresponding class label, the weights θ of the neural network are then optimized so that Φ_θ predicts the correct label for each image of the dataset.

In order to predict the correct labels, the neural network is designed to take a 3D matrix $\mathcal{X}$ as input and output a vector $z \in \mathbb{R}^C$ containing output logits corresponding to each of the C classes considered. Then a softmax activation function is applied to z in order to ensure each output represents the probability of the input image of belonging to a class :

$$p_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \tag{1.2}$$

with z_j the output logit j of the neural network and p_j the corresponding probability of the input image of belonging to the class j. The prediction of the neural network $\hat{y}$ is then the class with the highest probability :

$$\hat{y} = \underset{j}{\operatorname{argmax}}(p_j)$$

Finally, the labels y are transformed into one-hot vectors (vectors of size C filled with 0 and with a 1 at the position y) and the weights of the CNN are optimized via the minimization of the following categorical cross entropy loss function :

$$\mathcal{L}(\hat{y}, y) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{1.3}$$

This loss function measures the disagreement between the network's prediction and the ground truth label. Therefore, the goal of the supervised training process is to find the optimal set of weights θ^* that minimizes the loss over the training dataset $\mathcal{D}$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left\{ \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(\Phi_\theta(x), y) \right\}$$

There are usually millions of parameters in CNNs, resulting in an optimization problem with millions of dimensions. Due to the vast number of dimensions, exact solutions cannot be derived analytically and global optimization methods are too time-consuming to find the global optimum. Consequently, an iterative algorithm known as the mini-batch stochastic gradient descent (SGD) is employed to efficiently locate a satisfactory local optimum:

Algorithm 1 Pseudo-code of the SGD optimisation process

Require: dataset $\mathcal{D} = \{(x_i, y_i)_{i=1}^N\}$
Require: model Φ_θ with trainable parameters θ
Require: learning rate η, batch size $\mathcal{B}$, max number of epochs E
Require: loss function $\mathcal{L}$
  for epochs in $[1, E]$ **do**
    $X, Y = \{(x_i, y_i)_{i=1}^{\mathcal{B}}\} \leftarrow$ sample a minibatch of size $\mathcal{B}$ in $\mathcal{D}$
    Forward: $\hat{y} \leftarrow \Phi_\theta(X)$
    Compute loss: $\mathcal{L} \leftarrow \mathcal{L}(\hat{y}, Y)$
    Backward:
      Compute gradients: $\delta \leftarrow \nabla_\theta \mathcal{L}$
      Update parameters: $\theta \leftarrow \theta - \eta\delta$
  end for

This algorithm first realizes a "forward" pass of the neural network, a small batch of data is input to the first layer of neural network, it then goes through all the layers and the disagreement between the output and the ground truth labels is measured via the loss function. Then a "backward" pass is done where the gradient of the loss with respect to the weights of each layer of the neural network is computed.

This gradient is computed using the backpropagation algorithm first discovered in 1976 by S. Linnainmaa [3], and then first applied to neural networks in 1986 by Rumelhart et al. [4], and to deep CNNs by Lecun et al. in 1989 [5]. This algorithm computes gradients relative to each weight by accumulating and back-propagating errors from last layers to first layers of the neural network.

These gradients give the direction in which to update the network's weights in order to minimize the loss, and the SGD algorithm then makes use of a learning rate η in order to control the size of the step taken towards the minimum of the loss.

1.2 Incremental learning

As explained in previous sections, researchers have obtained state-of-the-art results with supervised learning and CNNs on a wide variety of tasks such as object recognition, image segmentation, natural language processing, up to sound classification. These CNNs even achieve or surpass human performances on many learning tasks [6]–[8], however one big drawback compared to the human brain faculty still remains, namely, the lack of adaptability. Indeed, humans learn continually, constantly adding knowledge to what they already know, whereas CNNs are trained once, then deployed and therefore will not change overtime even if the real world data changes. For example, a CNN trained to recognize someone's face might be very effective just after training, however, a few years later the person's face would have changed so much that the model would have much worse performances.

The main issue stems from the fact that these models are designed to be static, they are trained to perform well on one task and are then deployed to perform this particular task. Therefore, if we wish to add knowledge on top of already acquired knowledge, or accomodate a change in the real world data distribution, the model would have to be retrained from scratch using all previously seen data in addition to the newly encountered data.

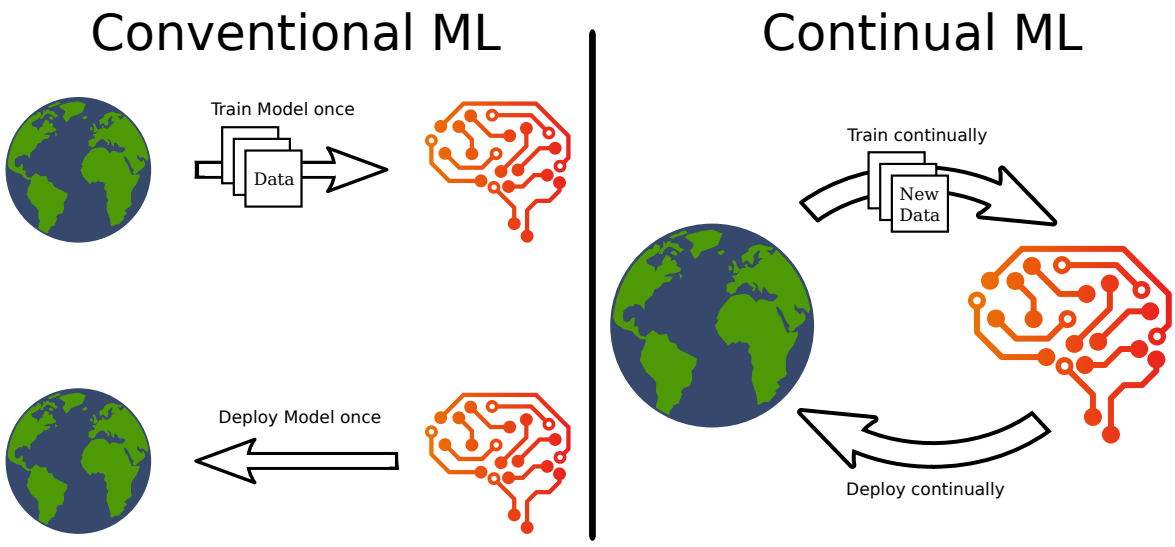


Figure 1.4 – Difference between classical machine learning and continual/incremental learning. Classical machine learning is considered static whereas continual learning revolves around constantly adapting models to newly acquired data.

Unlike traditional machine learning, the field known as incremental learning, continual learning, or lifelong learning (as discussed by Thrun et al. in [9]), focuses on creating models and training methods that excel at quickly adapting to new tasks and data. As illustrated in figure 1.4, the core idea resides in developing models with the ability to adapt to new data across an infinite range of incremental tasks.

Traditional machine learning models would necessitate retraining with all prior data in addition to the new data for adaptation, a cumbersome and resource-intensive approach. In contrast, continual learning methods concentrate solely on the new data, making them highly efficient and practical, especially in scenarios with limited computational resources and time constraints.

Incremental learning is the central field of study of this thesis and more specifically its application to image classification. The usual consideration for this application is that the model is trained on τ different tasks incrementally. Each incremental task requires an incremental training step $t \in [0, \tau[$, where the step $t = 0$ is the initial training step and each subsequent incremental task consists in training the model on a classification task of $C_t = C_{old} + C_{new}$ classes. During each incremental step, the goal is then to accommodate the C_{new} new classes with minimal forgetting of the C_{old} previous classes.

1.2.1 Challenges of class incremental learning

The straightforward solution for adapting image classification models to new data and classes would be to retrain them with all seen data whenever new data appears which corresponds to a conventional supervised training step and therefore optimal performance on all classes. However, this method presents a notable drawback: with each update, the dataset would continuously expand, resulting in longer training times and increased data storage demands. Eventually, these issues could render training impractical. Consequently, the core challenge in incremental learning lies in the ability to adapt models using only the newly acquired data. While updating models by training them exclusively on new data addresses storage and training time concerns, it also introduces the primary challenge of incremental learning: catastrophic forgetting.

As its name suggests, catastrophic forgetting represents an extreme drop of performance on the classes learned in previous incremental steps when doing a new training step. This issue arises because, as explained in section 1.1.3, all the weights of the model are optimized to minimize the classification loss on the dataset. Indeed, since only the new data is used for training, the previous classes are not represented in the dataset, therefore

the weights are optimized for performance only on the new classes. Since all the weights are updated, a weight drift happens as shown in Figure 1.5 which causes the weights that were important for previous classes to change, resulting in catastrophic forgetting.

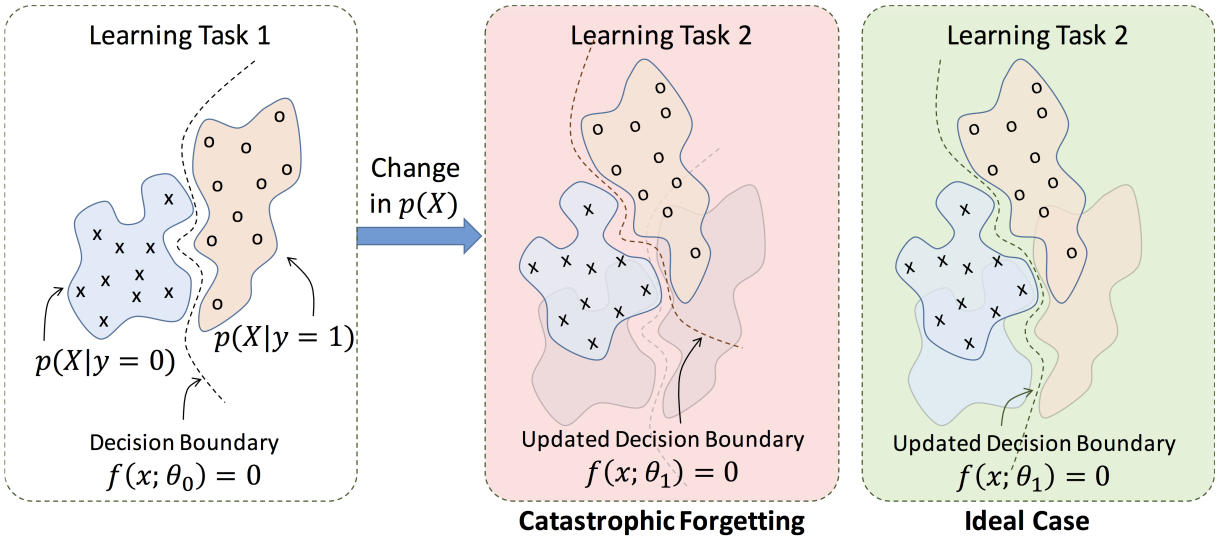


Figure 1.5 – schema representing the catastrophic forgetting happening in neural networks during incremental learning. The decision boundary changes to separate the new classes but does not separate the previous classes anymore.

While catastrophic forgetting may arguably be considered the main challenge of incremental learning, many other issues arise with different methods used to alleviate it. We therefore define some important properties of incremental learning algorithms and will refer to them when comparing different methods:

Memory overhead: Incremental learning techniques often require storing information from previous tasks in memory during the learning process, so as to prevent forgetting. This can include storing previous data and models, as well as features or statistics about previous tasks. However, this can result in a significant memory overhead for the learning algorithm, which can become a problem as the number of incremental tasks grows. Therefore, it is important to minimize the memory overhead of incremental learning algorithms to ensure their relevance and scalability in handling a large number of tasks.

Training time overload: During incremental learning, complex training procedures are often added to prevent forgetting, which can increase the forward and backward computation time of the stochastic gradient descent algorithm (see algorithm 1) or add

more training steps for each incremental task. This can result in a significant increase in training time, which can become a problem as the number of incremental tasks grows. While a minor increase in training time may not be an issue, a significant increase can lead to intractable training times for a large number of incremental tasks. Therefore, it is important to minimize the increase in training time during incremental learning to ensure its scalability and practicality in handling a high number of incremental tasks.

Plasticity and stability: The plasticity-stability trade-off is a fundamental challenge of continual learning which refers to the ability of a neural network to adapt to new tasks while retaining previously learned knowledge. On one hand, plasticity refers to the ability to learn and incorporate new information, which is crucial for successful continual learning. On the other hand, stability is the ability to maintain previously learned information without interference from new learning, which is essential for avoiding catastrophic forgetting. The challenge is to find the right balance between these two competing factors, as too much plasticity can lead to overfitting to new tasks and forgetting of previous knowledge, while too much stability can lead to underfitting and difficulty in learning new tasks. Therefore, the plasticity-stability trade-off problem is a crucial consideration in designing effective continual learning algorithms.

Average incremental accuracy and forgetting: When training neural networks in a supervised manner, the accuracy obtained at the end of the training step is usually used for performance comparison. However, in order to consider the plasticity-stability trade-off explained above, metrics known as forgetting and average incremental accuracy are mainly used to compare performance in incremental studies [10]–[14]. The average incremental accuracy consists in computing the standard test accuracy of the network on all seen classes at the end of each incremental step and averaging them. While conceptually simple, this incremental accuracy is convenient for measuring the plasticity-stability trade-off. In fact, a model only stable would reach high accuracy only in the first incremental step while a model with too much plasticity would reach high performance only in the last incremental step, which in both cases leads to poor average accuracy.

For a more precise evaluation of the stability of the model, forgetting measures are sometimes used and computed in the following ways :

Considering an incremental step t, a previous incremental task $k < t$, and $a_{k,t}$ the test accuracy obtained on the new classes of task k after the incremental training step t, the

forgetting measure [15] of task k after incremental step t is computed in the following way:

$$f_t^k = \max_{l \in \{1,\dots,t-1\}} (a_{k,l} - a_{k,t}), \forall k < t \tag{1.4}$$

This equation gives a triangular matrix of forgetting measures for each task $k < t$ and each incremental step t, therefore, it is usually summarized in one measure after each incremental step t, either the initial task forgetting f_t^0 or the average task forgetting F_t:

$$F_t = \frac{1}{t-1} \sum_{k=1}^{t-1} f_t^k \tag{1.5}$$

Compared to f_t^0, F_t is a better indicator of the overall forgetting happening during each incremental step, however it's computation is based on the maximal accuracy obtained for each incremental tasks, which is biased towards stability. Indeed, an incremental method allowing very little plasticity would reach very poor maximal accuracy on new classes which would inherently allow for very little forgetting of these classes while another method learning new tasks much better would have much more to forget. For this reason the initial task forgetting f_t^0 is sometimes preferred [15] because the initial task is the only one that is trained using standard supervised training which ensures any incremental method would reach a similar maximum accuracy for the same task which allows a much more precise measure of forgetting.

1.2.2 The class incremental setting

In this thesis, we focus on continual learning applied to image classification problems. As this topic remains very challenging (see section 1.2.1), many different simplifications and settings have been proposed to study the problem:

Online and Offline IL are two different approaches to continual learning. On one hand, online incremental learning trains a model continuously as new data becomes available over time. Data from any class can be seen at any time and the model has to learn by seeing each data only once. This approach is the most difficult form of continual learning and shares challenges with the field of few-shot learning where the challenge is to train models with few samples. Offline incremental learning, on the other hand, involves training a model on a fixed dataset during multiple incremental steps. During each incremental

step the model is trained on a different training task with a new dataset containing all the available data about C new classes. After each step, the model is expected to classify correctly both new and previous classes and is then used as a starting point for the next step.

For this thesis the choice was made to focus on the offline setting as it is the most commonly studied setting and allows us to study specifically catastrophic forgetting without having to also take few-shot learning issues into account.

No memory vs memory. A commonly employed technique to address forgetting in continual learning is the use of "rehearsal memory". In Section 1.3.1, we will provide a detailed explanation, but in essence, it involves storing samples from previous incremental steps in a fixed size memory and utilizing them during subsequent incremental steps. This memory is then added to the incremental dataset during training which incurs memory and training time overheads. Although avoiding the use of this memory is preferable, it is considered a much more challenging setting, and hence, many works on continual learning rely on it to alleviate forgetting.

In this thesis, this rehearsal memory is used ans studied because while it induces a small computational overhead it also allows incremental methods to reach much higher performances.

Single-head and Multi-head refer to different evaluation settings and are also sometimes called class incremental learning and task incremental learning. In the single-head setting, using one classification head, the model is required to classify correctly images from all seen classes without distinctions. In the multi-head setting, the model uses one classification head per incremental task and uses a task identifier for each test image to choose the correct classification head to classify it.

This multi-head setting is much simpler because the model is not required to discriminate between classes seen in different tasks and simply uses the task identifier instead at test time. In real world applications however, this task identifier is rarely obtainable, which is why we chose to study exclusively the single-head setting in this thesis.

1.2.3 Problem formulation

To summarise the class incremental setting considered in this thesis, a convolutional neural network is trained on $\mathcal{T}$ classification tasks sequentially. For each task t, a dataset

$\mathcal{D}_{new}$ containing all the available data $(\mathcal{X}_t, \mathcal{Y}_t)$ about C_t new classes is acquired, where $\mathcal{X}_t$ is a set of images and $\mathcal{Y}_t$ the ground truth labels associated. Let us denote $\Phi_\theta^t(\cdot)$ the model whose parameters θ are getting optimized during the incremental learning step t, and $\mathcal{L}$ the loss function used for the optimisation. If we consider the task t then the goal of the incremental learning process is to minimize the empirical risk of all seen tasks given access to data from previous tasks limited to the samples stored in the small rehearsal memory $\mathcal{M}_t$ (see section 1.3.1) and total access to data $(\mathcal{X}_t, \mathcal{Y}_t)$ from current task t:

$$\frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \mathcal{L}(\Phi_\theta^t(\tilde{\mathcal{X}}_t); \tilde{\mathcal{Y}}_t) \tag{1.6}$$

Therefore, the goal is find the optimum θ so that the cost function L is minimized, using $\mathcal{D}_t = \mathcal{D}_{new} + \mathcal{M}_t = \{\tilde{\mathcal{X}}_t, \tilde{\mathcal{Y}}_t\}$.

The baseline method for supervised learning of a classification task is to minimize the cross-entropy loss over the dataset with respect to the model parameters, but the quality of a model trained with cross-entropy loss is highly dependent on how well the dataset represents the real world distribution for the task. Therefore, if access to previous data is limited the performance of the model on the previous tasks will quickly degrade, and this is where the main problem of incremental learning, catastrophic forgetting, occurs.

1.3 Incremental learning state-of-the-art

The methods used to mitigate catastrophic forgetting in class-incremental learning can be divided into three categories. In the subsequent section, we discuss the first approach, often referred to as data rehearsal, where data from previous classes is rehearsed while learning new ones. We then elaborate on how regularization terms are incorporated into the training loss to mitigate forgetting, and further delve into the solutions that have been proposed to address the inherent imbalance of incremental training datasets.

1.3.1 Rehearsal

Rehearsal based methods constitute one of the first lines of works that emerged to address catastrophic forgetting. The basic premise of these methods stems from a straightforward observation: the lack of data from previous classes when learning new classes is the root cause of catastrophic forgetting. To mitigate this issue, rehearsal-based methods aim to retain some data during each incremental steps in order to replay them in later incremental steps.

The performance of these methods is heavily dependent on the quantity of data stored while the upper performance bound is the performance of joint training on all seen data. However, in order for these methods to remain applicable to a theoretical infinite number of incremental tasks, the memory used to store previous data has to be considered of fixed size. Since the size of this memory is fixed, and usually rather small to avoid memory and training time overhead, one of the challenges of this area of research is to make sure the most representative samples are stored for each class.

There are two main types of rehearsal, the first one often referred to as data rehearsal, consists in storing real samples seen in earlier incremental steps either in image or compressed feature vector form and replaying them during subsequent steps. The second one, on the other hand, is called pseudo-rehearsal and revolves around using generative models to generate previous classes' data without storing large amounts of previous data. Indeed, storing real images can be costly in terms of memory overhead especially for high resolution images, and replaying them artificially increases the incremental dataset size which induces a training time overhead. Therefore pseudo-rehearsal based methods have been proposed involving storing only a generative model able to generate pseudo-images or feature vectors of previous classes to reduce memory overhead.

Data rehearsal

As introduced earlier, rehearsal of previous classes' data should remain limited in order to maintain the method's scalability across a large number of incremental steps and classes. While early works [16], [17] considered fixing the number of samples per class in the memory, the majority of algorithms have since adopted the approach initially proposed by [10], which involves setting a fixed total memory size. This approach offers two key advantages: firstly, it ensures that the memory size remains constant as incremental steps increase, making the method highly scalable for numerous classes and steps. Secondly, in the early incremental stages when the model has not encountered many classes yet, much more samples can be stored. Indeed, the memory can be entirely filled until new classes are encountered and some of these stored exemplars must be replaced by with new ones from subsequent classes.

Given the fixed memory size, storing all previous samples is not possible. Instead, a selection process is required to retain only the most representative samples. The choice of this sampling strategy significantly impacts the effectiveness of data rehearsal, yet over the years, only a limited number of distinct methods have been put forth in this regard:

Random selection. It is the simplest sampling strategy and consists in randomly sampling K samples from the total amount of N images available for each new class $\mathcal{C}_t$ during each incremental step.

Distance-based sampling. In their algorithm Rwalk [15], the authors considered samples closer to decision boundaries of each class as the most representative ones. Therefore, considering a sample (x_i, y_i), they proposed using the corresponding output logit as a pseudo-distance to the decision boundary:

$$d(x_i) = o_{y_i} = \phi(x_i)^\top w^{y_i}$$

with ϕ the model's feature extractor and w^{y_i} the weights of the last fully connected layer for the class y_i. They then considered sampling exemplars x_i based on a probability $q(x_i)$ proportional to $\frac{1}{d(x_i)}$ in order to sample more exemplars with small distances for each class.

Entropy-based sampling. Another strategy proposed by the same authors [15] was to instead store in memory the samples for which the model is the most uncertain about. The uncertainty of samples was measured using the entropy of the output softmax distribution and the higher the entropy the more likely the sample would be sampled.

Herding selection. In iCarL [10] authors took another approach and instead proposed to keep in memory samples based on the class mean feature vector. For each class a set of selected exemplars is constructed iteratively so that adding the selected exemplar to the set of already selected ones minimize the distance between the set mean feature vector and the real mean feature vector of the class (see algorithm 2).

Interestingly, the simple random selection has been shown in several studies [15], [18] to outperform both the distance and entropy based methods. Indeed, random selection demonstrated performances very similar to the best performing method : herding selection, while being a lot less computationally intensive. Herding selection, however, is more suitable for very long-term incremental learning, where there are many classes and only one example per class can be stored in a fixed memory size. In this case, herding guarantees that the stored example is the most representative one because it is the closest to the mean of the class.

Algorithm 2 Herding sampling strategy

Require: image set $X = \{x_1, ..., x_n\}$ of class y
Require: feature extractor of the current incremental model : $\phi()$
Require: number of exemplars to sample m for class y
  $\mu \leftarrow \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)$ ▷ compute the class mean feature vector
  for $k = 1,...,m$ **do**
    $s_k \leftarrow \underset{x \in X}{\operatorname{argmin}} ||\mu - \frac{1}{k}[\phi(x) + \sum_{j=1}^{k-1} \phi(s_j)]||$ ▷ choose the exemplar that keeps the set mean the closest to the real mean
    $X \leftarrow X \setminus \{s_k\}$ ▷ Remove selected sample to avoid duplicates
  end for
  $S \leftarrow \{s_1, ..., s_m\}$
  Output: Set of selected exemplars S for the class y

While random and herding selection are now considered as baselines sampling methods, some other improvements have been proposed to enhance the rehearsal memory.

Rehearsal improvements

Storing real images for rehearsal during an incremental learning process induces a memory footprint that can be expensive storage-wise for some applications, therefore another line of works [19], [20] focuses on reducing the amount of storage needed to store previous data. One way to do so is to avoid storing images, and store previous data in feature vector form instead. Feature vectors are intermediate representations of the initial images on which are based the predictions of the network, they therefore contain all the discriminative information of real images while containing considerably less values. In fact, storing images in feature vector form would reduce considerably the amount of storage needed per sample, thus allowing much more data to be stored, resulting in improved rehearsal performances. An important limitation of these works is that in order for the features stored to remain relevant in later incremental steps, the feature extractor of the model has to be frozen during the entire incremental process. Since the feature extractor is frozen during the incremental steps, the performances of these methods is highly dependant on the quality of the initial feature extractor training [19].

Conversely, in their work "Mnemonics training", Liu et al. [21] proposed to store images in the rehearsal memory, make the stored exemplars optimizable and optimize them to minimize forgetting. Indeed, they propose a "bilevel" optimization scheme consisting of two training steps, a model-level and an exemplar-level optimisation step. Considering an incremental step t, in the first model-level optimisation step they perform the regular training step of a baseline incremental algorithm to train the model $\Phi_\theta^t(\cdot)$. Then, they add a secondary optimisation step where $\Phi_\theta^t(\cdot)$ is frozen and optimize the rehearsal exemplars stored to minimize the cross-entropy loss of the model on the current incremental dataset.

While adding a lot of complexity to the chosen baseline incremental algorithm, they showed experimentally that their method outperformed the classic random and herding sampling strategies when added to multiple incremental methods. They also analysed the classes' clusters in the feature space of the model and showed that their approach sampled exemplars mostly located on the boundaries between clusters. This finding is particularly noteworthy as it contradicts the conventional idea that exemplars closer to the mean of the class are more representative, idea on which the herding sampling strategy is based.

Pseudo-rehearsal

Pseudo-rehearsal methods follow the same objective than feature-based approaches as they both focus on improving the number of samples used for rehearsal. Pseudo-rehearsal methods avoid storing data and instead train and store generative models with the aim of being able to generate as many and varied synthetic samples as needed during incremental steps. This type of approaches typically require storing and training large generative models which, similarly to data rehearsal, induces a memory and training time overhead. However their advantage resides in their ability to generate much more samples and more diverse ones than with data rehearsal. Considering a perfect generator, able to generate an unlimited amount of diverse images from all previous classes, would not only improve rehearsal quantity and quality, but also solve the imbalance issue faced by incremental methods that we describe in more details in section 1.3.3.

Many works therefore emerged from this concept, experimenting with different generative models such as autoencoders [22], Generative Adversarial Networks (GaNs) [23], [24] [25], and class conditional GaNs [26]. These generative methods all require a complex training procedure that usually involves training a generator and a discriminator network simultaneously and remain limited by the quality of the current state-of-the-art models in this domain that still produce images far from perfect. Moreover, in an incremental setting the generator has to be trained incrementally aswell, which implies that it is subject to catastrophic forgetting [27] exactly like the classification model which adds to the difficulty of producing correct samples to replay in later incremental steps. While these pseudo-rehearsal methods typically reduce the memory overhead of rehearsal, and improve the quantity and diversity of the rehearsed data, the performances remain limited due the rapid degradation of generated images' quality with the incremental steps.

While pseudo-rehearsal is better in terms of rehearsal quantity, data rehearsal is superior in terms of rehearsal quality, therefore combinations of both have been studied [28] to benefit from the advantages of each strategy while limiting their drawbacks. In fact, it was shown by Solinas et al. [28] that combining real and synthesized samples coming from a reduced rehearsal memory and a generative model leads to improved performances and reduced forgetting.

The approaches outlined in this section aim to mitigate forgetting by simply replaying information about past classes during incremental steps, in the subsequent sections we will describe other effective techniques have been proposed based on entirely different principles. Specifically, we will describe regularization-based methods that modify and add

terms to the loss function optimized during incremental steps. These adjustments allow a better control of the training process that helps mitigating catastrophic forgetting.

1.3.2 Regularizations

Regularization is a technique used in machine learning used to control the training process that works by adding terms to the loss function so that the optimization process tends to move weights to a favorable optimum. L2 regularization, for example, is generally used when training deep CNNs to reduce the overfitting which occurs when a model becomes too complex and fits too closely to the training data, leading to poor performance on testing data. The L2 regularization, also known as weight decay, adds a constraint to the weights in the form:

$$L_2(\theta) = ||\theta||^2$$

with $||x|| = \sqrt{x_1^2 + ... + x_n^2}$ being the Euclidean norm, which adds a penalty based on the amplitude of the weights of the model. This L_2 term is then added to the loss being minimized during the optimization process which will lead to an optimum with weights much closer to zero, which has been shown to reduce overfitting.

In incremental learning these methods are not used to reduce overfitting but instead to prevent forgetting of previous knowledge by forcing the model's weights and biases to retain the important information about past tasks when learning new ones. Two different categories of regularization-based methods are used in incremental learning, in the following sections we will describe both of them.

The first one is often referred to as weight drift regularization and regroups approaches that aim at reducing the weight drift explained in 1.2.1. These approaches consider that the most important weights for past tasks should change as little as possible when the model is trained on new tasks in order to alleviate forgetting.

While these approaches have demonstrated impressive results in alleviating forgetting they have been shown to hinder the plasticity of models trained incrementally which has led to another type of regularization-based methods called knowledge distillation. This knowledge distillation allows better plasticity of the model by not constraining weights directly but only the output distribution of the models.

Weight drift regularizations

The main reasoning behind weight drift regularizations is that many sets of locally optimal θ_t will result in the same performance when optimizing the model on the new task t [29]. Therefore, it is likely that one of them lies close to the optimal set of parameters from the last task θ_{t-1}. Consequently, restricting changes on the weights important for

previous tasks performance when learning new tasks will lead the optimization towards a set of parameters θ_t optimal for the new task while being close to θ_{t-1} and therefore minimizing forgetting.

Preventing important weights from previous tasks from drifting while allowing the others to move towards the new optimum, however, requires some metric on which to base the decision. The classical approach for this metric consists in assigning an importance value to each weight and prevent each weight's drift based on this importance value. Many different approaches [15], [29]–[32] have been proposed for computing this importance value, and most of them introduce a regularization term of the following form:

$$L_{reg}(\theta_t) = \frac{1}{2} \sum_{i=1}^{|\theta_{t-1}|} I_i(\theta_{t-1}^i - \theta_t^i)^2 \tag{1.7}$$

where t represents the current incremental task, θ_t^i the weight i of the network trained during the current incremental step, $|\theta_t|$ the total number of weights of the network, and I_i the importance value of the weight i for the previous task.

This regularization loss therefore adds a penalty to the loss for each weight of the model proportional to the importance for the past task and the amount of drift from the optimal value found during previous task θ_{t-1}^i. Thus, the more important the weight is for the previous task the less it will be allowed to drift during the optimization process.

Many approaches have been introduced to approximate I_i, the importance of each weight θ^i in the previous incremental task. [29], [30] proposed to view the CNN training from a probabilistic perspective and relate the importance value to the diagonal of the Fisher information matrix. The Fisher information measures the amount of information that a random variable X conveys about the parameters of the distribution that models X. The intuition behind this method was that all the information about previous tasks must be contained in the posterior distribution $p(\theta_{t-1}|\mathcal{D}_{t-1})$. Therefore, using a Laplace approximation allows to approximate this distribution as a Gaussian distribution centered on θ_{t-1} and thus to capture the amount of information each θ_{t-1}^i carries about the previous task with the Fisher information matrix.

In contrast to the above mentioned works, [15], [31] proposed to approximate each weight importance I_i by computing the amount each weight contributed to the reduction of the loss during each incremental step. This was done by keeping track of a running sum of gradients during the optimization process and obtained relatively similar results to the Fisher information approaches. Therefore in RWalk [15], Chaudhry et al. proposed

to combine both approaches and use the sum of the Fisher based importance and optimization based importance as the final weight importance values for their method and managed to outperform all other weight drift regularization methods.

In the RWalk study [15], the authors not only introduce a new weight drift regularization approach, but also put forward a set of metrics to evaluate the *forgetting* and the *intransigence* of incremental methods. These metrics are respectively linked to the plasticity and stability discussed in section 1.2.1. The results of their study demonstrate that weight drift regularization methods are effective in mitigating forgetting while allowing plasticity in the multi-head setting. In the more realistic single-head setting, however, they showed that these methods perform relatively well in handling forgetting but constraint the model's plasticity by preventing some weights to change when learning new incremental tasks.

Knowledge distillation methods

In order to maintain model plasticity while avoiding forgetting during incremental learning, a different type of regularization has been proposed, inspired by the knowledge distillation process used in the field of transfer learning.

The concept of knowledge distillation was first introduced by Bucila et al. [33] and subsequently generalized by Hinton et al. for use in transfer learning [34]. The aim was to compress the knowledge from an ensemble of neural networks, which are known to be cumbersome but perform better than a single model, into a simpler and more computationally efficient model. More generally, the objective was to be able to train a "teacher" model of high complexity to achieve best possible performance and then compress it into a smaller "student" model much easier to deploy and use.

In their work, Hinton et al. consider an ensemble of multiple convolutional neural networks as one big teacher network and distill the knowledge from this teacher network into a single student network by comparing the output probability distributions obtained on a transfer dataset. During the training of the student, images are fed to both the student and the teacher model, and a distillation loss term is added to the usual classification loss. This distillation loss will be minimal if both models produce the same probability distribution and will add a penalty depending on how diverging both distributions are.

Specifically, considering a classification problem with data about n classes, a teacher model Φ^T already trained for maximal performance on these classes and a student model Φ^S in which we want to compress the teacher model's knowledge. Both neural networks

have output logits $z_i, ..., z_n$ for all classes and the output class probabilities are obtained using the softmax activation function :

$$p_i = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}} \tag{1.8}$$

For the distillation loss, Hinton et al. showed that because of the teacher's expected high performance, given an image x with label y, the output probability distribution $\{p_i\}_{i=1}^{n}$ would be very similar to the target one-hot vector and therefore does not contain much information about the teacher knowledge. Instead, they proposed to compare softened probability distributions :

$$q_i = \frac{e^{z_i/\tau}}{\sum_{k=1}^{n} e^{z_k/\tau}} \tag{1.9}$$

with τ the temperature hyper-parameter used to soften output probability distributions. The final knowledge distillation loss is then the Kullback-Leibler divergence between both model's soften probability distributions :

$$\mathcal{L}_{KD} = KL(q^T||q^S) \tag{1.10}$$

where $KL(P||Q) = \sum_{x\in\mathcal{X}} P(x)log(\frac{P(x)}{Q(x)})$ is the Kullback-Leibler divergence, measures the difference or relative discrepancy between two probability distributions. Moreover, since the teacher model is frozen during training of the student, $q^T log(q^T)$ is a constant that is not relevant for the minimization of the loss and therefore the loss can be reduced to :

$$\begin{aligned} \mathcal{L}_{KD} &= \sum_{x\in\mathcal{X}} q^T(x)log(\frac{q^T(x)}{q^S(x)}) \\ \Leftrightarrow \mathcal{L}_{KD} &= \sum_{x\in\mathcal{X}} q^T(x)log(q^T(x)) - q^T(x)log(q^S(x)) \\ \Leftrightarrow \mathcal{L}_{KD} &= -\sum_{x\in\mathcal{X}} q^T(x)log(q^S(x)) \end{aligned} \tag{1.11}$$

This knowledge distillation loss is then added as a regularization term to the classical categorical cross-entropy classification loss (equation 1.3) when training the student model.

The use of knowledge distillation in incremental learning has been first proposed by Li et al. [11] in their algorithm LwF. They proposed to use the model from the previous incremental step Φ_θ^{t-1} as the teacher model when training the current model Φ_θ^t in order

to prevent forgetting. Indeed, from an incremental point of view, the model Φ_θ^{t-1} is an expert about the old classes therefore using it as a teacher transfers important knowledge about past classes into the model learning new ones and thus prevents forgetting.

Specifically, considering $\mathcal{C}_t = \mathcal{C}_{t-1} + \mathcal{C}_{new}$ classes, Φ_θ^{t-1} and Φ_θ^t respectively have $\mathcal{C}_{t-1}$ and $\mathcal{C}_t$ outputs, therefore in order to properly compare the same probability distributions only the $\mathcal{C}_{t-1}$ outputs of Φ_θ^t are considered for the knowledge distillation loss :

$$\mathcal{L}_{KD} = -\sum_{x \in \mathcal{X}} \sum_{c=1}^{\mathcal{C}_{t-1}} q_c^{t-1} log(q_c^t) \tag{1.12}$$

with $q_c^{t-1} = \frac{e^{z_c^{t-1}/\tau}}{\sum_{k=1}^{\mathcal{C}_{t-1}} e^{z_k^{t-1}/\tau}}$ and $q_c^t = \frac{e^{z_c^t/\tau}}{\sum_{k=1}^{\mathcal{C}_t} e^{z_k^t/\tau}}$ the softmax output probabilities of $\Phi_\theta^t(x)$ and $\Phi_\theta^{t-1}(x)$ softened by an hyperparameter τ.

Many knowledge distillation improvements have been proposed in the literature since its first application to incremental learning. First of all, based on the observation that the training loss is composed of one term for learning new classes and one for remembering old ones, Wu et al. [12] proposed to weight them using an adaptive weighting based on the number of old and new classes during each incremental step :

$$\mathcal{L} = \lambda \mathcal{L}_{KD} + (1 - \lambda)\mathcal{L}_{CE} \tag{1.13}$$

with $\mathcal{L}_{CE}$ the categorical cross entropy classification loss described in equation 1.3, and $\lambda = \frac{\mathcal{C}_{t-1}}{\mathcal{C}_t}$ the weighting coefficient for the distillation loss. Using this adaptive weighting favors learning new classes during the start of the incremental training, and then focus more on not forgetting old classes during later incremental steps when there are a lot more old classes than new ones. This adaptive weighting was shown in many works [12], [35] to allow the incremental model to reach a higher average incremental accuracy and therefore a better trade-off between plasticity and stability overall during the incremental training.

Several studies also examined the importance of the transfer dataset used for the distillation [10], [36] and showed that better knowledge will be distilled from the teacher if the dataset used for the distillation resembles the dataset on which the teacher was trained [36]. This result led researchers to start using knowledge distillation in combination with the rehearsal memory [10] described in section 1.3.1. Indeed, since the rehearsal memory is added to the incremental dataset during each incremental step, the final dataset used for the distillation contains some images on which the teacher model was trained, which

allows the distillation loss to capture better knowledge about past classes.

Following the same objective of extracting better knowledge from the teacher model, other improvements have been proposed based on the distillation loss itself. Instead of comparing models output probability vectors, some works proposed replacing the $\mathcal{L}_{KD}$ loss with other distillation terms based on more informative parts of the neural networks [16], [37]–[39]. Several works introduced different distillation losses comparing features extracted by both models [16], [37] and showed that it allowed the distillation loss to extract better knowledge from the teacher.

Finally, a third line of works rose along with the concepts of knowledge distillation and rehearsal memory to alleviate forgetting even more. As will be further explained in the next section, these works focus on the imbalance inherent to incremental datasets and attempt to reduce the classification bias that it causes in the last layer of the neural network.

1.3.3 Task recency bias correction

Another line of research tackles a different problem of incremental learning which is the problem of dealing with an imbalanced dataset. Indeed, when adapting the model with all the new classes data and either none or a few samples stored in rehearsal memory for old classes, the resulting dataset is heavily imbalanced in favor of new classes. This imbalance issue is indirectly related to catastrophic forgetting, as training on imbalanced datasets has been found to favor classes with the most available data [40].

In incremental learning, the bias towards new classes due to the imbalanced dataset is often referred to as a "new classes' bias". After each incremental step, the new classes become old classes for the next incremental step, resulting in a number of old classes increasing over time while the memory allocated to store their samples remains fixed. This issue leads to an incremental dataset that becomes more and more biased towards new classes which, when considering multiple incremental steps in succession, causes a "task recency" bias, where the learning process becomes more biased towards the most recently learned classes.

Class imbalance is a well-known issue in deep learning, as many datasets in computer vision [41], medical diagnosis [42], fraud detection [43], and other applications are typically imbalanced, and researchers have been actively exploring solutions to this problem [40]. In incremental learning, this imbalance has been shown in various studies [16], [18], [35] to lead to a prediction bias in the classification layer of the model. Specifically, in the final layer of the CNN, the weights corresponding to the new classes tend to have much larger norms than those corresponding to the old ones. This effect can be seen in Figure 1.6.

As can be seen in equation 1.14, the classification layer computes the dot product between the extracted feature vector and a weight vector for each class to obtain outputs that are then converted to probabilities using a softmax function as in equation 1.8:

$$p_i(x) = \frac{\exp(\theta_i^T \cdot \phi(x) + b_i)}{\sum_j \exp(\theta_j^T \cdot \phi(x) + b_j)} \tag{1.14}$$

with $p_i(x)$ being the probability of belonging to the class i for the image x, $\phi(x)$ the feature vector extracted by the CNN, θ_i and b_i the weights and biases of the classification layer. Therefore, the fact that these weights are larger means that overall the outputs, and therefore the activations, of the neural network will be stronger for new classes than for previous classes, effectively introducing a bias in the probability distributions created by the neural network leading to a prediction bias towards new classes.

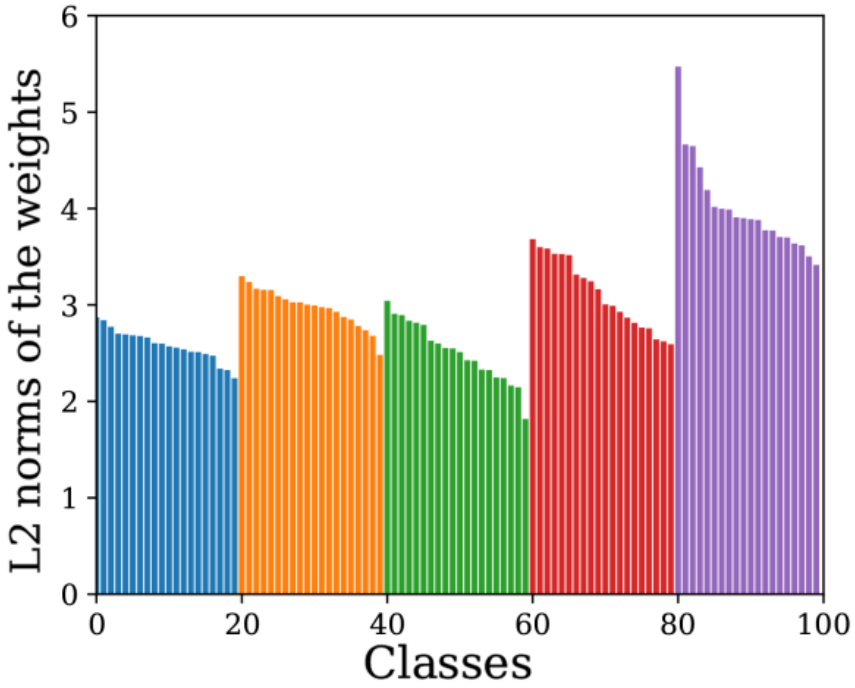


Figure 1.6 – Figure borrowed from the survey [18] showing class specific weights norm in the classification layer of the neural network after training on the CIFAR-100 dataset for 5 incremental steps of 20 classes/step. Weights have been sorted and colored by incremental training steps for better visibility.

In order to reduce this bias, two main lines of work are explored. The first one studies the use of different classifiers less prone to bias to avoid using the biased classification layer of the neural network at test time. The second one consists in using different methods to modify the learnt weights after each incremental steps in order to alleviate this bias without losing the discriminative information learnt.

Classifier changes

This approach was first introduced by Rebuffi et al. in their algorithm iCarL [10] where they proposed to discard the biased classification layer at test time and instead use a non parametric classification scheme that they call Nearest exemplar mean (NEM) classification. They make use of a rehearsal memory during incremental steps and propose to compute at the end of each incremental step the mean feature vector μ_c of all trainset exemplars for each class c. Then they suggest to classify each test image based on the closest mean feature vector:

$$y* = \underset{c=1,\dots,\mathcal{C}^t}{\operatorname{argmin}} ||\phi(x) - \mu_c|| \tag{1.15}$$

This classification process directly classify samples x based on feature vectors $\phi(x)$ and therefore does not use the biased classification layer of the model. Many approaches demonstrated its efficacy in incremental learning despite its simplicity [10], [44], however, this classifier is not completely unbiased as it computes mean feature vectors of each class based on all samples available in the incremental training dataset and therefore makes a better approximation of the mean for new classes than old ones.

Some other approaches have also been proposed to normalize the classification layer to reach the higher performance that the parameterized classifiers offers while retaining the unbiasedness of the NEM classifier [16], [39]. These approaches considered normalizing the last layer of the model with a cosine normalization [45] which normalizes both the feature vectors and the weight vectors before computing the dot product :

$$p_i(x) = \frac{\exp[\eta(\overline{\theta}_i^\top \cdot \overline{\phi}(x))]}{\sum_j \exp[\eta(\overline{\theta}_j^\top \cdot \overline{\phi}(x))]} \tag{1.16}$$

where $\overline{v} = \frac{v}{||v||}$ represents L_2-normalized vectors, and η is an added learnable parameter to control the peakedness of the softmax distribution that is necessary because both θ_i and $\phi(x)$ have unit norm, therefore the range of $\theta_i^\top \cdot \phi(x)$ is restricted to $[-1, 1]$. The comparison between NEM and this cosine similarity done in [16] revealed that this classifier was not only effective in reducing the task recency bias in the classification layer but also superior to NEM in terms of performances.

Post training correction

An alternative approach that has been proposed to reduce the task recency bias issue is to add a post training bias correction step after each incremental task. To this end, in End-to-End Incremental Learning (EtEIL) [17] Castro et al. proposed to add a small finetuning step on a balanced subset of the available data at the end of each incremental step. Essentially, at the end incremental steps they proposed to undersample the new classes data and use this subset of new data together with the rehearsal memory to create a balanced dataset that they then used to finetune only the classification layer with a small learning rate. While this approach adds another training step to each incremental steps it has been shown to be quite effective in removing bias and is still being used in

very recent incremental studies [46].

Another simple and very similar approach to prevent the task recency bias was proposed in [12] by Wu et al. in the algorithm they called BiC (incremental Bias Correction). In this algorithm designed specifically for large scale datasets, they propose to add another small training step similarly to EtEIL, but instead of retraining the entire classification layer, consider only learning parameters of a linear transformation of the output logits to compensate for the task recency bias. Indeed, during incremental steps 10% of the data is transferred into a small validation set that is then used during the bias correction step to train α and β that constitute the parameters of the bias correction layer :

$$q_k = \begin{cases} o_k & 1 \leq k \leq \mathcal{C}_{t-1} \\ \alpha o_k + \beta & \mathcal{C}_{t-1} + 1 \leq k \leq \mathcal{C}_t \end{cases} \tag{1.17}$$

where o_k represents the output logit for the class k obtained after the dot product of the classification layer and before the softmax activation, C_{t-1} is the number of old classes, and C_t the number of new classes. This simple linear rescaling of new classes output logits has shown to be very effective in reducing the classification bias [12] while being faster and easier to optimize than the EtEIL method due to the reduced number of parameters trained.

Different approaches have been proposed since then to remove the added training step needed for this bias correction step [13], [14], [35]. The most noticeable improvement has been the weight alignment proposed by Zhao et al. [35], they consider not just rescaling the output logits to make new classes outputs and previous classes outputs comparable but instead rescale directly the weights of the classification layer themselves. Specifically, with a classifier $\mathcal{H}_t$ containing weights vectors W_c for $c \in [1, \mathcal{C}_t]$ with $\mathcal{C}_{old}$ old classes:

$$\mathcal{H}_t = \{W_1, \ldots, W_{\mathcal{C}_t}\} = (W_{old}, W_{new})$$

$$W_{old} = \{W_1, \ldots, W_{\mathcal{C}_{old}}\}$$

$$W_{new} = \{W_{\mathcal{C}_{old}+1}, \ldots, W_{\mathcal{C}_t}\}$$

They compute the norm of the weights vectors of each class $\mathcal{N}_c$ in the classifier and rescale the weights of new classes based on the mean norm of old $\overline{\mathcal{N}_{old}}$ and new classes $\overline{\mathcal{N}_{new}}$:

$$\begin{aligned}
\mathcal{N} &= \{\mathcal{N}_1, \ldots, \mathcal{N}_{\mathcal{C}_t}\} = \{||W_1||, \ldots, ||W_{\mathcal{C}_t}||\} \\
\mathcal{N}_{old} &= \{\mathcal{N}_1, \ldots, \mathcal{N}_{\mathcal{C}_{old}}\} \\
\mathcal{N}_{new} &= \{\mathcal{N}_{\mathcal{C}_{old}+1}, \ldots, \mathcal{N}_{\mathcal{C}_t}\} \\
\gamma &= \frac{\overline{\mathcal{N}_{old}}}{\overline{\mathcal{N}_{new}}}
\end{aligned}$$

Finally, the weights of new classes are rescaled with this γ parameter :

$$\widehat{W_{new}} = \gamma \cdot W_{new} \tag{1.18}$$

Modifying the weights of all new classes with the same rescaling parameter γ ensures the relative magnitude of new classes weights and therefore the discriminative information between each new class is retained. Furthermore, this rescaling aligns the norms of old and new classes weights which removes the bias due to the new classes weights being larger (figure 1.6). Despite its simplicity this approach was shown to outperform both EtEIL [17] and BiC [12] bias correction methods while also requiring much less computation overhead [35].

1.4 Conclusion

In this chapter we first provided the necessary background information for understanding the operation of convolutional neural networks and how they are used in the context of class-incremental learning. Following that, we detailed the most notable components that recent state-of-the-art algorithms use to alleviate catastrophic forgetting in class-incremental learning. We showed that most of the algorithms effective in alleviating catastrophic forgetting rely on three main components : data rehearsal, training loss regularization, and task recency bias correction. Many methods have been proposed over the years to improve various aspects of these components, but the most notable approaches could be summarized in the algorithm 3 used as a baseline for our work.

In this algorithm that we also visually illustrate in figure 1.7, a simple rehearsal memory with herding sampling is employed to store and replay the most representative exemplars of previous classes during each incremental steps. Furthermore, a knowledge distillation term is added to the training loss to prevent forgetting and the weight alignment approach introduced in equation 1.18 is used at the end of each incremental step to remove the bias from the classifier. This algorithm that we used as a baseline for our works represents a simplified version of the one described in [35] which was the state-of-the-art algorithm obtaining the best performances in 2020 when we first started our thesis study.

Algorithm 3 Pseudo-code of the baseline incremental method

Require: T the number of incremental steps
Require: Φ_θ^0 model composed of a feature extractor $\phi(\cdot)$ and a classifier $\mathcal{H}(\cdot)$
  $\mathcal{M}_0 \leftarrow \emptyset$ ▷ initialise rehearsal memory
  for t=1,...,T **do**
    Require: $\mathcal{D}_{new}$ the dataset containing the new data
    $\Phi_\theta^t \leftarrow \Phi_\theta^{t-1}$ ▷ Initialize the new model without the previous model
    $\mathcal{D}_t \leftarrow \{\mathcal{M}_{t-1}, \mathcal{D}_{new}\}$ ▷ add the memory to the new dataset
    $\mathcal{L} \leftarrow \lambda\mathcal{L}_{KD} + (1-\lambda)\mathcal{L}_{CE}$ ▷ combine standard cross entropy and knowledge distillation with adaptive weighting
    $\Phi_\theta^t \leftarrow$SGD$(\mathcal{L}, \Phi_\theta^t, \Phi_\theta^{t-1})$ ▷ use the SGD algorithm to train the model (see algorithm 1)
    $\mathcal{M}_t \leftarrow$Herding$(\mathcal{D}_{new}, \phi)$ ▷ use the herding sampling strategy to update the rehearsal memory
    $\mathcal{H} \leftarrow$Wa$(\mathcal{H})$ ▷ remove bias from the classifier with the weight alignment method from section 1.3.3
  end for

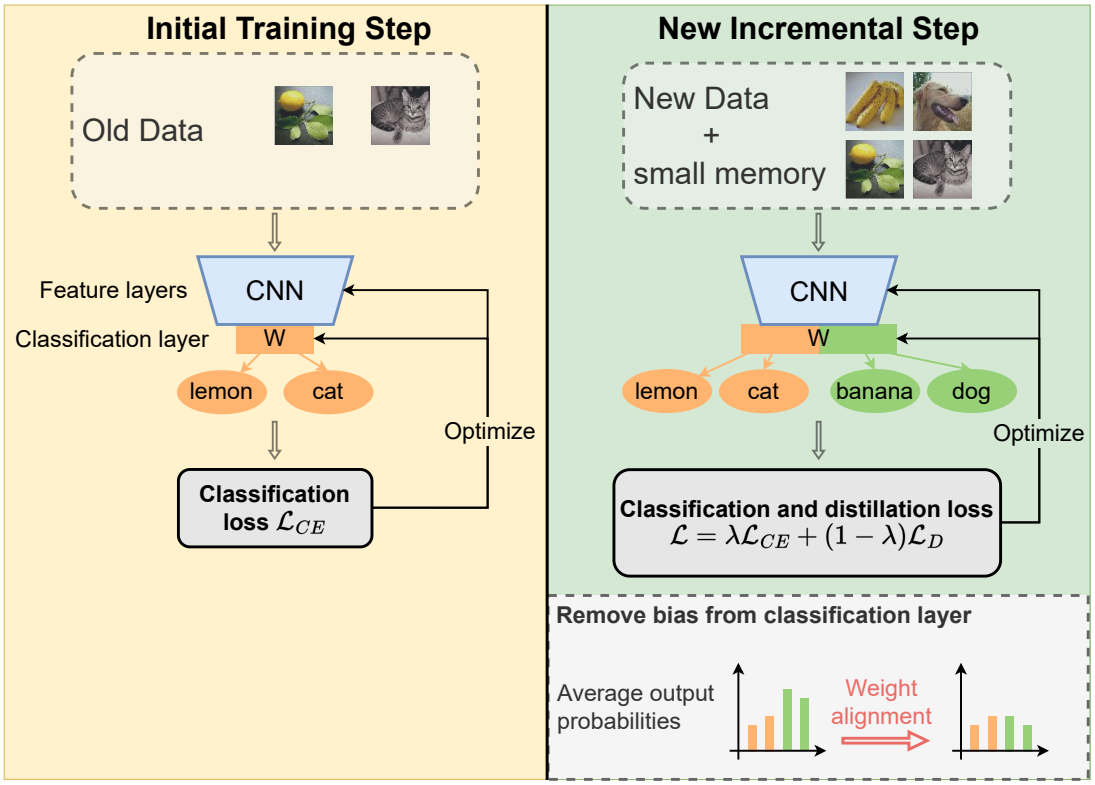


Figure 1.7 – Illustration of the baseline incremental algorithm used in our study. In the initial training step only the classification loss is used to train the model while the following training step adds the rehearsal memory to the dataset and a knowledge distillation term to the training loss. The weight alignment method is also further used after each incremental step to remove the bias from the classification layer.

Chapter 2

Experimentation on the open challenges faced by incremental methods

In this chapter we will present the different problems directly related to catastrophic forgetting that we managed to observe during the first year of the thesis. Neural networks are often considered as a black box, they are composed of many neurons that are linked together with different weights which, if set properly, allow neural networks to model any function. As explained in the previous chapter, the general outcome of a neural network can be controlled by training it with a specific loss function, however explaining how it operates inside of this black box is not trivial. For example, for our specific subject, catastrophic forgetting can be observed easily in the output logits and probabilities of networks trained incrementally, however explaining how it happens, which neuron is forgetting or which part of the network contains information about specific classes is very challenging.

In order to comprehend the mechanisms behind catastrophic forgetting, we started by analysing separately different state-of-the-art components of incremental algorithms. In section 2.1 we start by describing the general setup used for all the experiments that we will describe in this chapter. We then presents in section 2.2 the visualisation of the feature space and experiments done on the feature extractor part of the neural network. Finally, in sections 2.3 and 2.4 we studied the rehearsal memory and distillation components used by state-of-the-art incremental methods and experimented with ways of improving them.

The experiments and results presented in this chapter were conducted on a singular dataset and backbone and may not translate totally to other incremental settings. The idea was to explore many different directions of research and choose one particularly interesting to study in details during the remainder of the thesis instead of doing a thorough analysis of a specific component on different incremental settings and datasets. This is the reason why these results were not published but are still presented here as an integral part of

the research done during this thesis.

2.1 General experimental setup

In this section we will describe the general experimental setup used for the experiments described in this chapter. Many experiments were realized to understand how the different components of successful incremental algorithms effectively alleviate forgetting and find ways to improve them. These experiments were done on the algorithm 3 that makes use of the 3 state-of-the-art components of incremental methods to alleviate forgetting, namely the rehearsal memory, knowledge distillation, and task recency bias correction (see section 1.3.3).

More specifically, for the rehearsal memory our algorithm samples representative exemplars with the herding sampling approach explained in section 1.3.1 and we further compare its impact to other sampling methods in section 2.3. For the knowledge distillation component we implemented the standard knowledge distillation loss $\mathcal{L}_{KD}$ (equation 1.12), combined it to the classification loss using adaptive weighting as in equation 1.13, and compared it to a feature-based distillation method that we describe in the section 2.4. Finally for the last component, the task recency bias correction, all the state-of-the-art approaches have effects only on the classification layer of the model (see section 1.3.3). Since the focus of this thesis is on the features extracted during incremental learning, we implemented the simplest approach called Weight alignment (equation 1.18) that also attains the highest performances according to Zhao et al. [35]. Instead of experimenting with the bias correction at a classifier level, however, we visualize and analyse thoroughly the internal representation of the model trained incrementally in section 2.2 in order to observe if a similar bias appears in the features of the model.

Furthermore, following standard incremental learning practice [10], [12], [35], our experiments were carried out on the CIFAR-100 dataset and with the 32-layer ResNet [8] backbone neural network architecture. This dataset comprises 500 training and 100 validation images per class, encompassing a total of 100 classes. The images are in RGB format and measure 32 by 32 pixels, depicting diverse objects like boats, fruits, furniture, alongside animals such as fishes, insects, and mammals. The choice of this dataset was motivated by its relatively low amount of data with a high number of varied classes that allowed fast incremental trainings adding multiple classes incrementally up to the final 100 classes contained in the dataset. For all of the experiments, following the most stan-

dard procedure [10], [12], [35], this dataset was separated in 10 incremental training steps adding all the available training data for 10 classes during each step.

Finally, the training hyper-parameters used were determined experimentally to maximize performances. Specifically, we used the SGD optimizer with a momentum of 0.9, a weight decay of 0.0005 and use a batch size of 128. We trained our models for 200 epochs, with a learning rate starting at 0.1 and divided by ten after 100, 150, and 180 epochs. The data augmentation applied to training images consists in random cropping, horizontal flip and normalization. Following standard practice [10], [12], [35], the temperature parameter τ was set to 2 in $\mathcal{L}_{KD}$ (see equation 1.12) and the total size of the rehearsal memory was set to 2000 images. The herding sampling strategy was used to choose samples to store in memory and the memory was filled entirely during each incremental step.

2.2 Analysis of the feature space

While catastrophic forgetting has been shown multiple times [18], [35] to affect the classifier part of the model during incremental learning by introducing a classification bias towards new classes, few studies analyse the features extracted by incremental models and the effect of catastrophic forgetting on them.

In this section, we will present the methods used during this thesis to visualize the features used by the last layer, the classification layer, to categorize all classes. These features correspond to the outputs of the penultimate layer, however, visualizing them is difficult because there are too many of them. In fact, in incremental learning, the neural networks generally used in most studies [10], [12], [35] are the 32 and 18 layers ResNet networks [8], that make use respectively of 64 and 128 features.

In order to analyse what is happening in this highly dimensional space during incremental learning we first considered specifically the clusters formed by each class. Many incremental studies assume that these clusters are Gaussian, for example the herding strategy explained in section 1.3.1 considers that representative samples that should be stored in memory are the ones whose feature vector is closest to the mean, therefore implying that clusters are Gaussian and that the mean represents the cluster well.

Due to the high dimensionality of the feature space, proving that these clusters are Gaussian is not possible. We first considered different state-of-the-art statistical tests such as Shapiro-Wilk, Kolmogorov-Smirnov, Pearson, and skewness and kurtosis analysis but quickly realized that the high number of dimensions made it impossible to conclude

anything from the results. Instead of proving that the clusters were Gaussians, we then considered visualizing specific properties of these clusters using normalized features extracted from all training images. Specifically, after each incremental step, the normalized features are computed with :

$$f = \{\frac{\phi(x)}{||\phi(x)||}, \forall x \in \mathcal{D}_{tot}\} \tag{2.1}$$

with $\mathcal{D}_{tot}$ the dataset containing all the data about all seen classes and ϕ the feature extractor trained during the incremental step. These features were then used for the visualization of specific properties of the clusters, namely, the representativity of the mean, and the intra and inter-class distances and a general two dimension visualisation of their evolution during incremental steps.

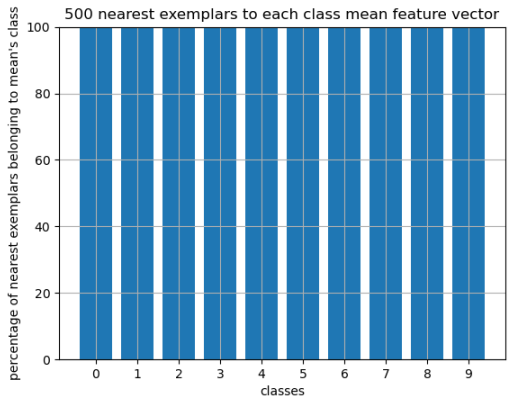
2.2.1 Mean representativity visualisation

For incremental learning, one of the most interesting aspect of the feature space corresponds to the representativity of each cluster's mean. In fact, this mean feature vector for each class is used in many incremental methods, for example to choose the best samples to store in memory or even to classify with the NME classification scheme explained in section 1.3.3. All these methods assume that the clusters of the feature space are Gaussian-like and that most of the samples from each class lie close to their mean, but this is not necessarily the case. In order to assess how representative of the class the mean is for each cluster, we propose a visualization based on the k-nearest neighbors algorithm.

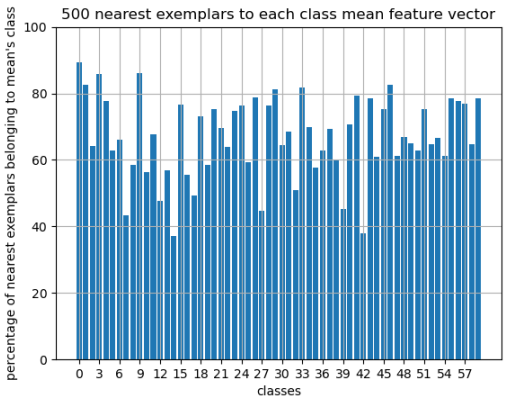
Specifically, after each incremental training step we considered all seen classes' mean feature vectors, used the k-nearest neighbor algorithm to classify the K nearest neighbors of each mean and measured the accuracy for each class mean. The CIFAR-100 dataset used for our experiments contains 500 training images for each class, therefore this K number was set to 500 in order to visualize this accuracy as a percentage of all the training data.

Using the general experimental setup presented in section 2.1 we visualize the means representativity at the start, middle and end of the incremental training and compare them to those of a non incremental training of all classes in figure 2.1. This non incremental training was a simple training done on the same dataset, with the same model and hyperparameters but without incremental steps, training all the classes at once using all the available data for each class.

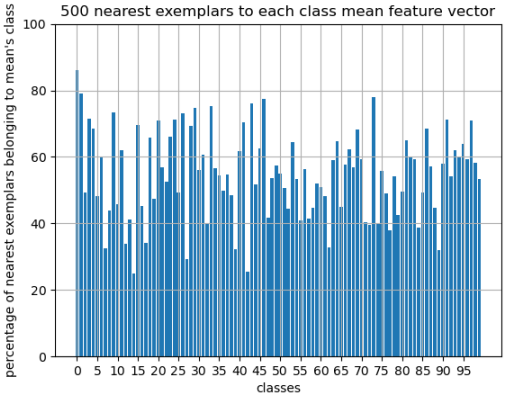
First of all, looking at figure 2.1d, we can see that for a classical training of 100 classes



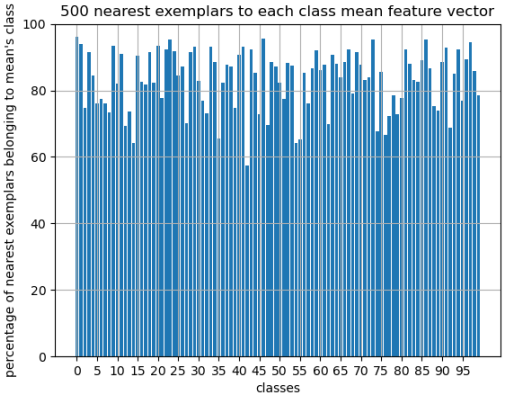
(a) mean representativity percentage after the first incremental step.



(b) mean representativity percentage after the fifth incremental step.



(c) mean representativity percentage after the last incremental step.



(d) mean representativity percentage of a non incremental training.

Figure 2.1 – Visualisation of each class mean representativity during incremental learning and non incremental learning. For each class mean feature vector, the 500 nearest neighbors were used to visualise the representativity of the mean as a percentage of good classification of those neighbors.

around 70-80% of the neighbors of each class mean feature vectors belong to the same class. This result is reasonable, as the CIFAR-100 dataset is considered a relatively easy dataset for standard deep learning techniques, and is therefore well clustered when the model is not trained incrementally.

Furthermore we can see in figure 2.1a that at the start of the incremental training process the means are even more representative of the clusters, with 99-100% for all classes. This is due to the fact that only 10 classes are trained during the first incremental step which combined with the relative simplicity of the dataset leads to training accuracies of almost 100% and to a feature space where all training data points are well clustered

around their mean. After some incremental steps however, catastrophic forgetting happens and we can see in figures 2.1b and 2.1c that the representativity of the means severely drops to around 40% in average at the end of the incremental training. This demonstrates the fact that the loss of information due to catastrophic forgetting impacts the feature space and makes data points that were initially close to their mean feature vector drift to be closer to other classes' mean feature. This could be caused by two problems, either clusters becoming less and less separated to the point where they end up on top of each other, or clusters becoming more and more spread out, or both at the same time.

In order to answer this question, we describe in the next section another experiment realized during this thesis to visualize the clusters inter-class and intra-class distances during training of an incremental model.

2.2.2 Inter-class distances and intra-class variance

The preceding visualisation established that catastrophic forgetting impacts negatively the feature space by making each clusters' mean value less and less informative of the global class features. The reason why that happens, however, was not fully answered, therefore another experiment was done to analyse these clusters in more details and characterize them better. Ideally, the clusters should be as separated and compact as possible in order for the classification head to be able to correctly classify each class easily. We therefore propose to estimate and visualize both the inter-class and intra-class distances between and within clusters the following way :

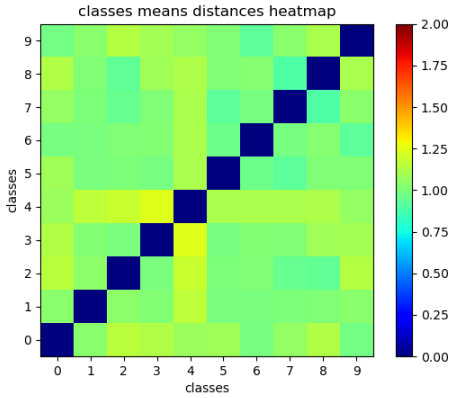
$$D_{inter}(C_i, C_j) = ||\mathbb{E}[f_{C_i}] - \mathbb{E}[f_{C_j}]|| \tag{2.2}$$

$$D_{intra}(C_i) = \mathbb{E}[(f_{C_i} - \mathbb{E}[f_{C_i}])(f_{C_i} - \mathbb{E}[f_{C_i}])^T]$$

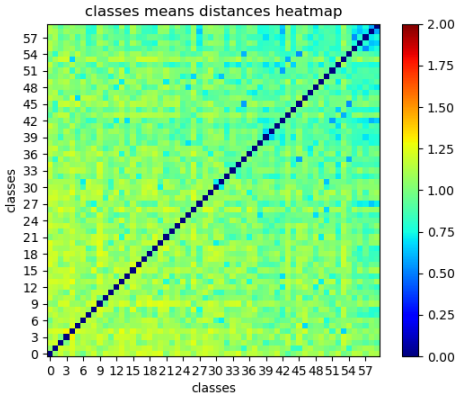
$$D_{intra}(C_i) = \mathrm{Var}[f_{C_i}] \tag{2.3}$$

where f_{C_i} represents all the extracted features f of a specific class C_i, D_{inter} the estimator of inter-class distances, and D_{intra} the estimator of intra-class distances. Essentially, D_{inter} corresponds to the Euclidean distance between two cluster mean feature vector, and is bounded by 2 because the feature vectors are normalized (see eq. 2.1).

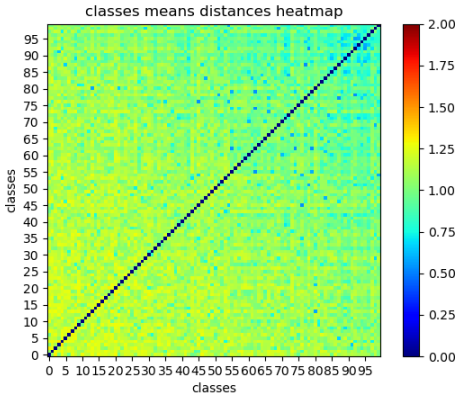
Figures 2.2 and 2.3 show the results obtained respectively for the inter-class and intra-class distances. In a similar fashion than the last experiment on mean representativity, a comparison is done between these distances within the feature space obtained with a non incremental training, and after the first, fifth and last step of an incremental training



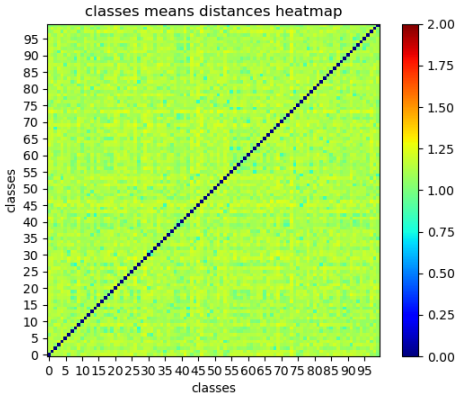
(a) Heatmap of inter-class distances after the first incremental step.



(b) Heatmap of inter-class distances after the fifth incremental step.



(c) Heatmap of inter-class distances after the last incremental step.



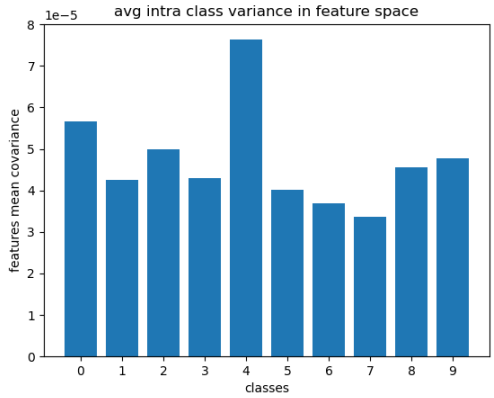
(d) Heatmap of inter-class distances of a non incremental training.

Figure 2.2 – Heatmap representations of inter-class distances computed with the euclidean distance between each cluster's mean feature vector. As the feature vectors are normalized to have a unit norm, the heatmap is bounded by 2.

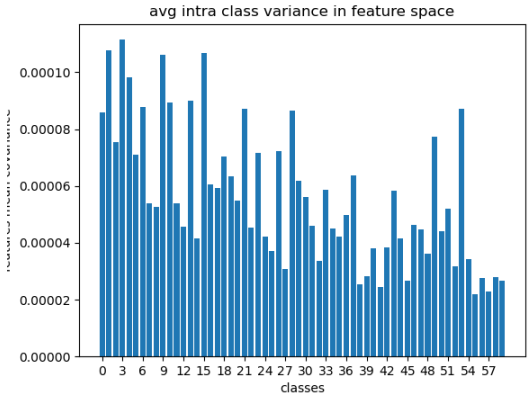
using the general setup described in section 2.1.

Looking at figure 2.2d, we can see that when trained in a traditional manner the model extracts features that do not maximize cluster separation yet separates evenly every cluster. Moreover, figure 2.3d establishes that the clusters formed have very low variance of around $3e^{-6}$, which illustrates the ideal feature space that contains compact clusters evenly separated.

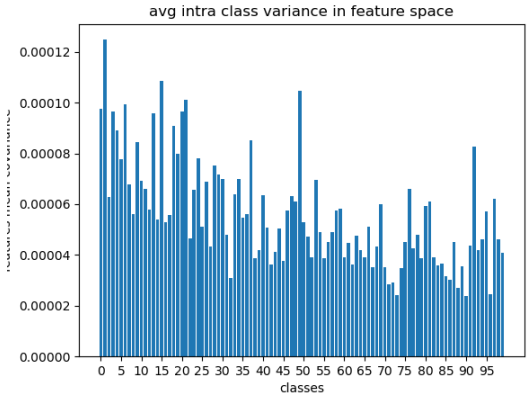
During incremental learning, however, as can be seen in figures 2.2a, 2.2b, and 2.2c, the clusters are not evenly separated at all. In fact, interestingly the less separated clusters are the new classes clusters while the past classes' clusters seem well separated. This is



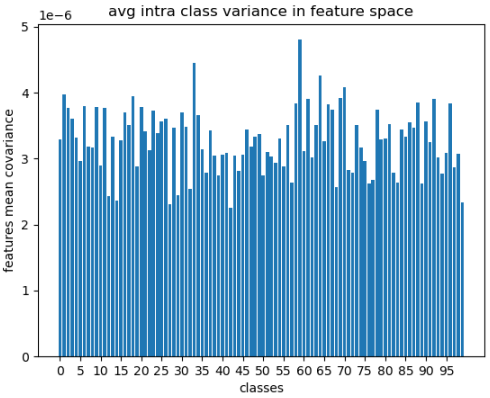
(a) Histogram of class feature variances after the first incremental step.



(b) Histogram of class feature variances after the fifth incremental step.



(c) Histogram of class feature variances after the last incremental step.



(d) Histogram of class feature variances of a non incremental training.

Figure 2.3 – Histogram visualisation of clusters' variance evolution over the course of the incremental training.

interesting because catastrophic forgetting is supposed to have a negative impact on past classes but not on new ones.

This result combined with the higher variances of past classes that can be observed in figures 2.3a, 2.3b, and 2.3c illustrates that catastrophic does not make past classes' clusters less separated, but rather more and more spread out. In fact, we believe that rehearsal learning and knowledge distillation are responsible for the high separation of cluster mean feature vectors, however the lack of variability of training data for past classes induces a loss of information at the edges of the clusters, leading to clusters very spread out at the end of the incremental training.

Moreover, in figure 2.3 while the variance of new classes is lower than the one of past classes, it is still around $3e^{-5}$ which is 10 times higher than the variance obtained with the non incremental training. This demonstrates that catastrophic forgetting also makes it more difficult to separate and compact new classes clusters, which is probably caused by the loss of information on past classes.

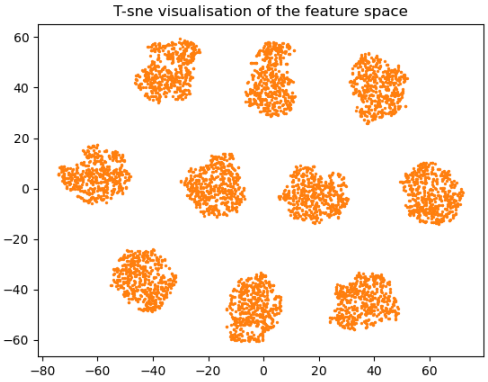
In order to validate these results and visualize better the evolution of the global structure of the feature space during an incremental training, we further present in the following section the visualizations obtained using a popular dimensionality reduction method called T-SNE.

2.2.3 T-SNE visualisation

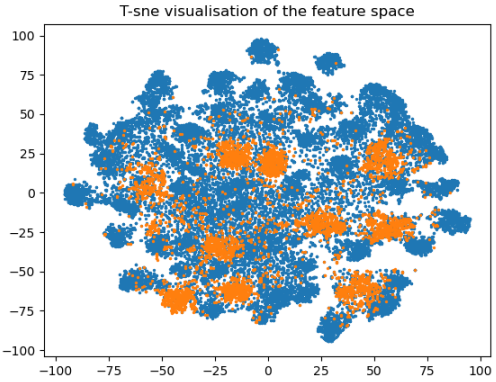
Finally, in order to better visualize the global structure of the feature space and observe which clusters are closer than others we used the t-Distributed Stochastic Neighbor Embedding (T-SNE) technique[47]. This T-SNE is a dimensionality reduction method that allows to visualize high-dimensional data in lower dimensions. Contrarily to traditional dimensionality reduction methods like the Principal Component Analysis (PCA), however, the T-SNE is non deterministic and gives different results with different random initialization. This leads to 2D visualisations that cannot be used to obtain quantitative results on the structure of the feature space, nevertheless, the T-SNE focuses on retaining the similarities between samples which experimentally led to better qualitative results than using a PCA.

In a similar fashion than previous visualisation experiments, we run both the general incremental training algorithm and a non incremental training with the same parameters on the CIFAR-100 dataset. After each incremental training step, we then extract f the set of features obtained with all the available data about the classes learned and compute the 2D T-SNE visualisation of the feature space. Figure 2.4 shows the visualisation obtained with the incremental method after the first, the fifth, and the last incremental step and compares it to the one obtained with a non incremental training. In the T-SNE computed during the incremental training, the new classes' clusters are shown in orange whereas the old classes's clusters are in blue.

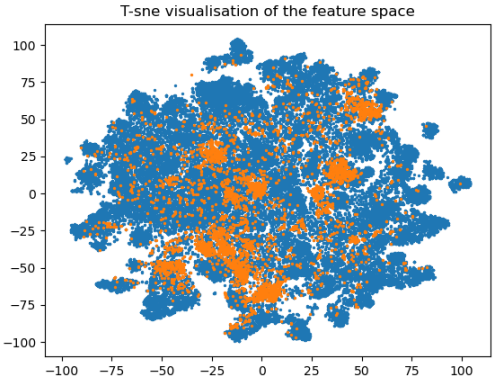
As can be seen in this figure, after the first incremental training step of 10 classes, each data point seem to belong to a well defined cluster and all 10 clusters seem clearly separated from the others. In the following incremental steps, however, we can see that the feature space progressively becomes more and more cluttered, with clusters less separated



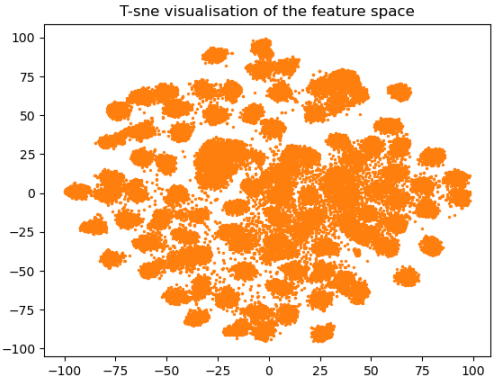
(a) T-SNE of the feature space after the first incremental step.



(b) T-SNE of the feature space after the fifth incremental step.



(c) T-SNE of the feature space after the last incremental step.



(d) T-SNE of the feature space obtained with non incremental learning.

Figure 2.4 – Visualisation of the general structure of the feature space during incremental learning using a T-SNE. Samples from new classes are shown in orange while samples from previous classes are in blue.

and some points a lot more scattered.

This loss of information becomes even more apparent when comparing the last incremental T-SNE to the non incremental one where we can see for the same number of classes that the clusters are much more separated and recognizable. Only a few samples seem a bit scattered, while when trained incrementally almost the entirety of the points look very dispersed and the clusters are difficult to notice, which demonstrate the effect of catastrophic forgetting on the features.

Interestingly, when looking at the orange data points representing the new classes learned, we can see that the loss of information happens also on new classes clusters for

which all the available data was used during the last training step. We believe this is due to the loss of information about past classes that makes it harder to separate new classes from past classes and hinders clustering of new classes.

2.2.4 Conclusion

In this section we described the different experiments and visualisations done during this thesis to analyse the evolution of the feature space during incremental learning. We first proposed a visualization of the clusters mean representativity, we then further characterized the feature space by visualizing the distances within and between clusters and finally used a dimensionality reduction method to further visualize the global structure of the feature space qualitatively.

We compared and visualized the feature space of an incremental model to the one of a model trained in a conventional manner. The experiments done demonstrated a general loss of information over the incremental training steps. The initially well clustered feature space becomes more and more cluttered because clusters become very spread out and therefore not very separated from each other, which leads to sub-optimal performances of the classifier.

Moreover, this general loss of information due to catastrophic forgetting was shown

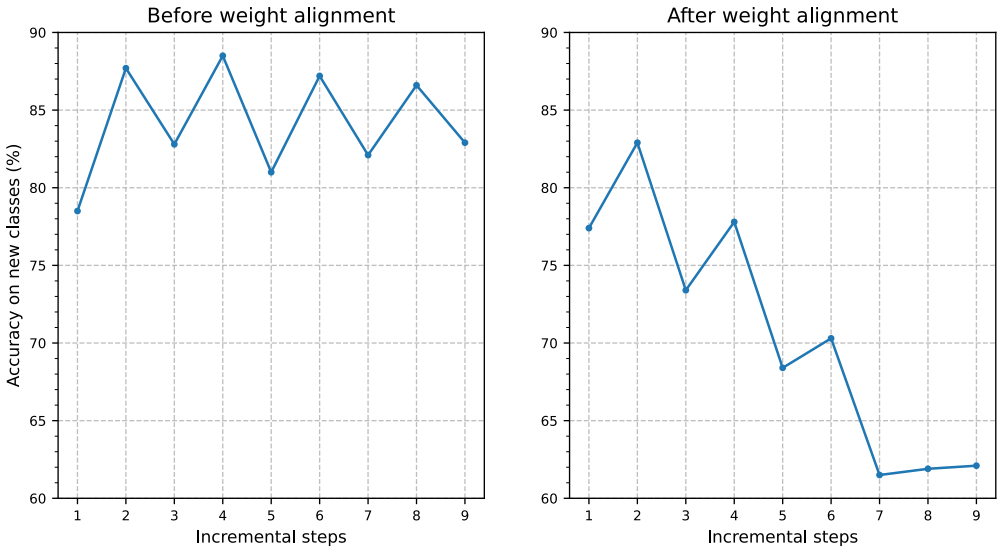


Figure 2.5 – Accuracy obtained on the new classes before and after weight alignment during each incremental step.

to not only impact past classes but also new classes. Since all data is available at train time for the new classes, we believe this effect is caused by the loss of information about past classes that clutters the feature space and prevents good clustering of new classes. Interestingly, as shown in figure 2.5, this impact on new classes is not observed on the classifier outputs during training but appears clearly after the bias is removed from the classifier with the weight alignment part of the algorithm.

2.3 Exemplar selection based on features

The next component used by most incremental methods that we studied is the rehearsal memory. This memory was introduced first in the paper iCarL [10] and allowed its authors to surpass the previous state-of-the-art performances (attained by the method LwF [11]) by over 20% of accuracy on the CIFAR-100 dataset at the end of 10 incremental steps with 10 classes per step. This huge performance improvement was obtained using the herding sampling strategy described in section 1.3.1 to choose adequate samples to store in memory after each incremental step, however since then very few works [15], [18] studied other exemplar selection strategies.

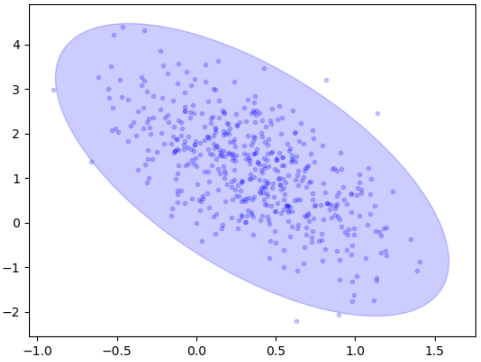
As explained in section 1.3.1 the herding sampling strategy is based on the mean feature vector of the class and performs better than other opposite strategies that focus on boundaries between class. This makes sense because the memory size remains fixed which means eventually only 1 sample per class will be stored in memory and the mean of the class is better suited in this case. However we believe that when more samples per class are available, sampling only exemplars close to the mean is suboptimal. In order to verify this hypothesis, we experimented with other sampling strategies that would store the closest sample to the mean but also allow more varied exemplars to be stored when more space is available.

Specifically, we compared the following 4 different sampling methods :

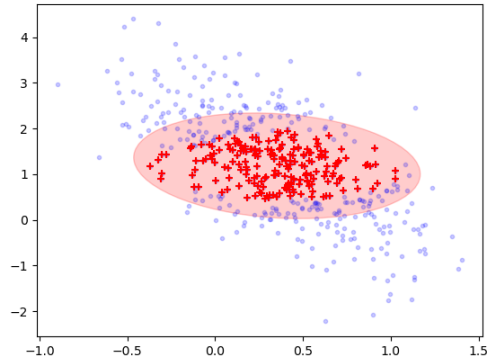
Herding sampling. The standard state-of-the-art method sampling mostly exemplars close to the class mean.

Random sampling. The most basic sampling strategy that samples exemplars uniformly at random.

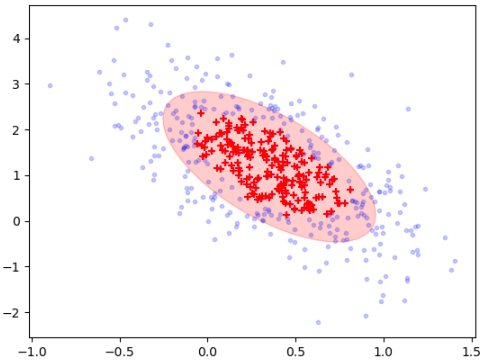
Mahalanobis herding sampling. A variation of the herding strategy that uses the mahalanobis distance instead of the euclidean distance for sampling. This has the



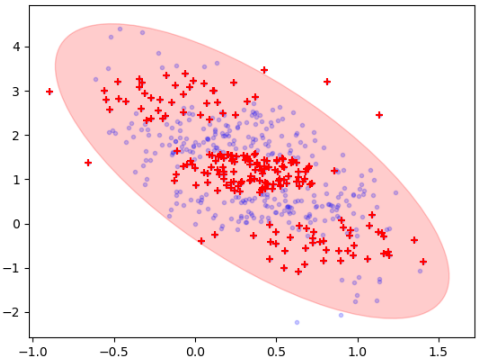
(a) Original data following a random 2d gaussian distribution



(b) Subset obtained with herding sampling



(c) Subset obtained with mahalanobis herding



(d) Subset obtained with variance herding

Figure 2.6 – Visualisation of different sampling methods on artificial data following a 2d gaussian distribution. Original data points are represented in blue and sampled ones in red, confidence ellipses show the area included within 3 standard deviation of the clusters' means.

effect of accounting for the cluster orientation when choosing closest exemplars to the mean.

Variance herding sampling. A variation of the herding strategy that samples exemplars so that the subset of choosen exemplars best represent the mean and variance of the class instead of only the mean as the usual herding strategy does.

Figure 2.6 shows the effect of each variant of the herding sampling strategy proposed on artificial data coming from a 2D gaussian distribution. 500 data points where randomly generated and a subset of 200 data points was sampled with each herding sampling variant

methods to properly analyse their effects.

As can be seen in that figure, the herding method samples a subset of exemplars that are close to the mean therefore the cluster around the data sampled does not preserve the variance of the original data. The mahalanobis herding method on the other hand, preserves the orientation and the mean of the cluster but does not maintain the original variance of the data. Finally, the variance herding method samples some points close to the mean and some at the edges of the cluster to conserve also the overall variation of the original data.

2.3.1 Experiment

Like the preceding experiments outlined in this chapter, the exemplar selection experiments were conducted using the CIFAR-100 dataset, separated in 10 incremental steps of 10 classes each. The employed setup involved the ResNet-32 model and a rehearsal memory containing 2000 samples. We executed a conventional incremental algorithm that incorporated rehearsal memory, knowledge distillation, and utilized the WA bias removal technique. In these experiments, we substituted the herding sampling strategy used in the rehearsal memory with each of the four methods detailed above and compared the incremental accuracies obtained with the same incremental class ordering.

	Accuracies	
	Last	Average
Random	50.06	64.75
Herding	50.49	64.49
Mahalanobis herding	50.19	64.71
Variance herding	50.40	64.4

Table 2.1 – Accuracies obtained on the dataset CIFAR-100 separated in 10 incremental steps with four different sampling methods. The accuracy obtained at the end of the last incremental step and in average over all 10 incremental steps are reported. Each accuracy reported represents the average of 3 different random class orders but with the same random seed for each method.

Specifically, table 2.1 shows the average incremental accuracy and the accuracy obtained after the last incremental step with the different samplers. As can be seen in this table none of the methods notably enhance the performance. This suggests that, while the rehearsal memory effectively mitigates the issue of catastrophic forgetting, the impact of the sampling method employed is rather limited. This outcome could be attributed

to two distinct factors. The first factor is the fact that stochastic data augmentations are applied during training, which randomly alter each selected image to enhance the training data variability. As a result, the need for selecting optimal exemplars becomes less crucial. The second factor mitigating the significance of the sampling method is that with each incremental step, the underlying features change to accommodate new classes. Consequently, samples positioned close to the mean feature before an incremental step might undergo drastic transformations, ending up significantly distant from it after the incremental step has been taken.

2.4 Feature distillation

Knowledge distillation has been shown in many incremental study to be very effective in alleviating catastrophic forgetting [10]–[12], [17], [35]. As explained in detail in section 1.3.2, the knowledge distillation loss operates by forcing the softened output past classes probabilities of the model to stay similar to those of the model from the previous incremental step.

In transfer learning where the concept of knowledge distillation originates, however, recent state-of-the-art methods demonstrated even better performances using outputs of intermediate layers instead of output probabilities [48]–[52]. In fact, these works revealed that higher-level information could be extracted and transferred from features and outputs of intermediate layers. Moreover, it was established in transfer learning that this information extracted from intermediate layers was complementary with the one extracted by output probabilities and that using both a feature based and the original knowledge distillation loss resulted in improved performances [50].

Some works in incremental learning made use of attention or feature based distillations [16], [38], [39], however none of them really compared the feature based distillation to its output probability based counterpart. For this reason, experimentations to compare both types of distillations were realised during this thesis to understand each distillation strengths and weaknesses in the context of incremental learning.

2.4.1 Experiments

For these experiments the traditional incremental knowledge distillation L_{KD} (see equation 1.12) was compared to the similarity-preserving loss [49] that was shown to

perform well on many different architectures of WideResNet, very similar architectures to the ResNet ones used in most incremental studies.

Specifically, given a training mini-batch $\mathcal{X} = \{(x_i, y_i)_{i=1}^N\}$, we define the activation map produced by the CNN at a layer l as $A^{(l)} \in \mathbb{R}^{N \times c \times h \times w}$, with N the batch size, and c, h, w respectively the output channels, height and width of the layer. The similarity-preserving loss first flattens the activation maps $A^{(l)}$ into the 2D matrices $Q^{(l)} \in \mathbb{R}^{N \times chw}$ and then computes the pairwise similarity matrices $G_T^{(l)}$ and $G_S^{(l)}$, with T being the teacher model and S the student :

$$\begin{aligned} \tilde{G}_T^{(l)} &= Q_T^{(l)} \cdot Q_T^{(l)\top} \\ G_{T[i,:]}^{(l)} &= \frac{\tilde{G}_{T[i,:]}^{(l)}}{||\tilde{G}_{T[i,:]}^{(l)}||_2} \end{aligned} \tag{2.4}$$

Where $G_T^{(l)}$ and $G_S^{(l)}$ are $N \times N$ matrices where $G^{(l)}[i, j]$ can intuitively be seen as the similarity between the activation map produced by layer l for the input images x_i and x_j. Therefore, the similarity-preserving loss penalises differences between $G_T^{(l)}$ and $G_S^{(l)}$ to encourage the student model to have activation maps that produce the same similarities between pairs of images than the teacher :

$$\mathcal{L}_{SP}(G_T, G_S) = \frac{1}{N^2} \sum_{l \in I} ||G_T^{(l)} - G_S^{(l)}||_F^2 \tag{2.5}$$

Where I represents the set of layers l at the end of each convolution blocks in the teacher and student models, and $|| \cdot ||_F$ is the frobenius norm.

As explained in section 2.1, the experiments were realized on the CIFAR-100 dataset with the ResNet-32 model. This model contains 4 blocks of convolution layers that each extract less and less abstract features until the last layer of the last block that extracts the final features used for prediction by the classification layer. The last layer of each of those 4 blocks of convolutions was thus used in the equation 2.5 to encourage the current incremental model Φ_θ^t to retain similar activation maps than the model Φ_θ^{t-1} from the previous incremental step.

In order to compare the benefits of feature distillations like this similarity-preserving one over the standard knowledge distillation, two different experiments were conducted.

For the first experiment, most of the incremental components of our standard algorithm were removed in order to compare the quantity of information about past classes

retained by both losses without any exterior influence. Specifically, two ResNet-32 models were considered, the first one $\Phi_\theta^T(\cdot)$ was trained on 50 out of the 100 classes of the CIFAR-100 dataset with the standard cross-entropy classification loss. The training of the second model $\Phi_\theta^S(\cdot)$, however, was done on the remaining of the dataset combined with a rehearsal memory of 2000 samples of the 50 initial classes and only with a distillation loss using $\Phi_\theta^T(\cdot)$ as the teacher. This setup allows us to observe how much information about the first 50 classes is transferred with different knowledge distillation losses using the standard incremental dataset.

Results are shown in table 2.2, since the similarity-preserving loss does not train the classifier part of the model, we report the NEM accuracy (equation 1.15) along with the standard top-1 accuracy for all models. This NEM accuracy is obtained by classifying samples directly from the feature vectors in an unparameterized way, and is therefore useful to compare performance even when the classification layer of the model is not trained.

Losses	Φ_θ^T Acc	Φ_θ^T NEM Acc	Φ_θ^S Acc	Φ_θ^S NEM Acc
$\mathcal{L}_{KD}$	76.28	75.77	72.78	62.54
$\mathcal{L}_{SP}$			-	62.24
$\mathcal{L}_{KD} + \mathcal{L}_{SP}$			74.66	63.6

Table 2.2 – Comparison of the quantity of information transferred with different distillation losses. We report both the NEM and standard top-1 accuracy on the 50 initial classes. The similarity-preserving loss does not train the classification layer of the model, therefore only the NEM accuracy could be reported for Φ_θ^S trained with $\mathcal{L}_{SP}$.

First of all, the NEM accuracy reached by the student model $\Phi_\theta^S(\cdot)$ is around 10% lower than the standard accuracy when trained with any loss. This is the case because we train the student in an incremental setting with a rehearsal memory of 2000 samples. Since Φ_θ^T was trained on 50 classes the rehearsal memory contains 40 samples per classes instead of the original 500 images that the dataset contains for each class. For those 50 classes, the NEM classifier therefore estimates the mean feature vector of each class with only 40 samples which leads to those lower NEM accuracies.

Interestingly, the NEM accuracy reached by Φ_θ^S with the similarity-preserving loss is not significantly higher than the one reached with standard knowledge distillation. In fact, they both reach around the same accuracy which is different from the results obtained by the original paper [49] where the similarity-preserving loss reaches slightly higher accuracy. This difference could be explained by the fact that our ResNet-32 model

contains less intermediary features than their WideResNet architectures which leads to slightly less informative vectors used by the distillation loss.

The most interesting result, however, resides in the performance of the combined $\mathcal{L}_{KD} + \mathcal{L}_{SP}$ loss. Indeed, both the NEM accuracy and standard top-1 accuracy are significantly higher which demonstrates the benefits of feature based distillations. Training with any of the two losses alone gives similar performance but training with both at the same time allows for improved student accuracy. This observation showcases that feature-based distillations extract different knowledge that is complementary to the knowledge transferred to the student via the standard knowledge distillation. Moreover, this accuracy improvement is observed both at a classification and feature level as can be seen respectively from the standard and NEM accuracies.

Following this experiment we compared the losses in a more realistic incremental setting where the objective is not only to remember past classes but also adapt to new ones. For the second experiment, a regular incremental training was carried out on the CIFAR-100 dataset with our setup making use of rehearsal memory, knowledge distillation and weight alignment as explained in section 2.1. Only one incremental step was however considered in order to observe better the difference of accuracy between old and new classes. Specifically, our incremental model was first trained on 50 classes regularly, and then both the classification and distillation losses were used to train the model to remember the 50 first classes while learning the new 50 classes. In a similar fashion than for the last experiment, the incremental loss was replaced with each of the following losses :

1. $\mathcal{L} = \mathcal{L}_{CE}$
2. $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{KD}$
3. $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{SP}$
4. $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{KD} + \mathcal{L}_{SP}$

The results obtained for this experiment are shown in table 2.3. As can be seen in this table, the knowledge distillation loss greatly reduce forgetting of previous classes which leads to an improvement of over 7% of accuracy overall compared to the training with only a classification loss. Moreover, the similarity-preserving loss demonstrates an even greater impact on forgetting, reducing it by almost 20% compared to the baseline training without distillation. Furthermore, when both distillations are used for training, the same observation than for the previous experiment can be made, the distillation losses benefit from each other, leading to even more forgetting reduction.

Losses	Initial Accuracy	Incremental Accuracy		
		Past classes Acc	New classes Acc	Overall Acc
$\mathcal{L}_{CE}$	76.14	47.36	67.38	57.37
$\mathcal{L}_{CE} + \mathcal{L}_{KD}$		59.06	68.34	63.69
$\mathcal{L}_{CE} + \mathcal{L}_{SP}$		67.3	50.22	58.76
$\mathcal{L}_{CE} + \mathcal{L}_{KD} + \mathcal{L}_{SP}$		69.84	54.62	62.23

Table 2.3 – Comparison between similarity-preserving and standard knowledge distillation during an incremental training of 50 initial classes with 50 more classes added in 1 incremental step. The top-1 accuracy on both the 50 first classes, the 50 new classes and all 100 classes are reported for all methods.

When the similarity-preserving loss is added for training, however, the adaptation to new classes is also greatly reduced, leading to an overall worse plasticity-stability trade-off than when using the standard knowledge distillation. Indeed, the knowledge distillation keeps only the outputs of past classes similar to the previous model outputs (see equation 1.12), whereas the similarity-preserving loss keeps all intermediate feature vectors similar to those of the previous model. In fact, this illustrates the main advantage of the standard knowledge distillation over feature-based ones in incremental learning. When dealing with the output probability vectors, the distillation can be applied only on past classes knowledge very easily by only considering the past classes outputs before computing the probabilities via the softmax function. With feature-based distillation, however, this is not possible because all images are treated similarly for feature extraction and no components of the feature vectors contain only information about past classes.

This issue leads the standard knowledge distillation loss to have a much better plasticity-stability trade-off, however, we investigate in the next chapter the use of a contrastive feature-based distillation that makes use of several components to reduce its constraints on new classes to ensure better feature adaptation.

2.5 Conclusion

The work described in this chapter covers the most interesting results obtained during the first year of this thesis. In this chapter many experiments were presented on multiple components of a baseline incremental algorithm.

Specifically, the experiments conducted on the feature space of a convolutional neural network trained incrementally revealed a big impact of catastrophic forgetting on the

classes clusters. The initially well separated and compact clusters become very spread out after a few incremental steps, causing them to end up on top of each other which makes it impossible for the classification layer of the neural network to correctly classify classes.

Interestingly, experiments done on the rehearsal memory part of the incremental algorithm revealed that the sampling method used to choose which images should be stored during the whole incremental process did not have much impact on the performances. However, experiments done on the knowledge distillation loss used to alleviate forgetting during incremental steps revealed that feature-based knowledge distillations and the standard probability-based one transfer different informations and can be combined for an even greater impact on forgetting. Feature-based knowledge distillation, however, also displayed a negative impact on feature adaptation to new classes that needs to be addressed to attain a better plasticity-stability trade-off than when using only the standard knowledge distillation.

The observation of the feature space becoming cluttered and the beneficial impact of feature-based knowledge distillation were the foundation of the work that will be presented next. In fact, by leveraging contrastive learning concepts, we present in the following chapter our joint contrastive and incremental algorithm that focuses on reducing the feature space clutteredness and adds a contrastive feature-based distillation term to the training loss.

Chapter 3

Joint incremental and contrastive learning

3.1 Introduction

In this chapter, an approach we published in 2022 will be presented. This approach builds upon the recent state-of-the-art of incremental learning by incorporating contrastive methods to further mitigate catastrophic forgetting.

Traditional state-of-the-art class incremental methods have focused on three main concepts to alleviate forgetting, namely data rehearsal (section 1.3.1, task recency bias correction (section 1.3.3) and knowledge distillation (section 1.3.2). Rehearsal-based methods generate or store a small portion of data from previous tasks and add them to the current task training data [10], [12], [21] in order to keep information about previous tasks in the dataset. As none or only a few samples from past tasks are stored and rehearsed, the dataset used for training is heavily imbalanced which leads to a score magnitude bias in the output of the last fully connected layer of the neural network towards most recent tasks; bias that some works attempt to minimize [12], [18], [35]. Finally, knowledge distillation approaches are regularisation-based methods that add a term to the loss function in order to transfer knowledge of the previous tasks towards the model being trained for the current task [10], [16], [38], [39], [53].

These methods have seen a lot of success over the years and have been proven quite effective in alleviating forgetting in many works but only operate at a classifier level. However, as highlighted in chapter 2, catastrophic forgetting also causes a global loss of information in the feature space, or representation space, of the convolutional network. This loss of information is directly observable in terms of clusters separation and compactness (see section 2.2). In fact, clusters become very scattered during the incremental process, leading to an entangled feature space without a clear separation of each class which hinders the capability of the classifier.

An emerging trend of representation learning called contrastive learning [54], [55] has recently been shown to improve the discriminativeness of model's representations in both the unsupervised and supervised learning setting [54], [56]. An analogy between contrastive and incremental learning is thus drawn in this chapter to improve the discriminativeness of the model's representation during incremental steps.

Specifically, the presented approach trains jointly the feature extractor and classification components of the model, via both contrastive and incremental learning. On the one hand, contrastive learning is employed to learn a more discriminative representation for new classes while avoiding the forgetting of discriminative information of the previous representation for old classes. On the other hand, state-of-the-art incremental methods train an unbiased classifier that also adapts to new classes without forgetting previous ones.

3.2 Background information

In this section, an overview of contrastive methods and how they operate will first be given. Subsequently, the section explores previous studies that have integrated contrastive methods into incremental learning similarly to our proposed approach.

3.2.1 Unsupervised contrastive learning

Initially introduced for unsupervised learning in SimCLR [54], contrastive methods have been shown to learn discriminative representations in unsupervised and supervised scenarios [56]. Conceptually, the idea is that similar images should have similar feature vectors in the representation of the model, therefore pushing similar samples together while pushing dissimilar ones apart ensures the discriminativeness of the feature space learnt this way.

Traditional machine learning employs minibatch training, where minibatches of images are randomly sampled from the dataset and fed to the model in order to compute the loss gradients and update the model's parameters. In SimCLR, data augmentation is used to create two views of each image of the minibatch, by rotating it or changing the colors for example. These two views of the same sample are then considered as a "positive pair", a pair of images that should have similar feature vectors, and views of two different samples constitute a "negative pair", images that should have dissimilar feature vectors (see image

3.1). A contrastive loss is then employed to pull the feature vectors of positive examples together while pushing negative ones apart in a self-supervised fashion.

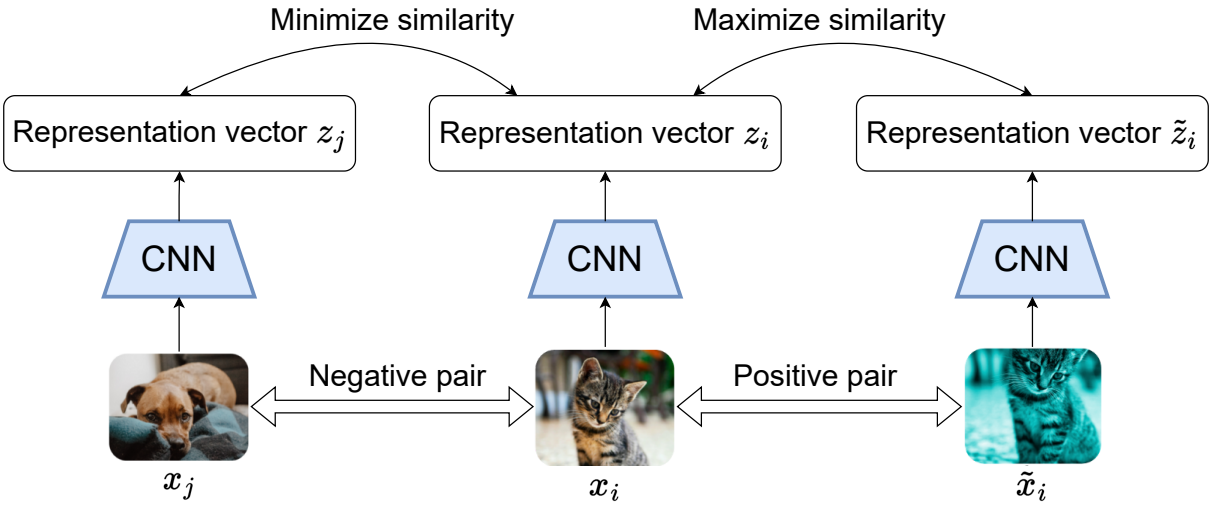


Figure 3.1 – Conceptual schema of contrastive learning. Contrastive methods consider that similar images should have very similar features, therefore they pull positive pairs closer in the representation space while pushing negative pairs apart, resulting in a disentangled feature space.

Specifically, considering a minibatch $\{x_i\}_{i=1}^{N}$ of N unlabeled samples, two augmented views $\{\tilde{x}_i\}_{i=1}^{N}$ are generated to obtain an augmented minibatch $\{\tilde{x}_i\}_{i=1}^{2N}$. All samples from this augmented minibatch are then fed into the feature extractor of the model $\phi(\cdot)$ to obtain representation vectors $\{z_i\}_{i=1}^{2N}$, and a contrastive loss function is computed for each pair of positive vectors (z_i, z_j) :

$$\ell_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(sim(z_i, z_k)/\tau)}$$
$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} \ell(i, j) \tag{3.1}$$

With τ a temperature scalar hyper-parameter and $sim(z_i, z_j) = \frac{z_i^\top \cdot z_j}{||z_i|| ||z_j||}$ the dot product between L_2 normalized feature vectors. This dot product between normalized vectors $sim(z_i, z_j) \in [-1, 1]$ is mathematically equivalent to the cosine similarity between two vectors which measures the angular similarity of vectors. Two parallel feature vectors will have a cosine similarity of 1 and two orthogonal ones will have a cosine similarity of 0.

Therefore, considering the vector of similarities between a representation vector z_i and all other ones in the augmented batch; this loss measures the softmax probability of two

images being a positive pair based on their similarity and corresponds to the temperature-softened categorical cross entropy of this measure. Intuitively the minimization of this loss can be seen as optimizing the feature extractor $\phi(\cdot)$ to maximize the cosine similarity between positive pairs of images and minimize the similarity between negative ones.

Furthermore, in their work SimCLR [54], Chen et al. showed that the performances of such contrastive methods rely mostly on three factors, that are now standard practice in most contrastive learning methods [50], [56], [57].

The first one resides in the data augmentation involved in the generation of positive pairs. Strong data augmentations such as intense color alterations, typically detrimental to supervised learning due to excessive data distortion have been discovered to actually benefit contrastive learning. Moreover, the combination of random crop and color distortions was found to outperform most other data augmentations, and in other studies [56] even more complex ones like autoaugment [58], mixup [59] and cutmix [60].

The second useful component for contrastive methods is a nonlinear projection head, usually a simple multi-layer perceptron with one hidden layer and a ReLU activation function that projects the model's features onto a contrastive space during the training process and is then discarded after training. Specifically, considering a model with a feature extractor $\phi(\cdot)$, the projection head $g_\theta(\dot{)}$ and an image x from the dataset, the representation vectors z used for the contrastive loss are obtained in the following way:

$$z = g_\theta(\phi(x))$$

with θ the parameters of the projection head that are trained with the contrastive loss at the same time as the model's parameters. The data augmentations used to generate the images imply that z is trained to be invariant to those transformations. Thus important information for the classification, like the color or orientation of objects, would be lost without this projection layer. Indeed, it was shown that this projection allows z to become invariant to the data transformations in order to minimize the contrastive loss while the real features $f = \phi(x)$ retain the necessary information used for the downstream classification task.

Finally, the last factor shown to improve capabilities of contrastive methods resides in an increased batch size. In fact, contrarily to supervised learning where smaller batch sizes typically perform better [61], during contrastive learning, bigger batch sizes provide more negative samples for the contrastive loss which favours its convergence and results in a better optimum.

3.2.2 Supervised contrastive learning

While contrastive learning was initially proposed for unsupervised or semi-supervised problems, it has since then been adapted to supervised scenarios by Khosla et al. [56]. Conceptually, as illustrated in Figure 3.2, their approach involves using labels to push together not only an image and its artificially generated positive view, but also all the other samples with the same label within the minibatch.

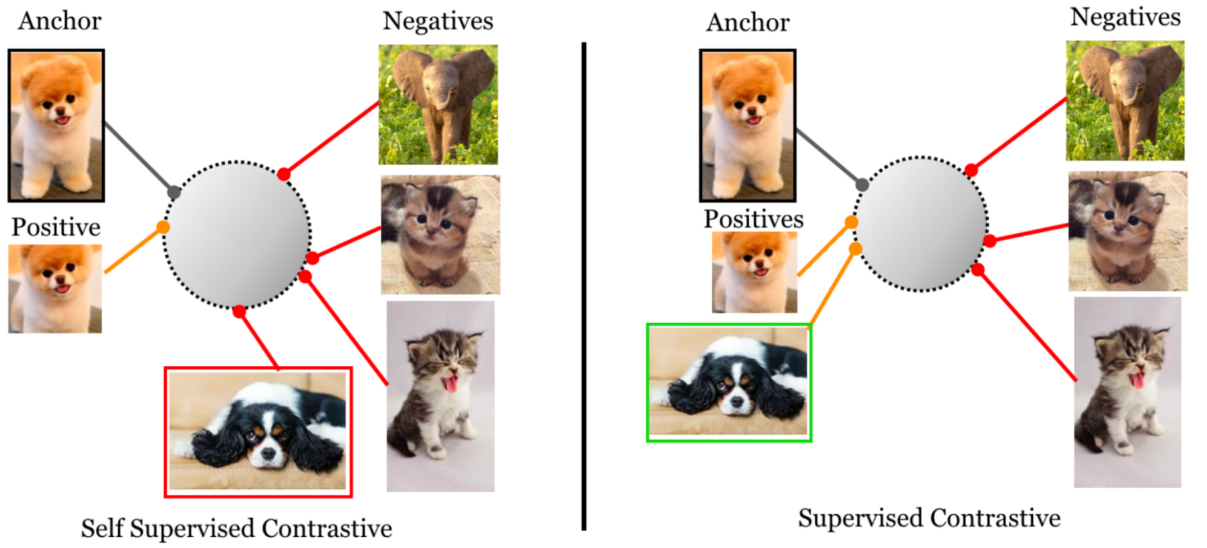


Figure 3.2 – Schema taken from [56] comparing self-supervised and supervised contrastive learning. Considering an anchor image, the self-supervised contrastive loss contrasts one generated positive view against a set of negatives consisting of the entire remainder of the minibatch. The supervised contrastive loss, on the other hand, considers all samples that share the same label as positive views.

Specifically, considering an augmented minibatch I, they proposed a generalisation of the contrastive loss to an arbitrary number of positives with the following loss :

$$\mathcal{L}_{SupCon} = \sum_{i \in I} \frac{-1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \log\left(\frac{\exp(z_i \cdot z_j / \tau)}{\sum_{\substack{k \in I \\ k \neq i}} \exp(z_i \cdot z_k / \tau)}\right) \tag{3.2}$$

where $\mathcal{P}_i$ the subset of the minibatch I containing all the positives of sample i, i.e. all the samples and augmented samples of the same label, τ is a temperature hyperparameter, and all z_i are normalized representation vectors obtained with the projection head.

This supervised contrastive loss was shown to learn more discriminative representations with better generalisation capabilities than ones learned with conventional cross-entropy. Moreover, in incremental learning it has been shown to learn representations

that contain general knowledge useful for the classification of unseen classes which is particularly beneficial to transfer to upcoming incremental steps [57].

3.2.3 Contrastive distillation

In the field of transfer learning, while standard knowledge distillation compares output probability distributions to transfer knowledge from a teacher to a student model, recent works showed that richer knowledge can be transferred from the features of the models [38], [39], [48]. Based on this observation numerous contrastive learning methods have been developed to transfer knowledge from models representations. One approach introduced in CRD [50] and improved in WCoRD [52] was to consider the representation of the teacher and the student as two different view of the same image and use the contrastive loss to maximize mutual information between both representations.

While this approach demonstrated impressive performances against the state-of-the-art of transfer learning methods [50], other contrastive approaches have been proposed recently based on a different concept. Indeed, in SEED [51] and SSKD [62], instead of using the teacher as a contrastive view of the image, they propose to generate a contrastive view with data augmentation and then compare pairwise similarities between the representation of the teacher and student. Conceptually, they propose to compute for both the teacher and the student model, the distribution of pairwise similarities in the augmented minibatch similarly to unsupervised contrastive methods like SimCLR [54] :

$$A_{i,j} = \frac{\exp(z_i \cdot z_j/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(z_i \cdot z_k/\tau)}$$

Each line of this matrix A contains a probability distribution representing the similarity between an image i and all other images j, therefore similarly to standard knowledge distillation the Kullback-Leibler divergence loss can be used to transfer knowledge from the teacher to the student:

$$\mathcal{L} = \sum_{p_i \in A} KL(p_i^T || p_i^S)$$

With p_i^T and p_i^S the distributions extracted respectively from the teacher and the student model. This objective does not constraint directly features of the student to be similar to those from the teacher. It instead constraints the similarities between images produced in both feature spaces, which allows more plasticity in the student model while still captur-

ing important structural knowledge from the teacher feature space. This type of feature similarity distillation demonstrated impressive performances especially for transferring knowledge between different teacher and student network architectures [49], [51], [62].

3.2.4 Contrastive incremental learning

The idea of applying contrastive learning methods to incremental learning in order to improve the feature space of incremental models has been explored in very few works. In Supervised Contrastive Replay [44], Mai et al. proposed a contrastive method for online incremental learning (a different setting than the one studied in this thesis, see section 1.2.2). They paired contrastive learning with the NEM classifier from iCarL [10] (see section 1.3.3) in order to remove the need of a classification layer entirely.

In the offline incremental setting, contrastive methods were studied in the Co2L paper [57]. In this work the author learn representations via contrastive learning with an asymmetric supervised contrastive loss that only pulls together samples of new classes during each incremental steps, and retain knowledge about old classes with a contrastive distillation similar to the one employed in SEED [51]. Their method demonstrated great improvements in the feature space of the network, however since they employed only contrastive learning to learn representations they had to add another training step to each incremental step to learn a classifier on top of the contrastively learnt representation.

Overall, contrastive methods demonstrated impressive performance both in terms of learning discriminative representations and retaining them via knowledge distillation, but lack impact on the classification layer of the model, while incremental methods work at a classifier level and lack impact on the representation of the model. This is why we propose a new method that jointly trains the feature extractor and classification layer in a contrastive incremental setup.

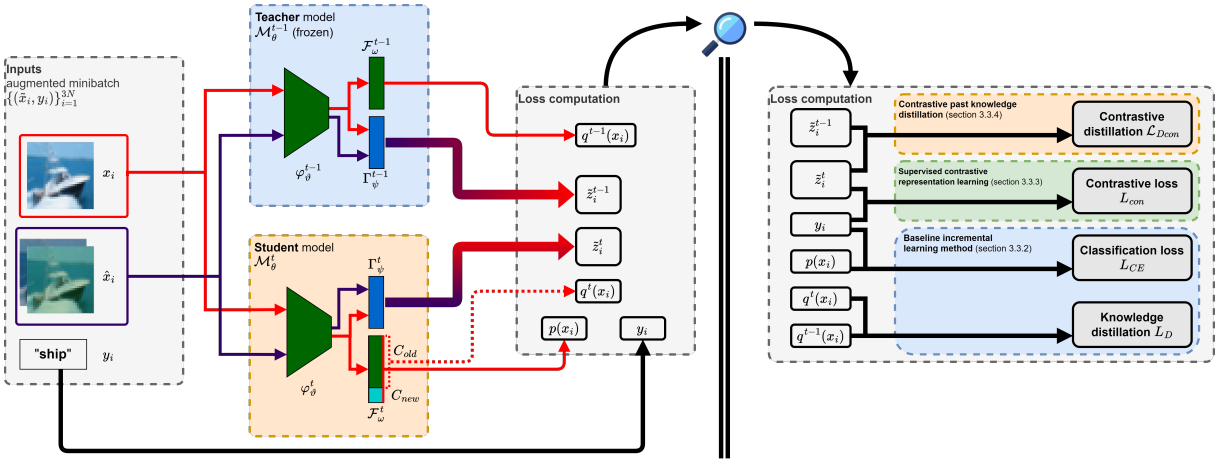


Figure 3.3 – Pipeline of the proposed approach. During each incremental step, images are sampled in minibatches from the incremental dataset containing new data and rehearsal data in order to compute the four losses $\mathcal{L}_{CE}$, $\mathcal{L}_{con}$, $\mathcal{L}_{D}$, and $\mathcal{L}_{Dcon}$ used in eq. 3.4.

3.3 Proposed method

In this section, our approach for joint contrastive and incremental learning will be presented. For this algorithm, the standard state-of-the-art incremental methods are employed to alleviate forgetting at a classifier level, while contrastive classification and distillation losses are leveraged to learn and retain a discriminative feature space during incremental steps.

3.3.1 Overview

Like many state-of-the-art methods for incremental learning [10], [12], [35], we employ rehearsal-based training with the cross-entropy loss $\mathcal{L}_{CE}$ (see equation 3.5) to learn new classes during each incremental step and the distillation loss $\mathcal{L}_{D}$ (see equation 3.6) to preserve knowledge about previously learnt classes. The usual baseline incremental loss used in most studies is :

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{CE} + \lambda\mathcal{L}_{D} \tag{3.3}$$

with λ set to $\frac{C_{old}}{C_{new}+C_{old}}$ [12], [35], C_{old} representing the number of past classes and C_{new} the number of new classes.

This loss, however, only operates at a classifier level. We therefore propose the addition of a contrastive version of both the classification and distillation loss, operating at a

feature level, in order to improve the representation of new classes as well as preserve the discriminative features learnt in previous tasks. We consequently introduce the following training loss function:

$$\mathcal{L} = (1-\lambda)(\frac{\mathcal{L}_{CE} + \mathcal{L}_{con}}{2}) + \lambda(\frac{\mathcal{L}_{D} + \mathcal{L}_{Dcon}}{2}) \tag{3.4}$$

with $\mathcal{L}_{con}$ being a contrastive loss that we describe in section 3.3.3, responsible for learning better representations for new classes, and $\mathcal{L}_{Dcon}$ the contrastive distillation loss described in section 3.3.4 for preserving the representation of past classes. During each incremental step we then optimize the parameters of the model in order to minimize the loss function described in equation (3.4) over the incremental dataset containing the data from new classes and the rehearsal memory. The overall pipeline of our framework is shown in Figure 3.3 and explained in detail in the following subsections.

3.3.2 Incremental learning baseline

In this section we will describe the incremental learning scheme using knowledge distillation and data rehearsal on which is based our method. Let us consider the classification task at incremental step t with C_t classes comprising C_{new} new classes and C_{old} past classes, the task dataset $\mathcal{D}_t = \{(x_i, y_i)_{i=1}^{N}\}$ contains all the available data about the C_{new} classes and only the rehearsal samples stored about the previous C_{old} classes. The model parameters θ are initialized with the values obtained at the previous incremental step, and C_{new} new randomly initialized output nodes are added for the new classes. We then train the model with the standard cross entropy loss $\mathcal{L}_{CE}$ to learn new classes and with the knowledge distillation loss $\mathcal{L}_D$ to alleviate forgetting of the previous classes :

$$\mathcal{L}_{CE}(x, y) = \sum_{c=1}^{C_t} -y_c \ \log(p_c(x)) \tag{3.5}$$

$$\mathcal{L}_{D}(x) = \sum_{c=1}^{C_{old}} -q_c^{t-1}(x) \ \log(q_c^t(x)) \tag{3.6}$$

where $p_c(x)$ is the output softmax probability for the c^{th} class, $q_c^t(x) = \frac{e^{o_c(x)/\tau}}{\sum_{i=1}^{C_{old}} e^{o_i(x)/\tau}}$ is the softened softmax probability obtained from output node o_c of the model, $q_c^{t-1}(x)$ is the same softened softmax probability but obtained from the outputs of the model from task $t-1$, and τ is a temperature parameter.

Furthermore, following each incremental learning step we adopt the weight aligning bias correction method introduced in [35] (see section 1.3.3) that proved to be very effective in removing the classification layer bias while requiring negligible computation time.

3.3.3 Contrastive features separation

In order to learn good representations for new classes with contrastive learning, we use a setup similar to CO2L [57]. First, when a batch of N samples $\{(x_i, y_i)\}_1^N$ is drawn from the dataset we use heavy data augmentation to generate 2 augmentations $\{(\hat{x}_i, y_i)\}_1^{2N}$ of each image. Then, considering the augmented minibatch $\{(\tilde{x}_i, y_i)\}_1^{3N}$, we extract the features $\phi_i = \varphi_\vartheta^t(\tilde{x}_i)$. Following [50], [51], [57], [62], a projection map Γ_ψ^t parameterized by ψ is used to project features onto a d-dimensional unit hypersphere : $\tilde{z}_i = \frac{\Gamma_\psi^t(\phi_i)}{\|\Gamma_\psi^t(\phi_i)\|}$, and the parameters ψ are optimized together with the model parameters to minimize our overall loss described in eq. 3.4. Finally, the contrastive features $\tilde{z}_i$ of the augmented batch are used to minimize the asymmetric supervised contrastive loss introduced in [57] :

$$\mathcal{L}_{con} = \sum_{i \in S} \frac{-1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \log\left(\frac{\exp(\tilde{z}_i \cdot \tilde{z}_j / \tau)}{\sum_{\substack{k \in I \\ k \neq i}} \exp(\tilde{z}_i \cdot \tilde{z}_k / \tau)}\right) \tag{3.7}$$

where I is the augmented minibatch, S is the subset of I containing only samples of new classes, $\mathcal{P}_i$ the subset of S containing the positives of sample i, i.e. all the samples and augmented samples of the same label, and τ is a temperature hyperparameter.

Since $\tilde{z}_i$ and $\tilde{z}_j$ represent the projection of the features onto a d-dimensional unit hypersphere, the dot product is equivalent to a cosine similarity. Therefore, this loss can be seen as maximizing similarity between new classes samples and their positives while minimizing similarity with negatives. Thus "pulling" together new classes samples and their positives in the representation while "pushing" away all other samples.

3.3.4 Contrastive features distillation

Using the asymmetric supervised contrastive loss allows the model to learn better representations for new classes but not for past classes. In order to preserve the good representation learnt previously for those classes we introduce a new supervised contrastive distillation loss inspired from [62]. The general goal of this loss is to allow the representation of the model to change to extract discriminative features for new classes but to ensure the new features produce the same similarities between rehearsal samples than the

previous features. This way the representation is allowed to adapt to new classes but the underlying information about past classes is preserved.

Since the model Φ_θ^{t-1} from the previous incremental step has not been trained on data about the new classes, the representation obtained for the $\{(x_i, y_i)\}_{y_i \in C_{new}}$ is not necessarily a very discriminative one. We therefore ignore samples from new classes when computing this distillation loss in order to focus on preserving similarities between representations of samples kept in the rehearsal memory.

Similarly to the loss $\mathcal{L}_{con}$ described in the previous section, using $\{(x_i, y_i), y_i \in C_{old}\}$ and $\{(\hat{x}_i, y_i), y_i \in C_{old}\}$, we compute z_i^t, and $\hat{z}_i^t$, but also z_i^{t-1} and $\hat{z}_i^{t-1}$, to obtain the contrastive representation produced by Φ_θ^t and Φ_θ^{t-1} for each image of past classes from the minibatch and the augmented versions of these images. We then compute the pairwise similarities between z_i and $\hat{z}_i$ for each model and organize them into matrices B^t and B^{t-1} with :

$$B_{i,j}^t = \frac{z_i^t \cdot \hat{z}_j^t}{\tau} \tag{3.8}$$

where $B_{i,j}^t$ contains the similarity between the contrastive representation of Φ_θ^t for x_i and $\hat{x_j}$, and τ is another temperature hyperparameter. We then apply softmax to each row of the matrices B^t and B^{t-1} to obtain probability distributions, and in analogy to the distillation process described in eq. 3.6, we minimise the divergence between those two probability matrices :

$$\mathcal{L}_{Dcon} = -\tau^2 \sum_{i,j} B_{i,j}^{t-1} \log(B_{i,j}^t) \tag{3.9}$$

Overall, this loss allows the representation of the model to adapt to new classes but ensures that the representation of samples from previous incremental steps produce the same similarities than in the representation of the previous model.

3.4 Experiments

In the following sections we will describe the details of our algorithm and the general incremental setup we used, compare our algorithm to other state-of-the-art methods and conduct ablation studies to validate the effectiveness of our method.

3.4.1 Experimental setup

We evaluate our algorithm and other methods on two datasets that are widely used in incremental learning [10], [12], [35], Cifar-100 and ImageNet-100. Cifar-100 [63] contains 32 × 32 pixel color images of 100 classes, with 500 images per class for training and 100 images per class for validation. ImageNet-100 on the other hand contains images of 64 × 64 pixels and represents a subset of 100 random classes from the ImageNet ILSVRC 2012 [7] dataset containing 1000 classes. Imagenet-100 contains 500 images per class for training and 50 images per class for validation.

We employed PyTorch in our implementation and following [10], [12], [35] we chose the 32-layer Resnet model for Cifar-100 dataset and 18-layer Resnet [8] for ImageNet-100. We used the optimizer SGD with a momentum of 0.9, a batch size of 128, and a weight decay of 0.0002. We trained our models for 250 epochs during each incremental step, the learning rate starts at 0.1 and is divided by ten after 150, 180, and 210 epochs. The data augmentation applied to training images consists in random cropping, horizontal flip and normalization. The temperature parameter τ was set to 2 in $\mathcal{L}_D$ and 0.2 for the contrastive losses $\mathcal{L}_{con}$ and $\mathcal{L}_{Dcon}$. Moreover, for the contrastive losses we create 2 images with the same data augmentation than the initial image with the addition of color jitter and random color dropping similarly to [57]. Following other contrastive learning methods [62] we use a 2-layer MLP to project features onto the contrastive unit hypersphere. We separate each dataset in 10 incremental steps, starting initial learning with 10 classes and adding 10 classes per step. Following [10] we use a rehearsal memory of 2000 images and use the herding sampling strategy.

3.4.2 Comparison to other methods

We compare our method to several other rehearsal based competitive incremental learning algorithms :

Incremental Classifier And Representation Learning (iCarL). [10] This algorithm uses a nearest-exemplar-mean (NEM) classifier to remove new classes bias during evaluation time, trains the model using a binary cross-entropy based classification and distillation loss, and uses data rehearsal with the herding selection strategy.

Maintaining Discrimination and Fairness in Class Incremental Learning (MDFCIL). [35] This method differentiates itself from iCarL by using the conventional cross-entropy for the classification and distillation losses and adding the weight alignment

	Cifar-100		ImageNet-100	
	last Acc	avg Acc	last Acc	avg Acc
Finetuning	8.81 ±0.38%	18.76 ±0.14%	8.66 ±0.28%	17.58 ±0.43%
iCarL_CNN	39.47 ±0.75%	54.78 ±1.24%	40.65 ±1.17%	53.58 ±1.66%
iCarL_NEM	47.80 ±0.73%	59.51 ±1.28%	**47.82 ±1.07%**	57.83 ±1.60%
CO2L	32.15 ±0.18%	47.27 ±0.10%	33.51 ±0.20%	50.01 ±1.03%
MDFCIL	50.43 ±0.71%	62.02 ±2.14%	46.29 ±1.84%	56.08 ±1.75%
Ours	**50.81 ±0.59%**	**64.13 ±0.52%**	47.64 ±1.32%	**59.13 ±1.59%**
Joint Training	69.39 ±0.26%		67.24 ±0.78%	

Table 3.1 – Class incremental learning performance on Cifar-100 and ImageNet-100 with 10 incremental steps and 10 classes added per step. The top-1 average accuracy over all the incremental steps as well as the accuracy after the last one are reported. For each method we report the mean over 10 runs with random class orderings for fair comparison. For the method iCarL we report performances using the model classification layer (iCarL_CNN) and using their nearest exemplar mean classifier (iCarL_NEM).

step that we used in our algorithm after each incremental training step to remove bias from the classification layer of the model.

Contrastive Continual Learning (CO2L). [57] This method trains a feature extractor using contrastive versions of the classification and distillation losses used in incremental learning. Compared to the contrastive losses used in our method the main difference is the equation of their contrastive distillation loss and its computation on all samples from the minibatches instead of just samples coming from the rehearsal memory. Since their method trains only a feature extractor they further add a second training step to train a classifier with the conventional cross-entropy.

Finetuning. Finetuning represents the lower bound of performance achievable in incremental learning. Finetuning is a simple training setup with only the conventional cross entropy applied to finetune the model with each incremental dataset and no other incremental learning parts.

Joint-training. Joint-training in the other hand represents the upper bound of performances. It corresponds to training a model from scratch with conventional cross-entropy during each incremental steps with the total dataset containing all data about new and past classes.

For thorough comparison of our method to state-of-the-art ones we run each algorithm 10 times on the two datasets considered, each method with the same 10 random

class orderings. For fair comparison, we use the same models for each algorithms, so 32-layer Resnet for the dataset Cifar-100 and 18-layer Resnet for the dataset Imagenet-100. Performances of contrastive methods are positively correlated with large batch sizes [50], [54], but it is not the case for incremental methods. Therefore we run all algorithms with a batch size of 128 on both datasets for an unbiased comparison and coherence with other incremental non-contrastive studies. We report in table 3.1 the top-1 accuracy obtained after the last incremental step and the average incremental accuracy over all incremental steps, ignoring the accuracy of the initial non-incremental learning step. We further provide in figure 3.4 the accuracy of each method on Cifar-100 as a function of the number of classes seen during the incremental process.

As can be seen in table 3.1 and in figure 3.4 our method slightly surpass all other methods on both datasets, verifying our idea that contrastive methods can be used together with current state-of-the-art incremental methods to improve the representation of the model which in turn improves downstream classification accuracy. Besides, we can also observe on the figure 3.4 that the performances of the other contrastive learning method CO2L are quite low on Cifar-100 which can be explained by the batch size used relatively low for the dataset compared to usual contrastive learning batch sizes. However, since our method uses contrastive losses jointly with incremental losses we can see that it is much more robust to small batch sizes.

3.4.3 Ablation study

In order to validate the effectiveness of our method we performed the following ablation studies :

- **Ablation A.** Ablation of $\mathcal{L}_{Dcon}$. We remove the contrastive distillation loss from the optimization process to evaluate the impact of this new distillation.
- **Ablation B.** Ablation of all contrastive losses. Removing only $\mathcal{L}_{con}$ would also impact $\mathcal{L}_{Dcon}$ because Γ_{ψ}^{t-1} would not have been trained by $\mathcal{L}_{con}$ during the previous incremental step. Therefore we instead perform an ablation of both $\mathcal{L}_{con}$ and $\mathcal{L}_{Dcon}$ to see the added benefit of contrastive losses and compare to ablation A to see the added benefit of individual contrastive losses.
- **Ablation C.** Ablation of non-contrastive losses. In order to observe the impact of the incremental losses we perform an ablation of $\mathcal{L}_{CE}$ and $\mathcal{L}_{D}$ and keep only $\mathcal{L}_{con}$ and $\mathcal{L}_{Dcon}$.

For a more straightforward comparison and since the focus of this ablation study is to

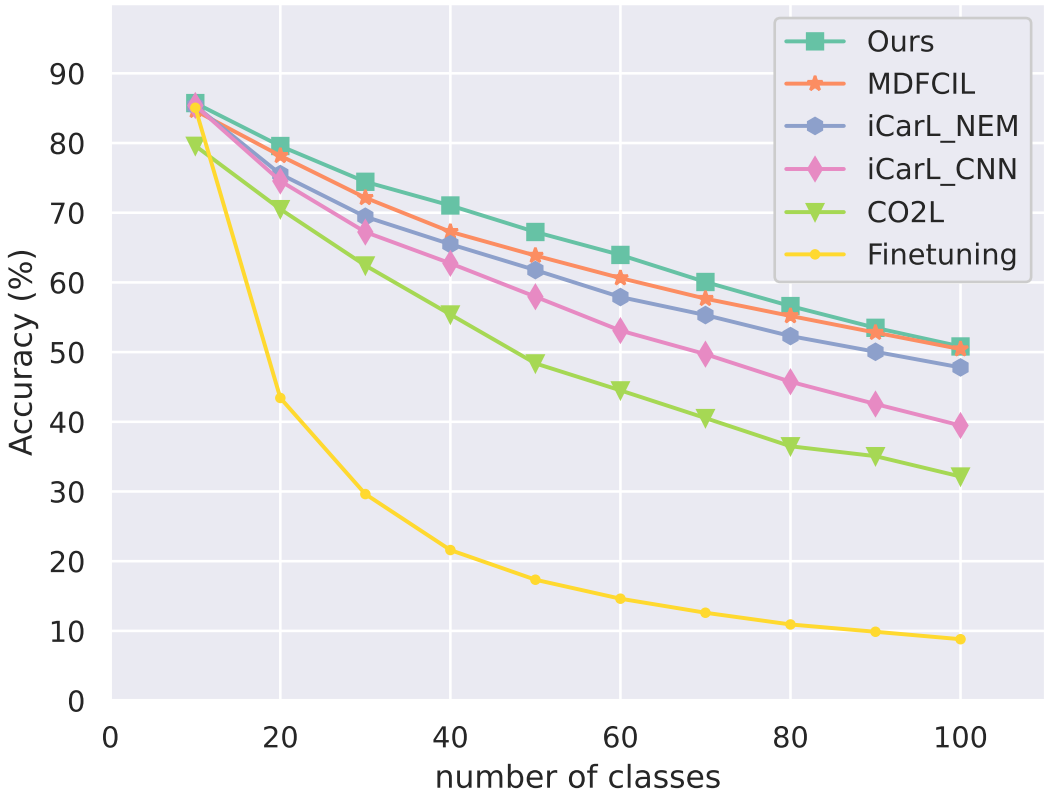


Figure 3.4 – Evolution of the accuracy as a function of the number of classes learnt incrementally. The mean performance obtained on Cifar-100 over 10 training trials with random class orderings is shown, excluding standard deviations for clarity.

evaluate the impact of each part of our method and not to evaluate differences between datasets, we compared performances only on the dataset CIFAR-100. We can see that the performances of our method in table 3.1 and 3.2 are similar but not exactly the same, this is the case because of different class orderings, therefore for fair comparison we used the same random class orderings for each ablation. We report in table 3.2 the top-1 average incremental accuracy obtained with the model classifier and with the nearest exemplar mean (NEM) classifier from [10].

The NEM classifier allows us to evaluate the representations of models without being impacted by the classification layer. This classifier first computes the mean feature vector of each class using the incremental training set and the rehearsal memory after each incremental step. Then, at test time, images are classified to the closest mean vector, therefore

classifying images directly within the feature space without the use of the classification layer or any other parameters. Since all incremental losses are removed in ablation C, the classification layer of the model is not trained at all, therefore we make use of the NEM classifier to compare its performances to the other ablations.

	Top-1 accuracy	Top-1 NEM accuracy
Full method	**63.91** ±0.96%	**63.10** ±1.10%
Ablation A	63.33 ±1.18%	62.71 ±1.28%
Ablation B	62.65 ±1.32%	61.39 ±1.18%
Ablation C	-	58.51 ±1.05%

Table 3.2 – Ablation study done on CIFAR-100 with 10 incremental steps. We report the top-1 average incremental accuracy and NEM accuracy. Each method was run 10 times with random class orders but with the same ones for each ablation for fair comparison. Ablation C does not train a classification layer therefore top-1 accuracy can not reported and NEM accuracy is used instead to compare performances.

By comparing the full method and ablation C in table 3.2, we can clearly see that adding incremental losses to the contrastive ones improves the method. Indeed, in ablation C the top-1 accuracy is not provided because contrastive losses only train the representation of the neural network and not the classifier. This is the most straightforward benefit of using them with incremental losses, the classifier is trained jointly with the representation. Moreover, comparing NEM accuracies we can see that incremental losses also improve the representation of the model which is mainly due to $\mathcal{L}_D$ that can extract knowledge about past classes from images of new classes where $\mathcal{L}_{Dcon}$ uses only the rehearsal memory to extract knowledge about past classes.

On the other hand, comparing ablations A and B to the full method shows that incremental losses also benefit from contrastive ones as the addition of each contrastive loss slightly improves accuracies. Indeed, in ablation A where only $\mathcal{L}_{con}$ is added compared to ablation B, the accuracies are slightly higher which can be explained by the representation of new classes during each incremental step that is improved. And the same observation can be done when comparing the full method to ablation A, the addition of $\mathcal{L}_{Dcon}$ further improves the representation of the model by alleviating catastrophic of the features from previous classes therefore improving accuracy.

Overall, the ablation results show that the removal of the incremental and contrastive losses both decrease performances, therefore validating our hypothesis that both the standard distillation and the contrastive distillation alleviate forgetting and that the model

benefits from using both. Performance gains from contrastive losses however remain moderate, we therefore believe it would be interesting for subsequent works to increase overall importance of contrastive losses compared to incremental ones during training and validate the method on large scale datasets.

3.5 Conclusion

In this chapter we described the state-of-the-art of contrastive learning and its applications to incremental learning. We showed that contrastive learning methods work at a feature level to learn more discriminative representations, and that knowledge distillation based contrastive approaches where especially effective in transferring those representations.

We therefore proposed a new incremental method based on contrastive learning that we published in the continual learning workshop of the Computer Vision and Pattern Recognition Conference (CVPR 2022). This method jointly trains the classifier and feature space of the model respectively with incremental and contrastive methods to improve the feature space of the model. More specifically, we proposed improving the discriminativeness of new classes' features with a contrastive supervised classification loss while preventing forgetting of this discriminative information with a contrastive knowledge distillation loss. Using those losses together with the classical incremental equivalents that work at a classifier level allowed us to make sure catastrophic forgetting is alleviated both in the classifier and in the feature extractor during the entire incremental training.

We demonstrated the effectiveness of our method by comparing it against state-of-the-art incremental methods and outperformed all of them on two baseline datasets. While this method improves the average incremental accuracy by around 2%, it requires generating 2 contrastive views for each image of the training mini-batches which induces a non negligible training time and Graphics processing unit (GPU) memory overhead. For the small datasets on which we tested this method this overhead was not a problem however when we started studying the very large Imagenet-1000 dataset this overhead became an issue. For these reasons we will present in the next chapter a new approach completely different we developed during the third year of the thesis while making sure it could remain applicable to large datasets.

Chapter 4

FECIL : Feature Expansion and enhanced Compression for Incremental Learning

This chapter presents the concluding contribution of the thesis, on the basis of the previous findings and insights gained. In particular, while our approach based on contrastive methods demonstrated quite effective at improving the feature space, it also introduced a GPU memory overhead that prevented us from testing it on larger datasets. We therefore took a closer look at an emerging trend of incremental learning based on dynamic models.

Dynamic models have attained impressive performances in incremental learning, managing to reach better plasticity-stability trade-offs than classical methods at the cost of an increasing number of parameters. One particular method called Dynamically Expandable Representations (DER) [46] stands out in this field, as it demonstrated impressive feature quality outperforming the previous state-of-the-art by a large margin.

In the following sections 4.1 and 4.2, the operation of dynamic models and the way by which they attain better plasticity-stability trade-offs is first described in detail, followed by an explanation of our approach that uses a dynamic network to improve the quality of the feature space and compresses it to avoid the increasing number of parameters over the incremental steps.

4.1 Background

Conceptually very different from the traditional methods used to alleviate forgetting presented in the chapter 1, the use of dynamic neural network architectures has recently gained popularity for incremental learning. This rise of popularity arrived with the publication of the Dynamically Expendable Representation (DER) algorithm [46], that attained a much better optimum in terms of features quality than previous state-of-the-art

methods.

In the following sections we thus provide a small background on dynamic networks and the mixup procedure that we both leverage in our method FECIL that will be explained in the remaining of this chapter.

4.1.1 Dynamic architectures

An emerging concept in incremental learning advocates the use of dynamic models so as to attain better trade-offs between learning of new classes and retaining information from past classes [46].

The underlying idea is that in order to prevent forgetting of previous information, different neurons should be used for each incremental task. Initial approaches took advantage of the fact that neural networks are generally largely over-parameterized [64] and proposed choosing different neuronal paths through the model [65] or different subsets of the model [64] to use during each task. These approaches demonstrated a crucial attribute of dynamic models for incremental learning : their immunity to catastrophic forgetting. In fact, as each incremental task employs a distinct set of parameters, the parameters associated with prior tasks remain entirely unchanged. Thus, by definition, catastrophic forgetting does not occur.

While catastrophic forgetting is prevented, many challenges of incremental learning remain, such as the task recency bias explained in section 1.3.3, and inter-task confusions. In fact, deep models trained with supervision learn to differentiate each class from all others represented in the dataset. Therefore, even with no forgetting, since the incremental dataset contains mostly samples of new classes, learning to differentiate new classes from past classes is not trivial and leads to inter-task confusions compared to joint training on the full dataset. In addition, many challenges arise around the sparsity of models trained this way [64] which refers to training networks where a significant number of connections or weights are set to zero. Indeed, these approaches are inherently bounded by the network capacity, which can be quickly attained just after a few incremental learning steps. Therefore by promoting sparsity, researchers aim to improve the efficiency of models to increase the longevity of those dynamic models in incremental learning scenarios.

Towards overcoming this network capacity limitation, recent works explored the addition of new neurons incrementally to accommodate new classes [46], [66]–[68]. Freezing previous feature extractors and adding entirely new ones incrementally has indeed been shown to not only prevent features forgetting by keeping previously trained neurons but

also allow positive forward transfer from previous tasks to new ones [69]. This forward transfer therefore leads to faster training and better performances for new tasks at the cost of a number of parameters rapidly increasing with the incremental steps due to the addition of new feature extractors.

Indeed, adding new parameters for each incremental task alleviates the network capacity problem of earlier methods but raises parameter growth issues. The main solution explored in several works [46], [64], [66], [70] to these issues revolves around the concept of neural network pruning and sparsity losses.

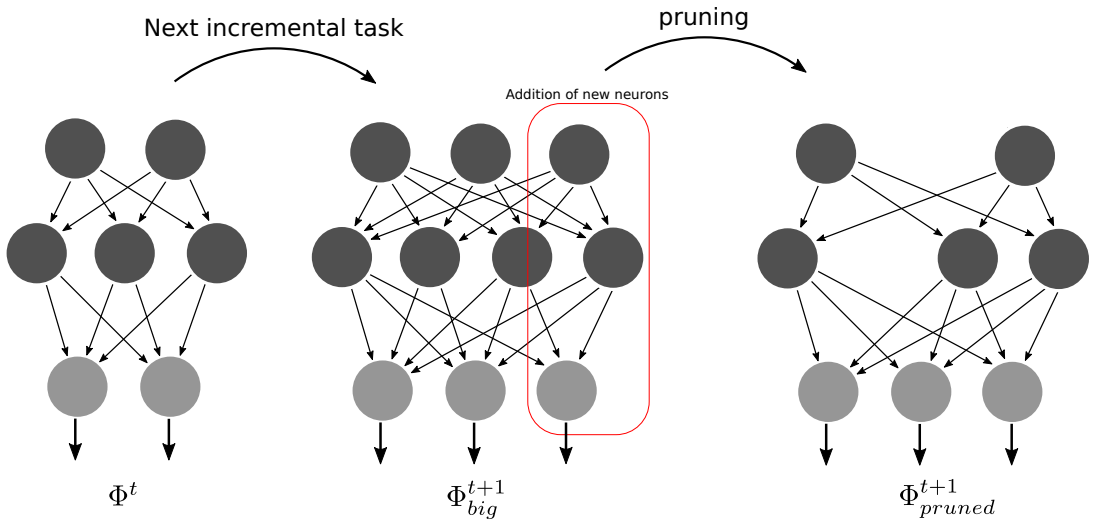


Figure 4.1 – Conceptual schema of dynamic architectures and neural pruning. During incremental steps new neurons are added to an existing model Φ^t, producing a bigger model Φ^{t+1}_{big} and least used neurons are then pruned to reduce the size of this new model.

Conceptually, these approaches propose training models with a sparsity loss added to the classification loss to learn models with many parameters very close to zero and then prune these unused weights. Doing so during the initial and each incremental training step allows these algorithms to use as little weights as possible to accommodate new classes, effectively reducing the size of the model and mitigating the parameter growth issue.

Specifically, for each convolutional layer l with an input feature map f_l representing the output of the previous layer, these approaches generally consider trainable channel-level masks $m_l \in [0,1]^{c_l}$, where $m_l^i \in [0,1]$ and c_l is the number of channels of f_l. They then propose to modulate the input of each convolutional layer of the model with the following equation [46], [70] :

$$f_l' = f_l \odot m_l$$

where $\odot$ represents the channel-level multiplication, and a sigmoid is used as a gating function to ensure the values of m_l stay in the range $[0, 1]$.

Then the addition of a sparsity loss similar to the l_2 loss introduced in section 1.3.2 encourages the parameters of these masks to be close to zero during the training process. Finally, after training, these masks are binarized and the channels corresponding to the zero values of the masks are pruned from the network, leading to a network with a reduced number of parameters. While these pruning methods effectively reduce the number of parameters, they impact the performance of models and their size still increases with each incremental task. Some works therefore required many hyper-parameters carefully set depending on the dataset and model architecture used to obtain an acceptable number of parameters with minimal performance drop [46], while others avoided using hyper-parameters [66] but had less control on the parameter growth which leads to generally better performance but bigger model sizes than the static architecture counterparts.

Some other strategies have been explored, e.g. the authors of DyTox [71] used a transformer architecture allowing a very reduced parameter growth and removed entirely the need for a pruning mechanism. However, transformers require a lot of data to train on and still struggle to reach CNNs image classification performances in incremental learning problems where data from previous classes is very limited [71].

Relatedly, we propose a two stage training approach where we first expand the representation like dynamic architectures [46], [66], [67] and then we use knowledge distillation to compress the dynamic architecture back to its original size. We also introduce a new rehearsal mixup procedure to improve the distillation of previous classes and show that it can be added to most state-of-the-art distillation based incremental methods to greatly improve their performance. This two-stage procedure and improved distillation allows our method to attain dynamic model performance at the benefit of keeping the model size fixed over the series of incremental tasks.

4.1.2 Data Mixup

During the process of incremental learning, past class samples are stored in a small rehearsal memory of fixed size, therefore learning the most out of each sample is crucial. To this end, data augmentations are generally used to improve the variability of training samples, thus improving the downstream models generalisation capability. This is the reason why we investigate the use of a variant of the Mixup [72] data augmentation in our algorithm and demonstrate its effectiveness experimentally. The Mixup augmenta-

tion proved to be very effective in improving generalisation capability of models and, in particular, was shown to improve the separability of classes in the feature space of models.

As illustrated in figure 4.2, Mixup replaces the training samples by linear interpolations between two samples to train the model more towards the boundaries between classes, therefore improving separability of classes in the feature space. Specifically, considering a mini-batch $\{x_i, y_i\}_{i=0}^{N}$, the Mixup augmentation creates artificial training samples in the following way :

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda) y_j \end{aligned} \tag{4.1}$$

where (x_i, y_i) and (x_j, y_j) are two random samples from the current training mini-batch and $\lambda \in [0, 1]$ is an interpolation random variable sampled from a beta distribution $\lambda \sim Beta(\alpha, \alpha)$. This α hyper-parameter controls the overall strength of the interpolation. For example $\alpha = 1$ changes the distribution to a uniform distribution thus giving the same probability for all λ interpolations between x_i and x_j, whereas $\alpha = .1$ would give a much higher probability of λ being sampled close to 0 and 1.

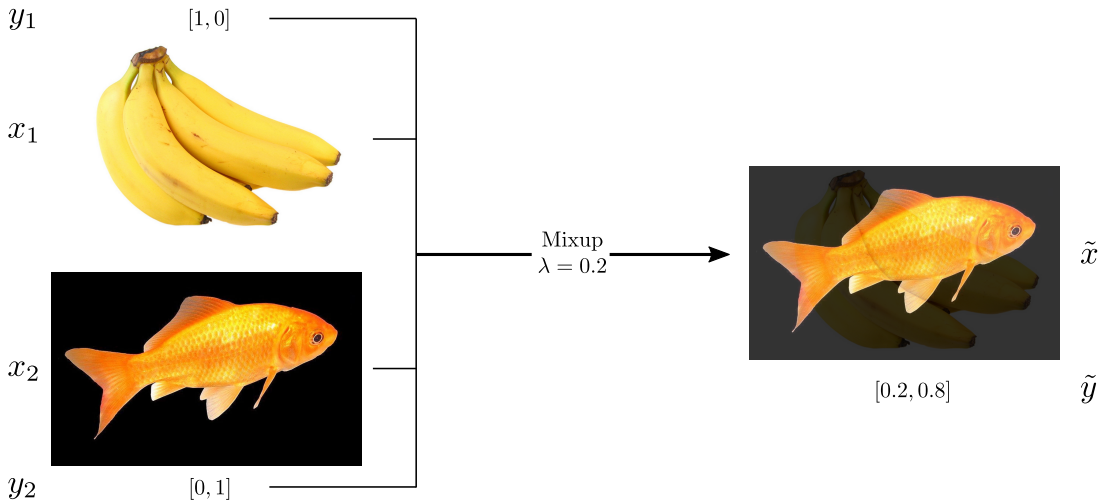


Figure 4.2 – Effect of the mixup data augmentation. During training, images and their associated label are artificially generated using linear interpolations between pairs of samples of each mini-batch.

This simple data augmentation operation has been shown to improve generalization capability of models at negligible computational cost which has led to many applications and improvements proposed over the years [60], [73], [74].

Moreover, in their work DyTox [71], Douillard et al. trained a vision transformer

incrementally and showed that the addition of Mixup not only improved the performances of their transformer, but also reduced the forgetting of the model. Indeed, while their core transformer based method barely reached the latest dynamic CNN incremental method, the addition of the Mixup training procedure allowed them to convincingly surpass all other incremental approaches.

For our algorithm, we propose to build upon this work and use a variant of Mixup that we call "rehearsal mixup" that samples one image from the incremental dataset and the second image from the rehearsal memory to specifically train the model more at the boundaries between new and old classes to reduce inter-task confusions and further alleviate forgetting.

4.2 Method

In this section, our new method for class incremental learning called Feature Expansion and enhanced Compression for Incremental Learning (FECIL) will be presented. This approach is based on a dynamic model that first expands its feature space to accommodate new classes and then compresses it back to its original size to completely avoid the growing number of parameters issue faced by other dynamic model methods.

Each incremental step is split into two training phases. The first one is the expansion phase where we expand the network by adding a new feature extractor and weights for the new classes in the classification layer before updating the model's parameters on the new data task. Upon freezing of the resulting dynamic network we then use knowledge distillation to transfer its knowledge (both the features extractor and the classifier) to a compressed network that will then be used for the next incremental step. A schema representing the overall pipeline is shown in Figure 4.3 while the details of the aforementioned two stages are explained in sections 4.2.1 and 4.2.2 respectively.

4.2.1 Dynamic feature expansion

This phase bears similarities with the one introduced in DER [46]. At incremental task t, like DER, we create the dynamic model Φ^t_{big} that expands the previous feature space with a new feature extractor φ^t_{new} to accommodate new classes. Unlike DER, however, our dynamic model doesn't require all previous extractors but only two, φ^t_{new} and φ^{t-1} because of the compression step we will detail in section 4.2.2. Both the previous and new feature extractors are then fed into a new classifier $\mathcal{H}^t_{big}$ with C_t (the total number of old and new categories) outputs.

More specifically, given an input image from the incremental dataset $x \in \mathcal{D}_t$, the feature vector ϕ of the model becomes the concatenation of both feature extractor outputs:

$$\phi = \{\varphi^{t-1}_{frozen}(x), \varphi^t_{new}(x)\}$$

In order to avoid forgetting of past classes, the previous feature extractor φ^{t-1} is frozen during the entire training process. The feature vector ϕ is then fed into the dense classification layer $\mathcal{H}^t_{big}$ and softmax is applied to the output logits of this classifier to make the prediction for each class:

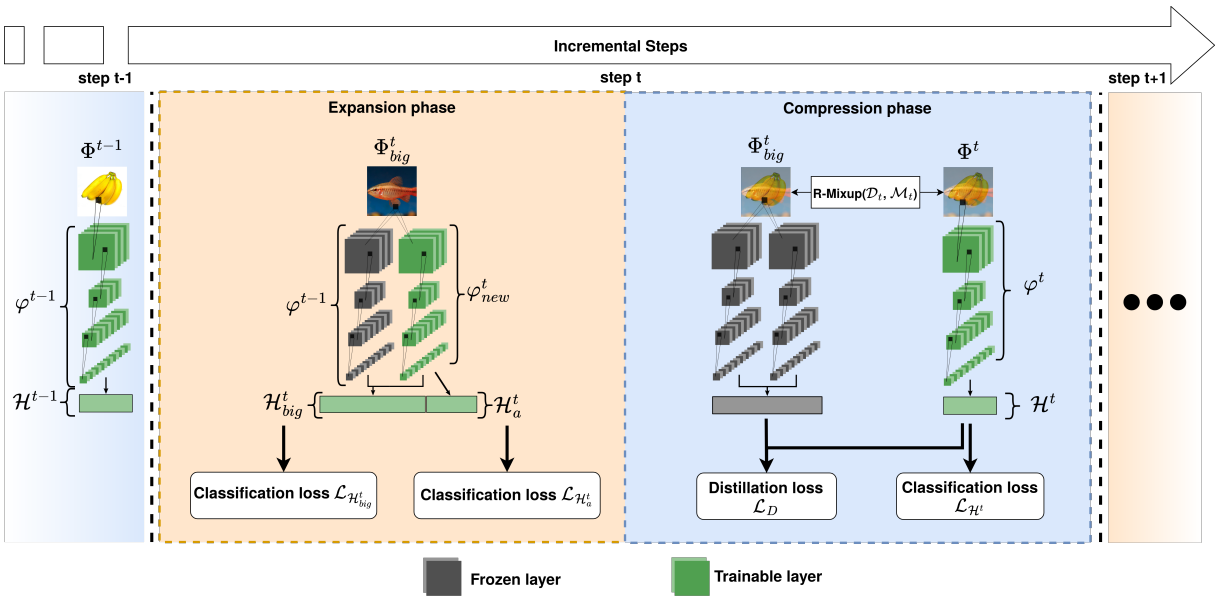


Figure 4.3 – Pipeline of the proposed approach. Each incremental step consists of two training phases, first the expansion phase where we dynamically expand Φ^{t-1} to learn new classes, and then the second phase where we compress the expanded model Φ^t_{big} back to its original state with minimal performance drop using our rehearsal mixup distillation mechanism.

$$p_{\mathcal{H}^t_{big}}(y|x) = Softmax(\mathcal{H}^t_{big}(x))$$

In order to reuse the previously learned classification parameters, the weights of the new classifier $\mathcal{H}^t_{big}$ corresponding to the old features are initiated with the weights of $\mathcal{H}_{t-1}$. On the other hand, in order to allow positive forward transfer between incremental tasks, the newly generated weights of φ^t_{new} are initialized with φ^{t-1}, the previously learned feature extractor.

Classifier training. When training this model we have two different objectives, the first being to train the classifier $\mathcal{H}^t_{big}$ to classify both past and new classes. This is done by minimizing the cross entropy loss on the incremental dataset $\mathcal{D}_t$:

$$\mathcal{L}_{\mathcal{H}^t_{big}} = -\frac{1}{B}\sum_{i=1}^{B}\log(p_{\mathcal{H}^t_{big}}(y = y_i|x_i))$$

where B is size of the batch of images sampled from $\mathcal{D}_t$, x_i is one image of the batch and y_i is its label.

Representation learning. The second objective is to train the new feature extractor to discriminate new classes from each other and past classes but refrain from discriminating between past classes. Indeed, since the previous feature extractor is kept and frozen, the discriminative features about past classes are already learned. Learning features to discriminate between past classes in the new feature extractor would further overfit to the memory data and would negatively impact performances as shown in [46].

To enforce the model to learn discriminative features for new classes we employ the auxiliary classifier $\mathcal{H}_a^t$ introduced in DER. The features from φ_{new}^t are fed into $\mathcal{H}_a^t$ made with $C_{new}+1$ outputs in order to classify all new classes and treat all past classes as one category. Initialized randomly, this classifier is then trained with a cross-entropy loss $\mathcal{L}_{\mathcal{H}_a^t}$:

$$\mathcal{L}_{\mathcal{H}_a^t} = -\frac{1}{B}\sum_{i=1}^{B}\log(p_{\mathcal{H}_a^t}(y=\tilde{y}_i|x_i))$$

where x_i is an image and $\tilde{y}_i$ is the modified one-hot target vector of size $C_{new}+1$.

Our total loss for the expansion training phase therefore becomes the following linear combination of the previously explained losses:

$$L = \mathcal{L}_{\mathcal{H}_{big}^t} + \mathcal{L}_{\mathcal{H}_a^t}$$

Finally, upon the completion of the expansion training phase $\mathcal{H}_a^t$ is discarded and a bias correction method is used to remove the bias towards new classes from $\mathcal{H}_{big}^t$. In fact, the weight alignment method introduced in WA [35] is employed to remove the classification bias induced by the imbalance of the dataset in favor of the new classes.

This weight alignment method is described in details in section 1.3.3 but in short the norms $\mathcal{N}_{old}$ and $\mathcal{N}_{new}$ are derived from each weight vectors of $\mathcal{H}_{big}^t$ corresponding to new and past classes. The average past and new classes' weights norm is then used to compute the rebalancing coefficient $\gamma = \frac{\overline{\mathcal{N}_{old}}}{\mathcal{N}_{new}}$. And finally, all new classes weights are multiplied by this coefficient which reduces them to make them comparable to past classes weights in terms of norm leading to a reduction of the classification bias.

This weight alignment method is adopted at the end of the expansion step in order to make sure the bias is not transferred to the final model Φ_t during the following compression step, its impact is further investigated in section 4.3.4.

4.2.2 Rehearsal mixup compression

Upon the completion of the feature expansion phase, we obtain the model Φ^t_{big} that is bigger than the initial model because of the two feature extractors. Therefore we propose the addition of a compression step that restores the model back to its original size so that after each incremental step the size of the learned model remains the same.

In order to compress our model Φ^t_{big} we first initialize a model Φ^t composed of one feature extractor φ^t and a classifier $\mathcal{H}^t$. We initialize this model with Φ^{t-1} and train it on the incremental dataset with the following loss:

$$L = \eta \mathcal{L}_{\mathcal{H}^t} + (1 - \eta)\mathcal{L}_D$$

where η is a balancing weight, $\mathcal{L}_{\mathcal{H}^t}$ is the standard cross entropy loss computed on the outputs of $\mathcal{H}^t$, and $\mathcal{L}_D$ is a knowledge distillation loss using all the output logits of $\mathcal{H}^t_{big}$ as targets :

$$\mathcal{L}_D(x) = \sum_{c=1}^{C_t} -q_c^{\mathcal{H}^t_{big}}(x) \ \log(q_c^{\mathcal{H}^t}(x))$$

with $q_c^{\mathcal{H}^t}(x) = \frac{e^{o_c(x)/\tau}}{\sum_{i=1}^{C^t} e^{o_i(x)/\tau}}$ the softened softmax probability obtained from output node o_c of the model, τ a temperature hyper-parameter, and $q_c^{\mathcal{H}^t_{big}}(x)$ the same softened softmax probability but obtained from the outputs of the big model Φ^t_{big}.

Due to the imbalance of the incremental dataset, however, it was empirically observed that such a compression scheme performs better for new rather than past classes, leading to a catastrophic forgetting of previous classes. We therefore devised a new method called rehearsal mixup to improve the distillation of past classes.

Rehearsal mixup. Mixup [59] is a data augmentation strategy that was originally proposed to improve neural network generalization by learning better boundaries between classes. Mixup generates virtual training samples by doing random interpolations between images of each sampled minibatches. In incremental learning, however, since the incremental dataset contains mostly new classes, this has the effect of training mostly at the boundaries between new classes which further increases performances on the majority classes.

In order to solve the issue of new classes being distilled better than past classes we propose to use a variant of this mixup strategy that we call "rehearsal mixup". In contrast

to the conventional mixup method, we propose to mix images of the incremental dataset minibatches with images randomly sampled from the memory. Doing so forces each training sample to be an interpolation between an image of the new incremental dataset $\mathcal{D}^t$ and an image of the memory $\mathcal{M}_t$. This allows the compressed model to train more at the boundaries between past and new classes, while also effectively reducing the bias towards new classes inherent to the imbalanced incremental dataset. In order to further reduce this bias we also employ the weight alignment method explained in section 4.2.1 on the compressed model after the compression phase.

4.3 Experiments

This section provides extensive evaluation of our approach within two datasets that are widely used in class incremental learning, namely, CIFAR-100 and ImageNet-1000. We first compare the performance of our method against several state-of-the-art algorithms (see section 4.3.2) using a fixed size model as well as against DER without pruning [46], the method that attains the best performances by storing and making use of all previous feature extractors during incremental learning. Furthermore, in section 4.3.4, we conduct an ablation study so as to assess the contribution of each component of our method. Finally, we juxtapose the performance of several well known state-of-the-art distillation-based methods with and without the addition of our rehearsal mixup approach.

4.3.1 Experimental setup and implementation details

Datasets and protocols. The CIFAR-100 dataset [63] is composed of 32x32 pixels color images representing 100 classes with 500 training images and 100 evaluation images for each class. The ImageNet-1000 dataset [7] on the other hand is a large scale dataset with 1.2 million training images and 50,000 validation images of 1000 different categories. Following the standard practice [10], [12], [35], for CIFAR-100 we train on all 100 classes in 5 and 10 incremental steps of 20 and 10 classes respectively with a fixed rehearsal memory size of 2000 samples while for ImageNet-1000 we split all 1000 classes in 10 incremental steps of 100 classes and keep a rehearsal memory of 20000 samples.

Implementation details. Our method is implemented in PyTorch, following [10], [12], [35] we chose the 32-layer Resnet backbone architecture for CIFAR-100 dataset and 18-layer Resnet [8] for ImageNet-1000. We used the SGD optimizer with a momentum of 0.9, a weight decay of 0.0005 and use a batch size of 128 for CIFAR-100 and 256 for ImageNet-1000. We trained our models for 200 epochs, with a learning rate starting at 0.1 and gradually decaying to 0 with a cosine annealing schedule. The data augmentation applied to training images consists in random cropping, horizontal flip, AutoAugment [58] and normalization. For the compression phase we set τ to 4 and η to 0.1 according to other non-incremental compression methods [34], and the beta distribution parameter α used by mixup to 0.8.

Methods	5 steps			10 steps		
	#Params	Avg	Last	#Params	Avg	Last
Joint Training	0.47	-	72.96	0.47	-	72.96
iCaRL	0.47	62.48	49.40	0.47	56.33	39.35
BiC	0.47	66.25	53.08	0.47	61.9	46.59
WA	0.47	68.43	54.01	0.47	65.78	49.94
DER	2.3	72.39	**64.01**	4.7	71.54	**61.49**
Ours (FECIL)	0.47	**73.56**	63.61	0.47	**73.29**	60.09

Table 4.1 – Results on CIFAR-100 averaged over three different class orders. "Avg" and "Last" respectively represent the average accuracy during the entire incremental training and the accuracy obtained after the last incremental training step. "#Params" corresponds to the number of parameters used by the model at the end of the incremental training (in millions). We denote by "Joint training" the upper bound of performance obtained with a non-incremental training on the whole dataset.

4.3.2 Comparison with other methods

In order to properly evaluate the effectiveness of our compression phase we compare our method to DER without pruning [46], which is a dynamic model method that stores each feature extractor during the incremental steps and makes predictions using the concatenated output of all of them. Furthermore, we evaluate the effectiveness of our method by comparing it to other state-of-the-art methods that keep the model size fixed: iCaRL [10], BiC [12] and WA [35].

Tables 4.1 and 4.2 respectively summarize the results obtained on CIFAR-100 and ImageNet-1000. Firstly, we can see that DER consistently outperforms the fixed-size model methods by a large margin, demonstrating the better stability-plasticity trade-off of dynamic models that we presented in section 4.1.1. However, DER adds parameters to the model during each incremental steps so the number of parameters used at the end of the incremental training is very high compared to other methods. On the other hand, our method manages to reach DER performance and even surpass it in average accuracy on both CIFAR-100 and Imagenet-1000 while keeping the model size fixed by compressing it after each incremental step.

More specifically, compared to DER we can observe that in all benchmarks our method reaches higher average accuracies but around 1-2% lower accuracies at the last incremental step, while using 10 times less parameters. This result must, however, be tempered because while our method does indeed prevent the size of the model from growing with each incremental step like DER, it does use an expansion training step that temporarily doubles

Methods	#Params	ImageNet-1000 10 steps			
		top-1		top-5	
		Avg	Last	Avg	Last
Joint Training	11.2	-	69.73	-	89.08
iCaRL	11.2	38.4	22.7	63.7	44.0
BiC	11.2	62.73	50.1	83.80	72.70
WA	11.2	65.67	55.60	86.21	77.80
DER	112.2	68.84	**60.16**	88.17	**82.86**
Ours (FECIL)	11.2	**69.02**	58.40	**89.12**	81.72

Table 4.2 – Results on ImageNet-1000. The dataset was separated in 10 incremental steps of 100 classes to learn and we report the average and last top-1 and top-5 accuracy obtained. DER results are imported directly from [46].

the feature space and therefore doubles the number of parameters.

Furthermore, figure 4.4 shows the detail of the evolution of performances during the entire training process on the CIFAR-100 10 steps benchmark. This figure demonstrates an interesting point about our method, namely, that our compressed model exhibits slightly better performance than our dynamic model in the first incremental steps. This can be explained by our rehearsal-mixup procedure that we employ during the compression step that not only helps compressing past classes' information better but also improves the classification loss that we use together with the distillation loss. In fact, the performances of the compressed model only starts to be inferior to the dynamic model in the last few incremental steps due to information loss during compression resulting from the lack of past class data. This indicates that our rehearsal mixup based compression effectively manages to compress the model with minimal performance loss.

4.3.3 Qualitative comparison

In order to further demonstrate the effectiveness of our rehearsal mixup compression step we compare qualitatively the features of different classes in the internal representation of the model trained with different incremental methods. Specifically, we compare the feature space obtained with our FECIL approach, with the state-of-the-art fixed architecture method WA [35] and with DER without pruning, the dynamic model approach that stores and uses all incremental feature extractors.

In order to observe qualitatively the feature space of models we employ the T-SNE visualisation method introduced in 2.2. This T-SNE represents a non-linear 2D projection

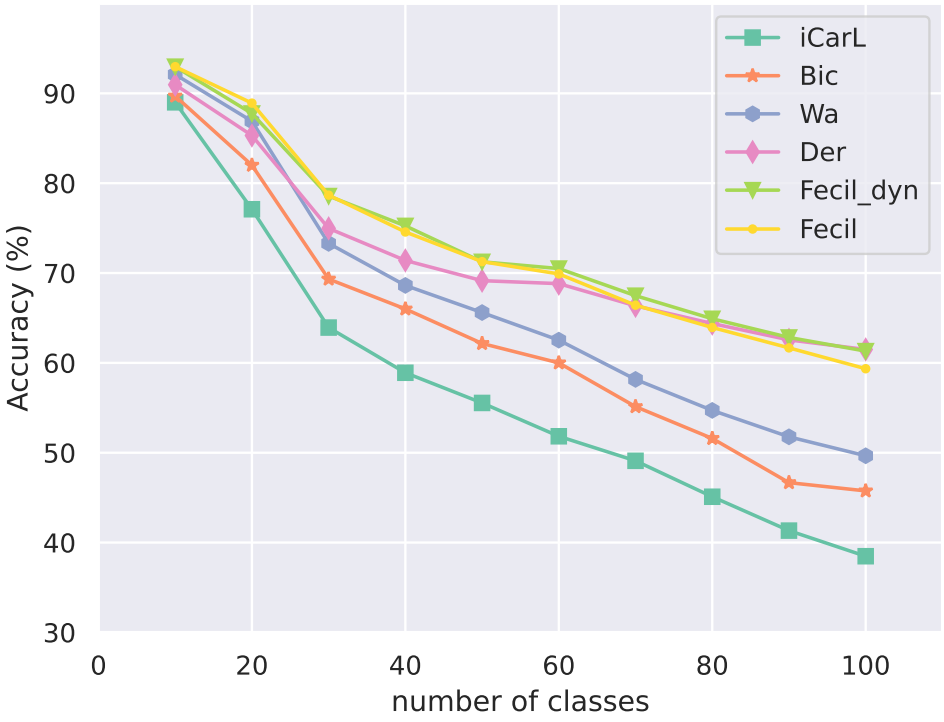


Figure 4.4 – Performance evolution on the CIFAR-100 10 steps benchmark. The mean of 3 runs is represented for each method, each with the same class orders and the top-1 accuracy (%) is reported after learning each task. Fecil_dyn represents the performance attained by our dynamic model before the compression step.

of the feature vectors obtained for all the training data of 10 classes each shown in a different color.

We show in figure 4.5 the features extracted at the same point during incremental learning for the 3 considered approaches on the CIFAR-100 10 steps benchmark. In this figure, the effect of catastrophic forgetting can be seen in the features extracted by WA, animals and plants are starting to get forgotten and the clusters related to those classes are becoming very spread out and not clearly separated. In the T-SNE obtained with DER, however, while not being very compact, these clusters are still well separated, which showcases the improved plasticity-stability trade-off of methods such as DER relying on dynamic models. Finally, in the T-SNE obtained with our method, the effectiveness of our rehearsal mixup compression step can be directly observed. Indeed, we can see a clear improvement in cluster separation compared to WA, the other fixed size model approach. In fact, we can see that the clusters are very close to DER in terms of quality of separation

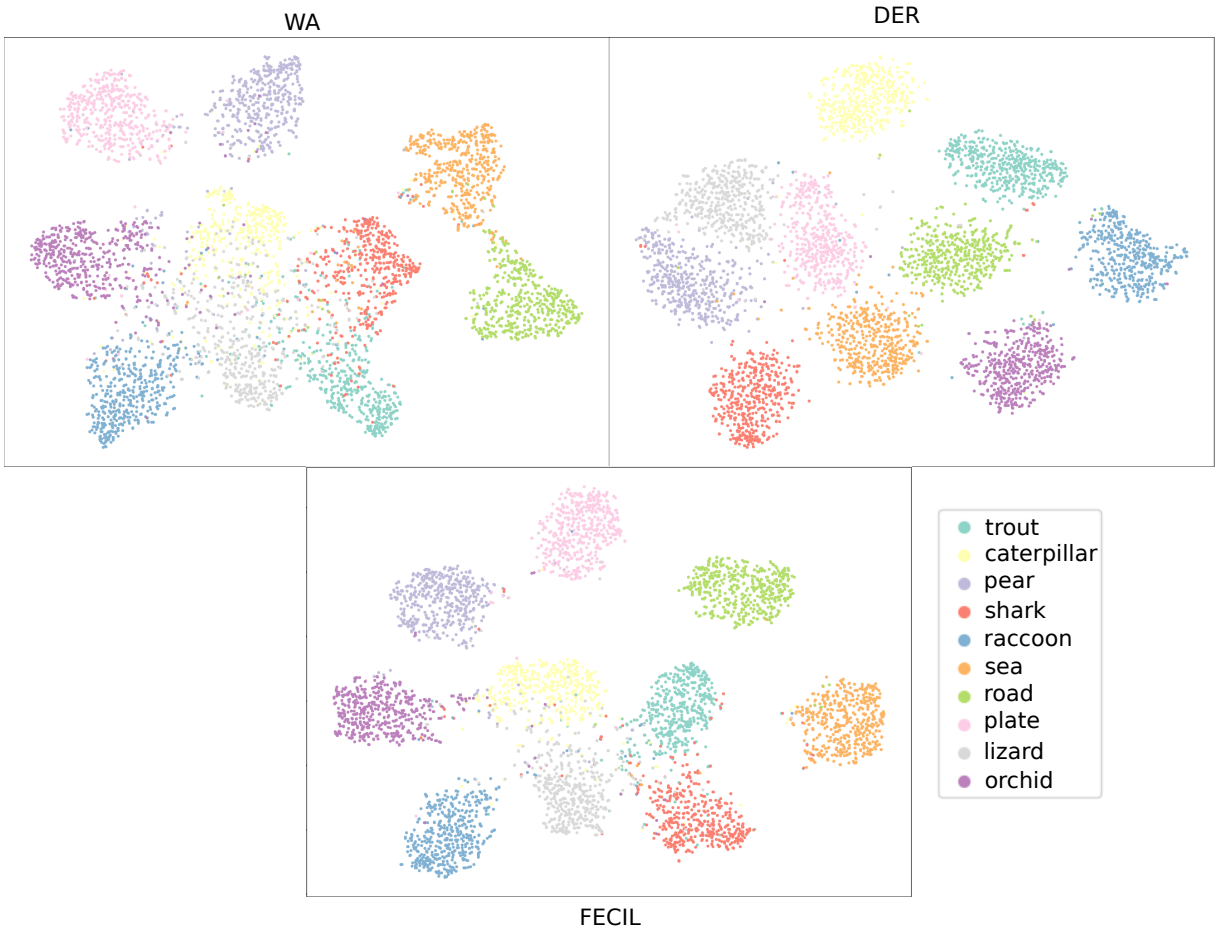


Figure 4.5 – 2D T-SNE visualisation of features extracted for 10 classes after the third incremental step on the CIFAR-100 10 steps benchmark. A qualitative comparison of those features when training with WA, DER, and FECIL is provided.

while making use of only one feature extractor.

4.3.4 Ablation study

We further evaluate our method and the contribution of each specific component by conducting an exhaustive ablation study on the 10 step CIFAR-100 benchmark. Specifically, we evaluate the contributions of the autoAugment[58] and mixup[59] data augmentations as well as compare the standard mixup procedure to our rehearsal-mixup.

We report the average incremental accuracy as well as the accuracy after the last incremental step for all ablations in table 4.3. We denote by "$\mathcal{H}_{big}$ Wa" the ablation of the weight alignment component introduced in 4.2.1 that we use on the classifier of the dynamic model before the compression step. Furthermore we denote with "Mixup"

and "R-Mixup" respectively the standard mixup augmentation and our rehearsal mixup procedure.

	$\mathcal{H}_{big}$ Wa	AutoAug	Mixup	R-Mixup	Avg	Last
Ablations					64.08	46.68
	✓				67.17	51.26
	✓	✓			70.79	56.26
	✓		✓		68.27	53.5
	✓			✓	69.74	55.35
	✓	✓	✓		70.31	56.14
Full	✓	✓		✓	73.29	60.09

Table 4.3 – Ablation of different key components of our method. The average incremental accuracy as well as the accuracy after the last incremental step are reported for each ablation. All experiments were done on the CIFAR-100 dataset separated in 10 incremental steps each adding new 10 classes.

As can be seen with the ablation of $\mathcal{H}_{big}$ Wa, removing the bias from the dynamic model prior to compression greatly improves the performance of the compressed model. Indeed, this result makes sense since we use a knowledge distillation loss that compares classifier outputs to compress the dynamic model, therefore removing the bias before the compression avoids transferring it to the compressed model.

Moreover, comparing our full method with the ablation of the different data augmentations shows that significant improvements of almost 9% accuracy after the last incremental step are obtained with their addition. These data augmentations techniques are fairly inexpensive for the training of incremental models, yet lead to the most gains in our method, suggesting that the usual data augmentations used by most incremental studies are not sufficient in incremental learning where data from previous classes are scarce.

Furthermore, these ablation studies show that the classical mixup augmentation and autoAugment lead to similar gains and autoAugment is even superior by a small margin, but most importantly that they do not benefit from each other. Indeed, with autoAugment and Mixup alone the average incremental accuracy is respectively 70.79% and 68.27% and with both data augmentations the accuracy stays around 70%. This can be explained by the fact that both augmentations have the same effect of improving variability of the incremental dataset, which is composed mostly of new classes data, and therefore

both augmentations impact more new classes than past classes. This leads to both data augmentations improving mostly the training of new classes which gets overshadowed by the forgetting of past classes occurring during incremental steps.

However, with the addition of our rehearsal-mixup strategy, we can observe completely different results. While the standard mixup procedure was previously shown to reduce forgetting [71], we can see that our rehearsal-mixup increases the last accuracy obtained with mixup from 53.5% to 55.35% demonstrating that specifically targeting decision boundaries between new and past classes is more beneficial and prevents forgetting even more. Moreover, we can notice that unlike standard mixup, rehearsal-mixup and autoAugment are complementary. In fact, they greatly benefit from each other, as separately they each improve the average accuracy by about 3% but when used together the accuracy goes up by more than 6%, from 67.17% to 73.29%.

We further investigate the effectiveness of rehearsal-mixup in the following section where we add it to several well known knowledge distillation based incremental methods to showcase its ability to improve distillation and prevent forgetting.

4.3.5 Mixup rehearsal added to other methods

Based on the aforementioned observations and as an additional test of our approach, we investigated the addition of our data augmentations rehearsal mixup and AutoAugment to other well known state-of-the-art methods. The purpose of our rehearsal mixup method is to improve the distillation of our compression step by focusing on the boundaries between new and old classes to reduce forgetting. For this reason, we chose three distillation based state-of-the-art methods to conduct this experiment. We evaluate the addition of each data augmentation on the CIFAR-100 10 steps benchmark and report both the average and last incremental accuracy obtained in table 4.4.

As can be seen in this table, despite having trivial cost both in terms of training time and memory needed, the addition of both of these components greatly benefits all of the tested methods. The standard practice [10] for state-of-the-art incremental methods has been to use only random crop and random horizontal flip data augmentations, following the classical training procedure for ResNet models [8]. However, we believe that in the incremental learning setup where data about past classes only comes from a small fixed size rehearsal memory, more data augmentations should be used so as to alleviate the forgetting of previous classes. The addition of AutoAugment improves the accuracy of all tested methods by over 3% by simply adding many different augmentations like rotations

Methods	Base		+ R-mixup		+ R-mixup +AutoAug	
	Avg	Last	Avg	Last	Avg	Last
iCaRL	56.33	39.35	59.11	41.08	**60.23**	**43.41**
BiC	61.90	46.59	64.72	50.26	**67.41**	**51.63**
WA	65.78	49.94	67.43	51.49	**69.51**	**53.68**
Ours	66.93	49.51	70.79	56.26	**73.29**	**60.09**

Table 4.4 – Addition to other distillation based methods. R-mixup and AutoAug respectively represent the addition of rehearsal mixup and AutoAugment to each of the algorithms.

and color changes to increase training data variability, which demonstrates that the usual augmentations may be insufficient for incremental training.

Finally, we observe for all algorithms very similar gains from rehearsal-mixup and AutoAugment as for our method, showcasing once more that rehearsal-mixup and AutoAugment have complementary effects that help learn new classes better as well as reduce forgetting of previous classes.

4.4 Conclusion

In this chapter, we described FECIL, a novel two-stage training procedure that we developed for class incremental learning during the third year of the thesis. Our method first proceeds by expanding the features of the model in order to accommodate new classes without forgetting past ones; It then compresses it back to its original size in order to keep the model size fixed over the course of the entire incremental training process. Specifically, we introduce a rehearsal-mixup procedure in order to compress past classes' information well even with limited available data.

We have conducted comprehensive experiments to evaluate our method, against both fixed-size model and dynamic model methods that add parameters over the incremental steps to attain a better plasticity-stability tradeoff at the cost of a growing number of parameters. Our results demonstrated that our method attains the highest performance among dynamic models while keeping the size of the model fixed by compressing it after each incremental step. More specifically, we showed that the data augmentations autoAugment and rehearsal-mixup have complementary effects that allow our model to learn new classes better as well as reduce forgetting of past ones.

Finally, we demonstrated that our rehearsal mixup procedure can be plugged into other knowledge distillation based state-of-the-art methods to reduce forgetting and improve their average incremental accuracy by about 3% at negligible extra cost.

Conclusion

In this final chapter of our thesis the findings and contributions of our research will be summarized, discussed and some directions for future work will be outlined.

4.5 General discussion

In this thesis entitled "Mitigating catastrophic forgetting via feature transfer and knowledge consolidation for deep class-incremental learning", intensive studies were realized on the features extracted and used by neural networks trained incrementally.

Experiments described in chapter 2 shed light on the massive negative impact of catastrophic forgetting on the feature space created by convolutional neural networks which hinders the downstream classification performances. Based on these observations, in chapter3, a joint contrastive and incremental method was proposed to specifically target catastrophic forgetting at a feature level as well as improve the general representation of the model. Finally, in the chapter 4, we investigate how the emerging concept of dynamic neural networks can be leveraged to further improve the features extracted incrementally, while also significantly reducing the computation overhead compared to our previous method.

4.6 Directions for future Works

The performances of incremental methods are getting closer and closer to those of standard joint training of all classes, however, a gap still remains. While the root cause of this performance gap is catastrophic forgetting, many different components of incremental methods could still be improved in order to alleviate forgetting even more. In the following paragraphs we propose some improvements for our methods as well as some general directions for future work that we believe could lead to new successful incremental approaches in the coming years.

Distillation improvements. Numerous studies, including our own, have highlighted the effectiveness of knowledge distillation in mitigating catastrophic forgetting. In our research on contrastive learning, we found that incorporating a feature-based distillation loss can enhance the overall quality of distillation and significantly decrease forgetting. Moreover, we observed that even simpler data augmentation techniques, such as our rehearsal mixup method, can effectively improve the quality of the standard probability-based distillation. This suggests that combining contrastive distillation or other upcoming state-of-the-art feature-based knowledge distillation methods with our rehearsal-mixup based data-augmentation could improve distillation even further.

incremental contrastive learning. successful contrastive-based incremental approaches have already surfaced in recent literature [44], [57], [75], showcasing the promise of contrastive techniques for incremental learning. Contrastive methods are known to be particularly effective for learning discriminative representations in unsupervised and semi-supervised contexts, and these settings bear substantial resemblances to the incremental learning scenario. For instance, semi-supervised learning frequently deals with imbalanced datasets, often featuring a surplus of unlabeled data in comparison to labeled instances, mirroring the distribution of data in incremental datasets. Furthermore, incremental training requires training with limited data about past classes, a challenge analogous to the scarcity of labels encountered in semi-supervised and unsupervised learning. These parallels strongly indicate that integrating contrastive methods into incremental learning could be a highly promising direction for further research exploration.

Incremental architectures. The concept of dynamic architectures designed for continual learning have gained popularity recently, particularly due to DER's pioneering work [46], which demonstrated a superior trade-off between adaptability and stability compared to existing state-of-the-art methods. This superior performance of dynamic architectures came at a cost in terms of parameter growth, that an emerging line of research focuses on solving. Some approaches have been proposed based on vision transformers-based incremental models [76], [77] that were shown to not only significantly reduce this growth but also learn class-agnostic features that revealed particularly useful for continual learning [71], [78]. Similarly, various methods like the one presented in chapter 4 have been introduced to compress models after each incremental step to avoid parameter growth with minimal performance drop [67], [79]. These approaches require an added compres-

sion training step, however, a very interesting future improvement could be made by compressing the model while it is adapting to new classes in order to remove this added compression step and reduce training time overhead.

Generative replay. With the advances of transformer-based architectures and particularly masked auto-encoders [77], the line of research about generative replay also seems particularly prone to great developments in the coming years. Indeed, a few years ago the performances of generative models represented the bottleneck of generative replay methods that struggled to reach standard rehearsal performances, however a recent method based on masked autoencoders (MAEs) demonstrated impressive performances with partial generative replay [78]. There masked autoencoder architecture displayed better performances than concurrent standard rehearsal state-of-the-art methods by storing only small patches of pixels instead of full images allowing them to store much more exemplars per class for the same storage cost.

BIBLIOGRAPHY

[1] M. McCloskey and N. J. Cohen, « Catastrophic interference in connectionist networks: the sequential learning problem », *in Psychology of Learning and Motivation*, G. H. Bower, Ed., vol. 24, Academic Press, Jan. 1, 1989, pp. 109–165.

[2] R. M. French, « Catastrophic forgetting in connectionist networks », *Trends in Cognitive Sciences*, vol. 3, *4*, pp. 128–135, Apr. 1, 1999, Publisher: Elsevier.

[3] S. Linnainmaa, « Taylor expansion of the accumulated rounding error », *BIT*, vol. 16, *2*, pp. 146–160, Jun. 1976.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, « Learning representations by back-propagating errors », *Nature*, vol. 323, *6088*, pp. 533–536, Oct. 1986, Number: 6088 Publisher: Nature Publishing Group.

[5] Y. LeCun, B. Boser, J. Denker, *et al.*, « Handwritten digit recognition with a back-propagation network », *in Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann, 1989.

[6] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, « A general reinforcement learning algorithm that masters chess, shogi, and go through self-play », *Science*, vol. 362, *6419*, pp. 1140–1144, Dec. 7, 2018, Publisher: American Association for the Advancement of Science Section: Report.

[7] O. Russakovsky, J. Deng, H. Su, *et al.*, « ImageNet large scale visual recognition challenge », *Int J Comput Vis*, vol. 115, *3*, pp. 211–252, Dec. 1, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, « Deep residual learning for image recognition », *arXiv:1512.03385 [cs]*, Dec. 10, 2015. arXiv: 1512.03385.

[9] S. Thrun and L. Pratt, *Learning to Learn*. Springer Science & Business Media, Dec. 6, 2012, 346 pp., Google-Books-ID: X_jpBwAAQBAJ.

[10] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, « iCaRL: incremental classifier and representation learning », presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[11] Z. Li and D. Hoiem, « Learning without forgetting », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, *12*, pp. 2935–2947, Dec. 2018.

[12] Y. Wu, Y. Chen, L. Wang, *et al.*, « Large scale incremental learning », presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 374–382.

[13] E. Belouadah and A. Popescu, « IL2m: class incremental learning with dual memory », presented at the Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 583–592.

[14] E. Belouadah and A. Popescu, « ScaIL: classifier weights scaling for class incremental learning », presented at the The IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 1266–1275.

[15] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, « Riemannian walk for incremental learning: understanding forgetting and intransigence », presented at the Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 532–547.

[16] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, « Learning a unified classifier incrementally via rebalancing », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 831–839.

[17] F. M. Castro, M. J. Marin-Jimenez, N. Guil, C. Schmid, and K. Alahari, « End-to-end incremental learning », presented at the Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 233–248.

[18] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, « Class-incremental learning: survey and performance evaluation », *arXiv:2010.15277 [cs]*, Oct. 28, 2020.

[19] E. Belouadah and A. Popescu, « Deesil: deep-shallow incremental learning. », *in Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[20] G. Boukli Hacene, V. Gripon, N. Farrugia, M. Arzel, and M. Jezequel, « Transfer incremental learning using data augmentation », *Applied Sciences*, vol. 8, *12*, p. 2512, Dec. 2018.

[21] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, and Q. Sun, « Mnemonics training: multi-class incremental learning without forgetting », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12 245–12 254.

[22] R. Kemker and C. Kanan, « FearNet: brain-inspired model for incremental learning », *arXiv:1711.10563 [cs]*, Feb. 23, 2018. arXiv: 1711.10563.

[23] H. Shin, J. K. Lee, J. Kim, and J. Kim, « Continual learning with deep generative replay », *arXiv:1705.08690 [cs]*, Dec. 11, 2017. arXiv: 1705.08690.

[24] Y. Wu, Y. Chen, L. Wang, *et al.*, « Incremental classifier learning with generative adversarial networks », *arXiv:1802.00853 [cs]*, Feb. 2, 2018. arXiv: 1802.00853.

[25] X. Liu, C. Wu, M. Menta, *et al.*, « Generative feature replay for class-incremental learning », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 226–227.

[26] Y. Xiang, Y. Fu, P. Ji, and H. Huang, « Incremental learning using conditional adversarial networks », presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6619–6628.

[27] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, « Lifelong GAN: continual learning for conditional image generation », presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2759–2768.

[28] M. Solinas, S. Rousset, R. Cohendet, *et al.*, « Beneficial effect of combined replay for continual learning: » *in Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, 2021, pp. 205–217.

[29] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, « Overcoming catastrophic forgetting in neural networks », *PNAS*, vol. 114, *13*, pp. 3521–3526, Mar. 28, 2017, Publisher: National Academy of Sciences Section: Biological Sciences.

[30] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. López, and A. D. Bagdanov, « Rotate your networks: better weight consolidation and less catastrophic forgetting », *in 2018 24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018, pp. 2262–2268.

[31] F. Zenke, B. Poole, and S. Ganguli, « Continual learning through synaptic intelligence », *in International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 17, 2017, pp. 3987–3995.

[32] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, « Memory aware synapses: learning what (not) to forget », presented at the Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 139–154.

[33] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, « Model compression », *in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '06, Association for Computing Machinery, Aug. 20, 2006, pp. 535–541.

[34] G. Hinton, O. Vinyals, and J. Dean, « Distilling the knowledge in a neural network », *arXiv:1503.02531 [cs, stat]*, Mar. 9, 2015. arXiv: 1503.02531.

[35] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, « Maintaining discrimination and fairness in class incremental learning », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13 208–13 217.

[36] J. Zhang, J. Zhang, S. Ghosh, *et al.*, « Class-incremental learning via deep model consolidation », presented at the The IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 1131–1140.

[37] H. Jung, J. Ju, M. Jung, and J. Kim, « Less-forgetting learning in deep neural networks », *arXiv:1607.00122 [cs]*, Jul. 1, 2016. arXiv: 1607.00122.

[38] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, « Learning without memorizing », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5138–5146.

[39] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, « Small-task incremental learning », *arXiv:2004.13513 [cs]*, Apr. 28, 2020.

[40] M. Buda, A. Maki, and M. A. Mazurowski, « A systematic study of the class imbalance problem in convolutional neural networks », *Neural Networks*, vol. 106, pp. 249–259, Oct. 2018. arXiv: 1710.05381.

[41] G. Van Horn, O. Mac Aodha, Y. Song, *et al.*, « The INaturalist species classification and detection dataset », presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8769–8778.

[42] B. Mac Namee, P. Cunningham, S. Byrne, and O. I. Corrigan, « The problem of bias in training data in regression problems in medical decision support », *Artificial Intelligence in Medicine*, vol. 24, *1*, pp. 51–70, Jan. 1, 2002.

[43] P. Chan, W. Fan, A. Prodromidis, and S. Stolfo, « Distributed data mining in credit card fraud detection », *IEEE Intelligent Systems and their Applications*, vol. 14, *6*, pp. 67–74, Nov. 1999, Conference Name: IEEE Intelligent Systems and their Applications.

[44] Z. Mai, R. Li, H. Kim, and S. Sanner, « Supervised contrastive replay: revisiting the nearest class mean classifier in online class-incremental continual learning », *arXiv:2103.13885 [cs]*, Apr. 23, 2021. arXiv: 2103.13885.

[45] C. Luo, J. Zhan, X. Xue, L. Wang, R. Ren, and Q. Yang, « Cosine normalization: using cosine similarity instead of dot product in neural networks », *in Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, pp. 382–391.

[46] S. Yan, J. Xie, and X. He, « DER: dynamically expandable representation for class incremental learning », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[47] L. Van der Maaten and G. Hinton, « Visualizing data using t-SNE », *Journal of Machine Learning Research 9*, 2008.

[48] S. Zagoruyko and N. Komodakis, « Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer », *arXiv: 1612.03928 [cs]*, Feb. 12, 2017.

[49] F. Tung and G. Mori, « Similarity-preserving knowledge distillation », presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1365–1374.

[50] Y. Tian, D. Krishnan, and P. Isola, « Contrastive multiview coding », *arXiv: 1906.05849 [cs]*, Dec. 18, 2020.

[51] Z. Fang, J. Wang, L. Wang, L. Zhang, Y. Yang, and Z. Liu, « SEED: self-supervised distillation for visual representation », *arXiv:2101.04731 [cs]*, Apr. 15, 2021. arXiv: 2101.04731.

[52] L. Chen, D. Wang, Z. Gan, J. Liu, R. Henao, and L. Carin, « Wasserstein contrastive representation distillation », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[53] P. Zhou, L. Mai, J. Zhang, N. Xu, Z. Wu, and L. S. Davis, « M2kd: multi-model and multi-level knowledge distillation for incremental learning », *arXiv:1904.01769 [cs]*, Apr. 3, 2019. arXiv: 1904.01769.

[54] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, « A simple framework for contrastive learning of visual representations », *in International Conference on Machine Learning*, PMLR, Nov. 21, 2020, pp. 1597–1607.

[55] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, « Momentum contrast for unsupervised visual representation learning », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.

[56] P. Khosla, P. Teterwak, C. Wang, *et al.*, « Supervised contrastive learning », *arXiv: 2004.11362 [cs, stat]*, Mar. 10, 2021.

[57] H. Cha, J. Lee, and J. Shin, « Co2l: contrastive continual learning », presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9516–9525.

[58] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, *AutoAugment: learning augmentation policies from data*, Apr. 11, 2019. arXiv: 1805.09501[cs,stat].

[59] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. « Mixup: beyond empirical risk minimization », arXiv.org. (Oct. 25, 2017).

[60] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, « Cutmix: regularization strategy to train strong classifiers with localizable features », *in Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[61] P. Goyal, P. Dollár, R. Girshick, *et al.*, *Accurate, large minibatch SGD: training ImageNet in 1 hour*, Apr. 30, 2018. arXiv: 1706.02677[cs].

[62] G. Xu, Z. Liu, X. Li, and C. C. Loy, « Knowledge distillation meets self-supervision », *in Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Springer International Publishing, 2020, pp. 588–604.

[63] A. Krizhevsky, « Learning multiple layers of features from tiny images », p. 60, 2009.

[64] S. Golkar, M. Kagan, and K. Cho, *Continual learning via neural pruning*, Mar. 11, 2019.

[65] C. Fernando, D. Banarse, C. Blundell, *et al.*, *PathNet: evolution channels gradient descent in super neural networks*, Jan. 30, 2017. arXiv: 1701.08734[cs].

[66] Z. Li, C. Zhong, S. Liu, R. Wang, and W.-S. Zheng, *Preserving earlier knowledge in continual learning with the help of all previous feature extractors*, Apr. 28, 2021. arXiv: 2104.13614[cs].

[67] F.-Y. Wang, D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, *FOSTER: feature boosting and compression for class-incremental learning*, Jul. 20, 2022. arXiv: 2204.04662[cs].

[68] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, and C.-S. Chen, « Compacting, picking and growing for unforgetting continual learning », *in Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[69] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, *et al.*, *Progressive neural networks*, Oct. 22, 2022. arXiv: 1606.04671[cs].

[70] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, « Overcoming catastrophic forgetting with hard attention to the task », *in Proceedings of the 35th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 3, 2018, pp. 4548–4557.

[71] A. Douillard, A. Ramé, G. Couairon, and M. Cord, « DyTox: transformers for continual learning with DYnamic TOken eXpansion », arXiv, arXiv:2111.11326, Mar. 27, 2022.

[72] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, *Mixup: beyond empirical risk minimization*, Apr. 27, 2018. arXiv: 1710.09412[cs,stat].

[73] V. Verma, A. Lamb, C. Beckham, *et al.*, « Manifold mixup: better representations by interpolating hidden states », *in Proceedings of the 36th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, May 24, 2019, pp. 6438–6447.

[74] H.-P. Chou, S.-C. Chang, J.-Y. Pan, W. Wei, and D.-C. Juan, « Remix: rebalanced mixup », *in Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds., ser. Lecture Notes in Computer Science, Springer International Publishing, 2020, pp. 95–110.

[75] Q. Ferdinand, B. Clement, Q. Oliveau, G. Le Chenadec, and P. Papadakis, « Attenuating catastrophic forgetting by joint contrastive and incremental learning », presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3782–3789.

[76] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: transformers for image recognition at scale*, Jun. 3, 2021. arXiv: 2010.11929[cs].

[77] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, *Masked autoencoders are scalable vision learners*, Dec. 19, 2021. arXiv: 2111.06377[cs].

[78] J.-T. Zhai, X. Liu, A. D. Bagdanov, K. Li, and M.-M. Cheng, *Masked autoencoders are efficient class incremental learners*, Aug. 23, 2023. arXiv: 2308.12510[cs].

[79] X. Chen and X. Chang, *Dynamic residual classifier for class incremental learning*, Aug. 25, 2023. arXiv: 2308.13305[cs].

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

ENSTA BRETAGNE

Titre : réduction de l'oubli catastrophique à l'aide de méthodes de distillation et de transfert de caractéristiques pour l'apprentissage incremental profond

Mot clés : apprentissage incrémental, oubli catastrophique, distillation de connaissances, réseau de neurone convolutif, apprentissage supervisé

Résumé : Les méthodes d'apprentissage profond actuelles sont conçues de manière statique, avec un certain nombre de classes à reconnaître connu et prédéfini à l'avance. L'apprentissage incrémental, en revanche, étudie le scénario plus réaliste dans lequel certaines classes inconnues peuvent arriver au fur et à mesure et le modèle doit s'adapter et apprendre à reconnaître ces nouvelles catégories. Malheureusement, les réseaux neuronaux profonds entrainés de cette manière souffrent d'un oubli catastrophique, provoquant une dégradation significative des performances sur les classes précédemment rencontrées.

Diverses méthodes ont été explorées pour atténuer ce problème en rectifiant la couche de classification des modèles incrémentaux. Cependant, un aspect tout aussi important réside dans la dégradation de la qualité des caractéristiques que le modèle extrait des images. Cette thèse se concentre spécifiquement sur l'exploration de diverses approches visant à améliorer leur pouvoir discriminant, facilitant ainsi l'adaptation à de nouvelles classes tout en préservant les caractéristiques discriminatives des classes précédentes lors de l'entraînement incrémental.

Plus précisément, deux méthodes ont été proposées et rigoureusement évaluées, atteignant des performances comparables où supérieures à l'état de l'art. La première approche présentée explore l'utilisation de méthodes contrastives pendant l'apprentissage incrémental afin d'améliorer les caractéristiques extraites par le modèle, tandis que la seconde utilise une stratégies d'expansion et de compression de la partie responsable de l'extraction des caractéristiques dans le réseau de neurone afin de réduire significativement l'oubli.

Title: Mitigating catastrophic forgetting via feature transfer and knowledge consolidation for deep class-incremental learning

Keywords: incremental learning, catastrophic forgetting, knowledge distillation, convolutional neural network, supervised learning

Abstract: Deep learning methods are designed for closed-set recognition, where a predefined set of known classes is assumed. However, in real-world scenarios, open-set recognition is more realistic, allowing for the possibility of encountering unknown or novel classes during testing. Class-incremental learning specifically addresses this problem by focusing on continuously improving models through the incorporation of new categories over time. Unfortunately, deep neural networks trained in this manner suffer from catastrophic forgetting, resulting in significant performance degradation on previously encountered classes.

While various methods have been explored to alleviate this issue by rectifying the classification layer of deep incremental models, an equally important aspect resides in the degradation of feature quality, which can impede downstream classification performance. This thesis specifically focuses on investigating diverse approaches to enhance feature quality, facilitating adaptation to new classes while preserving discriminative features for past classes during incremental training.

Specifically two methods have been proposed and rigorously evaluated on widely established benchmarks, attaining performances either comparable or superior to the state-of-the-art . The first approach presented investigates the use of contrastive methods during incremental learning in order to improve the features extracted by incremental models while the second one uses an expansion and compression scheme to greatly reduce the forgetting happening at a feature level.