Apprentissage incrémental semi-supervisé pour les applications de vision artificielle

Alexis Lechat

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Apprentissage incrémental semi-supervisé pour les applications de vision artificielle

Présentée et soutenue par
ALEXIS LECHAT

Thèse soutenue le 23/12/2021 devant le jury composé de		
M. MICHEL CRUCIANU	Professeur des universités, Conservatoire Nal des Arts et Métiers	Rapporteur du jury
MME MATHILDE MOUGEOT	Professeur des universités, École Normale Supérieure Paris-Saclay	Rapporteur du jury
M. STÉPHANE HERBIN	Chargé de recherche HDR, ONERA	Membre du jury
MME SYLVAINE PICARD	Responsable industriel, Safran	Membre du jury
M. DAVID FILLIAT	Professeur des universités, ENSTA	Président du jury
M. FREDERIC JURIE	Professeur des universités, Université Caen Normandie	Directeur de thèse

Thèse dirigée par FREDERIC JURIE, Groupe de recherche en informatique, image, automatique et instrumentation

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC
Laboratoire de recherche en sciences du numérique

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Apprentissage incrémental semi-supervisé pour les applications de vision artificielle

Présentée et soutenue par

Alexis LECHAT

Thèse soutenue le 23/12/2021
devant le jury composé de

M MICHEL CRUCIANU	Professeur des universités, Conservatoire National des Arts et Métiers	Rapporteur du jury
MME MATHILDE MOUGEOT	Professeur des universités, École Normale Supérieure Paris-Saclay	Rapporteur du jury
M DAVID FILLIAT	Professeur des universités, ENSTA	Membre du jury
MME SYLVAINE PICARD	Responsable industriel, Safran	Membre du jury
M STÉPHANE HERBIN	Chargé de recherche HDR, ONERA	Membre du jury
M FRÉDÉRIC JURIE	Professeur des universités, Univ. Caen Normandie, détaché auprès de Safran	Directeur de thèse

Thèse dirigée par FRÉDÉRIC JURIE, Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC

ed 590 MIIS
Mathématiques, Information, Ingénierie des Systèmes

Normandie Université

ONERA
THE FRENCH AEROSPACE LAB

Table des Matières

Table des figures

Liste des tableaux

Liste des abréviations

AAE *Adversarial Autoencoder*

AE Auto-encodeur

AR Apprentissage des Représentations

CNN *Convolutional Neural Network*

DC Distillation de Connaissances

DNN *Deep Neural Network*

EB Entraînement Batch

FC *Fully-Connected*

GAN *Generative Adversarial Networks*

i.i.d. indépendemment et identiquement distribué

IC Incrémental de Classes

ID Incrémental de Domaines

IT Incrémental de Tâches

k-NN *k-nearest neighbours*

MLP *Multilayer Perceptron*

OC Oubli Catastrophique

OvA *One-versus-All*

PL *Pseudo-Labeling*

PLCiL *Pseudo-Labeling for Class-incremental Learning*

QRV Questions-Réponses Visuelles

RVB Rouge Vert Bleu

VAE *Variational Autoencoder*

VT Vérité terrain

Chapitre 1

Introduction

Contents

1.1 Contexte

La vision par ordinateur est une domaine de l'intelligence artificielle visant à concevoir des ordinateurs dotés d'une compréhension visuelle de haut niveau. En pratique, elle développe des systèmes capables d'automatiser les tâches de traitement et d'analyse d'images. Ce domaine regroupe une multitude d'applications : reconnaissance d'objets, description d'images, segmentation sémantique, questions-réponses visuelles, super-résolution, suivi d'objets dans les vidéos, etc. Ces applications nécessitent de s'appuyer sur des connaissances du domaine visuel.

Les progrès en vision par ordinateur ont principalement été rendus possibles grâce aux approches d'apprentissage profond. Ces algorithmes exploitent des réseaux de neurones artificiels profonds où des millions de paramètres entrent en jeu afin de traiter une tâche cible. Le succès de ces approches est directement lié au développement de l'ère numérique où la croissance des moyens matériels et l'explosion des quantité de données ont rendu possible le déploiement et l'entraînement de ces algorithmes.

Les méthodes les plus fiables pour l'entraînement d'un réseau de neurones sont dites supervisées. On présente un ensemble d'exemples de la tâche à effectuer et via une phase d'apprentissage basée sur des algorithmes d'optimisation statistique, le modèle apprend à la reproduire. Les capacités de l'algorithme final dépendent grandement de sa structure et du nombre de ses paramètres mais surtout du nombre et de la qualité des exemples décrivant la tâche.

Les exemples utilisés pour l'apprentissage d'une tâche constituent l'ensemble, ou *dataset*, d'entraînement. Pour beaucoup d'applications, la fabrication d'un jeu de données adapté à l'apprentissage est le principal facteur limitant. La supervision requiert, pour chaque image, une vérité terrain qui est la solution au problème cible. La création de ces vérités terrains repose sur une tâche d'annotation plus ou moins complexe. Pour la reconnaissance d'objet par exemple, la solution attendue est une unique catégorie pour toute l'image alors que la segmentation sémantique nécessite un masque indiquant la classe de chaque pixel. L'annotation est généralement un processus manuel coûteux en temps et en main d'oeuvre.

Le scénario idéal pour un apprentissage profond supervisé est celui où une grande quantité de données annotées est disponible afin de représenter de la manière la plus exhaustive possible la tâche visuelle. Une fois les données collectées, l'optimisation du modèle se fait en itérant à de multiples reprises sur le jeu d'entraînement jusqu'à convergence vers une solution satisfaisante. Avec ce paradigme d'entraînement couramment utilisé, le réseau de neurones obtenu possède un état fini. Ses paramètres, et donc sa connaissance, ne seront plus amenés à être mis à jour. Le cadre de développement des méthodes d'apprentissage profond peut se résumer en trois étapes consécutives : collecte et annotation des données,

entraînement du modèle, déploiement pour application.

Les tendances récentes de l'apprentissage profond cherchent à proposer des alternatives à ce paradigme supervisé. Cette thèse s'inscrit dans cette démarche en se focalisant sur deux enjeux en particulier : proposer des modèles évolutifs capables de mettre à jour leurs connaissances tout au long de leur vie et pouvoir s'affranchir de l'étape d'annotation.

1.2 Domaines

Deux domaines majeurs sont abordés dans cette thèse, celui de l'apprentissage incrémental et celui de l'apprentissage des représentations visuelles.

L'apprentissage incrémental, aussi appelé apprentissage continu, s'intéresse au développement de solutions évolutives. Contrairement aux approches habituelles qui s'orientent vers une unique phase d'apprentissage hors ligne sur un domaine fini, un dispositif incrémental doit être capable d'enrichir sa connaissance et ses compétences tout au long de son cycle de vie.

L'apprentissage des représentations visuelles repose sur un concept fondamental : des applications de vision, même différentes, utilisent potentiellement un socle commun de connaissances visuelles. De ce fait, des représentations apprises sur une tâche peuvent servir à faciliter l'apprentissage d'une nouvelle.

Pour cela, la tendance est notamment à l'apprentissage non-supervisé de représentations visuelles. Ce sous-domaine est justifié par la simplicité d'accès à de larges quantités données brutes souvent synonymes de grande diversité visuelle. Les approches non-supervisées proposent des moyens d'exploiter ces données sans avoir besoin de les annoter manuellement. Ces solutions sont de ce fait moins coûteuses à mettre en place. De plus, apprendre des représentations de qualité qui se transfèrent bien à la tâche principale, supervisée, permet généralement de limiter la dépendance aux exemples de cette dernière. De ce fait, un apprentissage des représentations permet de réduire le besoin en annotations et même de traiter des problèmes qui auparavant étaient trop pauvres en exemples d'entraînement.

Dans nos travaux, nous nous intéressons en particulier au cas où le modèle cherche à exploiter simultanément des données annotées et non-annotées. Ce cas de figure est appelé apprentissage semi-supervisé.

L'application de vision artificielle principalement étudiée dans nos contributions est la reconnaissance d'objet. C'est une tâche de classification où l'algorithme doit prédire la catégorie de l'objet représenté par l'image. Ce problème est une référence en apprentissage profond en raison de la simplicité de sa formulation et de la disponibilité de grands ensembles de données annotés.

Nos travaux ont aussi abordé deux autres applications : les systèmes de questions-réponses visuelles et la classification multi-label.

1.3 Objectif de la thèse

Notre principal axe de recherche s'intéresse à l'apprentissage incrémental. Ce paradigme d'entraînement propose de soumettre en permanence de nouveaux exemples au système afin d'augmenter ses connaissances.

La principale faiblesse du schéma incrémental vient de la difficulté à étendre les compétences initiales d'un réseau de neurones artificiels. En effet, apprendre incrémentalement provoque un phénomène appelé « oubli catastrophique », i.e. le fait que de nouvelles données ou tâches puissent détériorer ce qui a été appris antérieurement.

Au cours de nos travaux, nous proposons de nous intéresser aux solutions d'apprentissage des représentations comme solution aux problématiques de l'apprentissage continu. L'intuition est qu'un modèle avec des représentations adaptées et optimisées pour un large éventail de tâches visuelles, sera apte à apprendre la nouveauté rapidement, c'est-à-dire avec peu d'effort d'adaptation de ses poids. Minimiser l'amplitude des optimisations, c'est contribuer à la stabilité du modèle et donc limiter les risques d'oublier ce qui à été appris.

Un autre objectif est d'étendre l'apprentissage incrémental à des applications de vision autre que la reconnaissance d'objet qui est au coeur de la majorité des travaux existants dans ce domaine. On propose notamment de définir des scénarios continus pour l'apprentissage de systèmes de questions-réponses visuelles et pour la classification multi-labels.

1.4 Contributions

La contribution principale de nos travaux est de présenter des solutions à l'apprentissage incrémental via des méthodes d'apprentissage des représentations. Nous proposons pour cela d'ajouter au schéma incrémental habituel l'hypothèse que des données non-annotées externes sont disponibles pendant l'apprentissage. Le point clé de nos travaux est d'implémenter une méthode efficace pour exploiter ces données additionnelles en parallèle du processus incrémental standard.

Dans une première approche, nous choisissons d'utiliser les données sans étiquette pour apprendre une tâche de reconstruction. Pour cela, nous nous appuyons sur une architecture de type auto-encodeur adverse dans sa version conçue pour apprendre deux tâches simultanées : classification et reconstruction. Grâce à cet algorithme, nous réalisons notre première étude expérimentale sur le scénario incrémental semi-supervisé. Nos résultats valident que les données annexes, même exploitées par une tâche non-supervisée

élémentaire, contribuent à l'amélioration des performances incrémentales.

Dans une deuxième approche, *Pseudo-Labeling* pour l'incrémental de classes, nous proposons d'utiliser la régularisation de consistance par le *pseudo-labeling*. La régularisation de consistance apprend au modèle à générer des sorties invariantes aux transformations préservant la sémantique des images. Cette méthode est spécifiquement conçue pour apprendre des représentations discriminatives sur des images non-annotées. Nous proposons aussi d'adapter la régularisation de consistance pour limiter l'oubli catastrophique. Expérimentalement, la semi-supervision permet de surclasser les méthodes supervisées. Nous montrons aussi que ces données externes apportent la diversité visuelle nécessaire pour traiter des problèmes de classification incrémentaux où l'accès aux annotations est limité.

Finalement, dans une logique de proposer des perspectives d'extension de l'apprentissage incrémental à de nouvelles applications visuelles, nous proposons une première définition des scénarios incrémentaux de questions-réponses visuelles et de classification multi-label.

1.5 Structure du manuscrit

Le manuscrit s'organise de la manière suivante :

— Dans le chapitre 2, nous introduisons en détail les domaines de l'apprentissage incrémental et de l'apprentissage continu. Nous décrivons notamment les tendances de la littérature de ces domaines en accordant une attention particulière aux approches qui servent de base à nos contributions.

— Puis, avec le chapitre 3, nous présentons notre première approche d'apprentissage continu semi-supervisée pour traiter un problème d'incrémental de classes. L'algorithme s'appuie sur une architecture de type auto-encodeur et exploite les données non-annotées via une tâche de reconstruction.

— Dans le chapitre 4, nous poursuivons les travaux initié sur l'incrémental de classes semi-supervisé. Encouragés par les résultats obtenus avec notre première approche, nous développons une nouvelle méthode basée sur la régularisation de consistance sur des données non-annotées. L'approche proposée permet d'améliorer les capacités discriminatives du modèle pour la classification, mais aussi de limiter efficacement l'oubli catastrophique.

— Finalement, avec le chapitre 5, nous proposons d'étendre le schéma incrémental à des applications plus complexes : les systèmes de questions-réponses visuelles et la classification multi-labels. Nous montrons que les adapter à un schéma incrémental n'est pas immédiat et introduit de nouveaux challenges à considérer pour de futur travaux dans le domaine de l'apprentissage continu.

1.6 Publications

- Lechat, A., Herbin, S. & Jurie, F. (2019). Adaptation du problème de questions-réponses visuelles à un contexte d'apprentissage continu. *27ème Colloque du GRETSI, 2019*, (Oral)
- Lechat, A., Herbin, S. & Jurie, F. (2020). Semi-Supervised Class Incremental Learning. *25th International Conference on Pattern Recognition (ICPR), 2020*, 10383-10389, (Poster)
- Lechat, A., Herbin, S. & Jurie, F. (2021). Pseudo-Labeling for Class Incremental Learning. *Proceedings of the British Machine Vision Conference, BMVC, 2021*, (Oral)

Chapitre 2

Domaines et État-de-l'art

Contents

2.1 Introduction

Les travaux menés lors de cette thèse sont ancrés dans le domaine du *deep learning* appliqué à la vision par ordinateur. Dans ce contexte, notre principal axe de recherche concerne l'apprentissage incrémental, un paradigme d'apprentissage atypique qui s'intéresse à la conception de modèles évolutifs. Nos contributions reposent sur des approches en lien direct avec la littérature de l'apprentissage des représentations pour les réseaux de neurones artificiels.

Ce chapitre s'organise donc de la manière suivante :

1. Les sections 2.2 à 2.4 sont dédiées au domaine de l'apprentissage incrémental. En particulier, nous nous concentrons sur les scénarios et les méthodes liés à l'incrémental de classes.
2. La section 2.5 introduit le domaine de l'apprentissage des représentations. On présente une synthèse des tendances de l'état-de-l'art, notamment pour les aspects auto et semi-supervisés.
3. La section 2.7 présente les jeux de données couramment utilisés pour évaluer l'apprentissage incrémental.

2.2 Généralités sur l'Apprentissage Incrémental

2.2.1 Principe et Motivations

L'apprentissage incrémental propose de définir un nouveau paradigme qui, à la différence des approches hors ligne basées « batch », vise à concevoir un système capable d'apprendre en continu tout au long de son cycle de vie.

Entraînement Batch (EB)

Les réseaux de neurones profonds (DNN) obtiennent d'excellentes performances sur de nombreuses tâches de vision par ordinateur modernes. Leur entraînement repose sur des algorithmes d'optimisation en batch tels que la descente de gradient.

Un algorithme dit « batch » ou « hors-ligne », est un algorithme d'optimisation statistique dont la fonction d'application est calculée sur la totalité du jeu de données – *dataset* –, i.e. la totalité du *batch*, afin de mettre à jour les paramètres du modèle. Ce processus est itératif, cette étape d'optimisation, aussi appelée *epoch*, correspond à une passe sur le batch et est répétée jusqu'à convergence vers une solution.

Ces algorithmes sont dit « supervisés » lorsqu'ils sont conçus pour apprendre sur des jeux de données annotés. L'un des critères clés pour le bon fonctionnement de ces algo-

rithmes est que le *dataset* soit indépendemment et identiquement distribué (i.i.d.). La performance finale du réseau dépend de nombreux facteurs mais les études existantes indiquent un clair bénéfice à pouvoir utiliser toujours plus de données avec des DNN toujours plus grands.

Le déroulement d'un entraînement batch (EB) se divise en deux phases distinctes. D'abord la collecte et l'annotation des données, afin de constituer un *dataset* aussi exhaustif que possible pour la tâche visée. Puis l'optimisation des paramètres du DNN sur ce même *dataset*. Ces deux phases permettent en général de garantir un résultat compétitif sur la tâche visée. Celle-ci cependant doit être clairement délimitée et le modèle final est figé.

L'un des avantages des DNN est que leur entraînement est compatible avec une approche « *end-to-end* ». Pour une entrée donnée, on la soumet au modèle afin de calculer la réponse en sortie. Celle-ci sert à calculer une fonction de coût pour ensuite, via rétropropagation du gradient, optimiser les poids automatiquement de la couche de sortie à la couche d'entrée.

Dans la littérature de l'incrémental, le terme entraînement batch est interchangeable avec les dénominations paradigme/entraînement hors-ligne, *offline* ou conjoint.

Paradigme Incrémental

L'apprentissage incrémental, aussi qualifié de continu, *lifelong* ou *online*, a pour objectif de concevoir un système évolutif capable d'apprendre en permanence sur un continuum de données. Contrairement à l'approche batch qui propose un modèle fini et fixe après entraînement, une solution incrémentale doit être capable à tout moment de se mettre à jour afin d'enrichir ses connaissances et le répertoire des tâches dédiées.

Un problème de classification incrémental par exemple correspond à un problème dont les frontières, i.e. le nombre total de classes, n'est pas connu. Le modèle lors de sa création, n'a aucun a priori définitif sur l'ampleur du problème qu'il devra être capable de traiter. Son objectif est alors d'étendre ses capacités de classification afin d'être capable de discriminer aussi bien les classes qu'il aurait vues auparavant que celles nouvellement introduites. Une approche naïve est de simplement prendre le modèle actuel, d'étendre sa sortie afin d'intégrer les nouvelles classes, puis d'effectuer une mise-à-jour des paramètres (*fine-tuning*) sur les nouvelles données.

Dès lors apparaît le principal problème de l'apprentissage incrémental (French, 1999 ; Goodfellow et al., 2013) : l'Oubli Catastrophique (OC), parfois aussi appelé Interférences Catastrophiques (Parisi et al., 2019). Ce phénomène correspond au fait que le modèle à tendance à complètement oublier ses connaissances passées lors d'un apprentissage sur de nouvelles données. La nouvelle connaissance vient totalement interférer et

écraser l'information précédemment stockée. Pour revenir à notre exemple, en optimisant le modèle uniquement sur les nouvelles images, le modèle sera capable de devenir performant sur les nouvelles classes mais deviendra incompétent sur les anciennes classes sur lesquelles il était auparavant expert.

L'oubli catastrophique est un phénomène inhérent aux réseaux connexionnistes et représente l'enjeu principal de la littérature de l'apprentissage incrémental.

2.2.2 Les différents scénarios de l'apprentissage incrémental

L'apprentissage continu peut référer à différents contextes. Il n'est pas envisageable d'arriver à une seule définition du problème qui fasse consensus. Il est cependant possible d'identifier certaines tendances dans la littérature qui restreignent l'étude de l'apprentissage incrémental à quelques scénarios (Parisi et al., 2019)).

La logique pour définir et évaluer les *baselines* incrémentales est la suivante : prendre un problème connu et maîtrisé, puis réarranger les données pour qu'elles soient présentées au modèle de manière séquentielle. À noter que puisque la majorité des travaux concerne la classification d'images, on s'intéresse principalement à cette application pour la suite de cette section.

La définition des différents scénarios incrémentaux est intrinsèquement orientée données. C'est-à-dire que le type d'incrémental dépend principalement du type de données d'apprentissage et de la façon avec laquelle elles sont arrangées au sein du continuum (Prabhu et al., 2020). En classification, les trois scénarios majeurs sont (van de Ven & Tolias, 2019) :

— Incrémental de Classes
— Incrémental de Tâches
— Incrémental de Domaines

Incrémental de Classes (IC) correspond à une tâche de classification unique sans limite définie. Le modèle doit être capable de résoudre un problème de classification dont le nombre total de classes possibles n'est pas connu à l'avance. A n'importe quel instant, une nouvelle image associée avec un label jamais vu auparavant peut être soumise au modèle. Celui-ci doit alors étendre son espace de sortie pour accorder le nouveau label. L'objectif de ce scénario est d'apprendre à discriminer les nouvelles classes tout en maintenant une performance consistante sur les anciennes.

Incrémental de Tâches (IT) considère une approche différente. Ici, le modèle doit apprendre une séquence de tâches définies et dont les limites sont connues par le modèle. Celui-ci possède une tête de sortie propre à chaque tâche. Lorsqu'on soumet une image en

entrée du modèle, on précise aussi la tâche afin que le modèle utilise la tête correspondante. Cette approche est qualifiée de *multi-head*. Par exemple, prenons un modèle capable de résoudre une tâche de classification T_1 à 10 classes, incrémenter une tâche supplémentaire consiste à introduire au modèle T_2 avec 10 nouvelles classes. Un système IT doit être capable de traiter aussi bien T_1 que T_2, mais requiert de connaître la tâche en entrée afin d'éviter toute ambiguïté. Le fait d'indiquer la tâche en entrée permet au modèle de limiter l'espace des sorties possibles.

Incrémental de Domaines (ID) vise à augmenter le domaine d'utilisation du modèle sur un même problème. L'intérêt est de pouvoir d'apprendre successivement sur des jeux de données avec des distributions et des caractéristiques différentes (e.g. données synthétiques et données réelles), pour in fine être capable de tous les traiter.

2.2.3 Les enjeux de l'apprentissage incrémental

Optimisation Comme mentionné dans les motivations, l'intérêt principal de l'apprentissage incrémental vient des applications toujours plus complexes et coûteuses à la fois en données et en puissance de calcul lorsque l'on considère les approches hors ligne classiques.

L'optimisation est donc une problématique majeure. Sont considérés les facteurs tels que le temps d'entraînement, la vitesse de convergence, la mémoire requise ou encore le nombre de données annotées nécessaires.

Transfert de connaissances En parallèle de ces enjeux calculatoires, l'apprentissage incrémental s'intéresse aussi à l'amélioration de la capacité d'apprentissage du système. Cette problématique est justifiée par l'analogie avec la biologie. L'apprentissage des êtres vivants est un phénomène intrinsèquement continu. Un individu humain est polyvalent, capable d'apprendre de nouvelles tâches tout au long de sa vie. Même si des phénomènes d'oubli peuvent survenir, apprendre de nouvelles connaissances ne vient pas supprimer les anciennes. Au contraire, les diverses compétences acquises au cours du temps vont être bénéfiques les unes aux autres (Lopez-Paz & Ranzato, 2017) : les acquis permettent d'apprendre de nouvelles tâches liées plus rapidement (transfert prospectif – *forward transfer*) et de nouvelles connaissances peuvent contribuer à améliorer des compétences déjà connues (transfert rétroactif – *backward transfer*).

Pour que l'apprentissage incrémental des DNN se rapproche de l'apprentissage humain, reproduire ces mécanismes est crucial afin de concevoir des modèles polyvalents et évolutifs.

Dilemme Plasticité-Stabilité Ce dilemme est aussi un concept emprunté au domaine de la biologie (Parisi et al., 2019). Du point de vue d'un DNN, on cherche à maximiser sa capacité d'apprentissage en lui allouant un maximum de degrés de liberté (plasticité) tout en étant capable d'assurer la conservation et l'intégrité des acquis pendant les futures phases d'apprentissage (stabilité). On parle de dilemme car il s'agit d'un problème d'optimisation pour trouver le meilleur compromis entre plasticité et stabilité.

Les travaux en hors ligne, notamment sur les architectures toujours plus massives des DNN, ont grandement contribué à maximiser la plasticité des systèmes, capables aujourd'hui d'apprendre des tâches à grande échelle. La stabilité n'est pas essentielle puisque la phase d'entraînement est unique. C'est lorsqu'on passe dans une configuration incrémentale où la prévalence de l'oubli catastrophique est notable, que la question de la stabilité devient cruciale.

Les méthodes d'apprentissage incrémental s'intéressent donc à développer des solutions pour augmenter la stabilité des modèles, et ainsi remédier à l'oubli.

2.3 Incrémental de Classes (IC)

L'incrémental de classes peut être assimilé à une reformulation du problème de classification standard dans un contexte incrémental. L'objectif est similaire : apprendre un modèle à discriminer un nombre N de classes. Cependant, au lieu d'apprendre sur un ensemble représentatif et uniforme de ces N classes, le modèle est exposé aux classes progressivement et ne connaît pas la complexité finale du problème, i.e. la valeur de N.

En vision par ordinateur, la reconnaissance d'objet est l'un des problèmes les plus étudiés. L'incrémental de classes en est une évolution naturelle et les *benchmarks* tels que MNIST, CIFAR ou ImageNet sont facilement adaptables à ce scénario. Ces facteurs ont contribué à faire de l'incrémental de classes le cas de figure le plus populaire de la littérature du continu.

Nos travaux sur l'apprentissage incrémental se sont principalement concentrés sur le scénario incrémental de classes.

2.3.1 Définition et Notations

On divise la présentation de l'incrémental de classes en deux parties. La première est la définition immédiate du point de vue des données. L'incrémental de classes est fondamentalement un problème orienté données du fait que sa spécificité provient de l'organisation des donnés d'entrée. Nous présentons ensuite plus spécifiquement l'incrémental de classes appliqué à un modèle de type DNN.

Formulation orientée données

Soit une partition d'un ensemble d'images $\mathcal{X} = \{X^1, X^2, ..., X^j, ..., X^N\}$ tel que $\forall j \in [\![1, N]\!], X^j = \{x_1^j, ..., x_{n_j}^j\}$ est un sous-ensemble contenant n_j images associées à la classe j.

Comme mentionné précédemment, l'hypothèse incrémentale admet que le modèle n'aura jamais accès simultanément à l'intégralité de $\mathcal{X}$ mais seulement aux sous-ensembles X^j. L'incrémental de classes est un processus séquentiel, c'est-à-dire une succession d'étapes d'apprentissage que l'on qualifie par la suite de sessions.

Une session correspond à une phase d'apprentissage sur un sous-ensemble de données, appelé batch, contenant une fraction des classes de $\mathcal{X}$. Contrairement au format hors-ligne où « batch » désigne l'intégralité des données pour l'apprentissage, « batch » en incrémental désigne le sous-ensemble des données actuellement accessibles pour l'apprentissage. On note C_i le nombre total de classes vues par le modèle à la i-ème session. Supposons que les classes soient indexées dans leur ordre d'arrivée, de 1 à N. Le batch d'entraînement correspondant à la session i est défini par $D_i = \{X^{C_{i-1}+1}, ..., X^{C_i}\}$. Sans perte de généralité, il est courant de fixer un pas d'incrément de classes constant s tel que $C_i = is$.

Avec cette formulation, on suppose que le modèle est exposé aux classes allant de $C_{i-1} + 1$ à C_i pendant la session i. Il n'a jamais vu ces classes auparavant et une fois la session terminée, D_i est écarté et le modèle ne reverra plus aucun exemple apparenté à ces classes.

Notations du modèle

Les DNN couramment utilisés en classification d'images s'appuient sur un schéma standard : un encodeur ou extracteur de caractéristiques visuelles, suivi par une tête de classification, en général constituée de couches entièrement connectées.

En incrémental, le modèle est amené à évoluer au cours du temps, avec une modification des paramètres pendant l'optimisation voire même une modification de l'architecture. On note Θ_i l'ensemble des paramètres du DNN à la session i.

L'encodeur est généralement un réseau convolutif (CNN) lorsque les données d'entrée sont des images, c'est-à-dire un enchaînement de couches de convolution, *pooling* et de normalisation. Dans ces travaux, on utilise en particulier des CNN avec liaisons résiduelles : ResNet (He et al., 2016) et sa variante Wide-ResNet (Zagoruyko & Komodakis, 2016). On note Φ_i les paramètres de l'encodeur. Son rôle est d'apprendre une application f_i allant de l'espace image (souvent en dimension $C(anaux) \times H(auteur) \times L(argeur)$) vers un espace de représentation de dimension d telle que $f_i : [\![0, 255]\!]^{H \times L \times C} \rightarrow \mathbb{R}^d$.

Le classifieur à une tête g_i est généralement une simple couche linéaire caractérisée par W_i avec $W_i \in \mathbb{R}^{d \times C_i}$ tel que, pour un vecteur encodé $z \in \mathbb{R}^d$, $o = g_i(z) = W_i^T z$ avec $o \in \mathbb{R}^{C_i}$

le vecteur de logits obtenu en sortie du modèle. Ce classifieur est dit a tête unique car il produit une unique distribution englobant l'ensemble des classes possibles. Cette tête est incrémentale : on ajoute des poids dans la matrice W_i (des colonnes supplémentaires) pour augmenter le nombre de sorties, et donc de classes, que le système peut prédire. Le passage aux probabilités se fait via une fonction `softmax`. On note $h(x)$ la distribution des classes prédites. $h(x)$ est un vecteur de probabilités contenant C_i élément. Chaque composante, notée $h_y(x) = p(y|x)$ correspond à la probabilité de l'image d'appartenir à la classe y. Le label prédit $\bar{h}(x)$ correspond à l'`argmax` de ces probabilités (ou directement des logits).

Résumé des Notations

- Encodeur ou extracteur de *features* : f_i de paramètres Φ_i
- Classifieur : g_i de paramètres W_i
- Model complet : $\Theta_i = (\Phi_i, W_i)$
- Vecteur latent ou *features* : $z = f_i(x) = f_i(x; \Phi_i)$
- Logits : $o = g_i(z) = g_i \circ f_i(x) = W_i^T z$
- Distribution prédite : $h(x) = h(x, \Theta_i) = \text{softmax}(o)$
- Probabilité de la classe y : $h_y(x) = p(y|x) = p(y|x; \Theta_i)$
- Prédiction : $\bar{h}(x) = \text{argmax}_y(h(x)) = \text{argmax}_y(p(y|x))$

2.3.2 Métriques et évaluation

Précision incrémentale

Évaluer un modèle en reconnaissance d'objet revient à quantifier sa capacité à discriminer les différentes classes qu'il a apprises. En entraînement batch, la tâche de classification est clairement délimitée et connue. On peut directement évaluer le modèle sur un jeu de validation représentatif des N classes d'applications.

Soit un ensemble de validation $\mathcal{V} = \{(x, y)\}$ avec y l'index de la classe de x. Soit h la fonction associée au modèle tel que $\bar{h}(x) = \text{argmax}_y(p(y|x))$ retourne l'index de classe prédit par le modèle sachant l'image x. Le taux de précision est donné par :

$$\alpha(\mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{(x,y)\in\mathcal{V}} \mathbb{1}_{y=\bar{h}(x)} \tag{2.1}$$

Pour l'incrémental de classes, les limites du problème de classification sont supposées inconnues. On ne dispose donc pas d'une tâche d'évaluation globale fixe. A la place, on cherche à évaluer le modèle à différents moments de son cycle de vie, par exemple à la fin

de chaque session. De plus, pour que l'évaluation soit pertinente, le calcul de la précision doit se faire sur l'ensemble des classes vues par le modèle à cet instant. Soit $\mathcal{V}_i$ l'ensemble de validation concernant les classes apprises à la i-ème session. On note $\mathcal{V}_{1:i} = \mathcal{V}_1 \cup ... \cup \mathcal{V}_i$ et on appelle *précision incrémentale* la quantité :

$$\alpha_i(\mathcal{V}_{1:i}) = \frac{1}{|\mathcal{V}_{1:i}|} \sum_{(x,y)\in\mathcal{V}_{1:i}} \mathbb{1}_{y=\bar{h}(x))} \tag{2.2}$$

Notons que l'indice i lié au α signifie que la précision a été mesurée avec le modèle qui vient de terminer la i-ème session. Il est d'usage en incrémental de classes d'illustrer la performance d'un modèle en rapportant la précision incrémentale obtenue pour chaque session.

Intuitivement, l'objectif est d'obtenir la meilleure précision à tout instant. Cependant, quelques subtilités sont à noter. Entre la session i et $i+1$, le nombre de classes possibles en sortie augmente de s. Le jeu de validation est lui aussi incrémenté. Pour un jeu de validation équilibré, la performance attendue d'un modèle aléatoire est $\alpha_i = \frac{1}{si}$. Du fait de cette augmentation progressive en difficulté, on s'attend à une décroissance des α_i au fil des sessions. L'oubli catastrophique contribue à amplifier ce phénomène.

La précision incrémentale, calculée à chaque session, est la métrique de référence. Il est d'usage de la présenter en fonction du nombre de classes apprises. En pratique, dans les *benchmarks* standards, le *dataset*, et donc le nombre de classes N qu'il contient, est fini. Avec un pas d'incrément fixe s, le nombre de sessions est $M = \frac{N}{s}$. Il est aussi possible de synthétiser la performance d'un modèle sous la forme d'une précision finale α_{last} et d'une précision moyenne α_{avg} :

$$\alpha_{last} = \alpha_M(\mathcal{V}_{1:M}) \qquad \alpha_{avg} = \frac{1}{M-1}\sum_{i=2}^{M} \alpha_i(\mathcal{V}_{1:i}) \tag{2.3}$$

La moyenne exclut la première session qui n'est pas considérée comme incrémentale.

Mesurer l'oubli catastrophique

Limiter l'oubli catastrohpique est au coeur des méthodes incrémentales. Cependant, mesurer l'amplitude de l'oubli ou l'impact d'une méthode sur celui-ci est complexe. Quelques travaux s'essayent à définir des métriques propres à la quantification de l'oubli. Kemker et al., 2018 proposent un ensemble de 3 métriques :

$$\Omega_{base} = \frac{1}{M-1}\sum_{i=2}^{M} \frac{\alpha_{base,i}}{\alpha_{ideal}} \tag{2.4}$$

$$\Omega_{new} = \frac{1}{M-1} \sum_{i=2}^{M} \alpha_{new,i} \tag{2.5}$$

$$\Omega_{all} = \frac{1}{M-1} \sum_{i=2}^{M} \frac{\alpha_i}{\alpha_{ideal}} \tag{2.6}$$

α_{ideal} est la précision obtenue par un modèle entraîné en batch. $\alpha_{base,i}$ est la précision du modèle évaluée uniquement sur $\mathcal{V}_1$, i.e. les s premières classes, à la fin de la session i. $\alpha_{new,i}$ est la précision évaluée sur les nouvelles classes apprises lors de la session i ($\mathcal{V}_i$).

Ω_{base} et Ω_{all} sont des métriques normalisées d'aire sous la courbe. Ω_{base} évalue l'a capacité à retenir l'information apprise lors de la première session. Ω_{all} est simplement la précision incrémentale moyenne normalisée. Ces métriques entre $[0, 1]$ facilitent la comparaison entre les *datasets*, α_{ideal} n'étant presque jamais surclassée. Ω_{new} donne un indice du comportement du modèle face à la nouveauté.

On constate que ces métriques restent très proches de la précision incrémentale. La mesure de l'oubli catastrophique est subtile et souvent, se limite à l'évaluation d'une tâche fixe au cours du temps (Ω_{base}). Beaucoup de travaux associent l'oubli à la dégradation de performance entre une approche continue et celle en batch. α_{ideal} représente en quelque sorte la borne supérieure atteignable uniquement dans une configuration idéale hors ligne, et l'oubli catastrophique est donc quantifié via la différence ou le rapport entre α_{last} et α_{ideal}. Cela suppose que la différence entre hors ligne et incrémental est uniquement due à ce phénomène. Cependant, ce n'est pas la seule dynamique à prendre en compte.

Mesurer les transferts de connaissances

Comme introduit dans la section 2.2.3, la nature séquentielle de l'incrémental de classes laisse présager d'éventuelles interactions entre les connaissances apprises à différents instants.

Lopez-Paz et Ranzato, 2017 introduisent deux métriques pour quantifier le transfert rétroactif (BWT – *Backward Transfer*) et le transfert prospectif (FWT – *Forward Transfer*) dans le cas de l'incrémental de tâches. Ces métriques adaptées à l'incrémental de classes s'expriment de la forme suivante :

$$BWT = \frac{1}{M-1} \sum_{i=1}^{M-1} \alpha_M(\mathcal{V}_i) - \alpha_i(\mathcal{V}_i) \tag{2.7}$$

$$FWT = \frac{1}{M-1} \sum_{i=2}^{M} \alpha_{i-1}(\mathcal{V}_i) - \frac{1}{C_i} \tag{2.8}$$

Le transfert rétroactif correspond à la différence moyenne de performance sur les classes

entre le moment final et le moment où les classes ont été apprises. Un BWT négatif est une autre manifestation de l'oubli et dans le cas positif, cela indique un renforcement des connaissances passées via la nouveauté.

Le transfert prospectif quant à lui cherche à évaluer si un modèle est déjà capable sur les classes qu'il est sur le point d'apprendre grâce à la connaissance précédemment acquise. Pour cela, $\alpha_{i-1}(\mathcal{V}_i)$ est comparé à la précision d'un modèle aléatoire. Cette métrique est rarement utilisée puisque calculer $\alpha_{i-1}(\mathcal{V}_i)$ requiert un système capable d'effectuer une classification *zero-shot*, c'est-à-dire sans données d'apprentissage annotées, sur les nouvelles classes.

Coût et optimisation

Les travaux en apprentissage incrémental visent à produire des méthodes optimisées à la fois en temps de calcul et en besoin matériel. Cette problématique d'optimisation est moins présente que celle de la performance en termes de précision. Elle est cependant au coeur de nombreux travaux et reste un indicateur pertinent pour comparer deux méthodes (Rodríguez et al., 2018).

Le *CVPR Continual Learning Challenge* (Lomonaco et al., 2020) classe les soumissions via une métrique jointe calculée à partir de la précision finale, la précision incrémentale moyenne, le temps total d'entraînement et de test, l'occupation de la RAM et l'occupation du disque.

2.4 Méthodes d'apprentissage incrémental

La notion d'oubli catastrophique dans les réseaux connexionistes est définie dès la fin des années 90 (French, 1999) et réintroduite dans le cas des DNN en 2013 avec une étude proposée par Goodfellow et al., 2013. C'est surtout depuis 2016 que le domaine de l'apprentissage incrémental s'est popularisé. Parmi les récentes méthodes, deux voies ont particulièrement été explorées : la répétition et la distillation de connaissances. Dans cette section, nous définissons d'abord ces 2 axes méthodologiques en détail avant de présenter plus brièvement le reste des solutions.

2.4.1 Apprentissage avec répétition

La répétition est une méthode simple qui s'avère pourtant la plus efficace. C'est une composante essentielle sur laquelle est basée la majorité des travaux de la littérature récente de l'apprentissage continu.

Nous détaillons ici quelques méthodes notoires (Buzzega et al., 2020 ; Castro et al., 2018 ; Hayes et al., 2020 ; Iscen et al., 2020 ; Y. Liu et al., 2020 ; Prabhu et al., 2020 ; Rebuffi et al., 2017) fortement basées sur la répétitions. Cependant, il est à noter que dans l'état-de-l'art actuel de l'apprentissage continu, la répétition est presque systématique. De ce fait, la grande majorité des travaux d'apprentissage incrémental présentés dans ce chapitre, y compris ceux non-référencé dans la section dédiée à la répétition, implémentent une forme de répétition en plus de leur contribution. Quelques travaux s'intéressent à l'apprentissage incrémental sans répétition (Kirkpatrick et al., 2017 ; Ostapenko et al., 2019 ; Yu et al., 2020) mais sont difficilement compétitifs avec les méthodes ayant accès à un échantillon des données passées.

Principe de base

Des exemples vus au cours des précédentes sessions, et donc associés aux anciennes classes, sont inclus – répétés – dans la nouvelle phase d'entraînement. En pratique, cela consiste simplement à implémenter un *buffer*, aussi appelé mémoire épisodique, qui stocke une partie des données introduites à chaque session. Cette approche est paramétrée par un budget mémoire alloué au système, soit en bits soit en nombre d'exemples.

Dans la suite, on note $\mathcal{B}$ la mémoire épisodique avec un budget K désignant la capacité maximale d'exemples conservés. La majorité des méthodes se contentent d'un échantillonnage aléatoire des exemples à conserver tout en maintenant un équilibre entre les classes du buffer.

Pour une session i avec un batch d'entraînement $D_i = \{X^{C_{i-1}+1}, ..., X^{C_i}\}$, la mémoire $\mathcal{B}_{i-1}$ contient $\lfloor \frac{K}{C_{i-1}} \rfloor$ images pour chaque classe allant de 1 à C_{i-1}. Le jeu d'entraînement pour cette session est donc $D_i \cup \mathcal{B}_{i-1}$.

A la fin de la session, le buffer doit être mis-à-jour afin de sauvegarder un échantillon des classes tout juste apprises avant que D_i ne soit mis à l'écart. Pour cela, il est possible de procéder par échantillonnage aléatoire. Dans un premier temps, en raison de l'augmentation du nombre de classes, une partie de la mémoire doit être libérée, c'est-à-dire qu'on supprime une partie des images stockées pour n'en conserver que $\lfloor \frac{K}{C_i} \rfloor$ par ancienne classe. Puis, on prélève au hasard des images de D_i jusqu'à collecter $\lfloor \frac{K}{C_i} \rfloor$ éléments pour chaque classe allant de $C_{i-1} + 1$ à C_i.

Rebuffi et al., 2017 utilisent un algorithme de *herding* pour conserver des exemples représentatifs selon un critère de distance dans l'espace de représentation du modèle. Cependant, la majorité des travaux se contentent d'une approche aléatoire et plusieurs ablations (Javed & Shafait, 2018 ; Y. Wu et al., 2019) montrent que le *herding* ne bénéficie pas à la performance.

Impact de la mémoire

Les méthodes compétitives de l'état-de-l'art incorporent toutes une forme de répétition. Cette approche requiert cependant un coût supplémentaire justifié par ce besoin en mémoire. La valeur de K, le nombre d'exemples en mémoire, est primordiale. La répétition suit une tendance assez intuitive : un budget mémoire plus important donne de meilleur résultats, le cas idéal étant une mémoire infinie qui revient à faire un entraînement batch sur le *dataset* complet $\mathcal{T}_{1:i}$ à chaque session. Le choix de K est donc un compromis entre performance et budget mémoire que l'on s'autorise. Ce budget est souvent contraint par le contexte et l'application. Par ailleurs, certaines applications sensibles peuvent imposer des politiques de confidentialité empêchant la sauvegarde de données au sein du buffer.

L'objectif des méthodes de répétition est de ce fait de maximiser la précision incrémentale tout en minimisant K. Le paradigme incrémental dans sa formulation d'origine voudrait que la totalité des anciennes données soient inaccessibles au modèle une fois utilisées. Une alternative à la répétition, propose d'utiliser un modèle génératif pour créer artificiellement des exemples passés (Lesort et al., 2019 ; Shin et al., 2017 ; Y. Wu et al., 2018). L'avantage est que l'on peut se passer de la mémoire. Cela requiert néanmoins un coût d'entraînement supplémentaire pour apprendre un tel modèle génératif par session. La qualité de la répétition est dépendante de la qualité du générateur.

2.4.2 Distillation de Connaissances (DC)

Distiller de la connaissance est une technique d'apprentissage de représentation (Hinton et al., 2015). Elle est conçue pour compresser et transférer la connaissance d'un modèle « professeur » vers un modèle « étudiant » souvent plus petit. L'intérêt de la distillation de connaissances est double. Compresser le DNN en réduisant le nombre de paramètres requis pour une tâche et optimiser l'étudiant pour qu'il possède de meilleures capacités de généralisation que le professeur.

Il est possible aussi d'effectuer un transfert entre un ensemble de modèles vers un unique modèle. Par exemple, pour un *dataset* d'entraînement de grande taille, il est possible d'entraîner en parallèle des modèles experts sur des sous-ensemble pour accélérer l'entraînement, puis de distiller cet ensemble de spécialistes dans un modèle unique capable de traiter la tâche complète.

Fonction de coût pour la distillation de connaissances

La composante clé pour la distillation de connaissances est la fonction de coût, ou *loss* que l'on notera l_{kd}. Soit $\mathcal{C}$ un ensemble de classes et un modèle enseignant paramétré par $\Theta_{teacher}$. On considère $\Theta_{teacher}$ optimisé sur $\mathcal{C}$. Pour la phase de distillation, on prend un

snapshot du modèle enseignant, i.e. ses poids sont figés. On initialise le modèle étudiant $\Theta_{student}$ de manière aléatoire.

Soit $\mathcal{D}$ un ensemble d'images. On peut calculer les logits en sortie de chacun des modèles :

$$\forall x \in \mathcal{D}, \quad \hat{o}(x) = f(x; \Theta_{teacher}) \qquad o(x) = f(x; \Theta_{student}) \tag{2.9}$$

Distiller la connaissance peut être défini comme du *logits matching* : on veut régulariser $\Theta_{student}$ pour que $o(x)$ soit similaire à $\hat{o}(x)$. On écrit l_{kd} de la façon suivante :

$$l_{kd}(\mathcal{D}, \Theta_{teacher}, \Theta_{student}) = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{y \in \mathcal{C}} \hat{q}_y(x) \log q_y(x) \tag{2.10}$$

avec le passage des logits aux probabilités exprimé par :

$$\hat{q}_y(x) = \frac{\exp\left(\frac{\hat{o}_y(x)}{T}\right)}{\sum_{j \in \mathcal{C}} \exp\left(\frac{\hat{o}_j(x)}{T}\right)} \qquad q_y(x) = \frac{\exp\left(\frac{o_y(x)}{T}\right)}{\sum_{j \in \mathcal{C}} \exp\left(\frac{o_j(x)}{T}\right)} \tag{2.11}$$

C'est un `softmax` avec un paramètre de température T contrôlant le lissage des probabilités. Une valeur forte génère une distribution plus lisse sur l'ensemble des classes. On observe que l_{kd} est simplement l'entropie croisée entre les deux distributions en sortie des modèles.

Pour choisir les images de $\mathcal{D}$, il est d'usage de prendre parmi celles qui ont servi à entraîner l'enseignant. C'est le moyen le plus sûr de garantir que l'on se trouve dans le même domaine et donc que la prédiction proposée par l'enseignant soit pertinente pour l'étudiant. Il est toutefois possible d'utiliser d'autres données, et en particulier des données non-annotées. La distillation de connaissances est un processus que l'on qualifie d'auto-supervisé : le label d'une image est généré automatiquement. Ici, c'est le modèle enseignant qui a pour tâche de créer les vecteurs de vérité terrain utilisés dans l_{kd} pour chaque image de $\mathcal{D}$.

Distillation de connaissances en version incrémentale

Dans un contexte incrémental, on peut intuitivement faire l'analogie entre les réseaux enseignant/étudiant avec les réseaux qui se succèdent au fil des sessions. On garde cette logique d'héritage de la connaissance.

On dispose d'un modèle Θ_{i-1} qui possède l'information sur les C_{i-1} classes passées : c'est l'enseignant. L'étudiant est donc Θ_i. La nuance est que l'étudiant doit aussi apprendre la nouveauté de la session i

L'approche classique propose une optimisation conjointe où la distillation est simplement ajoutée dans la fonction de coût global. C'est la première utilisation de la DC en

incrémental introduite par Z. Li et Hoiem, 2017 :

$$\begin{aligned} l(\mathcal{D}, \Theta_{i-1}, \Theta_i) &= l_{clf}(\mathcal{D}, \Theta_i) + l_{kd}(\mathcal{D}, \Theta_{i-1}, \Theta_i) \\ &= -\frac{1}{|\mathcal{D}|}\left(\sum_{(x,y)\in\mathcal{D}} \log p(y|x;\Theta_i) + \sum_{x\in\mathcal{D}}\sum_{y=1}^{C_{i-1}} \hat{q}_y(x)\log q_y(x)\right) \end{aligned} \tag{2.12}$$

Ici, on a $\mathcal{D} = D_i$ (ou $D_i \cup \mathcal{B}$ si répétition). Le premier terme est une *loss* de classification standard (entropie croisée) pour un apprentissage supervisé. Le second terme est la distillation des anciennes connaissances dans le nouveau modèle. A noter que même si Θ_i possède C_i sorties, la *loss* l_{kd} ne concerne que les anciennes classes. Autrement dit, seules les sorties de 1 à C_{i-1} sont régularisées par la distillation (on ignore les nouvelles classes dans le calcul de $q(x)$).

En incrémental de classes, sur un problème avec beaucoup de classes, les sessions avancées ont un ratio $\frac{C_{i-1}}{C_i}$ qui se rapproche de 1. Cela signifie que le nombre de nouvelles classes est très faible comparé aux classes déjà apprises. Dans une logique de maximiser la performance globale, il est plus pertinent de favoriser la rétention des anciennes classes. Au contraire, dans les premières sessions, on cherche surtout à favoriser l'apprentissage de la nouveauté. Cela se fait via l'équilibrage entre les composantes de la *loss* via un coefficient scalaire : $l = (1-\lambda)l_{clf} + \lambda l_{kd}$ avec $\lambda = \frac{C_{i-1}}{C_i}$.

Rebuffi et al., 2017 proposent une version une version alternative de la distillation adaptée à l'entropie croisée binaire :

$$\begin{aligned} l(\mathcal{D}, \Theta_{i-1}, \Theta_i) = -\sum_{(x,y)\in\mathcal{D}}\Bigg(&\sum_{j=C_{i-1}+1}^{C_i} \delta_{j=y}\log p(j|x;\Theta_i) + \delta_{j\neq y}\log(1-p(j|x;\Theta_i)) \\ &+ \sum_{j=1}^{C_{i-1}} \hat{q}_j(x)\log q_j(x) + (1-\hat{q}_j(x))\log(1-q_j(x))\Bigg) \end{aligned} \tag{2.13}$$

Distillation de connaissances avec données non-annotées

Une deuxième version de la distillation incrémentale s'inspire des ensembles de modèles experts. A chaque nouvelle session, on apprend un modèle spécialisé sur les nouvelles classes uniquement. On distille ensuite la connaissance de cet expert et du précédent modèle Θ_{i-1} dans Θ_i. Il est aussi possible de conserver les modèles experts de chaque session pour constituer l'ensemble enseignant. La difficulté provient du fait que les données qui ont servi à apprendre les experts précédents ne sont plus accessibles. Avec uniquement les données de $\mathcal{D}$, il n'est pas certain que l'on puisse couvrir l'ensemble du domaine de connaissances contenue dans les experts. Pour cette raison, des méthodes comme *Deep*

Model Consolidation (DMC) (J. Zhang et al., 2020) et *Global Distillation* (GD) (Lee et al., 2019) soulignent l'intérêt d'utiliser des données non annotées externes pour la DC.

Ces deux méthodes suivent la même approche. On suppose que l'on dispose du modèle Θ_{i-1} de la session précédente. On initialise un modèle temporaire Θ_{new} qui est entraîné en hors ligne uniquement sur les nouvelles classes de la session i. On initialise ensuite un nouveau modèle Θ_i avec C_i sorties capable donc de traiter toutes les classes possibles. Les connaissances de Θ_{i-1} et Θ_{new} sont distillées dans Θ_i en utilisant uniquement des données non-annotées.

A noter que même si ces méthodes s'appuient sur des données non-annotées, l'apprentissage des classes dépend uniquement du Θ_{new} qui lui est appris en supervisé batch. Les données non-annotées sont donc seulement utilisées pour couvrir le plus grand domaine visuel possible lors de la distillation mais ne sont pas exploitées dans une logique d'améliorer la performance discriminative du modèle.

2.4.3 Incrémental et Biais

Rebuffi et al., 2017 furent les premiers à combiner répétition et distillation. Leur méthode, iCaRL, sert encore de référence pour l'évaluation des nouvelles approches et de nombreux travaux utilisent iCaRL comme *baseline* : Ahn et Moon, 2020 ; Castro et al., 2018 ; Hou et al., 2019 ; Lee et al., 2019 ; Y. Wu et al., 2019 ; Zhao et al., 2020.

Rebuffi et al., 2017 ; Y. Wu et al., 2019 ; Zhao et al., 2020 s'intéressent en particulier à l'une des problématiques inhérente à l'incrémental de classes : le déséquilibre entre nouvelles et anciennes classes. Ce déséquilibre est présent sur deux aspects : le nombre de classes en lui-même et le nombre de données associées à ces classes.

Exemple de biais en IC : CIFAR-100

CIFAR-100 (Krizhevsky, 2009) est un dataset populaire dans sa version incrémentale de classe. Pour l'entraînement, on dispose de 500 images par classe avec un total de 100 classes. Pour un pas incrémental fixe, par exemple $s = 10$, le *dataset* est partitionné en 10 sous-ensembles de 10 classes. Chaque sous-ensemble est un batch d'entraînement pour une session avec $|D_i| = 5000$. Le protocole iCaRL propose d'allouer une mémoire de taille $K = 2000$ sur ce problème.

Le déséquilibre provient de ce budget mémoire souvent très restreint ($K < |D_i|$). Dans le cas où l'on maintient un buffer équilibré, au début de la session i, celui-ci contient $\lfloor \frac{K}{is} \rfloor$ exemple pour chaque ancienne classe. Dans notre exemple, cela commence à 200 images par classes pour la session 2 pour finir à seulement 22 images lors de la session 10. On est loin des 500 images par classe disponibles pour les nouvelles.

De ce fait, le batch d'entraînement de chaque session $\mathcal{D} = D_i \cup \mathcal{B}$ a beau contenir l'ensemble des C_i classes, il ne respecte pas l'hypothèse i.i.d. au sens où les classes ne sont pas uniformément distribuées. On dit que le *dataset* est déséquilibré. Utiliser une *loss* telle que l'entropie croisée avec un tel ensemble à pour effet de biaiser le modèle en faveur des classes majoritaires. En incrémental de classes, cela se traduit par un modèle uniquement capable de répondre un label associé au dernières classes, quelque soit l'image en entrée. La figure 2.1 donne un exemple de matrice de confusion pour un modèle ayant suivi ce type d'apprentissage avec répétition. On observe directement les erreurs du réseaux qui prédit des nouvelles classes pour les images associées aux anciennes.

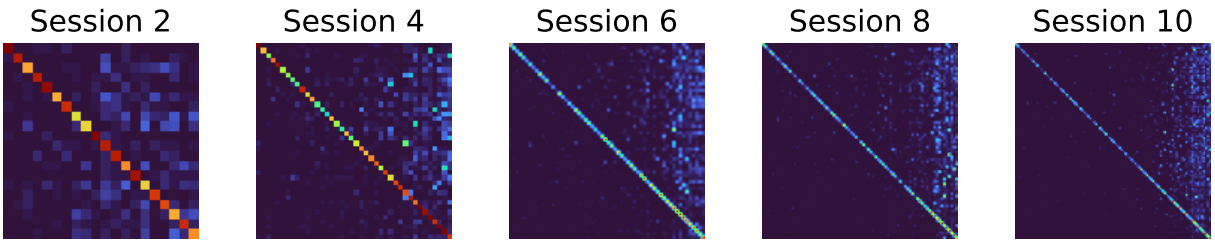

FIGURE 2.1 – Matrice de confusion obtenue sur CIFAR-100 pour un modèle avec répétition + distillation sur CIFAR-100 avec les classes apprises par 10. En abscisse, la prédiction du modèle et en ordonnée, le label.

La conséquence d'une optimisation biaisée

Lorsqu'on optimise le modèle avec un algorithme de descente de gradient stochastique, chaque image de $\mathcal{D}$ contribue au calcul du gradient de manière égale.

Une image x associée à la classe y est un exemple positif pour cette classe. Le problème ici est à label unique. Cela implique que x est par conséquent un exemple négatif pour toutes les autres classes (en raison du `softmax`). L'entropie croisée favorise une réponse piquée sur les exemples positifs et écrase la sortie pour tous les exemples négatifs. Les anciennes classes reçoivent donc peu de signaux positifs pour un grand nombre de négatifs.

Y. WU et al., 2019 ; ZHAO et al., 2020 montrent que l'effet de la descente de gradient est notamment observable au niveau du classifieur linéaire, caractérisé par W_i. Soit $w_i^y \in \mathbb{R}^d$ la colonne de W_i d'indice y. Les w_i^y sont dit spécifiques à chaque classe y. A la fin d'une session d'apprentissage, la norme des vecteurs poids associés aux nouvelles classes est bien supérieure à la norme de ceux associés aux anciennes (fig. 2.2).

Correction du biais

Le biais en faveur des nouvelles classes est inhérent aux approches par répétition et se manifeste notamment lorsqu'on utilise un classifieur linéaire.

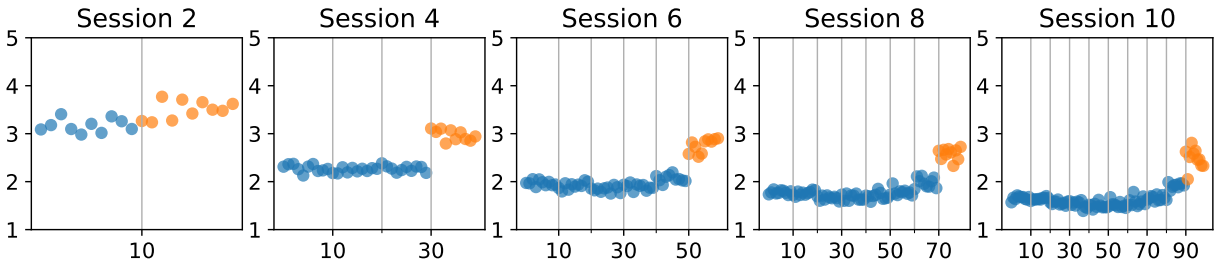

FIGURE 2.2 – Normes des poids w_i^y spécifique à chaque classe. Résultats obtenus avec une approche simple de répétition + distillation sur CIFAR-100 avec les classes apprises par 10. Les normes associées aux anciennes classes sont en bleu. Les points oranges sont les classes qui viennent d'être apprises.

A noter que l'ajout d'une régularisation sous forme de distillation de connaissance n'aide en aucun cas à corriger le biais. En effet, l_kd permet de maintenir la capacité du modèle à discriminer les anciennes classes entre elles mais ne contribuent pas à positionner les nouvelles classes par rapport aux précédentes.

Les nombreuses méthodes combinant répétition et distillation se distinguent par leur approche pour traiter le biais. iCaRL prend l'approche la plus drastique et supprime le classifieur linéaire pour l'inférence. A la place, il utilise les exemples du buffer pour calculer un prototype par classe dans l'espace des représentations et la classification se fait via plus proche centroïde. Y. WU et al., 2019 ; ZHAO et al., 2020 appliquent un post-traitement, soit sur les sorties, soit directement sur les poids du classifieur pour corriger le biais. AHN et MOON, 2020 proposent de séparer la *loss* en utilisant 2 `softmax` indépendant, un sur les anciennes et un sur les nouvelles sorties, afin de limiter l'effet d'écrasement. DOUILLARD et al., 2020 ; HOU et al., 2019 ; TAO et al., 2020 ajoutent une contrainte topologique sur les représentations en supplément de la distillation afin d'avoir une régularisation protégeant les anciennes classes des nouvelles.

PRABHU et al., 2020 proposent une approche naïve baptisée GDumb. Cette méthode consiste à mettre à jour la mémoire en amont de la session, afin de créer $\mathcal{B}_i$ un ensemble équilibré des C_i classes. Le modèle Θ_i est initialisé aléatoirement et entraîné en batch uniquement sur $\mathcal{B}_i$. De ce fait, la majorité des nouvelles données est directement défaussée sans jamais être passée au modèle. On conserve $\lfloor \frac{K}{C_i} \rfloor$ image par classes. Avec GDumb, les auteurs veulent définir une borne inférieure de la performance par répétition. GDumb surpasse d'autres méthodes plus complexes seulement grâce au fait qu'elle est non-biaisée, et ce malgré la restriction en nombre d'images disponibles.

2.4.4 Autres axes de recherche

Dans cette section, nous présentons une liste non-exhaustive des autres méthodes notables de la littérature.

Régularisation interne des poids

Parmi les méthodes orientées modèle, on retrouve les approches intervenant directement sur les paramètres du modèle. On peut notamment mentionner quatre travaux pionniers pour ces approches : *Elastic Weight Consolidation* (Kirkpatrick et al., 2017), *Synaptic Intelligence* (Zenke et al., 2017), *Riemannian walk* (Chaudhry et al., 2018) et *Memory Aware Synapse* (Aljundi et al., 2018).

L'idée globale est que chaque paramètre du DNN, les synapses par analogie avec la biologie, possède un potentiel limité de connaissances. L'objectif est de contrôler la répartition des connaissances au sein du modèle.

Lors de l'apprentissage, certaines synapses vont se spécialiser sur des tâches précises. Ces méthodes implémentent une régularisation en attribuant des poids à chaque paramètres afin de contrôler l'amplitude de leur mise-à-jour. Cette pondération est à un double rôle : limiter les mises-à-jour des synapses déjà spécialisées et orienter les nouvelles connaissances vers les paramètres encore peu utilisés.

En pratique, cela se fait en mesurant et en conservant l'historique des gradients associés à chaque paramètre. En effet, un paramètre dont les gradients passés ont une forte amplitude indique des mises-à-jour importantes. On considère alors qu'il est saturé en connaissance et doit être protégé à l'avenir. Inversement, un paramètre avec des gradients faibles est un paramètres peu optimisé. Son budget d'apprentissage est disponible de nouvelles connaissances.

Architectures Dynamiques

Autre famille de méthodes orientées modèle, elles reprennent l'idée fondamentale des méthodes de régularisation interne : les paramètres spécialisées doivent être protégés pour prévenir l'oubli (Abati et al., 2020 ; Hung et al., 2019 ; X. Li et al., 2019 ; Mallya & Lazebnik, 2018 ; Rajasegaran et al., 2019 ; Rusu et al., 2016 ; Schwarz et al., 2018 ; Serrà et al., 2018 ; Yan et al., 2021 ; Yoon et al., 2018).

Au lieu d'imposer une régularisation sur les synapses existantes, ces approches proposent d'allouer dynamiquement des nouveaux paramètres dédiés à l'apprentissage des nouvelles connaissances.

Les *Progressive Networks* (Rusu et al., 2016) proposent par exemple d'initialiser une copie du modèle pour chaque tâche. Des connexions entre les différents modèles permettent

d'exploiter les corrélations entre tâches. Le principal défaut est la complexité linéairement croissante avec le nombre de tâches. Une étape de compression est de ce fait requise pour limiter ce phénomène (Schwarz et al., 2018 ; Yan et al., 2021).

A l'exception de Hung et al., 2019 ; Rajasegaran et al., 2019 ; Yan et al., 2021, ces travaux sont spécifiquement adaptés pour l'incrémental de tâche et ont besoin qu'on leur précise la tâche en cours afin qu'ils utilisent les paramètres adaptées à celle-ci.

2.5 Apprentissage des Représentations (AR)

La force de l'apprentissage batch est la capacité à traiter simultanément un grand nombre de données. Dans le cas d'une application de vision, on constate que la performance est souvent liée à la quantité d'images accessibles. Apprendre sur une plus grande diversité d'images permet au modèle d'atteindre un meilleur niveau de compréhension visuelle. On parle souvent de représentations apprises par le modèle. Ces représentations sont vues comme une projection ou une interprétation du monde des images dans un nouvel espace propre au modèle. C'est dans cet espace, appelé espace de représentation, espace des *features* ou encore espace latent, que le modèle raisonne pour résoudre sa tâche.

2.5.1 Motivations

Apprendre des représentations est inné pour un DNN et dicte sa performance. Du point de vue de la classification, il est essentiel d'apprendre un encodeur f de qualité afin que le classifieur puisse facilement discriminer les *features*. La situation idéale étant que toutes les classes soient linéairement séparables dans l'espace latent.

L'apprentissage des représentations a donc pour objectif de créer des *features* représentatives et discriminantes. L'idée est que le vecteur latent $z = f(x)$, encode un maximum d'informations à la fois inter-classes et intra-classes.

En pratique, les DNN sont généralement entraînés selon un paradigme dit *end-to-end*. C'est-à-dire qu'on optimise l'intégralité du système simultanément. Cela se fait via rétropropagation qui part du gradient obtenu en sortie du classifieur et qui est remonté jusqu'à l'entrée de l'encodeur. Une telle optimisation vise à optimiser l'encodeur spécifiquement pour le domaine décrit par le jeu d'entraînement. Dans un cas où l'on dispose de suffisamment de données, cette approche suffit à produire un modèle compétitif et capable de généraliser.

Cependant, toutes les applications ne disposent pas toujours de *datasets* extensifs. Dans ces situations, le potentiel d'un DNN est limité et souvent impacté par des effets tels que le sur-apprentissage (*overfitting*) ou la non-convergence. Ces phénomènes sont amplifiés auprès des architectures récentes avec toujours plus de paramètres et toujours

plus avides en données. La première solution, souvent coûteuse, est de collecter plus de données quand cela est possible.

L'apprentissage des représentations (Bengio et al., 2013) propose à la place de rajouter une étape d'entraînement dédiée à l'optimisation de l'encodeur f. Elle repose sur un principe majeur : deux tâches en vision, même différentes, peuvent utiliser les mêmes ressources, et donc les mêmes représentations. Par exemple, apprendre à reconnaître des objets et segmenter des d'images sont deux applications avec leurs propres nuances mais on peut supposer qu'elles requièrent un ensemble d'informations utiles commun. Selon cette logique, il est pertinent d'entraîner l'encodeur sur une tâche (A), pour utiliser les *features* sur une tâche (B). (A) étant en principe une tâche proche de (B) mais avec des données plus facilement accessibles.

(A) est usuellement appelé tâche source ou tâche prétexte alors que (B) est la tâche cible ou principale.

2.5.2 Pré-entraînement supervisé

Lorsque les étapes d'entraînement sont séquentielles, on parle de pré-entraînement. La condition pour construire une tâche prétexte est donc principalement d'avoir un *dataset* adapté. La classification, du fait de sa popularité, est la tâche qui rassemble les plus grands *datasets* annotés (ImageNet, Open Images) et est naturellement devenu le type de pré-entraînement le plus utilisé.

Ce pré-entraînement est supervisé. On entraîne simplement un modèle (Φ, W_{source}) sur la tâche de classification, par exemple les 1000 classes du challenge ImageNet (ILSRVC12). Les poids de Φ contiennent les représentations apprises et W est une tête de sortie optimisée à résoudre la tâche source. On peut donc supprimer la tête W pour connecter une nouvelle tête W_{cible} spécifique à la tâche cible. L'entraînement reprend alors sur les données de la tâche principale. Si les représentations apprises sont suffisantes, on peut éventuellement geler les poids – empêcher toute mise à jour – de Φ et seulement optimiser W_{cible}. Dans la majorité des cas, Φ doit subir de légers réglages supplémentaires (*fine-tuning*) et on laisse ses paramètres évoluer pendant l'apprentissage.

On dit alors que le pré-entraînement sert d'initialisation : il donne un meilleur point de départ pour l'algorithme de descente de gradient stochastique. Meilleur au sens que le modèle à plus de chance de converger vers un meilleur minimum local, mais aussi plus rapidement (en moins de pas de gradient) et donc avec moins de données.

En pratique, il est devenu presque systématique d'utiliser une initialisation via pré-entraînement supervisé. Notamment en raison de l'accessibilité des poids déjà pré-entraînés pour la majorité des architectures communes. Il est donc possible d'omettre l'apprentissage sur (A) souvent coûteux et directement utiliser les poids fournis comme initialisation.

2.5.3 Pré-entraînement non-supervisé

Les récentes avancées en apprentissage des représentations se concentrent sur un point : s'affranchir du besoin en annotations pour le *dataset* de pré-entraînement (Erhan et al., 2010). L'idée est que les données brutes collectées (images sur les réseaux sociaux, vidéos collectées par une flotte de voitures, images satellites...) sont désormais facilement accessibles et toujours croissantes. Produire un flux d'annotations équivalent nécessiterait des moyens irréalistes. Le but de la non-supervision est de proposer une solution pour apprendre des représentations sans labels afin de proposer le meilleur point d'initialisation possible pour la tâche principale et ainsi limiter le travail d'annotation pour celle-ci. La façon la plus optimisée de faire du non-supervisé est l'auto-supervision.

L'auto-supervision est le domaine à l'origine du terme tâche prétexte. Son principe repose sur le fait qu'on dispose de tous les outils et techniques pour entraîner des DNN de manière supervisée. Il serait donc profitable de pouvoir tout réutiliser y compris lorsque les données n'ont aucun label associé. Pour cela, la tâche prétexte se définit comme une tâche dont il est possible de générer automatiquement – de manière artificielle – les labels associés à chaque image.

2.5.4 Les tâches prétextes

La littérature de l'auto-supervision propose un large panel de tâches prétextes. Une présentation exhaustive de ces méthodes est proposée par Jing et Tian, 2019. Quelques exemples notables sont :

— La prédiction de rotations ou *RotNet* (Gidaris et al., 2018) : une image x subit une rotation d'angle $(0, 90, 180, 270)$ degrés. Le modèle doit prédire quelle rotation a subit l'image. C'est une tâche de classification à 4 classes.

— Résolution de puzzles (Noroozi & Favaro, 2016) : l'image est séparée en patches disjoints (e.g. une grille 3×3) dont les positions sont permutées. Le modèle doit reconstruire l'image d'origine.

Ces tâches reposent sur des compétences associées au « contexte spatial ». Le modèle doit acquérir une compréhension spatiale de l'image et de son contenu pour les résoudre. D'autres tâches simples à implémenter sont les tâches génératives (X. Chen et al., 2016 ; Donahue & Simonyan, 2019 ; Goodfellow et al., 2014 ; Larsson et al., 2016 ; Makhzani et al., 2015 ; Pathak et al., 2016 ; R. Zhang et al., 2017) telles que la colorisation, l'*inpainting* ou la génération d'images (e.g. GAN). Ces tâches utilisent l'image elle-même en tant que vérité terrain. A noter que des tâches plus complexes sont accessibles notamment lorsqu'il existe déjà des algorithmes pour générer des VT : prédiction de flot optique, estimation de profondeurs...

RotNet a longtemps fait office d'état-de-l'art du fait de sa performance sur tout type d'architectures CNN (Kolesnikov et al., 2019) mais aussi de par sa simplicité.

Cependant, deux nouvelles catégories de méthodes discriminatives ont récemment pris le pas dans la littérature de l'auto-supervision : le *contrastive learning* et la régularisation de consistance (*consistency regularization*).

Reconstruction ou génération d'image

La reconstruction d'image est une tâche basique ne nécessitant aucune annotation. D'un point de vue AR, l'idée est généralement de compresser l'information de l'espace image dans un espace latent de plus faible dimension. Effectuer l'opération inverse de reconstruire l'image à partir de sa représentation contraint celle-ci à encoder un maximum d'information visuelle. Les architectures de type auto-encodeurs sont particulièrement connues pour ça (Vincent et al., 2008).

Plus récemment, tout une branche de la littérature s'est développée autour de l'idée de créer des espaces de représentations continus et cohérents. Pour cela, deux grandes familles de méthodes se sont démarquées : les réseaux adverses génératifs (GAN) (Goodfellow et al., 2014) et les auto-encodeurs variationnels (VAE) (Kingma & Welling, 2014).

Vouloir contrôler l'espace de représentation signifie généralement imposer un prior, appris ou imposé arbitrairement, sur les représentations. Par exemple, le VAE impose un prior gaussien sur ses représentations. C'est-à-dire qu'il contraint son encodeur à générer des vecteurs de représentations suivant une distribution gaussienne, via ce qu'on appelle l'astuce de la reparamétrisation. Des variantes du VAE ont été développées pour fonctionner avec différents priors (Makhzani et al., 2015 ; van den Oord et al., 2017).

Les GAN sont principalement conçus pour traiter la tâche de génération. Un modèle dit génératif est une fonction de mapping de l'espace de représentation vers l'espace image. Il est analogue au module décodeur d'un AE. Son entraînement est adverse car il repose sur l'opposition entre un discriminateur qui doit distinguer une image générée d'une image réelle et un générateur qui doit apprendre à créer des images pouvant tromper le discriminateur. Se référer à Tschannen et al., 2018 pour une liste plus exhaustive des méthodes basées AE et Creswell et al., 2018 pour les GAN.

Il existe une multitude de méthodes différentes mais elles partagent le même point essentiel : le choix du prior. Tschannen et al., 2018 distinguent 4 catégories de priors possibles qui vont influer sur l'espace de représentation résultant :

— *Disentanglement* : on encode les différents facteurs de variation indépendants au sein de variables indépendantes. Par exemple, un vecteur de représentations *disentangled* encodant une image d'objet aura une dimension dédiée pour chaque caractéristiques de l'objet telles que son orientation, sa taille, l'exposition...

— Hiérarchique : une image peut se décrire par des concepts ou attributs à différents niveaux d'abstraction ou différentes échelles spatiales.
— Semi-supervision : l'espace de représentation pour la reconstruction/génération est partagé avec une tâche supervisée (modèle bi-tâches). Une tâche supervisée de classification va par exemple contraindre les représentations selon la sémantique des classes. La tâche non-supervisée est là pour assister à apprendre des représentations plus complètes et plus généralisables. Surtout quand la tâche supervisée manque de données annotées.
— *Clustering* : Un *dataset* est généralement organisable en catégories selon différents facteurs (e.g. classes d'objets). Le *clustering* propose de modéliser ces catégories directement dans l'espace de représentation en imposant par exemple, un prior de type mélange de gaussiennes où chaque composante représente une catégorie.

Contrastive Learning

Le *Contrastive Learning* dérive de la classification niveau instance, une tâche prétexte de classification où chaque image est associée à sa propre classe (Dosovitskiy et al., 2014 ; Z. Wu et al., 2018).

Le *Contrastive Learning* (T. Chen, Kornblith, Norouzi et al., 2020 ; T. Chen, Kornblith, Swersky et al., 2020 ; He et al., 2020 ; Hénaff, 2020 ; Hjelm et al., 2019 ; van den Oord et al., 2018) garde cette idée que chaque image appartient à sa propre classe. De plus, via augmentation de données, il est possible de créer plusieurs versions modifiées de cette image. La *loss* contrastive a pour objectif de rapprocher ce qui se ressemble et éloigner ce qui est différent.

Prenons pour exemple l'algorithme SimCLR (T. Chen, Kornblith, Norouzi et al., 2020). Soit $\mathcal{D}$ un ensemble de N images et deux fonctions d'augmentation t et t'. Pour chaque image $x \in \mathcal{D}$, on applique les fonctions de transformations et on les soumet à l'encodeur f tel que on obtiennent $z_i = f(t(x))$ et $z_j = f(t'(x))$. On dit que (z_i, z_j) constitue une paire positive. On génère une paire positive pour chaque image de $\mathcal{D}$ pour un total de $2N$ vecteurs z_k. $\forall k \notin \{i, j\}, (z_i, z_k)$ sont les paires négatives de z_i. Le terme de la *loss* associé à la paire positive (i, j) est noté :

$$l_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/T)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(\text{sim}(z_i, z_k)/T)} \tag{2.14}$$

avec `sim` la similarité cosinus (produit scalaire des vecteurs normalisés). Ce terme, appelé NT-Xent (pour *Normalized temperature-scaled cross-entropy loss*), doit être minimisé. Le rôle du numérateur est de maximiser la similarité de la paire (z_i, z_j) alors que le dénominateur contraint les paires négatives à être dissemblables.

Le *contrastive learning* n'a pas besoin de tête spécifique pour une tâche prétexte car sa *loss* est calculable directement dans l'espace des *features*. SimCLR montre qu'ajouter une tête de projection supplémentaire g tels que $\bar{z}_i = g(z_i)$ pour travailler dans cet espace latent temporaire permet un apprentissage plus stable et une meilleure généralisation une fois que l'on supprime g. Mais l'idée reste qu'on veut modeler l'espace de représentation de manière cohérente en fonction des caractéristiques visuelles présentes dans les images. Et notamment, si la tâche principale est la classification, on vise à apprendre au préalable un espace où les différentes classes sont naturellement regroupées.

Pour le choix des transformations, SimCLR combine rognage d'image et distorsion de couleurs. De plus, l'efficacité de la *loss* dépend directement de la diversité des paires. Pour chaque pas de gradient, la *loss* est calculée uniquement sur un mini-batch de taille N. Dans l'ensemble, les différentes méthodes requièrent un N immense ce qui rend les méthodes de *contrastive learning* très coûteuses en calcul.

Pseudo-Labeling

Comme mentionné précédemment, le concept de l'auto-supervision est d'exploiter des données non-annotées sur une tâche supervisée connue et maîtrisée. Le *pseudo-labeling* ou *self-labeling* s'appuie donc sur la tâche sur laquelle la littérature du *deep learning* dispose de la meilleure expertise : la classification supervisée. Alors que celle-ci requiert une annotation humaine afin d'attribuer des labels ayant un sens sémantique, l'idée ici est d'automatiser le processus d'annotation. Ces labels artificiels, générés artificiellement, servent de « pseudo-labels » dans un processus supervisé.

La difficulté du *pseudo-labeling* est de concevoir des moyens pour créer ces annotations. Plus la distribution des pseudo-labels sera proche de la réalité, plus le transfert vers la tâche principale sera efficace.

Caron et al., 2018 proposent d'utiliser un *k-means* sur l'ensemble des données projetées dans l'espace de représentation par l'encodeur f. En assimilant chaque cluster à une classe, on génère ainsi une table d'attribution Q qui associe chaque image à l'un des K labels possible. Cet algorithme, *DeepCluster*, alterne entre *clustering* via *k-means* et optimisation des paramètres du DNN.

Asano et al., 2020 itèrent sur cette logique d'alternance entre (i) la création de Q sachant Φ et l'optimisation de Φ sachant Q. La différence avec DeepCluster est que le mécanisme de *pseudo-labeling* s'inspire de l'algorithme Sinkhorn-Knopp (Cuturi, 2013) de la littérature du transport optimal.

Gidaris et al., 2020 proposent un modèle basé sur un *bag of visual words* (BoWNet) au lieu de simples labels. BoWNet encode chaque image en une famille de vecteurs de *features*. En appliquant directement un *k-means* sur ces vecteurs pour l'ensemble du *dataset*, il est

possible de créer un dictionnaire de K *mots visuels*. Pour une image donnée, ses vecteurs de *features* sont quantifiés via un algorithme de plus proche voisin avec le dictionnaire. L'histogramme de ces vecteurs quantifié produit un sac de mots visuels qui sert de vérité terrain.

2.5.5 Semi-supervision avec régularisation de consistance

La semi-supervision suppose que le modèle apprend à la fois sur des données annotées et non-annotées. Autrement dit, la phase d'apprentissage optimise simultanément la tâche prétexte et la tâche principale. Cette approche est couramment utilisée lorsqu'on a accès à un grand nombre de données d'un même domaine mais que seulement une fraction est annotée (Oliver et al., 2018).

On soulignera notamment les approches semi-supervisées conçues autour de la régularisation de consistance – *consistency regularisation* : Berthelot et al., 2020 ; Berthelot et al., 2019 ; Laine et Aila, 2017 ; Sohn et al., 2020 ; Tarvainen et Valpola, 2017 ; Q. Xie et al., 2020.

Ces méthodes sont les plus compétitives pour la classification semi-supervisée. Elles reposent sur le fait que, puisque l'on dispose de la tâche principale de classification dès le début de l'apprentissage, l'idéal serait de l'utiliser aussi en tâche prétexte. Pour chaque image sans label, il faut générer un vecteur cible de vérité terrain (VT) dans l'espace de sortie du classifieur principal, i.e. un vecteur de probabilités couvrant les C classes possibles. En version *pseudo-labeling*, on remplace ce vecteur de probabilité par un label obtenu en prenant le maximum des classes prédites.

La régularisation de consistance propose d'utiliser le modèle $\Theta = (\Phi, W)$ lui-même afin de produire ces VT. A l'instar du *contrastive learning*, il est nécessaire de définir deux fonctions de transformations t et t'. Pour une image x, on calcule les probabilités $p_y(x) = p(y|t(x);\Theta)$ et $p'_y(x) = p(y|t'(x);\Theta)$.

L'une de ses distributions correspond au label artificiel et l'autre à la prédiction. Arbitrairement, prenons $p(x)$ la prédiction et $p'(x)$ le label de x. Via la fonction coût, on contraint le modèle à rapprocher sa prédiction $p(x)$ de $p'(x)$. Autrement dit, on contraint le modèle à produire une réponse consistante entre deux versions augmentées de la même image, d'où ce terme de régularisation de consistance. Afin que les prédictions soient cohérentes, les transformations t et t' doivent préserver la sémantique de l'image.

En pratique, les probabilités utilisées en tant que label requièrent de passer par un processus de *sharpening* contrôlé par une température T :

$$q'(x) = \text{sharpen}(p'(x), T) \quad \text{avec} \quad \forall y, \quad q'_y(x) \frac{p'_y(x)^{\frac{1}{T}}}{\sum_{j=1}^{C} p'_j(x)^{\frac{1}{T}}} \tag{2.15}$$

Notez que lorsque les probabilités sont issues d'un `softmax` appliqué aux logits $o(x)$, on a l'égalité $\text{softmax}(o(x)/T) = \text{sharpen}(\text{softmax}(o(x)), T)$.

Plusieurs fonctions de régularisation sont possibles. Sont communément utilisées l'entropie croisée (Q. Xie et al., 2020) ou l'erreur quadratique moyenne (Berthelot et al., 2019). Soit un ensemble de données non-annotées $\mathcal{U}$, on a :

$$l(\mathcal{U}, \Theta) = -\frac{1}{|\mathcal{U}|C} \sum_{x \in \mathcal{U}} \sum_{y=1}^{C} q'_y(x) \log p_y(x) \tag{2.16}$$

$$l(\mathcal{U}, \Theta) = \frac{1}{|\mathcal{U}|C} \sum_{x \in \mathcal{U}} ||q'(x) - p(x)||_2^2 \tag{2.17}$$

Le processus de *sharpening* est identique au lissage via température utilisé pour la distillation de connaissance (eq. 2.11). Le lissage intervient pour $T > 1$ et le *sharpening* pour $T < 1$. L'intérêt du *sharpening* est de produire des VT plus piquées et ainsi se rapprocher le plus possible du cas supervisé. Autrement dit, on veut réduire l'entropie de la distribution des labels. En particulier, pour $T \to 0$, la sortie de $\text{sharpen}(p'(x), T)$ se rapproche d'un Dirac. Pour référence, les méthodes citées utilisent $T = 0.5$.

Avec l'algorithme *FixMatch* illustré en figure 2.3, Sohn et al., 2020 proposent d'utiliser des pseudo-labels. Au lieu de conserver $p'(x)$ comme label artificiel, *FixMatch* calcule l'indice de son maximum $\hat{y} = \text{argmax}_y(p'(x))$ afin d'obtenir une unique classe. *FixMatch* peut donc être vu comme le cas $T \to 0$ de la fonction de *sharpening*. La *loss* est définie de la façon suivante :

$$l(x, \Theta) = -\frac{1}{|\mathcal{U}|} \sum_{x \in \mathcal{U}} \mathbb{1}_{\max(p'(x)) > \tau} \log p(\hat{y}|t(x); \Theta) \tag{2.18}$$

Cette méthode ne dépend donc plus du paramètre de température. Néanmoins, elle introduit un nouveau scalaire τ qui définit un seuil de sélection. La fonction indicatrice de la *loss* vérifie si la valeur de $\max(p'(x))$ franchit ce seuil. Via ce critère, on s'assure que le modèle est confiant sur le PL $\hat{y}$ qu'il a attribué à l'image $t'(x)$. Si un jamais un PL a sa probabilité associée en deçà du seuil, il est considéré invalide et simplement ignoré dans la *loss* (indicatrice nulle). Ce mécanisme vise à minimiser l'entropie. Il est spécifique au sens où il filtre les échantillons de $\mathcal{U}$ sur lesquels Θ prédits des probabilités trop « lisses » au lieu de les forcer à être piquées via un *sharpening*.

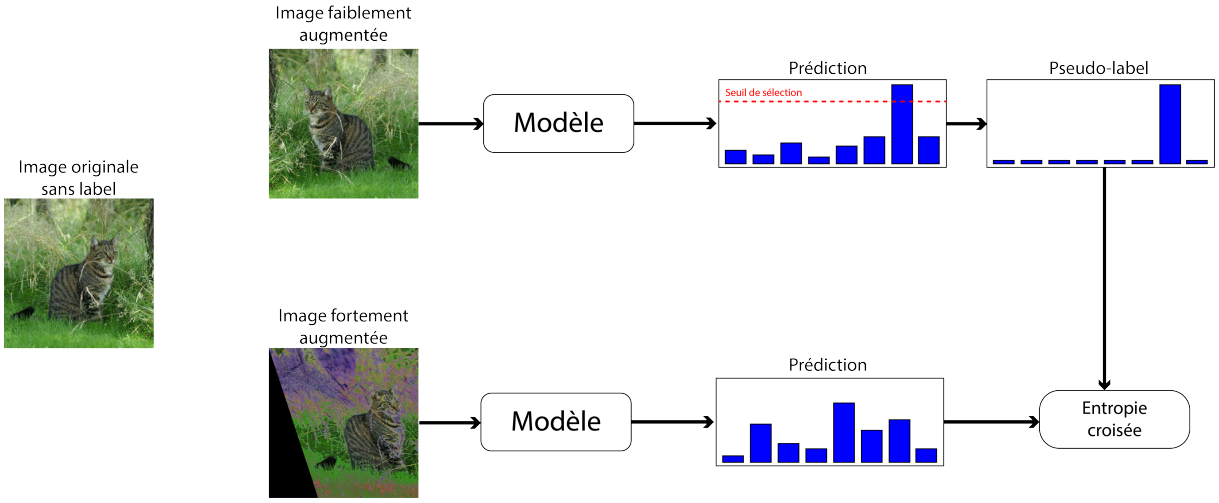

FIGURE 2.3 – Mécanisme de pseudo-labeling utilisé dans FixMatch (SOHN et al., 2020).

2.6 Conclusion : Vers un incrémental de classes semi-supervisé

Notre étude se concentre sur l'apprentissage continu et notamment, le scénario de l'incrémental de classes. On prend pour point de départ, les méthodes implémentant de l'apprentissage par répétition et la distillation de connaissances. Ces deux approches sont largement exploitées par l'état de l'art de l'apprentissage incrémental,

On cherche en particulier à résoudre le problème de l'apprentissage continu avec une approche focalisée sur l'apprentissage des représentations. Un contexte incrémental impose néanmoins une limitation des données annotées accessibles pour l'entraînement.

Implémenter un apprentissage des représentations requiert généralement un grand nombre de données. Pour cela, il est de ce fait nécessaire de compter sur une source externe. Nous proposons donc d'incorporer ces données supplémentaires lors du processus incrémental.

Nos contributions vont de ce fait s'orienter sur l'adaptation d'un processus incrémental avec l'hypothèse que des données externes sont aussi disponibles. Inspirés par les tendances récentes du domaine de l'apprentissage des représentations, on suppose qu'on dispose de données non-annotées, facilement accessibles en quantité importante et à moindre coût. On se place donc dans une configuration favorable à une méthode d'apprentissage incrémental semi-supervisée. C'est-à-dire, un algorithme qui est capable d'exploiter simultanément les données du continuum incrémental et celles non-annotée provenant d'une source secondaire. L'idée principale est d'améliorer la qualité des représentations apprises par les méthodes incrémentales mais aussi d'observer si une régularisation semi-supervisée

permet de pallier à l'oubli catastrophique.

2.7 *Datasets*

Cette section liste les différents jeux de données, ou *datasets*, qui seront utilisés dans la suite. Ces *datasets* sont couramment utilisés dans les *benchmarks* de reconnaissance d'objet. Ils sont généralement constitués d'images contenant une unique entité. Si le *dataset* est annoté, les données sont sous la forme de tuples (image, label).

2.7.1 MNIST et E-MNIST

La base de données MNIST (Modified National Institute of Standards and Technology) est un ensemble d'images totalement annoté représentant des chiffres manuscrits allant de 0 à 9. Un échantillon d'images est fourni en figure 2.4 (LeCun et al., 1998).

MNIST utilisé pour la classification comporte les propriétés suivantes :

— Nombre d'images entraînement : 60 000
— Nombre d'images de test : 10 000
— Dimensions ($C \times H \times L$) : $1 \times 28 \times 28$
— Nombre de classes : 10
— I.i.d. : OUI

Figure 2.4 – Exemples d'images pour MNIST.

Une version étendue E-MNIST est proposée par Cohen et al., 2017. Toujours composée de caractères manuscrits en dimension $1 \times 28 \times 28$, elle intègre aussi les lettres de l'alphabet latin pour un total de 814 255 images. Pour chaque lettre de A à Z, il est possible de considérer une classe ou deux selon si on prend en compte la distinction majuscule/minuscule. Dans sa totalité, les classes ne sont pas représentées de manière égale et donc le E-MNIST n'est pas i.i.d. L'ensemble des images d'E-MNIST représentant les chiffres est strictement disjoint de MNIST.

Plusieurs scénarios sont donc possibles selon les classes considérées et si l'hypothèse i.i.d. doit être respectée. Les différentes partitions possibles sont résumées en table 2.1. Le cas *ByClass* considère toutes les classes possibles : [0-9], [a-z] et [A-Z] pour un total de 62 classes. *ByMerge* et *Balanced* fusionnent les classes pour lesquelles les majuscules sont très proches des minuscules (C, I, J, K, L, M, O, P, S, U,V, W, X, Y et Z) d'où la réduction à 47 classes.

Partition	ByClass	ByMerge	Balanced	Letters	Digits	MNIST
Entraînement	697 932	697 932	112 800	88 800	240 000	60 000
Test	116 323	116 323	18 800	14 800	40 000	10 000
Classes	62	47	47	26	10	10
i.i.d.	Non	Non	Oui	Oui	Oui	Oui

TABLE 2.1 – Synthèse des différentes partitions possibles pour E-MNIST.

2.7.2 CIFAR

CIFAR-10 et CIFAR-100 (KRIZHEVSKY, 2009) sont deux sous-ensembles annotés de la base d'images « 80 millions tiny images ». Chaque élement est une image RVB de faible résolution ($3 \times 32 \times 32$) représentant un unique objet. Ce dataset est utilisé pour la reconnaissance d'objet avec CIFAR-10 (resp. CIFAR-100) constituant un problème à 10 classes (resp. 100 classes).

CIFAR-10 contient un total de 50 000 images d'entraînement pour 10 000 de test. Le dataset et ses classes sont illustrées en figure 2.5.

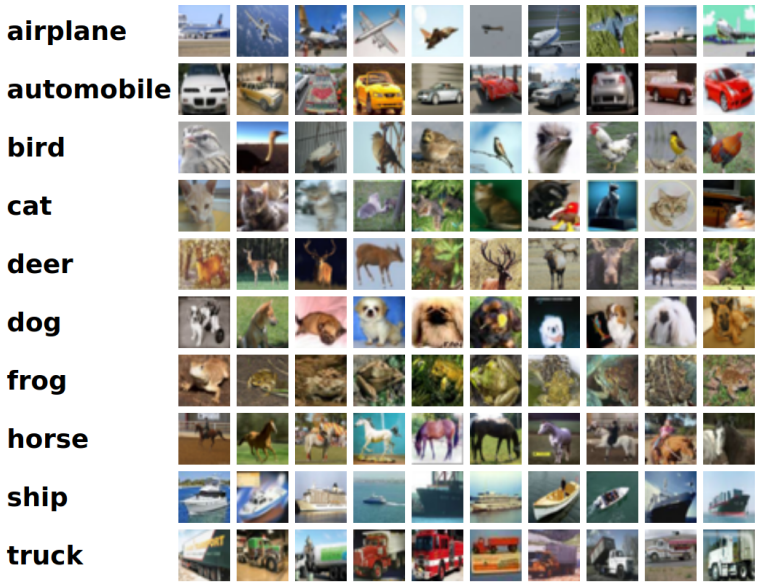

FIGURE 2.5 – Les 10 classes de CIFAR-10 avec des exemples.

Les données de CIFAR-100 se présentent sous le même format mais étendent le problème à 100 classes. Le nombre total d'images étant le même, cela divise le nombre d'exemples par classe (500/classe pour l'entraînement). Les 100 classes peuvent être regroupées en 20 super-classes (e.g. la super-classe « grand carnivore » regroupe les classes « ours », « léopard », « lion », « tigre » et « loup »). Ce *dataset* requiert donc des capacités de discrimination plus affinées pour traiter les classes proches.

2.7.3 STL-10

Coates et al., 2011 s'inspirent de STL-10 pour concevoir un *benchmark* de reconnaissance d'images adapté aux méthodes non-supervisées. STL-10 est divisé en deux parties. D'une part, un ensemble annoté, composé des mêmes classes que CIFAR-10, avec pour chaque catégorie, 500 images d'entraînement et 800 de test. D'autre part, STL-10 met à disposition 100 000 images sans labels à exploiter de manière non-supervisée. Toutes les images sont issues d'ImageNet. Les images non-annotées sont donc de distribution similaire. Elles incluent les classes de la partie annotée mais couvrent un large panel de nouvelles catégories proches (e.g. d'autres classes d'animaux, de véhicules...).

La faible résolution des images CIFAR étant parfois considérée comme limitant pour l'apprentissage de caractéristiques visuelles, STL-10 utilise des images en $3 \times 96 \times 96$. Cela permet de gagner en détails tout restant raisonnable en besoins matériels.

2.7.4 ImageNet

ImageNet (Russakovsky et al., 2015) est une immense banque d'images dédiée à la reconnaissance d'objets. Elle contient plus de 14 millions d'images annotées et organisées selon la taxonomie des noms de WordNet.

Pour les *benchmark*, il est d'usage de se limiter à un ensemble i.i.d. de 1000 classes définies lors du ILSVRC (ImageNet Large Scale Visual Recognition) challenge 2012. Ce *dataset* contient près de 1,3 millions d'images d'entraînement, 50 000 pour la validation et 150 000 pour le test. Le test étant restreint au challenge, les VT ne sont pas publiques. Les résultats rapportés dans la littérature concernent l'ensemble de validation.

Les images ont une résolution variable. Pour des raisons pratiques, il est établi de les redimensionner au format $3 \times 224 \times 224$.

A noter qu'en général, les termes ImageNet, ImageNet-1000 ou ILSVRC2012 sont interchangeables et réfèrent tous au *dataset* de 1000 classes du challenge et non à la banque d'image complète. Dans la suite, nous utiliserons aussi un sous-ensemble composé des 100 premières classes uniquement : ImageNet-100.

Ridnik et al., 2021 ont récemment publié la version étendue du *dataset* avec les 14 millions d'images couvrant plus de 21 000 classes : ImageNet-21K.

2.7.5 Version Incrémentale ou faiblement supervisée

Dans nos expérimentations, nous utilisons ces *datasets* pour évaluer des méthodes d'incrémental de classes (IC). Cela requiert de créer un partitionnement de l'ensemble de données en fonction de ses classes afin de créer le continuum d'apprentissage.

Sauf si précisé autrement, toutes les expérimentations se font à pas incrémental s fixe. Un benchmark d'incrémental de classes consiste donc à préciser le *dataset* et son pas d'incrément.

De même, nous évaluerons parfois des méthodes sur leur robustesse à traiter des problèmes avec une supervision limitée, i.e. peu d'annotations disponibles. Pour cela, on considère un dataset usuel et sous-échantillonne son nombre de labels. Une notation « DATASET-X% » indique que l'on utilise les données de DATASET avec seulement X% des labels.

Voici une liste d'exemples qui seront couramment utilisés dans la suite :

— MNIST avec $s = 2$: apprentissage IC sur MNIST avec introduction des classes 2 par 2. MNIST étant un problème à 10 classes, cela représente 5 sessions.

— CIFAR-100-full avec $s = 10$: IC sur les 100 classes de CIFAR-100 avec un incrément de 10 classes à chaque session.

— CIFAR-100-20% avec $s = 10$: protocole précédent avec seulement 100 exemples/classe au lieu des 500 disponibles habituellement.

— ImageNet-100 avec $s = 10$: IC sur les 100 premières classes d'ImageNet avec un pas de 10.

— ImageNet-100-10% avec $s = 10$: protocole précédent avec 130 exemples/classe

Chapitre 3

Apprentissage de représentations pour l'apprentissage incrémental

Contents

3.1 Introduction

L'apprentissage continu, dans le cadre d'un incrémental de classes, est un problème difficile pour les réseaux de neurones artificiels. La littérature présente souvent l'oubli catastrophique comme la problématique majeure à traiter pour résoudre l'apprentissage continu. Les causes de l'oubli sont cependant difficiles à identifier.

Apprendre un DNN requiert généralement d'accorder une attention particulière aux représentations apprises, notamment si l'application cible est compliquée ou pauvre en données. Intuitivement, de meilleures représentations sont censées être bénéfiques à l'apprentissage incrémental. Nous proposons dans un premier temps d'établir le lien entre représentations apprises incrémentalement et oubli catastrophique. Puis nous nous intéressons à une méthode d'apprentissage continu combinant à la fois l'apprentissage incrémental classes et apprentissage auto-supervisé de représentations visuelles sur des données non-annotées.

Le chapitre s'organise autour des points suivant :

— Une décomposition des composantes du modèle afin d'établir le rôle de chacune dans l'oubli catastrophique. On expose notamment le lien entre représentations apprises et oubli afin de motiver notre objectif de construire des représentations adaptées à un processus incrémental.

— La présentation notre hypothèse principale : l'accès à des données non-annotées pendant l'apprentissage incrémental. Le choix des données non-annotées à exploiter doit être justifié et respecter les contraintes imposées par l'incrémental de classes.

— Une étude expérimentale préliminaire qui s'intéresse à la méthode la plus immédiate pour apprendre des représentations : le pré-entraînement.

— La présentation de l'AAE, un algorithme qui permet à la fois de choisir la structure de l'espace des représentations mais aussi d'exploiter les données non-annotées grâce à une tâche de reconstruction.

— La mise en place d'un apprentissage incrémental avec semi-supervision et régularisation des représentations avec l'AAE. Deux variantes de l'AAE sont présentées et utilisées pour notre étude expérimentale.

3.2 Motivations

Les réseaux de neurones utilisés en classification reposent généralement sur une architecture, ou *backbone*, composée de deux modules : un encodeur f de paramètres Φ suivi d'une tête de classification g de paramètres W. La combinaison des deux constitue le modèle complet $\Theta = (\Phi, W)$. Le rôle de l'encodeur est d'apprendre la projection de l'espace image vers un espace de représentation de plus faible dimension. L'encodeur peut

se désigner par le terme extracteur de *features* : à partir d'une image, il extrait un vecteur de représentation, ou de *features*, encodant l'information de l'image à un différent niveau d'abstraction.

L'une des forces du *deep learning* en image provient de la capacités des CNN à apprendre à extraire des *features* pertinentes via une succession de couches convolutives. Le classifieur quant à lui est souvent une simple couche linéaire entièrement connectée (couche FC) voire une succession de couches FC (dans ce cas on parle de Perceptron Multi-Couche ou MLP). Son rôle dans le système est d'apprendre à tracer les frontières de décision au sein de l'espace de représentation défini par l'encodeur en amont.

Lors d'un entraînement hors ligne *end-to-end*, l'optimisation des deux modules est conjointe. L'objectif d'un modèle pour optimiser un objectif de classification peut se résumer de la façon suivante : le classifieur définit les frontières de décision dans l'espace de représentation et simultanément, l'encodeur modifie cet espace pour séparer aux mieux les différentes classes.

L'entraînement batch rend cela possible car à chaque itération, le modèle fait une passe sur l'ensemble des classes. Autrement dit, il a une vision globale du problème de classification et peut donc organiser toutes les classes relativement entre elles dans son espace de représentation. En exploitant ensuite globalement les données annotées projetées dans l'espace latent, le classifieur peut simplement tracer ses frontières avec un processus analogue à un classifieur *One-versus-All* (OvA).

La difficulté de l'apprentissage incrémental provient du fait que cette vision globale n'est pas praticable. Dans le cas de l'incrémental de classes, l'algorithme d'apprentissage n'a accès qu'à des sous-problèmes de classifications disjoints à chaque session. Son rôle est de les apprendre successivement et de parvenir ensuite à les intégrer en une unique tâche.

3.3 Représentations et Oubli Catastrophique

3.3.1 Vers de meilleurs représentations pour l'incrémental

On s'intéresse dans cette section à la complexité du passage à l'incrémental. En incrémental de classes, une session est analogue à un entraînement batch sur les nouvelles classes uniquement. De ce fait, apprendre à discriminer les nouvelles classes entre-elles se fait naturellement. Cependant, la difficulté apparaît lorsqu'il faut tenir compte des sessions précédentes. Pour le modèle, il s'agit de savoir comment positionner les nouvelles classes par rapport aux anciennes. L'oubli catastrophique peut s'interpréter comme la résultante des phénomènes suivant :

— Au niveau de l'espace latent, sans aucune régularisation, le modèle va totalement ignorer les anciennes classes et modifier l'espace pour être optimisé uniquement sur

les nouvelles.

— Pour le classifieur, si l'espace de représentation est modifié, les frontières précédemment apprises entre les anciennes classes ne sont plus pertinentes.

— Le classifieur, via son processus OvA, ne peut que placer les frontières entre les classes auxquelles il a accès. Sans répétition, il ne peut ni définir celles entre les anciennes et les nouvelles classes ni mettre à jour les frontières précédemment établies.

Contrer l'oubli catastrophique revient à trouver une solution à ces différents phénomènes.

Au cours de ce chapitre, on s'intéresse à une régularisation intervenant au niveau de l'encodeur f. L'objectif est de régulariser l'espace de représentation pour que les nouvelles classes s'y organisent sans créer d'interférences avec les anciennes. L'intuition derrière cet objectif est que l'incrémental de classes devient immédiat s'il est possible de construire un espace de représentation idéal, c'est-à-dire un espace où dès qu'une nouvelle classe est introduite, elle est attribuée à une région libre linéairement séparable des autres.

Un encodeur considéré comme idéal est un encodeur qui requiert un minimum d'adaptation pour apprendre la tâche principale. Pour cela, il est courant d'évaluer la qualité des *features* apprises par un encodeur par une méthode non-paramétrique telle que les k-plus proche voisins (k-NN).

On a donc ici un parallèle entre les domaines de l'incrémental et de l'apprentissage des représentations au sens où, si un encodeur pré-appris est compétitif avec un classifieur non-paramétrique, la performance finale sera équivalente peu importe que les données soient exploitées incrémentalement ou non.

Autrement dit, cela revient à disposer d'un encodeur idéal θ_{ideal} et pour toute session i, $\theta_i = \theta_{ideal}$. Cela permet une régularisation ultime des paramètres contre l'oubli en les gardant figés tout au long de l'apprentissage.

En pratique, le moyen de se rapprocher du θ_{ideal} est d'entraîner l'encodeur en batch supervisé. C'est ainsi qu'on définit la borne supérieure de performance d'un modèle. On peut cependant mentionner que les techniques d'apprentissage des représentations récentes s'en rapprochent de plus en plus grâce à l'auto-supervision.

3.3.2 Améliorer les représentations pour l'apprentissage incrémental

Le pré-entraînement pour initialiser la première session

Un pré-entraînement correspond simplement à ajouter une phase d'apprentissage en amont sur un *dataset* annexe. On génère ainsi un ensemble de poids Φ_0 qui sert d'initia-

lisation à l'encodeur pour apprendre la première session sur les données D_1.

Le pré-entraînement a donc un impact direct sur la première session. Cependant, son influence sur les sessions plus tardives reste à prouver. En effet, l'optimisation du modèle à chaque session i se fait en partant des poids Φ_{i-1} et les met à jour en fonction de la log-vraisemblance $\log p(\Theta_i|D_i)$ avec $\Theta_i = (\Phi_i, W_i)$. L'oubli catastrophique provient justement des poids du modèle qui sont optimisés uniquement sur les nouvelles données sans tenir compte des anciennes classes.

On peut donc questionner l'intérêt d'un pré-entraînement pour l'apprentissage incrémental. En effet, si pré-entraîner s'assimile juste à ajouter une session 0 qui précède la 1, on a simplement rajouté une étape dans le processus incrémental. On peut alors s'attendre à ce que l'oubli catastrophique qui se manifeste au fil des sessions vienne totalement annihiler les bénéfices apportés par les poids pré-appris.

L'intérêt pour le pré-entraînement en incrémental de classe

Malgré les réticences envers le pré-entraînement exposé dans le paragraphe précédent, celui-ci reste une piste à explorer. En effet, cette question de l'initialisation pour les méthodes incrémentales gagne récemment en traction.

En apprentissage batch, le pré-entraînement est l'approche de référence pour traiter des problèmes où les données sont insuffisantes pour apprendre les représentations requises pour traiter le problème. Les difficultés d'apprentissage sont notamment présentes lorsque le nombre de paramètres du modèle est disproportionné par rapport aux nombre d'exemples disponibles.

Dans le domaine de l'incrémental de classes, des études reprennent cette idée que l'initialisation du DNN est un point crucial (Douillard et al., 2020 ; Prabhu et al., 2020). Le contexte incrémental, avec ses contraintes sur les données, impose souvent de travailler avec peu d'exemples à chaque session, rendant parfois difficile d'apprendre la première session. De ce fait, un pré-entraînement peut s'avérer simplement obligatoire pour certaines méthodes ou certains *datasets*. Par exemple, le bilan du challenge CVPR 2020 (Lomonaco et al., 2020) sur CORe50 montre que l'intégralité des soumissions partent d'un DNN pré-entraîné sur ImageNet supervisé, CORe50 seul ne contenant pas assez de données pour entraîner un grand modèle (ResNet-50 et supérieur). De même, quelques approches récentes (Douillard et al., 2020 ; Prabhu et al., 2020) sont conçues pour fonctionner avec un régime *from-half*. C'est-à-dire que le modèle est entraîné en batch sur un grand nombre de classes avant de commencer la phase incrémentale. Par exemple, pour un problème à 100 classes, 10 sessions en approche standard dite *from-scratch* consistent à faire un incrémental avec un pas $s = 10$. Un training *from-half* commence à apprendre les 50 premières classes en batch pour ensuite faire 10 sessions

avec $s = 5$. Le terme *from-half* vient de cette séparation $50-50$ courante pour les *datasets* à 100 classes mais les proportions peuvent changer selon le *benchmark*. Dans tous les cas, que l'on utilise un pré-entraînement ImageNet ou bien un paradigme *from-half*, le DNN au point de départ de l'incrémental possède déjà des représentations optimisées.

3.3.3 La sensibilité des classifieurs discriminatifs à l'oubli catastrophique

Comme introduit en section 2.4.3, la couche FC utilisée dans les méthodes d'apprentissage incrémental avec répétition et distillation présente un défaut majeur : son biais vers les nouvelles classes.

En conséquence, l'entraînement du DNN dans sa globalité est réalisé *end-to-end* avec la couche FC biaisée. On peut s'attendre à ce que la rétro-propagation du biais jusqu'aux poids de l'encodeur fasse lui aussi oublier les anciennes classes au profit des nouvelles.

Le biais du classifieur FC final étant un facteur important de l'oubli, il est donc crucial de le prendre en compte dans notre régularisation via l'apprentissage des représentations. L'idée est que, encodeur et classifieur ont une relation d'interdépendance lors de leur optimisation via l'alternance des passes *forward* et *backward* de l'apprentissage. De ce fait, même si le classifieur peut avoir un impact négatif sur les représentations, inversement, des représentations régularisées peuvent aider à limiter l'oubli du classifieur.

3.4 Utiliser des données externes non-annotées pour l'incrémental de classes : hypothèses et contraintes

Avant de proposer une étude expérimentale, il est important de discuter d'une hypothèse majeure pour nos contributions : nous modifions le scénario de l'incrémental de classes en ajoutant l'hypothèse que des données non-annotées sont accessibles en parallèle des données annotées.

3.4.1 L'origine des données non-annotées pendant l'incrémental

L'incrémental de classes, par définition, impose une forte contrainte sur le flux de données. Il doit être partitionné en sous-ensembles dont les classes et les images sont strictement disjointes. Ce scénario garantit ainsi que les données introduites n'ont jamais été vues précédemment par le système et qu'elles ne sont pas présentes dans les futures sous-ensembles d'entraînement. Le seul moyen d'avoir une répétition est donc un stockage annexe via une mémoire dédiée.

En apprentissage des représentations pour un entraînement batch, les contraintes sont parfois lâches entre la tâche source et la tâche cible. Par exemple, on peut très bien pré-entraîner en supervisé sur ImageNet voire ImageNet-21K puis transférer les features sur un autre *dataset* de reconnaissance d'objet contenant des classes similaires. L'un des *benchmark* standard en semi-supervisé (Zhai et al., 2019) consiste à diviser un dataset en deux parties disjointes, chacune se distribuant de manière i.i.d. sur l'ensemble des classes. Par exemple, sur ImageNet, on conserve 10% des données annotées (130 images/classe) pour faire l'ensemble annoté et tout le reste en données sans étiquette. Dans ce cas là, on à le cas idéal où les distributions des classes décrites par l'ensemble avec et celui sans labels sont identiques. Dans STL-10, on propose un cas plus réaliste où les données non-annotées couvrent un ensemble de classes plus grand que celui contenu dans le *dataset* annoté.

Notre objectif est donc de proposer un cadre où notre hypothèse de données non-annotées annexes respecte les critères du paradigme incrémental.

Premièrement, l'idée même de l'existence de ces données est à discuter. La logique de l'incrémental est de pouvoir traiter des problèmes avec peu de données. Considérons deux familles d'applications :

1. Annotations rares : les données sont une ressource abondante. Cependant, la tâche d'annotation est coûteuse rendant impossible une annotation dense du *dataset*. L'incrémental est donc régulé par le processus d'annotations.

2. Données rares : pour certaines applications, les données en elles-mêmes sont rares ou limitées pour des raisons de confidentialité. Le facteur limitant est donc le régime de collecte des données.

La condition cruciale pour considérer des données non-annotées est qu'elles soient accessibles à moindre coût. Pour le premier cas, la condition est facilement réalisable. Cela concerne des applications dont la collection est souvent automatisée ou « crowdsourcée », générant ainsi des quantités conséquentes d'images brutes dans le même domaine et avec une distribution similaire à celle de la tâche principale. L'intérêt d'une solution capable d'exploiter ces données en plus de l'ensemble annoté est donc immédiat. Pour la deuxième famille, cette condition est difficilement atteinte. L'alternative est de considérer un dataset annexe, généralement de la première famille, au prix de devoir considérer les problématiques de transfert entre distributions et domaines différents. Dans la suite, nous considérons principalement le cas (1).

3.4.2 Les scénarios possibles pour l'incrémental de classes avec données non-annotées

On note $\mathcal{U}$ le flux de données non-annotées et $\mathcal{C}_u$ les classes représentées dans $\mathcal{U}$. A chaque session, deux possibilités : $\mathcal{U}$ est accessible par le modèle dans son intégralité ou seul un sous-ensemble de $\mathcal{U}$ est échantillonné au début de la session. Cela dépend si l'on considère $\mathcal{U}$ comme un *dataset* fixe stocké en mémoire ou bien un flux/banque de données que l'on échantillonne sur demande.

Dans notre scénario d'incrémental de classes, pour reprendre les notations de la section 2.3.1, on a $\mathcal{X}$ le flux de données annotées contenant les classes $\mathcal{C} = [\![1, N]\!]$. $\mathcal{X}$ respecte tous les critères de l'incrémental de classes ; les classes de $\mathcal{C}$ sont donc apprises incrémentalement. Dans le cas où $\mathcal{U}$ et $\mathcal{X}$ appartiennent au même domaine, le choix des contraintes sur $\mathcal{U}$, notamment sur $\mathcal{C}_u$, permettent d'envisager différents scénarios :

1. **Classes communes** : $\mathcal{C} = \mathcal{C}_u$. Le scénario idéal où $\mathcal{U}$ décrit parfaitement la tâche principale mais sans annotations. Le modèle est donc directement exposé à la distribution globale de la tâche principale. Il se produit donc ce qu'on appelle des « fuites d'informations » pendant l'incrémental au sens où il pourra observer des instances avant même d'apprendre leur classe et qu'il pourra revoir des exemples associés à des classes passées. Ce scénario est donc le plus avantageux pour apprendre les classes en incrémental. C'est le cas où l'on sait avoir collecté l'ensemble des données requises à une tâche tandis que le travail d'annotations est progressif.
2. **Classes incluses** : $\mathcal{C} \subset \mathcal{C}_u$ ou plus généralement $\mathcal{C} \cup \mathcal{C}_u \neq \emptyset$. En pratique, la constitution de $\mathcal{U}$ se fait à l'aveugle sans contrôle sur $\mathcal{C}_u$. Puisqu'on échantillonne des données du même domaine, il est fortement probable de sélectionner des instances de $\mathcal{C}$. C'est l'approche la plus réaliste. Cependant, même si le phénomène est mitigé, il y a toujours des fuites d'informations entre les sessions comme dans le scénario précédent.
3. **Classes disjointes** : $\mathcal{C} \cup \mathcal{C}_u = \emptyset$. On accède à des données annexes à la distribution similaires mais de classes disjointes. Dans ce scénario, aucune fuite n'est possible et l'on se rapproche donc des critères de l'incrémental de classes supervisé. Il est artificiel car il requiert la connaissance de $\mathcal{C}_u$ et de $\mathcal{C}$ en amont de l'incrémental de classes.

Ces trois scénarios ont donc leur cadre d'application propre, notamment en termes de classes communes et classes incluses. L'évaluation de classes disjointes est cependant nécessaire pour montrer, dans le cas où la fuite est impossible, quels sont les bénéfices d'avoir accès à une plus grande diversité visuelle pour la qualité des représentations et l'oubli catastrophique.

3.4.3 Exploiter les données non-annotées

A chaque session i, on dispose donc de l'ensemble des nouvelles données annotées D_i issu du flux $\mathcal{X}$, d'un éventuel buffer mémoire $\mathcal{B}_{i-1}$ et d'un *dataset* annexe non-annoté $\mathcal{U}$. Cela permet d'augmenter considérablement le nombre de données d'apprentissage pour une session mais nécessite une mise en place d'une solution adaptée pour exploiter les données de $\mathcal{U}$.

La littérature du semi-supervisé propose une multitude de solutions capables d'apprendre simultanément sur des données annotées et non-annotées. Ces méthodes ont fait leurs preuves en entraînement batch pour ce qui est de renforcer la tâche principale. Dans un contexte continu, on s'interroge en plus sur la capacité d'un paradigme semi-supervisé pour régulariser l'oubli catastrophique et pour préparer l'apprentissage des classes futures.

On propose donc de s'intéresser à l'incrémental de classes semi-supervisé qu'on illustre par la figure 3.1.

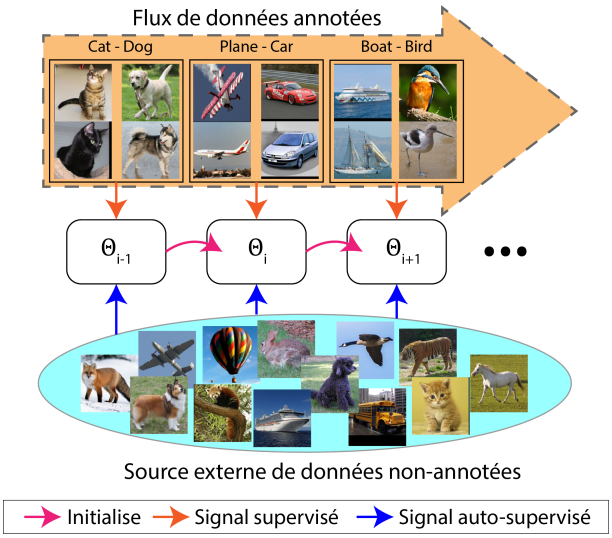

FIGURE 3.1 – Schématisation de l'incrémental de classes semi-supervisé.

3.5 Étude expérimentale préliminaire

3.5.1 Objectif

Maintenant qu'on a établi que des données non-annotées sont disponibles, il faut pouvoir les exploiter de façon bénéfique à l'apprentissage incrémental. Avant de directement passer à l'étape de la semi-supervision, nous proposons d'approfondir l'idée introduite en section 3.3.2, celle qu'une meilleur initialisation peut impacter le déroulement de l'incrémental. Pour cela, un pré-entraînement représente la façon la plus immédiate de profiter

des données supplémentaires.

Nous proposons une étude expérimentale préliminaire autour de trois problématiques :

1. Des représentations pré-apprises ont-elles un impact positif sur processus incrémental, i.e. réduisent l'oubli ?
2. Existe-t-il des représentations idéales pour l'initialisation de l'incrémental ?
3. L'architecture des modèles est-elle contrainte par le processus incrémental ?

Les expérimentations suivantes concernent deux méthodes de l'état-de-l'art : iCaRL (Rebuffi et al., 2017) et WA (Zhao et al., 2020). iCaRL est la méthode de référence pour la répétition combinée à la distillation de connaissances. L'inférence se fait via plus proche centroïde (non-paramétrique). WA (*Weights Aligning*) peut s'interpréter comme un iCaRL où, au lieu de supprimer la couche FC pour la classification, on corrige son biais avec un étape supplémentaire de post-traitement. Pour cela, WA uniformise la norme des poids associés à chaque classe contenu dans W_i en les multipliant simplement par un facteur correcteur calculé en fin de session.

3.5.2 Le choix de l'initialisation

La première étape est donc de s'intéresser à la plus basique des méthodes d'apprentissage des représentations : le pré-entraînement. Aller chercher une meilleure initialisation est devenu presque systématique pour les apprentissages batch en raison de sa simplicité à mettre en place et du gain direct en performance sur la tâche principale.

Nous considérons le *benchmark* CIFAR-100 avec un pas $s = 10$. Nous évaluons deux types d'initialisation pour les poids de l'encodeur Φ_0 :

— Entraînement batch (EB) : on récupère l'encodeur d'un modèle appris conjointement en supervisé sur l'intégralité de CIFAR-100.
— RotNet : l'encodeur est appris sur la tâche auto-supervisée RotNet avec les images issues d'ImageNet.

Dans chaque cas, le classifieur incrémental W_0 est initialisé aléatoirement au début de la première session.

A noter que le cas initialisation EB ne respecte en rien l'hypothèse incrémental de classes puisque cela signifie que l'on a appris le modèle en batch supervisé sur les données mêmes que l'on voit ensuite en incrémental. Cependant, c'est un moyen simple de construire un encodeur proche de ce qu'on pourrait qualifier d'encodeur « idéal », puisque ce sont les représentations qui obtiennent la meilleure précision expérimentalement atteignable.

Résultats

Pour iCaRL et WA, on se place dans 4 configurations possibles : la méthode originale avec une initialisation aléatoire, deux versions avec l'encodeur EB avec les poids entraînables ou non, ainsi que la version avec l'encodeur pré-entraîné via RotNet. Les résultats sont inclus dans les deux graphes de la figure 3.2.

La borne supérieure de la précision est illustrée par la courbe « batch » qui, en chaque point d'abscisse C_i, correspond à un modèle entraîné en batch sur les C_i classes.

Le premier constat est que, dans le cas où les représentations sont idéales (EB), il est préférable de ne pas les modifier lors de l'incrémental. Dans ce cas, WA arrive à se rapprocher de la précision batch en entraînant seulement le classifieur. Pour iCaRL, cette configuration n'est pas représentative puisque le classifieur est non-paramétrique, aucun poids n'est appris. La légère baisse en précision comparée au batch montre juste que le classifieur plus proche centroïde avec le buffer est moins discriminant que le classifieur FC. On peut néanmoins conclure qu'un encodeur parfait qui ne requiert pas d'entraînement rend l'incrémental de classe immédiat.

Si on considère les représentations EB et RotNet en laissant la possibilité au modèle de calibrer les poids de l'encodeur, on observe directement la différence attendue de qualité entre les deux représentations. Un encodeur entraînable, même pré-appris, reste sensible à l'oubli catastrophique. Finalement l'impact d'un pré-entraînement est différent selon les méthodes, iCaRL montre un gain en précision consistant alors que pour WA, la performance reste très similaire.

Cette expérimentation montre le rôle déterminant de l'encodeur dans l'apprentissage incrémental. Une meilleure initialisation est un moyen d'augmenter la précision incrémentale. L'encodeur reste cependant très sensible à l'oubli lorsqu'il est entraînable. Même dans le cas avec l'initialisation sur des représentations apprises en batch, la dégradation de la précision indique que le modèle est incapable de régulariser lui-même un espace de représentation déjà adapté.

3.5.3 Pré-entraînement et taille des modèles

Le pré-entraînement avant un entraînement batch est presque obligatoire pour certaines architectures avec un grand nombre de paramètres ou pour des applications avec très peu de données annotées. La cause est inhérente aux algorithmes d'optimisation batch : entraîner un grand nombre de paramètres avec peu de données est enclin à ne pas converger ou à surapprendre sur l'ensemble d'entraînement.

L'apprentissage incrémental induit que, lors de chaque session, on se retrouve naturellement avec peu de données. Cela implique de prendre en compte un éventuel surap-

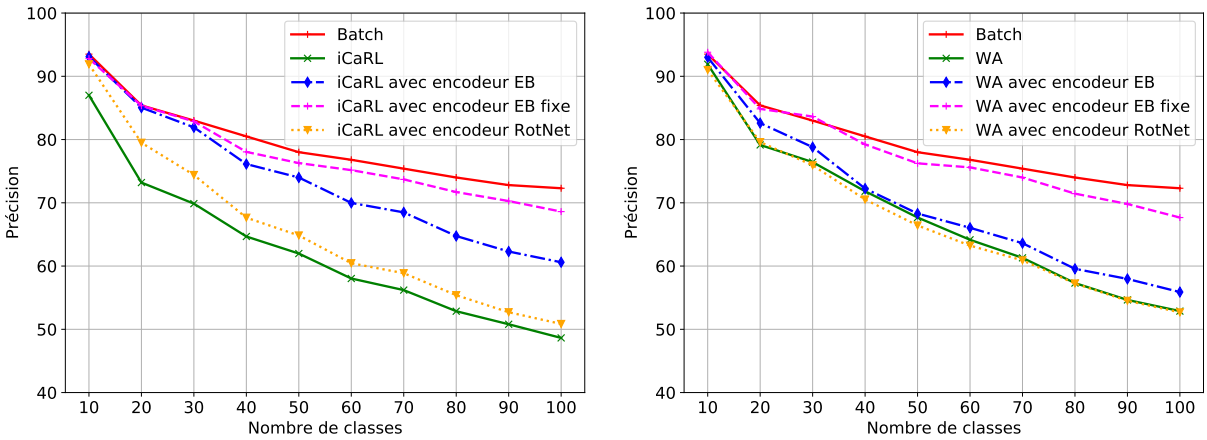

FIGURE 3.2 – Incrémental de classes sur CIFAR-100 avec un pas de 10. Les méthodes évaluées sont iCaRL (à gauche) et WA (à droite). L'initialisation des encodeurs pré-entraînés se fait avec des poids appris en batch sur CIFAR-100 lui-même (en bleu) ou appris en auto-supervisé avec RotNet sur ImageNet (en jaune). Pour les poids appris en batch, on propose aussi une version où l'encodeur à ses paramètres figés, i.e. non-entraînables (en magenta). On reporte aussi la précision obtenue par les méthodes sans pré-entraînement (en vert) et celle du modèle batch (en rouge).

prentissage sur les nouvelles données mais aussi le manque de diversité visuelle accessible simultanément pour apprendre des *features* visuelles généralisables.

CORe50 (LOMONACO & MALTONI, 2017) par exemple est un *dataset* avec très peu de diversité visuelle : 160 000 images représentant les 50 mêmes instances appartenant à 10 classes différentes. Lors du challenge (LOMONACO et al., 2020) sur CORe50, 100% des 11 finalistes utilisent un modèle pré-entraîné sur ImageNet avec des architectures de type ResNet-50 ($\approx$ 25M de paramètres) ou plus grande.

Autre constat, la grande majorité des méthodes se contente d'utiliser des architectures telles que ResNet-32 (470K paramètres) pour CIFAR ou ResNet-18 (11M paramètres) sur ImageNet. Ces deux architectures sont les plus petits ResNets parmi ceux introduits par HE et al., 2016. Or, les résultats de l'état-de-l'art en entraînement batch encouragent l'utilisation de ResNets, ou de ses variantes (DenseNet, Wide-ResNet, ResNeXt, EfficientNet), avec une taille bien supérieure pour obtenir de meilleurs résultats.

Pour cette expérience, nous essayons simplement d'observer le comportement d'un algorithme incrémental lors d'un passage à l'architecture supérieure. On choisit pour cela de conserver le ResNet-32 couramment utilisé. Un ResNet est caractérisé par deux paramètres entiers : sa profondeur (nombre de couches) et sa largeur (nombre de neurones de la première couche). Le ResNet-32 présenté par HE et al., 2016 correspond en réalité à un ResNet-32-16 : profondeur 32 et largeur 16. On incrémente donc ces deux paramètres. Pour la profondeur, on sélectionne 56, une valeur couramment utilisée sur CIFAR. Pour la largeur, on multiplie par 2 le nombre de neurones de chaque couche. Le nouveau DNN

est de ce fait noté ResNet-56-32 et contient 3.4M de paramètres entraînables.

Pour le pré-entraînement, on utilise le même algorithme RotNet sur ImageNet que dans l'expérience précédente. On retient iCaRL qui est la méthode ayant obtenue la meilleure réponse au pré-entraînement dans l'expérience précédente.

Résultats

Les résultats sont présentés sous forme de courbes de précision incrémentale en figure 3.3. Dans le cas sans pré-entraînement, l'augmentation du nombre de paramètres bénéficie à la précision finale (+2.3%) mais très peu à la précision moyenne (+0.6%). Le manque de données pour entraîner le ResNet-56-32 est particulièrement remarquable pour la première session où la précision est la pire obtenue sur les 10 classes.

Avec pré-entraînement, on observe une différence de comportement entre les deux architectures. Pour les deux, le gain est notable sur les premières sessions. Le ResNet-56-32 qui traitait difficilement la session 1 est désormais le plus précis. Le gain du pré-entraînement pour le ResNet-32 est très vite atténué par l'oubli au fil des sessions. La précision finale est seulement 2.2 points au-dessus de son alternative sans RotNet. Au contraire, ResNet-56-32, grâce au pré-entraînement, montre un gain consistant pour l'intégralité de l'apprentissage.

Un modèle plus grand peut donc être profitable pour l'incrémental dans la mesure où l'on dispose de moyens suffisants pour l'entraîner. Un pré-entraînement auto-supervisé basique pouvant suffire. Par ailleurs, les différents résultats obtenus au cours de nos expérimentations avec le ResNet-32 nous ont poussés à nous questionner sur la pertinence de cette architecture. En effet, elle semble intuitivement adaptée à l'échelle des problèmes incrémentaux où peu de données sont accessibles simultanément. Cependant, l'oubli catastrophique, dont l'amplitude semble amplifiée par le pré-entraînement, est pour nous l'indication d'un manque de stabilité du modèle.

3.5.4 Conclusion de l'étude préliminaire

Les expérimentations préliminaires ont montré qu'un pré-entraînement avait un impact positif sur les représentations. L'hypothèse de pouvoir apprendre de manière supervisée certaines classes en pré-entraînement est cependant contraire au principe même de l'incrémental. Le pré-entraînement non-supervisé est lui aussi bénéfique mais semble surtout bénéficier aux premières sessions. Il devient surtout utile dans le cas des problèmes manquant de données pour faire converger certaines architectures.

A l'instar de l'entraînement batch, dans un cadre applicatif, si un pré-entraînement est faisable, il aura généralement un impact positif et directement compatible avec la grande

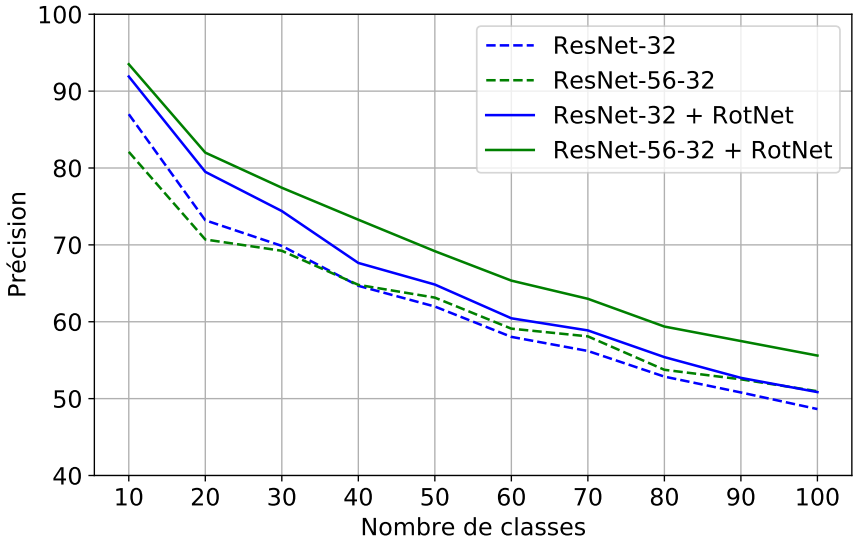

FIGURE 3.3 – Comparaison de la performance d'iCaRL sur CIFAR-100 pour deux *backbones*, ResNet-32-16 et ResNet-56-32. L'expérimentation est faite avec les modèles initialisés aléatoirement ou avec RotNet.

majorité des approches d'incrémental de classes. Cependant, dans le cadre d'une étude de l'incrémental, il apparaît important de cadrer la façon de pré-entraîner. Un modèle dont l'encodeur a déjà appris une partie des classes lors du pré-entraînement peut-il être vraiment considéré comme incrémental ? Pour que le principe de l'incrémental de classe soit respecté, ceci impose de prendre des précautions y compris avec des données non-annotées. Nous avons abordé cette discussion plus en détail en section 3.4.

En définitive, ces expérimentations préliminaires ont servi à valider l'intérêt d'introduire une étape d'apprentissage des représentations pour l'incrémental de classes. Le pré-entraînement propose pour cela une solution immédiate. Cependant, le cas où il serait possible de pré-apprendre un encodeur idéal est irréaliste, surtout dans un contexte incrémental où l'on ignore les tenants et aboutissants de la tâche cible au moment de l'apprentissage du système. Un pré-entraînement, dès lors que l'on doit continuer à entraîner l'encodeur, n'apporte pas de régularisation spécifique contre l'oubli. Pour la suite, on cherche à implémenter un apprentissage des représentations en parallèle de l'incrémental, l'objectif étant d'améliorer et de régulariser les représentations tout au long du cycle de vie du modèle. Puisque l'on se place dans une configuration où des données non-annotées sont accessibles aux côtés de celles qui le sont, on s'oriente naturellement vers les solutions semi-supervisées.

3.6 L'Autoencodeur Adverse (AAE)

Nous présentons dans cette section l'Autoencodeur Adverse, la *baseline* que nous allons utiliser pour explorer l'incrémental de classes semi-supervisé. Le choix de l'AAE (Makhzani et al., 2015) est justifié par le fait qu'il propose une solution adaptée pour tout ce qui est régularisation de l'espace des représentations, apprentissage non-supervisé et semi-supervisé, ou apprentissage génératif. On y retrouve donc un environnement directement adapté pour expérimenter les pistes conjecturées pendant l'étude préliminaire.

3.6.1 L'autoencodeur adverse (AAE) : les différents modules

Le modèle proposé par Makhzani et al., 2015 transforme un autoencodeur en un modèle génératif en imposant un prior connu sur la distribution latente.

L'autoencodeur (cf. fig. 3.4) est composé d'un encodeur f et d'un décodeur r de paramètres respectifs Φ et R. Pour une image x, on note z son vecteur latent tel que $z = f(x)$. Soit $p(x)$ la distribution des données, avec f, on génère la distribution $q(z|x)$ ainsi que la distribution a posteriori agrégée $q(z)$ telle que :

$$q(z) = \int_x q(z|x)p(x)dx \tag{3.1}$$

Le décodeur apprend à reconstruire l'image x en partant de z. L'encodeur et le décodeur peuvent être des MLP ou des CNN. Il est d'usage d'imposer une certaine symétrie des architectures entre encodeur et décodeur, i.e. un nombre similaire de couches/paramètres.

La spécificité de l'AAE provient de l'introduction d'un module discriminateur d_z de paramètres Δ_z qui prend en entrée les vecteurs de l'espace latent.

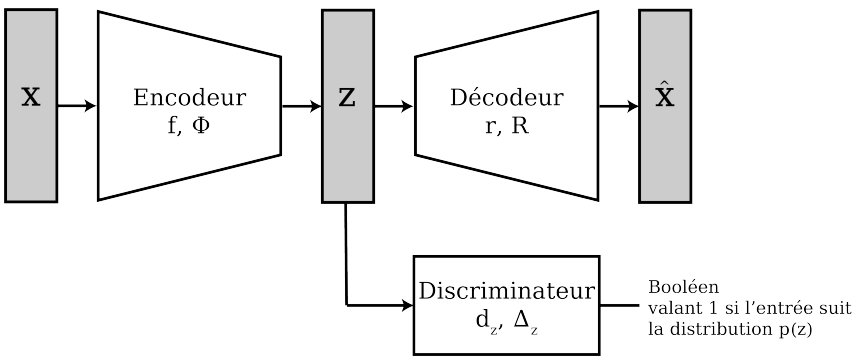

Figure 3.4 – AAE non-supervisé.

3.6.2 Entraînement et Fonctions de coût

L'AAE est en premier lieu un modèle non-supervisé. Pour un ensemble d'images non-annotées $\mathcal{U}$, il a pour tâche principale la reconstruction de l'entrée dont la *loss* quadratique l_{rec} est la suivante :

$$l_{\text{rec}}(\Phi, R, \mathcal{U}) = \sum_{x \in \mathcal{U}} ||x - \hat{x}||_2^2 \qquad \text{avec} \qquad \hat{x} = r(f(x)) \tag{3.2}$$

L'ajout d'une hypothèse forte sur la distribution des variables latentes consiste à choisir arbitrairement un prior $p(z)$ et contraindre $q(z)$ à lui correspondre. Dans le cas du VAE, cela se passe via minimisation de la divergence de Kullback-Leibler (KL) entre $p(z)$ et $q(z)$. Le calcul de la divergence KL devient insoluble pour certains prior, limitant généralement le VAE à un simple prior gaussien $p(z) \sim N(z; 0, I)$. L'AAE propose de remplacer la divergence KL par une *loss* adverse qui permet de généraliser à n'importe quel prior.

Une *loss* adverse repose sur l'utilisation du discriminateur d_z. Son rôle est d'effectuer une vérification d'appartenance à la distribution cible $p(z)$. Si le vecteur latent est appelé z, $d_z(z)$ désigne un booléen valant 1 si z suit la distribution $p(z)$ et 0 sinon.

d_z est un MLP qui requiert d'être optimisé sur sa tâche de discrimination. Pour cela, on utilise un paradigme d'entraînement dit adverse en deux étapes. Premièrement, la phase « positive » : d_z est entraîné à discriminer entre vrais et faux vecteurs. Soit Z un ensemble de vecteurs latents suivant la distribution objectif $p(z)$, on optimise Δ_z avec une fonction adverse générique qui s'exprime de la manière suivante :

$$l_{adv}(\Phi, \Delta_z, \mathcal{U}, Z) = -\sum_{x \in \mathcal{U}} \log(1 - d_z(f(x))) - \sum_{z \in Z} \log(d_z(x)) \tag{3.3}$$

Le discriminateur apprend à catégoriser les vecteurs latents issus de f comme « faux » et les vecteurs générés Z selon la distribution $p(z)$ comme « vrais ». Pendant cette phase, seuls les poids Δ_z sont mis à jour et Φ est fixé.

Vient ensuite la phase « négative », pour laquelle l'encodeur est entraîné à duper le discriminateur. Pour cela, le modèle doit générer des vecteurs latents qui, pour le discriminateur, sont considérés comme appartenant à $p(z)$. Cette fois, Δ_z est fixe et on optimise les poids Φ avec la *loss* :

$$l_{adv}(\Phi, \Delta_z, \mathcal{U}) = -\sum_{x \in \mathcal{U}} \log d_z(f(x)) \tag{3.4}$$

3.6.3 Prise en compte des labels lors de l'entraînement

Dans le cas où les données contiennent des labels, il est intéressant de pouvoir proposer une approche supervisée qui structure l'espace latent en fonction des classes.

La première solution proposée par Makhzani et al., 2015 est de choisir un prior suivant une loi de mélange dont chaque composante est attribuée à une classe. Pour cela, on rajoute un vecteur *one-hot* en entrée du discriminateur (cf. fig 3.5). Le processus d'entraînement est inchangé par rapport à la version non-supervisée.

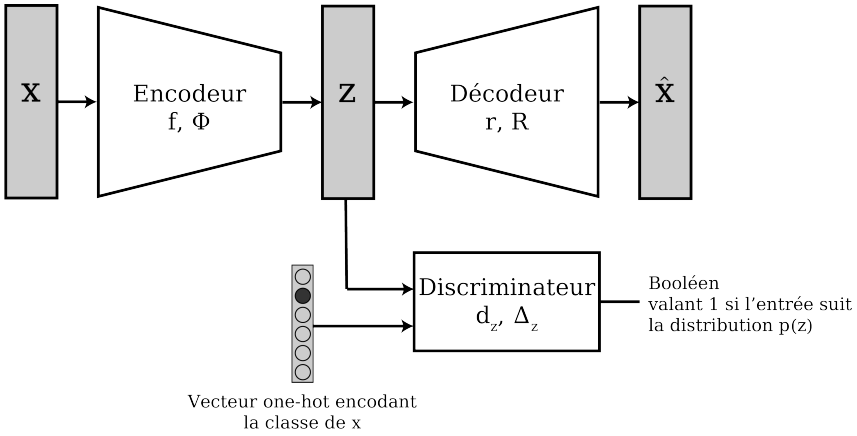

Figure 3.5 – AAE supervisé. Un vecteur one-hot indiquant l'indice de la classe est fourni à l'encodeur. Chaque indice correspond aussi à une composante du prior.

Lorsqu'on présente une donnée avec label, on encode la classe dans le vecteur *one-hot*. Prenons en exemple un problème à 10 classes sur lequel on veut imposer comme prior un mélange de 10 gaussiennes 2D. Une visualisation des données dans l'espace latent 2D donne la figure 3.6.

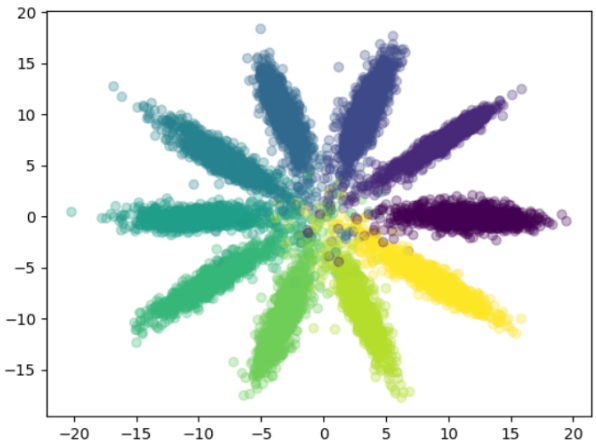

Figure 3.6 – Visualisation l'ensemble de validation de MNIST. Chaque image est encodée dans l'espace de représentation appris par l'AAE, on colore le point associé en fonction de sa classe. L'AAE a été entraîné en batch de manière supervisée sur l'intégralité des données de MNIST. Le prior est un mélange de gaussiennes 2D dont les moyennes sont uniformément réparties sur le cercle de rayon 10.

Ce modèle peut s'assimiler à un classifieur de type génératif. Contrairement à un

classifieur discriminatif qui cherche à approximer la distribution $p(y|X)$, un modèle génératif modélise la distribution $p(X|y)$. L'avantage d'une telle approche est, comme illustré dans l'exemple, de pouvoir attribuer manuellement un voisinage local de l'espace latent à chaque classe. Ajouter une nouvelle classe, c'est ajouter une nouvelle composante au prior dans une région libre en conservant fixe les anciens.

3.6.4 *Baseline* Semi-supervisée

La méthode précédente vise à construire un espace latent où chaque classe est modélisée par une composante. Si le prior est bien respecté, la classification est possible en comparant un vecteur encodé avec la distribution $p(X|y)$ pour chaque classe connue. Makhzani et al., 2015 montrent que cette méthode marche très bien sur un *dataset* simple avec beaucoup d'annotations tel que MNIST, mais perd en capacité de discrimination dès lors qu'on dispose d'un ensemble annoté restreint.

Ils proposent donc une approche dédiée à la classification mais aussi compatible avec la semi-supervision. Pour cela, on ajoute une tête de classification directement dans le modèle au niveau de l'espace latent. La nouvelle architecture (cf fig. 3.7) est de ce fait apte à être optimisée sur une *loss* de classification standard. En parallèle, on continue l'apprentissage régularisé de l'espace latent avec des données sans étiquettes pour renforcer la capacité de discrimination du modèle.

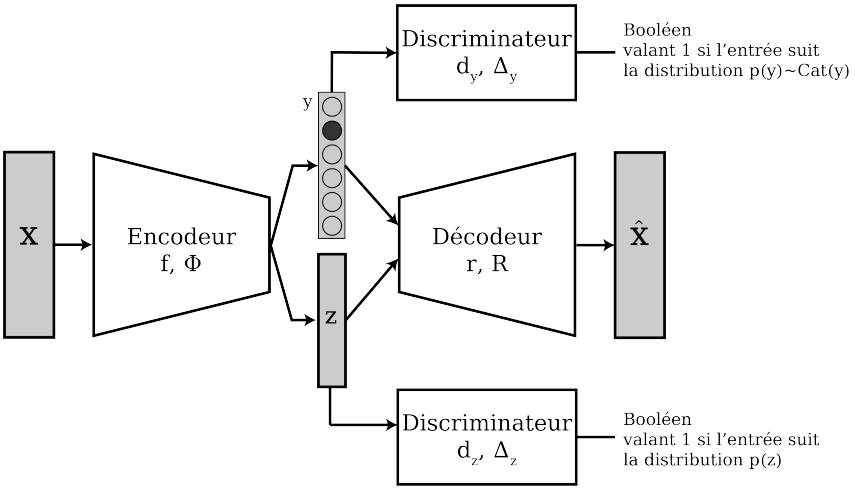

Figure 3.7 – AAE semi-supervisé. L'espace latent est séparé en deux composantes. Une partie encode la classe et l'autre le style. On dit que la représentation est *disentangled*.

Au lieu d'un unique vecteur latent z, on a désormais deux composantes. Un vecteur de probabilités y issu d'un softmax et un vecteur réel z. Sur z, le processus est inchangé, on impose un prior arbitraire, gaussien en général, grâce au discriminateur d_z. Le rôle

de y est d'encoder la catégorie. La *loss* de classification intervient directement à ce niveau puisqu'elle est calculée sur ce vecteur y pour les images annotées. Pour les images non-annotées, la contrainte provient d'un terme adverse. On introduit un nouveau discriminateur, noté d_y. Soit $q(y)$ la distribution associée à la sortie y de l'encodeur, le discriminateur impose un prior $p(y)$ qui suit une distribution dite « *categorical* », i.e. une distribution de vecteurs *one-hot*. On note :

$$p(y) = \text{Cat}(y) \qquad p(z) = \mathcal{N}(z|0, I) \tag{3.5}$$

De ce fait, quelque soit la donnée en entrée, annotée ou non-annotée, on force l'encodeur f à générer un vecteur *one-hot*. Ce vecteur peut s'interpréter comme une prédiction de classe avec un fort taux de confiance. Combiné à la *loss* de reconstruction qui s'applique à la fois aux images annotées et non-annotées, on permet au modèle d'apprendre un modèle génératif qui vient renforcer la tâche de classification. La représentation latente produite est alors dite *disentangled*. Toute l'information de classe est contenue dans y. Le style z doit être complètement indépendant.

D'un point de vue génératif, le décodeur r apprend à générer des images à partir des paires (y, z). Il est donc possible, en contrôlant indépendamment y et z, de générer des images d'une certaine classe avec un même style et vice-versa comme l'exemple en figure 3.8.

FIGURE 3.8 – Génération d'images MNIST dans le cas où les représentations sont *disentangled*. Chaque colonne correspond à une classe imposée en fixant y. Pour chaque ligne, les images générées sont à style z fixé.

Entraînement de l'AAE et fonctions coûts

Le processus d'entraînement est modifié pour incorporer la supervision. Il est désormais constitué de trois phases. Soit $\mathcal{U}$ le *dataset* non-annoté et $\mathcal{D}$ celui annoté. Notez que la sortie de l'encodeur $f(x)$ désigne un couple (y, z). Pour éviter les ambiguïtés, on note dans la suite $f_y(x)$ (resp. $f_z(x)$) la composante de catégorie (resp. la composante de style).

Reconstruction : Apprentissage de f et r via une *loss* d'erreur quadratique.

$$l_{\text{rec}}(\Phi, R, \mathcal{U}, \mathcal{D}) = \sum_{x \in \mathcal{U} \cup \mathcal{D}} ||x - \hat{x}||_2^2 \qquad \text{avec} \qquad \hat{x} = r(f(x)) \tag{3.6}$$

Régularisation : Apprentissage adverse entre encodeur et deux discriminateurs d_y et d_z. Phase positive avec l'encodeur f fixé. Les « vrais » échantillons des distributions prior $p(z)$ et $p(y)$ sont notés respectivement Z et Y :

$$l_{adv}^{+}(\Phi, \Delta_z, \mathcal{U}, \mathcal{D}, Z) = -\sum_{x \in \mathcal{U} \cup \mathcal{D}} \log(1 - d_z(f_z(x))) - \sum_{z \in Z} \log(d_z(z)) \tag{3.7}$$

$$l_{adv}^{+}(\Phi, \Delta_y, \mathcal{U}, \mathcal{D}, Y) = -\sum_{x \in \mathcal{U} \cup \mathcal{D}} \log(1 - d_y(f_y(x))) - \sum_{y \in Y} \log(d_y(x)) \tag{3.8}$$

Puis la phase négative. On fixe les discriminateurs pour régulariser l'encodeur :

$$l_{adv}^{-}(\Phi, \Delta_z, \Delta_y, \mathcal{U}, \mathcal{D}) = -\sum_{x \in \mathcal{U} \cup \mathcal{D}} (\log d_z(f_z(x)) + \log d_y(f_y(x))) \tag{3.9}$$

Supervision : On apprend l'encodeur f sur une fonction d'entropie croisée :

$$l_{clf}(\Phi, \mathcal{D}) = -\sum_{(x,y) \in \mathcal{D}} \log p(y|x, \Phi) \tag{3.10}$$

Pour les deux premières étapes, l'optimisation traite indifféremment les données annotées et non-annotées. La dernière étape supervisée n'est applicable que sur $\mathcal{D}$.

3.7 L'AAE, un *framework* pour l'incrémental de classes semi-supervisé

On définit ici notre démarche pour un incrémental de classes avec répétition exploitant une architecture de type AAE. Pour cela, nous proposons deux adaptations de l'algorithme original. L'algorithme #1 implémente la supervision avec prior de type mélange de gaussienne présentée en section 3.6.3. Elle a pour but d'utiliser un classifieur génératif pour contrôler la position dans l'espace latent de chaque classe. Le #2 se base sur l'approche

semi-supervisée avec les représentations *disentangled* défini en section 3.6.4.

3.7.1 Algorithme #1 : AAE incrémental avec espace latent suivant une loi de mélange

Notre première version de l'AAE pour l'apprentissage incrémental est présentée dans cette section. Il s'agit d'une première approche où l'on combine l'incrémental de classes avec une tâche non-supervisée : la reconstruction.

Cet algorithme reprend l'architecture décrite en section 3.6.3. Ce modèle, pour un prior suivant une loi de mélange, permet de construire un espace de représentation dans lequel chaque classe est modélisée par l'une des composantes de la distribution. Cela rejoint l'idée d'utiliser un classifieur génératif pour modéliser les classes.

La version incrémentale que l'on propose est à architecture fixe. Dans notre implémentation, on initialise un prior suivant une loi de mélange gaussien avec M composantes. Ces composantes constituent donc M clusters possibles. L'idée est donc, à chaque fois qu'une nouvelle classe est présentée au modèle, de l'attribuer à un cluster libre. M représente donc le potentiel maximal du modèle en nombre de classes prédictibles. En théorie, avec une connaissance parfaite du prior, il est toujours possible de rajouter des clusters dans des régions libres de l'espace latent, mais pour rester dans un cas simple, on suppose ici que l'on choisit un M suffisamment grand pour le continuum incrémental à venir. La régularisation contre l'oubli provient ici de la contrainte forte que l'on impose sur la structure de l'espace latent.

L'intérêt d'une telle approche est avant tout de s'intéresser au comportement de l'AAE, dans sa version supervisée la plus simple, lorsqu'on le place dans un contexte d'apprentissage incrémental. On attend une performance moindre de cette approche en termes de précision. Simplement parce qu'elle n'est pas conçue spécifiquement pour résoudre une tâche de classification, mais plutôt pour de la génération. Cependant, en imposant un prior $p(z)$ simple en 2D, on peut proposer une visualisation qualitative de l'espace de représentation.

3.7.2 Algorithme #2 : AAE incrémental semi-supervisé avec représentation *disentangled*

L'architecture suit celle définie en section 3.6.4. Comme pour l'algorithme #1, on travaille à architecture fixe.

Le coeur de cette méthode repose sur le classifieur One-versus-All que l'on initialise avec un nombre M de sorties – de labels – bien plus grand que le nombre de classes attendues. L'incrémental de classes est géré par un mécanisme d'affectation entre classes

et labels. Dès qu'une nouvelle classe est introduite, on lui attribue la sortie non-assignée qui en moyenne produit la réponse la plus forte sur les exemples observés étant donné l'état actuel du modèle. Pour l'inférence, on ne garde actifs que les labels assignés et on ignore les sorties restantes.

Le principe est de compter sur la grande diversité de données accessibles pour produire un *clustering* non-supervisé. En effet, l'espace latent à deux composantes permet de dissocier l'information de classe du style, avec une régularisation adaptée sur les deux. La régularisation avec un prior *categorical* oblige le modèle à prédire une catégorie quelque soit l'entrée. On utilise ce mécanisme pour clusteriser les échantillons indépendamment de leur statut d'annotation. Ainsi, le modèle est capable d'organiser son espace de caractéristiques avec des données non étiquetées, en anticipant l'apparition potentielle de labels correspondant à ces instances par la suite. Via la supervision, on va progressivement spécialiser certains clusters sur des classes précises. Ce signal supervisé est crucial pour forcer le modèle à être discriminatif sur y et éviter la solution triviale où le même label est tout le temps prédit par le modèle et seul z est utilisé pour la reconstruction.

3.7.3 Entraînement de l'AAE en incrémental de classe avec répétition

Soit un incrémental de classes à pas fixe s. On note $\mathcal{U}$ le dataset non-annoté. La session i correspond à l'apprentissage sur l'ensemble annoté D_i contenant les classes $\{(i-1)s + 1, ..., is\}$.

On définit les paramètres suivants : K le budget mémoire total en nombre d'images ainsi que M le nombre de clusters avec lequel on initialise l'AAE. On détaille le processus général de l'incrémental de classe dans l'algorithme 1 et une session d'apprentissage dans l' algorithme 2. Ils sont formulés ici pour l'approche #2 mais facilement adaptable au cas #1 (retrait de Δ_y et de la classification supervisée, ajout des labels en entrée de Δ_z).

Algorithm 1 CLASS_INCREMENTAL

Input : $\mathcal{X}$, $\mathcal{U}$, $p(z)$, $p(y)$, s, K

Initialisation :

$i = 0$

$\Theta_0(\Phi_0, R_0, \Delta_{z,0}, \Delta_{y,0})$ Initialisation de l'AAE

$\mathcal{B}_0 = \{\}$ Initialisation de la mémoire avec un budget K

while nouvelles classes dans $\mathcal{X}$ **do**

  $i \leftarrow i + 1$

  $D_i \leftarrow \{X^{(i-1)s+1}, ..., X^{is}\}$

  ASSIGN_CLUSTER(Θ_{i-1}, D_i) On assigne chaque nouvelle classe au label libre.

  Échantillonnage de U_i depuis $\mathcal{U}$

  Entraînement du modèle :

  $\mathcal{M}_i \leftarrow$ TRAINING_SESSION($\Theta_{i-1}, \mathcal{B}, D_i, U_i$)

  Sauvegarde aléatoire de $\frac{K}{si}$ exemples par classe.

  $\mathcal{B}_i \leftarrow$ UPDATE_MEMORY($\mathcal{B}_{i-1}, D_i, K$)

end while

Algorithm 2 TRAINING_SESSION

Input : $\Theta_{i-1}(\Phi_{i-1}, R_{i-1}, \Delta_{z,i-1}, \Delta_{y,i-1})$, $\mathcal{B}_{i-1}$, D_i, U_i, $p(z)$, $p(y)$

Initialisation :

$D_i \leftarrow \mathcal{B} \cup D_i$ Fusion du buffer avec les données annotées de la session

Notations intermédiaires pour la boucle FOR :

$\Phi, R, \Delta_z, \Delta_y \leftarrow \Phi_{i-1}, R_{i-1}, \Delta_{z,i-1}, \Delta_{y,i-1}$

for each epoch **do**

  Échantillonnage des vrais exemples $Z_i \sim p(z)$

  Échantillonnage des vrais exemples $Y_i \sim p(y)$

  Reconstruction :

  $\Phi, R \leftarrow l_{\text{rec}}(\Phi, R, U_i, D_i)$

  Entraînement des discriminateurs :

  $\Delta_z \leftarrow l^+_{\text{adv}}(\Phi, \Delta_z, U_i, D_i, Z_i)$

  $\Delta_y \leftarrow l^+_{\text{adv}}(\Phi, \Delta_z, U_i, D_i, Y_i)$

  Régularisation de l'encodeur :

  $\Phi \leftarrow l^-_{\text{adv}}(\Phi, \Delta_z, \Delta_y, U_i, D_i)$

  Classification supervisée :

  $\Phi_i \leftarrow l_{\text{clf}}(\Phi, D_i)$

end for

$\Phi_i, R_i, \Delta_{z,i}, \Delta_{y,i} \leftarrow \Phi, R, \Delta_z, \Delta_y$

return $\Theta_i(\Phi_i, R_i, \Delta_{z,i}, \Delta_{y,i})$

Pour les fonctions non détaillées :

— ASSIGN_CLUSTER : pour une nouvelle classe c, on sélectionne simplement le cluster non attribué de Θ_{i-1} qui à la plus forte réponse en moyenne sur X^c.
— UPDATE_MEMORY : processus aléatoires de sélection et suppression des échantillons dans la mémoire selon une loi uniforme.

3.8 Expérimentation et résultats

On expérimente l'AAE incrémental sur deux *datasets*. Tout d'abord, MNIST, le *dataset* principalement utilisé par Makhzani et al., 2015 dans leur introduction de l'AAE. On considère aussi STL-10, *benchmark* d'images naturelles conçu pour évaluer la semi-supervision. Pour les deux *datasets*, on fixe le pas $s = 2$ résultant en un total de 5 sessions.

Pour MNIST, f est un simple réseau convolutif à 4 couches de convolutions suivies d'un *fully-connected* (FC) produisant des *features* de dimensions 256. Deux cas sont ensuite possibles. Pour l'algorithme #1, on ajoute un FC pour projeter les features dans un espace latent 2D où sera imposé le prior mélange de gaussiennes. Pour le #2, on ajoute deux FC pour encoder séparément catégorie et style. Un premier avec $M = 50$ sorties pour générer les prédictions sur les clusters et un second pour produire les *features* de style en dimension 8.

Sur STL-10, nous évaluons uniquement le #2. On utilise un encodeur spécifique que l'on note ResNet-18-8. Le nombre de neurones, i.e. la largeur, de chaque couche du ResNet-18 standard (ResNet-18-16) a été divisée par 2. Cela réduit le nombre de paramètres de 11M à 2.8M. Le ResNet-18 ayant été conçu pour ImageNet (1000 classes, 1.3M d'images en $3 \times 224 \times 224$), cette réduction de paramètre est cohérente pour s'adapter à STL-10. Les dernières couches de l'encodeur suivent la même structure que pour MNIST. Un premier FC de dimension 256 connecté au FC de classification avec le nombre de clusters possibles $M = 120$ et un FC encodant le style en dimension 64.

Les décodeurs sont des réseaux convolutifs, avec des couches de convolution transposée pour l'opération de sur-échantillonnage servant à inverser le *pooling* de l'encodeur.

Tous les discriminateurs utilisés ici suivent le même modèle : un MLP avec deux couches cachées de 3000 neurones.

3.8.1 Algorithme #1 avec prior mélange de gaussiennes 2D

Comme mentionné en section 3.7.1, l'objectif de l'algorithme #1 est de valider le fonctionnement de l'AAE pour un incrémental de classes. En particulier, l'AAE est sensible aux problèmes de convergence inhérents aux méthodes adverses. L'objectif premier de l'expérimentation présentée dans cette section est donc d'observer si l'on arrive à reproduire

l'apprentissage d'un prior via entraînement adverse lorsque les classes sont introduites progressivement.

On reprend le *prior* utilisé par Makhzani et al., 2015 : un mélange de 10 gaussiennes 2D. Comme illustré en figure 3.6, ce type de *prior* est adapté pour MNIST en entraînement batch afin d'obtenir un espace latent structuré où chaque gaussienne représente une classe.

Benchmark MNIST incrémental standard

L'expérimentation sur MNIST consiste donc simplement à apprendre séquentiellement les 10 classes 2 par 2. On se place dans deux configurations différentes : le MNIST incrémental habituel avec 6000 images par classes (MNIST-full) ainsi qu'un cas avec moins de supervision en autorisant seulement 200 exemples par classes (MNIST-3%).

On attribue les composantes du mélange de gaussienne au fur et à mesure que les classes sont présentées en suivant le sens trigonométrique. La classification se fait via algorithme du plus proche voisin entre le vecteur encodé de l'image et les prototypes associés à chaque classe. Les prototypes sont simplement les moyennes de chaque composante du mélange de gaussiennes.

Résultats

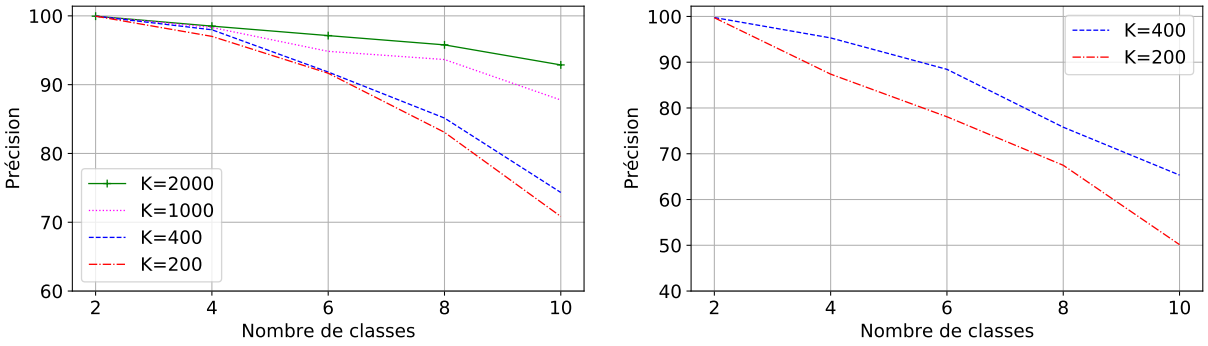

Figure 3.9 – Algorithme #1 sur MNIST incrémental avec différentes tailles de buffer K. Résultats pour MNIST-full à gauche et MNIST-3% à droite.

Pour MNIST-full (resp. MNIST-3%), on évalue notre méthode pour différents budgets mémoire $K \in \{200, 400, 1000, 2000\}$ (resp. $K \in \{200, 400\}$). La figure 3.9 affiche l'évolution de la précision incrémentale et la figure 3.10 permet de visualiser l'ensemble de test, avec un codage couleur par classe, dans l'espace latent 2D.

La précision en fonction de la taille de la mémoire suit le comportement attendu. Pour référence, un modèle de même avec le même encodeur convolutif et un classifieur discriminatif atteint 99.4% de précision sur les 10 classes de MNIST en hors ligne et 93.7% de précision finale avec pour seul outil contre l'oubli, de la répétition avec $K = 400$.

Cet algorithme #1 présente ici un niveau de précision insuffisant pour une tâche telle que MNIST. Un modèle discriminatif avec juste de la répétition le surpassant de presque 10% en précision finale, à taille de mémoire égale.

Mettons néanmoins de côté l'évaluation quantitative pour nous intéresser à l'évolution de l'espace latent visualisé en figure 3.10. Globalement, en considérant les quatre premières colonnes, le prior semble efficacement respecté dans les cas les plus favorables où l'ensemble des 6000 images par classes sont présentées au modèle.

L'impact de la mémoire pour maintenir la discrimination s'illustre notamment par la dernière ligne où pour les valeurs de K les plus faibles, les délimitations entre les clusters disparaissent augmentant la zone d'incertitude situé au centre de l'espace.

Le cas faiblement supervisé avec 200 images par classe est bien plus difficile à traiter pour le modèle. Le manque de données se constate dès les premières sessions où le prior est sous-appris. Dès la session 3, les clusters ne sont plus clairement séparés et le modèle n'arrive pas à apprendre la classe « 7 » introduite en session 4 et représentée par les points gris.

L'algorithme #1 n'est donc pas retenue comme candidat efficace pour l'apprentissage incrémental. Un prior sur espace latent de si faible dimension, combiné avec un positionnement arbitraire des classes entre elles, rend certainement la tâche de séparation trop complexe pour le modèle. On a par exemple les classes « 4 » et « 9 » qui sont connues pour être source de confusion dans MNIST. Avec notre implémentation, ces classes, visuellement proches, se retrouvent diamétralement opposées sur le cercle.

Cette expérimentation permet quand même de visualiser qualitativement l'impact de la mémoire pour maintenir la discrimination inter-classes et la dépendance aux données pour assurer une représentation intra-classe de qualité.

Elle permet aussi de valider la stabilité de l'AAE dans un contexte incrémental. Dans toutes les expérimentations, l'espace latent final reste cohérent avec le prior imposé.

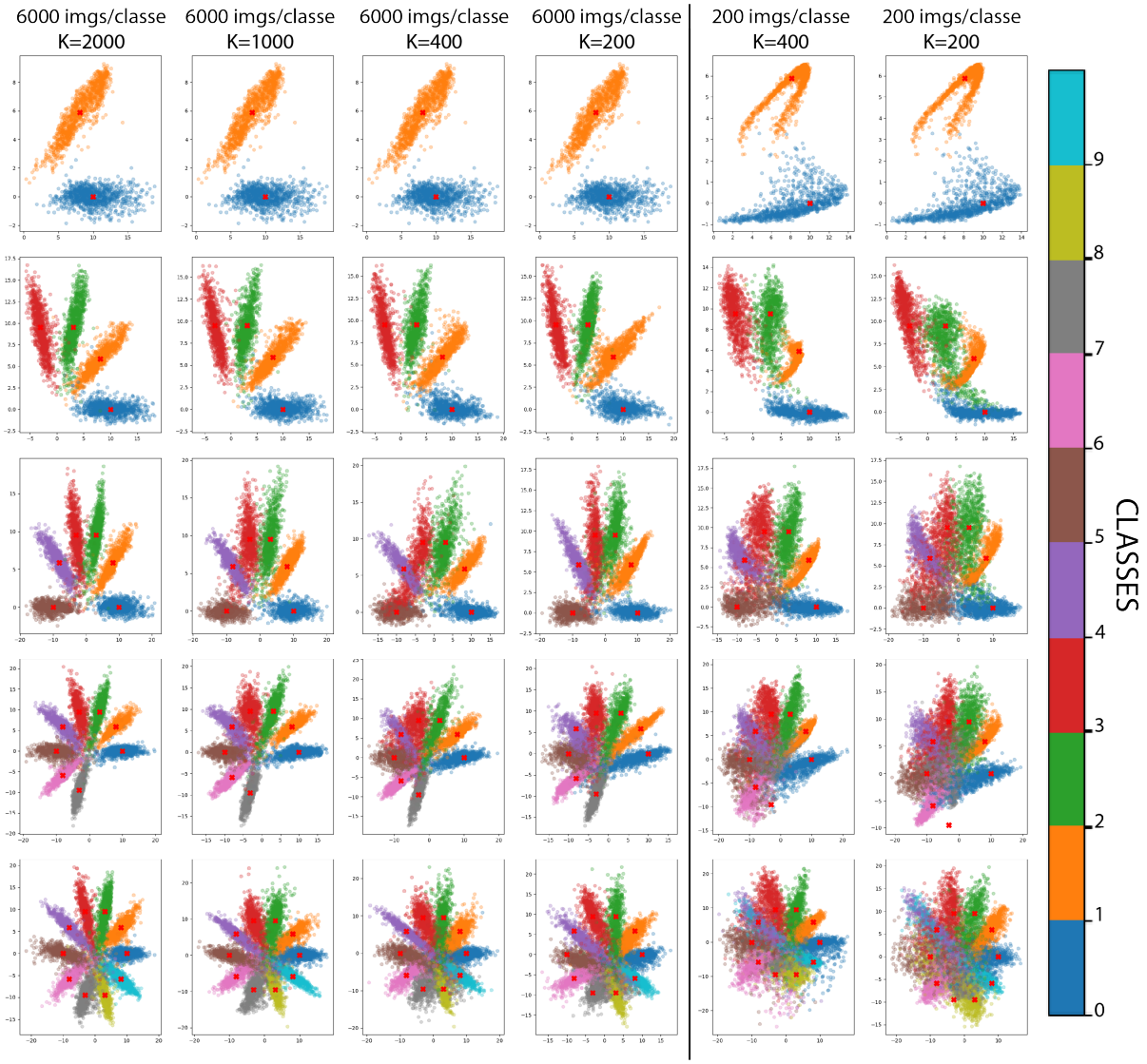

FIGURE 3.10 – Visualisation de l'espace de représentations obtenus au cours de l'incrémental de classes lors des différentes expérimentations avec l'algorithme #1 et MNIST.

3.8.2 Incrémental de classes semi-supervisé

Protocole

On s'intéresse ici à évaluer l'algorithme #2 dans un contexte où la supervision est très limitée. Cela signifie qu'un faible nombre de d'exemples annotés sera accessible à chaque session. L'AAE peut cependant compter sur une large source externe de données non-annotées.

Pour STL-10, le *benchmark* tel qu'il est proposé par COATES et al., 2011 contient de base peu d'exemples par classe dans son ensemble annoté. Ici, la différence est que les

classes sont introduites incrémentalement 2 par 2. Cela représente un ensemble de 1000 images annotées par session. En parallèle, on suppose que le modèle peut accéder à tout instant à la partie non-annotée de STL-10, i.e. les 100 000 images annexes.

Lorsqu'on évalue l'AAE sur MNIST, on restreint à 200 images par classe. Cela représente seulement 3.33% des 60 000 données d'entraînement habituellement accessibles. Les données non-annotées sont issues de la base E-MNIST. Donnant ainsi accès à jusqu'à 697 932 images supplémentaires. On profite qu'E-MNIST soit complètement annoté pour expérimenter les différents scénarios décrits en section 3.4.2 en contrôlant quelles classes seront présentes dans $\mathcal{U}$. On utilise notamment EMNIST-ByClass contenant les chiffres et les lettres (scénario classes incluses), EMNIST-Digits avec uniquement des chiffres (scénario classes communes) et EMNIST-Letters avec que des lettres (scénario classes disjointes). On fixe aussi $K = 400$.

Pour la répétition, le budget mémoire est $K = 500$ (resp. $K = 400$) pour STL-10 (resp. MNIST).

On compare notre approche à différentes méthodes de la littérature que l'on caractérise succinctement :

— LwF – Learning without Forgetting (Z. Li & Hoiem, 2017) : distillation de connaissance.

— iCaRL (Rebuffi et al., 2017) : répétition, distillation de connaissance et inférence avec plus proche centroïde.

— Weight Aligning (Y. Wu et al., 2019) : répétition, distillation de connaissance et post-traitement pour traiter le biais du classifieur FC.

Pour chaque méthode, on compare les résultats pour un même *backbone*. On utilise l'architecture définie pour l'encodeur de l'AAE. On évalue aussi la performance d'un oracle. L'oracle de la session i correspond à un modèle entraîné de manière conjointe sur la totalité des is classes. On évalue aussi deux approches élémentaires de référence. « *Fine-Tuning* », un apprentissage incrémental sans aucune solution contre l'oubli, ainsi que la « Répétition » qui autorise l'usage d'une mémoire épisodique.

Sur MNIST, seul l'AAE est limité à 2000 annotations. Les autres méthodes n'étant pas conçue pour fonctionner dans des régimes faiblement supervisés, sont maintenues dans un contexte d'incrémental de classe classique avec l'accès à l'intégralité des 60 000 images annotées. Notre méthode est cependant la seule capable d'exploiter des données non-annotées.

Résultats

Les résultats sont présentés sous forme de précisions finales et moyennes dans la table 3.1.

L'AAE surpasse toutes les méthodes mentionnées sur les deux *benchmarks*. Même sur MNIST, où notre modèle n'a reçu que 2 000 labels tout au long de sa durée de vie (environ 3% des labels vues par les autres méthodes), notre méthode obtient des performances globalement meilleures. La répétition de l'expérience avec différentes versions d'EMNIST comme flux de données non annotées $\mathcal{U}$ donne des résultats cohérents. Même lorsque $\mathcal{U}$ est constitué de classes complètement extérieures – ici on utilise des lettres manuscrites – le modèle atteint toujours une précision de pointe, ce qui montre que l'AAE tire profit des données supplémentaires, même si elles ne sont pas directement liées à la tâche principale. Cependant, le meilleur modèle est celui qui ne voit que des chiffres dans $\mathcal{U}$, ce qui indique que si le modèle est capable de spécialiser ses représentations, via l'auto-supervision, sur une distribution similaire à la tâche principale, l'apprentissage continu en bénéficie naturellement.

Sur STL-10, notre modèle obtient de bien meilleurs résultats que les méthodes précédentes. Cependant, il est important de noter que le jeu de données STL-10 est construit pour l'évaluation de l'apprentissage non-supervisé. Il est spécifiquement conçu pour être difficilement exploitable par les méthodes ne considérant que les quelques images annotées. Cela explique les faibles performances des méthodes précédentes (même pour l'oracle) alors que l'AAE exploite avec succès la grande quantité de données annexes disponibles dans STL-10.

Méthode	MNIST		STL-10	
	Finale (%)	Moyenne (%)	Finale (%)	Moyenne (%)
Oracle	99.4	99.7	67.2	73.5
Fine-Tuning	19.8	44.9	16.2	38.3
LwF (Z. Li & Hoiem, 2017)	71.3	85.2	17.9	42.5
Répétition	93.7	97.6	43.8	62.0
iCaRL (Rebuffi et al., 2017)	95.3	97.9	42.6	63.0
WA (Zhao et al., 2020)	96.0	98.3	47.3	63.5
AAE[a]	96.9	98.5	**57.3**	**72.0**
AAE[b] (EMNIST-Digits)	**98.1**	**99.0**		
AAE[b] (EMNIST-Letters)	95.9	98.5		

[a] Notre AAE sur MNIST utilise EMNIST-ByClass pour $\mathcal{U}$.
[b] Résultats additionnels sur MNIST avec EMNIST-Digits et EMNIST-Letters pour $\mathcal{U}$.

Table 3.1 – Comparaison de la précision finale et moyenne pour différentes méthodes d'incrémental de classes sur MNIST et STL-10 avec $s = 2$. L'oracle désigne la performance d'un modèle batch.

3.8.3 STL-10 avec pré-entraînement RotNet

Protocole

STL-10 est un *benchmark* particulièrement difficile en raison du peu d'images annotées accessibles. Cela signifie que trop peu d'exemples sont utilisables pour les méthodes purement supervisées, d'où la nécessité de pouvoir exploiter les données annexes du *dataset*.

Dans l'étude préliminaire (sec. 3.5.3), nous avions observé qu'un pré-entraînement aidait particulièrement dans le cas d'un DNN avec un nombre de paramètres trop important comparativement au nombre de données disponible. Ici, une session contient au plus 1500 images annotées ($|D_i| + K$). Cela explique la difficulté pour les méthodes uniquement supervisées d'apprendre le ResNet-18-8 (2.8M de paramètres). Ces méthodes ont d'ailleurs une performance catastrophique ou des difficultés à converger lorsqu'elles sont combinées avec le ResNet-18 standard de 11M de paramètres.

Ces hypothèses sont confirmées par la très faible performance des approches comparées dans la table 3.1. On a donc un *benchmark* où l'utilisation des données externes devient obligatoire. Afin de fournir un terrain de comparaison plus juste, on propose ici un pré-entraînement auto-supervisé via un RotNet sur les 100 000 images externes.

On ne retient pour cette évaluation que les méthodes avec répétition : Répétition seule, iCaRL, WA et AAE. L'encodeur de l'AAE lui aussi est initialisé via RotNet.

Résultats

L'impact d'un pré-entraînement auto-supervisé est clair : tous les modèles ont progressé et sont maintenant supérieurs à ceux de l'expérience précédente. WA est compétitif mais notre méthode continue de surpasser les autres. Cela conforte l'idée que si des données non-annotées sont accessibles, il est profitable de disposer d'un *framework* semi-supervisé capable de les exploiter tout au long du continuum d'apprentissage, comparé à une approche naïve limitée à un pré-entraînement.

Cette expérience vient aussi valider les résultats de l'étude préliminaire. STL-10 fait partie des *benchmark* où un pré-entraînement est vital pour les méthodes supervisées. Lorsque la quantité de données étiquetées est aussi limitée que dans STL-10, l'utilisation d'un modèle profond comme notre Resnet-18-8 (environ 2.8 millions de paramètres) soulève d'autres problèmes d'apprentissage spécifique aux cadres avec peu d'exemples. Ceci renforce notre intérêt pour l'apprentissage de représentation afin d'assister le schéma incrémental pour traiter ces cas faiblement supervisés ou compter sur de plus grandes architectures.

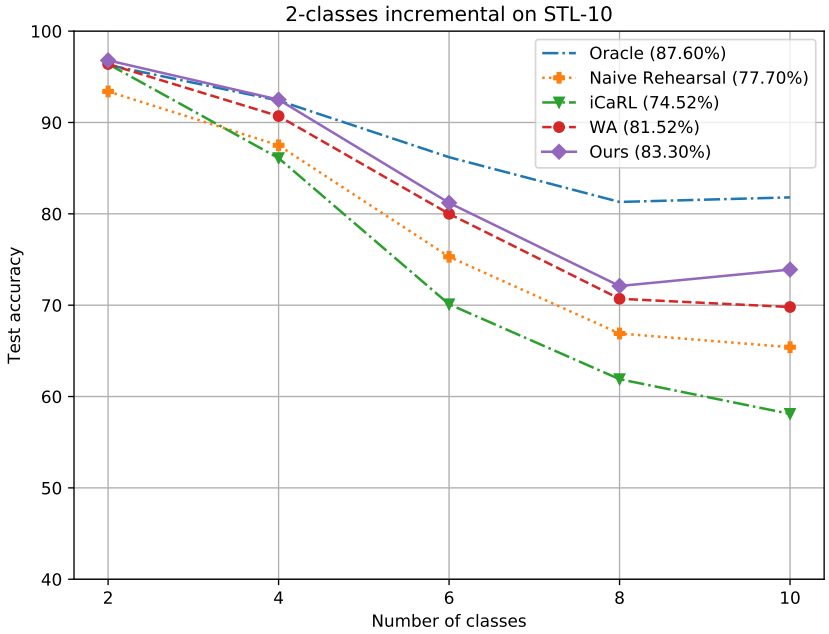

FIGURE 3.11 – Comparaison entre différentes méthodes avec répétition initialisé avec un encodeur auto-supervisé RotNet. On reporte la précision incrémentale moyenne dans la légende.

3.9 Conclusion

Au cours de ce chapitre, nous avons restitué notre démarche qui a mené à considérer l'apprentissage des représentations pour renforcer les modèles de l'apprentissage continu. Démarche qui a aboutit sur notre première contribution : un AAE capable d'apprendre sur un scénario d'incrémental de classe semi-supervisé.

Notre version de l'algorithme AAE tel qu'on le définit dans ce chapitre est capable d'apprendre des classes incrémentalement tout en exploitant des données non-annotées supposées disponibles. Les expérimentations permettent de valider que l'utilisation d'une tâche prétexte, même basique (e.g. la reconstruction), contribue à renforcer l'apprentissage de la tâche principale.

L'utilisation d'une source de données secondaire se présente comme une piste particulièrement intéressante pour traiter des problèmes continus avec des limitations drastiques sur la quantité de données annotées disponibles.

3.10 Publication

Lechat, A., Herbin, S. & Jurie, F. (2020). Semi-Supervised Class Incremental Learning. *25th International Conference on Pattern Recognition (ICPR), 2020*, 10383-10389, (Poster)

Chapitre 4

Pseudo-Labeling pour l'Incrémental de Classes

Contents

4.1 Introduction

Au cours de ce chapitre, nous poursuivons l'étude du scénario d'incrémental de classes avec une source secondaire de données non-annotées.

Nous introduisons l'algorithme « *Pseudo-Labeling for Class incremental Learning* » (PLCiL). Nous proposons, avec le PLCiL, une solution simple et optimisée pour traiter des données secondaires sans labels pendant un processus d'apprentissage incrémental de classes. Le coeur de la méthode implémente un mécanisme de régularisation de consistance, technique qui à fait ses preuves en apprentissage semi-supervisé des représentations.

Ce chapitre contient les sections suivantes :

— Une discussion autour des limitations de notre approche AAE qui, pour nous, sont les motivations à proposer une solution alternative plus généralisable.
— Une rapide introduction a la régularisation de consistance via *pseudo-labeling*.
— La présentation complète du PLCiL : les composantes clés de l'algorithme ainsi que son algorithme d'apprentissage.
— Une étude expérimentale poussée de l'incrémental semi-supervisé. Le PLCiL permet notamment d'aborder des *datasets* plus courants tels que CIFAR-100 et ImageNet.
— Une section dédiée à l'ablation de la méthode du PLCiL.

4.2 Motivations

La qualité des représentations est cruciale pour la performance du classifieur qui suit, mais difficilement atteignable dans un contexte purement continu. Les résultats du chapitre précédent sont encourageants pour poursuivre vers l'idée d'un incrémental de classe semi-supervisé. L'AAE permet de régulariser l'apprentissage incrémental en direct procurant ainsi de meilleures représentations. Il se montre particulièrement efficace dans une situation où la supervision est particulièrement restreinte.

Cependant, au fil de nos expérimentations avec l'AAE, nous avons mis en évidences deux faiblesses inhérentes à la méthode :

Entraînement adverse : La complexité du paradigme d'entraînement de l'AAE rend difficile le passage à l'échelle sur des *datasets* avec plus de classes où une plus haute résolution. On retrouve les problématiques des modèles adverses comme l'extinction de gradient mais surtout, l'équilibre entre les 4 *loss* utilisées pendant les 3 étapes d'apprentissage est difficile à trouver.

Représentations inadaptées pour la classification : Les représentations apprises pour la reconstruction/génération se transfèrent mal à la classification. Améliorer la qualité de la reconstruction en accordant plus de poids à la *loss* associée se fait au détriment de la performance de classification. Inversement, favoriser la capacité discriminatoire vient impacter négativement la qualité des images générées. Cela rejoint les tendances de la littérature. Les méthodes contrastives et celles à base de *pseudo-labeling* permettent de produire des *features* qui se transfèrent bien mieux à une tâche de classification, égalant presque l'approche batch supervisée. Même les *features* du RotNet surclassent celles apprises de manières adverses avec l'approche générative BiGAN (Donahue et al., 2017) pour la reconnaissance d'objets.

Au cours de ce chapitre, nous proposons donc une solution mieux adaptée pour l'incrémental de classes semi-supervisé.

4.3 *Pseudo-labeling* et Régularisation de consistance

4.3.1 Le choix de la tâche prétexte

Le coeur de la méthode repose donc sur la façon dont sont traitées les données non-annotées. On met de côté les approches génératives pour se concentrer sur celles discriminatives. En auto-supervision pure, deux approches rivalisent au sommet de l'état-de-l'art : le *contrastive learning* et le *pseudo-labeling*.

Alors que les dernières solutions contrastives concurrencent les approches supervisées (Caron et al., 2020 ; T. Chen, Kornblith, Norouzi et al., 2020), ces solutions sont très dépendantes de leur gestions des paires négatives. Elles requièrent généralement une grande quantité de ces paires négatives, induisant ainsi des coûts calculatoires conséquents.

Les approches reposant sur le *pseudo-labeling* et les méthodes professeur-étudiant, sont généralement plus légères que le contrastive. Elles reposent sur un mécanisme de génération automatique de labels artificiels. Gidaris et al., 2021 montrent avec OBOW que cette famille de méthodes peut surpasser le *contrastive learning* avec des ressources en calculs bien inférieures.

Dans un contexte semi-supervisé, la régularisation de consistance est plus particulièrement adaptée. Grâce à la supervision, on dispose d'un modèle capable d'émettre une prédiction dans l'espace des classes. Pour une donnée non-annotée, on peut utiliser cette prédiction en tant que label artificiel.

On s'intéresse donc à la régularisation de consistance pour l'incrémental de classes. Fondamentalement, la distillation de connaissance est une forme de régularisation de consistance qui à déjà fait ses preuves pour aider à maintenir la discrimination entre les anciennes classes. On propose ici d'aller plus loin et de l'étendre au contexte du continu

semi-supervisé avec l'objectif d'améliorer les représentations tout en les régularisant pour contrer l'oubli.

4.3.2 Pseudo-Labeling

On propose dans ce chapitre de s'intéresser à la régularisation de consistance par *Pseudo-Labeling* pour un incrémental de classe semi-supervisé. Cette section fournit un bref rappel de l'approche déjà présentée en état-de-l'art (sec. 2.5.4).

Le concept du pseudo-labeling en semi-supervisé est simple. Soit Θ un modèle construit pour la classification de C classes. Soit x une image. Si cette image possède un label, on utilise ce dernier directement au sein d'une *loss* d'entropie croisée. Dans le cas d'une image non-annotée, l'idée est de passer par un label artificiellement généré pour se ramener à la *loss* supervisée de classification.

Pour la génération des labels artificiels, Berthelot et al., 2020 ; Berthelot et al., 2019 ; Gidaris et al., 2021 ; Sohn et al., 2020 prennent le parti d'une solution professeur-étudiant. La sortie du modèle professeur devient la VT pour entraîner le modèle élève. La particularité de ce système professeur-étudiant est que, contrairement à la distillation de connaissance, le professeur et l'étudiant sont en fait le même modèle. Cela implique de prendre en compte deux facteurs :

— Le professeur est mis-à-jour en permanence. Les labels artificiels doivent donc être générés à la volée à chaque pas de gradient.
— On ne peut pas montrer exactement la même image x au professeur et à l'étudiant puisqu'ils produisent exactement la même réponse. Il est donc nécessaire de passer par des transformations t et t' afin de produire deux versions de l'image $t(x)$ et $t'(x)$. L'une sert à produire le label artificiel et l'autre à produire une prédiction pour l'apprentissage. Ces transformations doivent préserver la sémantique de l'image pour que les prédictions conservent leur sens.

En pratique, les labels artificiels peuvent être un vecteur de logits ou de probabilités dans l'espace de sortie du DNN, comme pour la distillation de connaissance. Dans le cas du « pseudo-labeling » (Sohn et al., 2020), les labels artificiels sont un indice de classe, ou un vecteur *one-hot*, ce qui revient à attribuer une catégorie parmi les C possibles à l'image non-annotée.

4.4 *Pseudo-Labeling for Class-incremental Learning* (PLCiL)

On décrit ici notre approche PLCiL, une solution d'incrémental de classes semi-supervisée exploitant les données non-annotées via *pseudo-labeling*.

4.4.1 Composantes de la *baseline* PLCiL

On propose de construire le PLCiL sur la *baseline* d'apprentissage avec répétition classique. On dispose donc, lors de la session i, des composantes suivantes :

- Un DNN $\Theta_i = (\Phi_i, W_i)$. Φ_i a le rôle d'encodeur. W_i est une tête de classification unique qui est incrémentale, on lui ajoute des sorties en fonction du nombre total de classes à traiter C_i.
- L'ancien DNN $\Theta_{i-1} = (\Phi_{i-1}, W_{i-1})$ expert sur les C_{i-1} classes précédentes.
- Une mémoire épisodique $\mathcal{B}_{i-1}$ avec un budget de K exemples. Cette mémoire contient, en principe, un ensemble i.i.d. des anciennes classes.
- Deux processus distincts d'augmentation de données. Les augmentations faibles et les augmentations fortes. Pour une image x, on note $\alpha(x)$ (resp. $\mathcal{A}(x)$) sa version faiblement (resp. fortement) augmentée.
- Le *dataset* D_i contenant les données annotées associées aux nouvelles classes.
- Le *dataset* U_i contenant les données non-annotées échantillonnées dans $\mathcal{U}$.

4.4.2 Augmentation de données

L'un des points clés du *pseudo-labeling* est le choix des familles de transformations appliquées pour augmenter les images de $\mathcal{U}$.

De nombreux travaux ont essayé différentes combinaisons de transformations pour la régularisation de consistance. Une approche particulière, retenue par (Berthelot et al., 2020 ; Sohn et al., 2020 ; Q. Xie et al., 2020), montre les meilleurs résultats : l'utilisation d'augmentations faibles pour générer les labels artificiels et d'augmentations fortes pour les prédictions.

Les faibles augmentations, pour des images naturelles RVB, suivent la stratégie classique du « *flip-and-shift* ». Une image subit une symétrie horizontale avec une probabilité de 50% ainsi que des translations aléatoires sur les axes horizontal et vertical. Ces transformations sont appliquées avec une magnitude sélectionnée aléatoirement dans un intervalle autorisé. On dénote $\alpha(.)$ l'application de ses transformations.

Pour les augmentations fortes, deux algorithmes dérivés d'AutoAugment (Cubuk et al., 2019) sont particulièrement populaires pour l'augmentation d'images naturelles pour

l'apprentissage de DNN : RandAugment (Cubuk et al., 2020) et CTAugment (Berthelot et al., 2020).

Ces méthodes suivent toutes les deux la même stratégie. Parmi une liste de transformations possibles (cf. table 4.1), deux sont sélectionnées au hasard. Puis la magnitude de chacune est échantillonnée sur un intervalle donné. Ces transformations sont ensuite appliquées consécutivement à l'image. On note cette application $\mathcal{A}$.

Alors que RandAugment échantillonne uniformément les magnitudes sur tout l'intervalle, CTAugment le sépare en sous-intervalles auxquels on attribue chacun un poids. Ces poids sont mis-à-jour pendant la phase d'apprentissage en suivant une fonction de vraisemblance qui favorise les sous-intervalles produisant une image correctement classifiée. L'échantillonnage des magnitudes est désormais pondéré par ces poids. Des exemples d'images transformées selon CTAugment sont fournis en figure 4.1.

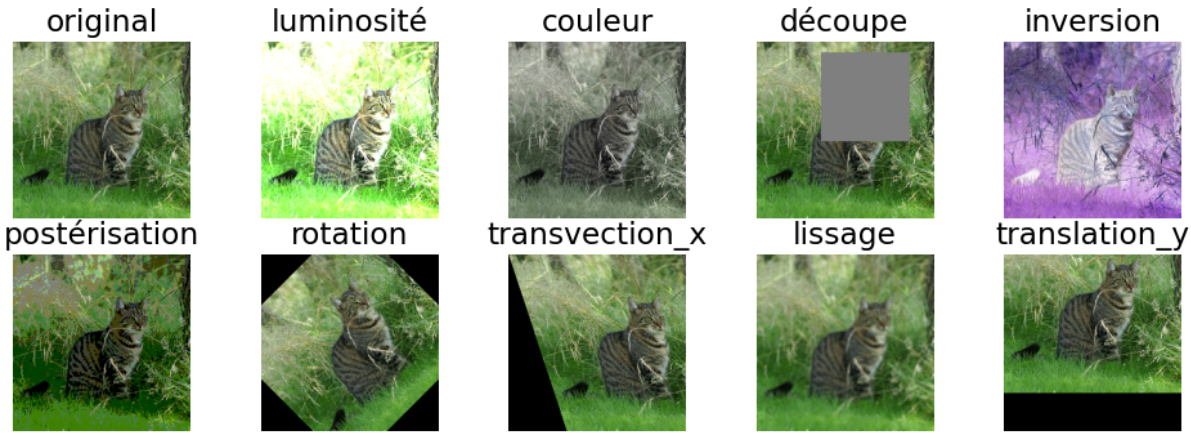

Figure 4.1 – Exemple de transformations possibles dans l'algorithme CTAugment.

4.4.3 Apprentissage du modèle

Maintenant que toutes les composantes clés du PLCiL ont été exposées, nous définissons ici sa procédure d'entraînement.

Les paramètres du réseau Θ_i sont appris par descente de gradient stochastique (SGD) à chaque session i. Comme c'est le cas généralement lorsqu'on applique la SGD, chaque étape de mise à jour des paramètres utilise un mini-batch de données qui est échantillonné de manière aléatoire. Dans le schéma auto-supervisé proposé dans notre approche, un tel mini-batch S est constitué de deux types de données : un sous-ensemble S_l contenant B images étiquetées échantillonnées à partir de $D_i \cup \mathcal{B}_{i-1}$ et un autre sous-ensemble S_u composé de μB images provenant de $\mathcal{U}$ où μ est un hyperparamètre scalaire.

Transformation	Description	Paramètre	Intervalle
Auto-contraste	Maximise le contraste en mettant le pixel le plus foncé (resp. clair) en noir (resp. blanc). Puis mélange avec l'image originale avec un ratio λ	λ	$[0, 1]$
Luminosité	Ajuste la luminosité. $B = 0$ donne une image noir et $B = 1$ l'image originale.	B	$[0, 1]$
Couleur	Ajuste la balance des couleurs. $C = 0$ donne une image en noir et blanc et $C = 1$ l'image originale	C	$[0, 1]$
Contraste	Contrôle le contraste de l'image. $C = 0$ donne une image grise et $C = 1$ l'image originale.	C	$[0, 1]$
Découpe	Passe un patch ($L \times$ Largeur aléatoire de pixels au gris	L	$[0, 0.5]$
Égalisation	Égalise l'histogramme. Puis mélange avec l'image originale avec un ratio λ	λ	$[0, 1]$
Inversion	Inverse les pixels de l'image. Puis mélange avec l'image originale avec un ratio λ	λ	$[0, 1]$
Identité	Retourne l'image originale.		
Postérisation	Réduit chaque pixel à B bits.	B	$[1, 8]$
Redimensionnement	Rogne un patch de taille $L \times$ Largeur au centre de l'image puis redimensionne via la méthode M	L M	$[0.5, 1]$ Cf. légende
Rotation	Tourne l'image de θ degrés.	θ	$[-45, 45]$
Netteté	Ajuste la netteté de l'image. $S = 0$ donne une image floue et $S = 1$ l'image originale.	S	$[0.05, 0.95]$
Transvection_x	Transvection de l'image le long de l'axe horizontal avec un taux R.	R	$[-0.3, 0.3]$
Transvection_y	Transvection de l'image le long de l'axe vertical avec un taux R.	R	$[-0.3, 0.3]$
Lissage	Ajuste le lissage de l'image. $S = 0$ donne une image lissée au maximum et $S = 1$ l'image originale	S	$[0, 1]$
Solarisation	Inversion des pixels au-dessus du seuil T.	T	$[0, 1]$
Translation_x	Translation horizontale de ($\lambda \times$ Largeur).	λ	$[-0.3, 0.3]$
Translation_y	Translation verticale de ($\lambda \times$ Hauteur).	λ	$[-0.3, 0.3]$

TABLE 4.1 – Collection des transformations possibles dans CTAugment (BERTHELOT et al., 2020). Tableau extrait et traduit de l'article original. Les méthodes M de redimensionnement possibles sont : anti-alias, bicubique, bilinéaire, box, hamming et nearest.

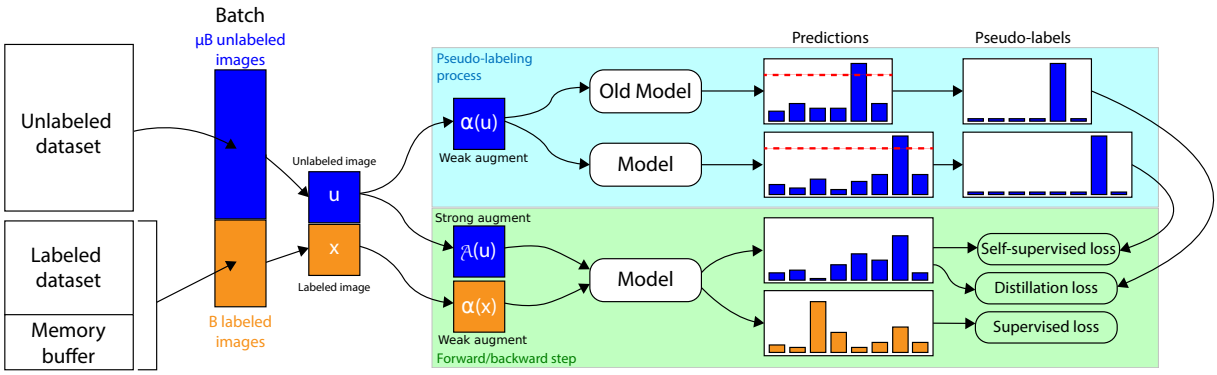

FIGURE 4.2 – Vue d'ensemble de la procédure d'entraînement incrémental du PLCiL. La partie supérieure (fond bleu) représente le processus de *pseudo-labeling* consacré à la génération automatique de labels artificiels pour une image non étiquetée u. La partie inférieure (fond vert) est un processus d'apprentissage forward/backward supervisé standard, incluant une fonction de coût à 3 termes : i) entropie croisée supervisée standard, ii) régularisation entre les sessions par distillation, iii) régularisation par auto-supervision.

L'algorithme PLCiL proposé repose sur l'optimisation de 3 fonctions de coûts combinées visant 3 objectifs différents : i) la supervision provenant des nouvelles données annotées, ii) la régularisation de la consistance en utilisant des pseudo-labels générées automatiquement sur les données non étiquetées, iii) la distillation de la connaissance auto-supervisée ajoutant une régularisation supplémentaire. Nous décrivons ces 3 *loss* dans ce qui suit. Pour alléger les notations, on omet l'indice de session i lorsque sa référence n'est pas nécessaire. Le processus global de formation est illustré dans la figure 4.2.

***Loss* supervisée l_{sup}**

Son rôle est d'utiliser des données étiquetées pour apprendre le modèle. Pour améliorer la robustesse, une étape d'augmentation des données est introduite en utilisant une transformation faible α. Le signal supervisé est rétro-propagé au réseau en utilisant la *loss* standard d'entropie croisée entre la prédiction du DNN et les véritables étiquettes. Elle peut être exprimée comme suit :

$$l_{\text{sup}}(\Theta, S_l) = \frac{1}{B} \sum_{(x,y)\in S_l} H(y, h(\alpha(x); \Theta)) \tag{4.1}$$

où $h(x;\Theta) \in \mathbb{R}^C$ est la distribution prédite des classes pour une entrée x, C étant le nombre de classes considérées dans la session courante. H est l'entropie croisée définie

par :

$$H(y, h(\alpha(x); \Theta)) = -\sum_{j=1}^{C} y_j \log(h_j(\alpha(x); \Theta)) \tag{4.2}$$

où les deux arguments sont des distributions de probabilités. Les entropie croisée sont utilisées ici avec des labels de classes. Autrement dit, y est un vecteur *one-hot* de la classe représentée par l'image. On simplifie donc la formule en assimilant y à un indice de classe et on obtient :

$$H(y, h(\alpha(x); \Theta)) = -\log(h_y(\alpha(x); \Theta)) \tag{4.3}$$

Loss **auto-supervisée** l_{self}

Son rôle est de régulariser les représentations d'images en imitant le processus supervisé mais sur des données non étiquetées u de S_u. On procède à l'aide de pseudo-labels sur et de deux niveaux de transformations d'images : faible et forte. Une donnée faiblement transformée $\alpha(u)$ est d'abord présentée au DNN et émet une prédiction. Si le modèle est suffisamment confiant sur sa sortie (selon un seuil τ sur les scores), cette prédiction est utilisée comme pseudo-label pour une *loss* d'entropie croisée sur l'image fortement augmentée $\mathcal{A}(u)$. En notant la prédiction sur les données faiblement augmentées $q_u = h(\alpha(u), \Theta)$ et le pseudo-label $\hat{q}_u = \text{argmax}_y(q_u)$, la *loss* auto-supervisée résultante est :

$$l_{\text{self}}(\Theta, S_u) = \frac{1}{\mu B} \sum_{u \in S_u} \mathbb{1}_{\max(q_u) > \tau} H(\hat{q}_u, h(\mathcal{A}(u); \Theta)) \tag{4.4}$$

$\hat{q}_u$ étant un un pseudo-label, il est possible d'écrire :

$$l_{\text{self}}(\Theta, S_u) = -\frac{1}{\mu B} \sum_{u \in S_u} \mathbb{1}_{\max(q_u) > \tau} \log(h_{\hat{q}_u}(\mathcal{A}(u); \Theta)) \tag{4.5}$$

Loss **de distillation de connaissances** l_{kd}

La distillation est utilisée pour assurer la cohérence des prédictions entre les sessions dans l'idée de réduire l'oubli. Nous utilisons à nouveau le *pseudo-labeling* avec un seuil de confiance, mais à la différence que les pseudo-étiquettes sont générées en utilisant le modèle de la session précédente Θ_{i-1}. Soit $q_{\text{old}} = h(\alpha(u), \Theta_{i-1})$ et $\hat{q}_{\text{old}} = \text{argmax}(q_{\text{old}})$ respectivement la prédiction et le pseudo-label issus de la session précédente. La *loss* de distillation des connaissances (KD) est définie de la manière suivante :

$$l_{\text{kd}}(\Theta_i, \Theta_{i-1}, S_u) = \frac{1}{\mu B} \sum_{u \in S_u} \mathbb{1}_{\max(q_{\text{old}}) > \tau} H(\hat{q}_{\text{old}}, h(\mathcal{A}(u); \Theta_i)) \tag{4.6}$$

ou encore :

$$l_{\text{kd}}(\Theta_i, \Theta_{i-1}, S_u) = -\frac{1}{\mu B} \sum_{u \in S_u} \mathbb{1}_{\max(q_{\text{old}})>\tau} \log(h_{\hat{q}_{\text{old}}}(\mathcal{A}(u); \Theta_i)) \tag{4.7}$$

Notez que dans notre approche, l_{kd} est calculé uniquement sur des échantillons non étiquetés.

Combinaison des 3 termes dans une *Loss* unique

Le PLCiL combine ces 3 termes en une seule *loss* afin d'optimiser simultanément les trois objectifs à chaque pas de gradient :

$$l = l_{\text{sup}} + \lambda(l_{\text{self}} + \eta l_{\text{kd}}) \tag{4.8}$$

avec $\eta = \frac{C_{i-1}}{C_i}$ le rapport entre le nombre de classes apprises par l'ancien modèle et le nombre actuel de classes. Cette pondération est utilisée par Z. Li et Hoiem, 2017 ; Y. Wu et al., 2019 ; Zhao et al., 2020 pour équilibrer la *loss* de distillation. Notez qu'avec l'hypothèse de s constant, $\eta = \frac{i-1}{i}$. λ est un hyperparamètre libre chargé d'équilibrer les contributions de la supervision et de l'auto-supervision.

4.4.4 Explication des composantes de la *loss*

Au travers de cette section, nous proposons de discuter du choix et des comportements attendus pour chaque composante de la *loss*, la façon dont elles interagissent et le rôle des hyperparamètres.

Équilibre entre supervision et semi-supervision

Lors de l'implémentation de notre algorithme, nous avons cherché à implémenter 3 idées clés : i) utiliser des données additionnelles sans étiquettes pour améliorer et stabiliser les représentations en apportant de la diversité visuelle ; ii) utiliser des données non-annotées pour ajouter une forme d'auto-régularisation sur la tête de classification FC ; iii) utiliser des pseudo-labels, générés par le modèle de la session antérieure, pour distiller la connaissance entre les étapes incrémentales.

La fondation du PLCiL est une approche d'incrémental de classes avec répétition. On lui ajoute un mécanisme de pseudo-labeling avec seuillage afin d'exploiter les données annexes.

La combinaison des deux premières *loss*, $l_{\text{sup}} + \lambda l_{\text{self}}$ constitue donc l'algorithme de pseudo-labeling semi-supervisé tel qu'il est utilisé en entraînement batch. La principale

complexité d'apprentissage à ce stade est l'équilibre entre ces deux termes. Plusieurs hyperparamètres choisis manuellement sont impliqués :

- μ : le ratio entre les données annotées et non-annotées. Pour chaque mini-batch annoté de taille B, le modèle reçoit aussi μB images non-étiquetées. Sohn et al., 2020 ; Q. Xie et al., 2020 montrent un gain significatif sur leurs méthodes pour les plus grandes valeurs de μ.
- λ : permet de contrôler manuellement l'importance de l'auto-supervision par rapport à la supervision. λ est choisi élevé si l'on à confiance en l'auto-supervision (e.g. *benchmarks* hors-ligne de semi-supervisé avec une distribution des classes identiques entre les *datasets* annotés et non-annotés).
- τ : le seuil à franchir pour qu'un pseudo-label soit retenu. Cela permet de filtrer les données de $\mathcal{U}$ pour n'exploiter que celles où le modèle est capable d'attribuer une catégorie avec un certain taux de confiance.

La majorité des algorithmes de semi-supervision dépendent grandement du choix de λ. Berthelot et al., 2020 ; Berthelot et al., 2019 ; Laine et Aila, 2017 ; Oliver et al., 2018 ; Tarvainen et Valpola, 2017 proposent d'augmenter λ au cours de l'entraînement. L'idée sous-jacente est que le modèle, au début de l'apprentissage, est peu fiable pour proposer des annotations. Il est donc nécessaire d'attendre que le modèle devienne pertinent, grâce à la supervision, avant de lui faire confiance pour l'auto-supervision.

Avec des pseudo-label contrôlés par un seuil, Sohn et al., 2020 montrent que le choix de λ est moins important et n'a pas besoin d'être incrémenté. La régularisation du poids de l_{self} est intrinsèque en raison du seuillage. En reprenant l'équation 4.5, on constate que l_{self} est une moyenne sur le mini-batch non-annoté complet et non une moyenne sur les exemples sélectionnés par la fonction indicatrice. Lors des premières itérations de l'entraînement, le modèle est peu confiant sur la tâche principale de classification. La condition $\max(q_u) > \tau$ sera typiquement non remplie. l_{self}, dont l'amplitude est directement proportionnelle aux nombre d'exemples retenus, aura une valeur très faible et donc un poids marginal dans l'optimisation. Plus tard dans l'apprentissage, le modèle devient confiant dans ses prédictions. Il est capable d'étiqueter plus de données avec $\max(q_u) > \tau$, augmentant ainsi l'amplitude de l_{self} et donc sa contribution dans l'entraînement. Ce mécanisme de sélection introduit un « curriculum » inné équilibrant automatiquement l'importance à accorder à l'auto-supervision.

Distillation de connaissance avec *pseudo-labeling*

La spécificité du semi-supervisé incrémental avec régularisation de consistance est que, l'espace des sorties étant le même pour la tâche cible et la tâche prétexte, l'espace des pseudo-labels est lui aussi incrémental. Nous proposons donc d'utiliser la distillation de

connaissance pour régulariser l'incrément. Au lieu de l'utiliser sous sa forme classique (Hinton et al., 2015), nous proposons de reprendre l'approche *pseudo-labeling*. L'objectif d'un tel choix est de conserver une cohérence entre l_{kd} et l_{self} afin que la distillation régularise aussi l'auto-supervision.

L'une des contributions principales du PLCiL est donc l_{kd} qui propose une distillation de connaissance sur les données non-annotées. La distillation est une forme de régularisation de consistance entre les distributions en sortie de deux modèles, le nouveau et l'ancien. L'approche standard pour la distillation repose sur un *soft sharpening* (Z. Li & Hoiem, 2017) qui vient lisser les sorties. Notre approche est donc spécifique sur deux points : l'utilisation d'un seuil τ ainsi que l'augmentation de données.

L'utilisation d'un seuil de confiance permet de sélectionner la nature de la connaissance distillée en ne retenant que les exemples pertinents. La distillation classique (Lee et al., 2019 ; Y. Wu et al., 2019 ; J. Zhang et al., 2020 ; Zhao et al., 2020) prend le partie de transférer à l'aveugle la connaissance complète contenue dans chaque exemple. Cela implique la distillation d'informations utiles mais aussi de bruit, d'où la nécessité d'utiliser un paramètre de température lissant pour limiter les transferts.

Notre distillation, basée sur un `argmax` l_{kd} exprimée dans l'équation 4.6, tire profit à la fois du mini-batch non étiqueté S_u et de l'ancien modèle Θ_{i-1}. En complément de la régularisation auto-supervisée (l_{self}), l_{kd} impose la consistance entre les sessions (i.e. entre Θ_{i-1} et Θ_i).

C'est-à-dire que, pour une image sans étiquette u, on propose d'utiliser deux expertises pour proposer un pseudo-label : le modèle lui-même Θ_i et son antécédent Θ_{i-1}. L'accès à Θ_{i-1} est inhérent au paradigme incrémental. Le rôle de ce modèle est donc de proposer des annotations qui vont contribuer à répéter les classes passées, créant ainsi le lien entre les sessions.

Le choix d'utiliser l'augmentation de données pour la distillation est purement arbitraire pour des raisons pratiques. Il permet de rester cohérent et de s'appuyer sur l'implémentation de l_{self} sans rajouter de complexité calculatoire. En effet, pour une image u, on calcule $\alpha(u)$ et $\mathcal{A}(u)$ puis on prédit un label grâce à $h(\alpha(u);\Theta_i)$ puis la prédiction $h(\mathcal{A}(u);\Theta_i)$. Seul le calcul de la prédiction $h(\mathcal{A}(u);\Theta_i)$ est coûteux car il requiert la sauvegarde locale des calculs intermédiaires pour préparer la rétro-propagation via les algorithmes d'auto-différentiation. Le *pseudo-labeling* en lui-même n'est qu'une simple inférence. Avec notre implémentation, l_{kd} utilise aussi la prédiction $h(\mathcal{A}(u);\Theta_i)$. On ne rajoute donc pas d'étape dans l'auto-différentiation. On se contente simplement d'inférer un pseudo-label supplémentaire $h(\alpha(u);\Theta_{i-1})$. Si l'on se réfère à la figure 4.2, les opérations de la section bleue se font sans gradient et seule la section verte concerne le *forward/backward* d'optimisation.

Par ailleurs, on constate lors de nos expérimentations que notre version de l_{kd} permet de maintenir une régularisation lors du passage d'une session à une autre. En effet, lorsqu'on initialise Θ_i, on récupère les poids de Θ_{i-1} auxquels on ajoute les nouvelles sorties possibles. Lorsqu'on applique la fonction `softmax`, l'augmentation brutale de la dimension du vecteur de logits vient écraser la distribution prédite. Pour une image u, on a donc $\max(q_u) \leq \max(q_{\text{old}})$. De ce fait, le nouveau modèle Θ_i accuse une baisse de confiance sur ses prédictions. Expérimentalement, on observe que juste après l'ajout des nouvelles sorties, les prédictions ne franchissent plus le seuil provoquant une diminution du nombre de pseudo-labels générés pour l'auto-supervision. Ce sera donc le rôle de Θ_{i-1} de faire perdurer le *pseudo-labeling* issue de la session précédente jusqu'à ce que Θ_i reprenne sa contribution pour l_{self}.

Le rôle du paramètre η

Ce paramètre $\eta = \frac{C_{i-1}}{C_i}$ est récurrent dans les approches de distillation depuis son introduction par Z. Li et Hoiem, 2017. Dans le cas supervisé usuel, il est utilisé pour équilibrer supervision et distillation : $l = (1-\eta)l_{\text{sup}} + \eta l_{\text{kd}}$.

L'idée est que pour les premières sessions ($\eta \approx 0.5$), le modèle connaît peu de classes. Il accorde donc autant d'importance à l'apprentissage des nouvelles classes que la régularisation des anciennes. Cependant, pendant les sessions tardives, l'ajout de quelques nouvelles classes devient marginal comparé à toutes celles qui sont déjà connues ($\eta \approx 1$). Il est donc dans l'intérêt de la précision globale de favoriser la rétention de connaissance sur la majorité et donc de donner priorité à la distillation, quitte à ne pas apprendre les nouvelles classes. Nous nous inspirons de cette solution pour maîtriser l'impact de l_{kd} au sein de la régularisation $l_{\text{self}} + \eta l_{\text{kd}}$.

Déjà, notre formulation de la *loss* en équation 4.8 implique que nous ne pénalisons jamais l'apprentissage des nouvelles classes. Le seul terme influencé par η est l_{kd}. Comme mentionné précédemment, on effectue une régularisation de consistance professeur-étudiant avec deux professeurs. On estime que, pour une image non annoté, plus le professeur connaît de classes, plus il aura de chance d'attribuer un label correct. η retranscrit ce taux de confiance. Lors des sessions les plus avancées où Θ_{i-1} et Θ_i connaissent un nombre similaire de classes ($\eta \approx 1$), on accorde autant de crédit à leurs expertises.

4.5 Expérimentations avec le PLCiL

Nous avons validé expérimentalement notre méthode sur les 2 jeux de données couramment utilisés pour évaluer les méthodes d'incrémental de classe, à savoir CIFAR-100 et ImageNet-100. ImageNet-100 est un sous-ensemble d'ImageNet-1000 où seules les 100

premières classes sont considérées (les mêmes classes que Castro et al., 2018 ; Rebuffi et al., 2017).

L'évaluation est faite en utilisant la métrique standard de précision incrémentale définie en section 2.3.2. Pour les expériences ImageNet, nous rapportons la précision top-5, c'est-à-dire qu'une prédiction est considérée correcte si la classe attendue est parmi les 5 classes ayant la probabilité prédite la plus forte. Les résultats sont la moyenne pour 3 exécutions pour CIFAR-100 et une pour ImageNet, comme cela est courant dans la littérature (Y. Wu et al., 2019 ; Zhao et al., 2020).

Comme mentionné dans nos expérimentations avec l'AAE, l'exploitation de données non étiquetées supplémentaires – qui est l'objectif principal de l'approche proposée – n'est pas directement réalisable avec les méthodes existantes puisqu'elles sont conçues pour fonctionner sous supervision complète. Seul DMC+ J. Zhang et al., 2020 est conçu pour fonctionner dans le même scénario. Ainsi, les résultats, plus qu'une comparaison brute directe entre les méthodes, doivent être considérés comme une démonstration des avantages de la semi-supervision lors de l'entraînement d'un modèle d'incrémental de classes. Par ailleurs, l'AAE ne convergeant pas pour les *benchmarks* à cette échelle, il n'est pas évalué ici.

Nous comparons néanmoins notre méthode aux solutions suivantes (entièrement supervisées) basées sur la répétition, qui sont connues pour fonctionner dans des scénarios d'incrémental de classes à grande échelle : GDumb (Prabhu et al., 2020), iCaRL (Rebuffi et al., 2017), BiC (Y. Wu et al., 2019), Weight Aligning (WA) (Zhao et al., 2020) et, comme mentionné ci-dessus, avec l'approche semi-supervisée DMC+ (J. Zhang et al., 2020). Pour donner aux méthodes entièrement supervisées une évaluation plus équitable, nous proposons quelques expériences dans lesquelles elles sont pré-entraînées avec les mêmes données non étiquetées.

Toutes les méthodes sont testées en utilisant la même architecture DNN : un Wide-ResNet-28-8 (Zagoruyko & Komodakis, 2016) (WRN28-8) pour CIFAR-100 et un ResNet-18 (He et al., 2016) pour ImageNet. À titre de référence, WRN28-8 atteint 82.8% sur CIFAR-100 et la précision top-5 de ResNet-18 sur ImageNet-100 est de 94.4%.

4.5.1 Choix des hyperparamètres du PLCiL

Sur CIFAR-100, chaque session comporte 150 époques. La taille du mini-batch étiqueté B est de 32 avec $\mu = 7$. Le seuil de confiance τ est fixé à 0.8 et λ à 1. Sur ImageNet-100, l'apprentissage comprend 70 époques par session et utilise le jeu de paramètres suivant : $\{B = 32, \mu = 7, \tau = 0.7, \lambda = 3\}$. Le choix de ces hyperparamètres sera discuté en section 4.6.5.

Toutes les expériences sont entraînées avec une descente de gradient stochastique.

Le taux d'apprentissage est initialisé à 0.03 avec un moment de Nesterov fixé à 0.9. Nous utilisons une contrainte de *weight decay* de poids 10^{-4}, une décroissance du taux d'apprentissage de type *cosine annealing with warm restart* (Loshchilov & Hutter, 2017) avec une période initiale $T_0 = 10$ multipliée par un facteur $T_{mult} = 2$ après chaque redémarrage. L'évolution du taux d'apprentissage est illustrée en figure 4.3.

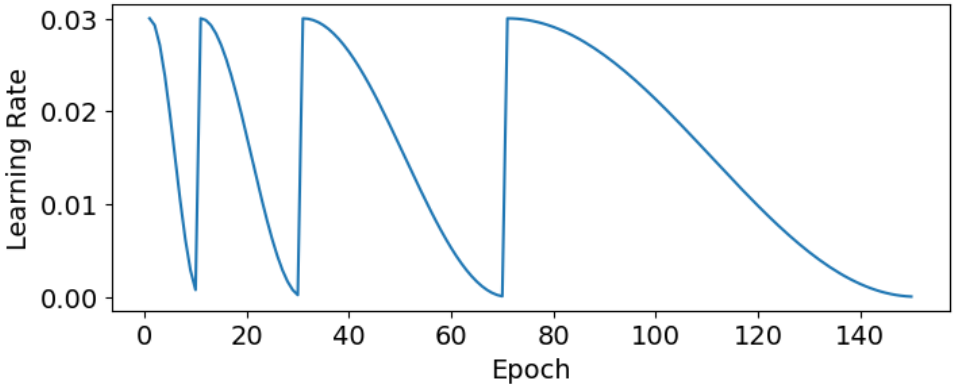

Figure 4.3 – Mise à jour du taux d'apprentissage avec *cosine annealing with warm restart* sur 150 epochs.

4.5.2 Incrémental de classes

Nous considérons en premier lieu le protocole classique d'incrémental des classes sur CIFAR-100 (Castro et al., 2018 ; Rebuffi et al., 2017). Nous fixons $s = 10$, c'est-à-dire 10 sessions de 10 classes. La taille de la mémoire pour la répétition K est fixée à 2000. Comme nous utilisons ici l'ensemble des données étiquetées, nous appelons ce protocole expérimental CIFAR-100-full. Pour DMC+ et PLCiL, nous émulons un flux de données non étiquetées en échantillonnant aléatoirement $100K$ d'images non étiquetées provenant d'ImageNet-1000 (sous-échantillonnées à 32×32).

Les résultats obtenus sont disponibles dans le tableau 4.2. Ils montrent l'intérêt d'exploiter des données non étiquetées dans un cadre d'incrémental de classes. En effet, notre méthode PLCiL surpasse systématiquement les autres méthodes de pointe sur CIFAR-100-full, gagnant +5.6% sur la précision finale. Dans la section suivante, nous montrons que la nature semi-supervisée de notre méthode est capable de traiter des scénarios encore plus difficiles où moins de supervision est disponible.

4.5.3 Incrémental de classes semi-supervisé

Les expériences suivantes étudient le comportement de notre méthode dans des scénarios encore plus proches des scénarios semi-supervisés : seul un ensemble très limité de

Méthode	Finale (%)	Moyenne (%)
GDumb	27.8	42.0
iCaRL	53.9	63.9
BiC	55.9	67.1
WA	50.8	64.4
DMC+	50.4	62.8
PLCiL	**61.5**	**74.0**

TABLE 4.2 – Comparaison des précisions finales et moyennes sur CIFAR-100-full.

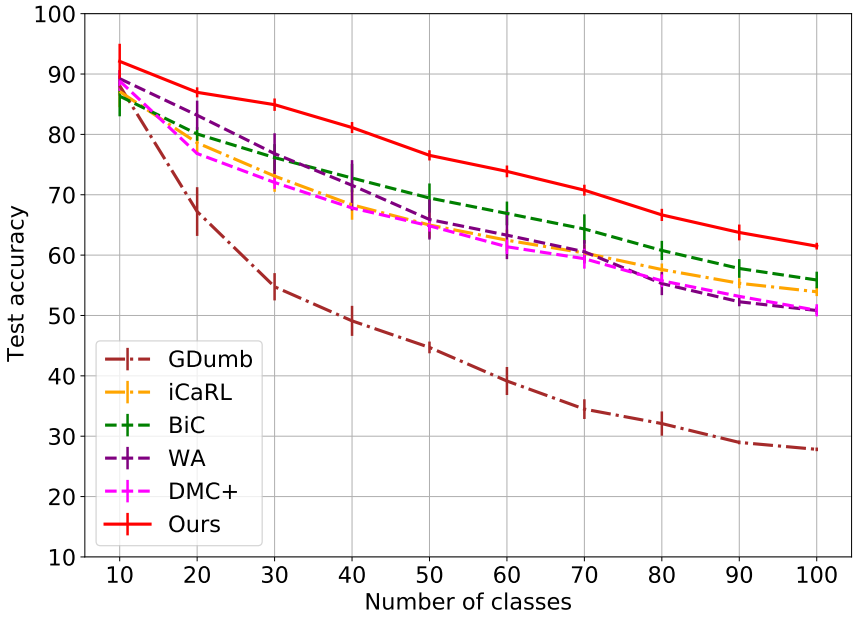

FIGURE 4.4 – Graphe des précisions incrémentales, avec écart-type, mesurées à chaque session de CIFAR-100-full.

données étiquetées est disponible, alors qu'une grande quantité de données non étiquetées bon marché est accessible. Cela rejoint la logique de l'évaluation de l'AAE effectuée en section 3.8.2.

Suivant ce principe, nous conservons les paramètres précédents avec $s = 10$ mais réduisons la taille de l'ensemble de données étiquetées. Pour CIFAR-100-20%, nous choisissons au hasard 100 échantillons par classe sur les 500, tandis que pour ImageNet-100-10%, 130 échantillons étiquetés par classe sont retenus. Cela correspond à la quantité de labels couramment utilisée dans les travaux semi-supervisés (SOHN et al., 2020). Le budget mémoire reste inchangé avec $K = 2000$.

Le processus de collecte des données non étiquetées est le même que précédemment : nous échantillonnons 100 000 données d'ImageNet-1000 au début de chaque session. Ici, on se limite au scénario « classes disjointes ». Ce scénario, défini en section 3.4.2, impose que les classes sont disjointes entre les données annotées et celles non-annotées ($\mathcal{C} \cup \mathcal{C}_u = \emptyset$). On

filtre donc ImageNet-1000 afin de retirer les images associées aux classes de $\mathcal{C}$. Par exemple, pour ImageNet-100-10%, cela correspond à retirer les 100 premières classes d'ImageNet-1000 $\mathcal{U}$. Les autres scénarios sont étudiés en section 4.5.4.

Comme ce contexte est très difficile pour les méthodes uniquement supervisées en raison de la rareté des données, nous comparons également avec une initialisation auto-supervisée sur $\mathcal{U}$. WRN28-8 et ResNet-18 ont été entraînés à l'aide de RotNet (Gidaris et al., 2018) sur ImageNet-1000 filtré de manière à respecter la condition c.d. ($\mathcal{C} \cup \mathcal{C}_u = \emptyset$). Le pré-entraînement auto-supervisé est le moyen le plus immédiat de permettre aux méthodes entièrement supervisées d'accéder à autant de données non étiquetées que PLCiL ou DMC+.

Les résultats sont présentés dans le tableau 4.3. Le manque de données est clairement perceptible pour les approches supervisées et seul GDumb donne les mêmes résultats que dans la configuration précédente. Cela s'explique du fait que son comportement ne dépend que de la taille du *buffer* puisque ses données d'apprentissage sont restreintes à celles stockées dans celui-ci. De son côté, PLCiL maintient le niveau de précision obtenu lors du premier protocole, prouvant que dans des scénarios avec très peu d'étiquettes, notre méthode peut exploiter efficacement des données non-annotées. De même, bien que les représentations obtenues par auto-supervision avec RotNet soient meilleures, l'exploitation des données non étiquetées est plus efficace avec PLCiL, qui surpasse toutes les méthodes supervisées pré-entraînées.

PLCiL est constamment meilleur que DMC+, qui exploite pourtant la même quantité de données que PLCiL. Nous pensons que cela est dû au fait que DMC+ exploite les données non-annotées uniquement pour distiller les connaissances d'un modèle appris avec une supervision complète. C'est-à-dire qu'à la session i, il apprend un modèle intermédiaire $\hat{\Theta}$ uniquement sur D_i, les 10 nouvelles classes. Puis la connaissance de Θ_{i-1} et $\hat{\Theta}$ est distillée dans un Θ_i en utilisant uniquement les données de $\mathcal{U}$. Un *fine-tuning* de Θ_i sur un *buffer* équilibré $\mathcal{B}_i$ contenant toute les classes permet d'incorporer la répétition. Du point de vue du dilemme plasticité-stabilité, cette approche se concentre sur l'exploitation des données non étiquetées pour améliorer la stabilité. La connaissance de Θ_i reste donc majoritairement dépendante de l'apprentissage supervisé de $\hat{\Theta}$ sachant qu'on se place dans un régime où le nombre de données dans D_i est très petit. Alors que PLCiL a un comportement similaire avec l_{kd}, i.e. on distille la connaissance de l'ancien modèle via des données non-annotées, il vise également à améliorer la plasticité en implémentant également un apprentissage de représentation auto-supervisé avec l_{self} et sa stratégie d'augmentation des données. L'étude de l'ablation donne plus de détails sur la contribution de chaque *loss*.

Méthode	CIFAR-100-20%				ImageNet-100-10%			
	Finale (%)	Moyenne (%)	Finale (%)	Moyenne (%)	Finale (%)	Moyenne (%)	Finale (%)	Moyenne (%)
	Init. aléatoire		Init. RotNet		Init. aléatoire		Init. RotNet	
GDumb	28.2	42.2	25.3	40.4	40.6	59.6	43.7	62.0
iCaRL	42.7	48.9	43.9	51.1	45.4	57.8	51.9	59.5
BiC	43.3	49.8	43.5	57.3	50.7	62.4	52.2	68.8
WA	40.5	49.7	45.5	55.4	30.2	54.7	40.9	64.5
DMC+	36.4	42.8	39.3	49.8	56.2	68.1	57.5	69.6
PLCiL	**59.8**	**66.5**	**59.5**	**67.6**	**61.3**	**73.8**	**61.2**	**75.0**

TABLE 4.3 – Expérimentations d'incrémental de classes avec une restriction sur la disponibilité des labels : CIFAR-100-20% and ImageNet-100-10%.

4.5.4 Influence du type de données non-annotées

Dans cette section, nous évaluons les trois scénarios définis en section 3.4.2 : classes disjointes, classes incluses et classes communes.

Dans les expériences effectuées précédemment, nous utilisons un ensemble de données non étiquetées qui est sémantiquement similaire à la cible tout en excluant tous les échantillons appartenant aux classes apprises afin de se placer dans le scénario de classes disjointes (scénario le plus stricte pour l'incrémental de classes). Ceci émule une application où les données non-annotées $\mathcal{U}$ et les données annotées $\mathcal{X}$ sont collectées dans le même environnement avec une forte contrainte sur les classes de $\mathcal{U}$. En pratique, cette contrainte n'est pas toujours applicable. Il est donc réaliste de considérer que les données sans label collectées sans a priori contiennent des instances de classes qui ont été ou seront apprises. Il est donc particulièrement intéressant de considérer également le scénario de classes incluses. On évalue aussi le cas de classes communes afin d'avoir le cas idéal de la semi-supervision comme borne supérieure de référence. Finalement, on essaie aussi le PLCiL avec un $\mathcal{U}$ provenant d'un dataset complètement extérieur et donc une distribution sémantique différente.

Nous répétons donc la même expérience sur ImageNet-100-10% avec initialisation aléatoire du DNN en utilisant 4 jeux de données non étiquetées différents pour $\mathcal{U}$:

- (a) classes disjointes : $\mathcal{U}$ contient toutes les données d'ImageNet-1000 en excluant celles associées aux 100 premières classes qui sont présentes dans ImageNet-100.
- (b) classes communes : ImageNet-100-10% n'utilisant que 10% des données annotées, on conserve les 90% restants pour constituer $\mathcal{U}$. On maintient ainsi le critère que les images soient disjointes entre $\mathcal{U}$ et $\mathcal{X}$.
- (c) classes incluses : $\mathcal{U}$ est l'intégralité d'ImageNet-1000 à l'exception des images présentes dans ImageNet-100-10%. Cela implique qu'approximativement 10% de $\mathcal{U}$ partage des classes avec $\mathcal{X}$.
- (d) données externes : $\mathcal{U}$ est le *dataset* Places-365 (ZHOU et al., 2018) qui contient

des données sans aucun lien sémantique à ImageNet.

Scenario	Last (%)	Avg (%)
a. classes disjointes	61.3	73.8
b. classes communes	76.9	83.3
c. classes incluses	65.1	76.2
d. données externes	59.6	72.7

TABLE 4.4 – Évaluations des différents scénarios d'incrémental de classes semi-supervisés sur ImageNet-100-10%.

Les résultats sont présentés dans le tableau 4.4. (c) montre que même une présence raisonnable des classes de $\mathcal{X}$ dans $\mathcal{U}$ est efficacement exploitée par notre modèle, améliorant ainsi les performances par rapport à notre scénario de base (a). Nous avons également essayé le scénario idéaliste de (b) avec exactement les mêmes classes dans les deux ensembles de données. Cela permet d'obtenir une borne supérieure de notre méthode pour un ensemble de données $\mathcal{U}$ parfaitement représentatif du flux annoté $\mathcal{X}$ dans sa globalité. Enfin, l'utilisation d'un ensemble de données sémantiquement différent dans (d) réduit légèrement les performances par rapport à (a), mais surpasse toujours les autres approches auxquelles on se compare dans le tableau 4.3.

4.6 Ablation du PLCiL

4.6.1 Bénéfice d'une plus grande architecture

Suites aux résultats obtenues dans l'étude préliminaire 3.5.3 sur iCaRL, nous avions 2 constats : i) les grandes architectures sont difficilement entraînable en incrémental supervisé en raison du faible nombre de données accessibles à chaque session ; ii) une plus grande architecture est bénéfique à la performance globale si on arrive à l'entraîner, via un pré-entraînement auto-supervisé par exemple. En raison des difficultés de passage à l'échelle rencontrés avec notre approche AAE, nous n'avions pas pu profiter du gain en plasticité apporté par un *backbone* avec plus de paramètres. Dans cette section, nous montrons que le PLCiL constitue un moyen adapté pour entraîner des grandes architectures en incrémental et sans pré-entraînement.

Nos expérimentations sur CIFAR-100 utilisent une architecture *backbone* WRN-28-8 (23M de paramètres entraînables) comme SOHN et al., 2020. Par ailleurs, l'auto-supervision – ou la semi-supervision – est utilisée pour entraîner de grands DNN pour les applications manquantes en données. Ces modèles, avec plusieurs dizaines voire centaines de millions de poids, sont désormais prédominants dans les *benchmarks* de classification.

Malgré cela, la plupart des méthodes de continu ne sont évaluées que sur des DNN plus petits : e.g. ResNet-32 pour CIFAR avec seulement 460K paramètres, loin de l'état-de-l'art en apprentissage batch. Cela remet en question la qualité de la représentation que des modèles aussi petits peuvent apprendre et, par conséquent, leur potentiel de plasticité. À titre de référence, la précision du traitement par lots est de 80,6% pour WRN28-8 et de 72.3% pour ResNet-32.

Nous répétons ici l'expérimentation CIFAR-100-full mais avec un ResNet-32 plus petit, pour observer l'influence de la taille du réseau. Nous présentons la comparaison des performances obtenue pour ces deux *backbones* dans la table 4.5.

Méthode	ResNet-32		WRN28-8	
	Finale (%)	Moyenne (%)	Finale (%)	Moyenne (%)
GDumb	20.7	35.0	27.8	42.0
iCaRL	47.1	57.9	53.9	63.9
BiC	50.8	62.7	55.9	67.1
WA	**52.1**	**65.6**	50.8	64.4
DMC+	43.9	58.4	50.4	62.8
Ours	46.9	62.1	**61.5**	**74.0**

TABLE 4.5 – Expérimentation CIFAR-100-full pour deux *backbones* : ResNet-32 (460K paramètres) et WRN28-8 (24M paramètres).

Avec le ResNet-32, notre PLCiL devient équivalent aux méthodes supervisées conçues avec ce *backbone* particulier. Cependant, lorsqu'on porte notre attention sur la manière dont les méthodes s'adaptent à des modèles plus grands, à l'exception de WA, toutes les méthodes deviennent plus précises avec des modèles plus grands. L'écart se creuse avec notre méthode, avec un résultat global de +12.4% obtenu en utilisant une architecture plus grande. L'utilisation de WRN28-8 au lieu de ResNet-32 semble moins profitable aux autres méthodes, et nuit même à la précision de WA. Notre PLCiL permet d'entraîner de plus grandes architectures grâce à la grande diversité visuelle soumise au modèle par le biais du signal auto-supervisé.

4.6.2 Rôle des différents termes de la loss

On décompose ici la *loss* du PLCiL en essayant plusieurs versions de la *loss* sur le *benchmark* CIFAR-100-full. L'objectif de cette évaluation est d'illustrer la contribution de chaque terme. On reporte les résultats obtenus pour chaque variation dans la partie supérieure de la table 4.6.

On remarque deux comportements complémentaires lorsque l_{kd} et l_{self} sont utilisés sé-

loss	10	20	30	40	50	60	70	80	90	100	Moyenne
l_{sup}	91.7	82.2	74.2	66.0	62.9	58.3	53.4	50.9	46.4	44.1	59.8
$l_{\text{sup}} + \lambda\eta l_{\text{kd}}$	91.7	47.6	59.1	68.9	71.5	68.7	67.6	66.1	63.5	61.9	63.9
$l_{\text{sup}} + \lambda l_{\text{self}}$	91.7	85.6	78.7	72.1	67.7	63.6	59.5	56.2	52.5	50.3	65.15
$l_{\text{sup}} + \lambda(l_{\text{self}} + \eta l_{\text{kd}})$	91.7	86.9	84.9	81.1	76.5	73.9	70.8	66.7	63.8	61.5	74.0
$l_{\text{sup}} + \lambda(l_{\text{self}} + \eta l_{\text{standardkd}})$	91.7	84.2	77.8	72.2	66.6	64.0	62.8	57.6	52.9	52.0	65.6

TABLE 4.6 – Incrémental de classes sur CIFAR-100-full en n'activant que certaines composantes de la *loss*. La dernière ligne concerne notre méthode lorsqu'on remplace notre distillation via pseudo-labeling par la distillation standard de Z. LI et HOIEM, 2017.

parément. La distillation avec *pseudo-labeling* l_{kd} se concentre sur la stabilité du modèle. Elle maintient la précision la plus constante d'une session à l'autre et obtient la meilleure précision finale malgré une moyenne plus faible. Cela est dû au fait qu'elle a du mal à apprendre de nouvelles classes, surtout pendant les premières étapes où la proportion d'anciennes classes est faible, ce qui rend la distillation moins pertinente. La version avec seulement $l_{selself}$ améliore la plasticité du modèle, permettant d'apprendre les nouvelles classes avec une meilleure précision comme cela est montré dans les premières sessions. Cependant, cette variante manque toujours de régularisation pour atténuer l'oubli catastrophique au cours des étapes ultérieures. En combinant les deux, PLCiL optimise le compromis *plasticité-stabilité* et donne des résultats satisfaisants pendant toutes les sessions, ce qui se manifeste par une moyenne incrémentale bien supérieure.

4.6.3 Notre distillation face à une distillation standard

Nous avons remplacé notre l_{kd} par une distillation de connaissance standard, comme celle utilisée par Z. LI et HOIEM, 2017 ; Y. WU et al., 2019 ; ZHAO et al., 2020, mais toujours appliquée sur des données non étiquetées uniquement. Les résultats sont donnés dans la dernière ligne du tableau 4.6. La distillation standard ($l_{\text{standardkd}}$) a peu ou pas d'effet et donne des résultats similaires à BiC et WA. Cette comparaison met en évidence l'efficacité de notre version de la distillation adaptée pour traiter des données non étiquetées via un pseudo-labeling.

4.6.4 Biais dans le classifieur FC du PLCiL

ZHAO et al., 2020 montrent que le biais du classifieur FC est identifiable en analysant directement les normes des poids associés à chaque classe dans $W_i = w_{iy=1}^{is}$. En particulier, les approches avec répétitions sont biaisés vers les nouvelles classes en raison des poids

bien plus fort :

$$\frac{1}{s}\sum_{y=(i-1)s+1}^{s}||w_i^y||_2 >> \frac{1}{(i-1)s}\sum_{y=1}^{(i-1)s}||w_i^y||_2 \tag{4.9}$$

Une simple visualisation des normes du classifieur FC (cf. figure 4.5) et des matrices de confusion (cf. figure 4.6) permet d'observer qu'un biais envers les nouvelles classes est présent. Ce phénomène est inhérent aux méthodes de répétition en raison du déséquilibre entre nouvelles classes et anciennes stockées en mémoire. Cependant, ce biais reste très faible et se remarque surtout pour la session 10. Dans tous les cas, l'amplitude séparant les anciennes classes et des nouvelles reste faible comparé à l'exemple introduit en section 2.4.3. L'homogénéité des poids indique que le PLCiL propose une régularisation implicite du classifieur FC, surtout lors des premières étapes d'apprentissage, alors qu'aucune solution spécifiquement conçue pour traiter le biais n'a été implémentée.

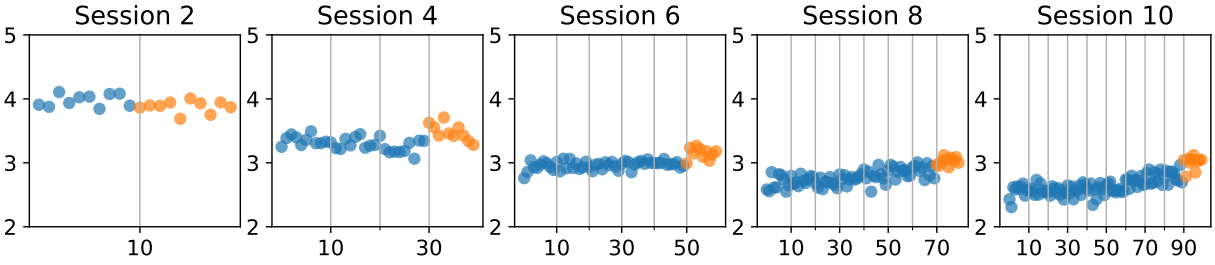

FIGURE 4.5 – Norme des poids associés à chaque classe dans la matrice W_i du classifieur FC. Résultats obtenus avec PLCiL à la fin des sessions 2, 4, 6, 8 et 10 sur CIFAR-100-full. Les normes associées aux anciennes classes sont en bleu. Les points oranges sont les classes qui viennent d'être apprises.

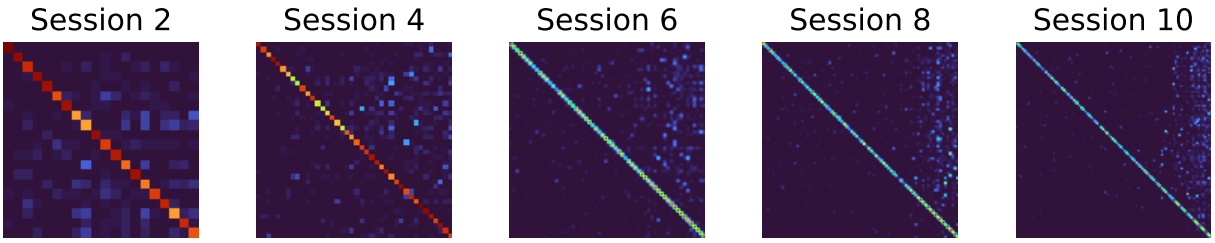

FIGURE 4.6 – Matrices de confusion du PLCiL obtenues pour les sessions 2, 4, 6, 8 et 10 sur CIFAR-100-full.

μ	0	1	2	3	7	15	31
Moyenne	59.8	69.2	71.2	72.1	74.0	74.4	75.0
Finale	44.1	55.5	58.9	59.5	61.5	62.2	63.6

TABLE 4.7 – Évolution de la performance en fonction des valeurs de μ sur CIFAR-100-full.

4.6.5 Sensibilité des hyperparamètres

Ratio entre mini-batch annoté/non-annoté : μ

L'hyperparamètre μ définit la quantité de données non étiquetées échantillonnées dans chaque mini-batch, c'est-à-dire que pour chaque taille de mini-batch annoté B, notre algorithme considère μB données sans labels. Ainsi, l'augmentation de μ accroît directement la diversité visuelle perçue par le modèle. Cependant, cela implique également des mini-batchs plus importants pour l'entraînement, ce qui peut induire des coûts importants en temps et en termes de calcul. Dans le tableau 4.7, avec B fixé à 32, nous constatons une augmentation constante des performances avec des mini-batchs non étiquetés plus grands. Nous avons choisi de garder $\mu = 7$ pour toutes nos expérimentations car cela donne des résultats satisfaisants tout en maintenant le temps d'apprentissage et les besoins matériels à des échelles comparables aux autres méthodes d'incrémental de classes (par exemple, $B + \mu B$ donne une taille totale de mini-batch de 256 avec $\mu = 7$ et $B = 32$). Des valeurs plus élevées de μ n'apportent que des améliorations mineures bien qu'elles soient beaucoup plus coûteuses.

Sélectivité de τ et poids du *pseudo-labeling* λ

Dans le tableau 4.8, nous affichons les résultats pour plusieurs combinaisons des hyperparamètres τ et λ. Les tendances indiquent que λ devrait être maintenu proche de 1, ce qui signifie que donner trop de poids à la partie auto-supervisée de la *loss* a un impact négatif sur l'apprentissage des classes.

Le PLCiL est moins sensible à la valeur du seuil. Pour $\lambda \leq 2$, notre modèle atteint au moins 72.6% de précision moyenne pour tous les τ testés ici. Cela est dû au fait que notre modèle répond avec confiance pour la majorité des données non étiquetées vues, en produisant des valeurs élevées qui vont au-delà de la plupart des valeurs seuils. Ceci est probablement dû à la nature *curated* de notre jeu de données non étiquetées (ImageNet) qui contient des informations visuelles facilement transférables à CIFAR (domaines proches). Cette dépendance aux « *curated datasets* », i.e. des données qui ont été collectées, sélectionnées et préparées pour l'entraînement de DNN, est l'une des principales faiblesses des méthodes semi ou auto-supervisées. Ces *datasets* étant disponibles au grand

τ	λ	Moyenne (%)	Finale (%)
0.5	1	73.6	61.9
0.5	2	73.5	61.0
0.5	5	71.0	60.9
0.5	7	69.4	58.5
0.5	10	62.2	56.9
0.7	1	73.6	61.7
0.7	2	73.6	61.5
0.7	5	69.6	62.3
0.7	7	69.1	60.8
0.7	10	67.3	57.5
0.8	1	**74.7**	62.3
0.8	2	74.4	**62.4**
0.8	5	73.4	62.3
0.8	7	72.7	59.3
0.8	10	67.3	59.3
0.9	1	72.6	59.3
0.9	2	73.6	61.6
0.9	5	73.4	58.5
0.9	7	72.3	59.4
0.9	10	68.5	50.9

TABLE 4.8 – Évaluation du PLCiL pour différentes paires (τ, λ). Pour cette expérimentation sur CIFAR-100-full, on reporte les résultats sur seulement une permutation des classes au lieu des trois exécutions habituelles.

public et uniformisés, ils permettent une évaluation standardisée des solutions. Cependant, on sait que la tendance à vouloir créer des ensembles respectant l'hypothèse i.i.d. facilite l'apprentissage. Cela ne permet pas de considérer une approche réaliste où des données collectées dans la nature ne suivent pas une distribution précise.

Une approche plus réaliste serait de collecter des données dans la nature à la volée pour pouvoir les exploiter directement via *pseudo-labeling*. Les hypothèses *curated* ou i.i.d. sont omises. Nous pensons que dans un tel contexte, τ devient crucial pour filtrer les informations sources de bruit ou non pertinentes.

4.6.6 Analyse qualitative du *pseudo-labeling*

On s'intéresse ici à analyser les pseudo-labels produits par les modèles professeurs pendant l'apprentissage.

Protocole

Pour cela, on définit un contexte artificiel précis en utilisant CIFAR-100-20%. Sur les 100 classes de CIFAR-100, on sélectionne manuellement 20 classes que l'on sépare en deux ensembles de 10 :

— C_1 = {'seal', 'ray', 'rose', 'bottle', 'apple', 'television', 'couch', 'lion', 'bridge', 'squirrel'}

— C_2 = {'dolphin', 'shark', 'sunflower', 'plate', 'orange', 'telephone', 'table', 'wolf', 'skyscraper', 'rabbit'}

Ces classes ont été sélectionnées en choisissant des paires de classes de même super-classe (e.g. (« seal », « dolphin ») dans « aquatic mammals », (« rose », « sunflower ») dans « flowers »). Cela permet de créer deux collections contenant des classes proches mais différentes.

On réalise un incrémental à deux sessions consécutives pour apprendre C_1 puis C_2. Puisqu'on utilise CIFAR-100-20%, cela restreint la partie annotée à 100 images par classe. On puise dans les 80% de données de CIFAR-100 non utilisées par CIFAR-100-20% (400 images/classe) pour constituer le *dataset* non-annoté en suivant les trois scénarios : classes disjointes, communes et incluses.

— (a) classes disjointes : $\mathcal{U}$ contient les 80 autres classes de CIFAR.

— (b) classes communes : $\mathcal{U}$ contient $C_1 \cup C_2$.

— (c) classes incluses : $\mathcal{U}$ contient les 100 classes de CIFAR.

Afin de voir un maximum de données non-annotées, on choisit volontairement une valeur très grande $\mu = 63$. On utilise un buffer mémoire avec $K = 400$.

Résultats

On propose ici une simple évaluation qualitative du *pseudo-labeling* via les matrices de confusion.

Pour chaque scénario, on effectue les relevés lors de la dernière epoch de la session 2. On dispose de Θ_2, le nouveau modèle en train d'être optimisé, et de Θ_1, le modèle de la première session qui sert à générer les labels pour l_{kd}. Une epoch correspond à une passe sur les données annotées $D_2 \cup \mathcal{B}_\infty$. Cela correspond à 1400 images réparties en 44 mini-batch de taille $B = 32$. Pour $\mu = 63$, on constitue un mini-batch non-annoté de 2016 échantillons, pour un total de $44 \times 2016 = 88704$ vus pendant l'epoch. On analyse donc les prédictions de Θ_2 et Θ_1 sur ces 88 704 images.

Le tableau 4.9 reporte le pourcentage des données non-annotées qui se voient attribuer un pseudo-label, i.e. dont la prédiction franchit le seuil $\tau = 0.8$. Comme on pouvait s'y attendre, plus les données non-annotées sont corrélées avec les classes apprises, plus le pseudo-labeling est pertinent. D'autre part, Θ_2 connaissant plus de classes est capable

Scénario	Θ_2	Θ_1
(a)	61.2	59.9
(b)	91.8	81.3
(c)	65.2	63.2

TABLE 4.9 – Pourcentage des données non-annotées de $\mathcal{U}$ qui produisent une prédiction supérieure au seuil τ et qui par conséquent, sont retenu pour le pseudo-labeling.

d'annoter plus de données. On remarque néanmoins que même dans les scénarios (a) et (c), où les données non-annotées sont théoriquement majoritairement voir complètement des instances de classes extérieures, le modèle est capable de décider d'un pseudo-label pour plus de 60% des images.

Les matrices de confusions obtenues sont disponibles dans les figures 4.7 à 4.9.

Pour le scénario (b) (fig. 4.8), les deux modèles sont pertinents pour annoter les classes qu'ils connaissent. C'est le cas où le *pseudo-labeling* est le plus efficace, d'où le net gain en précision que l'on avait observé dans l'expérience 4.5.4. Le modèle Θ_1, du fait qu'il ignore la moitié des classes, est incapable de proposer une étiquette correcte pour la moitié des données. On constate cependant que, plutôt que de rejeter ces images, il rabat généralement sa décision sur les classes proches appartenant à la même super-classe.

Ce comportement se retrouve dans les scénarios (a) et (c) ce qui semble expliquer le fort taux d'acceptation par le seuil. Les modèles ont tendance à favoriser l'attribution d'un pseudo-label plutôt que le rejet de l'image. Le *pseudo-labeling* est de ce fait bruité car les images de $\mathcal{U}$ sont mal étiquetées, mais conserve une certaine logique par le fait que le modèle, lorsqu'il se trompe, choisit quand même une catégorie visuellement similaire.

4.7 Conclusion

Avec ce chapitre consacré au PLCiL, nous introduisons une méthode d'incrémental de classes qui tire profit des données non-annotées pendant le processus d'apprentissage. La régularisation de consistance via *pseudo-labeling* est une solution simple et efficace pour la semi-supervision. Du fait qu'elle utilise directement la tête de classification de la tâche principale, aucun paramètre entraînable supplémentaire n'est introduit dans le modèle. Les régularisations interviennent principalement au niveau de la fonction coût. Les termes auto-supervisés viennent simplement s'additionner à la *loss* de classification usuelle qui est ensuite rétro-propagée dans l'ensemble des poids du DNN à la manière d'une méthode *end-to-end* standard. Le PLCiL est de ce fait une solution pouvant facilement s'intégrer à une méthode d'incrémental de classe qui souhaiterait exploiter des données supplémentaires.

Il constitue ainsi une alternative au pré-entraînement.

Par le biais de notre validation expérimentale, nous montrons l'intérêt de la semi-supervision pour plusieurs problématiques clés de l'apprentissage continu : la lutte contre l'oubli catastrophique, le recours à de plus grandes architectures et la possibilité de traiter des problèmes pauvres en données annotées. Tout cela grâce à un processus favorisant des représentations plus stables et plus affinées.

4.8 Publication

Lechat, A., Herbin, S. & Jurie, F. (2021). Pseudo-Labeling for Class Incremental Learning. *Proceedings of the British Machine Vision Conference, BMVC, 2021*, (Oral)

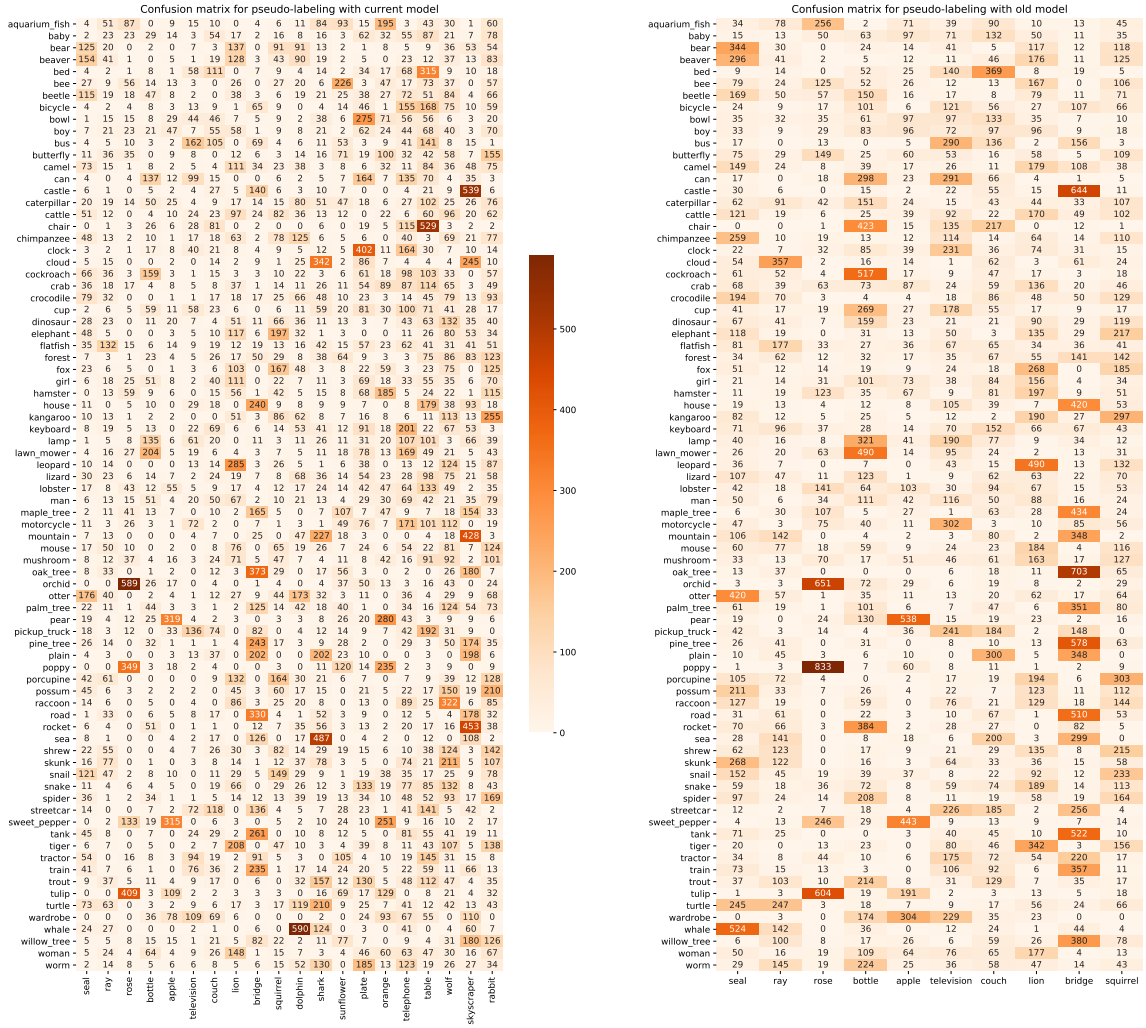

FIGURE 4.7 – Scénario a : classes disjointes. A gauche la matrice de confusion obtenue pour Θ_2 et à droite celle pour Θ_1. En ordonné, on indique la classe de l'image non-annoté et en abscisse le pseudo-label proposé par le modèle. À visualiser de préférence en PDF.

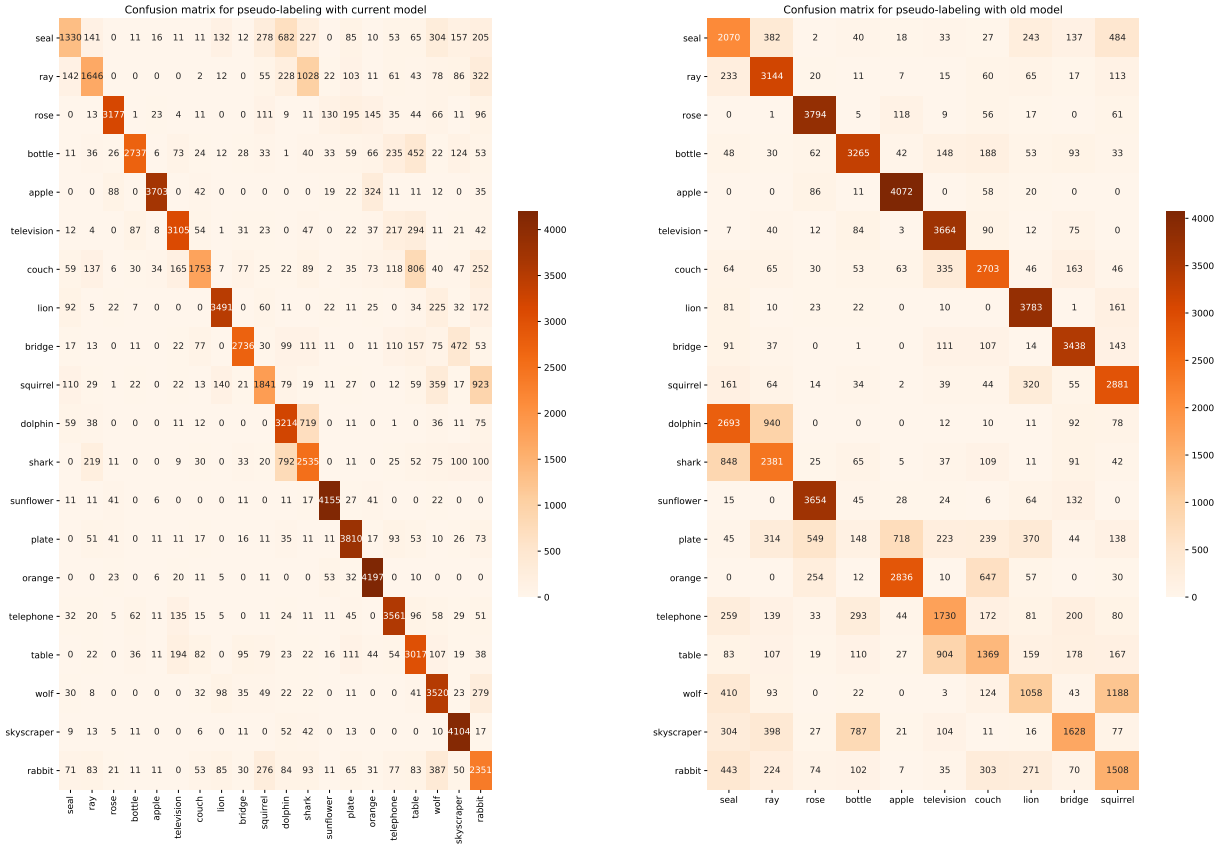

FIGURE 4.8 – Scénario b : classes communes. A gauche la matrice de confusion obtenue pour Θ_2 et à droite celle pour Θ_1. En ordonné, on indique la classe de l'image non-annoté et en abscisse le pseudo-label proposé par le modèle. À visualiser de préférence en PDF.

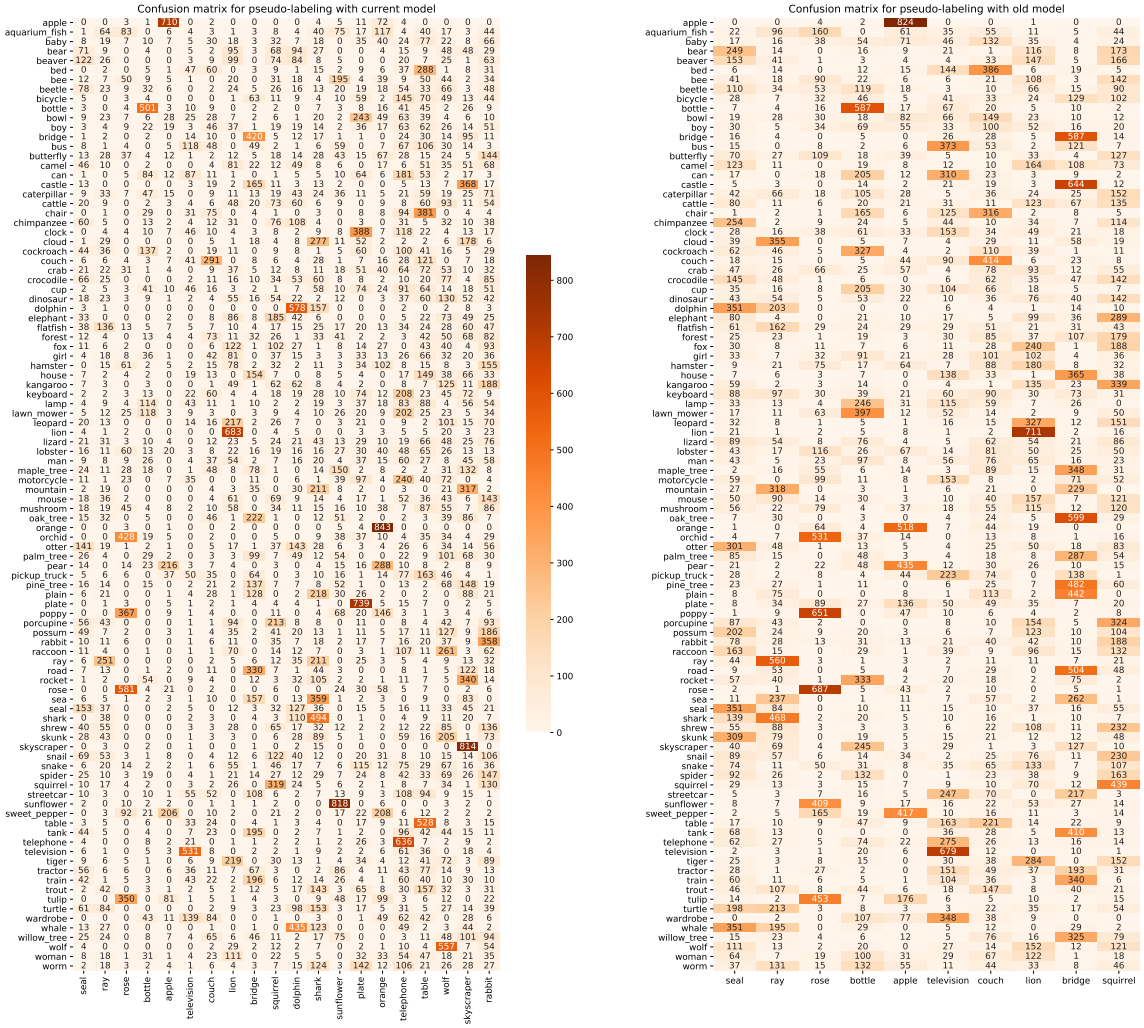

FIGURE 4.9 – Scénario c : classes incluses. Scénario a : classes disjointes. A gauche la matrice de confusion obtenue pour Θ_2 et à droite celle pour Θ_1. En ordonné, on indique la classe de l'image non-annoté et en abscisse le pseudo-label proposé par le modèle. À visualiser de préférence en PDF.

Chapitre 5

Perspectives : apprentissage incrémental pour les systèmes de questions-réponses visuelles et la classification multi-labels.

Contents

5.1 Introduction

Au cours de ce chapitre, nous nous intéressons à l'extension du problème d'apprentissage continu à différentes applications de visions artificielles.

L'apprentissage incrémental cherche à proposer une solution optimisée pour traiter les problèmes dont les données sont en perpétuelle évolution. Au cours des sections précédentes, nous avons restreint notre cadre d'étude à la reconnaissance d'objets. La classification est la tâche de référence de la vision artificielle et les innovations en *deep learning* passent souvent par une première validation sur des *benchmarks* standards tels que CIFAR ou ImageNet. Le domaine de l'apprentissage continu ne fait pas exception.

La reconnaissance d'objet est avant tout un moyen de proposer un cadre simple et maîtrisé où les données sont abondantes. L'apprentissage incrémental à cependant pour vocation de traiter des problèmes complexes où les contraintes sur les données sont le facteur limitant pour les méthodes batch usuelles.

L'objectif final de l'apprentissage continu est de concevoir un système pouvant apprendre et se perfectionner tout au long de sa vie, à la manière d'un être humain. On peut facilement imaginer l'intérêt d'un tel système pour n'importe quel domaine d'application.

Dans ce chapitre, on explore donc des applications de vision plus complexes : les questions-réponses visuelles (QRV) et la classification multi-labels (CML). Afin de conserver un lien avec les travaux précédents, on s'intéresse à des applications d'interprétation du contenu de l'image qui, pour nous, sont des extensions de la reconnaissance d'objet. On attend en effet un modèle capable d'une interprétation plus poussée du contenu des images.

5.2 Les contraintes des scénarios d'apprentissage incrémental en pratique

Les études présentées dans les chapitres précédents sont effectuées dans un contexte très précis, celui de l'incrémental de classes pour la classification mono-label. Ce problème s'appuie sur des hypothèses fortes et exploite des *datasets* parfaitement adaptés pour l'apprentissage de DNN. Ce contexte est favorable au développement et à l'analyse par ablation des méthodes.

Dès lors que l'on considère des applications pratiques plus complexes, il est raisonnable de discuter l'intérêt et même la faisabilité de telles contraintes. On ne peut souvent pas garantir les hypothèses habituelles : avoir accès à une quantité abondante de données, des classes qui sont i.i.d., des images qui appartiennent au même domaines ou encore des

classes qui arrivent de manière disjointes. Or c'était grâce à ces a priori sur les données qu'il était simple d'expliciter les différents scénarios de l'apprentissage incrémental dans le cas de la reconnaissance d'objet.

Lorsqu'on cherche à établir une définition d'apprentissage continu pour ces applications, on se rend compte que définir l'incrémental avec un point de vue orienté donnée est parfois complexe. Les corrélations, les biais ou encore la multi-modalité rendent difficile de visualiser des scénarios contraints où l'incrémental se produit sur un point précis (les classes, les domaines, les tâches...).

Dans la suite, nous essayons de proposer une discussion autour des possibles définitions de l'apprentissage incrémental appliqué au QRV et au CML. On base notre réflexion sur la composante fondamentale de l'apprentissage incrémental : un flux de données présenté au modèle de manière séquentielle. C'est la nature de ce flux qui dépend en pratique de différents facteurs. Il peut dépendre par exemple du régime de collecte et d'annotation des données mais il peut aussi être contrôlé manuellement, de façon à guider l'apprentissage.

5.3 Les systèmes de questions-réponses visuelles (QRV)

5.3.1 Définition du QRV

Un système de questions-réponses visuelles (QRV) doit procéder de la manière suivante : un utilisateur pose une question, en langage naturel, sur le contenu d'une image. L'algorithme, à partir de cette question et de l'image, doit générer une réponse correcte et interprétable par l'utilisateur. On donne un exemple en figure 5.1. Lorsque l'interaction entre l'individu et la machine est étendue à une succession de questions-réponses sur la même image, cela devient un dialogue visuel (cf. fig. 5.2).

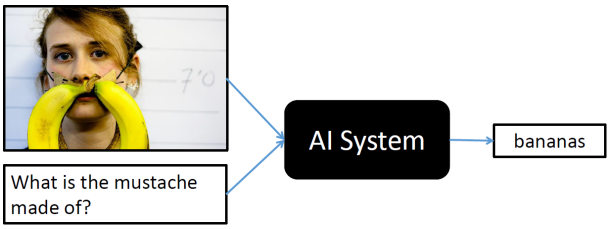

FIGURE 5.1 – Principe du QRV. Figure extraite de GOYAL et al., 2017.

Le QRV est un problème multimodal mêlant vision et langage naturel. Un système doit combiner sa compréhension visuelle avec des compétences linguistiques pour comprendre la question mais aussi formuler la réponse.

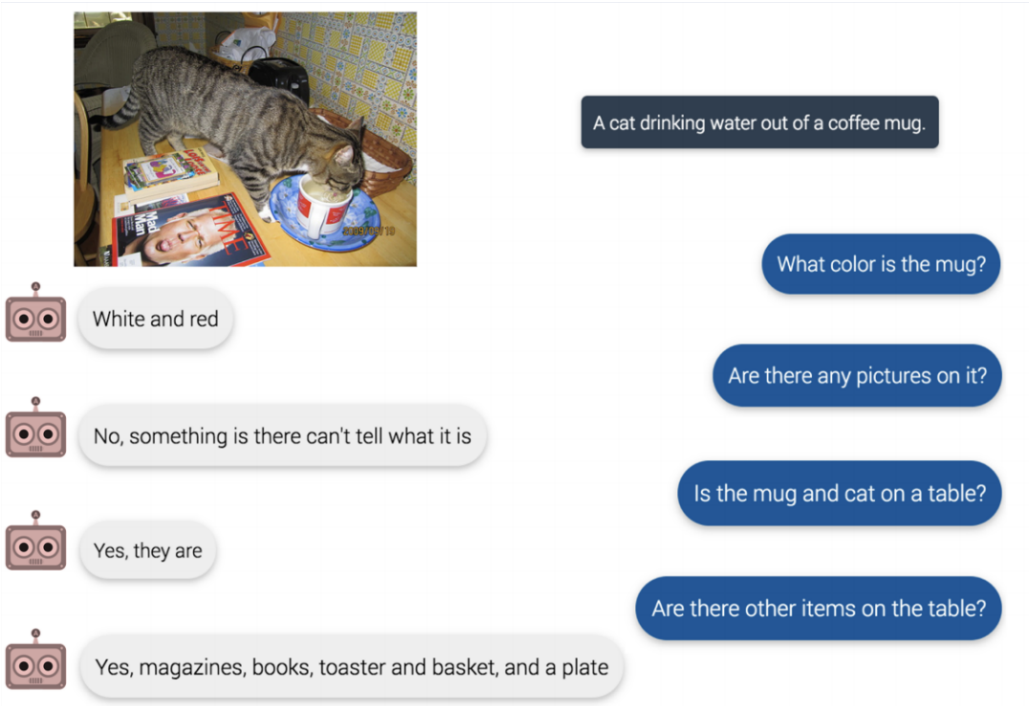

FIGURE 5.2 – Exemple d'un dialogue visuel complet sur une image. Figure extraite de DAS et al., 2017.

5.3.2 Les données du QRV

Les données des *benchmarks* QRV se présentent sous la forme de triplets (Image, Question, Réponse). On peut notamment citer les *datasets* suivants :

— CLEVR (JOHNSON et al., 2017) : un ensemble d'images synthétiques représentant des objets géométriques simples (cubes, sphères, cylindres...) dont la couleur, la taille et la position varie. Les questions demandent principalement de décrire les objets mais demande aussi au modèle de raisonner sur l'image en comparant ou identifiant les objets entre eux. Le *dataset* étant synthétique, il est possible de générer les questions-réponses de manière automatique.

— VQAv2 (GOYAL et al., 2017) : images naturelles venant de la base Microsoft COCO. Ces images représentent des scènes de la vie courante. Les questions-réponses ont été définies manuellement, crowdsourcée sur Amazon Mechanical Turk. La moitié des questions sont binaires (oui/non) et la grande majorité des réponses sont un mot unique.

— GQA (HUDSON & MANNING, 2019) : les images sont similaires à celles du VQAv2. Les questions et réponses sont générées artificiellement grâce à la représentation de chaque image en graphe de scène issu de Visual Genome (KRISHNA et al., 2017). Les questions sont de ce fait conçues pour demander un raisonnement de la part du système comme pour CLEVR.

GQA
1. Who wears the gloves? player
2. Are there any horses to the left of the man? no
3. Is the man to the right of the player that wears gloves? no
4. Is there a bag in the picture? no
5. Do the hat and the plate have different colors? yes

VQA
1. What is the man holding?
2. Where are the people playing?
3. Is the player safe?
4. What is the sport being played?

GQA
1. What is the person doing? playing
2. Is the entertainment center at the bottom or at the top? bottom
3. Is the entertainment center wooden and small? yes
4. Are the pants blue? no
5. Do you think the controller is red? no

VQA
1. What colors are the walls?
2. What game is the man playing?
3. Why do they stand to play?

GQA
1. Are there any coats? yes
2. Do you see a red coat in the image? no
3. Is the person that is to the left of the man exiting a truck? no
4. Which place is this? road

VQA
1. Where is the bus driver?
2. Why is the man in front of the bus?
3. What numbers are repeated in the bus number?

FIGURE 5.3 – Comparaisons sur les mêmes images des questions générées artificiellement pour GQA et les questions posées par les annotateurs du VQA. Figure extraite de HUDSON et MANNING, 2019.

La nature de la tâche rend la collecte des données complexe. À partir d'une image, il faut créer des questions-réponses pertinentes. Il est possible d'automatiser le processus comme le fait GQA à partir de Visual Genome puisque ce dernier contient des graphes de scènes. Cependant, les questions ainsi générées sont limitées et semblent peu naturelles pour l'humain. Le meilleur moyen d'avoir des triplets (I, Q, R) réalistes est de passer par des annotateurs humains, mais le processus devient alors coûteux en ressources et en temps. De ce fait, la taille des *datasets* annotés utilisés est faible en comparaison de la tâche que représente le QRV.

5.3.3 Enjeux et problématiques QRV

Multi-modalité : Le QRV est un challenge intéressant pour évaluer les modèles multi-modaux vision et langage. On dispose de deux entrées, l'image et la question, appartenant à deux domaines complètement différents. Le modèle doit être capable de les projeter dans des espaces de représentations compatibles pour pouvoir traiter l'information visuelle en fonction du contenu de la question.

Du point de vue de l'apprentissage des représentations, on a donc un besoin d'apprendre des représentations visuelles, des représentations linguistiques mais surtout un espace de représentation conjoint permettant de résoudre le QRV.

Données non-exhaustives : Le QRV est par nature un problème sans limites clairement définies. En pratique, le nombre de questions possibles pour une image donnée est imprévisible. Un utilisateur peut très bien poser également une question incohérente avec l'image. De même, l'ensemble des réponses possibles est virtuellement infini en rai-

son des multiples formulations possibles. On peut donc affirmer que constituer un *dataset* exhaustif de la tâche QRV n'est pas concevable.

Distribution des données non-uniformes : Le problème n'est pas uniforme au sens où les différentes questions et réponses ont des fréquences d'occurrence variables. Les questions-réponses simples sont courantes. A l'inverse, certaines questions ou réponses très spécifiques peuvent être très rares. Par exemple, le VQAv2 est constitué à moitié de questions-réponses oui/non et à 10% de questions commençant par « how many » dont la réponse est un nombre. De ce fait, les questions des *datasets* de QRV ont une distribution à longue traîne.

QRV à l'aveugle : Ce problème désigne un système qui n'utilise pas du tout l'information visuelle pour proposer une réponse. En effet, la question permet de limiter l'espace des réponses possibles. Si de plus, les questions-réponses présentes des biais ou corrélations statistique, le modèle peut obtenir une précision correcte en répondant à l'aveugle. Par exemple, le VQAv1 était connu pour être biaisé : « 2 » était la réponse à 41% des questions commençant par « How many » et « oui » était la réponse à 87% des questions commençant par « Do you see ».

Évaluation ambigu : L'évaluation est une problématique cruciale du QRV. Il est difficile de quantifier si une réponse est valide ou non. Une réponse en langage naturel peut se formuler de multiples façons, utiliser des synonymes ou différents niveaux d'abstraction ou de précision. Pour simplifier la tâche, de nombreux travaux se placent dans une configuration de type classification. C'est-à-dire que chaque réponse possible est codée comme une classe. On peut dès lors utiliser les métriques de la classification. Cependant, toutes les ambiguïtés ne sont pas levées. Par exemple, si l'on considère VQAv2 du point de vue de la classification, le problème contient les classes « computer », « laptop » et « macbook ». En termes de précision, seule la réponse qui est identique à la vérité terrain est comptée comme juste. Pour un humain, utiliser « laptop » pour désigner un « macbook » reste sémantiquement correct.

Raisonnement, logique et connaissances externes : Le QRV requiert des capacités cognitives avancées de la part du modèle. Les questions peuvent demander au modèle d'effectuer un raisonnement plus ou moins poussé. GQA et CLEVR s'intéressent particulièrement à évaluer la capacité de raisonnement des modèles. GQA, pour chaque triplet, fournit aussi le graphe de raisonnement détaillant les étapes successives à réaliser pour répondre à la question.
Exemple : *Is there any cat to the left of the white keyboard ?*

Graphe de raisonnement en 4 étapes :'select : keyboard -> filter color : white -> relate : cat,to the left of -> exist : ?'

De même, l'information requise pour répondre à une question n'est pas forcément entièrement contenue dans l'image. Un être humain doit souvent faire appel à sa connaissance externe, à sa logique ou à son sens commun pour proposer une réponse. Marino et al., 2019 proposent OK-VQA, un *benchmark* dérivé de VQAv2 dédié à l'évaluation des méthodes disposant d'un accès aux connaissances externes. Des exemples de triplets sont donnés en figure 5.4.

Figure 5.4 – Exemples tirés du *dataset* OK-VQA. Il propose des questions nécessitant une culture externe. Figure extraite de Marino et al., 2019.

5.4 Le QRV incrémental

5.4.1 Les spécificités du QRV incrémental

Les corrélations entre les domaines

Définir un scénario standardisé de QRV incrémental est subtil. On se rend vite compte qu'il serait difficile de limiter l'incrémental à une seule composante : les questions, les images ou les réponses. En effet, les trois domaines sont corrélées. Les questions sont directement corrélées avec les réponses possibles et même les images, selon ce qu'elles représentent, vont rendre possible de nouvelles paires questions-réponses.

En pratique, il semble naturel qu'un système QRV rencontre, au cours de son existence, de nouveaux contextes visuels, de nouveaux termes de vocabulaire ou de nouvelles tournures grammaticales. À chaque fois qu'il rencontre un nouveau contexte, de nouveaux

triplets (I, Q, R) sont présentés. Du fait des corrélations, il est possible qu'une image, une question ou une réponse soit pertinente dans plusieurs sessions. Chercher à contraindre les images et questions à être uniques nécessiterait un *dataset* illimité. Pour les réponses, des questions introduites à différentes sessions peuvent avoir des réponses identiques, notamment les questions binaires ou de dénombrement.

Guider la connaissance : décomposer le QRV en une succession de sous problèmes.

Pour les raisons définies dans le paragraphe précédent, et notamment parce que les *datasets* QRV sont encore limités, chercher à créer un protocole en imposant des fortes contraintes sur la façon de présenter les données n'est pas plausible pour le QRV.

L'apprentissage incrémental, bien que souvent présenté comme l'étude du comportement des systèmes sur ces régimes de données très contraints, ne se limite pas à cette approche. Il peut aussi s'envisager comme une façon de guider le modèle dans son apprentissage.

Cette approche guidée, en lien avec le domaine du *curriculum learning* (Soviany et al., 2021), consiste à manuellement sélectionner les connaissances apprises au modèle à chaque étape de sa vie. Le principe est analogue à l'apprentissage humain. Plutôt que de chercher à apprendre une tâche à grande échelle dans sa totalité en une fois, il est plus simple et plus adapté pour un individu de procéder par étapes qui se suivent avec une certaine logique. L'idée est d'apprendre les bases simples en premier puis de progressivement augmenter la difficulté jusqu'à la maîtrise.

Dans ce cas, l'apprentissage incrémental vise à proposer une alternative à l'apprentissage batch pour les tâches trop complexes ou dont les données trop limitées pour être apprises en une fois. Au lieu de considérer le problème dans sa globalité, on le décompose en sous-problèmes plus petits que l'on apprend successivement.

Le QRV est par définition une tâche quasi-infinie et à difficulté variable. Cela fait du QRV une application propice pour un apprentissage incrémental guidé.

5.4.2 Protocoles et Expérimentations

Dans cette section, nous proposons deux protocoles simples à mettre en place avec les *datasets* existants afin de mettre en place les idées exposées précédemment.

Afin d'obtenir des résultats expérimentaux, nous utilisons une version simplifiée du modèle état-de-l'art proposé par Anderson et al., 2018 et illustré en figure 5.5. Ce modèle est constitué de deux branches, l'une encodant l'information visuelle contenue dans les images à partir de caractéristiques profondes (ResNet) apprises sur la base Image-

Net, l'autre encodant la sémantique des questions à partir d'un réseau récurrent (GRU). Ces deux branches sont ensuite combinées dans un espace conjoint via un produit de Hadamard. Cette représentation conjointe est utilisée par une tête de classification pour prédire la réponse. Ce modèle ramène donc le QRV à un problème de classification où chaque réponse est une classe. Sa précision en entraînement batch sur VQAv2 est 53%.

La version simplifiée utilise un encodeur visuel qui retourne un unique vecteur de représentation de taille 2048, global pour toute l'image. La version complète proposée par Anderson et al., 2018 remplace le CNN par un faster R-CNN (Région CNN) appris sur les images COCO afin de détecter jusqu'à 100 objets d'intérêts. En pratique, on retient les 36 objets qui retournent le plus fort taux de confiance pour la détection. Le réseau extrait les patches contenant ces objets puis les encode indépendamment via un CNN. La résultante est une matrice de taille 36×2048. Afin de créer le vecteur unique de taille 2048, on effectue une somme pondérée des 36 lignes de la matrice. Les poids sont calculés via un mécanisme d'attention entre ces représentations visuelles et la représentation de la question.

Les modèles VQA basés sur deux branches indépendantes utilisent généralement des encodeurs pré-entraînés aussi bien pour le texte que pour l'image. Ces encodeurs ne sont pas entraînables pendant l'apprentissage du VQA.

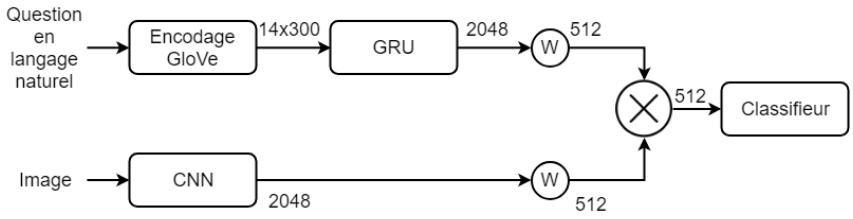

Figure 5.5 – Chaîne de traitement du modèle QRV de fusion multimodal utilisant des descripteurs visuels globaux.

Protocole 1 : Incrémenter les types de questions

Notre premier protocole propose un partitionnement du *dataset* VQAv2 en fonction des questions. L'idée est que le QRV est un problème multi-tâche où les questions indiquent au modèle quelle tâche il doit effectuer (e.g. dénombrer, classifier, comparer, décrire...). On peut donc, en contrôlant l'introduction des questions au modèle, proposer un protocole analogue à l'incrémental de tâches. Comme pour ce dernier, le modèle à conscience de la tâche à accomplir puisqu'il dispose de la question, cependant, l'espace des sorties n'est pas réduit.

Pour partitionner les questions de VQAv2 en fonction de la sémantique des questions, nous proposons de les clusteriser automatiquement. Pour cela, chaque question est encodée

en utilisant GloVe (Pennington et al., 2014). GloVe est un dictionnaire appris qui associe à chaque mot un vecteur de dimensions 300. L'espace de ces vecteurs est structuré de façon à ce que les opérations algébriques gardent un sens sémantique.

Chaque question est encodée en un unique vecteur de dimension 300 en moyennant sur l'ensemble des mots qui la compose. On considère l'union des questions d'entraînement et de validation (215 723 questions différentes). On obtient ainsi notre subdivision en tâches pour l'apprentissage ainsi que les différents sous-ensembles de validation spécifiques à chaque tâche. Dans nos expérimentations, la base VQAv2 est partitionnée en 20 tâches via un algorithme des K-moyennes utilisant la distance euclidienne.

Bien que simple, cette méthode permet d'obtenir des tâches sémantiquement homogènes. Les premiers mots de la question sont les plus déterminants lors du partitionnement. Par exemple, toutes les questions commençant par « Is there... » sont regroupée dans la même tâche, de même pour les questions débutant par « What is the... » ou « Does the [...] have... », ceci étant notamment dû à la fréquence prédominante de ces termes dans le corpus des questions. La figure 5.6 illustre le contenu des questions pour deux clusters obtenus avec les K-moyennes. On constate un déséquilibre du nombre d'exemples entre les tâches.

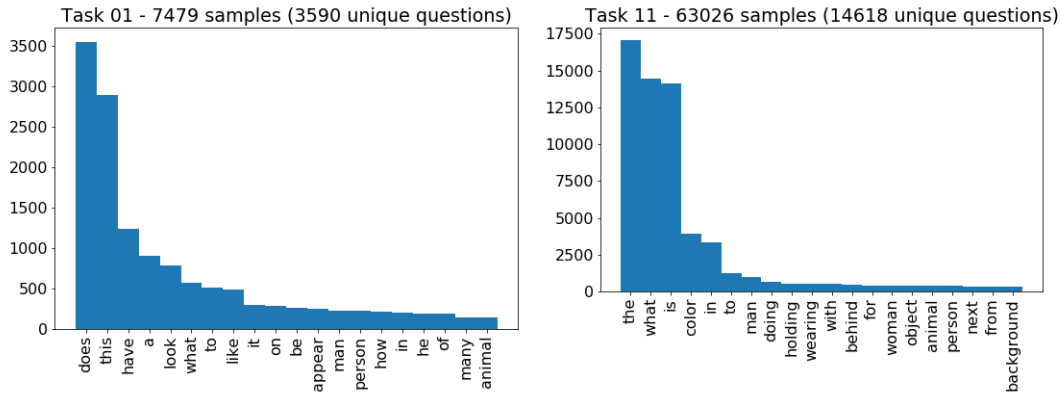

Figure 5.6 – Histogramme des mots contenus dans les questions des tâches 1 et 11.

Contrôler l'introduction des questions va naturellement organiser l'introduction des réponses et des images associées au sein de VQAv2.

On entraîne le modèle sur les 20 sous-ensembles de triplets. Pour lutter contre l'oubli catastrophique, on utilise la répétition avec mémoire. Le budget de cette dernière est fixé à $K = 4000$ et $K = 40\ 000$ triplets (I, Q, R), ce qui correspond respectivement à environ 1% et 10% de l'ensemble d'entraînement de VQAv2. On essaye aussi un modèle sans répétition et un avec une mémoire infini.

Au lieu d'équilibrer la mémoire par classe comme en incrémental de classes, on l'équilibre par tâche. À la fin de la session i, la mémoire contient $\frac{K}{i}$ triplets par tâche.

Résultats du protocole 1

Les résultats sont présentés en figure 5.7. L'algorithme des K-moyennes ayant été exécuté conjointement sur les questions de l'ensemble d'entraînement et celles de la validation, le *dataset* de validation est lui aussi partitionné en 20 sous-ensembles. Puisque le modèle traite le QRV avec une tête de classification, on peut donc calculer la précision incrémentale.

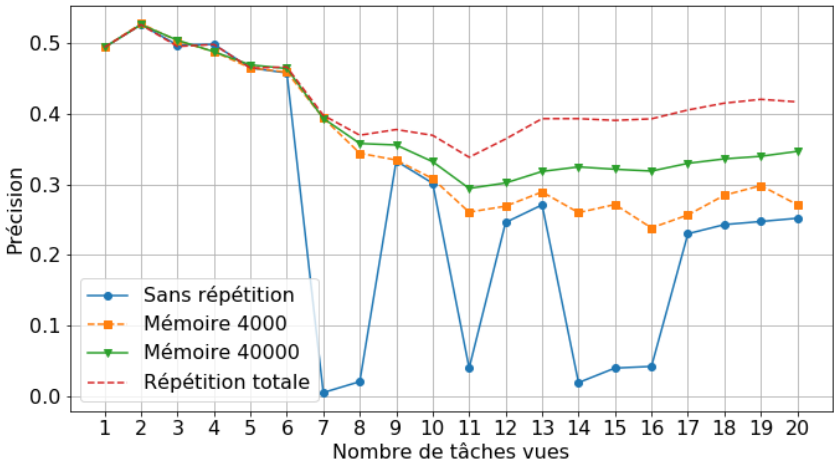

FIGURE 5.7 – Précision incrémentale obtenue sur un apprentissage de VQAv2 en 20 sessions.

Les résultats sont disponibles en figure 5.7. Le modèle sans répétition montre un phénomène d'oubli particulier. La performance globale du réseau se dégrade avec l'ajout de tâches, comme attendu dans ce contexte incrémental. Cependant, on constate que certaines tâches provoquent un oubli catastrophique notable. Le faible score à ces itérations prouve qu'en plus d'oublier les anciennes tâches, le réseau n'arrive pas à apprendre la nouvelle. La performance est cependant capable de remonter à la tâche suivante. Cela s'explique par la forte corrélation entre certaines tâches du problème de QRV et à notre partitionnement.

En effet, le VQAv2 est constitué à plus de 40% de questions fermées (classes « Yes » et « No »). La distribution des questions ouvertes est très hétérogène et une grande partie des réponses ont moins de dix exemples. On retrouve ce biais dans notre partitionnement avec les tâches 7, 8, 11 et 15 qui comportent principalement des questions « What... » alors que les 14 et 16 sont des tâches liées au dénombrement : « How many... ». Le reste des tâches comporte une plus grande proportion de questions fermées. C'est la cause principale des oscillations de la précision. Le réseau est capable d'apprendre rapidement la classification binaire *oui/non* et cela se répercute par un gain de performance sur l'ensemble des tâches contenant des questions fermées. Cependant, une tâche composée uniquement de classes rares provoque un oubli brutal de la connaissance acquise du modèle. La permutation de

l'ordre des tâches pendant l'apprentissage n'impacte pas le résultat observé : ce sont les mêmes tâches qui provoquent une chute des performances.

L'ajout d'une mémoire épisodique stabilise le comportement du réseau et augmente ses performances proportionnellement à sa taille.

La figure 5.8 montre l'évolution de la précision du modèle sur 4 tâches tout au long de son cycle de vie. La forte corrélation inter tâche, en particulier pour les questions fermées, est ici aussi illustrée par la précision élevée du réseau sur des tâches qu'il n'a pas encore rencontrées (par exemple, la tâche 6). D'autre part, la comparaison entre le modèle sans mémoire et celui avec une mémoire de taille 40 000 met clairement en valeur l'impact de la répétition. Les tâches sur lesquelles le réseau performe correctement voient leur précision se stabiliser et le modèle réussit à s'améliorer sur les tâches les plus difficiles dont les résultats étaient faibles dans le cas sans mémoire.

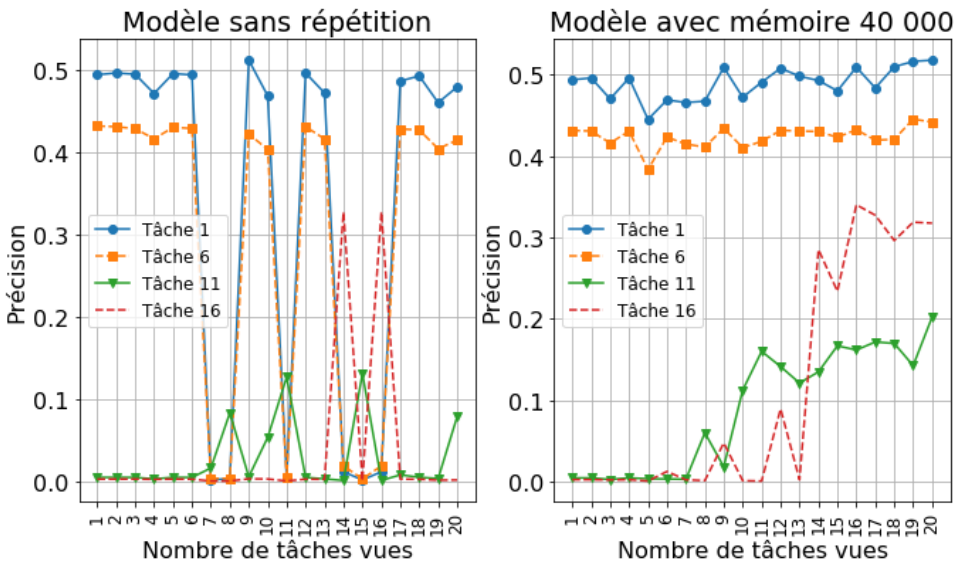

FIGURE 5.8 – Comparaison de l'évolution de la précision pour 4 tâches différentes entre un modèle sans mémoire (à gauche) et le modèle avec une mémoire de taille 40 000 (à droite). Chaque tâche voit sa précision mesurée sur son ensemble de validation propre.

Protocole 2 : Incrémenter la difficultés des questions

Une autre piste envisagée s'appuie sur le *dataset* GQA qui propose des graphes de raisonnement pour détailler le processus de résolution d'une question. Ainsi, HUDSON et MANNING, 2019 proposent plusieurs niveaux de difficultés pour les questions. Cette difficulté est quantifiée par le nombre d'étapes présentes dans le graphe de raisonnement.

Résultat en batch pour le Protocole 2

Ce protocole n'a pas été appliqué en incrémental. Cependant, afin d'étudier cette notion de difficulté, on a appris un modèle en batch sur GQA. Cette fois-ci, on utilise la version attentionnée complète de Anderson et al., 2018. Le *dataset* GQA proposant ses images directement sous formes de *représentations* extraites par un faster R-CNN optimisé sur COCO.

Le modèle final atteint une précision de 79%. On s'intéresse plus particulièrement à la précision obtenue pour chaque niveau de difficulté (cf. tableau 5.1). On observe une tendance opposée à nos attentes. Le modèle performe bien mieux sur les questions les plus « difficiles » que sur celles demandant peu de raisonnement.

La définition intuitive de « difficulté » au sens du nombre d'étapes de raisonnement est donc incorrecte du point de vue du modèle. Cela s'explique par le fait que, dans GQA, les questions demandant le plus d'étapes de raisonnement sont les questions les plus longues. Or plus une question est longue, plus elle aura tendance à décrire l'image et aider le modèle en limitant les réponses possibles. Inversement, une question demandant uniquement 2 étapes de raisonnement sera généralement plus courte et plus ouverte sur son espace de solutions.

Difficulté	2	3	4	5	6	7	8	9
Précision (%)	78.3	69.3	67.0	77.7	88.8	95.8	99.1	94.6

Table 5.1 – Précision mesurée sur GQA pour chaque niveau de difficulté. Un niveau de difficulté N réfère à l'ensemble des questions dont le graphe de raisonnement associé contient N étapes.

5.4.3 Conclusion des essais sur le QRV incrémental

La diversité et la complexité du problème QRV motivent l'intérêt de disposer d'un modèle capable d'apprendre de façon continue. Dans cette section, nous avons proposé une démarche d'évaluation incrémentale reposant sur un partitionnement sémantique des questions. Cette méthode permet d'organiser le flux de données d'apprentissages en tâches qui sont montrées au modèle de manière séquentielle.

L'étude réalisée ici sur la base VQAv2 permet d'illustrer les spécificités du QRV par rapport aux problèmes de classification usuellement utilisés dans la littérature de l'apprentissage continu. Les différents niveaux de corrélation entre tâches ont un impact important sur les performances. Il peut être positif — certaines tâches profitant des apprentissages passés —, mais également fortement négatif et provoquer un oubli catastrophique global sur les tâches anciennes. Des méthodes simples exploitant une mémoire épisodique permettent de limiter ces phénomènes. D'autre part, nos résultats indiquent que le niveau

de difficulté des tâches n'est pas constant pour le réseau, ceci étant possiblement dû aux biais du VQAv2 (forte occurrence de certaines questions comme les questions fermées et très peu d'exemples pour les questions ouvertes plus rares).

La notion de difficulté pour le QRV est difficilement quantifiable. Une question difficile pour l'humain pourra être au contraire très simple pour un modèle artificiel. Organiser manuellement la connaissance pour guider l'apprentissage du QRV est loin d'être immédiat. Il pourrait néanmoins être intéressant de se rapprocher de la littérature du *curriculum learning* qui propose des solutions pour que le modèle choisisse lui-même quelles connaissances il doit apprendre lors de ses phases d'apprentissage.

Le QRV reste encore un problème où les méthodes existantes ont des performances médiocres. Chercher à le combiner avec un paradigme d'apprentissage incrémental, encore loin d'être compétitif avec l'entraînement batch, est ambitieux et semble augmenter la complexité d'un problème déjà difficile à résoudre. Les quelques expérimentations effectuées donnent des résultats peu satisfaisants et il est difficile d'en identifier les causes. Les différents problèmes connus du QRV se combinent avec ceux de l'apprentissage continu et s'amplifient entre eux.

5.5 Perspective : La classification multi-label (CML)

Dans notre objectif d'adapter l'apprentissage incrémental sur des applications de vision plus complexe, nous nous sommes très rapidement retrouvés limités par la complexité du QRV incrémental. De ce fait, l'alternative est de nous intéresser à une application plus accessible : celle de la classification multi-label (W. Liu et al., 2020).

Au travers de cette section, nous proposons une ébauche d'une version incrémentale de la classification multi-label. En nous appuyant sur notre vision du problème, nous justifions son intérêt pour l'étude de l'apprentissage continu, notamment en présentant les nouvelles problématiques inhérentes au multi-labels et absente de l'incrémental de classe standard.

5.5.1 De la classification mono-label a la classification multi-label

La classification multi-label est l'extension directe de la reconnaissance d'objet. Cette dernière, dans sa définition la plus courante, est dite mono-label au sens où l'on attend une catégorie unique en sortie du modèle. En conséquence, ce type de problèmes utilise des images de structure particulière : un seul objet d'intérêt au premier plan et bien cadré. Ce type d'image n'est pas représentatif des images naturelles. Une scène est généralement une

composition sur différents plans de diverses entités. La classification multi-label cherche à lister toutes les catégories identifiables dans l'image. Ces catégories peuvent aussi bien représenter des objets que des attributs.

La CML est de ce fait considérée comme un problème de vision plus complexe, car elle demande au modèle d'acquérir une compréhension étendue de l'image afin de la décrire en listant tout ce qu'elle contient. C'est pour nous une piste à développer, puisque dans l'idée, un modèle capable de comprendre et analyser le contenu d'une image pour la CML peut servir de fondation à l'encodeur visuel d'un système de description d'image ou de QRV.

5.5.2 Les données de la CML

Les données en CML se présentent sous la forme de paire image et vérité terrain, comme pour la classification mono-label. La principale différence provient du format de la VT.

La VT est un vecteur de dimension égale au nombre de catégories du problème. Pour chaque indice, le vecteur contient un « 1 » si la classe associée est représentée dans l'image. On dit alors que le label est « positif ».

Si une classe est absente de l'image, son label est « négatif ». Il existe plusieurs manières de traiter les labels négatifs. Si on suppose que l'annotation des labels positifs est exhaustive, on peut par déduction attribuer un label négatif à toutes les catégories restantes. On a alors une annotation binaire. Un vecteur de VT est un vecteur de « 1 » et « 0 ».

La contrainte de cette approche est d'avoir des annotations exhaustives, ou denses. Si l'espace des classes est grand et que les images sont complexes, la création manuelle des vecteurs VT devient coûteuse et sensible aux erreurs des annotateurs humains. De ce fait, les données utilisant un étiquetage binaire sont souvent bruitées : les VT contiennent des labels négatifs pour des éléments manqués lors de l'annotation.

Pour contrer ce phénomène, certaines approches (Durand et al., 2019 ; M.-K. Xie & Huang, 2018) considèrent une annotation partielle. Le vecteur de VT peut alors contenir 3 valeurs : « 1 » pour un label positif, « -1 » pour un label négatif et « 0 » pour un label inconnu. Le travail de l'annotateur devient plus exigeant puisque pour chaque classe, il doit spécifier manuellement si le label est positif ou négatif. Tant qu'une classe n'a pas été traitée, on la laisse à « 0 » pour indiquer qu'on ne sait pas si la classe est observable dans l'image ou non.

5.5.3 Enjeux et problématiques de la CML

La CML induit de nouvelles problématiques par rapport à la classification mono-label. On retrouve des problématiques communes au QRV comme le fait que les distributions de données ne sont pas i.i.d., les corrélations entre les classes ou encore la difficulté à collecter des annotations exhaustives pour la tâche. Résoudre ces problèmes pour la CML est de ce fait une contribution utile pour progresser ensuite vers le QRV.

Distribution à longue traîne : On retrouve ici une problématique similaire au QRV. Un ensemble de scènes naturelles ne permet pas de couvrir uniformément les différentes classes. De ce fait, certaines classes seront sur-représentées alors que d'autres n'auront que quelques exemples. On illustre l'histogramme des occurrences de labels présents dans l'ensemble d'entraînement de COCO en figure 5.9. Plus de la moitié des images (45 174) contiennent au moins une personne. À l'opposé, les classes « toaster » et « hair dryer » apparaissent respectivement dans 151 et 128 images.

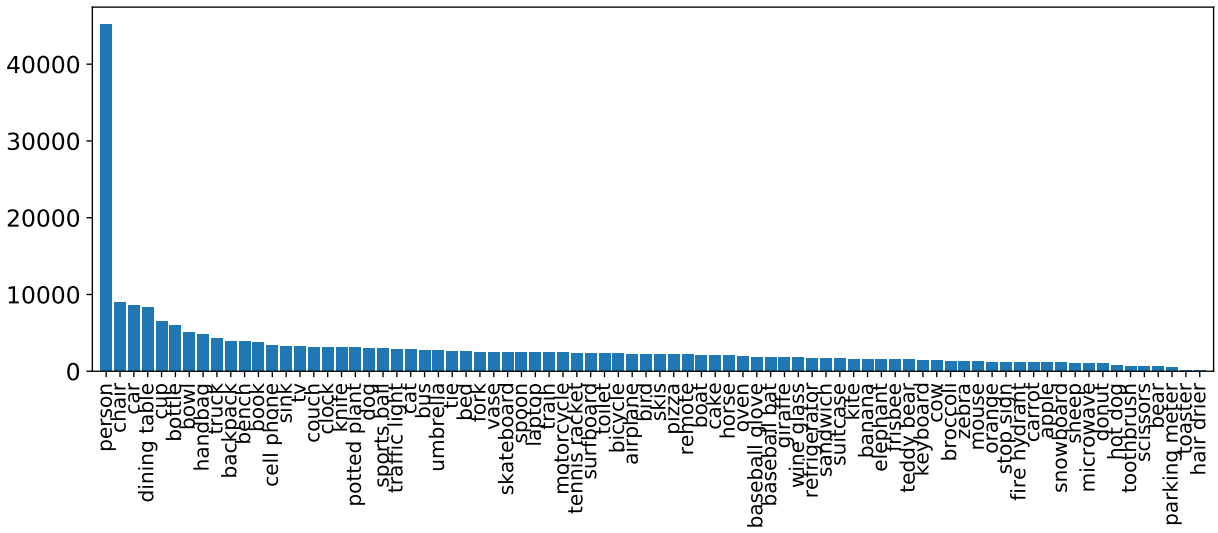

FIGURE 5.9 – Histogrammes des labels observables dans les 82 081 images de l'ensemble d'entraînement de COCO.

Annotations bruitées ou partielles : Le modèle doit être capable de traiter les différents cas d'annotations possibles, qu'elles soient bruitées ou incomplètes. Dans le cas des annotations bruitées, on peut souvent considérer que les erreurs d'étiquetage sont négligeables sur l'ensemble du *dataset*. Pour le cas des annotations partielles, le modèle doit être capable d'apprendre uniquement avec des vecteurs de labels parcimonieux.

Corrélations entre les classes : Certaines classes ont un fort taux de co-occurrences. Ces classes sont corrélées. C'est-à-dire que si l'une d'elle est détectée, il est fort probable que l'autre soit aussi présente. La figure 5.10 illustre les catégories les plus présentes dans les images de COCO contenant « TV » ou « baseball glove ». Les corrélations, dans le cas de d'images naturelles, sont liées au contexte de la scène. La classe « TV » indique généralement que l'on se trouve dans une pièce d'une maison. Cette classe est donc généralement corrélée avec les autres catégories de mobilier ou d'objet d'intérieur. « baseball glove » est presque tout le temps présent avec un individu. Les autres classes liées au baseball comme « baseball bat » ou « sports ball » ont de fortes chances d'être aussi présentes.

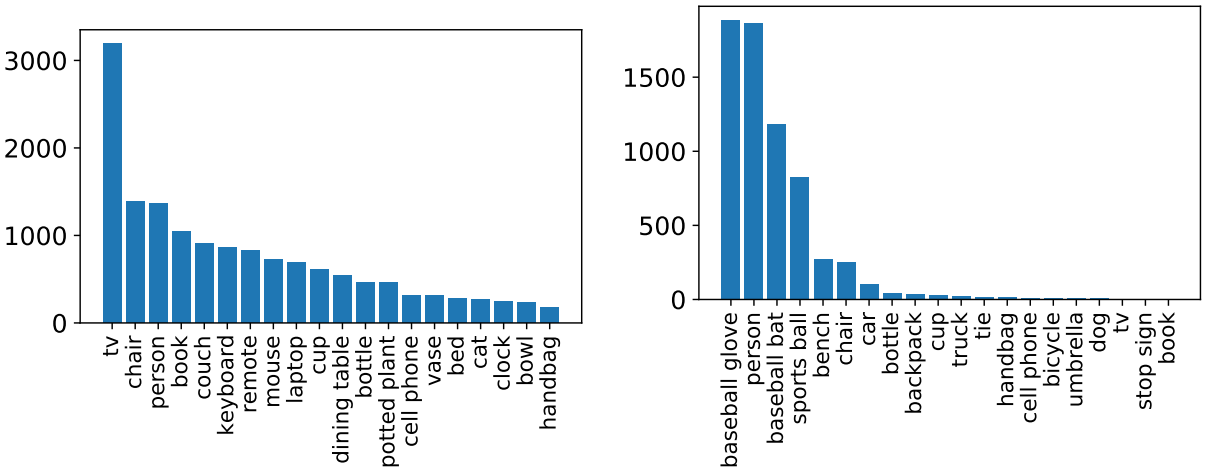

FIGURE 5.10 – Histogrammes des 20 catégories avec le plus d'occurrences dans les images contenant la classe « TV » (à gauche) ou la classe « baseball glove » (à droite).

Ces corrélations sont utiles pour le modèle. La présence de certaines classes va l'encourager à vérifier si les classes corrélées sont aussi présentes. Cependant, un modèle trop dépendant des corrélations risque de détecter systématiquement toutes les classes corrélées entre elles y compris lorsqu'une seule est vraiment présente.

5.6 La classification multi-label incrémentale

5.6.1 Les potentielles applications de la classification multi-label incrémentale

Dans cette section, nous soulignons les possibles intérêts pour de la classification multi-label incrémentale. Nous choisissons ici de fonder notre argumentation sur de potentielles applications concrètes.

La classification multi-label consiste à décrire le contenu d'une donnée par une liste de labels, aussi appelé attributs, *tags*, mots-clés ou étiquettes. Cette description est no-

tamment utilisée pour organiser et classer des grandes bases de données. On retrouve notamment beaucoup d'applications de la CML pour le traitement de documents textuels ou de pages internet.

Plus récemment, on à l'exemple les applications web comme les réseaux sociaux, les plateformes de partage de contenu ou les banques de données multimédia crowdsourcées proposent elles aussi un tel formatage pour décrire leur contenu. Proposer la description la plus fine possible est souvent la première couche des algorithmes de traitement des données (recherche, recommandations, filtrages...). Le contenu de ces sites étant mis à jour en permanence, de nouvelles données arrivent en permanence. De nouveaux labels sont aussi ajoutés en continu, les sites étant souvent conçus pour compter sur la participation des utilisateurs à l'annotation.

Lorsqu'on voit la taille d'un *dataset* comme Open Images (9M d'images haute résolution), on aborde des échelles où devoir ré-entraîner un modèle en batch à chaque fois que l'on cherche à ajouter des classes devient excessivement long et coûteux. L'apprentissage incrémental est donc une considération importante pour traiter ce type de problèmes.

5.6.2 Émuler un incrémental multi-label

En classification multi-label, les données entrantes sont de nouveaux labels ou de nouvelles images. Un nouveau label peut être une mise à jour de l'annotation d'une image déjà présente dans l'ensemble d'entraînement ou arriver avec une nouvelle image. Un nouveau label peut concerner une ancienne classe mais à une probabilité d'appartenir à une catégorie jamais vue auparavant.

Le format des labels

Selon le format du vecteur de labels utilisé (cf. sec. 5.5.2), deux configurations sont possibles :

— L'approche binaire, label négatif par défaut. On demande uniquement des labels positifs de la part de l'annotateur. Tous les labels non positifs sont par défaut négatifs lors de l'apprentissage. Un nouveau label entrant va soit introduire une nouvelle classe, soit mettre à jour un label déjà existant de négatif à positif. Le modèle doit gérer des labels bruités qui peuvent être amenés à changer. Un changement de label entre deux sessions peut être contradictoire en apprentissage continu. On peut par exemple avoir un terme de distillation où l'ancien modèle à appris la même image avec le label négatif.

— L'approche label inconnu par défaut. Lorsque les labels sont au choix « 1 » positif, « -1 » négatif ou « 0 » inconnu, il est plus cohérent de laisser les labels n'ayant reçu

aucun signal à 0. Cela implique un modèle pouvant apprendre avec des vecteurs de vérité terrain parcimonieux. Pour compléter ces annotations, il faut que pour chaque label manquant (chaque 0), un utilisateur spécifie si la classe est présente (+1) ou non (-1) dans l'image. La complexité du processus d'annotation est de ce fait alourdie car contrairement à l'approche binaire, les labels négatifs ne sont pas inférés automatiquement.

Restreindre ou non l'accès aux anciens labels

Laisser la possibilité de mettre à jour les annotations implique qu'une même image peut être revue à plusieurs instant de l'apprentissage.

Selon l'application, si les annotations sont stockées au fur et à mesure, il est possible de récupérer l'ancien vecteur d'annotation pour le mettre à jour. Le modèle, lorsqu'il apprend l'image, utilise l'ensemble des labels passés et présents.

Si les anciennes annotations ne sont plus disponibles, le modèle n'a accès qu'aux nouveaux labels de l'image introduits lors de la session actuelle. Il la traite de la même façon qu'une nouvelle image avec de nouvelles annotations.

Théoriquement, un modèle idéal n'aura pas besoin de ré-accéder aux anciens labels d'une image déjà vue puisqu'il est censé pouvoir les re-générer lui-même.

Apprentissage en *streaming*

Sans contrainte, n'importe quel label peut apparaître à n'importe quel moment dans le flux de données d'entraînement. On peut donc aussi bien revoir un label qui concerne une classe connue ou un label qui introduit une toute nouvelle catégorie. Ce type d'approche est appelé apprentissage en « *streaming* » (Hayes et al., 2018 ; Hayes et al., 2020).

L'objectif d'un modèle *streaming* est de traiter ces données qui arrivent au fur et à mesure sans contrainte. Cette configuration fait que lorsqu'une nouvelle classe est soumise au modèle, il n'y a aucune garantie sur le nombre d'exemples disponibles. Cela rajoute une difficulté d'apprentissage des nouvelles classes lorsque peu d'exemples sont présentés. L'apprentissage de la nouvelle classe peut nécessiter d'être retardé en attendant d'accumuler suffisamment d'échantillons.

Une configuration se rapprochant de l'incrémental de classes nécessiterait d'imposer des labels strictement disjoints entre les sessions. Une classe ne pourrait donc être apprise uniquement lors de sa session attribuée. Cependant, cela n'empêche pas de revoir les images à de multiples reprises et donc de compléter l'annotation via pseudo-labeling de l'ancien modèle par exemple.

5.6.3 CML incrémental : une définition à affiner

On a présenté ici une liste de points clés à considérer pour définir un apprentissage incrémental adapté à la classification multi-label. Cette liste est certainement loin d'être exhaustive mais donne déjà une première idée de la complexité à définir un protocole cohérent et pertinent. On diffère des études réalisées en incrémental de classe pour la reconnaissance d'objet où un scénario complet se contrôle via un unique paramètre : le pas d'incrément qui permet de partitionner le *dataset* en sous-ensemble à introduire successivement.

Un multi-label incrémental nécessite d'être défini via un ensemble de paramètres tels que le taux d'apparition de nouveaux labels, le taux de mise à jour des labels associés aux images déjà vues, l'utilisation ou non du label « inconnu », l'accès aux anciens labels... Rien que pour contrôler l'apparition des nouvelles classes, les conditions sont différentes de l'incrémental de classe. Du fait de la distribution à longue traîne, il serait logique que les classes peu fréquentes aient une probabilité plus faible d'apparaître comparé aux classes avec un fort taux d'occurrence. On peut aussi considérer qu'un modèle connaissant un très grand nombre de classes ait moins de chance de rencontrer des labels inconnus.

Le CML incrémental est donc un protocole ambitieux dont nous avons pu exposer une partie des enjeux ici. Sa définition reste à affiner et d'autres facteurs, comme les métriques d'évaluation, sont à développer avant de pouvoir proposer une *baseline* exploitable.

5.7 Conclusion

Au cours de ce chapitre, nous avons illustré les difficultés rencontrées pour adapter des problèmes de vision autres que la reconnaissance d'objet à un contexte incrémental. Dès l'étape de définition, on se rend compte que proposer une formulation pertinente de l'incrémental pour ces applications est délicat.

Premièrement, les problématiques propres à ces applications viennent s'ajouter à celle de l'incrémental. Ensuite, les jeux de données limités pour traiter ces tâches rendent difficile la définition de scénarios pertinents. En effet, pour la reconnaissance d'objet, les protocoles d'incrémental de classes ou d'incrémental de tâches définissent l'apprentissage incrémental comme une succession de sessions dont les données sont indépendantes. Cette contrainte est rendue possible lorsqu'on a accès à des *datasets* identiquement et indépendamment distribuées. Pour des applications plus complexes, les corrélations entre les données ainsi que les *datasets* limités, empêchent le respect de cette définition.

De ce fait, pour des applications QRV ou de classification multi-label, il semble favorable de s'éloigner des scénarios très artificiels conçus pour simplifier et standardiser l'évaluation. Deux configurations plus réalistes de l'apprentissage continu sont alors par-

ticulièrement intéressantes pour les études futures :

— **Une approche *streaming*** : on ne cherche pas à contraindre le flux de données entrant. Le modèle doit exploiter les données comme elles viennent. Celles-ci peuvent aussi bien introduire une nouvelle connaissance (e.g. une nouvelle catégorie en CML) ou venir renforcer une connaissance déjà établie. L'objectif premier du *streaming* est l'optimisation computationnelle. L'apprentissage à chaque étape est alors fait uniquement sur les nouvelles données.

— **Une approche *curriculum*** : l'objectif est de trouver un *curriculum* d'apprentissage optimisé. C'est-à-dire ordonner de manière séquentielle la connaissance à apprendre. L'idée sous-jacente est que pour une tâche très complexe, un apprentissage séquentiel logique et progressif peut être plus efficace et optimisé qu'un apprentissage en batch sur la globalité des connaissances.

Chacun de ces deux scénarios s'étend à de nouveaux sous-domaines de l'apprentissage continu. De ce fait, ils restent à définir formellement. Il est nécessaire d'identifier les paramètres clés dans la définition d'un protocole expérimental ainsi que de discuter des métriques pertinentes pour l'évaluation.

5.8 Publication

Lechat, A., Herbin, S. & Jurie, F. (2019). Adaptation du problème de questions-réponses visuelles à un contexte d'apprentissage continu. *27ème Colloque du GRETSI, 2019*, (Oral)

Chapitre 6

Conclusion

Nous revenons au cours de cette conclusion sur les principales thématiques et contributions abordées au fil de ce manuscrit. Nous avons étudié l'apprentissage incrémental en nous intéressant notamment aux problématiques que l'on peut résoudre par des solutions en lien avec l'apprentissage des représentations visuelles non-supervisées. Nos contributions ont été conçues et expérimentalement validées sur le scénario de l'incrémental de classes qui propose un schéma continu pour la reconnaissance d'objets. Nous présentons aussi en perspective les avantages qu'un cadre incrémental pourrait apporter à certaines applications plus complexes.

6.1 Le rôle des représentations dans l'apprentissage incrémental

Un modèle, dans le cas d'un scénario d'incrémental de classes, à deux objectifs principaux : apprendre les nouvelles classes qu'il rencontre et rester compétitif sur les anciennes. Lors d'un tel processus, les représentations, modélisées par les paramètres du réseau de neurones, sont directement concernées par le dilemme de stabilité-plasticité.

L'objectif d'un apprentissage des représentations pour l'incrémental est double.

D'abord, apprendre les représentations du DNN via une tâche annexe permet de découpler l'apprentissage de l'encodeur de la tâche incrémentale. Un encodeur ne dépendant plus des sessions incrémentales sera immunisé face à l'oubli. Si le découplage total n'est pas possible, de meilleures représentations permettent quand même de limiter les besoins en mise à jour des paramètres pour assimiler les nouvelles connaissances, favorisant la stabilité des poids.

Ensuite, un moyen immédiat d'augmenter la plasticité, i.e. le potentiel d'apprentissage d'un modèle est d'augmenter la taille des architectures. Un modèle plus grand requiert plus de puissance de calcul pour l'entraînement, mais requiert aussi une quantité suffisante de données pour coordonner l'apprentissage de tous les paramètres. Par définition, un schéma incrémental contraint le modèle à apprendre sur des ensembles limités d'exemples. De ce fait, le schéma incrémental est un facteur limitant inhérent à la taille des modèles

utilisables. Certains cadres d'application ont besoin de préparer leurs représentations avec une phase de pré-entraînement ou bien par un apprentissage hors ligne sur un grand nombre de classes.

Ces deux constats ont motivé le développement de méthodes incrémentales axées sur les représentations des modèles.

6.2 L'hypothèse de l'accès à des données non-annotées

L'apprentissage continu considère des scénarios où l'accès aux données annotées décrivant la tâche à apprendre est restreint mais progressif.

Pour certains domaines d'applications, se sont les données elles-mêmes qui sont rares. Mais pour bien d'autres, les données brutes sont abondantes et le facteur limitant provient du processus d'annotation.

Il est donc intéressant de considérer les cas d'apprentissage incrémental où, en parallèle des données annotées présentes dans le flux entrant, le modèle dispose d'un accès à une grande quantité de données secondaires sans labels. Nous avons de ce fait proposé d'intégrer cette hypothèse dans le scénario de l'incrémental de classes tout en prenant en considération les différentes possibilités sur le contenu des données non-annotées.

Une telle source externe permet l'application des solutions auto-supervisées qui constituent désormais l'état-de-l'art pour l'apprentissage des représentations visuelles.

6.3 L'apprentissage incrémental semi-supervisé

Une solution immédiate pour traiter les données sans étiquettes est le pré-entraînement. Cependant une telle approche bénéficie surtout aux premières phases du processus incrémental et voit son impact atténué au fil des multiples sessions d'optimisation.

Notre hypothèse suppose la disponibilité de données annexes à tout moment. De ce fait, nous optons de ne pas limiter leur utilisation à un rôle d'initialisation. À la place, en explorant la piste de la semi-supervision, nous développons des méthodes capables d'exploiter simultanément les données annotées provenant du continuum incrémental et les échantillons extérieurs.

Nous avons défini et expérimenté deux algorithmes : l'Autoencodeur adverse (AAE) incrémental et le *Pseudo-Labeling for Class incremental Learning* (PLCiL).

L'AAE a permis une prise en main du problème avec la tâche auto-supervisée la plus immédiate possible : la reconstruction. En utilisant cette démarche nous avons montré que l'auto-supervision introduit à la fois une amélioration des représentations mais aussi une régularisation innée du processus incrémental qui se déroule en parallèle. Nous avons aussi

montré qu'un classifieur génératif avec un prior connu permet de contrôler l'espace latent dans lequel s'effectue la décision. L'AAE présente cependant des limites, principalement conséquences de son processus d'entraînement complexe.

Avec le PLCiL, nous avons proposé une solution plus simple, compatible avec les *baselines* classique d'apprentissage avec répétition et distillation de connaissances. L'utilisation de la régularisation de consistance via *pseudo-labeling* permet d'apprendre en continu des représentations spécifiquement optimisées pour de la classification. Nous avons introduit également une nouvelle distillation de connaissance auto-supervisée qui garantit une régularisation efficace contre l'oubli catastrophique. Le PLCiL surpasse les approches supervisées sur les *benchmarks* de vision courant, y compris dans des régimes où l'on restreint les annotations disponibles.

6.4 L'apprentissage incrémental, un schéma adapté pour apprendre des tâches complexes

L'apprentissage incrémental, tel qu'il est défini pour les machines, s'inspire du domaine de la biologie. L'apprentissage d'un être humain sur une tâche complexe consiste en une suite logique d'acquisition des compétences. Cette séquence, appelée *curriculum*, est organisée de façon à d'abord introduire les bases nécessaires avant d'introduire les concepts et les exemples plus complexes.

Pour certaines applications de visions comme les questions-réponses visuelles (QRV) par exemple, le domaine et les compétences qu'elles impliquent sont tellement étendues qu'il semble inenvisageable de construire un *dataset* suffisamment exhaustif pour un apprentissage en *batch* hors ligne.

De ce fait, le développement du paradigme incrémental semble crucial pour pouvoir traiter ces problèmes. L'objectif peut être simplement de pouvoir apprendre une tâche complexe au rythme du processus de collecte des données sans avoir à devoir recommencer l'apprentissage à zéro. Mais il peut être de décomposer l'apprentissage en un *curriculum* d'apprentissage.

Bibliographie

Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R. & Bejnordi, B. E. (2020). Conditional Channel Gated Networks for Task-Aware Continual Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 3930-3939 (cf. p. 35).

Ahn, H. & Moon, T. (2020). A Simple Class Decision Balancing for Incremental Learning. *CoRR, abs/2003.13947* (cf. p. 32, 34).

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M. & Tuytelaars, T. (2018). Memory Aware Synapses : Learning What (not) to Forget. *European Conference on Computer Vision, ECCV* (cf. p. 35).

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S. & Zhang, L. (2018). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 6077-6086 (cf. p. 121, 122, 126).

Asano, Y. M., Rupprecht, C. & Vedaldi, A. (2020). Self-labelling via simultaneous clustering and representation learning. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (cf. p. 41).

Bengio, Y., Courville, A. C. & Vincent, P. (2013). Representation Learning : A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell., 35 8*, 1798-1828 (cf. p. 37).

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H. & Raffel, C. (2020). ReMixMatch : Semi-Supervised Learning with Distribution Matching and Augmentation Anchoring. *8th International Conference on Learning Representations, ICLR* (cf. p. 42, 86-89, 93).

Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A. & Raffel, C. (2019). MixMatch : A Holistic Approach to Semi-Supervised Learning. *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 5050-5060 (cf. p. 42, 43, 86, 93).

Buzzega, P., Boschini, M., Porrello, A., Abati, D. & Calderara, S. (2020). Dark Experience for General Continual Learning : a Strong, Simple Baseline. *Advances*

in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual. (Cf. p. 28).

Caron, M., Bojanowski, P., Joulin, A. & Douze, M. (2018). Deep Clustering for Unsupervised Learning of Visual Features. *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, *11218*, 139-156 (cf. p. 41).

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P. & Joulin, A. (2020). Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems, NeurIPS* (cf. p. 85).

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C. & Alahari, K. (2018). End-to-End Incremental Learning. *European Conference on Computer Vision, ECCV* (cf. p. 28, 32, 96, 97).

Chaudhry, A., Dokania, P. K., Ajanthan, T. & Torr, P. H. S. (2018). Riemannian Walk for Incremental Learning : Understanding Forgetting and Intransigence. *European Conference on Computer Vision, ECCV* (cf. p. 35).

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. E. (2020). A Simple Framework for Contrastive Learning of Visual Representations. *37th International Conference on Machine Learning, ICML* (cf. p. 40, 85).

Chen, T., Kornblith, S., Swersky, K., Norouzi, M. & Hinton, G. E. (2020). Big Self-Supervised Models are Strong Semi-Supervised Learners. *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* (Cf. p. 40).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. & Abbeel, P. (2016). InfoGAN : Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *Advances in Neural Information Processing Systems 29 : Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (p. 2172-2180). (Cf. p. 38).

Coates, A., Ng, A. Y. & Lee, H. (2011). An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, *15*, 215-223 (cf. p. 47, 76).

Cohen, G., Afshar, S., Tapson, J. & van Schaik, A. (2017). EMNIST : Extending MNIST to handwritten letters. *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, 2921-2926 (cf. p. 45).

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. & Bharath, A. A. (2018). Generative Adversarial Networks : An Overview. *IEEE Signal Process. Mag.*, *35 1*, 53-65 (cf. p. 39).

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V. & Le, Q. V. (2019). AutoAugment : Learning Augmentation Strategies From Data. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 113-123 (cf. p. 87).

Cubuk, E. D., Zoph, B., Shlens, J. & Le, Q. (2020). RandAugment : Practical Automated Data Augmentation with a Reduced Search Space. *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. (Cf. p. 88).

Cuturi, M. (2013). Sinkhorn Distances : Lightspeed Computation of Optimal Transport. *Advances in Neural Information Processing Systems 26 : 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States* (p. 2292-2300). (Cf. p. 41).

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M. F., Parikh, D. & Batra, D. (2017). Visual Dialog. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 1080-1089 (cf. p. 117).

Donahue, J., Krähenbühl, P. & Darrell, T. (2017). Adversarial Feature Learning. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cf. p. 85).

Donahue, J. & Simonyan, K. (2019). Large Scale Adversarial Representation Learning. *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (p. 10541-10551). (Cf. p. 38).

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M. A. & Brox, T. (2014). Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. *Advances in Neural Information Processing Systems 27 : Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (p. 766-774). (Cf. p. 40).

Douillard, A., Cord, M., Ollion, C., Robert, T. & Valle, E. (2020). PODNet : Pooled Outputs Distillation for Small-Tasks Incremental Learning. *European Conference on Computer Vision, ECCV* (cf. p. 34, 54).

Durand, T., Mehrasa, N. & Mori, G. (2019). Learning a Deep ConvNet for Multi-Label Classification With Partial Labels. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 647-657 (cf. p. 128).

Erhan, D., Bengio, Y., Courville, A. C., Manzagol, P.-A., Vincent, P. & Bengio, S. (2010). Why Does Unsupervised Pre-training Help Deep Learning ? *J. Mach. Learn. Res.*, *11*, 625-660 (cf. p. 38).

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, *34*, 128-135 (cf. p. 19, 27).

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P. & Cord, M. (2020). Learning Representations by Predicting Bags of Visual Words. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 6926-6936 (cf. p. 41).

Gidaris, S., Bursuc, A., Puy, G., Komodakis, N., Cord, M. & Pérez, P. (2021). OBoW : Online Bag-of-Visual-Words Generation for Self-Supervised Learning. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 6830-6840 (cf. p. 85, 86).

Gidaris, S., Singh, P. & Komodakis, N. (2018). Unsupervised Representation Learning by Predicting Image Rotations. *6th International Conference on Learning Representations, ICLR* (cf. p. 38, 99).

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A. & Bengio, Y. (2013). An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv :1312.6211* (cf. p. 19, 27).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C. & Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27 : Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (p. 2672-2680). (Cf. p. 38, 39).

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D. & Parikh, D. (2017). Making the V in VQA Matter : Elevating the Role of Image Understanding in Visual Question Answering. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 6325-6334 (cf. p. 116, 117).

Hayes, T. L., Cahill, N. D. & Kanan, C. (2018). Memory Efficient Experience Replay for Streaming Learning [arXiv : 1809.05922]. *arXiv :1809.05922 [cs, stat]* (cf. p. 132).

Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M. & Kanan, C. (2020). REMIND Your Neural Network to Prevent Catastrophic Forgetting. *European Conference on Computer Vision, ECCV* (cf. p. 28, 132).

He, K., Fan, H., Wu, Y., Xie, S. & Girshick, R. B. (2020). Momentum Contrast for Unsupervised Visual Representation Learning. *2020 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, 9726-9735 (cf. p. 40).

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770-778 (cf. p. 23, 61, 96).

Hénaff, O. J. (2020). Data-Efficient Image Recognition with Contrastive Predictive Coding. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, *119*, 4182-4192 (cf. p. 40).

Hinton, G. E., Vinyals, O. & Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv :1503.02531* (cf. p. 29, 94).

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A. & Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization. *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (cf. p. 40).

Hou, S., Pan, X., Loy, C. C., Wang, Z. & Lin, D. (2019). Learning a Unified Classifier Incrementally via Rebalancing. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (cf. p. 32, 34).

Hudson, D. A. & Manning, C. D. (2019). GQA : A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 6700-6709 (cf. p. 117, 118, 125).

Hung, S. C. Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M. & Chen, C.-S. (2019). Compacting, Picking and Growing for Unforgetting Continual Learning. *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (p. 13647-13657). (Cf. p. 35, 36).

Iscen, A., Zhang, J., Lazebnik, S. & Schmid, C. (2020). Memory-Efficient Incremental Learning Through Feature Adaptation. *European Conference on Computer Vision, ECCV* (cf. p. 28).

Javed, K. & Shafait, F. (2018). Revisiting Distillation and Incremental Classifier Learning. *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part VI* (p. 3-17). (Cf. p. 28).

Jing, L. & Tian, Y. (2019). Self-supervised Visual Feature Learning with Deep Neural Networks : A Survey. *CoRR*, *abs/1902.06162* (cf. p. 38).

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L. & Girshick, R. B. (2017). CLEVR : A Diagnostic Dataset for Compositional Lan-

guage and Elementary Visual Reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 1988-1997 (cf. p. 117).

Kemker, R., McClure, M., Abitino, A., Hayes, T. L. & Kanan, C. (2018). Measuring Catastrophic Forgetting in Neural Networks. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (p. 3390-3398). (Cf. p. 25).

Kingma, D. P. & Welling, M. (2014). Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. (Cf. p. 39).

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, *114 13*, 3521-3526 (cf. p. 28, 35).

Kolesnikov, A., Zhai, X. & Beyer, L. (2019). Revisiting Self-Supervised Visual Representation Learning. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 1920-1929 (cf. p. 39).

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S. & Fei-Fei, L. (2017). Visual Genome : Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *Int. J. Comput. Vis.*, *123 1*, 32-73 (cf. p. 117).

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images* (rapp. tech.). (Cf. p. 32, 46).

Laine, S. & Aila, T. (2017). Temporal Ensembling for Semi-Supervised Learning. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cf. p. 42, 93).

Larsson, G., Maire, M. & Shakhnarovich, G. (2016). Learning Representations for Automatic Colorization. *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV* (p. 577-593). (Cf. p. 38).

Lechat, A., Herbin, S. & Jurie, F. (2019). Adaptation du problème de questions-réponses visuelles à un contexte d'apprentissage continu. *27ème Colloque du GRETSI, 2019*, (Oral) (cf. p. 16, 134).

Lechat, A., Herbin, S. & Jurie, F. (2020). Semi-Supervised Class Incremental Learning. *25th International Conference on Pattern Recognition (ICPR), 2020*, 10383-10389, (Poster) (cf. p. 16, 81).

Lechat, A., Herbin, S. & Jurie, F. (2021). Pseudo-Labeling for Class Incremental Learning. *Proceedings of the British Machine Vision Conference, BMVC, 2021*, (Oral) (cf. p. 16, 109).

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-Based Learning applied to Document Recognition. *86 11*, 2278-2323 (cf. p. 45).

Lee, K., Lee, K., Shin, J. & Lee, H. (2019). Overcoming Catastrophic Forgetting With Unlabeled Data in the Wild. *International Conference on Computer Vision, ICCV* (cf. p. 32, 94).

Lesort, T., Caselles-Dupré, H., Ortiz, M. G., Stoian, A. & Filliat, D. (2019). Generative Models from the perspective of Continual Learning. *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, 1-8 (cf. p. 29).

Li, X., Zhou, Y., Wu, T., Socher, R. & Xiong, C. (2019). Learn to Grow : A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (p. 3925-3934). (Cf. p. 35).

Li, Z. & Hoiem, D. (2017). Learning without Forgetting. *IEEE transactions on pattern analysis and machine intelligence*, *40 12*, 2935-2947 (cf. p. 31, 77, 78, 92, 94, 95, 103).

Liu, W., Shen, X., Wang, H. & Tsang, I. W. (2020). The Emerging Trends of Multi-Label Learning. *CoRR*, *abs/2011.11197* (cf. p. 127).

Liu, Y., Su, Y., Liu, A.-A., Schiele, B. & Sun, Q. (2020). Mnemonics Training : Multi-Class Incremental Learning Without Forgetting. *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR* (cf. p. 28).

Lomonaco, V. & Maltoni, D. (2017). CORe50 : a New Dataset and Benchmark for Continuous Object Recognition. *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, *78*, 17-26 (cf. p. 61).

Lomonaco, V., Pellegrini, L., Rodríguez, P., Caccia, M., She, Q., Chen, Y., Jodelet, Q., Wang, R., Mai, Z., Vázquez, D., Parisi, G. I., Churamani, N., Pickett, M., Laradji, I. H. & Maltoni, D. (2020). CVPR 2020 Continual Learning in Computer Vision Competition : Approaches, Results, Current Challenges and Future Directions. *CoRR*, *abs/2009.09929* (cf. p. 27, 54, 61).

Lopez-Paz, D. & Ranzato, M. (2017). Gradient Episodic Memory for Continual Learning. *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 6467-6476 (cf. p. 21, 26).

Loshchilov, I. & Hutter, F. (2017). SGDR : Stochastic Gradient Descent with Warm Restarts. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (cf. p. 97).

Makhzani, A., Shlens, J., Jaitly, N. & Goodfellow, I. J. (2015). Adversarial Autoencoders. *CoRR, abs/1511.05644* (cf. p. 38, 39, 64, 66, 67, 73, 74).

Mallya, A. & Lazebnik, S. (2018). PackNet : Adding Multiple Tasks to a Single Network by Iterative Pruning. *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 7765-7773 (cf. p. 35).

Marino, K., Rastegari, M., Farhadi, A. & Mottaghi, R. (2019). OK-VQA : A Visual Question Answering Benchmark Requiring External Knowledge. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 3195-3204 (cf. p. 120).

Noroozi, M. & Favaro, P. (2016). Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, 9910*, 69-84 (cf. p. 38).

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D. & Goodfellow, I. J. (2018). Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *Advances in Neural Information Processing Systems 31 : Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (p. 3239-3250). (Cf. p. 42, 93).

Ostapenko, O., Puscas, M. M., Klein, T., Jähnichen, P. & Nabi, M. (2019). Learning to Remember : A Synaptic Plasticity Driven Framework for Continual Learning. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 11321-11329 (cf. p. 28).

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. & Wermter, S. (2019). Continual Lifelong Learning with Neural Networks : A Review. *Neural Networks, 113*, 54-71 (cf. p. 19, 20, 22).

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T. & Efros, A. A. (2016). Context Encoders : Feature Learning by Inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2536-2544 (cf. p. 38).

Pennington, J., Socher, R. & Manning, C. D. (2014). Glove : Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (p. 1532-1543). (Cf. p. 123).

Prabhu, A., Torr, P. & Dokania, P. (2020). GDumb : A Simple Approach that Questions Our Progress in Continual Learning. *The European Conference on Computer Vision (ECCV)* (cf. p. 20, 28, 34, 54, 96).

Rajasegaran, J., Hayat, M., Khan, S. H., Khan, F. S. & Shao, L. (2019). Random Path Selection for Continual Learning. *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems, NeurIPS* (cf. p. 35, 36).

Rebuffi, S.-A., Kolesnikov, A., Sperl, G. & Lampert, C. H. (2017). iCaRL : Incremental Classifier and Representation Learning. *IEEE Conference on Computer Vision and Pattern Recognition* (cf. p. 28, 31, 32, 59, 77, 78, 96, 97).

Ridnik, T., Baruch, E. B., Noy, A. & Zelnik-Manor, L. (2021). ImageNet-21K Pretraining for the Masses. *CoRR, abs/2104.10972* (cf. p. 48).

Rodríguez, N. D., Lomonaco, V., Filliat, D. & Maltoni, D. (2018). Don't forget, there is more than forgetting : new metrics for Continual Learning. *CoRR, abs/1810.13166* (cf. p. 27).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV), 115 3*, 211-252 (cf. p. 47).

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R. & Hadsell, R. (2016). Progressive Neural Networks [arXiv : 1606.04671]. *arXiv :1606.04671 [cs]* (cf. p. 35).

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R. & Hadsell, R. (2018). Progress & Compress : A scalable framework for continual learning. *35th International Conference on Machine Learning, ICML* (cf. p. 35, 36).

Serrà, J., Suris, D., Miron, M. & Karatzoglou, A. (2018). Overcoming Catastrophic Forgetting with Hard Attention to the Task. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (p. 4555-4564). (Cf. p. 35).

Shin, H., Lee, J. K., Kim, J. & Kim, J. (2017). Continual Learning with Deep Generative Replay. *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems, NeurIPS* (cf. p. 29).

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H. & Raffel, C. (2020). FixMatch : Simplifying Semi-Supervised Learning with Consistency and Confidence. *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS* (cf. p. 42-44, 86, 87, 93, 98, 101).

Soviany, P., Ionescu, R. T., Rota, P. & Sebe, N. (2021). Curriculum Learning : A Survey. *CoRR, abs/2101.10382* (cf. p. 121).

Tao, X., Chang, X., Hong, X., Wei, X. & Gong, Y. (2020). Topology-Preserving Class-Incremental Learning. *ECCV*, 16 (cf. p. 34).

Tarvainen, A. & Valpola, H. (2017). Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (p. 1195-1204). (Cf. p. 42, 93).

Tschannen, M., Bachem, O. & Lucic, M. (2018). Recent Advances in Autoencoder-Based Representation Learning. *CoRR, abs/1812.05069* (cf. p. 39).

van den Oord, A., Li, Y. & Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding. *CoRR, abs/1807.03748* (cf. p. 40).

van den Oord, A., Vinyals, O. & Kavukcuoglu, K. (2017). Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (p. 6306-6315). (Cf. p. 39).

van de Ven, G. M. & Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv :1904.07734* (cf. p. 20).

Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008* (p. 1096-1103). (Cf. p. 39).

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y. & Fu, Y. (2019). Large Scale Incremental Learning. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (cf. p. 28, 32-34, 77, 92, 94, 96, 103).

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Zhang, Z. & Fu, Y. (2018). Incremental Classifier Learning with Generative Adversarial Networks. *arXiv :1802.00853* (cf. p. 29).

Wu, Z., Xiong, Y., Yu, S. X. & Lin, D. (2018). Unsupervised Feature Learning via Non-Parametric Instance Discrimination. *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 3733-3742 (cf. p. 40).

Xie, M.-K. & Huang, S.-J. (2018). Partial Multi-Label Learning. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (p. 4302-4309). (Cf. p. 128).

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T. & Le, Q. V. (2020). Unsupervised Data Augmentation for Consistency Training. *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS,* (cf. p. 42, 43, 87, 93).

Yan, S., Xie, J. & He, X. (2021). DER : Dynamically Expandable Representation for Class Incremental Learning. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (cf. p. 35, 36).

Yoon, J., Yang, E., Lee, J. & Hwang, S. J. (2018). Lifelong Learning with Dynamically Expandable Networks. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (cf. p. 35).

Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S. & van de Weijer, J. (2020). Semantic Drift Compensation for Class-Incremental Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 6980-6989 (cf. p. 28).

Zagoruyko, S. & Komodakis, N. (2016). Wide Residual Networks. *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016* (cf. p. 23, 96).

Zenke, F., Poole, B. & Ganguli, S. (2017). Continual Learning Through Synaptic Intelligence. *34th International Conference on Machine Learning, ICML* (cf. p. 35).

Zhai, X., Oliver, A., Kolesnikov, A. & Beyer, L. (2019). S4L : Self-Supervised Semi-Supervised Learning [arXiv : 1905.03670]. *arXiv :1905.03670 [cs]* (cf. p. 56).

Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L. P., Zhang, H. & Kuo, C.-C. J. (2020). Class-incremental Learning via Deep Model Consolidation. *IEEE Winter Conference on Applications of Computer Vision, WACV* (cf. p. 32, 94, 96).

Zhang, R., Isola, P. & Efros, A. A. (2017). Split-Brain Autoencoders : Unsupervised Learning by Cross-Channel Prediction. *2017 IEEE Conference on Computer Vision*

and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, 645-654 (cf. p. 38).

Zhao, B., Xiao, X., Gan, G., Zhang, B. & Xia, S.-T. (2020). Maintaining Discrimination and Fairness in Class Incremental Learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR* (cf. p. 32-34, 59, 78, 92, 94, 96, 103).

Zhou, B., Lapedriza, À., Khosla, A., Oliva, A. & Torralba, A. (2018). Places : A 10 Million Image Database for Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, *40 6*, 1452-1464 (cf. p. 100).

Normandie Université

ed 590 MIIS
Mathématiques, Information, Ingénierie des Systèmes

Titre : Apprentissage incrémental semi-supervisé pour les applications de vision artificielle

Mot clés : Apprentissage continu ; apprentissage des représentations ; apprentissage semi-supervisé ; reconnaissance d'objet ; questions-réponses visuelles ; classification multi-label

Résumé : L'apprentissage incrémental propose un nouveau paradigme d'apprentissage pour les réseaux de neurones artificiels. Il vise à développer des systèmes capables d'enrichir leurs connaissances et leurs compétences après la phase d'entraînement initiale. Ce schéma d'entraînement est particulièrement intéressant pour des applications liées à un domaine de connaissances évolutif ou très étendu. Pour qu'une solution soit adaptée à l'apprentissage incrémental, le modèle doit remplir deux conditions : être capable d'apprendre la nouveauté et retenir ce qui a été précédemment appris, i.e. prévenir l'oubli catastrophique des acquis. Ce deuxième point est une problématique majeure des réseaux de neurones artificiels. L'ajout de nouvelles connaissances en incrémental vient écraser les précédentes. Au cours de cette thèse, nous proposons de traiter l'apprentissage continu du point de vue du domaine de l'apprentissage des représentations. En supposant l'existence de données non-annotées accessibles par le modèle pendant l'apprentissage, nous proposons des approches incrémentales semi-supervisées. Nous montrons qu'exploiter des données annexes permet de régulariser le modèle pendant le processus incrémental. Via la semi-supervision, les modèles proposés exploitent les représentations améliorées pour faciliter l'apprentissage de la nouveauté, mais aussi pour stabiliser les poids face à l'oubli catastrophique.

Title: Semi-supervised class incremental learning for computer vision

Keywords: Continual learning; representation learning; semi-supervised learning; object recognition; visual question answering; multi-label classification

Abstract: Incremental learning introduces a new learning paradigm for artificial neural networks. It aims at developing systems capable of enriching their knowledge and skills after the initial training phase. This learning scheme is particularly interesting for applications related to an evolving or very large knowledge domain. For a solution to be suitable for incremental learning, the model must fulfill two conditions: it must be able to learn novelty and retain what has been previously learned, i.e. prevent catastrophic forgetting. This second point is the major issue inherent to artificial neural networks. The addition of new knowledge in an incremental way overwrites the one currently stored in the model. In this thesis, we propose to tackle continuous learning from the point of view of the representation learning field. Assuming the existence of unlabeled data accessible by the model during training, we propose semi-supervised incremental approaches. We show that exploiting ancillary data allows to regularize the model during the incremental process. Via semi-supervision, the proposed models exploit the improved representations to facilitate learning the novelty, but also to stabilize the weights against catastrophic forgetting.