UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE Mathématiques, Informatique, Physique Théorique et Ingénierie des Systèmes

EA 4022 - LIFO

THÈSE présentée par :

Franck Anaël MBIAYA KWUITE

soutenue le : **12 Septembre 2025**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : **Informatique**

Intégration de connaissances dans des architectures profondes pour la classification d'images

THÈSE dirigée par :

Mme. VRAIN Christel	Professeur des Universités, Université d'Orléans
M. ROS Frédéric	Professeur des Universités Associé, Université d'Orléans

RAPPORTEURS :

Mme. HUDELOT Céline	Professeur des Universités, CentraleSupélec
M. FOURNEL Thierry	Professeur des Universités, Université Jean Monnet

---

JURY :

Mme. HUDELOT Céline		Professeur des Universités, CentraleSupélec
M. FOURNEL Thierry		Professeur des Universités, Université Jean Monnet
M. LEZORAY Olivier		Professeur des Universités, Université de Caen Normandie
Mme. FROMONT Elisa	(Présidente du jury)	Professeur des Universités, Université de Rennes
Mme. VRAIN Christel		Professeur des Universités, Université d'Orléans
M. ROS Frédéric		Professeur des Universités Associé, Université d'Orléans
Mme. DAO Thi-Bich-Hanh		Professeur des Universités, Université d'Orléans
M. LUCAS Yves		Maître de conférences - HDR, Université d'Orléans

Remerciements

Cette thèse a bénéficié de l'aide de nombreuses personnes que je tiens à remercier.

Je tiens tout d'abord à exprimer ma profonde gratitude et mes remerciements à mes directeurs de thèse Christel Vrain et Frédéric Ros, et mes co-encadrants Thi-Bich-Hanh Dao et Yves Lucas pour leur encadrement, leur disponibilité, leurs conseils et leurs encouragements qui ont été déterminants dans l'avancement et l'aboutissement de cette thèse.

Je souhaite également adresser mes sincères remerciements aux différents membres du jury d'avoir accepté d'évaluer cette thèse. Je remercie tout particulièrement Céline Hudelot et Thierry Fournel d'avoir accepté de rapporter sur ce travail. Je remercie également Olivier Ezoray et Elisa Fromont d'avoir accepté de participer au jury.

Je suis profondément reconnaissant pour les liens tissés au sein du laboratoire LIFO. Les échanges stimulants, la discutions bienveillantes et les moments de convivialité partagés avec les doctorants et les collègues enseignants ont grandement enrichi mon parcours, tant sur le plan académique que personnel. Je remercie tout particulièrement Isabelle Renard, qui m'a accompagné avec professionnalisme depuis mon stage de master. Son aide précieuse dans mes démarches administratives et académiques a grandement facilité mon parcours.

Je tiens à exprimer ma profonde gratitude envers ma famille, qui m'a toujours soutenu tout au long de ce parcours. Je remercie tout particulièrement ma mère, Céline, dont l'amour, la force et les sacrifices constants ont rendu possible chaque étape de mon chemin. Je suis également reconnaissant envers mes frères et sœurs, Christie, Hermine, Williams et Hilarie, pour leur présence bienveillante et leur soutien moral indéfectible. Une mention toute spéciale à Christie, sans qui je ne serais pas là aujourd'hui. Son aide, ses conseils et son soutien ont été essentiels dans l'aboutissement de ce projet de thèse.

Je souhaite également exprimer ma profonde gratitude à Léonie, pour sa patience, sa bienveillance et son soutien sans faille. Sa présence attentive m'a apporté la force et la sérénité nécessaires pour avancer, y compris dans les périodes les plus difficiles.

Je souhaite aussi exprimer ma profonde gratitude à toutes les personnes qui ont contribué à rendre ces années de thèse à la fois enrichissantes, humaines et pleines de sens. À celles et ceux, proches ou discrets, qui m'ont soutenu de près ou de loin, je dis merci. Je tiens à remercier tout particulièrement Serge, dont l'inspiration et les encouragements ont été déterminants dans ma décision d'entreprendre cette thèse.

Enfin, j'exprime ma profonde gratitude à l'Agence Nationale de la Recherche (ANR AI.iO ANR-20-THIA-0017-01) ainsi qu'à l'Université d'Orléans pour le financement qui a rendu cette thèse possible. Je remercie également la fédération CaScIModOT pour la mise à disposition des ressources de calcul du centre régional de calcul parallèle.

Résumé

Les architectures d'apprentissage profond ont transformé le paysage de nombreuses branches de l'intelligence artificielle, en particulier celui de la vision par ordinateur. Ces progrès reposent sur des modèles puissants, capables d'extraire des représentations complexes à partir de données visuelles brutes. Cependant, de nombreuses questions subsistent concernant leur robustesse, leur fiabilité, leur temps de calcul et leur explicabilité. De plus, ces modèles demeurent fortement dépendants de grandes quantités de données, qui ne sont pas toujours disponibles pour mener des travaux de recherche.

Les modèles d'apprentissage profond actuels excellent dans les tâches de classification d'images lorsqu'ils sont entraînés sur des jeux de données massifs et représentatifs. Toutefois, leur nature purement statistique les rend vulnérables face à des situations nouvelles ou ambiguës. Leur difficulté à intégrer des connaissances explicites limite leur compréhension sémantique, tandis que leur opacité soulève des préoccupations quant à leur utilisation dans des domaines critiques. Le défi consiste donc à concevoir des architectures capables non seulement de tirer parti des performances de ces modèles, mais aussi d'intégrer des éléments de raisonnement explicite pour mieux comprendre, expliquer et généraliser les décisions prises.

Cette thèse explore l'hybridation des modèles d'apprentissage profond avec des connaissances a priori dans le contexte de la classification d'images difficiles, dans le but de renforcer à la fois leur capacité de généralisation et leur explicabilité. Elle s'articule autour de trois contributions principales. Tout d'abord, nous proposons une méthode d'étiquetage et d'enrichissement de jeux de données à l'aide de règles, pour soutenir la recherche sur les modèles hybrides. Ensuite, nous introduisons une méthode d'apprentissage exploitant des graphes de connaissances au moyen d'une fonction de perte hybride. Enfin, nous étendons cette approche dans un cadre fondé sur les concepts, qui allie explicabilité, raisonnement et performance, grâce à l'utilisation de graphes sémantiques construits à partir de règles logiques.

Dans notre première contribution, nous proposons une méthodologie générique permettant, à partir d'informations complémentaires, d'étiqueter des jeux de données pour la classification d'images. Cet étiquetage est effectué sur des corpus existants, généralement utilisés pour des tâches de détection dans les images. Toujours à partir d'informations complémentaires, nous proposons également une méthode d'enrichissement des jeux de données en connaissances, celles-ci étant formalisées sous forme de règles. Cette contribution vise à combler un manque dans la communauté, en fournissant des ressources adaptées à l'évaluation de modèles s'appuyant sur des connaissances explicites.

Notre deuxième contribution propose une méthode d'intégration de graphes de connaissances dans une architecture d'apprentissage profond pour la classification. Le cœur de cette approche repose sur une fonction de perte hybride, combinant l'entropie croisée avec une mesure de distance dans un espace latent de représentations issues d'un graphe de connaissances. Ce mécanisme permet au modèle de concentrer son apprentissage sur les cas les plus difficiles, tout en exploitant la structure sémantique des connaissances intégrées. Les résultats expérimentaux montrent une amélioration des performances sur plusieurs jeux

de données, confirmant l'intérêt de cette approche pour renforcer la robustesse des modèles.

Notre troisième contribution introduit un cadre de classification d'images basé sur des concepts, visant à préserver la capacité d'explicabilité tout en améliorant les performances prédictives. Ce cadre s'appuie sur un graphe de connaissances pour, d'une part, modéliser les relations entre concepts et classes, et, d'autre part, propager l'information à travers les nœuds du graphe afin de raisonner sur les concepts. Ce modèle vise à concilier explicabilité et performance, en combinant raisonnement symbolique et apprentissage profond.

Abstract

Deep learning architectures have greatly transformed the field of artificial intelligence, particularly in computer vision. These advancements depend on powerful models that can extract complex patterns from raw visual data. However, several challenges persist, including issues of robustness, reliability, computational efficiency, and explainability. Moreover, these models generally require large amounts of annotated data, which are not always available, especially in research environments.

Current deep learning models achieve impressive results in image classification tasks when trained on large and representative datasets. However, their purely statistical nature renders them fragile when confronted with novel or ambiguous situations. Their limited capacity to incorporate explicit prior knowledge hinders their semantic understanding, and their black-box nature makes it hard to trust them in sensitive applications. The challenge is to design models that not only deliver high performance but also integrate reasoning capabilities to better understand, explain, and generalize their decisions.

This thesis explores the integration of prior knowledge into deep learning systems for complex image classification tasks, aiming to enhance both generalization and interpretability. It is structured around three main contributions. First, we propose a method to label and enrich datasets with logical rules to support research on hybrid models. Second, we introduce a learning approach that incorporates knowledge graphs using a hybrid loss function. Finally, we extend this approach through a concept-based framework that merges explainability, reasoning, and accuracy by utilizing semantic graphs built from logical rules.

Our first contribution presents a generic methodology for annotating image datasets using external semantic information. The datasets are built from existing image collentions, typically used for object detection, and enhanced with structured knowledge formalised into logical rules. This work fills a current gap by providing resources suitable for evaluating models that depend on explicit, human knowledge.

Our second contribution presents a method for integrating knowledge graphs into deep learning architectures for classification. This approach is built around a hybrid loss function that combines cross-entropy loss with a distance metric in a latent space structured by the knowledge graph. This enables the model to concentrate its learning on the most challenging cases while leveraging the semantic relationships encoded in the graph. Experimental results demonstrate consistent improvements across multiple datasets, demonstrating the robustness of this approach.

The third contribution introduces a concept-based classification framework that maintains model interpretability while enhancing predictive performance. It uses a knowledge graph to represent relationships between concepts and classes, as well as to propagate information across the nodes in the graph for reasoning. By combining symbolic reasoning with deep learning, this model achieves a balance between explainability and accuracy.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AI Artificial Intelligence
ABN Attention Branch Network
ANN Artificial Neural Network
BN Batch Normalization
CD Contextual Decomposition
CNN Convolutional Neural Network
FC Fully Connected
FCA Formal Concept Analysis
GAP Global Average Pooling
GCN Graph Convolutional Networks
GGNN Gated Graph Neural Network
GNN Graph Neural Network
GPU Graphics Processing Unit
GRU Gated Recurrent Unit
LR Learning Rate
LSTM Long Short-Term Memory
MLP MultiLayer Perceptron
ReLU Rectified Linear Unit
RNN Recurrent Neural Network
SGD Stochastic Gradient Descent
ViT Vision Transformers
XAI Explainable Articicial Intelligence

Chapitre 1

Introduction

Dans ce chapitre, nous décrivons le cadre général dans lequel s'inscrit cette thèse. Nous introduisons ensuite la problématique centrale qui a guidé ce travail, ainsi que les motivations qui ont guidé cette recherche. Nous présentons par la suite les objectifs et les contributions apportées au cours de cette thèse. Enfin, nous concluons ce chapitre par une présentation de l'organisation de ce manuscrit.

Contents

1.1 Contexte et motivation

L'intelligence artificielle ou en anglais Artificial Intelligence (AI), et plus particulièrement l'apprentissage profond, ont profondément révolutionné de nombreux domaines. C'est notamment le cas en vision par ordinateur, avec des applications variées tels que la conduite autonome (Parekh et al. 2022), la réalité augmentée (Mendoza-Ramírez et al. 2023), ou encore l'imagerie médicale (Wang et al. 2021a). Ces avancées ont permis de concevoir des modèles capables de résoudre des tâches complexes comme la classification d'images, la segmentation ou la reconnaissance d'objets, avec des performances parfois supérieures à celles des humains. Ces avancées ont été rendues possibles grâce à l'augmentation des capacités de calcul, à la disponibilité de grandes bases de données annotées et aux progrès des architectures neuronales, en particulier les réseaux de neurones convolutifs ou en anglais Convolutional Neural Network (CNN) et les modèles de type transformers (Khan et al. 2022).

Le principal verrou scientifique dans le domaine de la vision par ordinateur réside dans le fossé sémantique entre le signal d'entrée, souvent qualifié de bas niveau, et la prédiction sémantique à un niveau supérieur, comme la reconnaissance d'objets ou la classification (Klette 2014). En d'autres termes, le défi est de transformer des données brutes, souvent sous forme de pixels, en informations explicables et compréhensibles, comme la classe d'un objet ou la compréhension d'une scène. Ce décalage entre la perception visuelle brute et la compréhension cognitive est au cœur des recherches dans ce domaine, nécessitant des avancées en apprentissage automatique et en intelligence artificielle. Les différentes approches pour résoudre ce problème s'appuient sur des techniques de traitement du signal et d'apprentissage automatique. Toutefois, l'apprentissage automatique, en particulier les méthodes basées sur les réseaux de neurones profonds, se sont avérés particulièrement efficaces pour apprendre des modèles complexes à partir de grandes masses de données, avec un impact significatif à la fois sur le monde académique et sur le monde industriel.

Parmi les nombreuses applications de la vision par ordinateur, la classification d'images (Chen et al. 2021) occupe une place centrale. Il s'agit de l'une des tâches les plus répandues qui repose sur un ensemble d'images pouvant être annotées (dans le cas de l'apprentissage supervisé) ou non (dans le cadre non supervisé) (Schmarje et al. 2021). En apprentissage supervisé, les modèles d'apprentissage profond s'appuient sur des étiquettes indiquant la classe ou la catégorie à laquelle chaque image appartient. Par exemple, un modèle pourrait être entraîné pour classer des images en fonction des objets qu'elles contiennent (des chiens, des chats, etc.) ou pour détecter des caractéristiques spécifiques (des visages, des scènes, etc.). Le processus d'apprentissage consiste alors à ajuster les paramètres du réseau de neurones dans le but d'aligner ses prédictions et les étiquettes réelles des images. Dans cette thèse, nous nous intéresserons particulièrement à la classification d'images supervisée.

Malgré les performances incroyables des modèles d'apprentissage profond pour la classification d'images, ceux-ci restent largement dépendants de données massives et annotées. Cette dépendance peut limiter leur capacité à généraliser dans des contextes où les données sont rares ou imprécises. Également, il a été démontré que le succès de ces modèles n'est pas nécessairement lié à une capacité de raisonnement, mais plutôt à l'exploitation statistique des régularités et des biais présents dans les ensembles de données (Marcus 2018). Ceci impacte considérablement leur capacité de généralisation, surtout dans le contexte de la classification

à grains fins (Wei et al. 2022) où les classes sont difficiles à séparer. De plus, les modèles d'apprentissage profonds sont souvent perçus comme des boîtes noires en raison de leur opacité, ceci malgré leurs performances parfois impressionnantes (Zhang and Zhu 2018). Cette nature opaque pose des défis pour l'explicabilité et la confiance dans les décisions prises par ces modèles, en particulier dans des domaines critiques comme la médecine ou les systèmes autonomes.

Pour pallier ces lacunes, l'intégration de connaissances explicites dans les modèles d'apprentissage profond (Von Rueden et al. 2021) s'est imposée comme une approche prometteuse, permettant de renforcer leur capacité de généralisation et d'améliorer leur explicabilité. En apportant des informations structurées dans l'apprentissage tels que des règles précises, des ontologies ou des graphes de connaissances, cette méthode vise à compenser les faiblesses des modèles purement statistiques en les rendant moins dépendants des biais présents dans les données d'entraînement. Cette combinaison permet de guider l'apprentissage vers une compréhension plus explicable et plus robuste. Ainsi, l'intégration de connaissances permet de renforcer l'efficacité des modèles d'apprentissage profond, en les aidant à mieux capturer les relations sémantiques et les structures sous-jacentes des problèmes complexes auxquels ils sont confrontés.

1.2 Énoncé du problème et objectifs

Partiellement financée par l'ANR AI.iO (ANR-20-THIA-0017-01), cette thèse s'intéresse aux limites des modèles d'apprentissage profond actuels en termes de compréhension sémantique du contenu des images pour la classification. Cette problématique s'inscrit dans un cadre plus large visant à concevoir des modèles plus robustes, plus explicables et moins dépendants des données annotées. En effet, si les approches basées uniquement sur l'apprentissage statistique ont montré leurs limites en matière d'explicabilité, les approches symboliques, bien que plus explicables, peinent à capturer la richesse des données visuelles. L'enjeu est donc de combiner les modèles profonds et les connaissances a priori pour permettre aux modèles de généraliser à de nouveaux contextes ou à des situations pour lesquelles les biais présents dans les jeux de données d'entraînement ne sont plus valables.

L'objectif de cette thèse est d'explorer et de proposer des méthodes d'intégration de connaissances a priori dans les architectures d'apprentissage profond afin d'améliorer leur robustesse et la capacité de généralisation pour la classification d'images. Plus spécifiquement, notre objectif est de développer des méthodes hybrides qui, en plus des images étiquetées, intègrent des connaissances a priori. De plus, afin de pallier la rareté des données nécessaires pour mener des expérimentations dans le cadre de l'intégration de connaissances a priori, nous avons également pour objectif de proposer des jeux de données qui contribueront à la recherche dans ce domaine, et nous permettront d'évaluer la pertinence des approches que nous proposons.

1.3 Contributions générales

Dans cette thèse, nous apportons trois principales contributions. Dans la première contribution, nous proposons des jeux de données innovants pour le problème de classification d'images, intégrant des connaissances a priori formalisées sous forme de règles. Ces ensembles de données sont construits à partir de données existantes généralement utilisées pour la classification d'images à plusieurs étiquettes ou la détection d'objets. Nous proposons la construction de ces jeux de données en développant une méthode générique qui peut être utilisée pour enrichir d'autres jeux de données.

La deuxième contribution liée à cette thèse repose sur une méthode d'apprentissage profond pour la classification d'images qui exploite des connaissances formalisées sous forme d'un graphe. Ce graphe est construit à partir d'informations supplémentaires formalisées sous forme de paires attribut/valeur. La méthode que nous proposons dans ce cadre explore une fonction de perte qui combine l'entropie croisée classique, couramment utilisée en apprentissage profond, avec une nouvelle fonction de perte innovante. Cette nouvelle fonction de perte est dérivée de la représentation des nœuds après plongement du graphe de connaissances et intègre la proximité entre les nœuds de classe et d'image. Sa formulation permet au modèle de se concentrer sur l'identification des frontières entre les classes les plus difficiles à distinguer.

La troisième contribution s'inscrit dans un cadre d'apprentissage profond basé sur la notion de concept. En vision par ordinateur par exemple, les concepts peuvent représenter des éléments visuels comme la texture, la couleur ou la forme d'un objet. L'apprentissage basé sur les concepts est une approche récente qui, dans le cadre de la vision par ordinateur, vise à améliorer l'explicabilité sur la classification d'images, même si celle-ci se fait généralement au détriment de la performance de classification. Notre contribution réside dans l'utilisation d'un graphe créé à partir de règles propositionnelles. Nous utilisons un mécanisme de raisonnement fondé sur la logique floue pour propager les informations issues de ce graphe. Cette approche exploite les interdépendances entre les concepts et les classes pour améliorer la performance du modèle.

Les publications liées à cette thèse sont les suivantes :

Articles publiés dans des revues internationales

1. **Franck Anaël Mbiaya**, Christel Vrain, Frédéric Ros, Thi-Bich-Hanh Dao, and Yves Lucas. Dataset for image classification with knowledge. Data in Brief, volume 57 : pages 110893, 2024.
2. **Franck Anaël Mbiaya**, Christel Vrain, Frédéric Ros, Thi-Bich-Hanh Dao, and Yves Lucas. Knowledge graph-based image classification. Data & Knowledge Engineering, volume 151 : page 102285, 2024.

Articles soumis dans des revues internationales

- **Franck Anaël Mbiaya**, Christel Vrain, Frédéric Ros, Thi-Bich-Hanh Dao, and Yves Lucas. Concept-Based Reasoning Model. Knowledge-Based Systems, *1st revision*

Articles publiés dans des conférences nationales

1. **Franck Anaël Mbiaya**, Christel Vrain, Frédéric Ros, Thi-Bich-Hanh Dao, and Yves Lucas. Kgic : Intégration de graphe de connaissances pour la classification d'images. Revue des Nouvelles Technologies de l'Information, Extraction et Gestion des Connaissances, RNTI-E-39 : pages 259–272, 2023.

1.4 Aperçu général du manuscrit

Cette thèse se compose de neuf chapitres. Le premier chapitre est consacré à cette introduction, tandis que le dernier est dédié à la conclusion qui fait une synthèse des différentes contributions et présente des perspectives pour des travaux futurs. Le chapitre 2 présente des notions préliminaires importantes qui sont utilisées dans nos différentes contributions. Il fait un rappel sur les réseaux de neurones en vision par ordinateur, présente les notions liées à la fouille de données et l'analyse formelle de concepts, et se termine en introduisant le concept de plongement de graphe. Le chapitre 3 expose la problématique de la classification à grain fin et introduit les travaux de l'état de l'art ayant abordé ce défi complexe. Le chapitre 4 présente les travaux existants intégrant des connaissances dans les architectures profondes pour les problèmes de vision par ordinateur. Le chapitre 5 analyse les différentes techniques permettant d'expliquer les décisions prises par les modèles profonds en vision par ordinateur, et présente des travaux connexes. Les trois chapitres suivants détaillent les différentes contributions de cette thèse. Le chapitre 6 traite de la création et de l'enrichissement de bases d'images avec des connaissances, en vue de la création de nouveaux jeux de données. Le chapitre 7 présente une méthode d'intégration de graphes de connaissances pour la classification d'images. Enfin, le chapitre 8 aborde la classification d'images basée sur des concepts, en intégrant des règles pour améliorer la généralisation et l'explicabilité des modèles.

Première partie

Préliminaires et état de l'art

Chapitre 2

Préliminaires

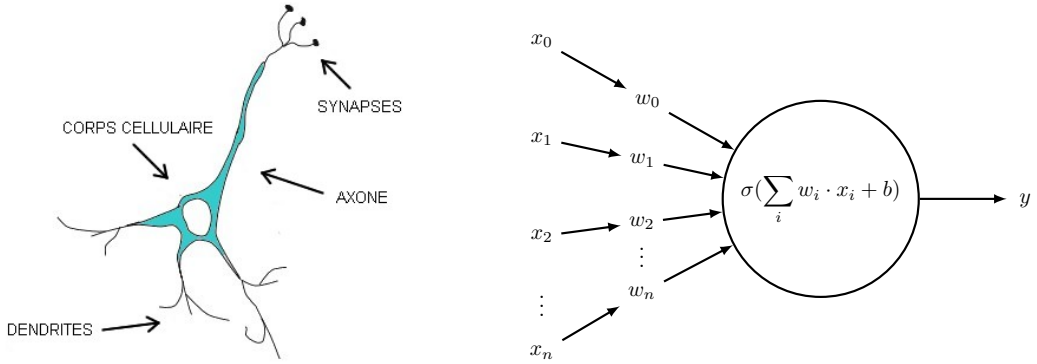
Les réseaux de neurones artificiels ou en anglais Artificial Neural Network (ANN) représentent une classe de modèles d'apprentissage automatique qui sont directement inspirés de la structure et du fonctionnement du cerveau humain. Leur développement a été un processus itératif, jalonné de découvertes théoriques et technologiques majeures ayant conduit à l'émergence de l'apprentissage profond (deep learning). Ce chapitre se propose d'explorer les fondements des réseaux de neurones artificiels et leurs évolutions vers des architectures plus complexes capables de traiter des tâches de plus en plus sophistiquées, notamment pour la classification d'images. À des fins d'expérimentations, nous avons dû enrichir des bases de données avec des connaissances. Pour les créer nous nous appuyons sur la fouille de motifs fréquents et l'analyse formelle de concepts, dont les notions de base sont introduites dans ce chapitre. Nous nous intéressons également à l'intégration de connaissances en classification d'images. Nos connaissances peuvent être transformées sous forme de graphes et nous introduisons dans ce chapitre les notions nécessaires à la compréhension de nos contributions.

Contents

2.1 Réseaux de neurones artificiels en vision par ordinateur

2.1.1 Perceptron

Proposé par Frank Rosenblatt en 1958, le perceptron (Rosenblatt 1958) est souvent considéré comme le premier modèle d'intelligence véritablement basé sur des neurones artificiels. Il tire son origine d'un modèle mathématique inspiré des neurones biologiques. Une cellule neuronale biologique est composée de trois parties principales : les dendrites, le corps cellulaire et l'axone (Figure 2.1a). Les dendrites jouent un rôle essentiel dans la réception des signaux électriques provenant d'autres neurones. Elles captent ces signaux et les transmettent vers le corps cellulaire. C'est dans le corps cellulaire que les signaux sont additionnés. Si le signal total dépasse un certain seuil, une impulsion est déclenchée et envoyée vers l'axone qui conduit ensuite cette impulsion électrique vers d'autres neurones. Ce processus de transmission est essentiel pour la communication neuronale dans le cerveau.



(a) Schéma d'un neurone biologique [1].

(b) Schéma d'un perceptron.

FIGURE 2.1 – Schéma illustrant l'analogie entre une cellule neuronale biologique et un neurone artificiel.

Cette analogie biologique est reflétée dans la conception des perceptrons artificiels. Un perceptron (Figure 2.1b) est composé d'un ou plusieurs signaux d'entrées, sur lesquels sont appliqués des poids synaptiques qui représentent la force ou l'importance de chaque connexion. L'ensemble de ces signaux pondérés est additionné et une valeur fixe appelée biais peut être ajoutée. Une fonction d'activation est ensuite utilisée, jouant un rôle de seuil de déclenchement : si la somme pondérée des entrées dépasse un certain seuil, la sortie du perceptron vaut 1, sinon celle-ci vaut 0. Mathématiquement, le perceptron peut être défini par la fonction suivante :

$$y = \sigma(\sum_i w_i \cdot x_i + b) \tag{2.1}$$

où x_i représente les entrées i, w_i représente les poids associés, b représente le biais, y représente la sortie du perceptron et σ représente la fonction d'activation.

Bien que le perceptron soit une avancée conceptuelle importante, il reste un modèle très simple composé d'une seule couche de neurones. L'objectif de Rosenblatt était de concevoir

1. Source :https ://commons.wikimedia.org/wiki/File :Neuron_biologique.JPG

une machine capable de classer des entrées en deux classes distinctes, basée sur une fonction de décision linéaire. En particulier, Minsky and Papert (1969) a démontré que le perceptron ne pouvait résoudre que des problèmes linéairement séparables.

2.1.2 Réseaux de Neurones Multi-couches

Pour surmonter les limitations du perceptron, un modèle combinant plusieurs perceptrons a été proposé pour permettre de répondre à des problèmes plus complexes. Ce modèle est appelé perceptron multi-couches ou en anglais MultiLayer Perceptron (MLP).

Un perceptron multi-couches est constitué de plusieurs rangées de neurones appelées couches (Figure 2.2). Il comporte trois types de couches : la couche d'entrée qui reçoit les données brutes, les couches cachées qui permettent d'apprendre des représentations internes complexes et la couche de sortie qui produit la décision finale du modèle. En dehors de la couche de sortie, chaque couche reçoit en entrée un vecteur contenant l'ensemble des sorties des neurones de la couche précédente. Les couches successives sont donc totalement connectées (Fully Connected (FC)). Le nombre de neurones dans la couche d'entrée et dans la couche de sortie dépend de la dimension de la variable d'entrée et de la variable de sortie. Cependant, le nombre de neurones dans les couches cachées est un hyperparamètre à définir avec l'architecture du réseau. L'ajout de couches cachées permet au perceptron multi-couches de capturer des relations non linéaires entre les données d'entrée et de sortie. Plus le nombre de couches cachées augmente, plus le réseau devient profond, ce qui lui permet d'apprendre des représentations hiérarchiques sur les données.

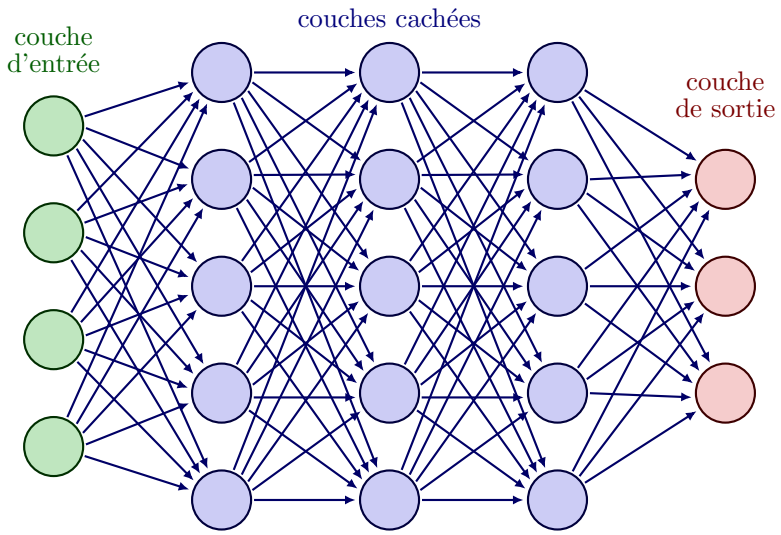


FIGURE 2.2 – Perceptron multi-couches (MLP) avec trois couches intermédiaires.

La fonction d'activation dans un perceptron est essentielle pour assurer la non-linéarité du réseau, à condition d'utiliser une fonction non linéaire. L'évolution des réseaux de neurones a vu l'utilisation de plusieurs fonctions d'activation, chacune offrant une propagation différente de l'information (Dubey et al. 2022). La principale innovation avec les réseaux de neurones multi-couches est l'utilisation de l'algorithme de rétropropagation du gradient (Rumelhart et al. 1986) pour ajuster les poids du réseau. Cet algorithme repose sur le calcul du gradient,

c'est-à-dire de la dérivée partielle de la fonction de perte par rapport aux différents poids, et permet l'apprentissage dans des architectures à plusieurs couches. Ainsi, les gradients sont calculés couche après couche en partant de la dernière, selon la formule suivante :

$$\delta_{ij}^{(k)} = \frac{\partial \mathcal{L}}{\partial \theta_{ij}^{(k)}} \tag{2.2}$$

où $\mathcal{L}$ est la fonction de perte, $\theta_{ij}^{(k)}$ est le poids entre le neurone j de la couche $k-1$ et le neurone i de la couche k, et $\delta_{ij}^{(k)}$ est le gradient de la perte par rapport à ce poids. Les gradients calculés sont utilisés pour mettre à jour les poids du réseau à l'aide de l'algorithme de descente de gradient, qui utilise un facteur λ, appelé taux d'apprentissage ou en anglais Learning Rate (LR), pour contrôler la vitesse de mise à jour des poids. Cette mise à jour a pour objectif de minimiser la fonction de perte du réseau, et s'effectue selon l'équation suivante :

$$\hat{\theta}_{ij}^{(k)} = \theta_{ij}^{(k)} - \lambda \delta_{ij}^{(k)} \tag{2.3}$$

où $\hat{\theta}_{ij}^{(k)}$ est la nouvelle valeur du poids entre le neurone j de la couche $k-1$ et le neurone i de la couche k après la mise à jour. Pour éviter de stocker en mémoire tous les résultats intermédiaires d'un jeu de données avant d'effectuer la rétropropagation, les données sont généralement traitées individuellement ou par petits groupes appelés batchs ou mini-batchs. Cette approche permet un apprentissage stochastique plus efficace et réduit la charge mémoire. On parle alors de descente de gradient stochastique ou en anglais Stochastic Gradient Descent (SGD) pour optimiser les poids du réseau. Le nombre de batchs est donc un hyper-paramètre qui est fonction de la taille du réseau, des données d'entrées et des ressources matérielles à disposition, et sa valeur peut influencer la vitesse de convergence et la capacité de généralisation du réseau (Keskar et al. 2017). Mis à part la descente de gradient stochastique, plusieurs autres algorithmes et méthodes ont été proposés pour contourner le problème lié à la convergence vers des minima locaux ou pour l'accélérer. Parmi eux, on peut citer le momentum (Sutskever et al. 2013) utilisé pour accélérer la convergence de la descente de gradient ou encore Adam (Kingma and Ba 2014) qui permet d'éviter les minima locaux grâce à ses ajustements dynamiques du taux d'apprentissage (LR). Lors de l'entraînement d'un modèle, une seule passe sur l'ensemble des données ne suffit généralement pas pour apprendre des représentations efficaces et minimiser la fonction de perte. L'entraînement est donc répété sur plusieurs époques pour permettre au modèle d'affiner progressivement ses paramètres.

2.1.3 Réseau de neurones convolutif

Bien que les MLPs soient puissants, ils ne sont pas adaptés aux données structurées comme les images, où les relations spatiales entre pixels sont cruciales. Les réseaux de neurones convolutifs (CNN), introduits par Yann LeCun avec l'architecture LeNet (LeCun et al. 1989) en 1989, sont spécialement conçus pour le traitement des images. Un CNN est généralement composé de plusieurs types de couches, parmi lesquelles :

- **Couches de convolution** : Ces couches appliquent des filtres (ou noyau de convolution) qui glissent sur l'image, détectant ainsi des motifs locaux comme des contours ou des textures.

- **Couches de regroupement (Pooling)** : Également appelées couches de sous-échantillonnage, elles sont utilisées pour réduire la taille des représentations intermédiaires tout en préservant les informations essentielles.

- **Couches entièrement connectées (FC)** : En fin de réseau, ces couches permettent d'intégrer toutes les caractéristiques extraites pour effectuer la tâche finale.

Appliquer une convolution sur une image I revient à calculer la somme pondérée du voisinage de chaque pixel et créer une nouvelle image avec ces valeurs. Un filtre ou noyau de convolution est un ensemble de poids stockés dans une matrice N et appliqués sur la totalité de l'image. Mathématiquement, le produit de convolution pour un pixel de coordonnées (a, b) s'exprime comme :

$$(I * N)(a, b) = \sum_i \sum_j I(a, b) \cdot N(a - i, b - j) \tag{2.4}$$

où $N(a - i, b - j)$ est le déplacement du noyau N sur l'image (N joue donc le rôle de fenêtre glissante), et la double somme représente une opération de moyennage. En faisant varier la taille du noyau et les valeurs de ses poids, on peut alors réaliser un certain nombre d'opérations telles que la détection des contours, l'amélioration de netteté, etc.

Pour pouvoir extraire le maximum d'informations dans les images, les réseaux utilisent généralement plusieurs noyaux différents pour chaque couche de convolution. Chaque couche génère ainsi autant d'images de sortie que de noyaux dans la couche. Ces images de sortie sont appelées canaux et possèdent toutes la même taille. Après les couches de convolution, des couches de regroupement sont généralement utilisées pour réduire la taille des canaux. Elles peuvent être de deux types : le *Max pooling* (regroupement par maximum) et le *Average pooling* (regroupement par moyenne). Ces couches permettent de concentrer l'information utile et de capturer les structures spatiales à différentes échelles en réduisant la taille des matrices. Cette diminution de la dimension des vecteurs de caractéristiques contribue également à une réduction de la complexité de calcul.

2.1.4 Architectures profondes pour la classification supervisée en vision par ordinateur

LeNet

La vision par ordinateur est l'un des domaines qui a le plus bénéficié des progrès sur les réseaux de neurones. Le tout premier réseau à utiliser des couches de convolution est LeNet (LeCun et al. 1989) en 1998. Développé pour la reconnaissance de caractères, LeNet est capable d'extraire automatiquement des caractéristiques à partir des images en utilisant des

filtres convolutifs, puis d'apprendre à classifier les images avec des couches entièrement connectées. Il comporte sept couches (Figure 2.3) dont trois couches de convolution, deux couches de regroupement et deux couches entièrement connectées. Le réseau prend en entrée une image de taille 32×32 et donne en sortie l'un des dix digits possibles. Encore appelée extracteur de caractéristiques, les couches de convolution possèdent respectivement 6, 16 et 120 canaux. Leur rôle est d'extraire un vecteur de caractéristiques qui sera utilisé par les couches entièrement connectées pour la résolution de la tâche. Les couches de regroupement interviennent après les deux premières couches de convolution, et permettent de faire la moyenne avec un filtre de 22 pour réduire la taille des canaux. Pour apporter la non-linéarité dans LeNet, une couche d'activation *Tanh* est ajoutée après chaque couche de convolution.

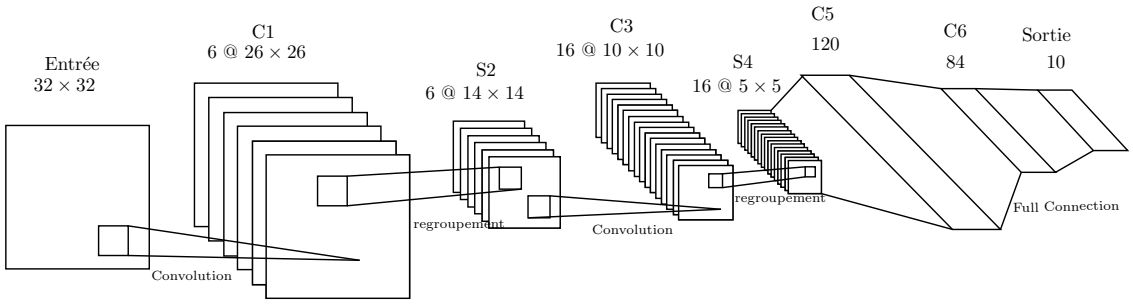


FIGURE 2.3 – Architecture du réseau convolutif LeNet (LeCun et al. 1989).

AlexNet

LeNet a créé les bases des réseaux convolutifs modernes. Cependant, il n'était pas populaire à l'époque en raison du manque de ressource matériel pour l'entraîner. La renaissance des réseaux de neurones profonds dans le domaine de la vision par ordinateur s'est produite avec AlexNet (Krizhevsky et al. 2012) qui a remporté la compétition *ImageNet* en 2012. Cette architecture reprend le principe général de LeNet mais est conçu pour traiter des images plus grandes ($224 \times 224 \times 3$) en utilisant un Graphics Processing Unit (GPU). AlexNet comporte cinq couches convolutives, trois couches de regroupement maximal et deux couches entièrement connectées (Figure 2.4) et une couche SoftMax pour la prédiction de classes. Chaque couche de convolution se compose d'un filtre de convolution et d'une fonction d'activation non linéaire Rectified Linear Unit (ReLU) (Agarap 2018). Alexnet utilise l'augmentation des données et un dropout (Srivastava et al. 2014) pour réduire le sur-apprentissage pendant l'entraînement du réseau. L'augmentation des données consiste à générer de nouvelles données d'entraînement à partir des données existantes, en appliquant diverses transformations (rotation, translation, symétrie, recadrage, etc.) afin d'enrichir le jeu de données. Le principe de base du dropout est de désactiver de manière aléatoire un certain nombre de neurones pendant l'entraînement du réseau, ce qui empêche le réseau de trop s'adapter aux données d'entraînement spécifiques et améliore ainsi sa capacité de généralisation du modèle.

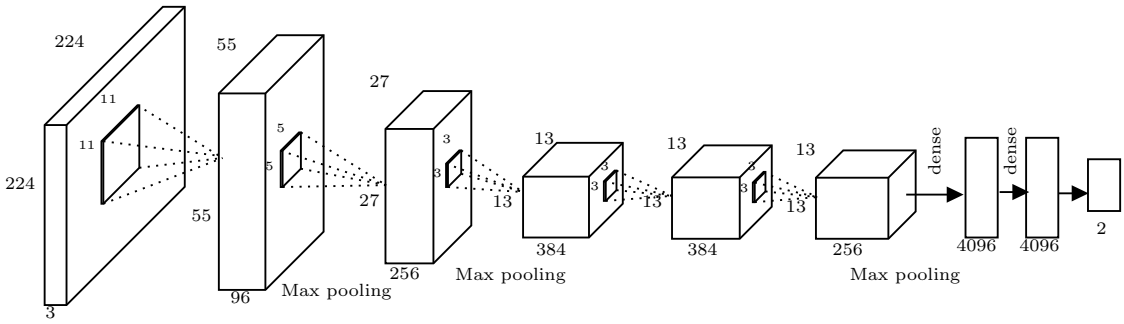


FIGURE 2.4 – Architecture du réseau convolutif AlexNet (Krizhevsky et al. 2012).

VGGNet

En 2014, VGGNet (Simonyan and Zisserman 2014) a poussé encore plus loin la profondeur des réseaux de neurones. Cette architecture a montré que l'utilisation de plusieurs petites couches convolutives (3×3) en cascade permet une extraction de caractéristiques plus fines à chaque couche convolutive et pouvait ainsi améliorer les performances sans augmenter de manière significative le nombre de paramètres. VGGNet accepte des images de 224×224 pixels en entrée, et se décline en plusieurs versions (VGG-11, VGG-16, VGG-19), où le chiffre indique le nombre total de couches. La version VGG-16 par exemple (Figure 2.5), comporte 16 couches. Elle possède 12 couches de convolution, chacune suivie d'une activation non linéaire ReLU. Elle possède également des couches de regroupement maximal, et quatre couches entièrement connectées parmi lesquels une couche de sortie Softmax. Bien que performant, VGGNet souffre de la taille massive de son modèle et son coût de calcul élevé, ce qui rend son entraînement très lent et limite son application dans des environnements contraints. L'augmentation de la profondeur du réseau peut également conduire au problème de disparition des gradients. En effet, plus le réseau devient profond, plus les gradients de la fonction de perte deviennent de plus en plus petits pendant la rétro-propagation, ce qui rend difficile l'apprentissage du réseau.

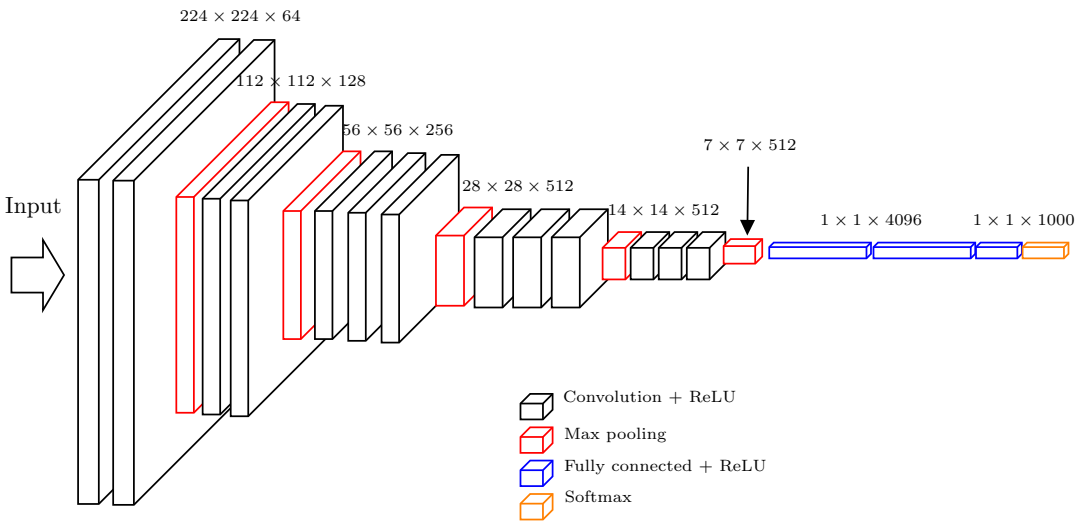
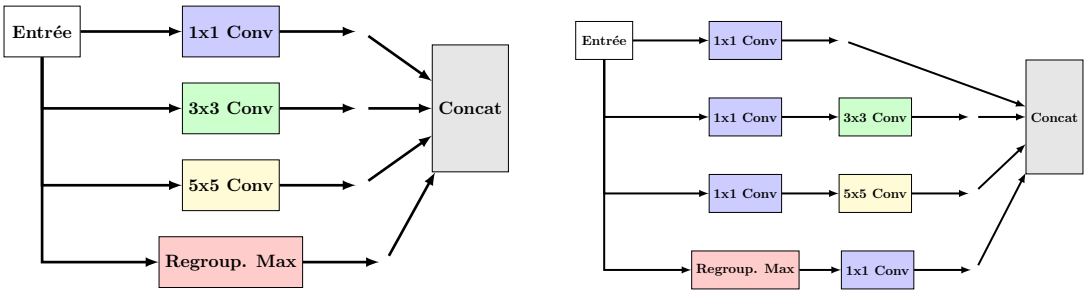


FIGURE 2.5 – Architecture du réseau convolutif VGG-16 (Simonyan and Zisserman 2014).

Inception (GoogLeNet)

En parallèle à VGGNet, Inception (Szegedy et al. 2014), aussi connu sous le nom de GoogLeNet, a proposé une architecture unique en ce qu'elle permet à chaque couche de traiter des informations à différentes échelles de manière parallèle en combinant des convolutions 1×1, 3×3, et 5×5 dans un même module appelé **module Inception** (Figure 2.6). Cette approche de traitement parallèle permet non seulement au réseau de capturer plus efficacement des caractéristiques à différentes échelles, mais aussi d'atténuer le problème de disparition des gradients causé par la profondeur du réseau. Plusieurs variantes d'Inception ont été développées afin d'optimiser l'efficacité de l'architecture.

La version Inception v2 (2015) introduit plusieurs améliorations, notamment la normalisation par lots (Batch Normalization (BN)) (Ioffe and Szegedy 2015) et les connexions par raccourcis. Elle affine également les modules en remplaçant les grandes convolutions par des convolutions plus petites, ce qui améliore la précision tout en réduisant le temps d'entraînement. La version Inception v3 (2015) intègre une variante de la convolution standard, appelée convolution à trous, qui permet d'élargir le champ réceptif du réseau sans augmenter le nombre de paramètres. Le principe repose sur l'insertion de **trous** (zéros) entre les valeurs du noyau de convolution, permettant ainsi de capturer davantage d'informations globales dans l'image sans augmenter la taille du noyau. Inception v4 et Inception-ResNet (2016) introduisent des connexions résiduelles, inspirées de ResNet (présenté ci-dessous), ce qui améliore encore les performances du réseau.



(a) Module Inception, version naïve.

(b) Module Inception avec réduction de dimension.

FIGURE 2.6 – Deux exemples de modules Inception.

ResNet

Pour pallier le problème de fuite des gradients des réseaux neuronaux très profonds, ResNet (He et al. 2015) a été introduit en 2015. Son innovation clé est l'utilisation de connexions résiduelles entre différentes couches, ce qui permet aux gradients de remonter plus facilement vers les premières couches. Dans une architecture ResNet, ce procédé est modélisé par des blocs appelés **blocs résiduels** (Figure 2.7). Chaque bloc résiduel est constitué de deux branches : une branche principale qui traverse plusieurs couches de convolution, souvent suivies de normalisation par lots (BN) et d'une fonction d'activation, et une connexion résiduelle qui contourne les couches convolutives de la branche principale et connecte directement l'entrée du bloc à sa sortie. La sortie finale du bloc résiduel est obtenue en ajoutant les résultats de la branche principale à ceux de la connexion résiduelle.

ResNet se décline en plusieurs variantes, avec un nombre de couches allant de 18 à 152 pour les réseaux les plus profonds.

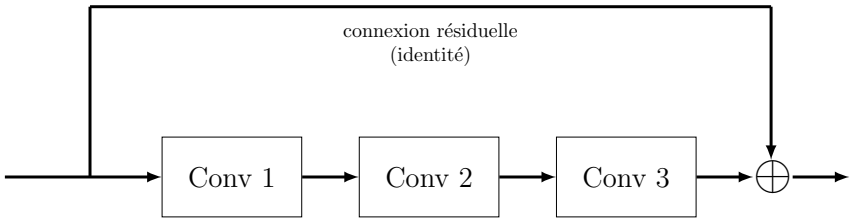


FIGURE 2.7 – Bloc résiduel du réseau ResNet (He et al. 2015).

DenseNet

Introduite en 2017, DenseNet (Huang et al. 2016) est une architecture conçue pour atténuer le problème de fuite des gradients grâce à l'utilisation de blocs denses. Un bloc dense est composé de plusieurs couches où chaque couche reçoit en entrée les sorties de toutes les couches précédentes et transmet sa propre sortie à toutes les couches suivantes (Figure 2.8). Cette connectivité renforcée favorise un meilleur flux d'information et une réutilisation efficace des caractéristiques apprises, ce qui améliore la convergence et réduit le nombre de paramètres nécessaires.

Ces connexions denses rendent l'apprentissage plus efficace car l'information n'est pas diluée à chaque étape. Toutes les caractéristiques précédemment extraites sont disponibles pour l'entraînement, augmentant ainsi la diversité des caractéristiques accessibles sans avoir à recalculer ou réapprendre de nouvelles représentations complexes. Les connexions denses permettent également à l'erreur de se rétro-propager jusqu'à la première couche sans être trop atténuée. De plus, la réutilisation des informations extraites dans les couches précédentes permet de réduire le nombre de paramètres dans le modèle par rapport aux architectures qui nécessitent de recalculer les caractéristiques à chaque couche comme dans ResNet. DenseNet a montré des améliorations significatives en termes de performances et d'efficacité, avec une profondeur allant jusqu'à 264 couches.

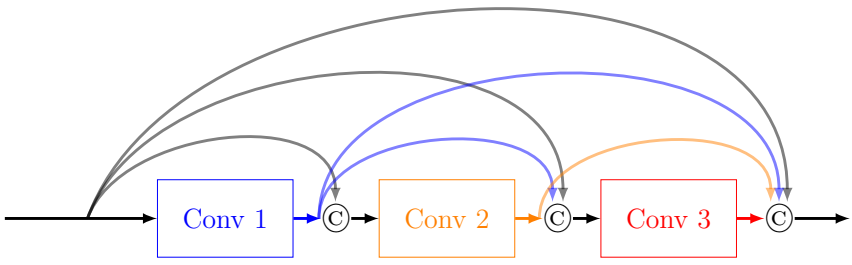


FIGURE 2.8 – Bloc dense du réseau DenseNet (Huang et al. 2016).

Transformers en vision par ordinateur (ViT)

Récemment, les Transformers, introduits initialement pour le traitement du langage naturel (Vaswani et al. 2017), ont été appliqués avec succès à la vision par ordinateur, donnant naissance aux Vision Transformers (ViT) (Dosovitskiy et al. 2020). Contrairement aux CNN qui reposent sur des convolutions pour extraire des caractéristiques locales, les *Vision Transformers* utilisent des mécanismes dit d'attention qui permettent de capturer des relations globales entre les pixels d'une image. Les mécanismes d'attention rendent les ViTs particulièrement efficaces pour des tâches nécessitant une compréhension globale des images, comme la détection d'objets ou la segmentation (Thisanke et al. 2023).

L'architecture des ViT se distingue par plusieurs étapes clés (Figure 2.9). La première étape est le découpage des images en patchs. Au lieu de traiter une image entière comme dans les CNN, les *Vision Transformers* découpent une image en petits carrés de taille fixe ou patchs, qui sont ensuite aplatis en vecteurs. La deuxième étape est l'encodage des patchs, qui consiste à appliquer une transformation linéaire sur chaque patch. La troisième étape consiste à ajouter un encodage positionnel à chaque patch pour conserver les informations de localisation spatiale. La quatrième étape consiste à passer chaque patch dans un encodeur, qui consiste en plusieurs couches d'encodage empilées. Pour finir, la dernière étape est un classifieur qui utilise un token de classification pour prédire la classe de l'image. Ce token encapsule toutes les informations de l'image via le mécanisme d'attention. Malgré ses grandes performances par rapport aux CNNs, les ViT nécessitent de grandes quantités de données et ont un coût de calcul très élevé.

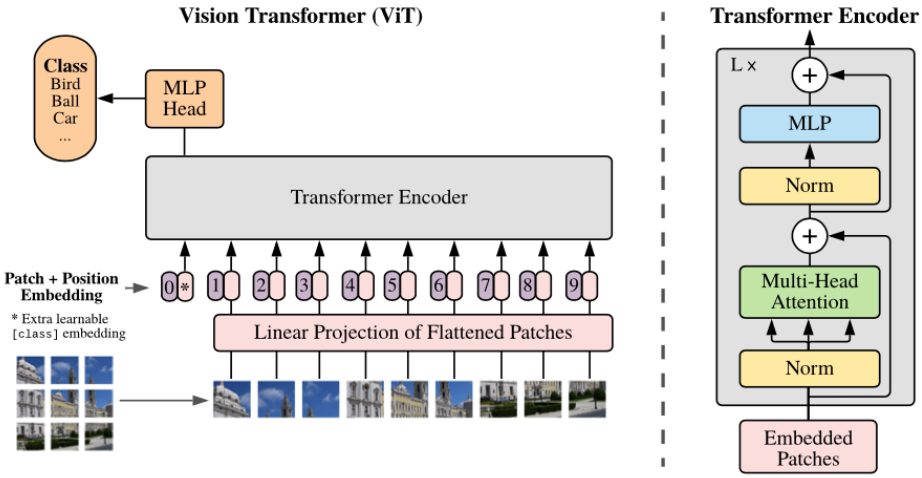


FIGURE 2.9 – Architecture d'un Vision Transformer (ViT) (Dosovitskiy et al. 2020).

2.2 Fouille de motifs

La fouille de données ou *data mining* (Jain and Srivastava 2013) consiste à extraire des informations à partir de données stockées dans des bases de données. Son but est de mieux comprendre ces données et/ou de prendre des décisions. Parmi les tâches fondamentales de fouille de données, on retrouve la fouille de motifs ou *pattern mining*.

La fouille de motifs est un sous-domaine de la fouille de données qui vise à identifier des motifs intéressants et utiles (Chand et al. 2012). L'objectif est de trouver des motifs facilement interprétables, ceci afin de faciliter la compréhension des données. Ces motifs peuvent être utilisés pour la prise de décision, mais aussi pour d'autres tâches comme la classification ou le regroupement (clustering), cadre dans lequel nous l'avons utilisé pour proposer de nouveaux jeux de données. Il existe plusieurs types de fouille de motifs selon la nature des données et les objectifs visés. Parmi les plus connus, on trouve la fouille de motifs fréquents qui consiste à identifier des ensembles d'éléments apparaissant fréquemment dans une transaction, ou encore la fouille de motifs séquentiels (Fournier-Viger et al. 2017) qui prend en compte l'ordre des éléments dans des séquences. Dans la suite de cette section, nous présenterons plus formellement la fouille de motifs fréquents, c'est l'approche que nous avons utilisé pour proposer de nouveaux jeux de données dans notre première contribution.

Dans le cadre de la fouille de motifs fréquents, l'un des objectifs classiques est d'extraire tous les ensembles de motifs ou item (itemsets) apparaissant dans un nombre suffisant de transactions, c'est-à-dire dépassant un seuil de support minimal. Cependant, l'extraction de tous les motifs fréquents peut engendrer une grande quantité de résultats redondants. Pour pallier ce problème, plusieurs sous-classes de motifs on été proposées, dont les motifs fermés fréquents. Ils permettent de compresser l'ensemble des motifs fréquents sans perdre d'informations en terme de support. Un itemset est dit fermé s'il n'existe aucun item supplémentaire pouvant être ajouté à celui-ci sans réduire son support, c'est-à-dire sans diminuer le nombre de transactions dans lesquelles il apparaît.

Pour formaliser cette notion, nous nous appuyons sur l'analyse formelle des concepts, ou en anglais Formal Concept Analysis (FCA) (Ganter et al. 2005), dans laquelle les données sont représentées sous forme d'une relation binaire $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ entre un ensemble de transactions $\mathcal{T}$ et un ensemble d'items $\mathcal{I}$. À chaque paire $(t, i) \in \mathcal{R}$, on associe la présence de l'item i dans la transaction t. Grâce aux opérateurs de fermeture f et g, la FCA identifie des concepts formels sous forme de paires (T, I), où $T \subseteq \mathcal{T}$ et $I \subseteq \mathcal{I}$ satisfont les propriétés $f(T) = I$ et $g(I) = T$. Les opérateurs f et g sont définis comme suit :

$$\begin{aligned} f(T) &= \{i \in \mathcal{I} | \forall t \in T, (t, i) \in \mathcal{R}\} \\ g(I) &= \{t \in \mathcal{T} | \forall i \in I, (t, i) \in \mathcal{R}\} \end{aligned} \tag{2.5}$$

L'ensemble $f(T)$ représente l'ensemble des items communs à toutes les transactions de T, et $g(I)$ représente l'ensemble des transactions contenant tous les items de I. Une transaction t supporte un itemset I si $t \in g(I)$. Dans ce cadre, un motif est fermé si et seulement si $f(g(I)) = I$. Un motif fermé est dit fréquent lorsque son support $|g(I)|$ dépasse un seuil.

2.3 Plongement de graphes

Dans le cadre de nos contributions, des structures de graphe sont utilisées pour modéliser des connaissances. Ces graphes servent de support pour intégrer des connaissances dans les architectures profondes. L'objectif de cette section est d'introduire formellement le concept de plongement de graphe, qui désigne l'ensemble de méthodes permettant de projeter le graphe ou des parties de ceux-ci dans dans un espace vectoriel.

2.3.1 Concepts fondamentaux

Un graphe est une structure mathématique permettant de modéliser des relations entre objets. Un graphe peut être formellement défini comme un couple $\mathcal{G} = (V, E)$, où V est un ensemble fini de nœuds, noté $V = \{v_1, \ldots, v_{|V|}\}$ et E est un ensemble d'arêtes, où chaque arête relie deux nœuds. Selon la nature des arêtes, il existe deux types principaux de graphes :

- Les graphes non orientés qui sont des graphes dans lesquels les arêtes n'ont pas de direction, c'est-à-dire que chaque arête relie deux nœuds de manière symétrique. Formulée de manière formelle, une arête dans un graphe non orienté est représentée par un ensemble non ordonné $e = \{u, v\}$ où $u, v \in V$.

- Les graphes orientés sont des graphes dans lesquels les arêtes possèdent une direction, ce qui signifie qu'une connexion va d'un nœud à un autre de manière asymétrique. Dans un graphe orienté, une arête est représentée par un couple ordonné $e = (u, v)$, indiquant une connexion dirigée du nœud u vers le nœud v, avec $u, v \in V$.

Ces deux types de graphes offrent la possibilité de modéliser une variété de structures selon les relations que l'on souhaite représenter. Ils peuvent être enrichis en associant des informations supplémentaires à chaque nœud ou arête, ou en attribuant des poids aux arêtes pour indiquer l'intensité ou la probabilité d'une relation. Dans ce dernier cas, on parle alors de graphes pondérés. Les graphes peuvent également être homogènes et hétérogènes. Un graphe homogène se compose de nœuds et d'arêtes d'un seul type, ce qui est adapté pour représenter des réseaux simples où tous les éléments ont la même nature. À l'inverse, un graphe hétérogène contient plusieurs types de nœuds et/ou d'arêtes, permettant de modéliser des relations plus complexes entre différentes entités.

Pour un graphe non orienté, le voisinage de v, noté $\mathcal{N}(v)$, est défini par $\mathcal{N}(v) = \{u \in V \mid \{u, v\} \in E\}$. Dans un graphe orienté, il est nécessaire de distinguer les voisins entrants et les voisins sortants. Pour un nœud v, le voisinage entrant, noté $\mathcal{N}_{in}(v)$, est défini par $\mathcal{N}_{in}(v) = \{u \in V \mid (u, v) \in E\}$. Le voisinage sortant de v, noté $\mathcal{N}_{out}(v)$, est défini par $\mathcal{N}_{out}(v) = \{u \in V \mid (v, u) \in E\}$.

Etant donné un graphe $\mathcal{G} = (V, E)$ et une dimension de plongement prédéfinie d, le problème de plongement de graphe consiste à créer une fonction $f : v \to \mathbf{h}_v$ qui encode chaque nœud v en un vecteur $\mathbf{h}_v \in \mathbb{R}^d$ dans lequel des propriétés sont conservées autant que possible. Ces propriétés de graphe peuvent être quantifiées à l'aide de mesures de proximité entre nœuds parmi lesquels :

- La proximité de premier ordre : la proximité de premier ordre entre les nœuds u et v correspond au poids de l'arête (u, v). Deux nœuds sont d'autant plus similaires que le poids de l'arête qui les relie est élevé. Cette propriété est généralement importante pour les graphes pondérés.

- La proximité de second ordre : la proximité de second ordre entre les nœuds u et v décrit la similarité entre les structures de voisinage $\mathcal{N}(u)$ et $\mathcal{N}(v)$. Deux nœuds

sont d'autant plus similaires s'ils partagent des connexions similaires avec les autres nœuds du graphe, même s'il ne sont pas directement reliés entre eux.

- La proximité d'ordre supérieur : la proximité d'ordre supérieur entre les nœuds u et v décrit la similarité entre leurs rôles ou positions dans la structure globale du graphe, au-delà de leur voisinage immédiat. Les proximités d'ordre supérieur sont parfois définies à l'aide de l'indice Katz (Katz 1953), l'Adar Adamic (Adamic and Adar 2003), etc.

Certaines méthodes de plongement vont au-delà de la simple représentation des nœuds, car elles cherchent également à apprendre des vecteurs pour les arêtes. Par ailleurs, des d'autres approches ciblent la représentation de sous-graphes afin de modéliser des motifs ou des structures locales complexes au sein du graphe.

Certaines connaissances plus complexes, souvent issues d'ontologie, sont modélisées à l'aide de graphes dit sémantiques. Ces graphes sont structurés sous forme de triplets (h, r, t) où h désigne le nœud entête, r le type de relation entre h et t, et t est le nœud cible. L'objectif du plongement est d'apprendre des représentations vectorielles pour h, r et t dans un espace vectoriel, de manière à ce que le triplet valide (h, r, t) minimise une fonction de perte, pendant que les triplets invalides (h', r, t') la maximisent.

2.3.2 Méthodes de plongement de graphe

Les différents algorithmes de plongement de graphe se distinguent par la façon dont ils déterminent quelles propriétés du graphe doivent être conservées. Ces algorithmes peuvent être regroupés en différentes catégories.

Méthodes basées sur la factorisation matricielle

Les méthodes de plongement de graphe basées sur la factorisation matricielle (Goyal and Ferrara 2018) visent à décomposer des matrices représentant les relations entre les nœuds, telles que les matrices d'adjacence ou les matrices de similarité, afin d'obtenir des représentations vectorielles dans un espace de moindre dimension qui préservent certaines propriétés structurelles du graphe. Formellement, la factorisation matricielle consiste à décomposer une matrice $\mathbf{M}$ de grande dimension en produit de deux ou plusieurs matrices de dimension réduite :

$$\mathbf{M} \approx U \cdot W^T \tag{2.6}$$

où $\mathbf{M}$ est la matrice d'origine représentant les relations entre les nœuds, et U et W sont des matrices contenant des représentations latentes des nœuds du graphe. L'objectif de la factorisation est de trouver U et W de manière à minimiser l'erreur entre $\mathbf{M}$ et $U \cdot W^T$, et ce problème est souvent formulé comme un problème d'optimisation :

$$\min_{U,W} \|\mathbf{M} - U \cdot W^T\| \tag{2.7}$$

High-Order Proximity preserved Embedding (HOPE) (Ou et al. 2016) est une méthode qui cherche à préserver les proximités d'ordre supérieur dans les graphes dirigés en utilisant une matrice de similarité $\mathbf{M}$ entre les nœuds définie par des mesures telles que la similarité de Katz. HOPE factorise la matrice $\mathbf{M} \approx U \cdot W^T$ où les matrices U et W contiennent les plongements des nœuds sources et cibles respectivement, capturant ainsi des dépendances dans le graphe. HOPE est particulièrement adapté aux graphes orientés et préserve les similarités d'ordre supérieur, mais il est limité en termes de passage à l'échelle pour de très grands graphes en raison des coûts de calcul élevés associés à la factorisation de matrices denses. Contrairement à HOPE qui se concentre sur la préservation de proximités d'ordre supérieur dans des graphes dirigés, Asymmetric Transitivity Preserving Graph Embedding (ATPG) (Ou et al. 2016) est spécifiquement optimisé pour les relations asymétriques transitives. En considérant que si un nœud u est lié à un nœud v, et que v est lui-même lié à un nœud w, alors une relation indirecte (transitive) peut exister entre u et w. Dans ce cas, la distance entre u et w est réduite dans l'espace de représentation pour refléter cette transitivité. ATPG capture ces relations asymétriques grâce aux proximités de type Katz ou Rooted PageRank.

Graph Representation (GraRep) (Cao et al. 2015) est une autre méthode de factorisation qui se concentre sur la préservation des proximités d'ordre k dans les graphes non orientés. Pour chaque valeur k, GraRep construit une matrice de transition $\mathbf{M}^k$ représentant les relations de k-ème voisinage, puis applique une factorisation de cette matrice pour extraire des vecteurs de plongement U_k capturant les interactions locales et les motifs jusqu'à k sauts de distance dans le graphe. Les vecteurs finaux sont obtenus par concaténation des vecteurs $U_1, \ldots, U_k$, ce qui permet de capturer des similarités locales et de plus longue portée. GraRep est toutefois limité par la nécessité de choisir une valeur de k adaptée, ainsi que par sa faiblesse de passage à l'échelle pour les graphes de grande taille.

Collective Matrix Factorization (CMF) (Zhao et al. 2015) étend la factorisation matricielle classique en prenant en compte plusieurs matrices $\mathbf{M}_i$, où chaque matrice représente un type de relations dans le graphe. CMF factorise ces matrices de manière conjointe en partageant des vecteurs latents pour chaque nœud afin d'apprendre une représentation cohérente des nœuds dans un espace de faible dimension. CMF est particulièrement adapté pour capturer la similarité entre les nœuds à travers plusieurs relations dans les graphes hétérogènes. Cependant, la méthode devient très coûteuse lorsque le nombre de relations ou la taille des matrices augmente.

Il existe d'autres méthodes basées sur la factorisation spectrale (SLE (Gong et al. 2014), NSHLRR (Yin et al. 2016)), qui décomposent le laplacien normalisé du graphe $\mathbf{L} = \mathbf{D} - \mathbf{M}$, où $\mathbf{D}$ est la matrice de degré (diagonale) et $\mathbf{M}$ est la matrice d'adjacence, afin d'extraire les vecteurs propres associés aux plus petites valeurs propres de $\mathbf{L}$. Ces vecteurs propres représentent des nœuds dans un espace euclidien, préservant ainsi des propriétés globales de connectivité du graphe. Les méthodes spectrales sont adaptées pour les graphes non orientés et permettent de capturer des motifs globaux. Cependant, elles sont coûteuses en calcul pour des graphes de grande taille.

Les méthodes basées sur la factorisation matricielle sont efficaces pour capturer des relations globales dans le graphe, en particulier pour les graphes denses et de taille modérée. Elles permettent souvent de préserver des similarités structurelles importantes et sont relativement faciles à interpréter. Cependant, elles peuvent devenir coûteuses en mémoire et en temps de calcul pour les très grands graphes, et peuvent ne pas capturer efficacement les

informations de contexte local.

Méthodes basées sur la marche aléatoire

Les méthodes de plongement de graphe basées sur les marches aléatoires exploitent des parcours aléatoires dans le graphe pour capturer des relations structurelles entre les nœuds. Ces méthodes ont pour idée centrale de générer des marches aléatoires sur le graphe pour créer un contexte local autour de chaque nœud. En utilisant des techniques d'apprentissage de type Skip-Gram ou CBOW (Mikolov et al. 2013) (utilisées dans le traitement du langage naturel), les nœuds qui apparaissent fréquemment ensemble dans ces contextes créés sont projetés dans des vecteurs ayant des représentations proches dans l'espace de plongement.

DeepWalk (Perozzi et al. 2014) est l'une des premières méthodes à appliquer des marches aléatoires pour le plongement de graphes. Elle génère des marches aléatoires de longueur fixe pour générer des phrases de nœuds sur un graphe non orienté et non pondéré. Après la marche aléatoire, le graphe est traité comme un corpus textuel, et les vecteurs de plongement sont appris à l'aide de l'algorithme Word2Vec (Mikolov et al. 2013) pour maximiser la probabilité d'observer des nœuds voisins d'un nœud cible. Étant donné un nœud cible v dans un graphe, la probabilité de générer un nœud voisin u est définie par :

$$P(u|v) = \frac{\exp(\mathbf{h}_u \cdot \mathbf{h}_v)}{\sum_{w \in V} \exp(\mathbf{h}_w \cdot \mathbf{h}_v)} \tag{2.8}$$

où $\mathbf{h}_u$ est le vecteur de plongement du nœud u.

Pour optimiser ce calcul, DeepWalk utilise un softmax qui permet d'accélérer le calcul de la fonction softmax. La probabilité $P(u|v)$ devient la probabilité de traverser un chemin spécifique de la racine jusqu'à u dans l'arbre, réduisant ainsi le nombre de calculs. Formellement, si $\mathtt{path}(u)$ est le chemin de la racine à u et chaque nœud w sur ce chemin a une probabilité associée $P(w|v)$, alors :

$$P(u|v) = \prod_{w \in \mathtt{path}(u)} P(w|v) \tag{2.9}$$

où $P(w|v) = \sigma(\mathbf{h}_w^\intercal \cdot \mathbf{h}_v)$ et $\sigma(\cdot)$ représente la fonction sigmoïde. DeepWalk préserve la proximité locale et les similarités structurelles de voisinage, mais est limité par sa capacité à capturer des relations de haut niveau et n'est pas optimal pour les graphes pondérés ou orientés.

Node2vec (Grover and Leskovec 2016) est une extension de DeepWalk qui se décline en deux principales étapes. La première étape consiste à faire des marches aléatoires biaisées sur le graphe pour générer des séquences de nœuds (comme des phrases), et la deuxième étape applique un modèle Skip-Gram pour apprendre des représentations vectorielles pour chaque nœud.

La marche aléatoire biaisée utilise deux hyperparamètres, p et q, pour permettre de choisir entre une exploration locale ou une exploration plus globale du graphe. Le paramètre p favorise l'exploration du graphe en influençant la probabilité de revisiter des nœuds,

tandis que le paramètre q favorise l'exploration en augmentant la probabilité de visiter des voisins non immédiats dans le graphe. Cette flexibilité permet à Node2vec de capturer des similarités structurelles à la fois locales et globales dans des graphes non orientés et pondérés. Plus formellement, lorsque Node2vec effectue une marche aléatoire à partir du nœud courant v pour de se déplacer vers un voisin x, il utilise un facteur de biais $\alpha_{pq}(t, x)$ qui permet d'ajuster la probabilité de transition vers x en tenant compte du nœud précédent t. La probabilité de transition est proportionnelle à :

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx} \tag{2.10}$$

où w_{vx} est le poids de l'arête entre v et x (souvent 1 dans un graphe non pondéré), et $\alpha_{pq}(t, x)$ est le facteur de biais défini par :

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{si } d_{tx} = 0 \quad \text{(retour vers le nœud précédent)} \\ 1 & \text{si } d_{tx} = 1 \quad \text{(voisin du nœud précédent)} \\ \frac{1}{q} & \text{si } d_{tx} = 2 \quad \text{(nœud éloigné du précédent)} \end{cases} \tag{2.11}$$

Une fois que Node2Vec a généré des séquences de nœuds à partir des marches aléatoires biaisées, il applique le modèle Skip-Gram, emprunté au traitement du langage naturel, pour apprendre les vecteurs de plongement des nœuds. L'objectif de Skip-Gram est, à partir d'un nœud cible, prédire ses nœuds de contexte (ceux proches dans la marche aléatoire). L'objectif est alors de maximiser la probabilité de voir les nœuds de contexte u_c autour d'un nœud cible u, selon une fenêtre de contexte de taille w. Ainsi, pour chaque paire cible-contexte, la probabilité est données par une fonction softmax :

$$P(u_c \mid u) = \frac{\exp(\mathbf{h}_{u_c} \cdot \mathbf{h}_u)}{\sum_{v \in V} \exp(\mathbf{h}_V \cdot \mathbf{h}_u)} \tag{2.12}$$

Le problème avec le softmax est qu'il nécessite de calculer la probabilité sur tous les nœuds du graphe, ce qui est très coûteux pour les grands graphes. L'échantillonnage négatif simplifie cela en reformulant la tâche en un problème de classification binaire. La fonction de perte pour une paire cible-contexte est alors définie par :

$$\mathcal{L}_{u,u_c} = \log P(u_c \mid u) + \sum_{k=1}^{K} \mathbb{E}_{u_{neg} \sim P_n(u)} \left[\log P(u_{neg} \mid u) \right] \tag{2.13}$$

où $P_n(u)$ la distribution de probabilité utilisée pour échantillonner les nœuds négatifs. Malgré ses bonnes performances générales, Node2vec n'est pas adapté pour les graphes orientés, et le choix des paramètres p et q peut influencer considérablement les résultats du plongement.

Contrairement à DeepWalk et Node2vec qui appliquent les marches aléatoires sur le graphe d'origine, HARP (Chen et al. 2018a) simplifie le graphe en introduisant une structure hiérarchique pour capturer les relations entre les nœuds à différents niveaux d'abstraction du graphe. Cette approche hiérarchique permet de capturer des informations globales au

niveau des groupes de nœuds, rendant l'apprentissage des représentations plus robuste et capable de mieux généraliser, en particulier pour des graphes de grande taille ou complexes. Meta-Graph2Vec (Zhang et al. 2018) est adapté pour des graphes hétérogènes en se basant sur des métagraphes, qui sont des motifs spécifiques dans un graphe, définissant des chemins types ou des relations particulières entre différents types de nœuds.

Méthodes basées sur les réseaux de neurones

Les méthodes de plongement de graphe basées sur les réseaux de neurones permettent d'apprendre des représentations vectorielles des nœuds en tenant compte de la structure du graphe. En général, l'apprentissage repose sur un processus d'agrégation des informations des voisins directs dans le graphe. Les réseaux de neurones pour graphe permettent ainsi de propager et d'agréger les informations des voisins pour obtenir des vecteurs qui capturent la structure locale autour de chaque nœud.

Introduit pour la première fois par Thomas Kipf et Max Welling en 2017, les réseau de neurones pour les graphes, ou en anglais Graph Convolutional Networks (GCN) (Kipf and Welling 2017) généralisent les CNNs pour les graphes non orientés et pondérés. Pour chaque couche l, un GCN est formalisé par l'opération d'agrégation suivante :

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}) \tag{2.14}$$

où $\tilde{A} = A + I$ est la matrice d'adjacence normalisée contenant les auto-connections, $\tilde{D}$ est la matrice de degré correspondante, $H^{(l)}$ est la matrice des représentations des nœuds à la couche l, $W^{(l)}$ est la matrice des poids de la couche l et σ est une fonction d'activation. GCN préserve la proximité locale en agrégeant les informations de voisinage direct à chaque couche. Cependant, cette méthode peut souffrir de sur-lissage lorsque le nombre de couches augmente, perdant les informations spécifiques aux nœuds. Dans le but d'étendre GCN pour les graphes de grande taille, Graph SAmple and aggreGatE (GraphSAGE) (Hamilton et al. 2017) introduit une approche d'échantillonnage inductive qui permet le traitement des grands graphes de manière plus efficace. Au lieu de calculer les représentations pour tous les nœuds et leurs voisins, GraphSAGE utilise un échantillonnage stochastique des voisins pour chaque nœud, ce qui réduit les coûts de calcul et de mémoire. De plus, GraphSAGE propose plusieurs fonctions d'agrégation, comme la moyenne, la convolution par réseaux récurrent de type Long Short-Term Memory (LSTM) ou la pondération basée sur des poids appris pour capturer des propriétés variées, bien que celles-ci peuvent affecter significativement la performance. MixHop (Abu-El-Haija et al. 2019) est une extension des GCNs qui empile plusieurs convolutions de différents ordres pour capturer des relations globales dans le graphe. Dans MixHop, chaque couche de convolution extrait des informations non seulement des voisins directs mais aussi des voisins d'ordre supérieur en appliquant des puissances multiples de la matrice d'adjacence. Ces informations sont ensuite concaténées pour obtenir des représentations plus riches. Dense Graph Convolutional Network (DGCN) intègre à la fois la matrice d'adjacence et la matrice de similarité, ce qui permet de capturer non seulement les connexions structurelles mais aussi les similarités entre nœuds non connectés.

Les Graph Attention Networks (GAT) (Veličković et al. 2018) introduisent un mécanisme d'attention dans les réseaux de neurones pour graphes, permettant de pondérer les contri-

butions des voisins dans le calcul de la représentation d'un nœud. Cette approche adapte l'architecture des réseaux de neurones pour capter de manière plus précise l'importance relative de chaque voisin dans le graphe, sans nécessiter de connaissance préalable sur la structure globale du graphe. Dans un GAT, pour chaque nœud v et chacun de ses voisins $u \in \mathcal{N}(v)$, un score d'attention α_{vu} est calculé pour déterminer le poids de la contribution du voisin u dans la nouvelle représentation de v :

$$\alpha_{vu} = \frac{\exp(\text{LeakyReLU}(a^T[W\mathbf{h}_v \,\|\, W\mathbf{h}_u]))}{\sum_{k\in\mathcal{N}(v)} \exp(\text{LeakyReLU}(a^T[W\mathbf{h}_v \,\|\, W\mathbf{h}_k]))} \tag{2.15}$$

où α_{vu} est le coefficient d'attention entre les nœuds v et u, a^T est le vecteur de poids pour l'attention, W est la matrice de poids et $\|$ représente la concaténation des vecteurs. L'équation d'agrégation pour un nœud v devient alors :

$$\mathbf{h}_v = \sigma\left(\sum_{u\in\mathcal{N}(v)} \alpha_{vu} \cdot W\mathbf{h}_u\right) \tag{2.16}$$

où σ est une fonction d'activation et W est la matrice de poids. GAT s'applique aux graphes homogène non orientés et pondérés, préservant les relations locales entre les nœuds, mais il est limité par sa complexité pour de grands graphes, ce qui complique son application dans la pratique. Pour prendre en compte les graphe dynamiques, Les Spatial Temporal Graph Attention Networks (ST-GAT) (Song et al. 2022) ajoutent une dimension temporelle aux GATs. ST-GAT combine l'attention spatiale, pour capturer les relations structurelles entre nœuds, et l'attention temporelle, qui intègre l'évolution de ces relations dans le temps. Les Graph Transformer Networks (GTN)s (Yun et al. 2020) s'inspirent de l'architecture des transformers pour introduire une attention globale dans les graphes. Contrairement aux GATs classiques qui limitent l'attention aux voisins directs, les GTN utilisent des mécanismes d'attention permettant de capturer des dépendances à longue portée entre les nœuds.

Certaines méthodes basées sur les réseaux de neurones utilisent des autoencodeurs (Chen and Guo 2023) pour apprendre des plongements de graphes. Dans ce contexte, les autoencodeurs permettent de capturer à la fois les proximités locales et les structures globales. Structural Deep Network Embedding (SDNE) est une méthode d'autoencodeur qui minimise l'erreur de reconstruction tout en ajoutant un terme de régularisation pour préserver les proximités de premier et de second ordre. Sa fonction de coût est définie par :

$$\sum_{(u,v)\in E} \|f(u) - f(v)\|^2 + \lambda \sum_{(u,v)\in E} \|g(f(u)) - f(v)\|^2 \tag{2.17}$$

où f représente l'encodeur et g représente le décodeur. Cette méthode est efficace pour les graphes non orientés et préserve la structure locale des nœuds connectés, mais elle peut être limitée par sa complexité pour des graphes de grande taille. D'autres autoencodeurs, tels que les Variational Graph Auto-Encoders (VGAE) (Kipf and Welling 2016) adoptent une approche variationnelle, en modélisant une distribution latente pour chaque nœud au lieu d'un vecteur. Cela permet de capturer l'incertitude dans les relations entre nœuds et de modéliser des graphes dans lesquels les liens peuvent être partiellement observés.

Méthodes basées sur la reconstruction des arêtes

Les méthodes basées sur la reconstruction des arêtes (ou relations) apprennent des représentations en se basant sur un graphe sémantique, c'est-à-dire un graphe structuré sous forme de triplet (h, r, t). Le tout premier algorithme basé sur la reconstruction des arêtes est TransE (Bordes et al. 2013) dont l'objectif est que le vecteur du nœud t soit proche du vecteur de h déplacé par le vecteur r. Formellement, TransE a pour objectif de minimiser la distance :

$$f(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2 \tag{2.18}$$

TransE est adapté pour les graphes simples, où chaque nœud source est lié à un seul nœud cible via chaque relation. Par contre, il est moins performant pour les relations où un nœud peut être lié à plusieurs autres nœuds dans le graphe via la même relation. Pour prendre en compte cette limite, TransH (Wang et al. 2014) introduit un hyperplan spécifique à chaque relation r. Les vecteurs des nœuds source et cible sont projetés sur cet hyperplan avant d'appliquer la translation, permettant de modéliser des relations plus complexes. TransH a pour objectif de minimiser la distance :

$$f(h, r, t) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|_2 \tag{2.19}$$

où $\mathbf{h}_\perp$ et $\mathbf{t}_\perp$ sont les projections des vecteurs de h et t sur l'hyperplan de la relation r. La projection sur des hyperplans permet de mieux capturer les propriétés complexes des relations tout en maintenant une efficacité de temps de calcul comparable à celle de TransE.

Toujours pour améliorer la représentation des entités et des relations, TransR (Zhang et al. 2021b) propose de modéliser les entités et les relations dans des espaces vectoriels distincts. TransR utilise un espace commun pour toutes les entités, et des espaces distincts pour chaque relation afin de capturer les nuances spécifiques à chaque type de relation. Lors de l'apprentissage, les entités sont d'abord projetées depuis l'espace des entités vers l'espace spécifique à la relation correspondante, puis une translation est effectuée dans cet espace projeté. Cette approche permet de mieux capturer les interactions complexes entre les entités et leurs relations. La distance à minimiser dans TransR est :

$$f(h, r, t) = \|M_r\mathbf{h} + \mathbf{r} - M_r\mathbf{t}\|_2 \tag{2.20}$$

où M_r est une matrice de projection propre à la relation r, permettant de passer de l'espace des entités à l'espace de relation. TransR est adapté aux graphes hétérogènes, et est plus flexible pour représenter les relations asymétriques. Toutefois, chaque relation ayant sa propre matrice de projection, TransR nécessite plus de mémoire et de calculs.

Contrairement à TransR, qui utilise une matrice de projection unique pour chaque relation afin de projeter les entités dans un espace spécifique à la relation, TransD (Ji et al. 2015) adopte une approche plus dynamique. Pour chaque triplet (h, r, t), TransD définit deux vecteurs de projection distincts : $\tilde{\mathbf{r}}_r$ qui est un vecteur de projection associé à la relation r, $\tilde{\mathbf{e}}_h$ et $\tilde{\mathbf{e}}_t$ qui sont des vecteurs de projection associés respectivement aux entités h et t. Ces vecteurs sont utilisés pour construire deux matrices de projection $M_{h,r} = \tilde{\mathbf{r}}_r\tilde{\mathbf{e}}_h^\top + \mathbf{I}^{d\times k}$ et

$M_{t,r} = \tilde{\mathbf{r}}_r \tilde{\mathbf{e}}_t^T + \mathbf{I}^{d \times k}$ qui permettent de projeter les entités h et t dans un espace spécifique à la relation r, facilitant ainsi la modélisation de relations complexes. L'objectif est de minimiser la fonction de perte suivante :

$$f(h, r, t) = \|(M_{h,r}\mathbf{h} + \mathbf{r}) - (M_{t,r}.\mathbf{t})\|_2 \tag{2.21}$$

ComplEx (Trouillon et al. 2016) utilise des nombres complexes pour représenter les nœuds et les relations, ce qui permet de modéliser des relations asymétriques et directionnelles dans le graphe. La fonction de perte est définie comme le produit hermitien[2] des vecteurs :

$$f(h, r, t) = Re(\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle) \tag{2.22}$$

ComplEx représente des relations subtiles et des dépendances complexes grâce à l'utilisation de l'espace complexe. Cependant, l'utilisation de nombres complexes peut rendre le modèle plus difficile à interpréter et à implémenter. RotatE (Sun et al. 2018b) utilise également l'espace complexe pour représenter les relations comme des rotations. Cette rotation est réalisée par un produit Hadamard avec le vecteur relation r de manière à modéliser des relations directionnelles et symétriques :

$$f(h, r, t) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|_2 \tag{2.23}$$

où $\circ$ représente le produit Hadamard.

2.4 Conclusion

Ce chapitre a exploré les fondements des réseaux de neurones artificiels appliqués à la vision par ordinateur, en mettant en lumière les différentes architectures utilisées pour la classification d'images. Nous avons d'abord présenté les modèles neuronaux classiques, du perceptron aux réseaux de neurones convolutifs, en passant par les architectures profondes adaptées aux tâches de classification supervisée. Nous avons ensuite présenté le concept de compilation de connaissances qui est utilisé dans notre première contribution pour créer des jeux de données d'images pour la classification, intégrant des connaissances. Nous avons enfin abordé les méthodes de plongement de graphes, qui jouent un rôle central pour la présentation de nos différentes contributions. Nous avons présenté les principales approches permettant de transformer un graphe dans un espace vectoriel, exploitable par les algorithmes d'apprentissage profond.

2. Le produit hermitien est une généralisation du produit scalaire pour les espaces vectoriels complexes, défini entre deux vecteurs x et y comme $\langle x, y \rangle = \sum x_i \overline{y_i}$, où $\overline{y_i}$ est le conjugué complexe de y_i, assurant que le produit est linéaire et que le résultat est un nombre complexe.

Chapitre 3

Intégration de connaissances dans les architectures profondes en vision par ordinateur

L'essor des architectures profondes a transformé la vision par ordinateur en permettant d'apprendre automatiquement des représentations complexes à partir de grandes quantités de données. L'utilisation de connaissances a priori est une approche de plus en plus utilisée qui permet de rendre ces modèles plus robustes et plus explicables. Ce défi soulève plusieurs questions, notamment sur la définition de la connaissance dans le contexte de la vision par ordinateur : sous quelle forme elle est représentée et comment l'intégrer efficacement dans les modèles. L'objectif de ce chapitre est, dans un premier temps, de clarifier ces notions en posant une définition de la connaissance dans le cadre de la vision par ordinateur. Nous passerons ensuite en revue les principales approches de la littérature qui cherchent à intégrer des connaissances dans les architectures profondes.

Contents

3.1 Préliminaires

Bien que les architectures profondes aient démontré un immense potentiel en vision par ordinateur, il existe de nombreuses situations où les approches basées uniquement sur les données atteignent leurs limites ou produisent des résultats insuffisants. Cela se manifeste notamment lorsque les données disponibles sont insuffisantes pour entraîner des modèles efficaces et généralisables. De plus, un modèle purement axé sur les données pourrait ne pas satisfaire certaines contraintes, telles que des lois naturelles ou des normes de sécurité et de conformité. Ces problèmes ont conduit à l'exploration de nouvelles approches qui combinent l'apprentissage automatique avec des connaissances préexistantes. Ces connaissances constituent ainsi une ressource précieuse dans les situations dans lesquelles les données seules se révèlent insuffisantes, peu fiables ou limitées, permettant d'améliorer la robustesse, la précision, et parfois même l'explicabilité du modèle.

3.1.1 Types de connaissances

Contrairement à l'intelligence artificielle où la notion de connaissance fait généralement référence à des règles, celle-ci est difficile à définir de façon universelle dans le contexte de l'apprentissage automatique, et plus particulièrement en vision par ordinateur. De manière générale, la connaissance désigne toute information externe, structurelle ou contextuelle, introduite dans un modèle afin de guider ou d'améliorer le processus d'apprentissage (Von Rueden et al. 2021). Cette connaissance peut être représentée sous des formes diverses et variées : règles, graphes, tableaux de données, fonctions (représentant des lois physiques par exemple), annotations visuelles, images (résultats de simulations par exemple), ou encore des intuitions expertes propres à un domaine d'application.

3.1.2 Intégration des connaissances

Pour intégrer les connaissances dans les architectures profondes, plusieurs approches ont été explorées, dépendant de la façon dont ces connaissances sont formalisées. Nous nous appuyons dans ce chapitre sur la taxonomie proposée dans la synthèse de Von Rueden et al. (2021) pour présenter ces méthodes.

Une approche courante pour intégrer des connaissances dans les architectures profondes consiste à les injecter directement dans les données d'entraînement. Cette stratégie répond en grande partie à la contrainte fréquente liée au manque de données annotées, notamment dans des domaines sensibles comme la vision médicale, où la collecte d'exemples étiquetés est coûteuse ou difficile. Dans ce cadre, les connaissances sont considérées comme des données additionnelles ou des représentations implicites. Les données additionnelles, souvent disponibles en plus des données d'entraînement sous la forme de tableaux structurées (Grzeszczyk et al. 2023), de textes (Zhang et al. 2017; Wang et al. 2018) ou d'images issues de simulations (Yang et al. 2019), sont souvent utilisées dans un contexte de multi-modalité. Les connaissances implicites, généralement issues d'intuitions ou d'expériences expertes, exploitent les structures latentes, les régularités statistiques ou les relations cachées dans les données d'entraînement. Elles sont généralement intégrées de manière indirecte via des

heuristiques de sélection de données afin de favoriser un apprentissage progressif et structuré des modèles. Les travaux qui exploitent les connaissances implicites s'appuient fréquemment sur des stratégies comme le curriculum learning (Bengio et al. 2009). Dans cette approche, les données sont organisées selon un ordre de complexité croissante (Haarburger et al. 2019; Wei et al. 2021; Maicas et al. 2019) ou décroissante (Wu et al. 2023), déterminé par des critères définis a priori (Jiménez-Sánchez et al. 2019; Haarburger et al. 2019; Wei et al. 2021) ou appris (Maicas et al. 2019; Jesson et al. 2017), afin de guider le modèle dans un parcours d'apprentissage qui reflète une pédagogie humaine. Dans ce cas, les connaissances influencent la façon dont les modèles extraient, interprètent et priorisent les informations dans les premières phases de l'apprentissage.

Une deuxième approche pour intégrer les connaissances dans les architectures profondes consiste à adapter l'architecture des réseaux afin d'y imposer des contraintes basées sur les connaissances préalables. Cette adaptation peut se traduire par l'introduction de modules spécialisés, par exemple des blocs de raisonnement (Wang et al. 2016a; Zhang et al. 2020), des mécanismes d'attention guidée (Guan et al. 2018; Wang et al. 2020b; Li et al. 2019; Mitsuhara et al. 2019) ou des structures de graphe (Chen et al. 2020a; Krichen et al. 2022). Ces modules spécialisés reflètent une compréhension experte et les connaissances sont souvent formalisées sous forme de graphe de connaissances, d'images issues de simulations ou d'intuition expertes. L'intégration de connaissances repose ainsi sur des hypothèses explicites quant à la nature des données ou aux relations sous-jacentes que le modèle devrait capturer. Par ailleurs, l'intégration peut également se faire à travers des hyper-paramètres, ceci en guidant leur choix ou leur adaptation selon des principes fondés sur l'expertise du domaine ou sur des tâches plus générales. C'est notamment le cas dans les approches d'apprentissage par transfert (Tan et al. 2018), où l'on cherche à réutiliser des connaissances acquises dans un contexte source (modèle pré-entraîné, initialisation spécifique, congélation de couche, etc.) pour améliorer la performance dans une tâche cible. Cette intégration permet d'incorporer implicitement des informations spécifiques au domaine, comme des distributions de caractéristiques ou des structures complexes, en améliorant la convergence du modèle ou en réduisant le besoin de données massives pour l'entraînement.

Une autre approche pour intégrer les connaissances dans les architectures profondes consiste à les intégrer directement dans l'algorithme d'apprentissage, en modifiant la fonction de perte à l'aide de termes de régularisation ou de pertes sémantiques. Cette approche permet de contraindre explicitement le comportement du modèle afin qu'il respecte certaines contraintes, règles ou relations issues des connaissances préexistantes, qu'elle soit physique, logique ou statistique. Par exemple, des termes de perte spécifiques peuvent être utilisées pour imposer le respect de règles logiques (Diligenti et al. 2017), des contraintes (Rieger et al. 2020) ou de lois physiques (équations différentielles) (Stewart and Ermon 2016), assurant ainsi que les prédictions du modèle restent cohérentes avec des principes fondamentaux. Les connaissances peuvent également permettre de créer des régularisations correspondantes à des connaissances variées telles que des graphes ou des régularités géométriques dans les classes (Jiang et al. 2018). De cette manière, le modèle apprend non seulement à minimiser l'erreur de ses prédictions, mais aussi à maximiser son adhérence aux connaissances, renforçant sa fiabilité et sa précision dans des contextes critiques où les connaissances expertes jouent un rôle déterminant.

Une dernière approche courante pour intégrer des connaissances dans les architectures profondes consiste à ajuster ou valider les prédictions du modèle a posteriori en les confron-

tant à des connaissances préexistantes. Dans cette stratégie, le modèle effectue d'abord une prédiction sur la base d'un apprentissage statistique, puis cette sortie est corrigée, filtrée ou validée à l'aide de contraintes dérivées de règles (Li et al. 2024; Ledaguenel et al. 2024) (souvent dans un cadre neuro-symbolique) ou de graphes (Fang et al. 2017; D'souza et al. 2023; Meurie et al. 2018) pour exploiter les relations sémantiques entre des objets dans un cadre de détection d'objets dans les images. L'objectif est d'assurer que les prédictions restent cohérentes avec des principes établis, que ces principes soient appris directement à partir des données d'entraînement ou pas.

Ces différentes approches montrent la diversité des stratégies pour exploiter les connaissances dans les architectures profondes. Dans le cadre de nos travaux de recherche, nos différentes contributions s'inscrivent principalement dans deux de ces stratégies : l'intégration de connaissances dans l'architecture et dans l'algorithme d'apprentissage. La section suivante est consacrée à une présentation plus détaillée des travaux existants dans ces deux directions et au positionnement de nos contributions par rapport à ces travaux.

3.2 Travaux existants

Dans le cadre général de la vision par ordinateur, plusieurs formulations du problème de classification peuvent être adoptées selon la nature des étiquettes associées aux images. En classification (ou classification mono-label), chaque image $x_i \in \mathcal{X}$ est associée à une unique étiquette $y_i \in \mathcal{Y}$, représentant une classe exclusive. L'objectif est d'apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ qui minimise une perte entre la prédiction $\hat{y}_i = f(x_i)$ et l'étiquette réelle y_i. En classification multi-label, chaque image peut appartenir simultanément à plusieurs classes. On cherche alors à prédire un vecteur binaire $\hat{y}_i \in \{0,1\}^{|\mathcal{Y}|}$, où chaque dimension indique la présence ou l'absence d'une classe. L'apprentissage se fait souvent via une somme de pertes binaires indépendantes. Enfin, en détection d'objets, l'objectif est double : localiser les objets présents dans l'image par des boîtes englobantes b_i et identifier leur classe. Ce problème est formalisé comme l'apprentissage d'une fonction $f : \mathcal{X} \to \{(\hat{b}_i, \hat{y}_i)\}_{i=1}^{m}$, où m est le nombre d'objets détectés, et où pour chaque boîte $\hat{b}_i$, le modèle prédit une étiquette $\hat{y}_i$. Ces trois cadres partagent une base commune, mais diffèrent dans leur formulation, leur complexité, et les stratégies d'apprentissage qu'ils mobilisent. Dans la suite de cette section, nous présenterons des travaux qui intègrent des connaissances dans ces différents cadres de vision par ordinateur.

3.2.1 Intégration dans l'architecture

Utilisation des blocs de raisonnement

Wang et al. (2016a) introduit un cadre pour la classification d'images à étiquettes multiples, appelé CNN-RNN, qui combine un réseau de neurones convolutif (CNN) et un réseau de neurones récurent (Recurrent Neural Network (RNN)) dans une architecture unifiée et entraînée de bout en bout (Figure 3.1). Le modèle prend en entrée une image $x \in \mathbb{R}^{H \times W \times 3}$, et génère en sortie une séquence ordonnée d'étiquettes $(y_1, \ldots, y_k)$ ou chaque $y_t \in \mathcal{Y}$ représente une étiquette associée à l'image x, sélectionné dans un espace de classes

de taille C. Contrairement aux approches classiques de classification multi-label qui traite chaque label de façon indépendante via des classifieurs binaires, CNN-RNN exploite un réseau récurrent de type LSTM (Graves and Graves 2012) pour exploiter explicitement les dépendances de co-occurrence entre les labels dans les images. La connaissance ici est présente dans les données d'entraînement, et celle-ci est apprise automatiquement pendant l'apprentissage du modèle grâce à la dynamique du LSTM.

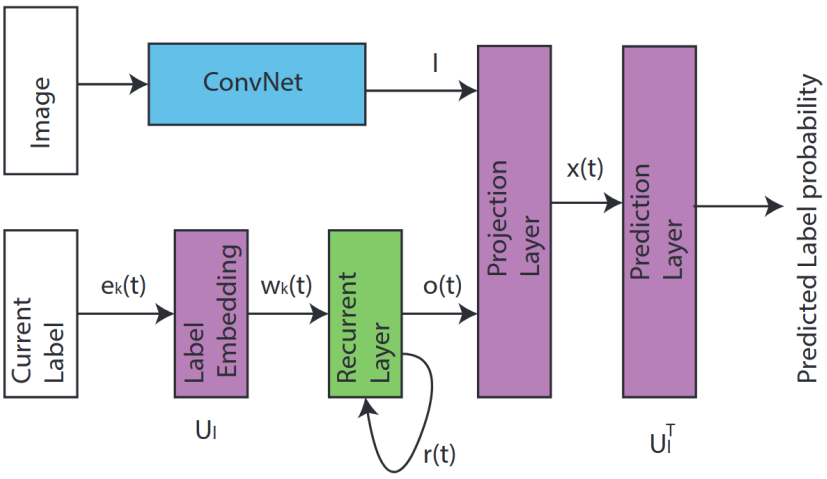


FIGURE 3.1 – Architecture CNN-RNN. Source : Wang et al. (2016a).

CNN-RNN décompose la prédiction multi-label comme une tâche de génération séquentielle, où chaque séquence d'étiquettes est interprétée comme un chemin de prédiction ordonné, dont la probabilité est calculé comme un produit de probabilités conditionnelles. Formellement, le modèle cherche à résoudre le problème de maximisation :

$$(y_1, \ldots, y_k) = \arg \max_{y_1,\ldots,y_k} \prod_{t=1}^{k} P(y_t \mid I, y_1, \ldots, y_{t-1}) \tag{3.1}$$

où $I \in \mathbb{R}^{d_I}$ est la représentation de l'image x encodée à partir du CNN. Les étiquettes sont représentée par des vecteurs one-hot $e_{y_i} \in \mathbb{R}^C$, projetés dans un espace de plongement de dimension réduite par une matrice $U_l \in \mathbb{R}^{d \times C}$, donnant un vecteur $w_{y_i} \in \mathbb{R}^d$, défini par :

$$w_{y_i} = U_l e_{y_i} \tag{3.2}$$

Le LSTM modélise ensuite la relation image/label et l'inter-dépendance entre les labels en prenant comme entrée à chaque temps t le plongement de l'étiquette prédite au pas de temps précédent et produit un état caché $r(t)$, ainsi qu'une sortie $o(t)$ via les fonctions non linéaires :

$$o(t) = h_o(r(t-1), w_{y_i}(t)), \quad r(t) = h_r(r(t-1), w_{y_i}(t)) \tag{3.3}$$

où $w_{y_i}(t)$ représente le plongement du $t^{\text{ème}}$ label dans le chemin de prédiction, et h_o, h_r sont des fonctions non linéaires du LSTM. La reprsentation I de l'image x et la sortie $o(t)$

du LSTM sont projetées dans le même espace latent à l'aide des matrices U_I^x et U_o^x, pour obtenir une représentation conjointe :

$$x(t) = h(U_I^x I + U_o^x o(t)) \in \mathbb{R}^d \tag{3.4}$$

Le score de compatibilité entre la représentation conjointe $x(t)$ et chaque étiquette y_i est donné par :

$$s_{y_i}(t) = x(t)^\top w_{y_i} = x(t)^\top (U_l e_{y_i}). \tag{3.5}$$

Un softmax est appliqué sur tous les scores pour obtenir la distribution de probabilité des labels au temps t. L'inférence repose sur la recherche du chemin de labels le plus probable. Étant donné que le premier label prédit peut être erroné, il sera très probable que la séquence entière ne puisse pas être correctement prédite. Pour remédier à ce problème, les auteurs utilisent une recherche en faisceau (*beam search*) afin d'identifier le chemin de prédiction le plus probable. Cette architecture permet d'apprendre une représentation capable de capturer à la fois la corrélation entre les labels et leur dépendance contextuelle par rapport au contenu visuel de l'image. L'intégration de la connaissance se fait de manière implicite, en apprenant automatiquement les régularités de co-occurrence dans les données d'entraînement via le LSTM, sans représentation explicite ou formalisée des relations sémantiques entre les labels.

Dans le cadre de nos travaux, notre deuxième contribution adopte une logique similaire, où nous tirons également parti des dépendances sémantiques. Toutefois, notre objectif ne porte pas sur la modélisation de la dépendance entre les classes comme dans cadre multi-label, mais sur les relations entre les attributs descriptifs qui caractérisent les classes. Ces attributs, bien que extraits à partir des données d'entraînement, permettent de mieux capturer les variations intra-classes et inter-classes, ce qui répond à des défis différents.

Zhang et al. (2020) propose un réseau de classification pour la reconnaissance d'actions dans les vidéos, appelé KINet (Knowledge Integration Networks). Ce réseau intègre de manière explicite et hiérarchique des connaissances humaines et contextuelles à travers une architecture à trois branches. Le modèle combine une branche principale dédiée à la reconnaissance d'actions avec deux branches auxiliaires, guidée par des réseaux enseignants, respectivement pour la segmentation humaine et la reconnaissance de scènes (Figure 3.2). KINet vise ainsi à intégrer le contexte de la scène et la connaissance humaine dans la reconnaissance de l'action humaine.

Le modèle prend en entrée une vidéo $V = \{s_1, \ldots, s_{N_{seg}}\}$ constituée de segments temporels s_i, et prédit une classe d'action $y \in \mathcal{Y}$. Les trois branches sont entraînées conjointement via une fonction de perte multi-tâche :

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{action}} + \lambda_2 \mathcal{L}_{\text{human}} + \lambda_3 \mathcal{L}_{\text{scene}} \tag{3.6}$$

où $\mathcal{L}_{\text{action}}$ et $\mathcal{L}_{\text{scene}}$ sont des pertes d'entropie croisées, et $\mathcal{L}_{\text{human}}$ une perte de segmentation sémantique. Les tâches de reconnaissance de scène et de segmentation humaines sont supervisées grâce aux pseudo-labels fournis par les deux réseaux enseignants correspondants,

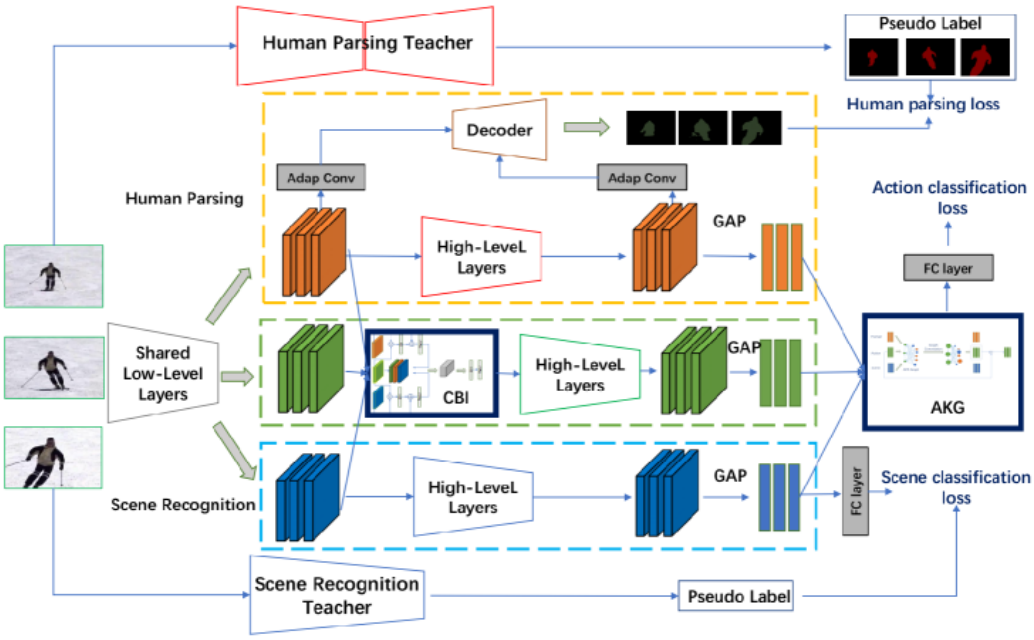


FIGURE 3.2 – Architecture du modèle KINet. Source : Zhang et al. (2020).

pré-entraînés à partir des jeux de données Places365 (Zhou et al. 2017) et LIP (Gong et al. 2017), respectivement.

La première forme d'intégration de connaissances utilise une CBI (*Cross Branch Integration*) au niveau intermédiaire du réseau. La CBI fusionne des cartes de caractéristiques intermédiaires extraites des trois branches (action, scène, humain) en effectuant une multiplication élément par élément, suivi d'une activation non linéaire et d'une convolution 1×1. Cette dernière opération de convolution permet de réduire le nombre de canaux afin de garantir l'égalité entre les canaux d'entrée et ceux de sortie de la CBI. Les cartes de caractéristiques ainsi obtenues sont utilisées dans la branche principale (Figure 3.2) afin que les cartes de caractéristiques des branches auxiliaires puissent interagir directement avec les caractéristiques d'action pour la classification.

Au niveau supérieur du réseau, un AKG (*Action Knowledge Graph*) est utilisé pour construire un graphe de connaissances après un échantillonage moyen appliqué, pour chaque segment temporel, à la sortie de chaque branche. Les vecteurs obtenus $\{h_i^{action}, h_i^{scene}, h_i^{human} \mid i = 1, \ldots, N_{seg}\}$ sont utilisés pour construire le graphe ayant pour matrice d'adjacence $G \in \mathbb{R}^{N \times N}$, où $N = 3 \times N_{seg}$ est le nombre total de nœuds dans le graphe. Chaque nœud encode ainsi une représentation d'un segment pour une modalité (action, scène ou humain). Pour créer les arêtes entre les nœuds, une normalisation par softmax est appliquée entre chaque paire de nœuds a et b :

$$G_{ab} = \frac{e^{f(h_a, h_b)}}{\sum_{b'=1}^{N} e^{f(h_a, h_b)}} \tag{3.7}$$

où $f(h_a, h_b)$ est la fonction qui calcule le score d'alignement (par exemple un produit scalaire) entre les nœuds a et b. Seules les arêtes connectées à des nœuds d'action sont conservées. Les relations scène-humain sont alors ignorées via un masque binaire $I_{mask} \in$

$\{0,1\}^{N\times N}$ tel que $G \leftarrow G \odot I_{mask}$. Enfin, la propagation des caractéristiques à travers le graphe est effectué via un réseau de convolution de graphes (GCN) (Kipf and Welling 2017), défini comme :

$$Z = \sigma(I_{mask} \odot GXW) \tag{3.8}$$

où $Z \in \mathbb{R}^{N\times d}$ est la sortie du GCN, W une matrice de poids, et σ une fonction d'activation. $X = \{h_i^{action}, h_i^{scene}, h_i^{human} \mid i = 1,\ldots,N_{seg}\}$ est la représentation des nœuds, où chaque nœud h_i^{action}, h_i^{scene}, $h_i^{human} \in \mathbb{R}^d$, avec d représentant la dimension des canaux de la dernière couche du réseau de base. Seules les représentations h_i^{action} sont conservées pour la prédiction finale à travers un classifieur. Cette propagation structuré permet à KINet d'exploiter des dépendances sémantiques issues des données, améliorant ainsi la robustesse des prédictions sans nécessiter d'annotations supplémentaires. Contrairement KINet qui extrait les connaissances au cours de l'apprentissage du modèle dans un cadre bout en bout, nos différentes contributions s'appuient des connaissances préexistantes, bien qu'elles soient elle-mêmes dérivée des données d'entraînement.

Utilisation de mécanismes d'attention

Guan et al. (2018) propose un modèle de classification multi-label des maladies du thorax à partir d'images radiographiques, appelé AG-CNN. Le modèle intègre des connaissances médicales implicites en focalisant l'apprentissage sur les régions discriminantes, simulant le comportement d'un radiologue. Il prend en entrée une image $x \in \mathbb{R}^{H\times W}$ et produit en sortie un vecteur de probabilité $\hat{y} = \tilde{p}(c \mid x) \in [0,1]^C$, où C représente le nombre de pathologies. Chaque composante $\hat{y}_c$ prédit la probabilité de présence de la pathologie c dans l'image. La méthodologie s'inspire du raisonnement clinique d'un radiologue, qui s'appuie à la fois sur une analyse globale de l'image radiologique et sur l'examen ciblé de régions anatomiques suspectes pour établir son diagnostic. L'architecture de AG-CNN exploite ces connaissances implicites pour guider le modèle vers les régions pertinentes en proposant un réseau composé de trois branches : une branche globale f^G qui traite l'image entière, une branche locale f^L qui se concentre sur une région d'intérêt déterminée automatiquement pendant l'apprentissage, et une branche de fusion f^F combinant les deux représentations (Figure 3.3).

La branche globale utilise un modèle ResNet, suivi d'un échantillonage maximal et d'une couche entièrement connectée pour prédire une sortie $\hat{y}_G$. La fonction de perte utilisée est l'entropie croisée binaire sur les étiquettes de chaque pathologie c :

$$\mathcal{L} = -\frac{1}{C}\sum_{c=1}^{C} [y_c \log \hat{y}_c + (1-y_c)\log(1-\hat{y}_c)] \tag{3.9}$$

où $y_c \in \{0,1\}$ représente la vérité terrain de la présence de la pathologie c dans l'image.

Pour guider la localisation des régions discriminantes, AG-CNN extrait une carte d'attention $H_G(a,b)$ à partir des activations finales du CNN de la branche globale, en prenant le maximum sur les différents canaux :

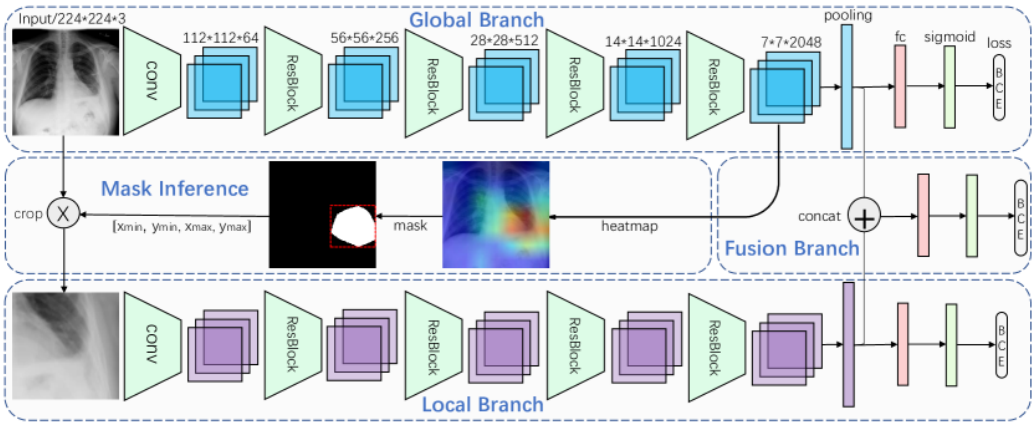


FIGURE 3.3 – Architecture du modèle AG-CNN. Source : Guan et al. (2018).

$$H_G(a,b) = \max_{k\in\{1,\dots,K\}} \left| f_k^G(a,b) \right| \tag{3.10}$$

où K représente le nombre de canaux et $f_k^G(a,b)$ représente l'activation à la position spatiale (a,b) dans le k-ième canal de la sortie de la dernière couche de convolution de la branche globale. Un masque binaire $M(a,b)$ est ensuite dérivé en appliquant un seuillage τ sur cette carte :

$$M(a,b) = \begin{cases} 1 & \text{si } H_G(a,b) > \tau \\ 0 & \text{sinon} \end{cases} \tag{3.11}$$

La plus grande région connectée du masque est encadrée pour générer l'image locale x_L, redimensionnée à la taille d'origine. la nouvelle image x_L est ensuite utilisée pour entraîner la branche locale f^L avec la même perte que la branche globale, mais avec des poids indépendants. Enfin, les sorties des couches de sous-échantillonage sont concaténées et utilisées comme entrée de la branche de fusion f^F, produisant la sortie $\tilde{p}(c \mid [x, x_L])$. L'entraînement de AG-CNN se fait en trois étapes séquentielles : fine-tuning de la branche globale (sur le jeu de données ImageNet (Deng et al. 2009)), entraînement de la branche locale à partie des région extraites et entraînement de la branche de fusion avec les branches locale et globale gelées pour ne pas modifier les poids. Contrairement à nos contributions qui utilisent des connaissances explicites, AG-CNN introduit un mécanisme d'attention pour extraire automatiquement des connaissances implicites à partir des activations du réseau. Cependant, le fait de ne pas utiliser un réseau entièrement entraîné de bout en bout ne permet pas au modèle d'optimiser conjointement la localisation des régions discriminantes et la classification, limitant ainsi sa capacité à ajuster dynamiquement les zones d'intérêt en fonction des objectifs de la tâche finale.

Toujours pour le problème de classification des maladies du thorax, mais sur des images 3D, KGZNet (Wang et al. 2020b) propose un modèle de classification multi-label à partir d'images radiologiques du thorax. L'entrée du modèle est une image CXR $x \in \mathbb{R}^{H\times W\times 3}$, et la sortie est un vecteur de probabilité $\hat{y} = [\hat{y}_1, \dots, \hat{y}_C]$, où C représente le nombre de pathologie, pour s'aligner avec la vérité terrain $y = [y_1, \dots, y_C]$, $y \in [0,1]^C$. La particularité

de KGZNet est l'intégration de connaissances médicales a priori sur l'anatomie thoracique (les maladies étant confinées aux poumons) pour guider l'apprentissage par attention multi-échelle. L'architecture repose sur trois branches CNN (GlobalNet, LungNet et LesionNet), toutes basées sur ResNet, qui extraient respectivement des représentations à partir de l'image x, de la région pulmonaire segmentée $x_r \subset x$, et d'une région localisée $x_l \subset x_r$ contenant une liaison (Figure 3.4).

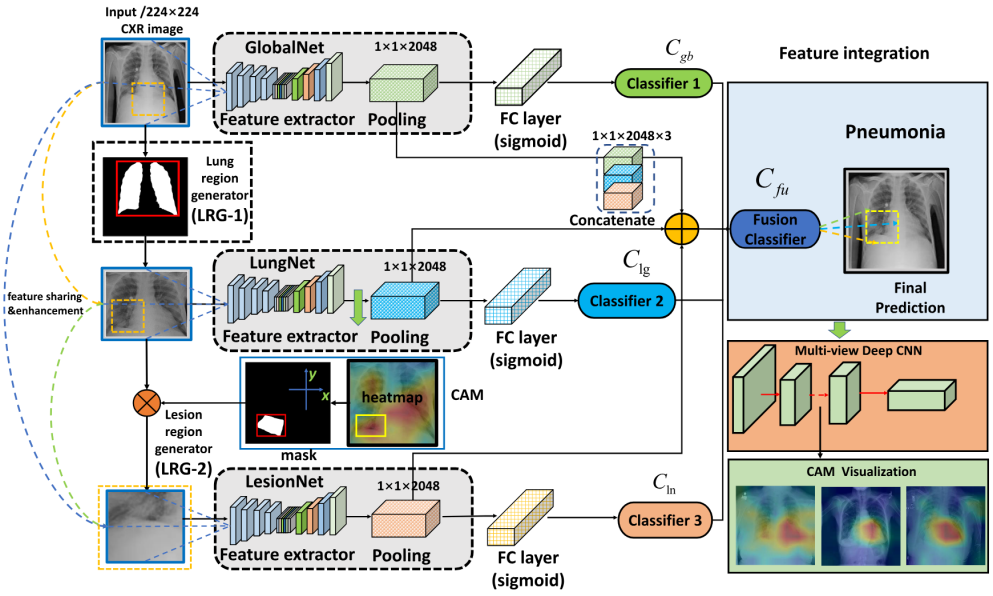


FIGURE 3.4 – Architecture du modèle KGZNet. Source : Wang et al. (2020b).

La branche GlobalNet du modèle KGZNet prend en entrée l'image x pour produire en sortie un vecteur de probabilité $\tilde{p}_G(y|x) \in [0,1]^C$ en utilisant une perte $\mathcal{L}_G$ basée sur l'entropie croisée binaire. La seconde branche LungNet exploite les connaissance anatomique selon laquelle la majorité des pathologies du thorax sont localisée dans les poumons. Pour cela, une région pulmonaire x_r est extraite à partie de l'image d'origine à l'aide du module de segmentation *Lung Region Generator*, basé sur un U-Net (Ronneberger et al. 2015). Cette image focalisée est ensuite redimensionnée et passée dans un CNN de même structure que la branche globale. Le vecteur de sortie $\tilde{p}_R(y|x_r) \in [0,1]^C$ est obtenu en utilisant une perte $\mathcal{L}_R$ également basée sur l'entropie croisée binaire, permettant de spécialiser l'apprentissage sur la morphologie pulmonaire.

La troisième branche LesionNet s'attaque aux région de lésions les plus informatives. À partir de l'image pulmonaire x_r, une carte d'attention $H_l(a,b)$ est construite en agrégeant les activations de la dernière couche CNN de LungNet. Pour chaque position spatiale (a,b), on calcule :

$$H_l(a,b) = \max_d \left| f_d^l(a,b) \right| \tag{3.12}$$

où $f_d^l(a,b)$ est l'activation du canal d à la position (a,b). Cette carte d'attention encode l'importance relative de chaque pixel pour la tâche de classification. En appliquant un seuil τ, le modèle génère un masque binaire S_l :

$$S_l(a,b) = \begin{cases} 1 & \text{si } H_l(a,b) > \tau \\ 0 & \text{sinon} \end{cases}, \tag{3.13}$$

puis une boîte englobante est ajustée sur la plus grande composante connexe du masque pour extraire une région discriminante x_l. Cette image est ensuite redimentionnée et traitée par un CNN identique aux deux précédents pour produire une sortie $\tilde{p}_L(y|x_l) \in [0,1]^C$ en utilisant également une entropie croisée binaire $\mathcal{L}_L$.

Enfin, les représentations des couches après un échantillonage global des trois branches sont concaténées pour former un vecteur de caractéristique. Cette concaténation est ensuite transmise à une couche entièrement connectée FC qui synthétise l'information provenant des différentes couches et produit la prédiction finale $\hat{y}$ à partir d'une perte $\mathcal{L}_F$ basée sur l'entropie croisée binaire. L'objectif global d'optimisation du modèle est défini par la combinaison pondérée des pertes :

$$\min \mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_G + \lambda_2 \mathcal{L}_R + \lambda_3 \mathcal{L}_L + \lambda_4 \mathcal{L}_F \tag{3.14}$$

où les coefficients λ_i contrôlent l'importance relative de chaque source d'information. Cette conception permet une spécialisation hiérarchique progressive de l'attention, exploitant efficacement la structure anatomique et les régions discriminantes sans annotation manuelle. Tout comme dans le travail précédent (AG-CNN), la connaissance ici est implicite, dérivée du fait que les pathologies thoraciques se manifestent généralement dans des régions focalisées et visuellement discriminantes dans les poumons, souvent localisées sous forme de nodules, masses ou infiltrats.

Dans une autre logique d'utilisation de l'attention, Li et al. (2019) propose un modèle de détection du glaucome à partir d'images du fond de l'œil intégrant des connaissances sous forme de cartes d'attention humaines. Les cartes d'attention sont obtenues à partir de simulations de suivi oculaire sur des experts ophtalmologistes, lesquelles localisent les régions pertinentes dans les images du fond de l'œil pour le diagnostic du glaucome. Le modèle prend en entrée une image $x \in \mathbb{R}^{224\times224\times3}$ et produit deux sorties : une carte d'attention $\hat{A} \in [0,1]^{112\times112}$ calculée par une sous branche d'attention qui est supervisée à partir des cartes d'attention produites par les experts ophtalmologistes, et une prédiction binaire $\hat{y} \in \{0,1\}$ (Figure3.5).

La première composante du modèle est le sous-réseau de prédiction d'attention, qui apprend à reproduire les cartes d'attention via une divergence de Kullback-Leibler :

$$\mathcal{L}_{\text{att}} = \frac{1}{I \cdot J} \sum_{i=1}^{I} \sum_{j=1}^{J} A_{ij} \log \left(\frac{A_{ij}}{\hat{A}_{ij}} \right), \tag{3.15}$$

où A est la carte d'attention de référence produite par les experts. Ensuite, un second sous-réseau, dédié à la localisation des régions pathologiques, ne prédit pas directement une sortie, mais affine l'information extraite par le modèle. Il prend en entrée l'image x pour générer une carte de visualisation $\hat{V}$ en utilisant la méthode de rétro-propagation guidée (*Guided Backpropagation*) appliquée à la carte d'attention $\hat{A}$, identifiant les zones de l'image ayant

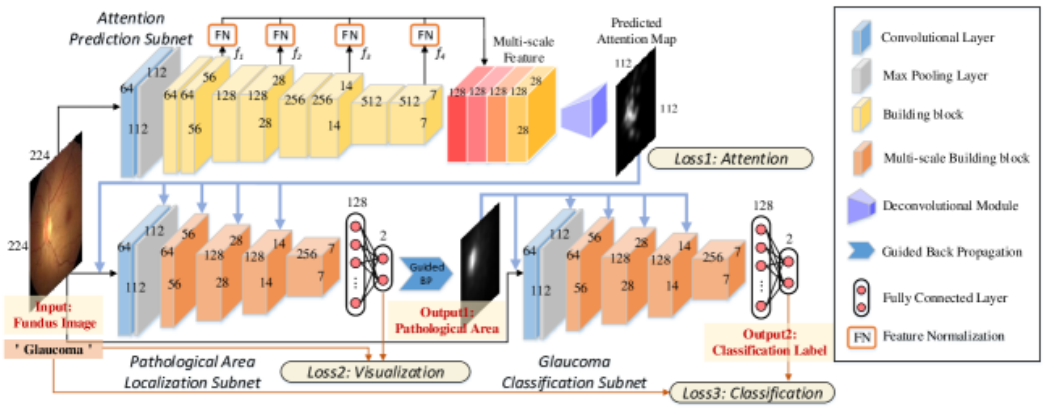


FIGURE 3.5 – Architecture du modèle de détection du glaucome dans l'œil. Source : Li et al. (2019).

le plus d'impact sur la décision. Cette carte $\hat{V}$ est ensuite utilisée pour pondérer les cartes de caractéristiques issues des convolutions intermédiaires F, c'est-à-dire les représentations internes de l'image extraites avant la classification, en produisant une version focalisée F' définie par :

$$F' = F \odot ((1 - \theta)\hat{V} + \theta), \tag{3.16}$$

où $\odot$ désigne le produit élément par élément et $\theta \in [0, 1]$ ajuste le degré d'attention locale. L'image originale, sa représentation F et sa version pondérée F' sont ensuite fusionnées et utilisées dans le dernier sous-réseau de classification qui prédit $\hat{y}$. L'ensemble du modèle est optimisé par la fonction :

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{att}} + \beta\mathcal{L}_{\text{visu}} + \gamma\mathcal{L}_{\text{cls}} \tag{3.17}$$

où $\mathcal{L}_{\text{att}}$ est la perte liée à l'attention, $\mathcal{L}_{\text{visu}}$ est la perte liée à régularisation de l'information visuelle et $\mathcal{L}_{\text{cls}}$ est la perte liée à la classification. Cette architecture exploite ainsi des connaissances humaines explicites à travers un mécanisme d'attention et visuel combiné, se distinguant de nos contributions qui mobilisent des connaissances sémantiques extraites automatiquement à partir des données.

Mitsuhara et al. (2019) propose une méthode d'intégration de connaissances humaines dans les réseaux profonds via l'édition manuelle des cartes d'attention produites par un modèle Attention Branch Network (ABN) (Figure 3.6).

Le processus d'apprentissage est divisé en trois étapes. La première étape consiste à entraîner le réseau ABN qui prend en entrée une image $x \in \mathbb{R}^{H \times W \times 3}$ à partir de la quelle est extraite une carte d'attention A. Seules les cartes d'attention des échantillons d'entraînement mal classés sont sélectionnées pour être éditées dans la deuxième étapes, produisant des cartes corrigées A' qui reflètent les régions considérées comme discriminantes selon la connaissance. L'édition est faite manuellement par une expertise humaine, ou à partir d'informations supplémentaires disponibles avec le jeu de données. Dans la troisième étape, une branches d'attention chargé de générer les cartes d'attention A et une branche de perception pour la classification sont affinés (fine-tuning) en minimisant une fonction de

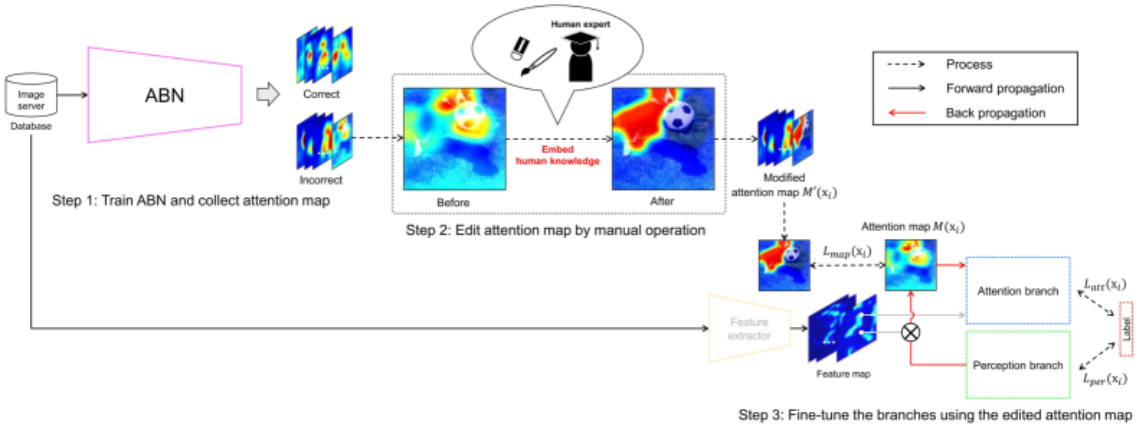


FIGURE 3.6 – Architecture du modèle proposé par Mitsuhara et al. (2019). Source : Mitsuhara et al. (2019).

perte composée à la fois de la perte ABN et d'un terme de régularisation basé sur la distance entre la carte générée par le modèle et la version éditée :

$$\mathcal{L} = \mathcal{L}_{\text{ABN}} + \mathcal{L}_{\text{map}} \tag{3.18}$$

avec

$$\mathcal{L}_{\text{ABN}} = \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{per}} \quad \text{et} \quad \mathcal{L}_{\text{map}} = \gamma \|A' - A\|^2 \tag{3.19}$$

où $\mathcal{L}_{\text{att}}$ et $\mathcal{L}_{\text{per}}$ sont des pertes d'entropie croisée pour les branches d'attention et de perception, et γ est un facteur d'échelle. Une limite importante de ce travail est sa forte dépendance à une intervention humaine manuelle. Seules les cartes d'attention des échantillons mal classés sont éditées par des experts, ce qui rend le processus non automatisable à grande échelle. Nos contributions diffèrent en ce qu'elles reposent sur des structures sémantiques construites automatiquement, sans intervention manuelle directe pendant l'apprentissage du modèle.

En complément des approches précédemment évoquées, plusieurs travaux tirent parti des mécanismes d'attention (Wang et al. 2021b; Sun et al. 2022) pour améliorer la localisation des régions discriminantes ou pour capturer des relations complexes entre parties visuelles, en particulier dans des contextes ou les différences entre classes sont subtiles. Ces méthodes sont particulièrement pertinentes dans le cadre de la classification à grain fin, un domaine auquel nos différentes contributions sont liées. Nous reviendrons plus en détail sur ces approches dans le chapitre suivant, spécifiquement dédié à cette problématique.

Utilisation de graphe

Chen et al. (2020a) propose un cadre d'apprentissage à peu d'exemples (*few-shot learning*) Song et al. (2023) appelé KGTN (*Knowledge Graph Transfer Network*) qui intègre des connaissances de corrélations sémantiques explicites sous forme de graphe de connaissances pour enrichir la représentation des classes faiblement échantillonnées. Le modèle prend en

entrée un ensemble d'images $x \in \mathbb{R}^{H\times W\times 3}$, extrait via un extracteur de caractéristique $\phi(x)$ une nouvelle représentation $\ell \in \mathbb{R}^d$ de l'image, et prédit une classe $\hat{y} = \arg\max_k f_k(\ell)$, où $f_k(\ell) = \mathbf{w}_k^\top \ell + b_k$ est le classifieur qui calcule la confiance d'appartenance de l'échantillon x à la classe k. $\mathbf{w}_k$ désigne le poids du classifieur et b_k est le terme de biais. En reformulant $f_k(\ell)$ comme une mesure de similarité et en considérant $b_k = 0$, on obtient :

$$f_k(\ell) = -\frac{1}{2}\|\ell - \mathbf{w}_k\|_2^2, \tag{3.20}$$

ce qui justifie l'interprétation de $\mathbf{w}_k$ comme prototype de la classe k. L'échantillon x est prédit comme appartenant à la classe k si sa caractéristique ℓ a la similarité maximale (ou la distance minimale) avec le prototype $\mathbf{w}_k$. Toutefois, en apprentissage à peu d'exemples, $\mathbf{w}_k$ est biaisé par la spécificité des rares exemples. Pour faire face à ce problème, KGTN introduit un graphe sémantique $G = (V, A)$ où chaque nœud $v_k \in V$ représente une classe, initialisé par son prototype initial $h_k^0 = \mathbf{w}_k^{\text{init}}$. Les arêtes $A = [a_{ij}]$ modélisent les corrélations sémantiques (issues de GloVe ou Wordnet) entre les nœuds i et j du graphe (Figure 3.7).

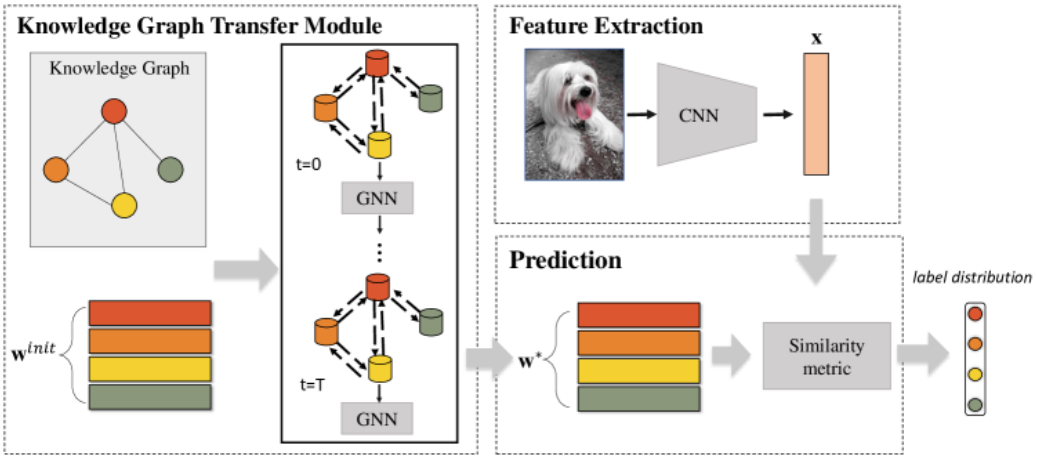


FIGURE 3.7 – Architecture du modèle KGTN. Source : Chen et al. (2020a).

Ce graphe est traité de façon itérative par un réseau de type Gated Graph Neural Network (GGNN) qui met à jour l'état de chaque nœud selon un mécanisme itératif. À chaque pas de temps t, les nœuds agrègent les messages de leurs voisins corrélés selon :

$$a_k^t = \left[\sum_{k'} a_{kk'} h_{k'}^{t-1}, \sum_{k'} a_{k'k} h_{k'}^{t-1}\right]. \tag{3.21}$$

Ensuite, chaque vecteur h_k^t est mis à jour par un bloc de type Gated Recurrent Unit (GRU) :

$$\begin{aligned} z_k^t &= \sigma(W_z a_k^t + U_z h_k^{t-1}) \\ r_k^t &= \sigma(W_r a_k^t + U_r h_k^{t-1}) \\ \tilde{h}_k^t &= \tanh(W a_k^t + U(r_k^t \odot h_k^{t-1})) \\ h_k^t &= (1 - z_k^t) \odot h_k^{t-1} + z_k^t \odot \tilde{h}_k^t \end{aligned} \tag{3.22}$$

où σ est la sigmoïde, $\odot$ le produit élément par élément, et W, U des matrices apprises. Après T itérations, chaque nœud produit un nouveau prototype w_k^* défini par :

$$w_k^* = o(h_k^T, h_k^0) \tag{3.23}$$

combinant l'état final et l'état initial. Ce vecteur est utilisé pour classer une image x en calculant une similarité $f_k(\phi(x))$ (produit scalaire, cosinus ou corrélation de Pearson), injectée dans un softmax. Ce cadre permet un transfert de connaissances explicites entre classes, même si les classes cibles ne possèdent qu'un ou deux exemples. Contrairement à notre deuxième contribution qui encode la connaissance dans une fonction de perte dérivée d'un graphe de connaissances intégrant les relations image-classe, KGTN exploite un graphe inter-classes et met à jour dynamiquement les classifieurs à l'aide d'un GGNN. Notre méthode agit donc sur la perte, alors que KGTN agit sur les classifieurs, ce qui rend notre approche plus légère et indépendante de la structure du classifieur. Il existe d'autres travaux qui exploitent également les relations image-classe pour le problème difficile de classification à grain fin (Chen et al. 2018b), et nous les présenterons dans le chapitre suivant qui y est consacré.

3.2.2 Intégration dans l'algorithme d'apprentissage

Utilisation d'équations

Stewart and Ermon (2016) introduit une méthode d'apprentissage des CNNs supervisée uniquement par des connaissances a priori exprimées sous forme de contraintes différentiables. Ces contraintes sont modélisées par une fonction $g(x, f(x_i))$ qui pénalise les sorties incohérentes du modèle de base $f(\cdot)$ avec la structure attendue. L'idée centrale est que, même sans étiquettes, on peut obtenir une bonne fonction prédictive dans un cadre supervisé en minimisant :

$$\mathcal{L} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} g(x_i, f(x_i)) + \mathcal{R}(f) \tag{3.24}$$

où x_i est une image d'entrée, et $\mathcal{R}(f)$ est un terme de régularisation. Par exemple, pour estimer la hauteur d'un objet en chute libre à partir d'un champs de vision (séquence d'images), les auteurs utilisent uniquement une loi physique plutôt que d'utiliser des étiquettes. Pour cela, un réseau de régression est entraîné, prenant en entrée une séquence de n_s images pour prédire n_s hauteurs estimées :

$$f : (x_1, \ldots, x_{n_s}) \mapsto (y_1, \ldots, y_{n_s}) \tag{3.25}$$

Pour superviser le modèle, la contrainte physique utilisée impose que la hauteur d'un objet en chute libre suit une trajectoire :

$$\mathbf{y}_i = y_0 + v_0 \cdot (i\Delta t) + a \cdot (i\Delta t)^2, \tag{3.26}$$

où $\Delta t = 0{,}1$ s est le temps entre deux images, $a = -9{,}8\,\mathrm{m/s^2}$ est l'accélération de la gravité, y_0 et v_0 sont les hauteurs et les vitesses initiales. Le réseau est alors entraîné pour respecter cette loi physique en minimisant la fonction définie par :

$$g(x_i, f(x_i)) = \sum_{i=1}^{n_s} |\mathbf{y}_i - f(\mathbf{x})_i| \tag{3.27}$$

où $\mathbf{x}$ est un vecteur de n_s images, et $\mathbf{y}$ est la fonction de la trajectoire parabolique. Outre les . Outre les lois physiques, les auteurs traitent également les relations logiques entre objets, telles que $o_1 \Rightarrow o_2$ (par exemple, si Peach est présente dans une image, alors Mario est aussi dans l'image). Les réseaux f_1 et f_2, construits pour détecter respectivement Peach et Mario, sont entraînés non pas à partir d'étiquettes, mais à partir de cette implication logique. Pour éviter les solutions triviales (comme $f_1(x) = f_2(x) = 1$ pour tout x), les auteurs introduisent trois termes de perte auxiliaires :

- $\mathcal{L}_{\text{inv}}(x, k)$: contrainte d'invariance par rotation aléatoire qui compare la probabilité de détection sur l'image originale et une version transformée $\rho(x)$, définie par :

$$\mathcal{L}_{\text{inv}}(x, k) = \frac{1}{B} \sum_{i=1}^{B} |\Pr[f_k(x_i) = 1] - \Pr[f_k(\rho(x_i)) = 1]| \tag{3.28}$$

- $\mathcal{L}_{\text{var}}(x, k)$: contrainte sur la variance, qui encourage la variabilité statistique des sorties sur le batch de taille B, en maximisant l'écart-type (toujours 0 ou toujours 1) :

$$\mathcal{L}_{\text{var}}(x, k) = -\text{std}_{i \in [1,\ldots,B]} \left(\Pr[f_k(x_i) = 1]\right) \tag{3.29}$$

- $\mathcal{L}_{\text{ent}}(x, k)$: contrainte sur l'entropie des prédictions binaires, qui régule la distribution des prédictions binaires combinées $v \in \{0, 1\}^2$ produites par $f = (f_1, f_2)$, pour assurer une répartition équilibrée des cas :

$$\mathcal{L}_{\text{ent}}(x, v) = \frac{1}{B} \sum_{i=1}^{B} \left(\Pr[f(x_i) = v] - \frac{1}{3} + \left(\frac{1}{3} - \mu_v \right) \right)^2 \tag{3.30}$$

où μ_v est la fréquence empirique de v comme sortie majoritaire :

$$\mu_v = \frac{1}{B} \sum_{i=1}^{B} \mathbb{1}\{v = \arg \max_{v' \in \{0,1\}^2} \Pr[f(x_i) = v']\} \tag{3.31}$$

Pour éviter que f_1 et f_2 exploitent les mêmes régions visuelles, un mécanisme de séparation spatiale est imposé par le modèle. Chaque réseau choisit un unique vecteur spatial issu d'un tenseur $7 \times 7 \times 64$, mais f_2 ne peut choisir de région trop proche de celle choisie par

f_1. Cela garantit une diversité dans les sources d'informations utilisées. La version finale de la fonction de perte est :

$$\mathcal{L}(x) = \mathbb{1}\{f_1(x) \Rightarrow f_2(x)\} + \sum_{k\in\{1,2\}} \gamma_1 \mathcal{L}_{inv}(x,k) + \gamma_2 \mathcal{L}_{var}(x,k) + \sum_{v\in\{0,1\}^2} \gamma_3 \mathcal{L}_{ent}(x,v) \quad (3.32)$$

où $\gamma_1, \gamma_2, \gamma_3$ sont des poids d'importance. Cette méthode démontre que des réseaux peuvent apprendre à reconnaître des concepts visuels complexes sans étiquettes, simplement en respectant des relations logiques ou des contraintes physiques, avec des régularisations adéquates pour éviter que les solutions ne dégénèrent. Bien que nous intégrons aussi des connaissances a priori dans la fonction de perte dans nos contributions, nos connaissances sont issues des données d'entraînement. De plus, nous ne supervision pas nos modèles uniquement avec des connaissances, mais aussi avec des étiquettes sur les images.

Utilisation de contraintes

CDEP (*Contextual Decomposition Explanation Penalization*) (Rieger et al. 2020) introduit une méthode d'apprentissage supervisée dans laquelle un réseau de neurones est contraint à apprendre pour les bonnes raisons, c'est-à-dire en utilisant des explications expertes fournies par l'expert pour pénaliser les cas où le modèle s'appuie sur des caractéristiques non pertinentes pour faire des prédictions. Ces explications désignent les sous-parties de l'entrée (pixels) considérées comme pertinentes pour la prédiction.

Plutôt que de superviser uniquement les sorties du modèle $\hat{y} = f(x)$, CDEP impose également des contraintes sur les explications internes du modèle en utilisant une décomposition contextuelle ou en anglais Contextual Decomposition (CD) (Murdoch et al. 2018). CD décompose les logits d'un modèle $g(x)$, où $f(x) = \text{Softmax}(g(x))$, en deux parties :

$$g_{CD}(x) = (\beta(x), \gamma(x)), \quad \text{avec} \quad g(x) = \beta(x) + \gamma(x) \quad (3.33)$$

où $\beta(x)$ capture la contribution du sous-ensemble de caractéristiques $x_S \subset x$ sélectionné par l'expert, et $\gamma(x)$ est le sous-ensemble restant de l'image d'entrée. Cette décomposition est appliquée couche après couche sur le CNN à travers les transformations $g_i, i = \{1, \ldots, L\}$ où L représente le nombre de couches de convolution :

$$g_{CD}(x) = g_{CD}^{L} \circ g_{CD}^{L-1} \circ \cdots \circ g_{CD}^{1}(x) \quad (3.34)$$

Le signal explicatif $\beta(x_S)$ est ensuite comparé à une cible $\text{expl}_{x,S}$ définie par l'expert (forte importance ou négligence d'une région par exemple). Cette comparaison est intégrée à la fonction de perte globale du modèle, définie par :

$$\mathcal{L} = \arg\min_{\theta} \sum_i \sum_c -y_{i,c} \log f(x_i)_c + \lambda \sum_i \sum_S \|\beta(x_{i,S}) - \text{expl}_{x_i,S}\|_1 \quad (3.35)$$

Cette formulation permet de superviser non seulement les sorties, mais aussi les mécanismes internes du modèle, en orientant ses décisions vers les facteurs jugés pertinents. Plusieurs méthodes récentes (Zhuang et al. 2020; He et al. 2022) cherchent également à intégrer les connaissances dans la fonction de perte en formulant des contraintes explicites pour la classification à grain fin. Ces approches seront abordées plus en détail dans le chapitre suivant, dédié à la classification à grain fin.

Utilisation de graphes

Jiang et al. (2018) propose une méthode pour améliorer la détection d'objets en intégrant des connaissances explicites (relations sémantiques entre classes) et implicite (régularités spatiales) pour enrichir la représentation des objets (Figure 3.8). Le problème est formulé comme l'apprentissage d'une fonction de détection :

$$f : X \rightarrow \left\{(\hat{b}_i, \hat{y}_i)\right\}_{i=1}^{m} \tag{3.36}$$

à partir d'un jeu de données étiqueté $\mathcal{D} = \{(x, \{(b_i, y_i)\}_{i=1}^{m})\}_{x \in X}$, où $x \in X$ est une image, b_i une boîte englobante, et $y_i \in Y$ une étiquette de classe. Le modèle prédit, pour chaque image x, un ensemble de boîte $\hat{b}_i$ localisant les objets, et leurs classes associées $\hat{y}_i$.

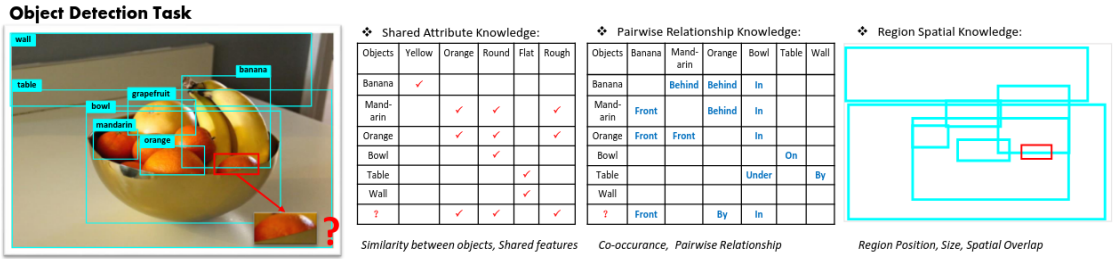


FIGURE 3.8 – Exemple de connaissances utilisées par HKRM pour la détection d'objets. Source : Jiang et al. (2018).

Pour résoudre ce problème, les auteurs introduisent le modèle HKRM (*Hybrid Knowledge Routed Modules*) (Figure 3.9) qui enrichit les représentations extraites d'un modèle de base de type Faster R-CNN (Ren 2015).

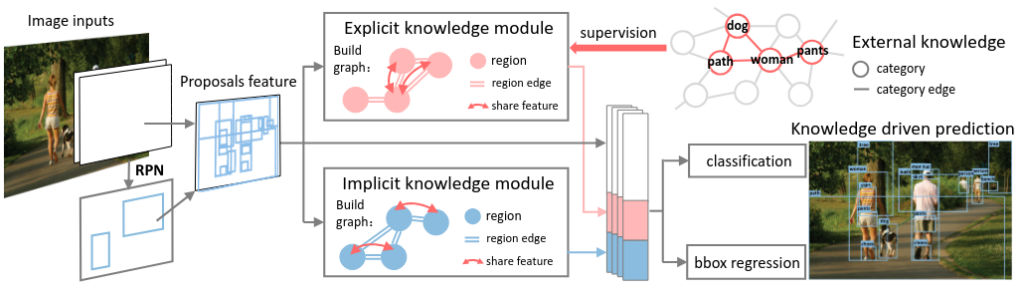


FIGURE 3.9 – Architecture du modèle HKRM. Source : Jiang et al. (2018).

Le module de connaissances explicites vise à améliorer les représentations $\mathbf{b}_i$ des régions b_i en intégrant des connaissances humaines structurées. Ces connaissances peuvent être des graphes d'attributs (la pomme est *rouge*) ou des graphe de relations sémantiques entre paire d'objets (l'homme *pédale* le vélo). Partant de la représentation de l'ensemble des m régions proposées $\hat{B} = [\hat{\mathbf{b}}_i]_{i=1}^{m}$, $\hat{\mathbf{b}}_i \in \mathbb{R}^d$, les auteurs construisent un graphe adaptatif région à région $\hat{G} = \langle \hat{V}, \hat{E} \rangle$, où chaque arête $\hat{e}_{ij} \in \hat{E}$ encode le lien entre les nœuds v_i et v_j, représentant les régions b_i et b_i, respectivement. Le graphe $\hat{G}$ est appris à l'aide d'un MLP supervisé par un graphe de connaissances classe à classe $G = \langle V, E \rangle$. Chaque arrête $\hat{e}_{ij}$ est définie par :

$$\hat{e}_{ij} = \text{MLP}\left(\|\hat{\mathbf{b}}_i - \hat{\mathbf{b}}_j\|_1\right), \tag{3.37}$$

et supervisée par la valeur cible e_{ij}, extraite du graphe de connaissances G à partir des vérités terrains v_i et v_j associées aux régions. Le MLP est donc entraîné pour minimiser la perte :

$$\mathcal{L}_{\text{exp}} = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{1}{2} (\hat{e}_{ij} - e_{ij})^2. \tag{3.38}$$

Une fois les arêtes apprises, elles sont utilisées pour faire une propagation des caractéristiques des régions connectées :

$$\hat{B}' = \hat{E}\hat{B}W_e, \tag{3.39}$$

où $\hat{E} \in \mathbb{R}^{m \times m}$ est la matrice des arêtes, $\hat{B} \in \mathbb{R}^{m \times d}$ les caractéristiques visuelles, $W_e \in \mathbb{R}^{d \times d_G}$ une matrice de poids apprise, et $\hat{B}' \in \mathbb{R}^{m \times d_G}$ les nouvelles représentations enrichies des régions.

Le module de connaissances implicites vise à capturer des régularités géométriques entre objets (le bateau est toujours au dessus de l'eau) sans supervision explicite, en exploitant la configuration spatiale des régions détectées. Étant donné un vecteur géométrique normalisé q_i pour chaque proposition de région $\hat{b}_i$, le module de connaissances implicites utilise plusieurs MLPs, comme celui présenté dans l'équation 3.37, pour apprendre plusieurs graphes $\hat{G}^{(k)}, k = 1, \ldots, n_K$ afin de capturer des relations spatiales de type « haut et bas », « gauche et droite » et « coin et centre ». le vecteur q_i est défini par :

$$q_i = \left(\frac{x_i}{\bar{w}}, \frac{y_i}{\bar{h}}, \frac{w_i}{\bar{w}}, \frac{h_i}{\bar{h}}, y_i\right), \tag{3.40}$$

où x_i, y_i sont les coordonnées du centre de la boîte $\hat{b}_i$, w_i sa largeur, h_i sa hauteur, $\bar{w}, \bar{h}$ les moyennes des largeurs et hauteurs de toutes les régions dans l'image, et y_i est le score de confiance initial du détecteur pour la boîte $\hat{b}_i$.

Les poids moyens de toutes les arêtes des graphes $\{\hat{G}^{(k)}\}$ sont calculés et ajoutés à une matrice identité I pour obtenir les poids finaux des arrêtes $\tilde{e}_{ij} \in \tilde{\hat{E}}$:

$$\tilde{e}_{ij} = \frac{1}{n_K} \sum_{k=1}^{n_K} \hat{e}_{ij}^{(k)} + I, \tag{3.41}$$

Une multiplication matricielle est ensuite utilisée pour calculer la représentation enrichie de chaque région :

$$\tilde{B} = \tilde{\hat{E}} \hat{B} W_g \tag{3.42}$$

où W_g est une matrice de poids apprise.

Les différentes représentations de propositions de régions issues de Faster R-CNN et celles enrichies en utilisant les graphes sont concaténées pour produire les prédictions finales de classification et de régression pour les cordonnées des boites englobantes. Bien que nos contributions exploitent également les co-occurrences entre classes ou objets présents dans les images, elles s'inscrivent dans un cadre de classification d'images et non de détection d'objets.

3.3 Conclusion

L'ensemble des travaux présentés dans ce chapitre met en évidence la richesse des approches développées pour intégrer des connaissances dans les architectures profondes appliquées à la vision par ordinateur. Qu'il s'agisse de connaissances explicites (règles, graphe, etc.) ou de connaissances implicites (co-occurrences, attention, etc.), ces méthodes permettent d'améliorer la robustesse, l'explicabilité ou la performance des modèles.

Nos contributions s'inscrivent dans cette dynamique, et se distinguent par leur focus sur la classification à grain fin. Nous nous appuyons sur des formes de connaissances explicites, extraites ou dérivées de données, que nous intégrons soit dans la structure des architectures, soit dans l'algorithme d'apprentissage à travers la fonction de perte. Nos approches ne cherchent pas à localiser ou à justifier la décision, mais à guider la représentation interne du modèle à travers des structuration sémantiques ou des raisonnements. Ce positionnement nous permet d'aborder de manière ciblée les défis propres à la classification à grain fin.

Chapitre 4

Classification à grain fin

Ce chapitre constitue une étape centrale de cette thèse, dans la mesure où il s'intéresse au cadre applicatif principal de nos contributions : la classification à grain fin. Ce type de classification soulève des enjeux spécifiques en matière de discrimination visuelle, et donc de modélisation, du fait de la proximité sémantique et visuelle entre les classes. L'objectif de ce chapitre est double. Dans un premier temps, nous décrirons les caractéristiques propres à la classification à grain fin ainsi que les grandes tendances méthodologiques identifiées dans la littérature. Dans un second temps, nous nous concentrerons sur les approches qui intègrent des formes de connaissance dans ce contexte, afin de positionner précisément nos contributions dans le paysage existant.

Contents

4.1 Préliminaires

La classification d'images à grain fin, ou en anglais Fine-Grained Image Classification (Wei et al. 2022), est une branche spécifique de la classification en apprentissage automatique et en vision par ordinateur. Elle vise à identifier des différences subtiles entre des catégories visuellement similaires dans les images, souvent invisibles à l'échelle globale de l'image. Contrairement à la classification d'images générique où les différences entre les classes sont souvent très marquées (par exemple, chien vs chat), la classification d'images à grain fin se concentre sur des discriminations plus nuancées, comme différencier des espèces d'oiseaux, des modèles de voitures ou des variétés de plantes (Figure 4.1). Les principaux défis incluent une forte variance au sein des classes, une similitude entre classes, et un besoin élevé en annotations expertes.

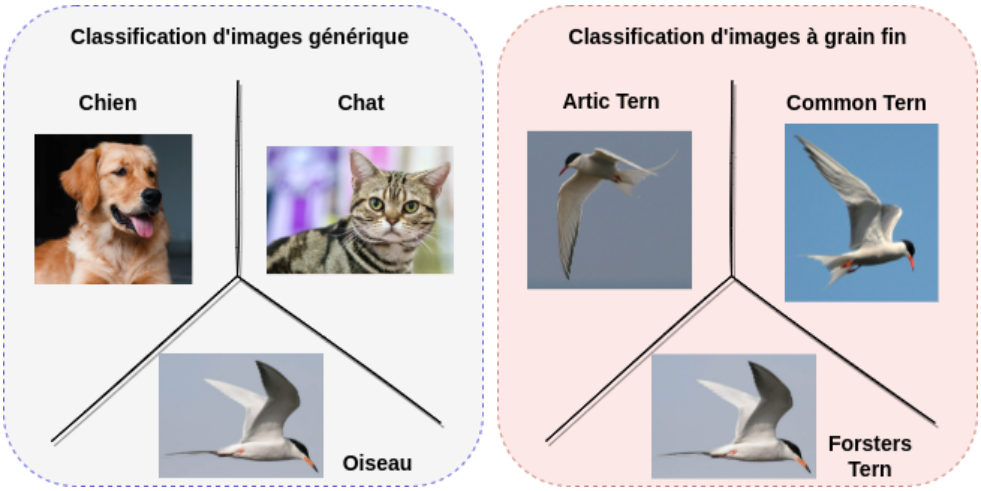


FIGURE 4.1 – Illustration de la différence entre la classification d'images générique et la classification d'images à grain fin.

Pour relever ce défi, plusieurs axes méthodologiques ont émergé (Wei et al. 2022, 2019; Xie et al. 2023). Une première grande famille d'approches repose sur la localisation automatique des régions discriminantes, c'est-à-dire les zones de l'image qui portent les indices visuels pertinents pour distinguer les classes proches. Ces méthodes combinent souvent deux sous-réseaux : un sous-réseau de localisation qui identifie des régions informatives (comme la tête d'un oiseau, les motifs d'un papillon, ou les phares d'un véhicule), et un sous-réseau de classification qui extrait des représentations spécifiques à ces régions. Certaines approches reposent sur des techniques de segmentation supervisée, où des annotations supplémentaires sont utilisées pour guider l'apprentissage de régions clés (Branson et al. 2014; Zhang et al. 2014; Girshick et al. 2014; Lin et al. 2015a), tandis que d'autres opèrent dans un cadre faiblement supervisé ou non supervisé, s'appuyant sur des régularités internes aux images ou des corrélations entre régions pour apprendre des informations pertinentes (Zhang et al. 2016b; He and Peng 2017b; Ge et al. 2019; Wang et al. 2020d; Liu et al. 2020). Une autre stratégie repose sur l'exploitation de filtres convolutifs localisés, où de petits noyaux de convolution génèrent des activations locales interprétables comme des descripteurs de parties discriminantes. Ces descripteurs peuvent être appris indépendamment de la classification (Xiao et al. 2014; Simon and Rodner 2015; Liu et al. 2014) ou intégrés directement dans un apprentissage de bout à bout (Wang et al. 2016b; Ding et al. 2019).

Enfin, les méthodes basées sur des mécanismes d'attention offrent une solution flexible et souvent auto-supervisée pour focaliser dynamiquement l'attention du modèle sur les régions les plus informatives (Fu et al. 2017; Zheng et al. 2019; Ji et al. 2019; Wang et al. 2021b; Sun et al. 2022). Ces approches rendent possible un apprentissage ciblé et contextuel, mais sont souvent limités par la capacité du premier sous-réseau à identifier les régions discriminantes.

Une seconde grande famille d'approches en classification à grain fin repose sur l'apprentissage de représentations discriminantes de bout en bout à l'intérieur du réseau. Ces approches tirent pleinement parti de la capacité des réseaux de neurones profonds à modéliser non seulement des activations visuelles riches, mais aussi les interactions complexes entre ces activations. L'idée sous-jacente est que, dans des contextes où les différences entre classes sont minimes et localisées, capturer uniquement des caractéristiques indépendantes est insuffisant. Ainsi, plusieurs travaux proposent d'exploiter des modèles bilinéaires qui utilisent des produits tensoriels ou des matrices de covariance pour représenter les co-occurrences entre paires de descripteurs extraits de différentes régions de l'image (Lin et al. 2015b; Gao et al. 2015; Kong and Fowlkes 2017; Yu et al. 2018; Li et al. 2017; Wang et al. 2017b). Ces représentations croisées permettent de capturer des relations cruciales pour différencier des catégories visuellement proches. D'autres travaux cherchent à enrichir ces interactions en introduisant des noyaux de convolution non linéaires, notamment des noyaux polynomiaux (Cui et al. 2017; Cai et al. 2017) ou des noyaux gaussiens (Engin et al. 2018), qui permettent de modéliser explicitement des dépendances plus complexes entre les caractéristiques extraites au sein même des couches du réseau. Parallèlement à l'enrichissement des représentations, une autre direction consiste à agir sur la fonction de perte pour mieux adapter l'optimisation aux contraintes de la classification à grain fin. Certaines méthodes visent par exemple à limiter l'excès de confiance du modèle vis-à-vis de classes très proches visuellement, en introduisant des régularisations spécifiques (Dubey et al. 2018b,a). D'autres emploient des pertes contrastives afin de forcer le modèle à rapprocher les représentations d'images similaires et à éloigner celles des images de classes différentes, en exploitant des paires d'images annotées (Sun et al. 2018a; Gao et al. 2020; Zhuang et al. 2020; He et al. 2022). Enfin, plusieurs travaux cherchent à intégrer une structure sémantique explicite entre les classes, comme des relations hiérarchiques ou des dépendances pédagogiques. Cela se traduit soit par l'ajout de tâches auxiliaires, comme la prédiction du niveau hiérarchique ou du groupe taxonomique auquel appartient une classe cible (Zhou and Lin 2015), soit par des techniques d'apprentissage par curriculum, qui organisent l'apprentissage de manière progressive selon la complexité des exemples (Du et al. 2020), soit par destruction et reconstruction de la structure spatiale de l'image (Chen et al. 2019b). Ces approches visent à guider le modèle dans une exploration plus structurée de l'espace des classes fines, en cohérence avec la manière dont un humain apprend à faire des distinctions expertes.

Une troisième famille d'approches pour la classification à grain fin s'appuie sur l'intégration de données multimodales issues du web, notamment l'image et le texte, ou encore des bases de connaissances structurées. Ces méthodes visent à enrichir la représentation visuelle avec des informations sémantiques complémentaires et peu coûteuses à annoter, comme les descriptions textuelles ou les graphes de connaissances. Certaines approches combinent les flux visuels et textuels dans un espace de représentation commun afin de maximiser la compatibilité entre une image et sa description dans des modèles à deux branches (Reed et al. 2016; He and Peng 2017a), ou intègrent des modules d'attention pour aligner des régions d'images avec des mots pertinents (Song et al. 2020). D'autres exploitent les graphes séman-

tiques comme source de connaissances structurées et les projettent dans des représentations partagées avec les caractéristiques visuelles en utilisant des mécanismes d'attention ou des réseaux de neurones à graphes (Graph Neural Network (GNN)) (Xu et al. 2018; Chen et al. 2018b). Une dernière catégorie inclut l'intervention humaine dans la boucle d'apprentissage, sous forme d'annotations, de retours ou d'aide à la sélection des parties discriminantes, permettant ainsi d'affiner itérativement la reconnaissance grâce à l'expertise humaine (Cui et al. 2016; Deng et al. 2016). Ces approches exploitent la complémentarité entre vision, langage et raisonnement humain pour pallier les limites des signaux purement visuels dans les tâches de classification à grain fin.

Dans la section suivante, nous présenterons une analyse plus détaillée sur les travaux de la littérature qui proposent des approches de classification à grain fin intégrant des connaissances, et celles qui sont les plus récentes. Ceci permettra non seulement de mieux comprendre les tendances en matière d'intégration de connaissances dans le domaine, mais aussi de garantir une comparaison équitable des travaux récents avec nos propres contributions.

4.2 Travaux Existants

Formellement, le problème de classification à grain fin est un problème de classification particulièrement difficile, avec une faible variance inter-classes et une forte variance intra-classe. Comme tout problème de classification, la classification à grain fin cherche à apprendre une fonction de classification f à partir d'un jeu de données annoté $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, où $x_i \in \mathbb{X}$ représente une image et $y_i \in \{0,1\}^K$ est une étiquette correspondant à l'une des K classes possible. Dans la suite de cette section, nous présentons un ensemble de travaux de référence qui servent de points de comparaison pour l'évaluation de la méthode introduite dans notre deuxième contribution.

4.2.1 Utilisation de tâches auxiliaires

Chen et al. (2019b) introduit DCL (*Destruction and Construction Learning for Fine-Grained Image Recognition*) pour résoudre le problème de classification à grain fin. Le modèle combine deux stratégies complémentaires : l'apprentissage par destruction pour forcer l'extraction de détails discriminants, et l'apprentissage par construction pour modéliser la structure globale des objets (Figure 4.2).

Le processus d'apprentissage de DCL repose sur un triplet $(x, \psi(x), y)$, où $x \in \mathcal{X}$ est l'image d'entrée, $\psi(x)$ sa version altérée via un mécanisme de confusion RCM (*Region Confusion Mechanism*) et $y \in \mathcal{Y}$ la classe associée. Le module CRM fragmente l'image x en une grille de $n \times n$ régions $R_{i,j}$, puis en perturbe la disposition spatiale en déplaçant chaque région $R_{i,j}$ vers une nouvelle position $\sigma(i,j)$. Il en résulte une version désorganisée $\psi(x)$ de l'image x, où chaque région est déplacée dans son voisinage. L'objectif de cette perturbation est de pousser le classifieur à ignorer les structures globales partagées entre les classes fines et à se concentrer sur les détails visuels subtils. Le réseau est alors entraîné à minimiser la fonction de perte :

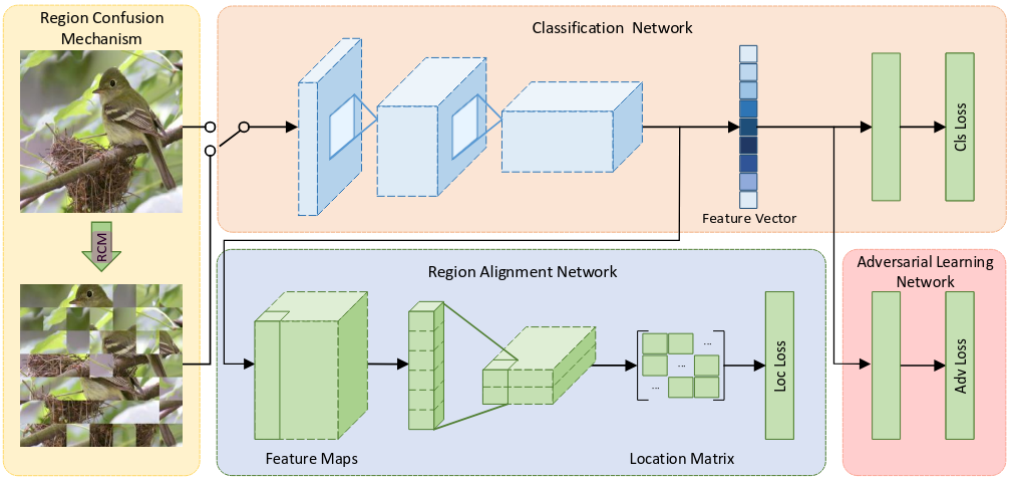


FIGURE 4.2 – Architecture du modèle DCL. Source : Chen et al. (2019b).

$$\mathcal{L}_{cls} = -\sum_{x \in \mathcal{X}} y \cdot \log\left[p(x) \cdot p(\psi(x))\right] \tag{4.1}$$

où $p(x)$ est la distribution de probabilité obtenue à la sortie de la fonction de classification f pour l'image x, et $p(\psi(x))$ est la probabilité pour l'image $\psi(x)$. Comme $\psi(x)$ peut contenir des formes, textures ou contours qui ne sont pas présents dans l'image de base x à cause de la pertubation, une branche adversariale est introduite pour détecter si une image est la version originale ou perturbée, en ajoutant la perte :

$$\mathcal{L}_{adv} = -\sum_{x \in \mathcal{X}} q \cdot \log f_D(x) + (1 - q) \cdot \log f_D(\psi(x)) \tag{4.2}$$

où f_D est une fonction linéaire agissant sur les activations intermédiaires du réseau, et $q \in \{0, 1\}^2$ est un vecteur one-hot qui indique si l'image est perturbée ou non.

La phase de construction de DCL est utilisé pour forcer le réseau à modéliser les relations sémantiques entre les régions locales dans une image. Elle s'appuie sur un sous-réseau d'alignement de régions qui prédit une matrice de localisation $M(x) \in \mathbb{R}^{2\times n\times n}$, où les deux derniers canaux correspondent aux coordonnées de localisation (a, b) de chaque sous-région. Une perte est utilisée dans ce sous-réseau pour mesurer l'écart entre les positions de régions prédites dans la matrice et les positions réelles dans l'image :

$$\mathcal{L}_{loc} = \sum_{x \in \mathcal{X}} \sum_{a=1}^{n} \sum_{b=1}^{n} \left\| M_{\sigma(a,b)}(\psi(x)) - \begin{bmatrix} a \\ b \end{bmatrix} \right\|_1 + \left\| M_{a,b}(x) - \begin{bmatrix} a \\ b \end{bmatrix} \right\|_1 \tag{4.3}$$

où le premier terme évalue la précision sur l'image permutée et le second terme évalue la précision sur l'image originale. L'objectif global de DCL devient :

$$\mathcal{L} = \alpha\mathcal{L}_{cls} + \beta\mathcal{L}_{adv} + \gamma\mathcal{L}_{loc} \tag{4.4}$$

où α, β et γ sont des pondérations. Le réseau est optimisé de bout en bout et permet d'apprendre à la fois des détails fins et une structure globale fiable en se basant sur la tâche auxiliaire de reconstruction de l'image perturbée.

Toujours dans la logique d'utilisation de tâche auxiliaire, PMG (Du et al. 2020) propose un cadre qui adopte une stratégie d'entraînement progressif en apprenant des représentations à différents niveaux de granularité de l'image d'entrée x. Le modèle repose sur deux mécanismes clés : l'entraînement progressif et une génération d'images désorganisées (Figure 4.3).

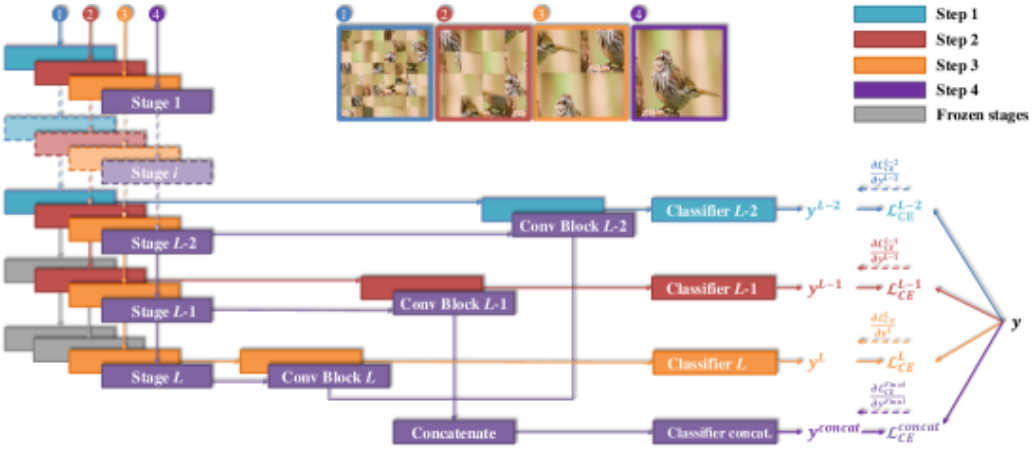


FIGURE 4.3 – Architecture du modèle PML. Source : Du et al. (2020).

L'entraînement progressif introduit les différentes couches du réseau progressivement, chacune entraînée à partir d'images correspondant à un niveau de granularité spécifique. Plus précisément, les premières couches apprennent à partir d'images morcelées en blocs fins, tandis que les couches suivantes utilisent des blocs de plus en plus larges, jusqu'à l'image complète. Cette stratégie permet à chaque niveau du réseau d'apprendre des informations discriminantes spécifiques à un niveau de granularité.

Le générateur d'images désorganisées $\psi(x)$ permet, à chaque étape de l'apprentissage, de découper les images en sous-blocs $n \times n$ (avec $n = 2^{L-l+1}$ pour la couche $l = \{1, 2, \ldots, L\}$), mélangées de manière aléatoire et recomposée pour forcer le modèle à se focaliser sur des détails spécifiques à la granularité correspondante. Contrairement à l'approche précédemment présentée DCL où le puzzle $\psi(x)$ est résolu pendant l'apprentissage, PMG utilise $\psi(x)$ uniquement pour créer des vues déstructurées afin de renforcer l'apprentissage à chaque échelle. Il utilise un extracteur de caractéristiques ϕ, composé de L blocs convolutifs, avec ϕ^l la sortie de la l-ième couche. Pour chaque couche l parmi les J dernières couches du réseau, un module H^l_{conv} est appliqué pour projeter la sortie ϕ^l en un vecteur $\ell^l = H^l_{conv}(\phi^l)$. Ensuite, un classifieur H^l_{class} est appliqué pour produire une prédiction y^l :

$$y^l = H^l_{class}(\ell^l). \tag{4.5}$$

À la fin de l'apprentissage progressif, les représentations $\ell^{L-J+1}, \ldots, \ell^L$ sont concaténées pour obtenir $\ell_{concat} = \text{concat}[\ell^{L-J+1}, ..., \ell^L]$, et une prédiction finale $\hat{y}$ est obtenue :

$$\hat{y} = H_{class}^{concat}(\ell_{concat}). \tag{4.6}$$

La fonction de perte totale combine les pertes de chaque couche l et une perte finale de classification sur la concaténation, définie par :

$$\mathcal{L} = \sum_{l=L-J+1}^{L} \alpha \cdot \mathcal{L}_{CE}(y^l, y) + \beta \cdot \mathcal{L}_{CE}(\hat{y}, y) \tag{4.7}$$

où y est la vérité terrain. Cette stratégie permet de structurer l'apprentissage autour des détails locaux dans un premier temps, puis d'intégrer progressivement les informations plus globales. Bien que DCL et PMG utilisent des tâches auxiliaires pour apprendre des informations locales et globales de l'image, ces modèles ne s'appuient cependant pas sur une forme de connaissance à priori, contrairement à nos contributions qui intègrent des connaissances externes pour guider l'apprentissage.

4.2.2 Utilisation de mécanismes d'attention

(Zheng et al. 2019) propose un réseau pour la classification à grain fin, nommé TASN (*Trilinear Attention Sampling Network*), qui combine trois modules (Figure 4.4) : un module d'attention trilinéaire pour localiser les détails visuels, un module de sélection d'attention pour extraire des vues localisées de l'image, et un module de distillation de connaissances pour transférer les informations fines vers un réseau maître.

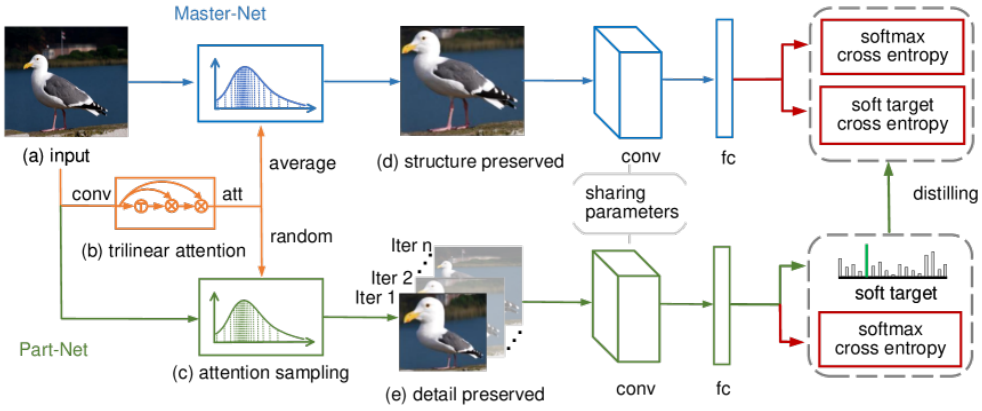


FIGURE 4.4 – Architecture du modèle TASN. Source : Zheng et al. (2019).

Le module d'attention trilinéaire transforme les cartes de caractéristiques X, obtenues à la sortie d'un CNN pour une image d'entrée $x \in \mathcal{X}$, en carte d'attention $A(X)$ selon l'opération :

$$A(X) := \text{softmax}(\text{softmax}(X)X^\top)X \tag{4.8}$$

Cette opération encode les corrélations spatiales entre les canaux (bilinéarité), suivies d'un affinage par projection trilinéaire (Figure 4.5). Chaque canal de la carte d'attention $A(X) \in \mathbb{R}^{c \times h \times w}$ obtenue représente un région d'attention.

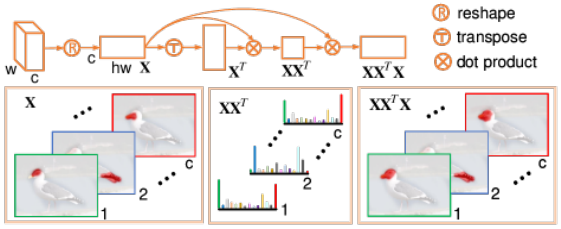


FIGURE 4.5 – Illustration de la projection trilinéaire du modèle TASN. Source : Zheng et al. (2019).

Le second module applique un échantillonage non uniforme à partir de cartes d'attention $A(X)$. Pour chaque image d'entrée x, deux nouvelles version d'images sont générées : une image x_s qui conserve la structure globale tout en supprimant les zones peu informatives, et une image x_d qui met l'accent sur une seule région discriminante avec une haute résolution. Ces deux images sont définies par :

$$x_s = S(x, R_1(A)), \quad x_d = S(x, R_2(A)) \tag{4.9}$$

où $S(\cdot)$ représente la fonction d'échantillonage, $R_1(A)$ correspond à la moyenne des cartes d'attention A, et $R_2(A)$ correspond à la sélection aléatoire d'un canal parmi les c canaux disponibles, pour extraire une seule région fine. Le mécanisme d'échantillonage repose sur l'inversion de la fonction de distillation cumulative (Devroye 1986) de l'attention (par axes a et b), définie par :

$$\mathcal{F}_a(n) = \sum_{j=1}^{n} \max_{1 \le i \le w} R_1(A)_{i,j}, \quad \mathcal{F}_b(n) = \sum_{i=1}^{n} \max_{1 \le j \le h} R_1(A)_{i,j} \tag{4.10}$$

où w et h représentent la largeur et la hauteur de la carte d'attention A, respectivement. L'image échantillonnée, par exemple pour l'image x_s, est alors donnée par :

$$S(x, R_1(A))_{i,j} = x_{\mathcal{F}_a^{-1}(i), \mathcal{F}_b^{-1}(j)} \tag{4.11}$$

où $\mathcal{F}^{-1}$ représente la fonction inverse de $\mathcal{F}$.

Le dernier module implémente une distillation de connaissances entre deux réseaux partageant les mêmes paramètres convolutifs : le part-net (enseignant) qui prend en entrée x_d pour produire le vecteur q_d, et le master-net (étudiant), qui prend en entrée l'image x_s pour produire le vecteur q_s. Les deux réseaux sont optimisés via la fonction de perte :

$$\mathcal{L}_{soft}(q_s, q_d) = -\sum_{i=1}^{N} q_d^{(i)} \log q_s^{(i)} \tag{4.12}$$

qui force la sortie du master-net q_s à imiter la sortie du part-net q_d. L'objectif final du master-net est alors :

$$\mathcal{L}(x_s) = \mathcal{L}_{cls}(q_s, y) + \lambda \mathcal{L}_{soft}(q_s, q_d) \tag{4.13}$$

où y représente la vérité terrain concernant la classe de l'image x, $\mathcal{L}_{cls}$ représente l'entropie croisée, et λ un poids d'équilibrage. TASN permet ainsi d'apprendre des représentations riches et localisées à l'aide d'un seul réseau, tout en transférant efficacement les informations détaillées à travers une supervision douce et ciblée. Toutefois le modèle a une complexité architecturale et un coût de calcul élevé en raison de la génération répétée d'images d'attention et du mécanisme de distillation entre part-net et master-net. De plus, le processus repose sur des stratégies stochastiques (sélection aléatoire d'attentions) sans supervision explicite, ce qui peut limiter la robustesse sur des classes rares ou ambiguës.

D'autres approches utilisent le mécanisme d'attention en se basant sur l'architecture ViT Dosovitskiy et al. (2020) pour améliorer la classification à grain fin. C'est le cas de FFVT (*Feature Fusion Vision Transformer*) (Wang et al. 2021b) qui conserve l'architecture standard de ViT, où chaque image est divisée en $n \times n$ blocs, chacun projeté dans un espace latent de dimension d. Un token de classification est ajouté dans l'espace latent, et l'ensemble est traité par une série de L couches comprenant chacune des blocs multi-têtes d'auto-attention (MSA) et des MLPs (Voir Figure 2.9) :

$$\mathbf{z}'_l = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}, \quad \mathbf{z}_l = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l \tag{4.14}$$

où LN est une normalisation et $\mathbf{z}_l$ est la représentation vectorielle intermédiaire des blocs d'images (tokens), y compris le tocken de classification, à la couche l. À la sortie de ces L couches, on obtient une nouvelle représentation pour chaque token, et une matrice d'attention $A \in \mathbb{R}^{(n\times n+1)\times(n\times n+1)}$ qui contient le score d'attention entre les tokens ($A_{i,j}$ représente l'importance ou le poids d'attention que le token i accorde au token j).

Cependant, les couches profondes de ViT ont tendances à capturer des informations globales, au détriment des détails discriminants. Pour y remédier, FFVT introduit un module de fusion de caractéristiques, qui injecte dans la dernière couche, les tokens sélectionnées depuis les couches précédentes, en préservant le token de classification original. Ces tokens sont choisis selon leur pertinence, mesurée par un mécanisme de *Mutual Attention Weight Selection* (WAMS) (Figure 4.6).

Formellement, les tokens sélectionnées à chaque couche l sont représentés par :

$$\mathbf{z}_l^{local} = [\mathbf{z}_l^1, ..., \mathbf{z}_l^K] \tag{4.15}$$

où K corresponds au nombre de tokens sélectionnés parmi les $n \times n$ tokens (excluant le token de classification). Ces tokens sont choisis selon leur poids d'attention mutuelle (ma_i pour le token i) avec le token de classification, définis par :

$$ma_i = a'_{0,i} \cdot b'_{i,0}, \tag{4.16}$$

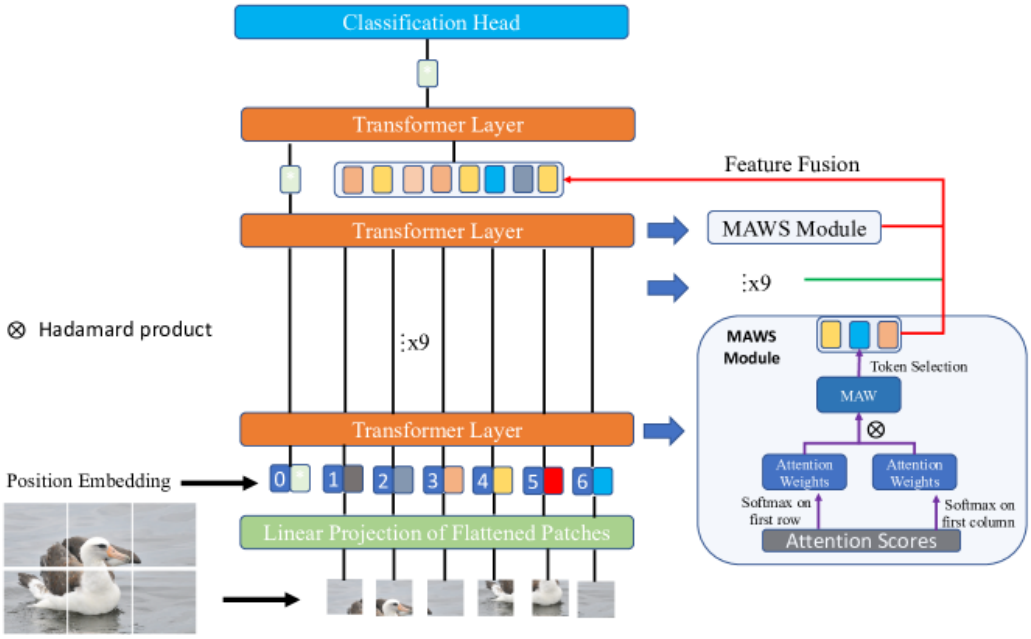


FIGURE 4.6 – Architecture du modèle FFVT. Source : Wang et al. (2021b).

avec

$$a'_{0,i} = \frac{e^{a_{0,i}}}{\sum_{j=0}^{n\times n} e^{a_{0,j}}}, \quad b'_{i,0} = \frac{e^{b_{i,0}}}{\sum_{j=0}^{n\times n} e^{b_{j,0}}}. \tag{4.17}$$

où $a_{0,j}$ est l'élément de la ligne 0 de la matrice d'attention A, c'est-à-dire le score d'attention entre le token de classification et le token j, et $b_{j,0}$ est l'élément de la colonne 0 de A, c'est-à-dire le score d'attention entre le token j et le token de classification. Ainsi, $a'_{0,i}$ représente le score d'attention normalisé que le token de classification donne au token i et $b'_{i,0}$ représente le score d'attention normalisé que le token i donne au token de classification. Cette double considération permet de favoriser les tokens fortement connectés au token de classification dans les deux directions, ce qui évite de sélectionner des tokens bruités.

Une autre approche récente qui utilise l'attention à partir d'un ViT est SIM-Trans (Sun et al. 2022). L'objectif de cette approche est de limiter la perte d'informations locale causée par la division des images en blocs et l'ignorance des relations spatiales entre les blocs. Pour atteindre ces objectifs, SIM-Trans est structuré autour de trois composants : un réseau de base basé sur ViT, un module d'apprentissage de l'information structurelle (SIL) et un module de renforcement multi-niveau des caractéristiques (MFB), comme illustré dans la Figure 4.7.

Pour exploiter les relations spatiales entre régions discriminantes dans les images, le module SIL utilise les poids d'attention entre chaque bloc et le token de classification. L'objectif est d'identifier les régions discriminantes, puis d'exprimer les relations spatiales entre ces régions sous forme de coordonnées polaires afin d'introduire une notion de structure géométrique dans le modèle. Pour atteindre cet objectif, le token de classification interagit avec tous les autres tokens dans chaque couche l du ViT. Formellement, à partir de la matrice d'attention $A \in \mathbb{R}^{(n\times n+1)\times(n\times n+1)}$ d'une couche, sont extraits les poids $Att \in \mathbb{R}^{(n\times n)\times(1)}$

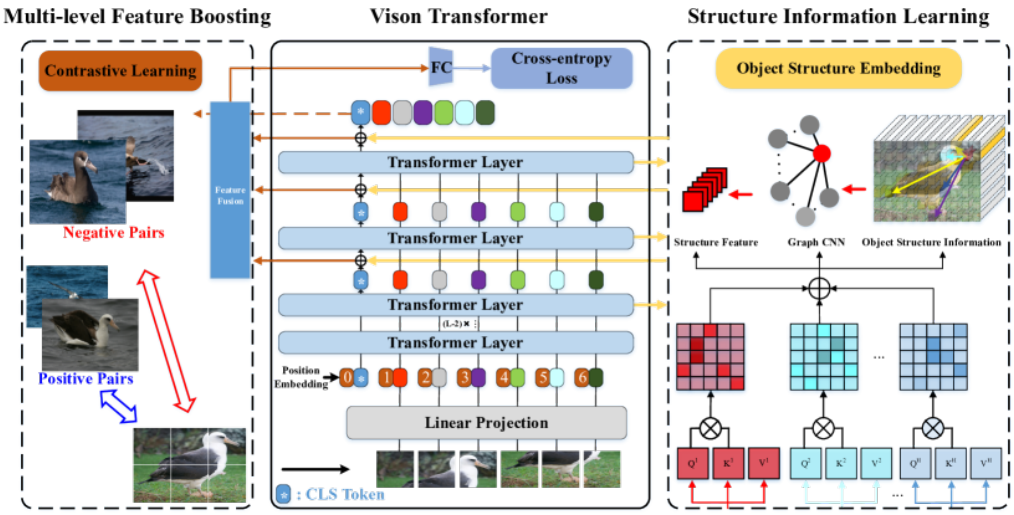


FIGURE 4.7 – Architecture du modèle SIM-Trans. Source : Sun et al. (2022).

entre chaque bloc et le token de classification pour obtenir un score de pertinence pour chaque bloc en fonction de son influence sur le token de classification. Pour éliminer les blocs peut pertinents (par exemple du fond ou du bruit), les auteurs calculent la moyenne des poids d'attention :

$$\bar{Att} = \frac{1}{n \times n} \sum_{i=1}^{n \times n} Att_i, \tag{4.18}$$

puis, ils appliquent un seuil :

$$Att^{new}_{(x,y)} = \begin{cases} Att_{(x,y)} & \text{si } Att_{(x,y)} > \bar{Att} \\ 0 & \text{sinon} \end{cases} \tag{4.19}$$

où (x, y) sont les coordonnées du bloc. Le bloc avec le plus grand poids $Att^{new}_{(x_0,y_0)}$ est désigné comme bloc de référence, considéré comme le bloc le plus important pour la classification. Les autres blocs filtrés sont ensuite exprimés relativement au bloc de référence dans un système de coordonnées polaires :

$$\rho_{x,y} = \sqrt{\left(\frac{x - x_0}{n}\right)^2 + \left(\frac{y - y_0}{n}\right)^2}, \quad \theta_{x,y} = \frac{\arctan 2(y - y_0, x - x_0) + \pi}{2\pi} \tag{4.20}$$

où $\rho_{x,y}$ est la distance radiale normalisée et $\theta_{x,y}$ est l'angle polaire. Ces relations sont utilisées pour construire un graphe dont les nœuds représentent les blocs, et les arêtes représentent les poids d'attention. À partir de la matrice d'adjacence $A_{dj} = Att^{new} \times (Att^{new})^\top$ issue de ce graphe, un GCN à deux couche est ensuite appliqué pour générer un vecteur de structure S :

$$S = \sigma(A_{dj} \cdot \sigma(A_{dj} \cdot X \cdot W^1) \cdot W^2), \tag{4.21}$$

où X représente la matrice de caractéristique des nœuds du graphe, qui décrit la corrélation du contexte spatial basé sur le calcul des coordonnées polaires, W^1, W^2 des matrices de poids apprises, et $\sigma(\cdot)$ une fonction d'activation. Ce vecteur de structure S, considéré comme la caractéristique structurelle de l'image, est ensuite fusionné avec avec le token de classification afin de renforcer l'encodage structurel dans le modèle.

En parallèle, le module MFB concatène le token de classification des trois dernières couches du ViT pour capturer des informations complémentaires à différents niveau d'abstraction. La représentation z obtenue est ensuite utilisée pour faire un apprentissage contrastif pour rapprocher les images de la même classe et éloigner celles de classes différentes, avec la perte :

$$\mathcal{L}_{CL} = \frac{1}{B^2} \sum_{i=1}^{B} \left[\sum_{j:y(i)=y(j)} \left(1 - \text{sim}(z_i, z_j^+)\right) + \sum_{j:y(i)\neq y(j)} \text{Indicator}_{i,j} \cdot \text{sim}(z_i, z_j^-) \right] \tag{4.22}$$

où B est le nombre d'image dans le batch, z_i est le vecteur de représentation de l'image i, z_j^+ est la représentation d'une image j de la même classe que i, z_j^- est la représentation d'une image j de classe différente à celle de l'image i, $\text{sim}(z_i, z_j)$ est la similarité cosinus entre les deux représentations, $y(i)$ l'étiquette de vérité terrain de l'image i et $\text{Indicator}_{i,j}$ un indicateur binaire qui vaut 1 si la paire négative i, j est difficile (similarité dépassant un certain seuil dans Att^{new}), et 0 sinon. La fonction de perte globale de SIM-Trans combine la classification standard et l'apprentissage contrastif, définie par :

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{CL}. \tag{4.23}$$

FFVT et SIM-Trans se distinguent par leur capacité à combiner efficacement les niveaux de représentation multi-échelles au sein d'un transformeur, sans ajouter de paramètres d'apprentissage supplémentaires. Toutefois, la sélection déterministe et rigide des tokens fondée exclusivement sur les scores d'attention mutuelle ne garanti pas la prise en compte de certains indices visuels subtils non corrélés au token de classification, mais pourtant discriminants. Dans nos différentes contributions, nous utilisons des connaissances a priori qui permettent de prendre en compte ce types d'informations, même ci celle-ci ne se trouve pas dans les images.

4.2.3 Utilisation de fonction de perte spécifiques

La méthode SIM-Trans (Sun et al. 2022) présentée dans la section précédente introduit une fonction de perte contrastive dans le but de renforcer la discrimination inter-classes et la cohérence intra-classe lors de l'apprentissage pour la classification à grain fin. Dans la même logique, TransFG (He et al. 2022), aussi basé sur un ViT, utilise une perte contrastive et exploite tous les tokens de classification z^l obtenues dans chacune des L couches pour

la classification. La perte contrastive est utilisée, en plus de la perte de classification, pour favoriser la proximité des représentation z de le même classe et éloigner celle des classes différentes, en n'intégrant que les paires négatives difficiles grâce à un seuil α. Cette perte est définie par :

$$\mathcal{L}_{con} = \frac{1}{B^2} \sum_{i=1}^{B} \left[\sum_{j:y_i=y_j} (1 - \text{Sim}(z_i, z_j)) + \sum_{j:y_i \neq y_j} \max(\text{Sim}(z_i, z_j) - \alpha, 0) \right] \tag{4.24}$$

où B est le nombre d'image dans le batch, z_i et z_j sont les tokens de classification des images i et j respectivement.

Dans la même logique de comparaison d'images par paire, la méthode proposée par API-Net (*Attentive Pairwise Interaction Network*) (Zhuang et al. 2020) adopte une stratégie basée sur l'interaction entre paires d'images (Figure 4.8).

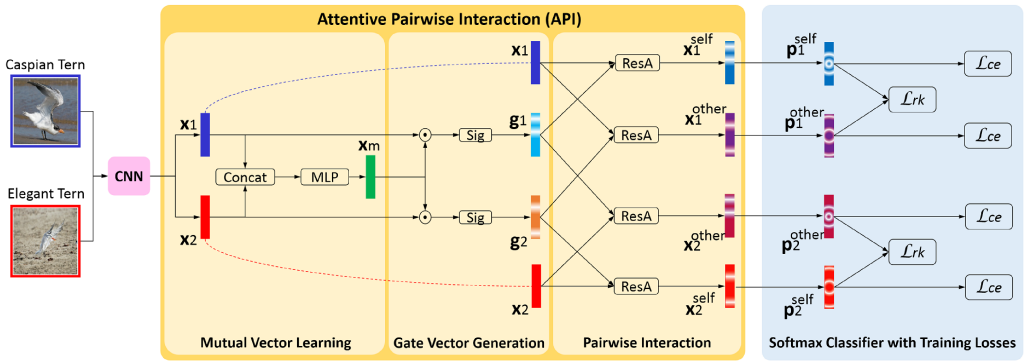


FIGURE 4.8 – Architecture du modèle API-Net. Source : Zhuang et al. (2020).

Étant donné deux vecteurs de caractéristiques $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$ extraits d'une paire d'images $x_1, x_2 \in \mathcal{X}$ en utilisant un CNN (ResNet par exemple), un vecteur mutuel $\mathbf{x}_m \in \mathbb{R}^D$ est appris via une fonction de projection f_m (un MLP) définie par :

$$\mathbf{x}_m = f_m([\mathbf{x}_1, \mathbf{x}_2]) \tag{4.25}$$

Ce vecteur a pour objectif d'encoder les différences sémantiques entre les deux images. Ensuite, API-Net génère, pour chacune des deux images, un vecteur de porte $\mathbf{g}_i$ qui agit comme une attention sur les canaux de $\mathbf{x}_i$ guidée par $\mathbf{x}_m$:

$$\mathbf{g}_i = \text{sigmoid}(\mathbf{x}_m \odot \mathbf{x}_i), \quad i \in \{1, 2\} \tag{4.26}$$

où $\odot$ représente le produit élément par élément. Ces vecteurs servent à produire des représentations d'auto-attentions $\mathbf{x}_i^{self}$ et d'attention croisées $\mathbf{x}_i^{other}$ définies par :

$$\begin{aligned} \mathbf{x}_i^{self} &= \mathbf{x}_i + \mathbf{x}_i \odot \mathbf{g}_i \\ \mathbf{x}_i^{other} &= \mathbf{x}_i + \mathbf{x}_i \odot \mathbf{g}_{\bar{i}} \end{aligned} \tag{4.27}$$

où $\bar{i}$ représente l'image opposée dans la paire. Chaque représentation $\mathbf{x}_i^j$ (avec $j \in \{\text{self}, \text{other}\}$) est ensuite soumise à une classification via un softmax :

$$\mathbf{p}_i^j = \text{softmax}(W\mathbf{x}_i^j + b) \tag{4.28}$$

où $\mathbf{p}_i$ représente la prédiction du modèle pour l'image i. Le modèle est appris en optimisant la perte :

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{rk} \tag{4.29}$$

où $\mathcal{L}_{CE}$ est l'entropie croisée sur la classe de vérité terrain y (one-hot représentation) de l'image x, et $\mathcal{L}_{rk}$ est une régularisation par classement de scores qui pénalise le modèle si la représentation croisée attribue à la classe correcte un score plus élevé que la représentation propre, avec une marge δ :

$$\mathcal{L}_{rk} = \sum_{i=1}^{2} \max\left(0, \mathbf{p}_i^{\text{other}}(c_i) - \mathbf{p}_i^{\text{self}}(c_i) + \delta\right) \tag{4.30}$$

où $\mathbf{p}_i^{\text{other}}(c_i) \in \mathbb{R}^D$ est la probabilité de prédiction obtenue pour la classe correcte c_i de l'image i. Cette régularisation renforce donc la priorité des représentation propres à une image et favorise l'apprentissage de représentation qui sont discriminantes.

Une autre méthode méthode qui modélise une fonction de perte contrastive est CIN (*Channel Interaction Network*) (Gao et al. 2020), qui s'appuie sur une modélisation des interactions entre les canaux des cartes de caractéristiques, tant à l'intérieur d'une image qu'entre paires d'images. Le pipeline (Figure 4.9) de CIN commence par l'extraction des cartes de caractéristiques X d'une image $x \in \mathcal{X}$ via un réseau CNN, suivi de deux modules : SCI (*Self-Channel Interaction*) et CCI (*Contrastive Channel Interaction*).

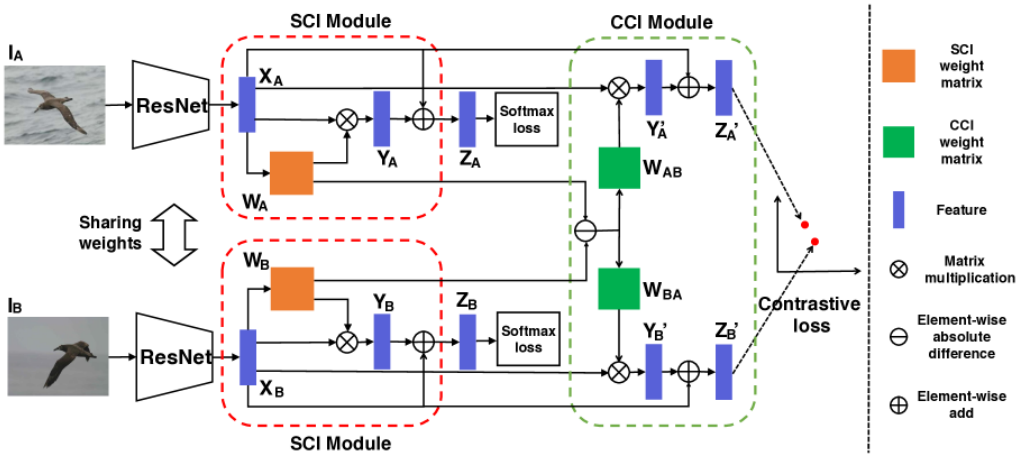


FIGURE 4.9 – Architecture du modèle CIN. Source : Gao et al. (2020).

Le module SCI vise à modéliser les relations sémantiques entre les canaux de l'image d'entrée. Étant donnée une carte de caractéristiques $X \in \mathbb{R}^{c\times w\times h}$ obtenue à la sortie du

CNN, celle-ci est aplatie en $\mathbf{X} \in \mathbb{R}^{c \times m}$, avec $m = w \times h$. Une matrice de poids $\mathbf{W} \in \mathbb{R}^{c \times c}$ est calculée pour représenter les interactions entre les canaux via une opération bilinéaire suivie d'un softmax inversé :

$$\mathbf{W}_{ij} = \frac{\exp(-\mathbf{X}_i \mathbf{X}_j^\top)}{\sum_{k=1}^{c} \exp(-\mathbf{X}_i \mathbf{X}_k^\top)} \tag{4.31}$$

Ce poids reflète à quel point le canal j est sémantiquement complémentaire au canal i. Le résultat de l'interaction pour chaque image est $\mathbf{Y} = \mathbf{W}\mathbf{X}$, et la sortie du module est une agrégation des caractéristiques initiales et enrichies Z définie par :

$$Z = \phi_{SCI}(\mathbf{Y}) + \mathbf{X} \tag{4.32}$$

où ϕ_{SCI} est une convolution 3×3. Cette opération met en valeur les corrélations négatives entre canaux afin de faire ressortir des informations complémentaires qui peuvent être négligées par l'extracteur de caractéristiques.

Pour modifier les différences fines entre deux images similaires n'appartenant pas à la même classe, le module CCI exploite les matrices de poids $\mathbf{W}_A$ et $\mathbf{W}_B$ appartenant à une paire d'images A et B. Deux matrices d'interaction contrastive $\mathbf{W}_{AB}$ et $\mathbf{W}_{BA}$ sont calculées par :

$$\mathbf{W}_{AB} = |\mathbf{W}_A - W_1 \mathbf{W}_B|, \quad \mathbf{W}_{BA} = |\mathbf{W}_B - W_2 \mathbf{W}_A| \tag{4.33}$$

où W_1 et W_2 sont des poids appris via une couche entièrement connectée prenant en entrée les paires de caractéristiques. Ces matrices mettent en évidence les différences de canaux de caractéristiques entre les images. Elles sont ensuite utilisées pour mettre à jours les représentations $\mathbf{X}_A$ et $\mathbf{X}_B$ pour y intégrer les informations contrastives apprises :

$$Z'_A = \phi_{CCI}(\mathbf{W}_{AB}\mathbf{X}_A) + \mathbf{X}_A, \quad Z'_B = \phi_{CCI}(\mathbf{W}_{BA}\mathbf{X}_B) + \mathbf{X}_B \tag{4.34}$$

où ϕ_{CCI} est une couche de convolution 3×3, Z'_A est la nouvelle représentation enrichie de l'image A et Z'_B la nouvelle représentation enrichie de l'image B. La fonction de perte finale est définie par :

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \cdot \mathcal{L}_{cont} \tag{4.35}$$

où $\mathcal{L}_{CE}$ est l'entropie croisée classique appliquée sur la sortie du module CSI, et $\mathcal{L}_{cont}$ est la perte constrastive utilisée pour distinguer les paires d'images similaires et les paires d'images dissemblables, définie par :

$$\mathcal{L}_{cont} = \begin{cases} \|\mathbf{h}(Z'_A) - \mathbf{h}(Z'_B)\|^2, & \text{si } y_A = y_B \\ \max(0, \beta - \|\mathbf{h}(Z'_A) - \mathbf{h}(Z'_B)\|)^2, & \text{si } y_A \neq y_B \end{cases} \tag{4.36}$$

avec $\mathbf{h}(\cdot)$ une fonction de projection dans un espace de dimension 512, et β une marge. Le module CCI est utilisé uniquement pendant l'entraînement. Bien que le modèle CNI semble exploiter efficacement les interactions entre canaux pour extraire des indices complémentaires et modéliser les relations inter-images, elle repose fortement sur la qualité des carte de caractéristiques extraites en amont par le CNN. Cela limite son adaptation dans les scénarios ou les données sont limités.

Les travaux qui utilisent une fonction de perte partagent l'objectif commun de renforcer la discrimination entre classes proches en mettant en évidence les différences subtiles entre images grâce à une perte contrastive. Cependant, ces méthodes apprennent exclusivement à partir de données visuelle, sans intégrer de connaissances a priori de façon explicite dans le modèle. Nous proposons, dans nos contributions, des méthodes qui intègrent des connaissances à priori dans le processus d'apprentissage, ce qui permet de guider efficacement le modèle vers des régions pertinentes et favoriser une meilleure généralisation.

4.2.4 Utilisation de graphes

KERL (*Knowledge-Embedded Representation Learning*) (Chen et al. 2018b) est une méthode qui propose une intégration explicite de connaissances sous forme de graphe dans le processus d'apprentissage (Figure 4.10). Le modèle est entraîné sur un jeu de données annoté, où chaque image x est associée à une étiquette de classe y et un ensemble de p attributs visuels qui décrivent l'image. Ces attributs fournissent des informations sémantiques détaillées telles que la couleur, la forme, ou la texture de certaines régions dans l'image. Par exemple, un oiseau peut être annoté avec des attributs comme *bec rouge*, *ailes rayées* ou *poitrine blanche*. KERL est divisé en deux principales phases : la création d'un graphe $\mathcal{G} = (V, E)$ à partir des attributs et l'intégration de ce graphe dans l'apprentissage.

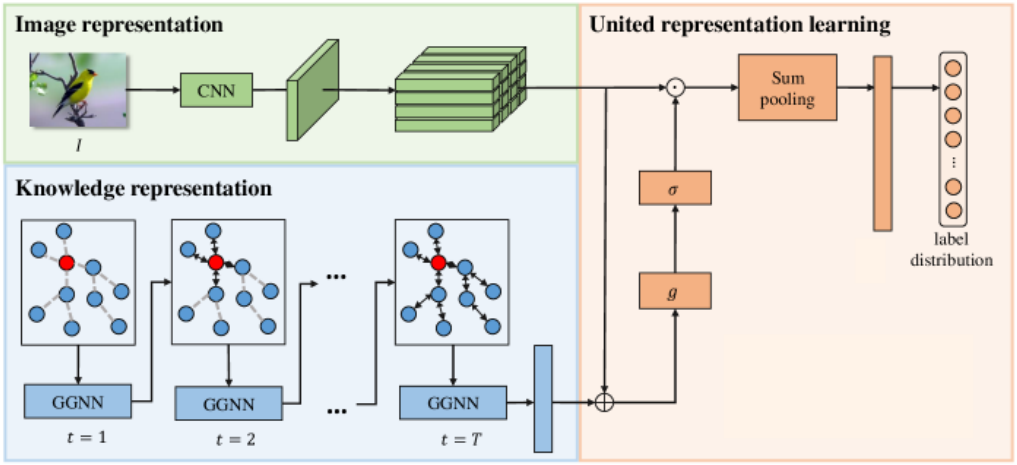


FIGURE 4.10 – Architecture du modèle KERL. Source : Chen et al. (2018b).

Le graphe $\mathcal{G}$ est construit à partir des annotations d'attributs dans le jeu de données. Chaque nœud du graphe correspond soit à une classe, soit à un attribut visuel (Figure 4.11). Les arêtes du graphe expriment les corrélations entre classes et attributs, mesurées par la fréquence moyenne de co-occurrence entre une classe et un attribut parmi les exemples

annotés. Ces valeurs sont normalisées dans une matrice $\mathcal{M} \in \mathbb{R}^{K \times p}$ (avec K le nombre de classe) utilisée pour former la matrice d'adjacence du graphe :

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{0}_{K \times K} & \mathcal{M} \\ \mathbf{0}_{p \times K} & \mathbf{0}_{p \times p} \end{bmatrix} \tag{4.37}$$

Le graphe complet $\mathbf{A}$ combine les messages directs et inverses, avec $\mathbf{A} = [\mathbf{A}_c \ \mathbf{A}_c^\top]$

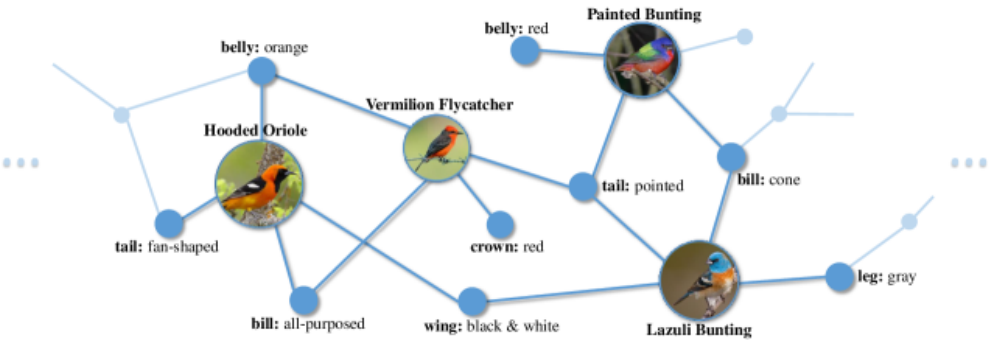


FIGURE 4.11 – Graphe construit par KERL à partir de classes et d'attributs. Source : Chen et al. (2018b).

Pour extraire une représentation riche du graphe, l'architecture utilise un GGNN qui met à jour les états cachés de chaque nœud h_v^t selon les équations de type GRU présenté à l'équation 3.22. Les nœuds correspondant à des classes sont initialisés par les scores de classification issus d'un classifieur pré-entraîné, tandis que les nœuds attributs sont initialisés à zéro. Après T itérations, les états finaux h_v^T sont transformés via une couche entièrement connectée pour obtenir les représentations o_v, concaténées pour former le vecteur global $\mathbf{H} = [o_1, \ldots, o_{\|V\|}]$ représentant la connaissance.

En parallèle, les caractéristiques d'une image x sont extraites à l'aide d'un CNN basé sur VGG, suivi d'un encodage bilinéaire compact pour produire des cartes de caractéristiques $X \in \mathbb{R}^{w \times h \times c}$, avec $c = 8192$. KERL applique ensuite un mécanisme de sélection basé sur l'attention, guidé par la connaissance pour pondérer les régions importantes :

$$\mathbf{F} = \sum_{i,j} \sigma(g(X_{i,j}, \mathbf{H})) \cdot X_{i,j} \tag{4.38}$$

où $X_{i,j}$ est la position (i, j) de la carte de caractéristiques, g est un petit réseau prenant en entrée la concaténation de la caractéristique locale $X_{i,j}$ et le vecteur de connaissance $\mathbf{H}$, et σ une fonction sigmoïde appliquée au score généré par g. $\sigma(g(X_{i,j}, \mathbf{H}))$ a pour but d'identifier les régions les plus informatives pour la tâche de classification. Elle attribue à chaque position un poids d'importance guidé par la connaissance. Le vecteur global $\mathbf{F}$ est ensuite obtenu en agrégeant ces régions pondérées, ce qui permet de concentrer la représentation finale sur les zones véritablement discriminantes.

La représentation finale $\mathbf{F}$ est enfin passée dans une couche FC pour produire les scores de classification L'ensemble du modèle est entraîné de bout en bout à l'aide d'une perte d'entropie croisée. Pendant l'inférence, seule l'image d'entrée est utilisée. Le modèle exploite

les représentations apprises par le graphe pendant l'entraînement pour guider l'attention sur les régions discriminantes de l'image, sans avoir besoin d'informations d'attributs explicites. Toutefois, KERL utilise des paramètres supplémentaires pour atteindre l'objectif de classification, et le nombre de paramètres supplémentaires dépend de la taille du graphe créé. Dans notre deuxième contribution, nous utilisons également des connaissances sous forme d'attributs visuels, mais de manière fondamentalement différente. Plutôt que de l'injecter dans l'architecture, nous créons une fonction de perte qui permet d'informer le modèle sur la difficulté de classification. Ce chois permet de contraindre l'apprentissage sans alourdir l'architecture, en guidant les représentations apprises pour qu'elles soient cohérentes avec la connaissance a priori.

4.3 Conclusion

Dans ce chapitre, nous avons ensuite présenté un panorama des méthodes récentes dédiées à la classification à grain fin, une tâche particulièrement exigeante en vision par ordinateur, où l'objectif est de distinguer des classes visuellement proches en s'appuyant sur des indices subtils. Nous avons vu que de nombreuses approches cherchent à surmonter la faible variance inter-classes en intégrant des mécanismes d'attention pour localiser automatiquement des régions discriminantes, des fonctions de perte contrastives pour renforcer l'apprentissage de différences fines entre paires d'images, ou encore des modules de comparaison inspirés du raisonnement humain.

Toutefois, bien que ces méthodes soient plus ou moins efficaces, elles reposent exclusivement sur les données visuelles, sans tirer parti de connaissances a priori. La seule méthode identifiée qui intègre explicitement des connaissances externes est KERL. Elle intègre les connaissances dans l'architecture du modèle, en ajoutant un réseau de graphe qui exploite un graphe de connaissances reliant les classes à des attributs visuels.

Notre deuxième contribution s'inscrit dans cette même logique d'enrichissement sémantique, mais se distingue sur un point fondamental : nous utilisons également des graphes créés à partir d'attributs, mais à la différence de KERL, nous les intégrons dans la fonction de perte. Notre troisième contribution intègre également des connaissances, mais dans un cadre particulier de modèle basé sur les concepts, que nous présenterons dans le chapitre suivant.

Chapitre 5

Explicabilité en classification d'images

Les réseaux de neurones convolutifs (CNN), bien que puissants pour la classification d'images et d'autres tâches de vision, sont souvent perçus comme des boîtes noires en raison de la complexité de leurs mécanismes internes et de la limite de compréhension des raisons derrière leurs décisions. Cette opacité, qui fait d'eux des boîtes noires, pose des défis pour la confiance et l'acceptation des modèles profonds, notamment dans des applications critiques comme les voitures autonomes ou les applications médicales. Pour répondre à cette problématique, l'explicabilité ou en anglais Explainable Articicial Intelligence (XAI) est essentielle. Dans ce chapitre, nous explorons les différentes approches visant à rendre les modèles de vision par ordinateur explicables, en mettant un accent particulier sur les méthodes basées sur des concepts. Cette dernière méthode nous intéressent particulièrement dans le cadre de cette thèse, car notre troisième contribution repose sur ce type de méthode pour concilier performance prédictive et capacité d'explication.

Contents

5.1 Préliminaires

Avec l'essor des réseaux de neurones profonds, les modèles de vision par ordinateur atteignent aujourd'hui des performances remarquables dans une large gamme de tâches parmi lesquels la classification d'images. Cependant, cette puissance prédictive s'accompagne d'une opacité, souvent critiquée dans les domaines sensibles où la transparence et la confiance dans les décisions algorithmiques sont cruciales. Face à cette problématique, l'explicabilité des modèles est devenue un champ de recherche à part entière, visant à fournir des justifications compréhensibles aux prédictions produites. Les travaux sur l'explicabilité des réseaux profonds en vision par ordinateur peuvent être classés en trois grandes catégories : les modèles d'architecture, les modèles de décision et les modèles basés sur les concepts.

Les modèles d'architecture s'intéressent à la structure interne des réseaux de neurones convolutifs dans le but de mieux comprendre le rôle des couches et des neurones dans le processus de prise de décision. Ils sont intrinsèques, c'est-à-dire qu'ils interviennent pendant la conception ou l'entraînement du modèle, contrairement aux approches post-hoc qui ont des techniques a posteriori. Dans ce sens, certains travaux intègrent des mécanismes d'attention (Linsley et al. 2019; Xiao et al. 2014; Wang et al. 2017a) permettant de localiser les régions visuelles importantes pour la prédiction. D'autres proposent de remplacer des composants standards, comme les couches de sous-échantillonnage ou entièrement connectées, par des alternatives plus interprétables, comme les convolutions classiques (Springenberg et al. 2014; Liang et al. 2020) ou le pooling adaptatif (Global Average Pooling (GAP)) (Lin et al. 2013) afin de préserver la correspondance entre activations et classes. Des fonctions de perte spécifiques sont également introduites (Zhang et al. 2021a; Yin et al. 2019; Li et al. 2022; Hwa Yoo et al. 2019) pour contraindre l'apprentissage à produire des activations localisées, distinctes et cohérentes, en s'appuyant sur l'information mutuelle ou la diversité des activations. En parallèle, certains modèles combinent les CNNs avec d'autres structures explicables, telles que des graphes sémantiques, des réseaux à prototypes ou des auto-encodeurs pour rendre les représentations internes plus compréhensibles (Zhang et al. 2016a; Tavanaei 2020; Chen et al. 2019a). Enfin, la simplification de l'architecture passe par des techniques comme l'extraction de règles à partir de couches internes (Zhang et al. 2019; Papernot and McDaniel 2018) en utilisant des modèles explicables (arbre de décision, classifieurs linéaires, etc.), la compression du réseau par élimination de filtres redondants (Wang et al. 2020c; Abbasi-Asl and Yu 2017), ou encore l'analyse de l'explicabilité des unités neuronales via des scores sémantiques et la visualisation de caractéristiques (Ribeiro et al. 2016; Bau et al. 2017) qui met en évidence les parties ou les attributs des données qui influencent le plus les prédictions du réseau. Ces approches permettent de rendre plus transparents les processus décisionnels des CNNs, tout en maintenant des performances compétitives.

Les modèles de décision sont généralement post-hoc et se concentrent sur l'interprétation des prédictions des CNNs en analysant les relations entre les entrées (images) et les sorties (classes prédites). Ils visent à produire des explications locales, c'est-à-dire spécifiques à chaque prédiction, en identifiant les régions ou les caractéristiques de l'image qui influencent le plus la décision du réseau. Une première approche consiste à regrouper les représentations internes en groupes (*clusters*) visuellement interprétables, puis à évaluer leur importance via des perturbations locales afin d'estimer leur contribution à la classification (Ventura and Cerquitelli 2019; Tamajka et al. 2019). D'autres travaux utilisent des réseaux générateurs ou

des décodeurs pour synthétiser ou reconstruire les images à partir des activations internes, ce qui permet de visualiser les motifs appris à différents niveaux de profondeur (Nguyen et al. 2016; Islam and Lee 2019). Des méthodes basées sur l'analyse de gradients examinent l'influence de chaque pixel ou caractéristique d'entrée en s'appuyant sur les gradients calculés lors de la rétro-propagation. Elle attribuent des scores de pertinence aux pixels en respectant des principes tels que l'invariance (Bach et al. 2015; Sundararajan et al. 2017; Shrikumar et al. 2017). Par ailleurs, plusieurs techniques de visualisation exploitent les cartes d'activation internes des CNNs pour générer des cartes de chaleur localisant les régions importantes de l'image. Certaines utilisent les gradients pour pondérer les activations (Zhou et al. 2015; Selvaraju et al. 2017; Chattopadhyay et al. 2017; Omeiza et al. 2019; Morbidelli et al. 2020), tandis que d'autres utilisent des méthodes de masquage ou d'ablation pour quantifier l'effet de chaque région sur la prédiction finale. (Wang et al. 2020a; Fong and Vedaldi 2017; Ramaswamy et al. 2020). Enfin, certains travaux intègrent directement des modules explicatifs dans la structure du réseau, par exemple en incorporant des prototypes hiérarchiques ou des modèles enseignant-élève afin de fournir des justificatifs visuels des prédictions dès l'apprentissage (Brahimi et al. 2019; Hase et al. 2019). Ces diverses stratégies offrent des niveaux variés d'explicabilité et permettent de mieux comprendre les facteurs déterminants des décisions par les modèles profonds.

Les modèles basés sur les concepts (Poeta et al. 2023) proposent une alternative plus intuitive et plus compréhensible que les approches basées sur les pixels pour expliquer les modèles. Ces modèles visent à interpréter les prédictions des réseaux de neurones en s'appuyant sur des concepts abstraits et compréhensibles par l'humain. Un concept peut être défini comme une unité sémantique ou symbolique qui regroupe des caractéristiques observables en une entité compréhensible. Par exemple, en vision par ordinateur, des concepts peuvent représenter des éléments visuels comme la texture, la couleur ou encore la forme d'un objet. Les concepts permettent de relier les décisions complexes des modèles à des notions familières, favorisant ainsi leur compréhension et leur interprétation. Par exemple pour la classification des oiseaux, les concepts pourraient être couleur des ailes, la forme du bec et le type de plumage. Dans la suite de ce chapitre, nous approfondirons cette approche, car elles constituent le fondement méthodologique de notre dernière contribution.

5.2 Modèles basés sur les concepts

Une explication à base de concepts consiste à analyser ou structurer les décisions d'un modèle en termes de concepts interprétables. Les approches basées sur les concepts fournissent une explication plus intuitive en reliant les décisions du modèle à des abstractions de haut niveau. Les méthodes d'explications basées sur les concepts se divisent en deux catégories principales (Poeta et al. 2023) : les explications post-hoc et les explications par conception. Les méthodes post-hoc analysent un modèle déjà entraîné sans modifier son architecture interne, fournissant des explications après l'entraînement. En revanche, les modèles explicables par conception intègrent directement des concepts dans leur architecture, en utilisant une couche dédiée pour prédire des scores de concepts. Ces scores quantifient la pertinence ou la présence de concepts spécifiques dans un échantillon donné et influencent les sorties du modèle. Ils permettent également d'étudier les relations entre concepts et classes ou de les visualiser. Selon les annotations et types de concepts, ces modèles peuvent

être supervisés (concepts annotés), ou non supervisés (concepts extraits automatiquement). Dans cette thèse, nous nous somme intéressés aux modèles supervisés explicables par des concepts.

Les modèles supervisés basés sur les concepts utilisent des ensembles de données annotés avec des concepts symboliques afin de superviser explicitement une couche intermédiaire dédiée à leur représentation. Lorsque ces annotations sont directement disponibles dans le même ensemble de données d'entraînement, une stratégie d'apprentissage simultané peut être utilisée, intégrant simultanément l'apprentissage des concepts et des classes de sortie. Dans les cas où ces annotations ne sont pas directement présentes, elles peuvent être incorporées dans le modèle via un apprentissage séparé sur un ensemble de données externe dédié à l'apprentissage des concepts. Cette dernière approche, appelée instillation de concepts, permet de transférer les connaissances acquises sur les concepts au modèle principal.

Les modèles basés sur les concepts nécessitent un ensemble de données composé de triplets $\mathcal{D} = \{(x_i, c_i, y_i)\}_{i=1}^{N}$, où chaque échantillon i est constitué d'une image d'entrée $x_i \in \mathcal{X}$, d'un vecteur binaire de concepts $c_i \in \mathcal{C} \subseteq \{0,1\}^o$ (indiquant si chacun des o conceptq est actif ou non dans l'image), et d'une étiquette $y_i \in \mathcal{Y}$ représentant la classe de l'image. Usuellement, $\mathbf{c}_1, \ldots, \mathbf{c}_o$ est utilisé pour représenter les o concepts qui décrivent les informations dans les images.

5.2.1 Apprentissage simultané

Les premières versions des modèles basés sur les concepts n'utilisaient pas les réseaux de neurones, et ont été utilisées pour des applications spécifiques (Kumar et al. 2009; Lampert et al. 2009). Récemment, (Losch et al. 2019; Chen et al. 2020b) ont exploré l'apprentissage de modèles basés sur des concepts via des ensembles de données auxiliaires créés par les humains. Dans les cas où les caractéristiques de l'espace d'entrée $\mathcal{X}$ sont difficiles à interpréter pour les humains, ces modèles s'appuient sur un encodeur de concepts $g : \mathcal{X} \to \mathcal{C}$ qui transforme l'image d'entrée dans un espace de concepts $\mathcal{C}$. Lorsque la vérité terrain sur les concepts est disponible, g est supervisé pour garantir $c_i \approx \hat{c}_i$ avec $\hat{c}_i = g(x_i)$. En l'absence de vérité terrain, les concepts peuvent être déduits de modèles pré-entraînés (Ghorbani et al. 2019; Magister et al. 2021). Ensuite une fonction $f : \mathcal{C} \to \mathcal{Y}$ est utilisée pour produire de façon supervisée des prédictions de classes $\hat{y}_i = f(\hat{c}_i)$ afin de respecter $\hat{y}_i \approx y_i$.

Certains modèles supervisés basés sur des concepts peuvent utiliser un seul neurone pour représenter les concepts. C'est le cas des Concept Bottleneck Models (CBM) (Koh et al. 2020), qui adoptent une architecture en deux étapes : une première fonction g (un CNN) prédit les concepts à partir de l'image d'entrée, puis une seconde fonction f (un classifieur) exploite ces concepts pour générer la prédiction finale (Figure 5.1). CBM utilise trois approches principales pour entraîner les fonctions f et g : (i) un entraînement séquentiel, où l'encodeur de concepts g est d'abord entraîné, et ses sorties sont ensuite utilisées pour entraîner f ; (ii) un entraînement indépendant, où g et f sont entraînées séparément, puis combinées pour former un modèle ; ou (iii) un entraînement joint, où g et f sont optimisées simultanément à l'aide d'une somme pondérée des pertes d'entropie croisée.

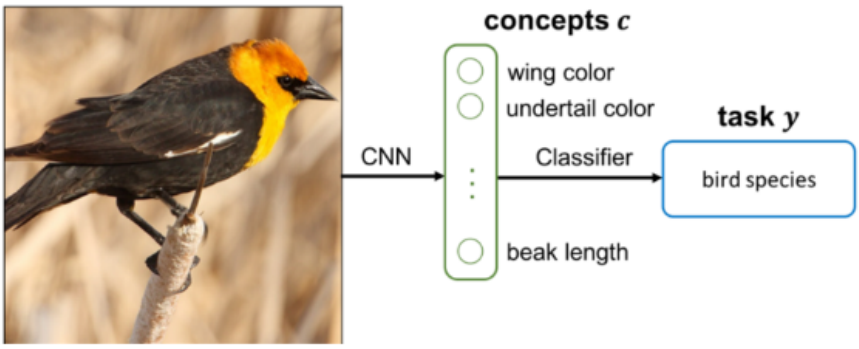


FIGURE 5.1 – Architecture du modèle CBM. Source : Koh et al. (2020).

L'utilisation d'un seul neurone pour représenter les concepts a pour conséquence de dégrader considérablement la précision de la tâche (Mahinpei et al. 2021; Zarlenga et al. 2022). Pour surmonter ce problème, CEM (Concept Embedding Model) (Zarlenga et al. 2022) utilise une représentation vectorielle obtenue par deux plongements pour représenter chaque concept $\mathbf{c}_i$ (Figure 5.2).

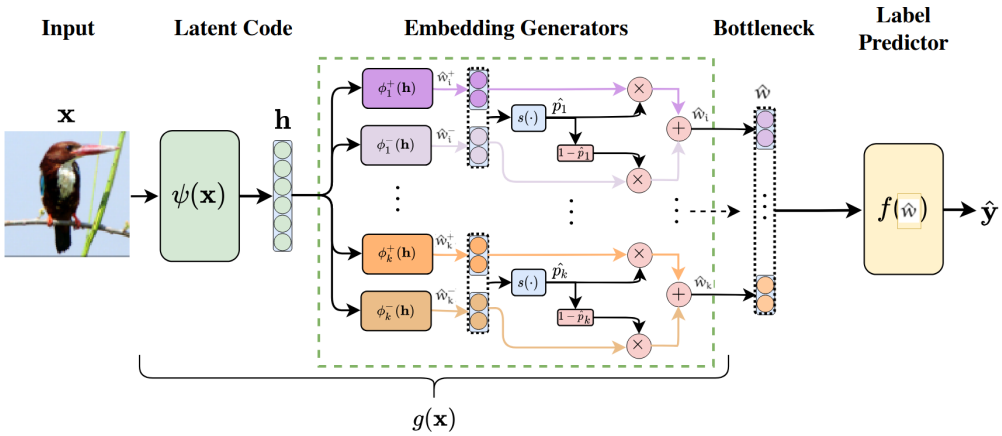


FIGURE 5.2 – Architecture du modèle CEM. Source : Zarlenga et al. (2022).

Le premier plongement, noté $\hat{w}_i^+ \in \mathbb{R}^m$, représente le concept $\mathbf{c}_i$ lorsqu'il est actif (présent dans l'image), et le deuxième plongement noté $\hat{w}_i^- \in \mathbb{R}^m$, représente le concept $\mathbf{c}_i$ lorsqu'il est inactif (non présent dans l'image). Ces plongements sont construits en passant une image x_j dans deux fonctions apprenables, ϕ_i^+ et ϕ_i^- (pour chaque concept i), implémentées comme des réseaux de neurones profonds partageant un module de prétraitement $\psi(x_j)$. Ces plongements positifs et négatifs sont ensuite transmis à une fonction s qui prédit la probabilité $\hat{p}_i$ que le concept $\mathbf{c}_i$ soit actif :

$$\hat{p}_i = s([\hat{w}_i^+, \hat{w}_i^-]). \tag{5.1}$$

Chaque plongement final $\hat{w}_i$ de concept est obtenu par une combinaison pondérée des plongements positifs et négatifs :

$$\hat{w}_i = \hat{p}_i \hat{w}_i^+ + (1 - \hat{p}_i)\hat{w}_i^-. \tag{5.2}$$

Ainsi, un encodeur de concepts $g(x_j) = \hat{w}$ est calculé, avec $\hat{w} = [\hat{w}_1, \ldots, \hat{w}_o], \hat{w}_i \in \mathbb{R}^m$. Enfin, CEM peut prédire une distribution de classe $\hat{y}$ pour l'image x_j à partir du classifieur f :

$$\hat{y} = f(\hat{w}). \tag{5.3}$$

Bien que l'approche proposée par CEM constitue une avancée importante pour améliorer l'explicabilité des réseaux, une limite notable réside dans le fait qu'elle ne prend pas en compte les interdépendances entre les concepts. En d'autres termes, les concepts sont considérés de manière indépendante, sans modéliser explicitement leurs corrélations. Cette limitation réduit la capacité du modèle à refléter fidèlement la complexité du raisonnement humain, qui repose souvent sur des interactions entre concepts. C'est dans cette direction que s'inscrit la troisième contribution de cette thèse, qui propose une méthode permettant d'intégrer explicitement les dépendances entre concepts à partir de connaissances a priori, afin de renforcer la cohérence des prédictions produites.

ProbCBM (Probabilistic Concept Bottleneck Models) (Kim et al. 2023) étend le cadre des modèles basés sur les concepts en intégrant des notions d'incertitude dans les représentations des concepts. Contrairement à CBM et CEM qui, pour chaque concept, calculent une probabilité que ce concept soit actif, ProbCBM modélise l'incertitude des concepts via des plongements probabilistes. Chaque concept $\mathbf{c}_i$ est modélisé via une distribution gaussienne $p(\hat{w}_i|x) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ générée par un module d'encodage probabiliste (PME) (Figure5.3).

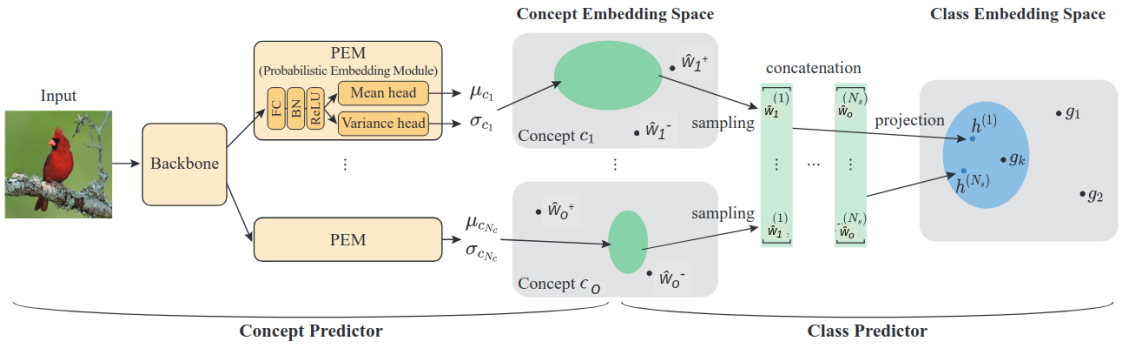


FIGURE 5.3 – Architecture du modèle ProbCBM. Source : Kim et al. (2023).

La probabilité d'existence du concept $\mathbf{c}_i$ est estimé via un échantillonage Monte Carlo, en comparant les distances des vecteurs échantillonnés aux deux prototypes $\hat{w}_i^+$ et $\hat{w}_i^-$, selon :

$$p(c_i = 1|x) \approx \frac{1}{N_s} \sum_{n=1}^{N_s} s\left(a\left(\|\hat{w}_i^{(n)} - \hat{w}_i^-\|^2 - \|\hat{w}_i^{(n)} - \hat{w}_i^+\|^2\right)\right), \tag{5.4}$$

où N_s est le nombre d'échantillons Monte Carlo tirés pour chaque concept, s la fonction sigmoïde et a un paramètre appris.

Ce mécanisme permet de quantifier l'incertitude sur l'existance de chaque concept en tenant compte de la dispersion des plongements échantillonnés autour de leur moyenne. La prédiction de la classe est ensuite obtenue à partir de la concaténation des plongements de concepts, projeté dans un espace de classe via une transformation linéaire définie par :

$$h^{(n)} = W[\hat{w}_1^{(n)}, \ldots, \hat{w}_o^{(n)}] + b \tag{5.5}$$

où W est une matrice de projection linéaire et b le biais. On estime alors la probabilité d'appartenance à chaque classe par une softmax pondérée utilisant la distance aux prototypes de classes g_k :

$$p(y_k = 1|x) \approx \frac{1}{N_s} \sum_{n=1}^{N_s} \frac{\exp(-d\|h^{(n)} - g_k\|^2)}{\sum_{k'} \exp(-d\|h^{(n)} - g_{k'}\|^2)} \tag{5.6}$$

où d est un paramètre appris. Ce processus permet de propager l'incertitude des concepts vers la prédicteur de la classe finale, offrant ainsi une meilleure robustesse face aux cas ambigus.

Le prédicteur de concept et le prédicteur de classe sont entraînés séparément, chacun avec sa propre fonction de perte. La fonction de perte pour le prédicteur de concepts est donnée par :

$$\mathcal{L}_{\text{concept}} = \mathcal{L}_{\text{BCE}} + \lambda_{\text{KL}} \cdot \mathcal{L}_{\text{KL}} \tag{5.7}$$

où $\mathcal{L}_{\text{BCE}}$ est l'entropie croisée binaire, et $\mathcal{L}_{\text{KL}}$ est la divergence de Kullback-Leibler entre la distribution prédite pour le concept $\mathbf{c}_i$ et une gaussienne standard $\mathcal{N}(0, I)$ jouant le rôle de régularisation.

Même si Prob-CBM introduit une modélisation innovante de l'incertitude via des embeddings probabilistes, son schéma d'apprentissage repose sur une procédure séquentielle qui constitue une limitation importante. En effet, le prédicteur de concepts est d'abord entraîné indépendamment pour approximer les distributions des concepts à partir des entrées, puis une fois figé, ses sorties sont utilisées comme base d'entraînement du prédicteur de classe. Cette séparation empêche toute co-adaptation dynamique entre les deux modules au cours de l'apprentissage.

DCR (Deep Concept Reasoners) (Barbiero et al. 2023) repoussent les limites des modèles basés sur les concepts explicables en combinant des représentations de plongement des concepts avec des règles logiques différentiables pour fournir des prédictions entièrement explicables. Contrairement aux modèles précédents où les prédictions sont directement basées sur des plongements des concepts souvent dépourvus de signification sémantique claire, DCR apprend des règles logiques symboliques et différentiables reliant directement les concepts aux classes, permetant une explication complète du processus de décision.

Dans ce cadre, les rôles d'un concept $\mathbf{c}_i$ sont définis à partir de deux fonctions. Une première fonction $\phi_i : \mathbb{R}^m \to [0, 1]$ qui évalue le degré de vérité (positif ou négatif) du concept $\mathbf{c}_i$ dans une règle logique, et une deuxième fonction $\psi_{ik} : \mathbb{R}^m \to [0, 1]$ qui indique dans quelle mesure le concept $\mathbf{c}_i$ est pertinent pour prédire la classe k.

Les règles pour chaque classe k sont ensuite construites comme une conjonction pondérée des contributions des concepts pertinents. L'expression logique prédictive pour la classe k est formulée comme :

$$\hat{y}_k = \bigwedge_i \psi_{ij}(\mathbf{c}_i) \cdot \phi_i(\mathbf{c}_i) \tag{5.8}$$

où $\psi_{ij}(\mathbf{c}_i)$ joue le rôle d'un poids d'activation du concept $\mathbf{c}_i$ pour la classe k, $\phi_i(\mathbf{c}_i)$ encode sa valence logique (présence ou absence), et $\bigwedge$ est une conjonction différentiable. Autrement dit, pour chaque classe, DCR construit une règle logique continue de type : la classe k est activée si tous les concepts pertinents pour k sont évaluée comme vrais. Cette construction rend chaque prédiction interprétable comme une règle logique.

5.2.2 Instillation de concepts

L'instillation de concepts est proposée par Chen et al. (2020b), qui intègre un module d'alignement de concepts pour modifier une couche donnée d'un réseau CNN afin de mieux comprendre le calcul menant à cette couche. Le module d'alignement de concepts se compose de deux parties : l'alignement, qui permet de décorréler et de normaliser les données, et la transformation orthogonale. Pendant l'entraînement, chaque concept est associé à un axe orthogonal de l'espace latent, et les données sont projetées sur ces axes pour capturer la correspondance avec ces concepts. CME (Concept-based Model Extraction) (Kazhdan et al. 2020) extrait des modèles explicables à partir d'un réseau profond préentraîné pour la classification sans concepts. CME transforme le réseau d'origine en deux fonctions, f et g. La fonction g est obtenue en analysant les différentes couches du réseau de neurones pour identifier celles qui contiennent le plus d'informations utiles sur les concepts. Ensuite, un modèle est entraîné pour prédire ces concepts à partir des activations de ces couches. Une fois les concepts extraits, un second modèle f est entraîné pour apprendre à prédire la classe finale en utilisant uniquement ces concepts. Dans la même logique, PCBM (Post-Hoc Concept Bottleneck Models) (Yuksekgonul et al. 2022) permet d'appliquer une structure explicable par des concepts à des modèles préentraînés dans un cadre post-hoc. PCBM apprend d'abord une base de concepts en projetant les représentations des données sur des vecteurs d'activation de concepts, puis utilise un modèle interprétable, comme une régression linéaire, pour effectuer la classification à partir de ces concepts.

CT (Attention-Based Interpretability with Concept Transformers) (Rigotti et al. 2022) s'appuie sur un mécanisme d'attention croisée entre les caractéristiques des images d'entrée x et des représentations denses des concepts pour relier les sorties du modèle à des concepts. L'espace de concepts modélise une matrice d'attention qui est utilisée pour calculer les probabilités de classification. Contrairement aux autres modèles basés sur les concepts, CT garantit à la fois la plausibilité (les explications sont convaincantes pour un humain) et la fidélité (les explications reflètent réellement le processus de décision du modèle) en intégrant des contraintes explicites dans la fonction de perte.

Les méthodes reposant sur l'instillation de concepts ne sont pas retenues pour la comparaison avec notre troisième contribution, car elles poursuivent des objectifs différents et s'inscrivent dans un cadre méthodologique distinct.

5.2.3 Interventions sur les concepts

Une propriété clé des modèles basés sur les concepts est leur capacité à permettre des interventions sur les concepts. Cela signifie qu'un expert humain peut corriger les prédictions erronées des concepts en imposant, à l'étape de test, une valeur correcte au concept mal prédit. Suite à cette intervention, le modèle met à jour la représentation des différents concepts et propage ce changement vers le prédicteur final pour améliorer les performances de la prédiction globale. En pratique, il est souvent irréaliste pour un humain d'intervenir sur tous les concepts ou de les sélectionner aléatoirement pour les corriger. Par conséquent, il est essentiel d'établir un ordre de priorité pour les interventions sur les concepts. L'objectif est de concentrer les efforts d'intervention sur les concepts les plus informatifs pour la tâche finale, mais aussi les plus difficiles à prédire pour le modèle. Cette approche optimise à la fois les ressources humaines et la précision des prédictions, en s'assurant que les corrections portent en priorité sur les concepts ayant le plus d'impact sur la performance du modèle. Pour atteindre cet objectif, de nombreux travaux ont introduit des politiques d'intervention pour déterminer un ordre optimal d'intervention sur les concepts.

Les politiques d'intervention dans les modèles basés sur les concepts permettent d'optimiser l'ordre des corrections de concepts pour améliorer les prédictions finales en minimisant le coût humain sur les modèles après que ceux-ci soient déjà entraînés. Parmi ces politiques, on distingue principalement (1) la politique aléatoire (Random) où les concepts sont sélectionnés de manière aléatoire. Cette stratégie constitue généralement une référence de base pour comparer d'autres politiques. (2) La politique basée sur l'incertitude (UCP) (Shin et al. 2022) cible les concepts avec la plus grande incertitude de prédiction, mesurée par $\frac{1}{|\hat{p}_i - 0.5|}$ pour cibler les concepts les plus difficiles à prédire par le modèle. (3) La politique coopérative (CooP) (Chauhan et al. 2023) maximise une combinaison linéaire entre l'incertitude de prédiction des concepts et l'impact attendu de l'intervention sur la prédiction finale. Cette approche permet une intervention dynamique basée sur les spécificités de chaque image. (4) Certaines politiques sont plus statiques, comme la CVA (Concept Validation Accuracy) (Koh et al. 2020) qui ordonne les concepts en fonction de leurs erreurs de validation, en privilégiant ceux ayant les plus grandes erreurs, tandis que la CVI (Concept Validation Improvement) (Chauhan et al. 2023) sélectionne les concepts qui améliorent le plus la précision du modèle lorsqu'ils sont corrigés. Enfin, la politique Skyline (Zarlenga et al. 2022) agit comme un oracle, sélectionnant à chaque étape le concept ayant le plus grand effet possible sur la prédiction finale. Toutes ces politiques diffèrent les unes des autres par leur niveau d'efficacité et leur coût en calcul.

Inspiré de CEM, IntCEM (Intervention-Aware Concept Embedding Models) (Espinosa Zarlenga et al. 2023) propose un modèle explicable et interactif qui apprend à anticiper les interventions humaines pendant l'entraînement, afin de mieux y répondre lors de l'inférence (Figure 5.4).

À chaque étape d'apprentissage, IntCEM simule des interventions humaines en introduisant un processus stochastique contrôlé. Tout d'abord, un masque initial d'intervention $\mu^{(0)} \in \{0, 1, \perp\}^k$ est échantillonné depuis une distribution à priori $p(\mu)$. Ce masque indique, pour chaque concept, s'il est observé (fourni au modèle, 1), non observé (à prédire, 0) ou à corriger (éligible à une correction, $\perp$). Ensuite, un nombre d'interventions à effectuer $T \in \{1, \ldots, k\}$ est tiré d'une distribution discrète $p(T)$ pour simuler T corrections successives.

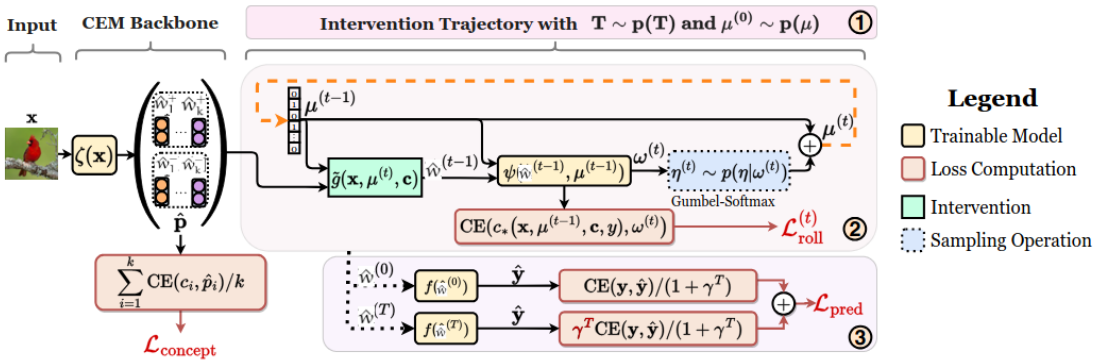


FIGURE 5.4 – Architecture du modèle IntCEM. Source : Espinosa Zarlenga et al. (2023).

À chaque étape $t \in \{1, \ldots, T\}$, une intervention unique $\eta^{(t)} \in \{0, 1\}^k$ sur un concept est échantillonnée à partir d'une distribution catégorique paramétrée par un vecteur $\omega^{(t)}$. Ce vecteur est prédit par une politique d'intervention $\psi(\hat{W}^{(t-1)}, \mu^{(t-1)})$, évaluée sur la représentation des concepts actuels $\hat{W}^{(t-1)} = [\hat{w}_1^{(t-1)}, \ldots, \hat{w}_o^{(t-1)}]$ et le masque courant $\mu^{(t-1)}$. L'état du masque est ensuite mis à jour par :

$$\mu^{(t)} = \mu^{(t-1)} + \eta^{(t)}. \tag{5.9}$$

La nouvelle représentation des concepts est notée $\hat{W}^{(t)} = \tilde{g}(x, \mu^{(t)}, c)$, où $\tilde{g}(\cdot)$ est un encodeur de concepts.

L'apprentissage du modèle est guidé par une fonction de perte composée, définie comme :

$$\mathcal{L}(x, c, y, T) = \lambda_{\text{roll}} \cdot \mathcal{L}_{\text{roll}} + \mathcal{L}_{\text{pred}} + \lambda_{\text{concept}} \cdot \mathcal{L}_{\text{concept}}, \tag{5.10}$$

où λ_{roll} apprend à la politique ψ à imiter un oracle en choisissant à chaque étape le concept à corriger maximisant la probabilité correcte de prédiction. Ce concept est défini par :

$$c^*(x, \mu, c, y) = \arg\max_i f(\tilde{g}(x, \mu \vee 1_i, c))_y, \tag{5.11}$$

avec $\mu \vee 1_i$ indiquant l'ajout d'une intervention sur le concept i. La perte qui permet d'imiter l'oracle est donnée par :

$$\mathcal{L}_{\text{roll}} = \frac{1}{T} \sum_{t=1}^{T} \text{CE}(c^*(x, \mu^{(t-1)}, c, y), \psi(\hat{W}^{(t-1)}, \mu^{(t-1)})), \tag{5.12}$$

où CE désigne la perte d'entropie croisée.

$\mathcal{L}_{\text{pred}}$ pénalise les erreurs de prédiction de classe avant et après la trajectoire d'intervention, en mettant plus de poids sur la performance finale via un facteur γ^T :

$$\mathcal{L}_{\text{pred}} = \frac{\text{CE}(y, f(\tilde{g}(x, \mu^{(0)}, c))) + \gamma^T \cdot \text{CE}(y, f(\tilde{g}(x, \mu^{(T)}, c)))}{1 + \gamma^T}, \tag{5.13}$$

avec $\gamma > 1$.

$\mathcal{L}_{\text{concept}}$ est la perte de prédiction des concepts, classiquement définie comme une entropie croisée sur les prédictions $\hat{p}$. L'objectif global est d'optimiser les paramètres du modèle θ via :

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(x,c,y)\sim\mathcal{D}} \left[\mathbb{E}_{\mu\sim p(\mu),\, T\sim p(T)} \mathcal{L}(x, c, y, T) \right], \tag{5.14}$$

où l'espérance intérieure est approximée par échantillonnage Monte Carlo.

En apprenant activement comment tirer profit d'interventions humaines, IntCEM améliore significativement les performances en contexte interactif, tout en maintenant la lisibilité des décisions via des concepts interprétables. Toutefois, l'ajout d'une politique d'intervention et la simulation de trajectoires rendent l'apprentissage plus complexe et coûteux que les CBM standards.

5.3 Conclusion

Dans ce chapitre, nous avons brièvement présenté différentes approches d'explicabilité en vision par ordinateur, en mettant un accent particulier sur les méthodes basées sur les concepts. Ces méthodes reposent sur l'idée que la prédiction finale d'un modèle peut être expliquée de manière transparente à travers un ensemble de concepts sémantiques intermédiaires. Cependant, une limite commune à la majorité de ces approches est qu'elles traitent les concepts de manière indépendante, ignorant les relations de co-occurrence naturellement présentes entre eux. Dans de nombreux contextes réels, certains concepts apparaissent fréquemment ensemble ou sont mutuellement exclusifs, et ces dépendances structurelles peuvent fournir des indices précieux pour améliorer la robustesse du modèle. Aussi, les modèles purement basés sur les concepts, malgré leur intérêt en termes d'explicabilité, sont souvent moins performants en classification que les modèles standards qui prédisent directement la sortie à partir des données brutes. En effet, le passage par une représentation intermédiaire contraint le flux d'informations.

Notre dernière contribution apporte une réponse à ces lacunes en introduisant un modèle basé sur les concepts enrichi par des règles dérivées de la co-occurrence des concepts. Ces règles viennent guider la prédiction du modèle par l'intermédiaire d'un graphe, permettant de corriger ou de contraindre certaines sorties des concepts en fonction du contexte global de l'image. Notre objectif n'est pas d'améliorer l'explicabilité, mais de proposer un compromis équilibré entre explicabilité et précision des prédictions.

Deuxième partie

Contributions

Chapitre 6

Création et enrichissement de bases d'images avec des connaissances

L'utilisation de connaissances à priori en complément des données brutes est de plus en plus répandue dans la recherche pour améliorer la classification d'images dans des scénarios complexes. Cependant, le nombre de jeux de données de classification d'images intégrant explicitement de telles connaissances reste très limité, ce qui constitue un frein aux avancées dans ce domaine. Ce chapitre présente un processus d'enrichissement de bases d'images existantes pour créer des jeux de données innovants pour la problématique de classification d'images (Mbiaya et al. 2024a). Ces jeux de données intègrent des informations supplémentaires formalisées sous forme d'attributs. Par ailleurs, nous introduisons une approche visant à enrichir ces ensembles d'images créés, ou d'autres, ayant des attributs additionnels, en intégrant des règles dérivées de ces attributs.

Contents

6.1 Motivations

Dans le domaine de la vision par ordinateur, les architectures dapprentissage profond ont démontré une efficacité remarquable. Cependant, ces modèles reposent sur laccès à de vastes volumes de données pour atteindre des performances optimales, ce qui représente une contrainte majeure dans de nombreux scénarios. Lintégration de connaissances à priori dans le processus dapprentissage offre une alternative prometteuse pour réduire cette dépendance aux données massives. Ces connaissances, formalisées sous forme dattributs, de graphes, dontologies, de textes, ou encore de règles, permettent aux modèles de mieux appréhender les relations sémantiques des données et daméliorer leur capacité à résoudre des tâches complexes, telles que la classification à grain fin ou lexplicabilité des modèles.

Malgré leur potentiel, les jeux de données intégrant explicitement des connaissances a priori restent rares, ce qui freine les avancées dans ce domaine. Afin de pallier cette insuffisance, nous proposons de nouveaux jeux de données, nommés COCO@x (COCO@11 , COCO@24 et COCO@48) (Mbiaya et al. 2024a), étiquetés et enrichis par des attributs supplémentaires. Ce chapitre introduit également dautres jeux de données créés en sinspirant de la méthode utilisé pour créer COCO@x. Construits à partir de MS COCO 2014 (Lin et al. 2014) pour COCO@x par exemple, ces jeux de données peuvent être adaptés pour répondre à des besoins spécifiques grâce à leur structure générique et paramétrable, étendant ainsi leur utilité à une variété de tâches. De plus, nous proposons un enrichissement des jeux de données existants grâce à lintégration de connaissances à priori. Ces connaissances sont formalisées sous forme de règles qui définissent les relations entre les attributs et les classes.

En parallèle de nos contributions, plusieurs travaux récents se sont attachés à structurer la recherche neuro-symbolique autour de jeux de données dédiés. Par exemple, Bourguin and Lewandowski (2024) proposent des jeux d'images ontologiques où chaque instance est explicitement reliée à une base de connaissances formelle, tandis que des initiatives telles que RSBench (Bortolotti et al. 2024) visent à évaluer de manière standardisée la capacité des modèles à exploiter des règles symboliques ou des contraintes logiques. Contrairement à ces benchmarks centrés sur des ontologies riches ou des structures logiques complexes, notre approche se concentre sur l'intégration d'attributs visuels et de règles simples mais directement exploitables pour la classification à grain fin. Les jeux de données que nous proposons visent ainsi une complémentarité : ils se situent entre les benchmarks fortement symboliques, orientés raisonnement logique, et les jeux de données visuels traditionnels, en offrant un cadre plus léger, paramétrable et directement applicable aux tâches de reconnaissance d'images classiques.

6.2 Création de bases dimages

Pour mieux illustrer la création des bases dimages, nous allons dans la suite de ce section, utiliser les jeux de données étendus à partir de MS COCO (Microsoft Common Objects in Context) (Lin et al. 2014). Les jeux de données COCO@x, où x indique le nombre de classes, sont dérivés de la version initiale (2014) de ce jeu de données.

MS COCO est un jeu de données destiné à la détection dobjets, composé d'environ

82 000 images d'entraînement, 41 000 images de validation et 41 000 images de test. Contrairement à certains jeux de données, aucune classe nest prédéfinie, les images ne sont donc pas étiquetées. Toutefois, le jeu de données contient 80 objets distincts, répartis sur les images avec une moyenne de 2,9 objets par image dans lensemble dentraînement. Notre but dans ce chapitre double : créer des classes dimages à partir des objets qui les composent, et créer des règles exprimant des propriétés entre les classes créées et la présence dobjets dans les images de ces classes.

Les nouveaux jeux de données COCO@x sont construits à partir de MS COCO, en exploitant la diversité des objets présents dans les images. Chaque jeu de données COCO@x est étendu pour répondre à une tâche de classification dimages. Seules les images des ensembles dentraînement et de validation sont utilisées, car lensemble de test initial ne fournit pas dinformations sur la présence dobjets dans les images. Lensemble de validation est utilisé pour constituer lensemble de test dans les nouveaux jeux de données.

La création des jeux de données COCO@x à partir de MS COCO repose sur trois étapes principales (Figure 6.1). Dans un premier temps, des motifs dobjets fermés fréquents [1] sont générés à partir des objets représentés dans les images. Ensuite, les classes sont définies à laide de lalgorithme ECCLAT (Durand and Crémilleux 2003), qui identifie des ensembles pertinents dimages en fonction des motifs identifiés. Enfin, des connaissances sont formalisées sous forme de règles, dérivées des classes et des objets qui les caractérisent. Nous présenterons lalgorithme de génération des règles dans la section suivante, car celui-ci est aussi utilisé pour générer les règles dans dautres jeux de données.

La création des classes dans les jeux de données COCO@x repose sur lanalyse formelle de concepts (Ganter et al. 2005). Chaque classe est associée à un motif, cest-à-dire un ensemble de descripteurs booléens indiquant la présence dun objet dans limage. Un motif est dit fermé lorsque les propriétés quil exprime sont satisfaites par les images décrivant la classe correspondante, et uniquement par celle-ci. Cela permet de caractériser précisément les propriétés dune classe. Lors de lapplication de la méthode ECCLAT pour construire les classes, la condition stricte davoir un motif fermé pour chaque classe est assouplie afin de garantir la formation dune partition complète des données.

6.2.1 Analyse formelle de concepts

Dans cette section, nous utiliserons les notions sur la FCA présentées dans la Section 2.2. Soit $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ un ensemble de données transactionnelles où $\mathcal{T}$ représente un ensemble de transactions, $\mathcal{I}$ représente un ensemble ditems et $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ est une relation binaire entre les transactions et les items. Chaque couple $(\mathtt{t}, \mathtt{i}) \in \mathcal{R}$ indique que la transaction $\mathtt{t}$ est associée à litem $\mathtt{i}$.

Dans notre contexte, nous travaillons avec un jeu de données composé dimages contenant des objets. Par conséquent, les transactions sont désignées comme des images et les items sont des objets. Ainsi, la paire $(\mathtt{t}, \mathtt{i}) \in \mathcal{R}$ représente la présence de lobjet $\mathtt{i}$ dans limage $\mathtt{t}$. Un itemset sera alors appelé **motif dobjets**, et nous utiliserons des motifs fermés fréquents dobjets pour créer les classes.

1. Un motif dobjets fermé fréquent est un ensemble dobjets fréquent dans une base de données transactionnelle, pour lequel tout sur-ensemble a un support strictement inférieur.

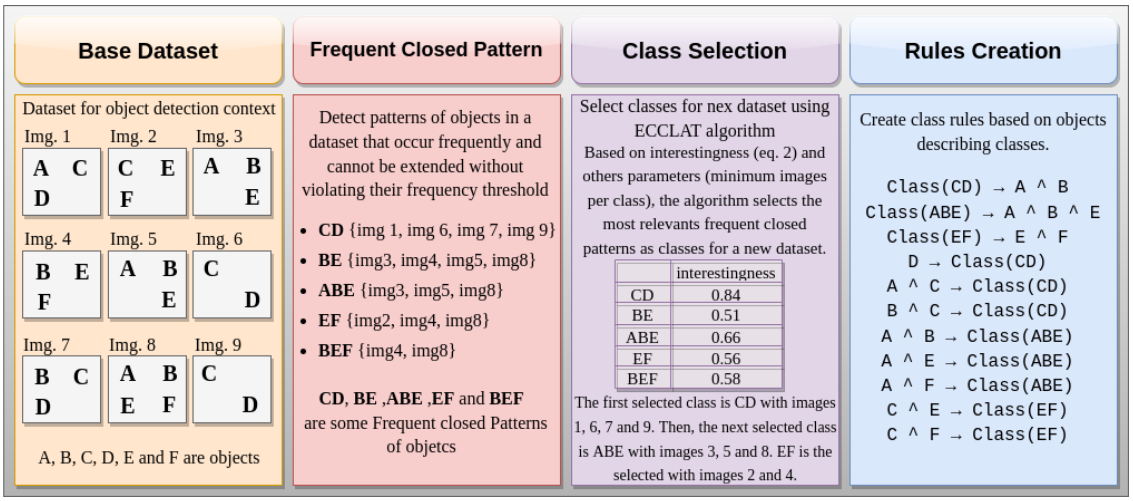


FIGURE 6.1 – Étapes de création des jeux de données COCO@x

6.2.2 Motifs fermés fréquents dobjets

Nous rappelons quun motif fermé dobjets est un ensemble maximal dobjets présents dans un groupe dimages, et uniquement dans celles-ci. Plus formellement, à partir dun ensemble de relations binaires entre un ensemble dimages et un ensemble dobjets, un ensemble Z est un motif fermé fréquent dobjets si $f(g(Z)) = Z$, où $g(Z)$ représente lensemble des images contenant tous les objets de Z, et $f(g(Z))$ représente lensemble des objets communs à toutes les images de $g(Z)$. La Table 6.1 illustre un exemple de contexte de reconnaissance dobjets, comprenant 9 images (avec des identifiants uniques) et 6 objets étiquetés de A à F. Une croix dans la table indique la présence dun objet dans une image donnée. Cette table sert de base aux exemples qui seront développés dans la suite de ce chapitre.

Images	**Objets**					
	A	B	C	D	E	F
1	×		×	×		
2			×		×	×
3	×	×			×	
4		×			×	×
5	×	×			×	
6			×	×		
7		×	×	×		
8	×	×			×	×
9			×	×		

TABLE 6.1 – Exemple de base de données pour la reconnaissance dobjets.

Nous considérons des motifs fermés dobjets X qui satisfont deux conditions. Premièrement, ils doivent être fréquents, cest-à-dire que leur fréquence $|g(X)|$ (le nombre dimages les contenant) doit être supérieure ou égale à un seuil de fréquence spécifique minfr ($|g(X)| \geq$ minfr). Deuxièmement, X doit contenir au moins minobj objets ($|X| \geq$ minobj). Chaque motif fermé est associé à un ensemble dimages, à savoir les images qui contiennent

les objets du motif. Cet ensemble dimages forme une classe candidate pour le nouveau jeu de données.

En utilisant lexemple de base de la base de données transactionnelle dimages de la Table 6.1, les motifs fermés fréquents dobjets avec les paramètres minfr $= 2$ et minobj $= 2$ sont **CD**, **BE**, **ABE**, **EF** et **BEF** (Table 6.2). En effet, les motifs **AB** et **AE**, par exemple, ne sont pas fermés car ils possèdent le même support que le super motif **ABE**. Pour le jeu de données MS COCO, nous avons utilisé exclusivement lensemble dentraînement pour générer les motifs fermés fréquents dobjets.

6.2.3 Création des classes

Dans le cadre de la discussion qui va suivre, X représente un motif fermé fréquent dobjets. Lobjectif est de construire des classes qui correspondent à des ensembles dimages liés aux motifs. Rappelons que chaque motif fermé est associé à un ensemble dimages qui satisfont ce motif. Lidée consiste à sélectionner les motifs les plus pertinents pour former des classes dobjets.

Mesures dévaluation des classes

Pour la sélection des classes, nous utilisons la méthode ECCLAT (Durand and Crémilleux 2003), qui évalue lintérêt dun motif fermé fréquent X en se basant sur une mesure de *pertinence* qui est définie par :

$$\mathrm{pertinence}(X) = \frac{\mathrm{homogénéité}(X) + \mathrm{concentration}(X)}{2} \tag{6.1}$$

Lhomogénéité dun motif fermé fréquent dobjets X est calculée dans le but de privilégier les classes potentielles contenant de nombreux objets communs à un grand nombre dimages. Elle est définie par :

$$\mathrm{homogénéité}(X) = \frac{|g(X)| \times |X|}{\mathrm{divergence}(X) + |g(X)| \times |X|} \tag{6.2}$$

où $\mathrm{divergence}(X) = \sum_{t \in g(X)} |o(t) - X|$, avec $o(t)$ représentant les objets présents dans limage t.

La divergence dun motif fermé fréquent X correspond, pour chaque image dans $g(X)$, au nombre dobjets qui ne font pas partie de X. Plus la divergence est grande, plus lhomogénéité du motif tend vers 0. La valeur dhomogénéité varie entre 0 et 1. Un motif fermé fréquent est davantage susceptible dêtre considéré comme une classe lorsquil présente un degré élevé dhomogénéité.

Dans la mesure de concentration, une préférence est accordée aux motifs fermés fréquents X dont les images associées apparaissent moins fréquemment dans dautres motifs fermés fréquents. Cette approche permet de limiter le chevauchement entre les différentes classes sélectionnées. La concentration de X est définie par :

$$\texttt{concentration}(X) = \frac{1}{|g(X)|} \times \sum_{t \in g(X)} \frac{1}{F(t)} \tag{6.3}$$

où $F(t)$ représente le nombre de motifs fermés fréquents dans lesquels limage t apparaît. La valeur de la $\texttt{concentration}$ varie entre 0 et 1. Lorsque les images de $g(X)$ satisfont plusieurs motifs parmi lensemble des motifs fermés fréquents, la valeur de la $\texttt{concentration}$ tend vers 0. Si toutes les transactions (images) ne satisfont que le motif fermé fréquent X, alors sa concentration est égale à 1. Un motif fermé fréquent dobjets avec une concentration proche de 1 est donc davantage susceptible dêtre sélectionné pour former une classe.

La $\texttt{pertinence}$ permet donc de sélectionner des classes présentant une similarité interne maximale et une similarité minimale avec les autres classes. La valeur de la $\texttt{pertinence}$ varie entre 0 et 1. Une valeur élevée indique un plus grand intérêt pour la classe X.

En se basant sur le jeu de données présenté dans la Table 6.1, la Table 6.2 fournit les valeurs de $\texttt{pertinence}$ associées à chaque motif fermé fréquent dobjets.

MFF	**hom.**	**conc.**	**pert.**
CD {1, 6, 7, 9}	$\frac{2\times4}{2\times4+(1+0+1+0)} = 0.8$	$\frac{1}{4} \times (\frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1}) = 0.88$	0.84
BE {3, 4, 5, 8}	$\frac{2\times4}{2\times4+(1+1+1+2)} = 0.62$	$\frac{1}{4} \times (\frac{1}{2} + \frac{1}{3} + \frac{1}{2} + \frac{1}{4}) = 0.4$	0.51
ABE {3, 5, 8}	$\frac{3\times3}{3\times3+(0+0+1)} = 0.9$	$\frac{1}{3} \times (\frac{1}{2} + \frac{1}{2} + \frac{1}{4}) = 0.42$	0.66
EF {2, 4, 8}	$\frac{2\times3}{2\times3+(1+1+2)} = 0.6$	$\frac{1}{3} \times (\frac{1}{1} + \frac{1}{3} + \frac{1}{4}) = 0.53$	0.56
BEF {4, 8}	$\frac{3\times2}{3\times2+(0+1)} = 0.86$	$\frac{1}{2} \times (\frac{1}{3} + \frac{1}{4}) = 0.29$	0.58

TABLE 6.2 – Pertinences des motifs fermés fréquents dobjets (MFF).

Algorithme de sélection des classes

Nous utilisons la mesure de $\texttt{pertinence}$ pour associer des classes aux motifs fermés fréquents dobjets. Lalgorithme repose sur le paramètre $\texttt{min\_img}$, qui correspond au nombre minimum dimages non classées quune nouvelle classe doit couvrir. Comme recommandé par Durand and Crémilleux (2003), la valeur de $\texttt{min\_img}$ est proche de $\texttt{minfr}$. Lalgorithme de sélection des classes, basé sur la mesure de $\texttt{pertinence}$ de chaque classe potentielle, est le suivant : Premièrement, on commence par sélectionner la classe potentielle avec la valeur de $\texttt{pertinence}$ la plus élevée. Ensuite, tant quil reste des images à classer et des possibles motifs dobjets disponibles, on sélectionne le motif qui a la valeur de $\texttt{pertinence}$ la plus élevée et dont $g(X)$ contient au moins $\texttt{min\_img}$ images non encore classées. La classe associée à ce motif est alors lensemble des images non encore associée à une classe et satisfaisant ce motif.

En considérant $\texttt{min\_img} = 2$ pour le jeu de données présenté dans la Table 6.2, les classes potentielles sont construites à partir de CD, BE, ABE, EF et BEF. Avec une valeur de $\texttt{pertinence}$ de $0,84$, la première classe sélectionnée est CD, qui contient les images 1, 6, 7 et 9. Ensuite, toujours sur la base des valeurs de $\texttt{pertinence}$ pré-calculées, la classe ABE, ayant une valeur de $\texttt{pertinence}$ de $0,66$, est choisie, incluant les images 3, 5 et 8.

Par la suite, les images 3, 5 et 8 sont retirées des classes potentielles BE, EF et BEF. La classe EF est alors sélectionnée, contenant les images 2 et 4. Le motif fermé fréquent BEF est exclu, car il ne satisfait pas la condition de minimum dimages (min_img) après le retrait de limage 8, déjà incluse dans la classe ABE. De même, BE ne peut pas être sélectionné comme classe pour la même raison.

Comme le montre lexemple, certains motifs peuvent couvrir les mêmes images, ce qui conduit à des classes qui se chevauchent. Cela explique pourquoi, étant donné un motif fermé X, lalgorithme ECCLAT supprime les images de $g(X)$ qui ont déjà été attribuées à dautres classes. Cela explique également pourquoi les règles $X \rightarrow Classe$ ne sont pas satisfaites sur lensemble de données dentraînement. Dans la section 6.3.1, nous proposons un algorithme pour rechercher des règles $B_1 \wedge B_2 \wedge \ldots \wedge B_j \rightarrow Classe$ qui sont satisfaites sur lensemble de données dentraînement.

Détails sur la création des jeux de données

La Table 6.3 montre les paramètres utilisés pour créer les différents ensembles de données COCO@x. Pour lensemble de données COCO@11 , nous avons utilisé $minfr = 200$ et $minobj = 5$. Nous avons obtenu 133 motifs fermés fréquents dobjets représentant des classes potentielles. En utilisant lalgorithme de sélection de classes avec $min\_img = 200$, nous avons obtenu 11 classes et 24 objets qui les décrivent (qui apparaissent dans au moins un motif dune classe). Pour lensemble de données COCO@24 , nous avons utilisé les paramètres $minfr = 250$ et $minobj = 4$. Nous avons obtenu 280 motifs fermés fréquents dobjets. Avec $min\_img = 220$, nous avons obtenu 24 classes décrites par 24 objets distincts. Pour lensemble de données COCO@48 , nous avons utilisé les paramètres $minfr = 300$ et $minobj = 3$. Nous avons obtenu 443 motifs fermés fréquents dobjets. Avec $min\_img = 300$, nous avons obtenu 48 classes décrites par 48 objets distincts.

Dataset	$minfr$	$minobj$	FCP	min_img	Classes	Objets
COCO@11	200	5	133	200	11	24
COCO@24	250	4	280	230	24	24
COCO@48	300	3	443	300	48	48

TABLE 6.3 – Détails sur la création des jeux de données COCO@x.

Nous avons généré des ensembles dentraînement et de test distincts pour chaque ensemble de données. Les échantillons dentraînement pour les nouveaux ensembles de données ont été dérivés des images dentraînement de lensemble de données COCO 2014. Les images de validation de COCO 2014 ont été utilisées pour constituer lensemble de test. Lensemble de test de COCO 2014 na pas été utilisé en raison de lindisponibilité des informations concernant la présence dobjets dans les images.

6.3 Enrichissement avec des connaissances

Lobjectif de lenrichissement des jeux de données avec des connaissances est de générer des règles propositionnelles qui capturent les relations dinter-dépendances entre les attributs décrivant les images. Il est alors important de préciser que cet enrichissement ne peut sappliquer quaux jeux de données contenant des informations supplémentaires sous forme dattributs, tels que les jeux de données COCO@x précédemment créés, où les attributs correspondent à la présence dobjets dans les images, ou dautres jeux de données souvent utilisés pour la classification basée sur les concepts.

Dans la suite de ce chapitre, nous utiliserons le terme concept pour désigner de manière générale une information additionnelle contenue dans les images des jeux de données. Un concept peut ainsi représenter un attribut, un objet, une couleur, une forme, ou toute autre caractéristique pertinente permettant de décrire le contenu visuel dune image.

Soit $\mathcal{D} = \{(NameCl_i, Cl_i, \mathcal{C}_i)\}_{i=1}^{k}$ un jeu de données contenant k classes, où $NameCl_i$ représente le nom de la i-ième classe, Cl_i représente lensemble des images de la i-ième classe, et $\mathcal{C}_i$ représente un ensemble de concepts décrivant la i-ième classe. Notons que $\forall i, Cl_i \neq \emptyset$, $\forall i, j, i \neq j, Cl_i \cap Cl_j = \emptyset$, et nous notons $P = \bigcup_{i=1}^{k} \mathcal{C}_i$ lensemble des concepts décrivant le jeu de données. Dans la suite, pour des raisons de simplicité, lorsque nous écrivons des règles sur les images, Cl_i représente la proposition booléenne exprimant quune image appartient à la classe Cl_i.

Définition 1 : Le support dune règle $A \rightarrow B$ dans un ensemble de données D est le nombre dimages de D qui satisfont à la fois A et B.

Définition 2 : La confiance dune règle $A \rightarrow B$ dans un ensemble de données D est le rapport entre le nombre dimages de D qui satisfont B et le nombre dimages qui satisfont A.

Pour exploiter linter-dépendance entre les concepts, nous générons deux types de règles : des règles de classe et des règles de concepts.

Règles de classe : Les règles de classe permettent dexprimer les relations entre une classe donnée et les concepts qui la définissent. On peut alors distinguer deux types de règles de classe :

- Les règles de la forme $Cl \rightarrow A_1 \wedge A_2 \wedge \ldots \wedge A_i$ avec $A_i \in P$. Notons que pour les bases de données créées avec la méthode présentée en Section 6.2, les règles de cette forme générées ont une confiance égale à 1 dans lensemble dentraînement. Cependant, cette propriété nest pas vérifiée sur lensemble de test, car certaines images de lensemble de test appartenant à la classe Cl peuvent ne pas satisfaire la conclusion de la règle.

- Les règles de la forme $B_1 \wedge B_2 \wedge \ldots \wedge B_j \rightarrow Cl$, avec $B_j \in P$. Notons en revanche que pour les bases de données créées avec la méthode présentée en Section 6.2,

les règles de cette forme générées nont pas toujours une confiance égale à 1 dans lensemble dentraînement.

Règles de concepts : Les règles de concepts capturent les interdépendances entre différents concepts visuels présents dans les images. Elles sont de la forme

$$A_1 \wedge A_2 \wedge \ldots \wedge A_i \rightarrow B_1 \wedge B_2 \wedge \ldots \wedge B_j$$

avec $A_i \in P$, $B_j \in P$.

6.3.1 Création des règles de classe

Pour chaque classe Cl_i, rappelons que $\mathcal{C}_i$ est lensemble des concept partagés par toutes les images de la classe i dans lensemble dapprentissage. Pour les classes étant construites en utilisant la méthode présentée dans la Section 6.2, la règle définie par :

$$Cl_i \longrightarrow \bigwedge_{j=1}^{|\mathcal{C}_i|} \mathcal{C}_{i_j} \tag{6.4}$$

a une confiance égale à 1.

Notons que la règle $\bigwedge_{j=1}^{|\mathcal{C}_i|} \mathcal{C}_{i_j} \longrightarrow Cl_i$ na pas toujours une confiance égale à 1, car certaines images dune classe peuvent correspondre à la description dautres classes. Par exemple, dans lexemple de la Section 6.2, limage 8 est étiquetée par la classe ABE dans la Table 6.2, mais elle satisfait également les propriétés de la classe EF. Dans un tel scénario, il nest pas possible de créer des règles déquivalence entre une classe et la conjonction de concepts partagés par toutes les images de cette classe. Par conséquent, nous proposons un algorithme pour construire des règles pour chaque classe, qui prendront la forme suivante :

$$\bigwedge_{j=1}^{|B|} B_j \longrightarrow Cl_i, \quad B \subseteq P \tag{6.5}$$

où P est lensemble des concepts apparaissant dans lensemble de données, avec un support supérieur à 1 pour la classe et une confiance égale à 1. De plus, nous cherchons les règles les plus générales (celles ayant le moins de littéraux dans la partie gauche de la règle) satisfaisant ces conditions. Rappelons que chaque élément B_j dans la partie gauche de la règle correspond à un concept et signifie la présence de ce concept dans limage, et quun motif est une conjonction de concepts. Pour créer ces règles, nous associons à chaque motif deux mesures : le nombre pos $(+)$ dexemples positifs (images de la classe Cl) qui satisfont le motif, et le nombre neg $(-)$ dexemples négatifs (images dune classe différente de Cl) qui satisfont le motif. Nous construisons un arbre dénumération des ensembles, en partant des motifs de taille 1 et en élaguant larbre de recherche selon deux conditions : lorsque $pos = 0$ et lorsque $neg = 0$.

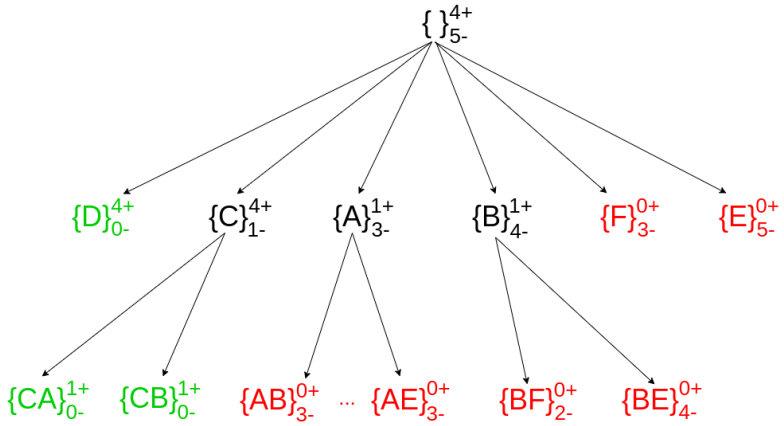


FIGURE 6.2 – Arbre dénumération de génération des règles pour la classe CD à partie de lexemple de la Section 6.2.

Les concepts sont ordonnés en fonction du nombre dexemples positifs quils couvrent dans la classe. Étant donné un motif P, $pos = 0$ signifie quaucun exemple positif dans la classe nest satisfait par ce motif. Par conséquent, il nest pas possible de construire une règle satisfaisant nos conditions avec ce motif, ni avec les motifs qui le contiennent. Ainsi, larbre de recherche peut être élagué à ce nœud. Dun autre côté, $neg = 0$ et $pos \neq 0$ signifient que la règle $P \rightarrow Cl$ satisfait les deux conditions. De plus, comme tous les motifs P qui sont plus spécifiques que P ($P \subseteq P$) conduiraient à des règles plus spécifiques, larbre de recherche peut également être élagué à ce nœud. Dans le cas où $pos \neq 0$ et $neg \neq 0$, la recherche continue. Limage 6.2 illustre larbre dénumération pour la génération de règles pour la classe CD pour lexemple de la Section 6.2. Les motifs verts représentent ceux qui aboutissent à une règle, tandis que les motifs rouges désignent ceux qui ne conduisent pas à la création dune règle.

Lalgorithme 1 décrit la génération de telles règles. Nous avons n images $I = [I_1, \ldots, I_n]$, p concepts $O = [O_1, \ldots, O_p]$, et k classes $C = [Cl_1, \ldots, Cl_k]$. C forme une partition de lensemble des images (pour tout i, $Cl_i \neq \emptyset$, $C = \bigcup_l Cl_i$, et pour tout i, j avec $i \neq j$, $Cl_i \cap Cl_j = \emptyset$). Chaque image inclut un ensemble de concepts, et la relation entre les images et les concepts est décrite à laide de la matrice $MaskO$ ($MaskO = \{v_{ij}\}^{p \times n}$, où $v_{ij} = \mathbf{1}[O_i \in I_j]$). $Maskc$ et Lt sont dabord générés (lignes 4 et 5). $Maskc$ ($\in \{0,1\}^n$) est un vecteur binaire indiquant les images appartenant à une classe c (1 si limage appartient à la classe, 0 sinon) (ligne 4). Lt est une liste initialisée avec des motifs de taille 1 et mise à jour à chaque itération (ligne 17) pour parcourir larbre dénumération (Figure 6.2).

Lorsque le nombre dexemples positifs est 0 (ligne 10), le motif actif ne couvre aucun exemple positif et ne peut pas être utilisé pour construire une règle pour cette classe. Ce motif est alors supprimé de la liste Lt. Le nombre dexemples positifs est calculé en effectuant le produit de Hadamard (multiplication élément par élément $\odot$) entre le masque de chaque concept dans le motif actif et le masque représentant la classe active. Rappelons quun motif est une liste de concepts, et que $MaskO[p[i]]$ ($\in \{0,1\}^n$) est un vecteur binaire indiquant si

une image contient le i-ème concept du motif p. Le nombre de 1 dans le vecteur résultant donne le nombre dexemples positifs. Si le nombre dexemples négatifs est 0 (ligne 12), cela signifie que le motif actif est satisfait uniquement par les images de cette classe. Une nouvelle règle peut alors être générée (ligne 13). Le nombre dexemples négatifs est calculé de manière similaire au nombre dexemples positifs, mais en considérant le masque inverse (ligne 9) de la classe active. Lalgorithme sarrête lorsquil ny a plus déléments à inspecter ($Lt = \emptyset$).

Algorithm 1 Rules generation algorithm

1: Inputs : I=[$I_1 ... I_n$], O=[$O_1 ... O_p$], MaskO, Cl = [$Cl_1 ... Cl_k$].
2: Outputs=[$Rules^1 ... Rules^k$] { k classes}
3: **for all** (Class c) **do**
4: $Rule^c = \varnothing$, $Mask^c = [b_i]^n$ $\{b_i = \mathbf{1}[I_i \in Cl_c]\}$
5: $L_t = \{\{O_1\}, \ldots \{O_p\}\}$
6: **while** ($L_t \neq \varnothing$) **do**
7: **for all** ($pattern \in L_t$) **do**
8: $pos_{pattern} = \sum_{j}^{n} \mathbf{1}[(V^+[j] == 1)] \mid V^+ = Mask^c \odot \prod_{(\odot)i}^{|pattern|} MaskO[pattern[i]]$
9: $neg_{pattern} = \sum_{j}^{n} \mathbf{1}[(V^-[j] == 1)] \mid V^- = \bar{Mask^c} \odot \prod_{(\odot)i}^{|pattern|} MaskO[pattern[i]]$
10: **if** ($pos_{pattern}$ == 0) **then**
11: $L_t = L_t \setminus \{pattern\}$
12: **else if** ($neg_{pattern} == 0$) **then**
13: $Rule^c = Rule^c \cup \{pattern\}$ {nouvelle règle avec $pattern$}
14: $L_t = L_t \setminus \{pattern\}$
15: **end if**
16: **end for**
17: Update L_t
18: **end while**
19: **end for**
20: return $\bigcup_k Rule^k$

En utilisant lalgorithme 1, les règles de classe dérivées de lensemble de données généré à partir de la Table 6.2 sont les suivantes :

$$
\begin{aligned}
Cl(CD) &\longrightarrow (C \wedge D)\\
Cl(ABE) &\longrightarrow (A \wedge B \wedge E)\\
Cl(EF) &\longrightarrow (E \wedge F)\\
D &\longrightarrow Cl(CD)\\
(A \wedge C) &\longrightarrow Cl(CD)\\
(B \wedge C) &\longrightarrow Cl(CD)\\
(A \wedge B) &\longrightarrow Cl(ABE)\\
(A \wedge E) &\longrightarrow Cl(ABE)\\
(A \wedge F) &\longrightarrow Cl(ABE)\\
(C \wedge E) &\longrightarrow Cl(EF)\\
(C \wedge F) &\longrightarrow Cl(EF)
\end{aligned}
$$

6.3.2 Création des règles de concepts

La création des règles de concepts consiste à créer des règles dassociation de concepts fermées sous la forme suivante :

$$A_1 \wedge A_2 \wedge \ldots \wedge A_i \rightarrow B_1 \wedge B_2 \wedge \ldots \wedge B_j$$

avec $A_i \in P$, $B_j \in P$.

Une règle dassociation fermée est une règle dassociation de la forme $A \longrightarrow B$ telle que lunion de A et B constitue un ensemble de concepts fermés. Nous avons utilisé lalgorithme proposé par (Szathmary 2006b), qui retourne toutes les règles dassociation fermées dont le support et la confiance sont respectivement supérieurs ou égaux aux seuils $minsup$ et $minconf$.

En utilisant lalgorithme proposé par (Szathmary 2006b), les règles de concepts dérivées de lensemble de données généré à partir de la Table 6.2 en utilisant les paramètres $minsup = 10\%$ et $minconf = 50\%$ sont les suivantes :

$$
\begin{aligned}
E &\longrightarrow B && (sup : 4,\ conf : 0.8)\\
B &\longrightarrow E && (sup : 4,\ conf : 0.8)\\
D &\longrightarrow C && (sup : 4,\ conf : 1.0)\\
C &\longrightarrow D && (sup : 4,\ conf : 0.8)\\
B \wedge E &\longrightarrow A && (sup : 3,\ conf : 0.75)\\
A \wedge E &\longrightarrow B && (sup : 3,\ conf : 1.0)\\
A \wedge B &\longrightarrow E && (sup : 3,\ conf : 1.0)\\
E &\longrightarrow A \wedge B && (sup : 3,\ conf : 0.6)\\
B &\longrightarrow A \wedge E && (sup : 3,\ conf : 0.6)\\
A &\longrightarrow B \wedge E && (sup : 3,\ conf : 0.75)\\
A \wedge D &\longrightarrow C && (sup : 1,\ conf : 1.0)\\
A \wedge C &\longrightarrow D && (sup : 1,\ conf : 1.0)\\
B \wedge D &\longrightarrow C && (sup : 1,\ conf : 1.0)\\
B \wedge C &\longrightarrow D && (sup : 1,\ conf : 1.0)
\end{aligned}
$$

6.4 Description des jeux de données

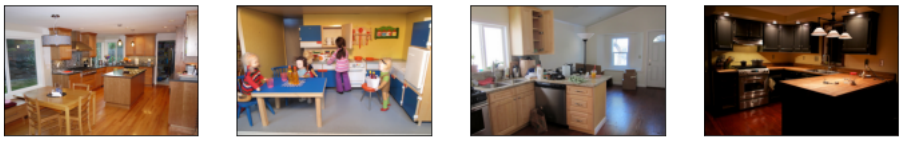
En appliquant la méthode décrite dans la Section 6.2, nous avons créé sept jeux de données en étendant des bases dimages existantes. Ces jeux de données sont COCO@11, COCO@24, COCO@48, SunIn, SunOut, SunInOut et Synthetic. Par ailleurs, nous introduisons également dautres jeux de données dont CelebA sur lequel nous créons des classes à partir dune méthode introduite par Espinosa Zarlenga et al. (2022), et CUB déjà étiqueté et utilisé pour créer CUB-Simp à partir de la méthode proposée par Koh et al. (2020). Pour chacun de ces jeux de données, des règles de classe et des règles de concepts ont été générées exclusivement à partir de lensemble dentraînement.

6.4.1 COCO@11

Le jeu de données COCO@11 contient 11 classes différentes et 24 objets distincts. Il comprend un total de 2 463 images, dont 1 692 pour lensemble dentraînement et 771 pour lensemble de test. Chaque classe est composée dau moins 200 images, avec au moins 5 objets par classe. Le jeu de données comprend 487 règles de classe et 650 règles de concepts générées en utilisant les méthodes présentées dans la Section 6.3 avec $\mathtt{minsup} = 5\%$ et $\mathtt{minconf} = 95\%$ pour les règles de concepts. Parmi celles-ci, seulement 15 règles de classe et 216 règles de concepts ont une confiance égale à 1 sur lensemble dapprentissage. La Table 6.4 présente la répartition des images entre les ensembles dentraînement et de test, décrit les objets définissant chaque classe et indique le nombre total de règles de classe générées pour chaque classe spécifique.

(a) Bus, Car, Person, Traffic light, Truck

(b) Cup, Dining table, Fork, Knife, Sandwich



(c) Cake, Chair, Dining table, Knife, Person

FIGURE 6.3 – Exemple dimages sélectionnées aléatoirement illustrant trois classes dans le jeu de données COCO@11.

Classes	Images		Liste des objets par classe	Règles de classe
	Train	Test		
Cl_1	141	59	bus car person traffic-light truck	4
Cl_2	139	61	book chair keyboard mouse tv	27
Cl_3	134	77	bottle bowl cup oven sink	53
Cl_4	160	60	cake chair dining-table knife person	93
Cl_5	143	59	cell-phone chair cup dining-table person	130
Cl_6	149	71	cup dining-table fork knife sandwich	53
Cl_7	180	85	chair cup dining-table person pizza	31
Cl_8	151	61	dining-table fork knife person pizza	29
Cl_9	173	86	bottle chair dining-table person wine-glass	43
Cl_{10}	166	74	bottle chair cup dining-table person	9
Cl_{11}	156	78	bowl cup dining-table fork spoon	15
Total	1 692	771	24	487

TABLE 6.4 – Description du jeu de données COCO@11.

La Figure 6.3 illustre plusieurs exemples dimages représentant trois classes sélectionnées aléatoirement parmi les 11 classes du jeu de données COCO@11 . On peut observer quil est intuitivement plus facile de distinguer la sémantique de la classe représentée par la Figure 6.3a par rapport aux deux autres classes.

6.4.2 COCO@24

Le jeu de données COCO@24 se compose de 24 classes et dun total de 8 601 images. Parmi ces images, 5 879 sont destinées à lensemble dentraînement et 2 722 à lensemble de test, avec une répartition dau moins 230 images par classe et au maximum 682 images. Ce jeu de données comprend 4 239 règles de classe et 36 règles de concepts générées en utilisant les méthodes présentées dans la Section 6.3. Les règles de concepts on été générées

en utilisant les hyper-paramètres $minsup = 5\%$ et $minconf = 95\%$. Parmi les règles générées, seulement 15 règles de classe et 216 règles de concepts ont une confiance égale à 1 sur lensemble dapprentissage. Chaque classe est caractérisée par au moins 4 objets, et lensemble de données contient 36 objets distincts. La Table 6.5 illustre la répartition des images dans les ensembles dentraînement et de test, décrit les objets définissant chaque classe et présente le nombre de règles de classe générées pour chaque classe respective.

Classes	Images		Liste des objets par classe	Règles
	Train	Test		
Cl_1	241	150	chair, person, sports-ball, tennis-racket	41
Cl_2	470	212	baseball-bat, baseball-glove, person, sports-ball	30
Cl_3	225	121	backpack, handbag, person, suitcase	90
Cl_4	176	79	backpack, handbag, person, umbrella	90
Cl_5	280	142	car, motorcycle, person, truck	46
Cl_6	373	168	car, person, traffic light, truck	37
Cl_7	163	90	bus, car, person, truck	19
Cl_8	285	116	bus, car, person, traffic-light	41
Cl_9	156	82	chair, dining-table, person, umbrella	140
Cl_{10}	307	94	chair, couch, person, remote	392
Cl_{11}	222	92	microwave, oven, refrigerator, sink	504
Cl_{12}	235	109	book, chair, couch, tv	605
Cl_{13}	163	81	chair, couch, potted-plant, vase	294
Cl_{14}	221	87	cake, dining-table, fork, person	170
Cl_{15}	244	101	cake, chair, dining-table, person	249
Cl_{16}	216	118	cup, dining-table, person, sandwich	242
Cl_{17}	215	91	cell-phone, cup, dining-table, person	269
Cl_{18}	304	153	chair, dining-table, person, pizza	111
Cl_{19}	143	90	bottle, cup, oven, sink	243
Cl_{20}	194	89	dining-table, fork, knife, pizza	32
Cl_{21}	161	69	chair, dining-table, person, potted-plant	264
Cl_{22}	376	178	bowl, cup, dining-table, spoon	204
Cl_{23}	315	142	chair, cup, dining-table, person	91
Cl_{24}	194	68	cup, dining-table, fork, knife	35
Total	5 879	2 722	24	4 239

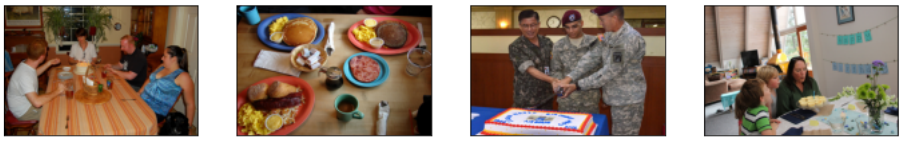
TABLE 6.5 – Description du jeu de données COCO@24

La Figure 6.4 présente plusieurs exemples dimages représentant trois classes parmi les 24 classes totales du jeu de données COCO@24. Comme pour le jeu de données précédent, certaines classes sont sémantiquement plus proches que dautres. Par exemple, la classe représentée par la Figure 6.4a se distingue clairement des autres classes. Bien que les classes des Figures 6.4b et 6.4c soient visuellement proches, les objets représentés dans ces classes permettent de les différencier sur le plan sémantique.

6.4.3 COCO@48

Lensemble de données COCO@48 comprend 48 classes et un total de 25 468 images. Lensemble dentraînement contient 17 278 images, tandis que lensemble de test en compte 8 190. Chaque classe comporte un minimum de 300 images et un maximum de 1 314 images (ensemble dentraînement et de test confondus). De plus, chaque classe est décrite par au moins 3 objets, et lensemble de données contient un total de 59 objets distincts. Ce

(a) Bus, Car, Person, Truck



(b) Cake, Dining-table, Fork, Person

(c) Bottle, Cup, Oven, Sink

FIGURE 6.4 – Exemple dimages sélectionnées aléatoirement illustrant trois classes dans lensemble de données COCO@24.

jeu de données comprend 20 158 règles de classe et 17 règles de concepts générées en utilisant les méthodes présentées dans la Section 6.3. Les règles de concepts on été générées en utilisant les hyper-paramètres $\texttt{minsup} = 5\%$ et $\texttt{minconf} = 95\%$. Parmi les règles générées, seulement 58 règles de classe et 1 règle de concepts ont une confiance égale à 1 sur lensemble dapprentissage. La Table 6.6 présente la répartition des images entre les ensembles dentraînement et de test, les objets caractéristiques de chaque classe, ainsi que le nombre de règles de classe générées pour chaque classe spécifique.

Classes	**Images**		**Liste des objets par classe**	**Règles**
	Train	**Test**		
Cl_1	448	226	backpack, person, skis	29
Cl_2	260	109	person, skis, snowboard	10
Cl_3	294	155	bottle, sink, toilet	41
Cl_4	253	115	airplane, person, truck	34
Cl_5	219	120	bench, person, skateboard	34
Cl_6	211	89	apple, banana, orange	710
Cl_7	381	180	car, person, skateboard	99
Cl_8	815	360	baseball glove, person, sports ball	69
Cl_9	228	118	car, horse, person	103
Cl_{10}	250	115	handbag, person, train	104
Cl_{11}	452	267	chair, person, tennis racket	87
Cl_{12}	208	112	bench, person, umbrella	299
Cl_{13}	263	122	car, person, stop sign	187
Cl_{14}	307	135	backpack, person, umbrella	330
Cl_{15}	749	366	car, motorcycle, person	330
Cl_{16}	207	118	car, clock, person	192
Cl_{17}	541	228	cell phone, handbag, person	1 240
Cl_{18}	222	95	car, fire hydrant, person	55
Cl_{19}	244	98	baseball bat, bench, person	49
Cl_{20}	309	152	bench, handbag, person	327
Cl_{21}	286	157	handbag, person, suitcase	268

Cl_{22}	344	183	chair, person, umbrella	288
Cl_{23}	971	449	car, person, truck	281
Cl_{24}	612	277	bus, car, person	29
Cl_25	621	355	keyboard, mouse, tv	655
Cl_26	196	106	car, traffic light, truck	23
Cl_27	259	127	bowl, broccoli, dining table	520
Cl_28	348	182	chair, person, tie	389
Cl_29	245	107	backpack, bicycle, person	116
Cl_30	273	124	car, person, traffic light	34
Cl_31	214	95	bottle, person, remote	1 450
Cl_32	380	182	book, couch, tv	1 363
Cl_33	353	143	cake, dining table, knife, person	788
Cl_{34}	390	147	chair, person, remote	416
Cl_{35}	337	132	microwave, oven, refrigerator	2492
Cl_{36}	396	198	chair, laptop, person	1304
Cl_{37}	187	114	bottle, cell phone, person	373
Cl_{38}	221	103	couch, potted plant, vase	542
Cl_{39}	258	106	cake, dining table, fork	142
Cl_{40}	522	262	cup, dining table, pizza	246
Cl_{41}	453	215	cup, dining table, sandwich	407
Cl_{42}	187	114	book, chair, person	468
Cl_{43}	323	147	chair, dining table, vase	805
Cl_{44}	245	102	bowl, oven, spoon	644
Cl_{45}	248	111	bottle, bowl, sink	688
Cl_{46}	908	406	chair, dining table, person	568
Cl_{47}	353	147	bowl, dining table, spoon	373
Cl_{48}	287	119	dining table, fork, knife	156
Total	17 278	8 190	48	20 158

TABLE 6.6 – Description du jeu de données COCO@48

6.4.4 CUB

Le jeu de données CUB (Caltech-UCSD Birds-200-2011 (Wah et al. 2011)) est un benchmark largement utilisé pour les tâches de classification dans le contexte à grain fin. Il comprend un total de 200 classes doiseaux, avec 5 994 images dentraînement et 5 794 images de test. Chaque classe contient environ 30 images dentraînement, ce qui rend ce jeu de données particulièrement exigeant en raison du faible nombre dexemples par classe. Chaque image comporte des annotations détaillées : 1 étiquette de classe, 15 emplacements de partie doiseau, 312 attributs binaires et 1 cadre de délimitation de loiseau dans limage. Les attributs binaires sont associés à une valeur de certitude allant de 1 à 4, reflétant le niveau de confiance des annotateurs quant à lexactitude quun attribut se trouve bien dans limage. Une certitude de 4 correspond à une haute confiance, tandis quune certitude de 1 correspond à une faible confiance. Les attributs binaires sont organisés de manière à décrire les caractéristiques visuelles des oiseaux sous forme de paires attribut/valeur. Les 28 attributs représentent des propriétés spécifiques des oiseaux, telles que les caractéristiques physiques (forme du bec, couleur des ailes, etc.), les motifs (rayures, points, etc.) ou des informations contextuelles (présence dune couronne sur la tête, etc.). Les valeurs correspondent à létat ou à la valeur discrète associée aux attributs, par exemple, ailes rouges (`has_wing_color::red`) ou bec crochu (`has_bill_shape::hooked`) comme présenté dans la Figure 6.5.

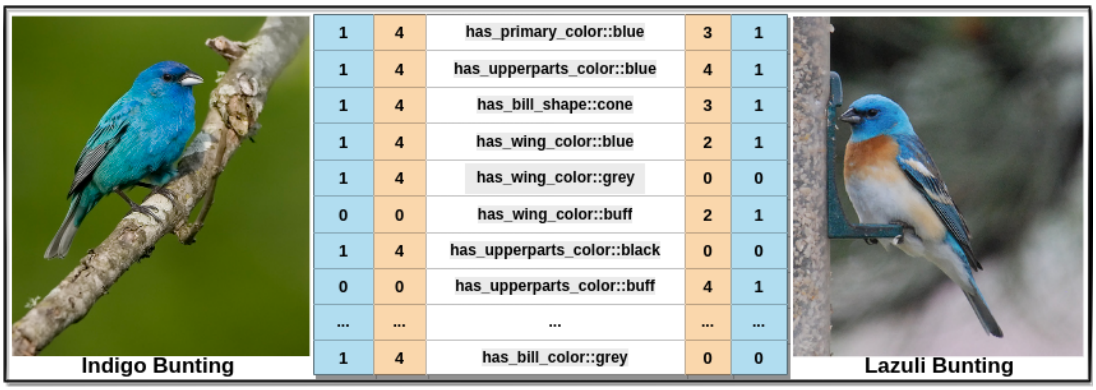


FIGURE 6.5 – Illustration du jeu de données CUB. La première image (à gauche) appartient à la classe *Indigo Bunting* et la deuxième image (à droite) appartient à la classe *Lazuli Bunting*. Les colonnes en bleu correspondent aux valeurs binaires relatives à la présence dun attribut/valeur dans limage (1 = présent, 0 = non présent). Les colonnes en jaune représentent les valeurs de certitude des quatre annotateurs sur la confiance des annotations.

6.4.5 CUB-Simp

Dans le jeu de données CUB, les annotations sont bruitées, car elles ont été fournies par des annotateurs qui ne sont pas des experts en ornithologie. Par exemple, un annotateur peut décrire le ventre dun oiseau comme étant rouge, tandis quun autre peut le décrire comme roux (brun-rougeâtre). Le jeu de données CUB-Simp est une version simplifiée de CUB, proposé par Koh et al. (2020), qui regroupe les annotations au niveau des instances en annotations au niveau des classes grâce à un vote majoritaire. Par exemple, si plus de 50% des corbeaux dans les données sont annotés comme ayant des ailes noires, les auteurs attribuent lannotation ailes noires à tous les corbeaux. Cette approche suppose que tous les oiseaux dune même espèce dans les données dentraînement partagent les mêmes annotations. Bien que cette hypothèse soit généralement valable pour ce jeu de données, il existe des exceptions dues à plusieurs facteurs, tels que locclusion visuelle (certaines parties de loiseau étant cachées dans les images), le dimorphisme sexuel (les mâles et les femelles ayant des apparences différentes), et le dimorphisme lié à lâge (les jeunes oiseaux et les adultes ayant des apparences différentes). Après lapplication du vote majoritaire, les auteurs affinent encore les annotations en filtrant les attributs trop rares. Plus précisément, ils conservent uniquement les attributs binaires qui restent présents après le vote majoritaire dans au moins 10 classes. Ce processus de filtrage permet dobtenir un ensemble final de 112 concepts utilisés dans le jeu de données CUB.

En utilisant les méthodes denrichissement par des connaissances présentées dans la Section 6.3, nous avons généré pour ce jeu de données 199 590 règles de classe et 598 règles de concepts. Les règles de concepts ont été générées en utilisant les hyper-paramètres $\mathtt{minsup} = 20\%$ et $\mathtt{minconf} = 95\%$. Parmi les règles générées, seulement 58 règles de classe et 1 règle de concepts ont une confiance égale à 1 sur lensemble dapprentissage.

6.4.6 CelebA

Le jeu de données CelebA est construit à partir déchantillons du jeu de données *CelebFaces Attributes Dataset* (Liu et al. 2015), en intégrant les annotations de tâches et de concepts spécifiées dans Espinosa Zarlenga et al. (2022). CelebFaces est un jeu de données dimages de visages annotées avec divers attributs, couramment utilisé pour des tâches de reconnaissance faciale et de classification dattributs visuels. Il contient un large ensemble dimages de célébrités, où chaque image est accompagnée de 40 annotations binaires décrivant des caractéristiques faciales spécifiques, telles que la présence de lunettes, la couleur des cheveux ou lexpression du visage.

Comme spécifié dans Espinosa Zarlenga et al. (2022), pour chaque image de CelebA, les 8 concepts binaires les plus fréquents (`Young`, `Smiling`, `Attractive`, `NoBeard`, `Makeup`, `Bangs`, `BlackHair`, `BlondHair`) sont dabord sélectionnés parmi les 40 concepts disponibles. Ensuite, un vecteur dannotations binaires de concepts est construit en utilisant uniquement les 6 premiers concepts les plus fréquents. Les différentes classes pour la tâche de classification sont construites en convertissant en base 10 un nombre binaire dont chaque bit correspond à létat dun attribut visuel spécifique. Cela aboutit à un total de $2^8 = 256$ classes pour le jeu de données. En utilisant les méthodes denrichissement par des connaissances présentées dans la Section 6.3, nous avons généré pour CelebA un total de 183 règles de classe et 55 règles de concepts en utilisant les hyper-paramètres $\texttt{minsup} = 20\%$ et $\texttt{minconf} = 95\%$. Parmi les règles générées, 183 règles de classe et aucune règle de concepts ont une confiance égale à 1 sur lensemble dapprentissage.

6.4.7 SUN Attribute

Le jeu de données SUN Attribute (Patterson et al. 2014) est un jeu de données utilisé pour la reconnaissance de scènes dans un contexte à grain fin. Ce jeu de données compte 707 classes qui représentent des scènes et 14 340 images. Chaque classe contient environ 20 images. Chaque image est décrite par 102 attributs caractérisant le contenu de la scène. Un attribut représente la présence ou labsence dun objet dans la scène et est associé à une valeur de certitude comprise entre 0 et 1, fournie par lannotateur. Les différentes classes sont regroupées en super-classes parmi lesquelles la super-classe représentant les scènes en intérieur (chambre, salon, etc.) et les scènes en extérieur (rue, forêt, etc.).

Dans le but de simplifier le jeu de données en réduisant le nombre de classes, et aussi de générer davantage de jeux de données, nous avons étendu trois jeux de données qui dérivent de SUN Attribute : SunIn, SunOut et SunInOut. SunIn est un jeu de données qui regroupe toutes les scènes en intérieur (église, usine, salon, garage, etc.) et contient un total de 78 classes et 68 concepts. Nous avons sélectionné uniquement les concepts apparaissant au moins 50 fois dans les images, et nous avons conservé un total de 68 concepts pour ce jeu de données. SunOut est un jeu de données qui regroupe toutes les scènes en extérieur et compte un total de 93 classes et 88 concepts que nous avons filtrés à 88 concepts en sélectionnant uniquement les concepts apparaissant au moins 50 fois dans les images. Enfin, SunInOut est un jeu de données qui inclut les scènes mixtes (intérieures et extérieures), totalisant 171 classes et 96 concepts en sélectionnant uniquement les concepts apparaissant au moins 50 fois dans les images.

Chacun des jeux de données a été enrichi en utilisant les méthodes présentées dans la Section 6.3. Pour rendre binaires les annotations sur les concepts, nous avons gardé à 1 tous les concepts dont la certitude est supérieure à 0 pour privilégier les présomptions de présence. Nous avons ainsi généré pour SunIn un total de 145 716 règles de classe parmi lesquelles 70 ont une confiance égale à 1 sur lensemble dapprentissage. En utilisant les paramètres $minsup = 20\%$ et $minconf = 95\%$, nous avons généré 125 règles de concepts parmi lesquels aucune na une confiance égale à 1 sur lensemble dapprentissage. Pour le jeu de données SunOut, nous avons généré un total de 796 263 règles de classe et 125 règles de concepts en utilisant les paramètres $minsup = 20\%$ et $minconf = 95\%$. De toutes les règles générées, 86 règles de classe et aucune règles de concepts ont une confiance égale à 1 sur lensemble dapprentissage. Pour le jeu de données SunInOut, nous avons généré un total de 796 263 règles de classe et 47 règles de concepts en utilisant les paramètres $minsup = 15\%$ et $minconf = 95\%$. De toutes les règles générées, 153 règles de classe et aucune règles de concepts ont une confiance égale à 1 sur lensemble dapprentissage.

6.4.8 Visual Genome

Le jeu de données Visual Genome (Krishna et al. 2017) est un jeu de données publique qui contient environ 108 000 images annotées de manière dense avec des objets, des attributs et des relations. Contrairement aux jeux de données de classification classiques, les images de Visual Genome ne sont pas regroupées en classes, ce qui signifie quil ne correspond pas initialement à un problème de classification. Pour ladapter à un problème de classification, les objets présents dans les images ont été utilisés comme critère de catégorisation. Deux nouveaux jeux de données ont ainsi été étendus, nommés VG12 et VG48, où les classes sont définies par les objets quelles contiennent. Pour créer ces jeux de données, les 500 objets les plus fréquents apparaissant dans les images ont été sélectionnés, puis les 20 objets les plus courants ont été retirés. Ensuite, la méthode présentée dans la section 6.2 a été utilisée pour générer les classes. Les jeux de données obtenus contiennent un total de 17 objets et 12 classes pour VG12, et 480 objets et 48 classes pour le jeu de données VG48. VG12 contient environ 150 images par classe, avec 5 objets décrivant chaque image. VG48 contient en moyenne 225 images par classe, avec 4 objets décrivant chaque image. Les jeux de données sont répartis en deux sous-ensemble : un sous-ensemble pour lentraînement et un sous-ensemble de test. La Figure 6.6 présente 4 exemples de classes du jeu de données VG12.

En utilisant les méthodes denrichissement par des connaissances présentées dans la Section 6.3, nous avons généré pour VG12 un total de 393 règles de classe et 221 règles de concepts en utilisant les paramètres $minsup = 30\%$ et $minconf = 95\%$. Parmi les règles générées, 25 règles de classe et 170 règles de concepts ont une confiance égale à 1 sur lensemble dapprentissage. Pour le jeu de données VG48, nous avons généré un total de 950 987 règles de classe et 101 règles de concepts en utilisant les paramètres $minsup = 3\%$ et $minconf = 95\%$. Parmi les règles générées, 50 règles de classe et 7 règles de concepts ont une confiance égale à 1.

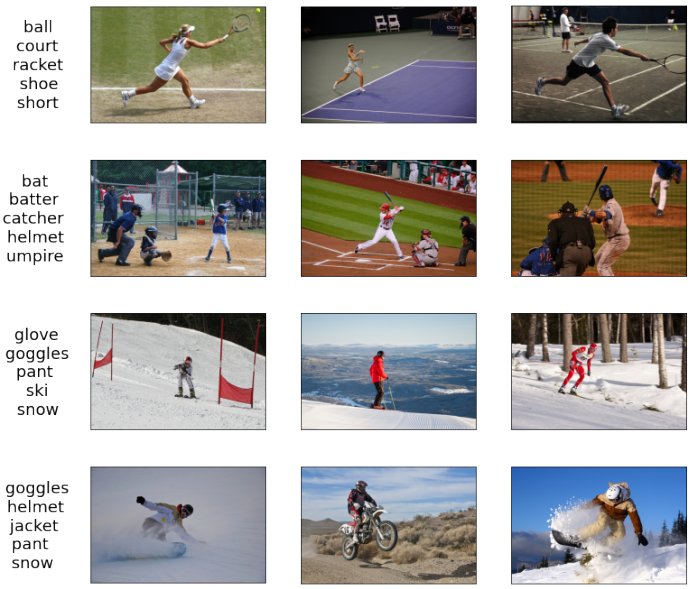


FIGURE 6.6 – Exemples dimages pour 4 classes du jeu de données VG12, où chaque ligne correspond à une classe spécifique. La première ligne représente une classe caractérisée par la présence des objets *ballon*, *terrain*, *raquette chaussure* et *short*. La deuxième ligne illustre une classe comprenant les objets *batte*, *batteur*, *receveur*, *casque* et *arbitre*. La troisième ligne est associée à une classe contenant *gant*, *lunettes de ski*, *pantalon*, *ski* et *neige*. Enfin, la dernière ligne représente une classe avec les objets *lunettes de ski*, *casque*, *veste*, *pantalon* et *neige*.

6.4.9 Synthetic

Nous avons créé le jeu de données Synthetic afin de fournir une base dentraînement simple et intuitive pour évaluer nos différentes méthodes. Ce jeu de données est généré artificiellement en utilisant quatre formes géométriques (cercle (**C**), rectangle (**R**), triangle (**T**), hexagone (**H**)) et quatre couleurs (rouge (**r**), rose (**p**), vert (**g**), jaune (**y**)). Ces formes et couleurs ont été combinées pour créer 16 concepts distincts, chaque concept représentant une association spécifique entre une forme et une couleur (**Cr**, **Cp**, **Cg**, **Cy**, **Rr**, **Rp**, **Rg**, **Ry**, **Tr**, **Tp**, **Tg**, **Ty**, **Hr**, **Hp**, **Hg**, **Hy**).

Notre objectif est de générer des images qui respectent deux contraintes spécifiques : chaque image doit contenir un minimum de 4 et un maximum de 5 concepts parmi les 16 concepts possibles, et chaque image doit inclure au moins une instance de chacune des formes géométriques (cercle, rectangle, triangle, hexagone).

Afin de limiter le nombre dimages possibles à générer, nous avons créé aléatoirement 10 règles définissant les combinaisons de concepts.

$$\begin{aligned}
\textbf{Ty} &\rightarrow \textbf{Cg} \\
\textbf{Cp} &\rightarrow \textbf{Hy} \\
\textbf{Hg} &\rightarrow \textbf{Tr} \\
\textbf{Cr} &\rightarrow \textbf{Rg} \\
\textbf{Rp} \wedge \textbf{Tp} &\rightarrow \textbf{Cg} \\
\textbf{Cy} \wedge \textbf{Hp} &\rightarrow \textbf{Tg} \\
\textbf{Tp} \wedge \textbf{Hy} &\rightarrow \textbf{Rp} \\
\textbf{Rg} \wedge \textbf{Cr} &\rightarrow \textbf{Hp} \\
\textbf{Hr} \wedge \textbf{Ry} &\rightarrow \textbf{Tp} \wedge \textbf{Cg} \\
\textbf{Cy} &\rightarrow \textbf{Hy} \wedge \textbf{Rp}
\end{aligned}$$

Nous avons ensuite généré des les combinaisons de concepts possibles respectant ces 10 règles ainsi que les deux contraintes mentionnées ci-dessus, aboutissant à un total de 599 images possibles. Les concepts ont été créés avec des tailles aléatoires dans les images, chaque image ayant une dimension de 300×300 pixels.

Pour créer les différentes classes, nous avons utilisé la méthode présentée dans la Section 6.3. Ce processus a abouti à un jeu de données contenant 8 classes et 488 images, comme décrit dans la Table 6.7. La Figure 6.7 illustre quatre images sélectionnées aléatoirement parmi les trois premières classes du jeu de données.

Classes	Listes des concepts	Nombre total dimages
Cl_1	**Cg**, **Hg**, **Tr**	62
Cl_2	**Cg**, **Hr**, **Tp**	50
Cl_3	**Cg**, **Hy**, **Tg**	58
Cl_4	**Cp**, **Hg**, **Tr**	56
Cl_5	**Cg**, **Hp**, **Ty**	56
Cl_6	**Cg**, **Ty**	63
Cl_7	**Hy**, **Rp**	65
Cl_8	**Hp**, **Rg**	78

TABLE 6.7 – Classes du jeu de données Synthetic avec les différents concepts décrivant chaque classe.

En utilisant les méthodes denrichissement par des connaissances présentées dans la Section 6.3, nous avons généré pour le jeu de données Synthetic un total de 157 règles de classe et 33 règles de concepts en utilisant les paramètres $\mathtt{minsup} = 5\%$ et $\mathtt{minconf} = 95\%$. Parmi les règles générées, 13 règles de classe et 32 règles de concepts ont une confiance égale à 1.

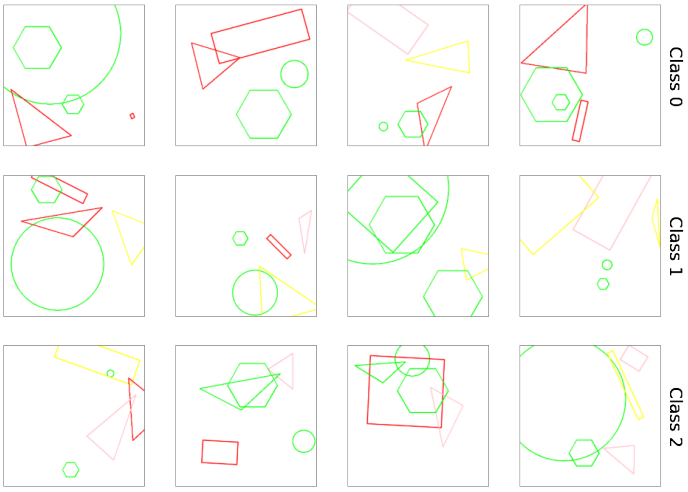


FIGURE 6.7 – Quelques exemples dimages sélectionnées aléatoirement illustrant les trois premières classes du jeu de données Synthetic.

6.5 Conclusion

Dans ce chapitre, nous avons présenté une méthodologie générique adoptée pour la création et lenrichissement de bases dimages dédiées à lapprentissage de modèles pour la classification. Nous avons dabord motivé limportance de disposer de bases de données adaptées, intégrant des annotations de concepts exploitables pour lapprentissage et lexplicabilité des modèles. Ensuite, nous avons décrit les différentes étapes de construction de ces bases dimages, en nous appuyant sur des approches danalyse formelle de concepts et lextraction de motifs fermés fréquents pour sélectionner des classes pertinentes. Les jeux de données étendus sur la base de cette méthode, COCO@11, COCO@24 et COCO@48, contiennent des images annotées, chacune appartenant à une classe, et des concepts étiquettes, permettant de les exploiter dans le contexte de modèle basés sur les concepts. Lenrichissement des bases dimages consiste aussi à exploiter les dépendances entre les concepts et les classes, ce qui nous a motiver à proposer la création des règles de classe et des règles de concepts. La création des règles de classe utilise également un algorithme génériques que nous proposons, qui peut sappliquer sur dautres jeux de données ayant des concepts à disposition.

Enfin, nous avons détaillé les différents jeux de données construits et enrichis dans le cadre de cette thèse. Ces jeux de données, pour la plus part étendus à partir de bases publiques existantes, ont été enrichi avec des règles de classe et des règles de concepts. Ces jeux de données constituent une base solide pour lévaluation des méthodes de classification dimages utilisant des connaissances a priori et ouvrent la voie à de nouvelles expérimentations visant à améliorer la compréhension et lexplicabilité des modèles dintelligence artificielle.

Chapitre 7

Intégration de graphe de connaissances pour la classification d'images

Ce chapitre présente une méthode de classification basée sur une architecture d'apprentissage profond qui intègre des connaissances représentées sous forme de graphe de connaissances (Mbiaya et al. 2023, 2024b). Nous proposons une méthode pour construire un graphe de connaissances à partir d'attributs décrivant les images. Ce graphe intègre non seulement des nœuds représentant des attributs, mais également des nœuds pour chaque image et chaque classe. Il peut être enrichi avec des connaissances supplémentaires. Nous calculons une autre représentation des nœuds du graphe et nous proposons une nouvelle fonction de perte utilisant cette représentation, combinée à la classique perte d'entropie croisée utilisée en apprentissage profond. Cette fonction de perte intègre la proximité entre les différents nœuds pour mieux capturer la difficulté de classification. La méthode que nous proposons est particulièrement adaptée pour le problème difficile de classification à grain fin. Une caractéristique importante de notre méthode est que les connaissances ne sont intégrés que pendant la phase d'entraînement et ne sont pas utiles lors de la phase de test, et donc ne nécessitant pas d'avoir des images annotées pour les classer.

Contents

7.1 Formalisation du problème

On considère un jeu de données $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, où $x_i \in \mathcal{X}$ représente une image, et $y_i \in \mathcal{Y}$ représente l'étiquette de classe de l'image. Les étiquettes sont représentées sous forme de vecteur one-hot $y_i \in \{0,1\}^K$, avec $y_{ik} = 1$ si x_i appartient à la classe k, et 0 sinon. Un élément important de notre méthode repose sur la disponibilité d'un graphe de connaissances incluant obligatoirement des nœuds correspondant à chaque image et à chaque classe du jeu de données. Ce graphe peut, dans de rares cas, être directement intégré au jeu de données $\mathcal{D}$. Toutefois, dans la majorité des situations, l'utilisateur fournit des informations sur les images à travers des descripteurs, à partir desquels le graphe doit être construit. Par exemple, les descripteurs `has_wing_color::red` et `has_tree::True` sont des couples `attribut::valeur` issus des ensembles de données CUB (Wah et al. 2011) et SUN Attribute (Patterson et al. 2014), respectivement.

Nous considérons alors des jeux de données d'images enrichis de connaissances, où ces connaissances peuvent être exprimées à travers un ensemble fixe d'attributs utilisés pour décrire les images. Ces attributs et leurs valeurs représentent les connaissances à intégrer, auxquels un poids peut être associé afin de refléter leur importance relative ou leur fiabilité dans le processus d'annotation. Ainsi, le jeu de données est constitué de P attributs $a_1, \ldots, a_P$ dont les domaines sont discrets (un ensemble fini de valeurs, éventuellement booléennes). Chaque image x_i est décrite par ses valeurs v_{ij} pour chaque attribut a_j et un poids $w_{x_i,(a_j,v_{ij})} \in [0,1]$ y est associé. De plus, l'ensemble de classes $y_1, \ldots, y_K$ peut être organisé sous une forme hiérarchique $\mathcal{H}$.

L'objectif est de créer un modèle qui permet de classifier les différentes images en tirant partie du graphe de connaissances, ou des descripteurs dans le cas où un graphe de connaissances n'est pas accessible.

7.2 Méthode proposée

Lors de la phase d'entraînement d'un réseau de neurones convolutif, les couches de convolution sont conçues pour extraire des représentations visuelles discriminantes en vue de séparer les différentes classes. Toutefois, dans des contextes complexes tels que la classification à grain fin, certaines classes peuvent présenter une forte similarité visuelle. Cette proximité sémantique et perceptuelle complique la tâche de discrimination, en limitant la capacité du modèle à apprendre des frontières de décision précises entre les classes concernées.

Pour résoudre ce problème, nous proposons la méthode *KGIC* (Knowledge Graph-based Image Classification) qui s'appuie sur un graphe de connaissances pour améliorer les performances des CNN en classification d'images. La Figure 7.1 donne une vue d'ensemble de *KGIC*, qui comporte deux étapes clés. La première étape consiste en la création d'un graphe de connaissances à partir des couples attributs/valeurs, associé au calcul d'un plongement pour les nœuds. La deuxième étape repose sur la formulation d'une fonction de perte qui prend compte de la similarité entre les représentations calculées. KGIC présente deux avantages principaux : premièrement, il peut être utilisé avec n'importe quelle architecture de

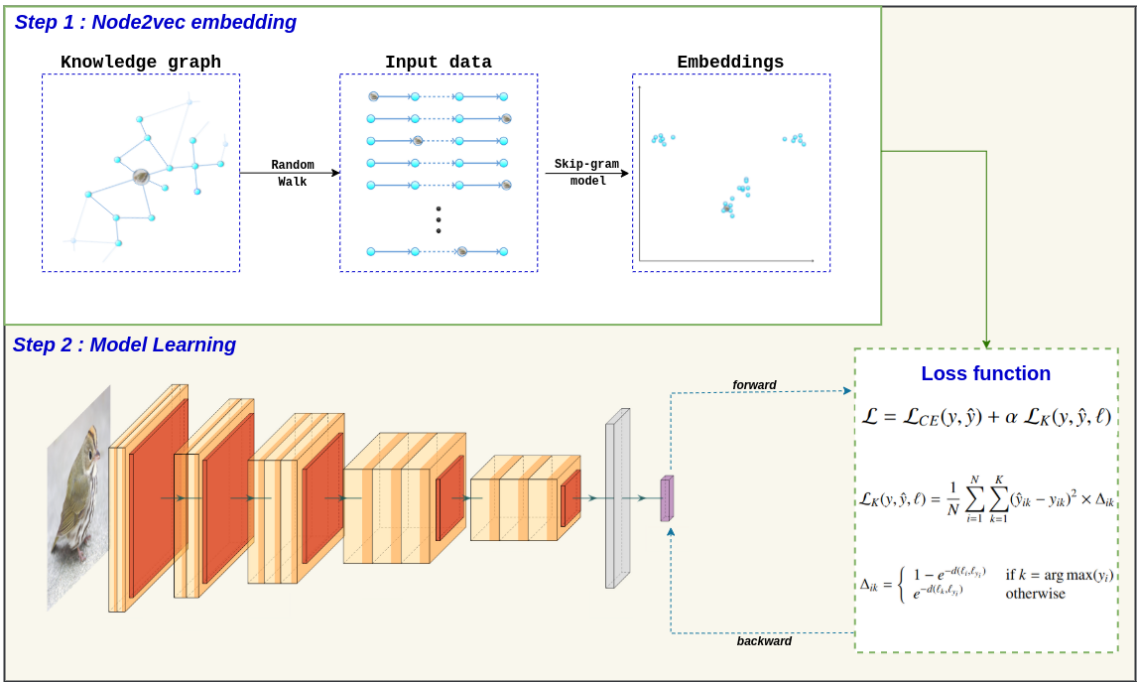


FIGURE 7.1 – Vue d'ensemble de KGIC : un algorithme de plongement (Node2vec) prend en entrée le graphe de connaissances et calcule une représentation des nœuds en fonction de leurs voisinages. Ces représentations sont utilisées dans la fonction de perte lors de l'étape d'apprentissage.

réseau profond, et deuxièmement, il ne nécessite pas de connaissances sur les images de test, puisque les plongements sont utilisés uniquement pendant la phase d'apprentissage.

7.2.1 Création du graphe de connaissances

Dans le cas où le jeu de données n'est pas associé à un graphe de connaissances, mais contient des descripteurs sous forme d'attribut/valeur pour chaque image, la création du graphe de connaissances devient alors cruciale. Pour créer le graphe de connaissances, deux scénarios se présentent en fonction de la nature du graphe, qu'il soit homogène ou hétérogène. Dans le cas hétérogène, une méthode consiste à considérer les images, les classes et les valeurs comme des nœuds, et ensuite considérer les attributs comme des arêtes. Le nombre de types d'arêtes correspond alors au nombre d'attributs. Les arêtes relient les nœuds de valeur aux nœuds d'image, et éventuellement aux nœuds de classe. Cette approche ne tire pas parti des poids des arêtes, ce qui motive l'utilisation des graphes homogènes.

Sur la base des descripteurs sous forme de couple attribut/valeur, nous construisons un graphe de connaissances $G = (V, E)$ comme suit.

- Les nœuds V sont :
  - un nœud (image) v_x pour chaque image x,
  - un nœud (classe) v_y pour chaque classe y,
  - un nœud (attribut-valeur) v_{av} pour chaque couple (a, v), où a est un attribut

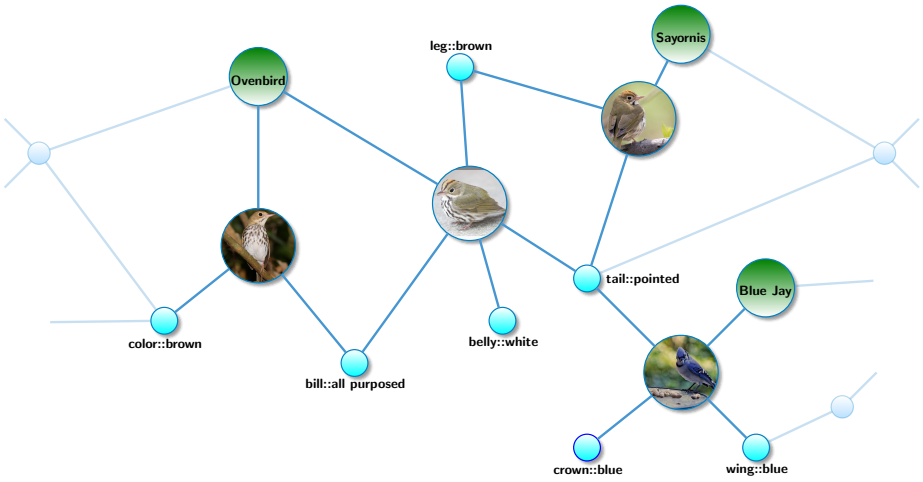


FIGURE 7.2 – Exemple d'un graphe de connaissances créé à partir du jeu de données CUB (Wah et al. 2011). Chaque nœud représente une image, un concept visuel (attribut/valeur) ou une classe. Les arêtes entre les nœuds représentant les images et les concepts visuels sont pondérées en fonction du degré de certitude disponible dans le jeu de données. Les arêtes entre les nœuds des classes et les concepts visuels sont pondérées par la moyenne des certitudes des images appartenant à cette classe.

et v est l'une des valeurs possibles de l'attribut a,

– lorsqu'une hiérarchie est disponible, un nœud (concept) v_c pour chaque concept dans la hiérarchie $\mathcal{H}$.

- Les arêtes, ainsi que leurs poids, sont créées comme suit :
  - Pour chaque image x, soit y sa classe, et v_x et v_y les nœuds représentant x et y respectivement. Une arête $\{v_x, v_y\}$ est créée avec un poids de valeur 1.
  - Pour chaque image x décrite par un attribut a avec la valeur v et un poids $w_{x,(a,v)}$, soit v_x et v_{av} les nœuds représentant x et (a, v) respectivement. Une arête $\{v_x, v_{av}\}$ est créée avec un poids de valeur $w_{x,(a,v)}$.
  - Pour chaque concept c et sa superclasse c', soit v_c et $v_{c'}$ les nœuds les représentant respectivement. Une arête $\{v_c, v_{c'}\}$ est créée avec un poids de valeur $s = 1$.
  - Pour chaque classe y et chaque couple (a, v), soit v_y et v_{av} les nœuds les représentant respectivement. Une arête $\{v_y, v_{av}\}$ est créée avec un poids qui représente la moyenne des poids associés au couple (a, v) sur les images de la classe :
    $$s = \frac{1}{n_y} \sum_{x' \in y} w_{x',(a,v)}$$
    où n_y est le nombre d'images dans la classe y.

La Figure 7.2 présente une partie du graphe de connaissances construit pour le jeu de données CUB.

7.2.2 Plongement du graphe de connaissances

Le plongement de graphes consiste à calculer une nouvelle représentation de ses nœuds dans un espace de dimension d inférieure ($d \ll |V|$), comme présenté dans la Section 2.3.2. Dans notre approche, l'objectif visé par l'intégration du graphe de connaissances est d'exploiter les informations disponibles pour identifier les images les plus difficiles à classifier. Ces informations sont ensuite utilisées pour guider l'apprentissage en mettant davantage l'accent sur les cas complexes afin d'améliorer la capacité du modèle à distinguer les classes visuellement similaires ou peu séparables.

Plus précisément, notre intuition est que les représentations apprises pour différencier les nœuds du graphe peuvent révéler des configurations associées à une difficulté de classification. Deux situations typiques peuvent être distinguées. La première situation est la dissociation image-classe, où une image et sa classe sont éloignées dans l'espace de représentation (faible similarité entre le plongement d'une image et le plongement de sa classe), ce qui caractérise la difficulté d'associer cette image à sa classe. La deuxième situation est le conflit inter-classes, où deux classes différentes sont proches dans l'espace de représentation (forte similarité entre les plongements de deux classes différentes), ce qui caractérise la difficulté de séparer ces deux classes avec un classifieur.

Pour que ces informations soient exploitables durant l'entraînement du modèle, le graphe doit au minimum contenir un nœud par image, un nœud par classe et un nœud par attribut/valeur. Dans ce contexte, nos objectifs lors du calcul des plongements sont triples :

- Maximiser la similarité entre les paires de nœuds d'images appartenant à la même classe ;
- Maximiser la similarité entre les nœuds d'images et les nœuds des classes correspondantes ;
- Maximiser la dissimilarité entre les paires de nœuds de classes.

Des expérimentations ont été menées pour sélectionner l'algorithme offrant les meilleures performances selon ces trois objectifs pour les graphes homogènes pondérés. Ces expérimentations montrent que les deux premiers objectifs correspondent à la préservation des similarités de premier et second ordre. L'algorithme Node2vec (Grover and Leskovec 2016) présenté dans la Section 2.3.2 s'est avéré le plus efficace après nos expérimentations. Il repose sur deux étapes principales : des marches aléatoires pour générer le voisinage de chaque nœud dans le graphe de connaissances, et l'utilisation d'un algorithme de type Skip-gram pour apprendre une nouvelle représentation pour chaque nœud.

7.2.3 Apprentissage pour la classification

Après avoir obtenu une nouvelle représentation pour tous les nœuds du graphe, le jeu de données devient $\mathcal{D} = \{(x_i, \ell_i, y_i)\}_{i=1}^{n}$, où $\ell_i \in \mathbb{R}^d$ représentent la nouvelle représentation de la i-ème image dans l'espace des connaissances obtenu à partir du graphe. Nous précisons

que le label $y_i = [0, \ldots, 1, \ldots, 0]$ est représenté sous la forme one-hot, avec $y_{i_k} = 1$ si x_i appartient à la classe k, et 0 sinon.

Perte liée à la connaissance

Les architectures profondes peuvent rencontrer des difficultés à distinguer les informations pertinentes permettant de séparer des classes visuellement similaires. Un exemple est illustré dans la Figure 7.3b, qui présente la matrice de confusion obtenue après l'entraînement d'un modèle ViT (Dosovitskiy et al. 2020) sur le jeu de données CUB. On observe que les classes 141 à 147 sont souvent confondues en raison de leur grande similarité visuelle (Figure 7.3a). L'intégration de connaissances complémentaires peut fournir des informations pendant le processus d'apprentissage sur la proximité entre une image et sa classe (difficulté d'association) ainsi que sur la proximité entre les classes (difficulté de séparation). Le modèle peut alors ajuster ses poids en tenant compte de la complexité de la tâche de classification. Plus précisément, une nouvelle fonction de perte $\mathcal{L}_K$ utilisant les représentations du graphe de connaissances est introduite comme suit :

$$\mathcal{L}_K(y, \hat{y}, \ell) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} (\hat{y}_{ik} - y_{ik})^2 \times \Delta_{ik} \tag{7.1}$$

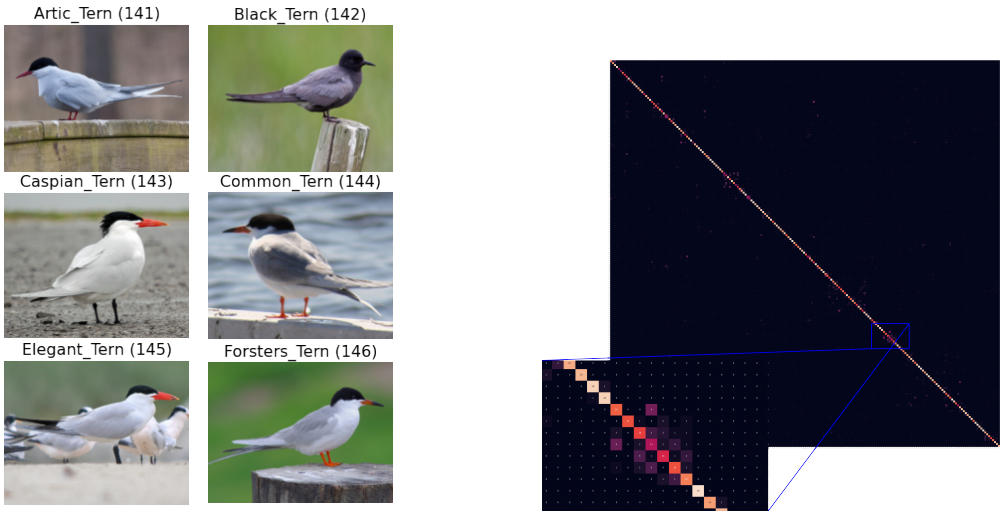
où $\hat{y}_{ik} \in [0, 1]$ représente la prédiction effectuée par le modèle, et Δ_{ik} est un poids qui augmente lorsque la séparation entre les classes devient plus difficile dans l'espace de plongement. En considérant la classe k, Δ_{ik} est formulé comme suit :

$$\Delta_{ik} = \begin{cases} 1 - e^{-d(\ell_i, \ell_{y_i})} & \text{si } k = \arg\max(y_i) \\ e^{-d(\ell_k, \ell_{y_i})} & \text{sinon} \end{cases} \tag{7.2}$$

Dans l'équation 7.2, en tenant compte du plongement du graphe de connaissances, ℓ_i correspond à la représentation du nœud associé à l'image x_i, ℓ_{y_i} correspond à la représentation du nœud de la classe de vérité terrain de x_i, et ℓ_k correspond à la représentation du nœud de la classe k. La fonction $d(\bullet)$ représente la distance euclidienne.

L'intervalle de Δ_{ik} est donc $[0, 1]$, et son comportement se décrit comme suit : lorsque la classe k correspond à la classe de vérité terrain attendue ($k = \arg\max(y_i)$), Δ_{ik} mesure la dissimilarité entre l'image i et la classe attendue : Δ_{ik} est plus grande lorsque la représentation de l'image est plus éloignée de celui de sa classe ($1 - e^{-d(\ell_i, \ell_{y_i})}$ augmente lorsque $d(\ell_i, \ell_{y_i})$ augmente). Cela revient à accorder un poids d'apprentissage plus important aux images dont la représentation dans l'espace de plongement est éloignée de celle de leur classe cible.

Cependant, lorsque la classe k ne correspond pas à la classe attendue ($k \neq \arg\max(y_i)$), Δ_{ik} mesure la similarité entre la classe k et la classe attendue : Δ_{ik} est plus petite lorsque la représentation de la classe k est plus éloignée de la représentation de la classe attendue (c'est-à-dire que $e^{-d(\ell_k, \ell_{y_i})}$ diminue lorsque $d(\ell_k, \ell_{y_i})$ augmente). Cela revient à accorder plus de poids lorsque les classes k et y_i sont proches dans l'espace de plongement.



(a) (b)

FIGURE 7.3 – Illustration de la difficulté de classification dans le jeu de donnée CUB. (7.3a) : Six exemples d'oiseaux appartenant à différentes classes qui sont difficiles à distinguer les uns des autres en raison de leur grande similitude visuelle. Ces classes incluent le Sterne arctique, le Sterne noir, le Sterne caspien, le Sterne commun, le Sterne élégant et le Sterne de Forster. (7.3b) : Matrice de confusion obtenue après l'entraînement d'un modèle ViT (Dosovitskiy et al. 2020) sans intégration de connaissances. Les classes mises en évidence sont les classes 141 à 147.

Perte finale

Le modèle est entraîné en utilisant la somme pondérée de deux pertes comme définie ci-dessous : la perte d'entropie croisée $\mathcal{L}_{CE}(y, \hat{y})$ entre la classe prédite $\hat{y}$ et la classe réelle y, et la perte liée à la connaissance.

$$\mathcal{L} = \mathcal{L}_{CE}(y, \hat{y}) + \alpha \, \mathcal{L}_{K}(y, \hat{y}, \ell) \tag{7.3}$$

où α est un hyperparamètre et $\mathcal{L}_{CE}$ est la perte d'entropie croisée standard, formulée par :

$$\mathcal{L}_{CE}(y, \hat{y}) = -\sum_{k=1}^{K} y_k \log(\hat{y}_k) \tag{7.4}$$

où y est la classe réelle en format one-hot et $\hat{y}$ est la prédiction du modèle. L'algorithme 2 décrit l'ensemble du processus d'apprentissage de notre méthode. Dans cet algorithme, f est un réseau profond qui prend en entrée un vecteur d'images X et génère une prédiction. Les plongements des nœuds dans le graphe de connaissances sont utilisés dans la fonction de perte. L'algorithme 2 peut donc fonctionner avec différents types de réseaux.

Algorithm 2 : KGIC

Input : X : images, Y : labels, G : Knowledge Graph, d p q : embedding parameter, $f(\bullet)$: Deep Network, λ : Learning rate, α : Model parameter

Output : Θ : Model Weights

$D \leftarrow Random\ Walks(G, p, q)$ {Eq. 2.10}
$L \leftarrow Skip-gram(D, d)$ {Eq. 2.13}
$\Theta \leftarrow$ Weights initialization of $f(\bullet)$
for all $epoch \in [1, num\_epoch]$ **do**
  for all $batch \in [1, num\_batch]$ **do**
    $\hat{Y} = f(X, \Theta)$
    $\mathcal{L}_{CE} = Cross\ Entropy(Y, \hat{Y})$ {Eq. 7.4}
    $\mathcal{L}_{K} = Knowledge\ loss(Y, \hat{Y}, L)$ {Eq. 7.1}
    $\mathcal{L}_{batch} = \mathcal{L}_{CE} + \alpha\mathcal{L}_{K}$ {Eq. 7.3}
    $\Theta = \Theta - \lambda\partial\mathcal{L}_{batch}$ {Learning parameter}
  end for
end for

7.3 Protocoles Expérimentaux

7.3.1 Jeux de données

Les jeux de données utilisés dans nos expérimentations sont CUB, SunIn, SunOut, SunInOut, VG12 et VG48. Particulièrement pour le jeu de données CUB, le graphe a été enrichi avec des informations hiérarchiques afin de mieux représenter les relations entre les différentes classes d'oiseaux. Plus précisément, des nœuds ont été ajoutés pour représenter une hiérarchie des classes, regroupées en genres, familles et ordres. Chaque nœud de classe est connecté à son nœud de genre correspondant, les nœuds de genre sont connectés à leurs nœuds de famille respectifs, et les nœuds de famille à leurs nœuds d'ordre. Le poids de tous ces liens hiérarchiques a été uniformément fixé à 1, garantissant une représentation fidèle de la hiérarchie dans la structure du graphe.

7.3.2 Apprentissage

Pour le plongement des graphes de connaissances, les paramètres p et q sont respectivement définis à 0.1 et 1.0. L'objectif est de garantir que la marche aléatoire préserve l'homophilie, de sorte que les nœuds voisins aient des représentations proches dans l'espace de plongement. Les graphes de connaissances sont créés uniquement à partir des données d'entraînement et les nœuds sont projetés dans un espace de $d = 128$ dimensions. Les pertes ne sont pas à la même échelle ($\mathcal{L}_{K} \ll \mathcal{L}_{CE}$) et la valeur de α dépend à la fois des caractéristiques des données et de la structure du graphe associé.

Dans le cadre de nos expérimentations, nous avons eu recours à une procédure de validation croisée classique afin de déterminer la valeur optimale du paramètre α pour le

jeu de données CUB. Plus précisément, nous avons exploré une plage de valeurs comprises entre 0.1 et 5, en appliquant une validation croisée à k-fold, avec $k = 10$, sur un sous-ensemble aléatoire de l'ensemble d'apprentissage de taille N. À chaque itération, l'ensemble d'apprentissage était partitionné en k segments de taille équivalente : l'un d'entre eux était réservé à l'évaluation (fold de validation), tandis que les $(k-1)$ autres étaient utilisés pour entraîner le modèle avec une valeur α donnée. Le taux d'erreur de classification obtenu sur le segment de validation était alors enregistré. Cette procédure a permis d'estimer, pour chaque valeur de α considérée, sa capacité à généraliser sur des données non vues, et ainsi de sélectionner la valeur offrant les meilleures performances moyennes. Ce processus a été répété pour chaque valeur de α et le taux d'erreur moyen à travers les k-folds a été pris en compte. La valeur optimale de α a été choisie comme celle qui donnait le taux d'erreur moyen le plus bas. Pour les autres jeux de données, la valeur de α choisie sur CUB a été maintenue. Toutefois, sur ces autres jeux de données, nous montrons l'influence de α sur les différents résultats.

Basé sur ViT, KGIC est pré-entraîné sur ImageNet (Deng et al. 2009) avec 1000 classes. La dernière couche linéaire du réseau a été modifiée pour correspondre au nombre de classes apprises. Comme pour d'autres méthodes de pointe, les images d'entrée sont d'abord redimensionnées à 600×600, puis recadrées à 448×448. Un retournement horizontal aléatoire est utilisé pour l'augmentation des données, et une augmentation de couleur supplémentaire est appliquée. Nous avons utilisé SGD pour optimiser le réseau avec un momentum de 0.9, et un taux d'apprentissage initialisé à $2 \cdot 10^{-2}$ avec un planificateur de décroissance cosinus. La taille du lot est fixée à 8 pour tous les ensembles de données. Pour le jeu de données CUB, nous avons utilisé les images d'entraînement et de test d'origine. Pour les autres jeux de données, nous adoptons une stratégie de validation croisée à 5 plis. Toutes les expériences sont réalisées sur des GPUs NVIDIA Tesla V100.

7.3.3 Méthodes comparatives

L'objectif de nos expérimentations est triple. Dans un premier temps, nous présentons les résultats de l'étape de plongement du graphe de connaissances et démontrons que celui-ci capture la difficulté de classification. Ensuite, nous analysons l'impact de l'intégration des connaissances dans l'apprentissage des architectures profondes. Enfin, nous évaluons la contribution de la connaissance dans les résultats finaux.

Notre méthode étant adaptée pour les problèmes de classification difficiles, nous comparons KGIC à plusieurs approches adaptées à la classification à grain fin. Les méthodes basées sur l'utilisation de tâches auxiliaires cherchent à détecter des parties discriminantes dans les images en proposant des architectures spécifiques (DCL (Chen et al. 2019b), NTS-Net (Yang et al. 2018)) ou en utilisant des formes désorganisées des images (DCL (Chen et al. 2019b), PMG (Du et al. 2020)). Ces modèles ont été présentés dans la Section 4.2.1, et s'appuient sur des architectures basées sur des CNNs sans utiliser aucune forme de connaissance. Alors que les méthodes précédentes utilisent des CNNs, FFVT (Wang et al. 2021b) et SIM-Trans (Sun et al. 2022) utilisent des architectures basées sur des ViTs. Leurs particularités reposent donc sur l'utilisation de mécanismes d'attention, présentées dans la Section 4.2.2. D'autres méthodes utilisent des fonctions de perte contrastive dans le but de renforcer la discrimination inter-classes et la cohérence intra-classe lors de l'apprentissage. Ces méthodes, basées sur

un ViTs pour TransFG (He et al. 2022) ou sur un CNNs pour API-Net (Zhuang et al. 2020), sont présentées dans la Section 4.2.3. Contrairement aux méthodes précédentes, KERL (Chen et al. 2018b) utilise des connaissances explicites pour le problème de classification à grain fin. Il exploite un mécanisme d'attention entre les caractéristiques extraites d'un réseau CNN et une représentation de graphe de connaissances calculée à partir d'un GGNNs pour guider la sélection de régions discriminantes. Comme présenté dans la Section 4.2.4, le graphe est construit à partir d'annotations d'attributs dans les jeux de données.

7.4 Résultats et discussions

7.4.1 Visualisation du graphe de connaissances

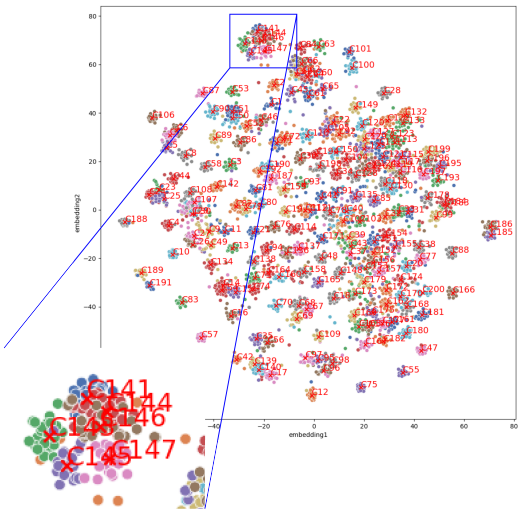
Suite à l'application de l'algorithme Node2Vec pour le plongement du graphe de connaissances, chaque nœud est projeté dans un espace euclidien, produisant ainsi une représentation vectorielle. La Figure 7.4 illustre le résultat de ce plongement pour les nœuds correspondant aux images et aux classes, dans les graphes construits à partir des jeux de données CUB, Synthetic, COCO@11 et SunIn. Les points colorés représentent les nœuds image, chaque couleur indiquant la classe associée, tandis que les croix rouges correspondent aux nœuds représentant les classes.

Les différentes visualisations des plongements montrent que les classes visuellement similaires tendent à se regrouper dans l'espace de représentation. Ce phénomène est particulièrement observable dans le cas du jeu de données CUB (Figure 7.4a), où les classes 141 à 147, difficiles à distinguer visuellement (comme illustré dans la Figure 7.3a), présentent des représentations proches après plongement. Pour les jeux de données Synthetic (Figure 7.4b) et COCO@11 (Figure 7.4c), les classes illustrées sont bien séparées, traduisant la capacité pour les attributs à discriminer les classes. En revanche, pour le jeu de données SunIn, les attributs disponibles semblent insuffisants pour assurer une séparation nette entre les classes, comme le suggèrent également les performances rapportées dans la Table 7.3.

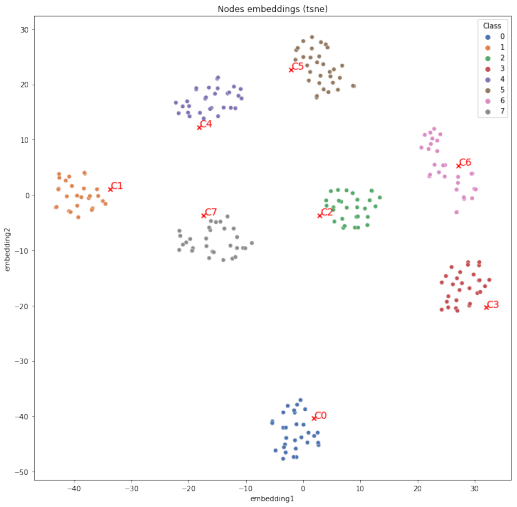
La Figure 7.4 met en évidence le fait que les nœuds images sont, dans la majorité des cas, correctement regroupés autour de leurs nœuds de classes respectifs. Cette configuration suggère que le graphe encode de manière cohérente les similarités structurelles et visuelles entre les classes, utiles pour les séparer. Toutefois, certains nœuds images apparaissent significativement éloignés de leur nœud classe associé, ce qui peut indiquer une ambiguïté visuelle accrue pour ces exemples (Figure 7.4d par exemple), rendant leur classification plus difficile dans un cadre d'apprentissage profond. Cette observation confirme la pertinence d'exploiter la structure du graphe pour guider l'apprentissage, notamment en facilitant la prise en compte des cas ambigus.

7.4.2 Comparaison avec les méthodes de l'état de l'art

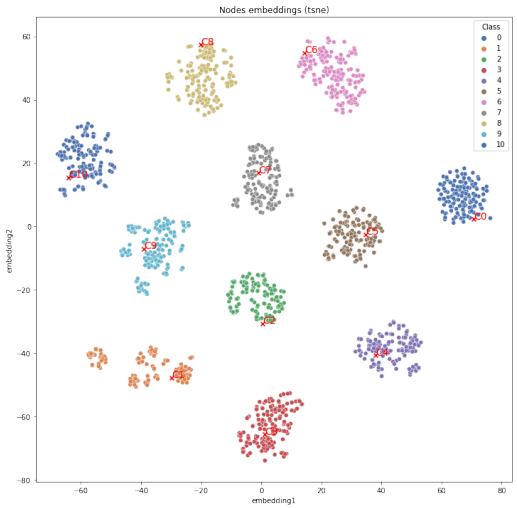
La Table 7.1 présente les performances de KGIC sur le jeu de données CUB en comparaison avec différentes méthodes de l'état de l'art pour la classification à grain fin. Le modèle atteint une précision de 91,7%, dépassant de 0,1% les méthodes FFVT et SIM-Trans,



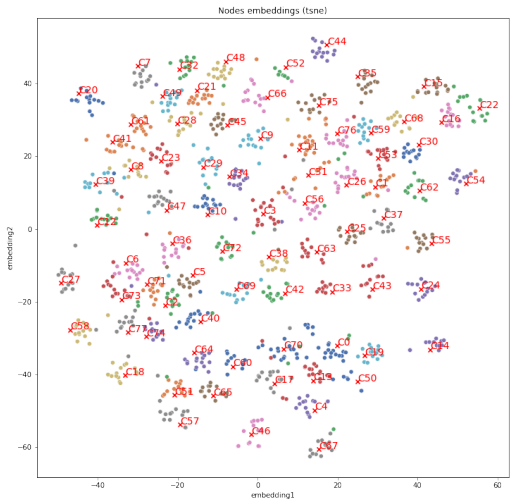
(a) CUB



(b) Synthetic



(c) COCO@11



(d) SunIn

FIGURE 7.4 – Projection avec t-SNE des images d'entraînement et des classes obtenues après le plongement des jeux de données CUB, Synthetic, COCO@11 et SunIn.

Méthodes	Backbone	Acc.(%)
DCL (Chen et al. 2019b)	ResNet-50	86.8 (87.8)
NTS-Net (Yang et al. 2018)	ResNet-50	87.6 (87.5)
PMG (Du et al. 2020)	ResNet-50	89.6 (89.6)
API-Net (Zhuang et al. 2020)	ResNet-50	89.8 (90.0)
ViT (Dosovitskiy et al. 2020)	ViT-B_16	90.2 (-)
TransFG (He et al. 2022)	ViT-B_16	91.3 (91.7)
FFVT (Wang et al. 2021b)	ViT-B_16	<u>91.6</u> (91.6)
SIM-Trans (Sun et al. 2022)	ViT-B_16	91.6 (91.8)
KERL (Chen et al. 2018b)	VGG-16	- (86.3)
KGIC (ours) (Mbiaya et al. 2024b)	ViT-B_16	**91.7**

TABLE 7.1 – Comparaison des méthodes de l'état de l'art sur le jeu de données CUB. La meilleure précision est mise en gras et la deuxième meilleure précision est soulignée. Les valeurs entre parenthèses correspondent aux résultats rapportés dans les articles respectifs. La valeur de α est fixée à 1.0 après validation croisée.

également basées sur l'architecture ViT-B_16, qui obtiennent les deuxièmes meilleurs résultats. KGIC surpasse nettement l'ensemble des approches concurrentes, notamment KERL, qui exploite également des annotations supplémentaires, avec un écart de 5% par rapport aux résultats reportés dans leurs travaux. Il est important de souligner que, lorsque le terme d'intégration de connaissances est désactivé (c'est à dire en fixant $\alpha = 0$ dans l'équation 7.3), l'architecture devient équivalente à celle de ViT. L'écart de performance de 1,5% observé entre ViT et KGIC met en évidence la contribution effective et efficace de l'intégration des connaissances durant l'apprentissage. Enfin, il est important de rappeler que la méthode proposée présente l'avantage d'être compatible avec n'importe quelle architecture de base, sans augmenter le nombre de paramètres.

La Table 7.2 compare les performances de KGIC à celles de plusieurs méthodes de classification à grain fin de l'état de l'art sur différents jeux de données. Pour rappel, la valeur de α a été déterminé automatiquement par validation croisée sur l'ensemble d'apprentissage. Le modèle KGIC surpasse toutes les autres méthodes sur la majorité des jeux de données, avec des gains particulièrement notables sur COCO@48, SunIn, SunOut, SunInOut et VG12, où il atteint respectivement 65.6%, 84.6%, 79.8%, 78.4% et 88.3% de précision, soit des marges respectives de 1.8, 1.2, 2.7, 3.1 et 2.9. Sur les jeux de données COCO@24 et VG48, KGIC obtient de meilleures performances avec des marges moins importantes, soit 0.8 et 0.6 respectivement. Sur le jeu de données Synthetic, KGIC obtient la deuxième meilleure performance avec une différence de 2.1 point par rapport à PMG qui obtient la meilleure performance. Comparé à ViT, qui correspond à KGIC sans utilisation de connaissances, notre approche atteint toujours des performances significativement plus élevées avec des marges allant de 0.9% sur VG48 à 6.0% sur SunOut. Ces résultats illustrent la capacité du modèle à exploiter efficacement les connaissances, en améliorant la représentation et la discrimination des classes dans des contextes visuellement complexes.

La Table 7.3 présente les résultats obtenus en n'utilisant que les connaissances, c'est-à-dire sans utilisation des informations visuelles issues des images. On constate que, malgré l'absence d'informations visuelles, les performances restent relativement élevées grâce à la richesse descriptives des attributs utilisés. Une exception est faite pour les jeux de données SunIn, SunOut et SunInOut, où les performances sont nettement plus faibles en l'absence d'informations visuelles. Cela s'explique par le fait que, pour ces jeux de données, les connaissances ne suffisent pas à elles seules à bien discriminer les classes. Néanmoins, lorsqu'elles sont intégrées conjointement aux informations visuelles, ces connaissances permettent d'orienter l'apprentissage et d'améliorer les performances du modèle. Il est important de noter que avec KGIC, les connaissances ne sont pas disponibles pendant la phase de test. C'est pourquoi notre approche les exploite exclusivement pendant l'entraînement à travers la fonction de perte, pour garantir que le modèle reste indépendant de celles-ci au moment de l'inférence.

Méthodes	Backbone	Synthetic	COCO@11	COCO@24	COCO@48	SunIn	SunOut	SunInOut	VG12	VG48
NTS-Net (Yang et al. 2018)	ResNet-50	46.2	59.3	56.5	54.8	64.3	56.4	56.1	83.7	54.2
PMG (Du et al. 2020)	ResNet-50	**52.9**	63.5	62.4	59.7	68.0	61.8	62.9	84.7	57.9
API-Net (Zhuang et al. 2020)	ResNet-101	42.7	60.4	58.7	50.0	71.1	62.1	62.7	83.8	57.2
ViT (Dosovitskiy et al. 2020)	ViT-B_16	45.7	63.8	64.5	63.6	79.2	73.8	73.8	84.2	58.8
TransFG (He et al. 2022)	ViT-B_16	46.2	<u>65.9</u>	63.3	61.7	77.8	73.7	72.3	84.8	<u>59.3</u>
FFVT (Wang et al. 2021b)	ViT-B_16	47.8	**66.0**	<u>65.7</u>	<u>63.8</u>	<u>83.4</u>	<u>77.1</u>	<u>75.3</u>	<u>85.4</u>	58.5
SIM-Trans (Sun et al. 2022)	ViT-B_16	47.2	61.1	64.0	61.5	80.4	72.2	74.0	84.2	58.2
KGIC (ours) (Mbiaya et al. 2024b)	ViT-B_16	<u>50.8</u>	65.8	**66.5**	**65.6**	**84.6**	**79.8**	**78.4**	**88.3**	**59.7**

TABLE 7.2 – Comparaison des méthodes à l'état de l'art sur les jeux de données SunIn, SunOut, SunInOut, VG12 et VG48. La meilleure précision est mise en gras et la deuxième meilleure précision est soulignée. La valeur de α est fixée à 1.0 comme sur CUB.

Jeu de données	Synthetic	COCO@11	COCO@24	COCO@48	CUB-Simp	VG12	SunIn	SunOut	SunInOut
Arbre de décision	97.99	97.15	97.47	96.83	100.0	97.83	18.38	14.16	14.91
Forêt aléatoire	99.00	99.22	98.73	99.38	100.0	97.83	31.41	24.73	27.68
SVM	100.0	98.70	97.61	98.60	100.0	98.19	30.13	28.14	30.12

TABLE 7.3 – Évaluation de la capacité des attributs à séparer les classes dans les différents jeux de données.

α	Synthetic	CUB	SunIn	SunOut	SunInOut	VG12	VG48	COCO@11	COCO@24	COCO@48
0.1	47.2	91.4	85.4	79.4	77.2	87.3	59.8	65.4	65.9	65.4
0.5	48.7	91.5	84.0	79.4	78.1	87.7	59.8	65.9	66.2	65.4
1.0	50.8	**91.7**	84.6	**79.8**	**78.4**	**88.3**	59.7	65.8	66.5	**65.6**
1.5	**51.3**	91.6	84.6	78.9	77.9	87.6	59.8	65.5	65.0	65.5
2.0	50.7	91.5	84.8	79.2	77.8	88.0	**60.0**	**66.7**	66.4	65.2
2.5	50.7	**91.7**	85.4	79.6	77.7	88.0	59.5	65.6	**67.0**	65.3
3.0	49.7	91.5	**85.8**	79.6	77.3	87.8	59.8	66.5	66.4	65.2
3.5	48.7	91.6	85.0	78.9	77.0	87.6	59.8	65.2	66.0	65.2
4.0	46.7	91.4	84.6	78.7	77.1	87.1	59.7	65.2	64.9	65.1

TABLE 7.4 – Résultats de notre méthode KGIC sur différents jeux de données avec différentes valeurs de α. La meilleure précision est mise en gras pour chaque jeu de données.

7.4.3 Contribution de la Connaissance

La Figure 7.5 illustre les performances obtenues par KGIC sur les dix jeux de données, pour différentes valeurs de l'hyperparamètre α, variant de 0.1 à 4.0. Cette figure présente la valeur de α pour laquelle la précision est maximale sur chaque jeu de données. Les résultats sont chiffrés dans la Table 7.4 où chaque ligne représente une valeur de α et chaque colonne représente un jeu de données. Pour rappel, le paramètre α contrôle le poids de la connaissances dans la perte de KGIC. On observe que la meilleure valeur de α dépend du jeu de données, avec une valeur $\alpha = 1.0$ qui permet d'obtenir les meilleures performances sur la majorité des jeux de données, notamment sur CUB (91.7%), SunOut (79.8%), SunInOut (78.4%) et VG12 (88.3%) et COCO@48 (65.6%). Sur les jeux de données VG48 et COCO@11, les performances maximales sont respectivement à 60.0% et 66.7% avec $\alpha = 2.0$. Synthetic, SunIn et COCO@24 atteignent respectivement leurs performances maximales avec $\alpha = 1.5$ (51.3%), $\alpha = 3.0$ (85.8%) et $\alpha = 2.5$ (67.0%). Ces résultats confirment une tendance générale selon laquelle les meilleures performances sont obtenues pour des valeurs intermédiaires de α, généralement comprises entre 1.0 et 3.0. Lorsque α devient trop faible (ex. 0.1) ou trop élevé (ex. 4.0), les performances se dégradent, ce qui indique qu'une pondération inadéquate de la contribution des connaissances peut nuire à l'apprentissage du modèle.

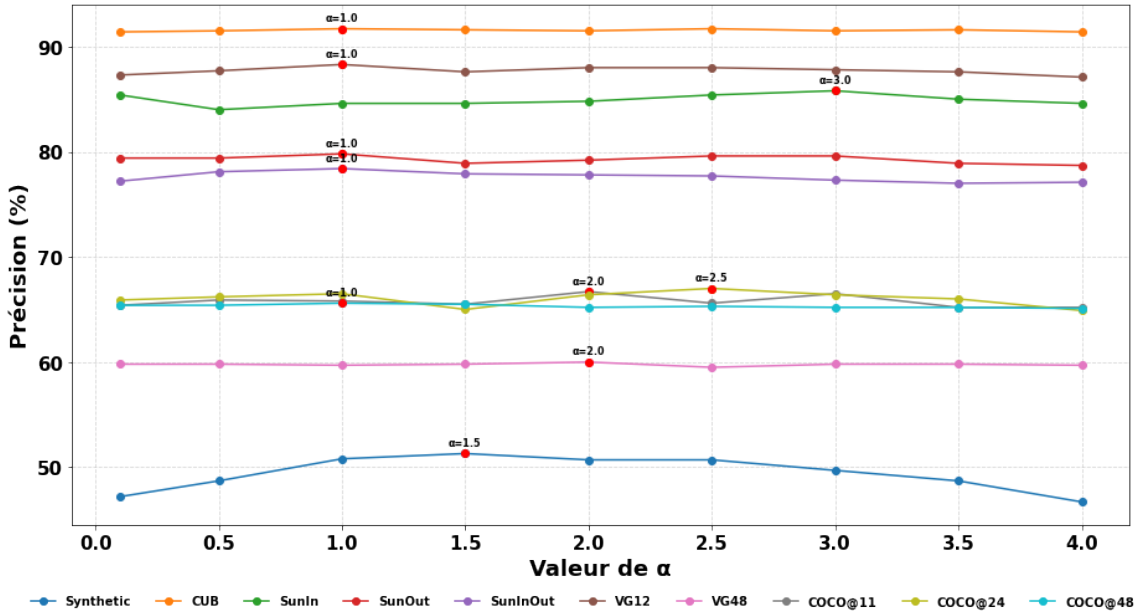


FIGURE 7.5 – Illustration des résultats sur différents jeux de données en fonction de l'hyperparamètre α pour KGIC.

7.5 Conclusion

Dans ce chapitre, nous avons introduit une méthode d'intégration de connaissances dans les architectures profondes pour la classification d'images. Ces connaissances sont représentées sous forme d'un graphe reliant des nœuds correspondant aux images et aux

classes. Ce graphe est exploité à travers une nouvelle fonction de perte. Cette dernière combine l'information directe issue du classifieur et la connaissance encodée dans le graphe. L'équilibre entre ces deux composantes est contrôlé par un paramètre de pondération, ajusté automatiquement par validation croisée. Cette intégration permet au modèle de mieux appréhender la proximité entre une image et ses classes associées, ainsi que les relations entre classes. La fonction de perte additionnelle oriente l'apprentissage vers les zones de confusion, incitant le modèle à renforcer sa capacité à distinguer les classes difficiles à séparer. Le modèle ajuste ainsi ses représentations en fonction de la complexité des cas à discriminer, guidé par les connaissances structurelles disponibles.

La méthode proposée est générique et peut être appliquée à toute architecture de réseau de neurones. Notamment, lors de la phase d'inférence, seules les images sont nécessaires en entrée, car les connaissances issues du graphe sont exploitées uniquement pendant l'apprentissage. Les comparaisons expérimentales avec l'état de l'art montrent une amélioration des performances sur l'ensemble des jeux de données, et une justification de l'intérêt d'utiliser des connaissances à priori pour guider le modèle pendant l'apprentissage. Dans le chapitre suivant, nous présenterons une méthode d'intégration de connaissances également basée sur des notions de concepts, mais dans une approche différente où ces concepts seront appris directement dans les images.

Chapitre 8

Intégration de règles dans la classification profonde à base de concepts

Ce chapitre présente une méthode de classification qui s'appuie sur l'approche de modèles basés sur les concepts, où les concepts sont des attributs visuels comme dans le chapitre précédent. L'objectif est de prédire une étiquette de classe pour une image à partir des concepts qu'elle contient, et donc d'identifier aussi les concepts qui caractérisent cette image. Plus précisément, chaque image est annotée à la fois avec ses étiquettes de classe et un ensemble de concepts. L'enjeu est alors de structurer l'apprentissage de manière à apprendre des représentations intermédiaires qui s'alignent avec la présence des concepts dans l'image, afin de favoriser une classification plus explicable. Nous proposons une méthode qui exploite des connaissances formalisées sous forme de règles, qui tirent parti de l'interdépendance entre les concepts et les classes. À partir de ces règles, nous introduisons un graphe de connaissances spécifique, qui permet d'effectuer un raisonnement sur ces connaissances, améliorant ainsi les performances du modèle basé sur les concepts et atteignant un équilibre entre explicabilité et efficacité. De plus, notre méthode prend en charge l'intervention humaine en phase de test, permettant aux experts d'interagir avec le modèle lors de l'inférence en corrigeant directement ses prédictions sur les concepts. Notre approche, qui intègre des connaissances, apporte une valeur ajoutée en corrigeant automatiquement d'autres concepts grâce au raisonnement : lorsqu'un humain intervient sur un concept, la phase de raisonnement ajuste les prédictions des autres concepts, ce qui améliore la précision de la tâche.

Contents

8.1 Motivation

Bien que les *Concept Bottleneck Models* (CBM) (Koh et al. 2020) et leurs variantes CEM (Zarlenga et al. 2022), ProbCBM (Kim et al. 2023) et IntCEM (Espinosa Zarlenga et al. 2023) aient apporté des avancées significatives en matière d'explicabilité et de gestion de l'incertitude, ils partagent une limitation commune : ils ne tirent pas pleinement parti des dépendances entre les concepts. CBM impose un goulot d'étranglement strict où les concepts sont traités comme des variables indépendantes, entraînant une perte d'information et une diminution du pouvoir prédictif. CEM améliore cette approche en utilisant des représentations de haute dimension pour chaque concept, capturant des informations plus riches, mais sans remettre en cause l'hypothèse d'indépendance des concepts. ProbCBM renforce la robustesse en modélisant l'incertitude dans les prédictions de concepts, sans pour autant exploiter les relations structurelles ou logiques entre eux. Même IntCEM, bien qu'offrant une approche plus expressive, ne tire pas pleinement parti des interdépendances conceptuelles de manière structurée. Dans de nombreuses tâches du monde réel, les concepts sont intrinsèquement liés, et ignorer ces dépendances limite la capacité du modèle à généraliser efficacement. Nous proposons de remédier à cette limite en intégrant explicitement les relations entre concepts, permettant ainsi des prédictions plus cohérentes et mieux contextualisées.

8.2 Formalisation du problème

On considère un jeu de données $\mathcal{D} = \{(x_i, c_i, y_i)\}_{i=1}^n$, où $x_i \in \mathcal{X}$ représente une image d'entrée, $c_i \in \mathcal{C} \subseteq \{0,1\}^o$ représente la vérité terrain sur la présence ou l'absence de o concepts dans l'image x_i (c_i est un vecteur binaire où chaque entrée représente si un concept est actif ou non), et $y_i \in \mathcal{Y}$ représente l'étiquette de classe de l'image. Les étiquettes sont représentées sous forme de vecteur one-hot $y_i \in \{0,1\}^K$, avec $y_{ik} = 1$ si x_i appartient à la classe k, et 0 sinon. La Figure 8.1 illustre des exemples d'images de différentes classes issues du jeu de données CUB (Wah et al. 2011) ainsi que quelques concepts associés à chaque image.

Soient $\mathbf{c}_1, \ldots, \mathbf{c}_o$ les concepts décrivant le contenu des images du jeu de données $\mathcal{D}$. Soient $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_o \in [0,1]$ les degrés de vérité de chaque concept pour une image donnée. Par exemple, la première image de la classe *Indigo Bunting* dans la Figure 8.1 (en haut à gauche) contient le concept $\texttt{wing\_color} : \texttt{blue}$ ($\mathbf{c}_{\texttt{wing\_color:blue}}$) et ne contient pas le concept $\texttt{wing\_color} : \texttt{black}$ ($\mathbf{c}_{\texttt{wing\_color:black}}$). Ainsi, pour cette image, on peut écrire $\hat{\mathbf{c}}_{\texttt{wing\_color:blue}} = 1$ (présence du concept $\texttt{wing\_color} : \texttt{blue}$ dans l'image), $\hat{\mathbf{c}}_{\texttt{wing\_color:black}} = 0$ (absence du concept $\texttt{wing\_color} : \texttt{black}$ dans l'image).

Dans notre cadre, le jeu de données contient des règles propositionnelles, soit fournies par un expert, soit générées à partir des données. Ces règles peuvent être de deux types : (1) des règles de classe et (2) des règles de concepts. Chaque règle est associée à un support. Le support d'une règle $A \rightarrow B$ dans un jeu de données $\mathcal{D}$ est défini comme le nombre d'images de $\mathcal{D}$ contenant les concepts de $A \cup B$. Les règles peuvent être directement disponibles avec le jeu de données. Dans le cas contraire, elles sont générées à l'aide de la méthode décrite dans le chapitre 6. L'objectif est de créer un modèle profond basé sur les concepts qui permet dans un premier temps de détecter les différents objets dans une image, ensuite de

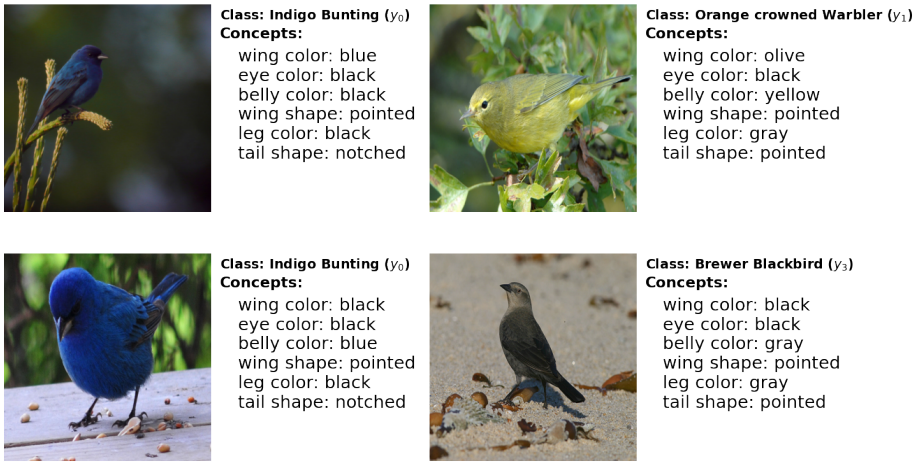


FIGURE 8.1 – Exemples d'images du jeu de données CUB (Wah et al. 2011) avec certains concepts correspondant à ces images. L'image en haut à gauche représente la classe *Indigo Bunting*, tandis que l'image en haut à droite présente un oiseau de la classe *Orange-crowned Warbler*. L'image en bas à gauche correspond également à la classe *Indigo Bunting*, et l'image en bas à droite présente un oiseau de la classe *Brewer Blackbird*.

déterminer la classe de cette image à partir des concepts détectés. Ce modèle doit tirer parti des connaissances disponibles dans le jeu de données (ou créées à partir des concepts) pour être plus performant. Il sera donc capital pour nous de pouvoir évaluer les performances du modèle avec et sans connaissances.

8.2.1 Règles de classe

Les règles de classe définissent les classes à partir des concepts. Lorsqu'elles sont générées à partir du jeu de données, elles possèdent une confiance égale à 1 dans l'ensemble d'entraînement. Nous rappelons qu'une règle de classe peut être structurée comme suit :

$$y_i \rightarrow \bigwedge_{j=1}^{\|\mathcal{C}'\|} \mathbf{c}'_j, \quad \mathcal{C}' \subseteq \mathcal{C}$$

Exemple : Les deux images de la classe *Indigo Bunting* présentées dans la Figure 8.1 (en haut à gauche et en bas à gauche) satisfont la règle suivante :

$$y_0 \rightarrow \mathbf{c}_{\text{eye\_color:black}} \wedge \mathbf{c}_{\text{wing\_shape:pointed}} \wedge \mathbf{c}_{\text{leg\_color:black}} \wedge \mathbf{c}_{\text{tail\_shape:notched}}$$

où y_0 représente la classe *Indigo Bunting*. Alternativement, une règle de classe peut également prendre la forme suivante :

$$\bigwedge_{j=1}^{\|\mathcal{C}'\|} \mathbf{c}'_j \rightarrow y_i, \quad \mathcal{C}' \subseteq \mathcal{C}$$

Exemple : En prenant en compte les images et leurs classes correspondantes présentées dans la Figure 8.1, on peut en dériver la règle suivante :

$$\mathbf{c}_{\text{leg\_color:black}} \bigwedge \mathbf{c}_{\text{tail\_shape:notched}} \rightarrow y_0$$

8.2.2 Règles de concept

Les règles de concept représentent des règles d'association sur les images. Elles sont générées à l'aide d'un algorithme de fouille de règles d'association fermées basé sur les concepts (Szathmary 2006a). Une règle de concept est structurée comme suit :

$$\bigwedge_{i=1}^{\|\mathcal{C}'\|} \mathbf{c}'_i \rightarrow \bigwedge_{j=1}^{\|\mathcal{C}''\|} \mathbf{c}''_j$$

où $\mathcal{C}' \subseteq \mathcal{C}$ et $\mathcal{C}'' \subseteq \mathcal{C}$.

Exemple : En considérant les images et les classes correspondantes présentées dans la Figure 8.1, elles satisfont la règle suivante :

$$\mathbf{c}_{\text{wing\_color:black}} \rightarrow \mathbf{c}_{\text{eye\_color:black}} \bigwedge \mathbf{c}_{\text{wing\_shape:pointed}}$$

Chaque règle de concept est associée à un support et à un niveau de confiance. La confiance d'une règle $A \rightarrow B$ dans un jeu de données $\mathcal{D}$ est définie comme le nombre d'images contenant les concepts de $A \cup B$ divisé par le nombre d'images contenant les concepts de A. Comme les règles de classe, les règles de concept peuvent être directement disponibles avec le jeu de données. Si ce n'est pas le cas, elles sont générées à l'aide de la méthode décrite dans le Chapitre 6.

8.3 Modèle proposé

Le modèle que nous proposons, que nous appellerons CRM (*Concept-Based Reasoning Model*), est formé de deux principaux composants : l'*embedding generator* (représenté par la boîte en pointillés bleue dans la Figure 8.2) et le *concept reasoner* (représenté par la boîte en pointillés verte dans la Figure 8.2). Le rôle de l'*embedding generator* est de prendre en entrée l'image et de produire un ensemble de représentations vectorielles, chacune correspondant à un concept. En parallèle, le *concept reasoner* utilise ces représentations comme entrée afin de produire la classification finale, en intégrant les connaissances disponibles pour affiner la prédiction.

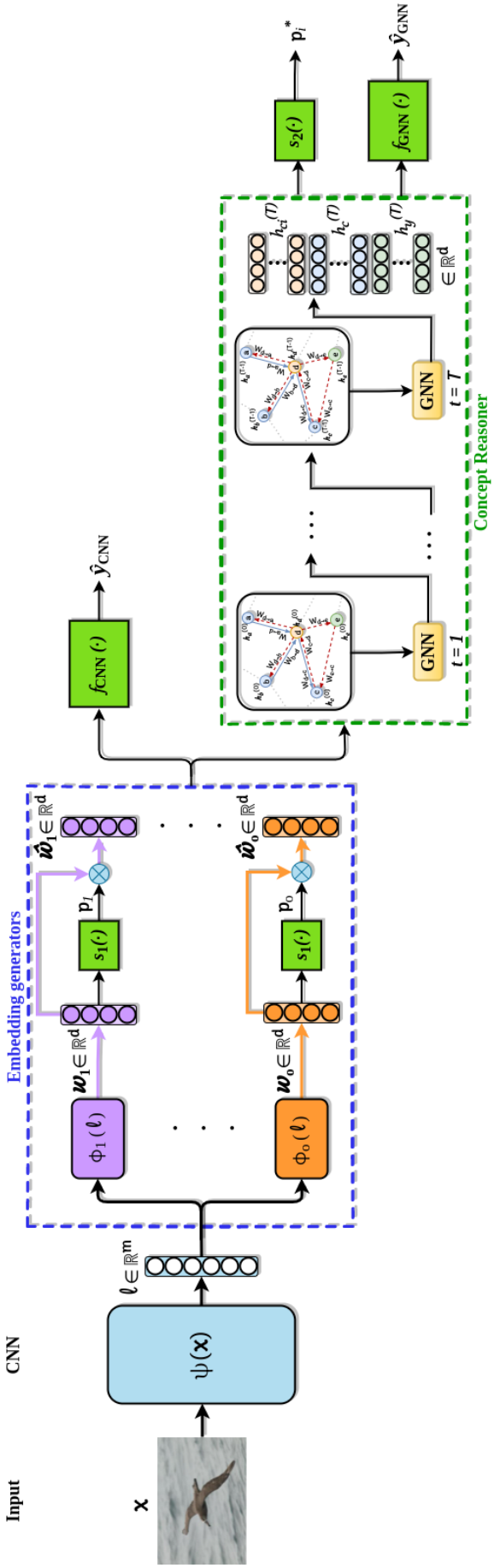


FIGURE 8.2 – Illustration générale du *Concept-Based Reasoning Model* (CRM). Il est composé de deux principales parties. Tout d'abord, **l'embedding generator** (boîte en pointillés bleue) génère une représentation pour chaque concept à partir de la représentation latente d'une image ℓ, obtenue via le CNN ψ. Ensuite, le **concept reasoner** (boîte en pointillés verte) exploite ces nouvelles représentations et applique un *message passing* pour corriger les incohérences dans les prédictions des concepts. Ce processus repose sur un graphe construit à partir de règles, qui capture les interdépendances entre concepts, permettant ainsi d'affiner et d'améliorer la précision des prédictions finales.

Pour chaque image en entrée, CRM utilise un CNN ψ afin de produire une représentation latente ℓ, qui est ensuite transmise à l'*embedding generator*. L'*embedding generator* applique une fonction ϕ_i spécifique à chaque concept, prenant la représentation latente ℓ comme entrée et générant une représentation $\hat{w}_i$, contenant des informations sur la présence (ou l'absence) du concept dans l'image. Ces représentations nouvellement apprises sont ensuite transmises au *concept reasoner*, qui fonctionne sur un graphe construit à partir de règles. Ce raisonneur applique un *message passing* entre les nœuds de ce graphe en utilisant des principes de logique floue pour prédire la présence de chaque concept dans l'image, corrigeant ainsi d'éventuelles erreurs de prédiction. Enfin, deux fonctions de perte distinctes sont optimisées : $f_{\text{CNN}}(\cdot)$, qui utilise les représentations de l'*embedding generator* pour évaluer la performance du modèle sans tenir compte des connaissances préalables, et $f_{\text{GNN}}(\cdot)$, qui utilise les représentations du *concept reasoner* pour évaluer la classification tout en intégrant les connaissances préalables. De plus, CRM prend en charge l'intervention humaine en phase de test, introduite par Koh et al. (2020), permettant aux experts d'améliorer les performances du modèle en corrigeant les concepts mal prédits.

8.3.1 L'*embedding generator*

Chaque image x est fournie en entrée au CNN ψ, qui apprend une représentation latente $\ell \in \mathbb{R}^m$. Cette représentation ℓ est ensuite transmise à l'*embedding generator* constitué de fonctions ϕ_i, qui calculent une représentation $\phi_1(\ell), \ldots, \phi_o(\ell)$ pour chaque concept. $\phi_i(\ell) \in \mathbb{R}^d$ est implémentée via des couches entièrement connectées spécifiques aux concepts, définies comme :

$$\phi_i(\ell) = a(W_i \ell + b_i), \tag{8.1}$$

où a [1] est une fonction d'activation, et W_i et b_i sont des paramètres d'apprentissage. Le vecteur $\phi_i(\ell)$ représente l'état (actif ou inactif) du concept i dans l'image d'entrée.

Notre architecture encourage l'alignement du plongement $\phi_i(\ell)$ avec la vérité terrain $\hat{\mathbf{c}}_i$ en utilisant une fonction de score $s_1 : \mathbb{R}^d \to [0, 1]$. La fonction s_1 est entraînée pour prédire la probabilité que le concept $\mathbf{c}_i$ soit actif dans l'espace de représentation :

$$p_i(\ell) = \sigma(W_{s_1} \phi_i(\ell) + b_{s_1}) \tag{8.2}$$

où W_{s_1} et b_{s_1} sont des paramètres partagés par tous les concepts. Une fois $\phi_i(\ell)$ calculé, la représentation finale $\hat{w}_i \in \mathbb{R}^d$ du concept est obtenue par :

$$\hat{w}_i(\ell) = p_i \times \phi_i(\ell) \tag{8.3}$$

L'équation 8.3 remplit deux rôles : (1) comme $\phi_i(\ell)$ contient les informations sur l'activation (présence ou absence) du concept $\mathbf{c}_i$ dans l'image, cette équation permet au modèle d'intégrer cette pertinence dans la représentation, et (2) elle facilite la mise en œuvre de la stratégie d'intervention, discutée dans la Section 8.3.5.

1. En pratique, nous utilisons une fonction d'activation ReLU pour a.

8.3.2 *Concept reasoner*

Une fois les plongements des concepts obtenus, les règles sont appliquées pour raisonner sur la présence de ces concepts dans l'image. Par exemple, si le concept $\mathbf{c}_{\text{wing\_color:black}}$ est détecté dans l'image, alors la règle $\mathbf{c}_{\text{wing\_color:black}} \rightarrow \mathbf{c}_{\text{eye\_color:black}} \wedge \mathbf{c}_{\text{wing\_shape:pointed}}$ permet d'inférer que les concepts $\mathbf{c}_{\text{eye\_color:black}}$ et $\mathbf{c}_{\text{wing\_shape:pointed}}$ sont également présents dans l'image. Ce processus de raisonnement est implémenté à l'aide d'un GCN combiné aux principes de la logique floue. Dans un premier temps, nous expliquerons comment nous construisons le graphe à partir des règles. Ensuite, nous détaillerons comment le raisonnement au sein du graphe permet de corriger d'éventuelles erreurs relatives à la présence des concepts dans une image commises par l'*embedding generator*.

Construction du graphe de connaissances

Chaque concept et chaque classe sont représentés sous forme de nœuds dans le graphe, avec l'ajout de nœuds intermédiaires pour représenter les conjonctions de concepts. Pour illustrer la construction du graphe à partir des règles, prenons l'exemple d'un scénario de classification d'images inspiré du jeu de données CelebA, où chaque image contient un ou plusieurs concepts parmi les suivants : { male, young, smiling, attractive, noBeard, makeup, bangs }. Nous nous baserons sur les règles de classe et les règles de concept suivantes associées à ce jeu de données :

$$y_1 \rightarrow \mathbf{c}_{\text{male}} \wedge \mathbf{c}_{\text{young}} \wedge \mathbf{c}_{\text{smiling}} \wedge \mathbf{c}_{\text{attractive}}$$
$$\mathbf{c}_{\text{male}} \wedge \mathbf{c}_{\text{attractive}} \wedge \mathbf{c}_{\text{smiling}} \rightarrow y_1$$
$$\mathbf{c}_{\text{young}} \wedge \mathbf{c}_{\text{smiling}} \rightarrow \mathbf{c}_{\text{attractive}}$$
$$\mathbf{c}_{\text{attractive}} \wedge \mathbf{c}_{\text{noBeard}} \rightarrow \mathbf{c}_{\text{makeup}} \wedge \mathbf{c}_{\text{bangs}}$$
$$\mathbf{c}_{\text{makeup}} \rightarrow \mathbf{c}_{\text{young}}$$

Le graphe est généré en suivant les étapes suivantes :

Étape 1 Lorsqu'une conjonction de concepts est présente dans une règle, une règle d'équivalence est créée entre cette conjonction de concepts et un nouveau concept intermédiaire.

Exemple :

$$\mathbf{c}_{\text{young}} \wedge \mathbf{c}_{\text{smiling}} \longleftrightarrow A$$
$$\mathbf{c}_{\text{attractive}} \wedge \mathbf{c}_{\text{noBeard}} \longleftrightarrow B$$
$$\mathbf{c}_{\text{makeup}} \wedge \mathbf{c}_{\text{bangs}} \longleftrightarrow C$$
$$\mathbf{c}_{\text{male}} \wedge \mathbf{c}_{\text{attractive}} \wedge \mathbf{c}_{\text{smiling}} \longleftrightarrow D$$
$$\mathbf{c}_{\text{male}} \wedge \mathbf{c}_{\text{young}} \wedge \mathbf{c}_{\text{smiling}} \wedge \mathbf{c}_{\text{attractive}} \longleftrightarrow E$$

A, B, C, D and E sont des concepts intermédiaires.

Étape 2 Chaque concept, y compris les concepts intermédiaires, ainsi que chaque classe, sont représentés par un nœud dans le graphe.

Étape 3 Chaque règle d'équivalence nouvellement créée est transformée en règles d'implication. L'ensemble des règles de classe et de concept est ensuite mis à jour en tenant compte des concepts intermédiaires nouvellement introduits.

Exemple : Les règles mises à jour avec les concepts intermédiaires sont :

$$\mathbf{c}_{young} \wedge \mathbf{c}_{smiling} \rightarrow A$$

$$A \rightarrow \mathbf{c}_{young}$$

$$A \rightarrow \mathbf{c}_{smiling}$$

$$\mathbf{c}_{attractive} \wedge \mathbf{c}_{noBeard} \rightarrow B$$

$$B \rightarrow \mathbf{c}_{attractive}$$

$$B \rightarrow \mathbf{c}_{noBeard}$$

$$\mathbf{c}_{makeup} \wedge \mathbf{c}_{bangs} \rightarrow C$$

$$C \rightarrow \mathbf{c}_{makeup}$$

$$C \rightarrow \mathbf{c}_{bangs}$$

$$\mathbf{c}_{male} \wedge \mathbf{c}_{attractive} \wedge \mathbf{c}_{smiling} \rightarrow D$$

$$D \rightarrow \mathbf{c}_{male}$$

$$D \rightarrow \mathbf{c}_{attractive}$$

$$D \rightarrow \mathbf{c}_{smiling}$$

$$A \wedge \mathbf{c}_{male} \wedge \mathbf{c}_{attractive} \rightarrow E$$

$$E \rightarrow A$$

$$E \rightarrow \mathbf{c}_{male}$$

$$E \rightarrow \mathbf{c}_{attractive}$$

$$y_1 \rightarrow E$$

$$D \rightarrow y_1$$

$$A \rightarrow \mathbf{c}_{attractive}$$

$$B \rightarrow \mathbf{c}_{young}$$

$$B \rightarrow C$$

$$\mathbf{c}_{makeup} \rightarrow \mathbf{c}_{young}$$

Étape 4 Des arcs étiquetés AND (ligne bleue continue) et des arcs étiquetés OR (ligne rouge en pointillés) sont générés à partir de toutes les règles d'implication, selon les scénarios suivants :

- Lorsque la prémisse est une conjonction de concepts (ou de concepts intermédiaires) et que la conclusion est un concept intermédiaire, des arcs AND sont créés entre chaque concept de la prémisse et la conclusion. Ainsi, les arcs AND pointent toujours vers des concepts intermédiaires.

  Exemple : $\mathbf{c}_{\texttt{young}} \wedge \mathbf{c}_{\texttt{smiling}} \rightarrow A$. Cela signifie que le concept intermédiaire A est présent dans l'image si les concepts $\texttt{young}$ et $\texttt{smiling}$ sont tous deux détectés. Des arcs AND sont alors ajoutés : $(\mathbf{c}_{\texttt{young}}, A)$ et $(\mathbf{c}_{\texttt{smiling}}, A)$.

- Lorsque la prémisse est un atome (concept, concept intermédiaire ou classe) et que la conclusion est un atome (concept, concept intermédiaire ou classe), un arc de type OR est créé de la prémisse vers la conclusion.

  Exemple : $A \rightarrow \mathbf{c}_{\texttt{young}}$, $\mathbf{c}_{\texttt{makeup}} \rightarrow \mathbf{c}_{\texttt{young}}$, $D \rightarrow y_1$. Un arc OR est ajouté pour chacune de ces règles.

La Figure 8.3 illustre le graphe généré à partir des règles de l'exemple. Les nœuds de concepts sont représentés en couleur bleue, les nœuds de concepts intermédiaires en couleur orange, et les nœuds de classes en couleur verte.

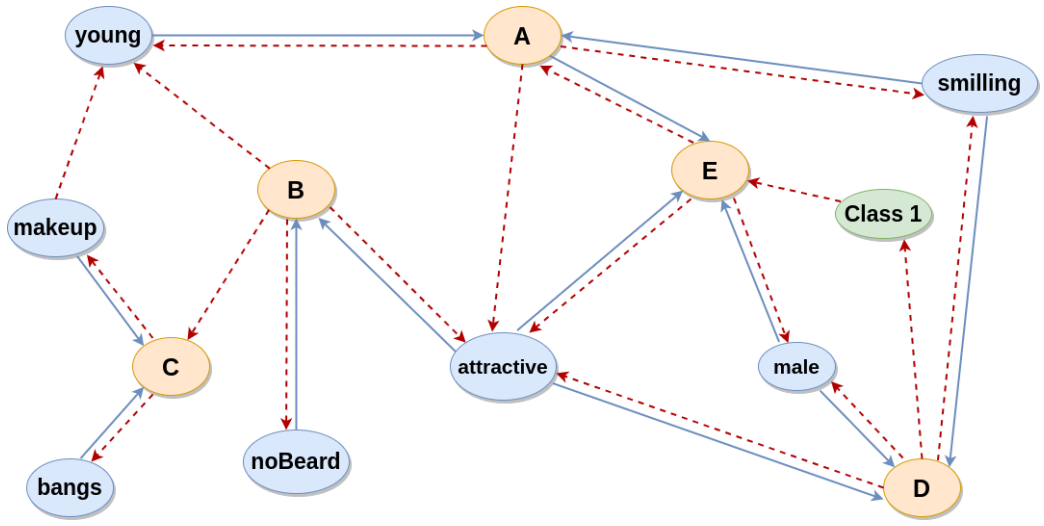


FIGURE 8.3 – Exemple de graphe généré à partir des règles. Les nœuds de concepts sont en bleu, les nœuds de concepts intermédiaires en orange, et les nœuds de classes en vert. Les arcs étiquetés AND sont représentés par des lignes bleues continues, tandis que les arcs étiquetés OR sont représentés par des lignes rouges en pointillés.

Raisonnement sur le graphe de connaissances

Graphe Soit $\mathcal{G}$ un graphe orienté représenté par $\mathcal{G} = (V, E)$, où $V = \{v_1, \ldots, v_{|V|}\}$ est un ensemble de nœuds, et $E = \{e_1, \ldots, e_{|E|}\}$ représente un ensemble d'arcs. Chaque arc e_{ij} est représenté par une paire de nœuds (v_i, v_j) avec $v_i, v_j \in V$. Il existe deux types d'arcs :

(i) les arcs de type AND et nous noterons E^1 l'ensemble de ces arcs (ligne bleue continue dans la Figure 8.3) et (ii) les arcs de type OR et nous noterons E^2 l'ensemble de ces arcs (ligne rouge en pointillés dans la Figure 8.3). De plus, on distingue trois types de nœuds : les nœuds de concepts (nœuds bleus), les nœuds de concepts intermédiaires (nœuds oranges) et les nœuds de classe (nœuds verts).

Voisinage Soit $\mathcal{N}^1(v_1) = \{v_2 \in V \mid (v_2, v_1) \in E^1\}$ l'ensemble des voisins de v_1 connectés par un arc entrant AND.

Soit $\mathcal{N}^2(v_1) = \{v_2 \in V \mid (v_2, v_1) \in E^2\}$ l'ensemble des voisins de v_1 connectés par un arc entrant OR.

Soit $\mathcal{N}(v_1) = \mathcal{N}^1(v_1) \cup \mathcal{N}^2(v_1)$ l'ensemble de tous les voisins de v_1, connectés par des arcs entrant AND ou OR.

Passage de messages dans le graphe

Étant donné un graphe $\mathcal{G} = (V, E)$ et les plongements des nœuds de concepts $\hat{w}_i(\ell)$, l'objectif est de prédire l'état de chaque nœud de concept dans une image en s'appuyant sur la topologie du graphe. Pour cela, nous utilisons un algorithme neuronal GCN de propagation de messages afin de diffuser l'information de chaque nœud vers ses nœuds adjacents. En commençant par un état de départ $h_v^{(0)}$, chaque nœud v reçoit, pendant T itérations, un message provenant de chacun de ses voisins. Les représentations initiales des nœuds $h_v^{(0)} \in \mathbb{R}^d$ dépendent du type de nœud :

- $h_v^{(0)} = \hat{w}_i(\ell)$ si v est un nœud de concept $\mathbf{c}_i$
- $h_v^{(0)} = [0_d]$ si v n'est pas un nœud de concept (nœud de concepts intermédiaires ou nœud de classe)

À chaque itération t, le message envoyé du nœud v_j au nœud v_i est défini par :

$$M_{j\to i}^{(t)} = \begin{cases} h_{v_j}^{(t)} & \text{if } (v_j, v_i) \in E^1 \\ h_{v_j}^{(t)} \times W_{j\to i} & \text{if } (v_j, v_i) \in E^2 \end{cases} \tag{8.4}$$

où $W_{j\to i} \in \mathbb{R}$ est le poids de l'arc reliant le nœud v_j au nœud v_i. Dans nos expérimentations, nous avons initialisé $W_{j\to i} = 0$. Ces poids permettent au modèle de sélectionner les arcs de type OR les plus pertinents afin d'améliorer son processus de raisonnement. Après la propagation des messages, chaque nœud v_i met à jour son propre état en fonction des messages reçus agrégés, en appliquant les principes de la logique floue :

$$h_{v_i}^{(t+1)} = \max(h_{v_i}^{(t)}, \max_{j\in\mathcal{N}^2(v_i)} M_{j\to i}^{(t)}, \min_{j\in\mathcal{N}^1(v_i)} M_{j\to i}^{(t)}) \tag{8.5}$$

Parce que tous les nœuds voisins (concepts) du nœud v_i connectés par un arc AND doivent être présents dans l'image d'entrée pour que le(s) concept(s) associé(s) au nœud

v_i soit également présent(s), nous appliquons l'opération minimum sur ses voisins $\mathcal{N}^1(v_i)$. Cela garantit que le nœud v_i devient actif uniquement si tous ses voisins $\mathcal{N}^1(v_i)$ sont actifs. De manière similaire, si au moins un des voisins $\mathcal{N}^2(v_i)$ de v_i connectés par un arc de type OR est actif, alors v_i devient actif en prenant le maximum sur ses voisins $\mathcal{N}^2(v_i)$. Enfin, nous nous assurons que le nœud v_i reste actif s'il l'était déjà avant la mise à jour, indépendamment des états de ses voisins $\mathcal{N}(v_i)$.

Prenons comme exemple les règles présentées dans la Section 8.3.2. Supposons qu'une image d'entrée active uniquement les concepts $\mathtt{male}$, $\mathtt{young}$ et $\mathtt{smiling}$. Dans ce cas, la règle $\mathbf{c}_{\mathtt{young}} \wedge \mathbf{c}_{\mathtt{smiling}} \rightarrow \mathbf{c}_{\mathtt{attractive}}$ activerait également le concept $\mathtt{attractive}$. Grâce à cette inférence, le modèle pourrait non seulement améliorer ses performances en termes de reconnaissance des concepts, mais aussi améliorer ses résultats sur la tâche principale. Plus précisément, la règle $\mathbf{c}_{\mathtt{male}} \wedge \mathbf{c}_{\mathtt{attractive}} \wedge \mathbf{c}_{\mathtt{smiling}} \rightarrow y_1$ permettrait d'associer l'image à la classe correcte y_1, illustrant ainsi comment les règles contribuent à la compréhension des concepts et à l'exactitude de la classification. De plus, l'état mis à jour des concepts, obtenu après le *concept reasoner*, est ensuite exploité pour prédire la classe de l'image d'entrée en utilisant la fonction de perte introduite dans la Section 8.3.4. Le gradient résultant de ce processus d'optimisation est rétropropagé dans tout le modèle, garantissant que les représentations des concepts ainsi que la classification finale bénéficient des dépendances inférées, conduisant ainsi à des prédictions plus cohérentes et robustes. Comme démontré dans la section expérimentale, CRM obtient des performances significativement supérieures sur ce jeu de données CelebA lorsque le module de raisonnement est utilisé.

Après T itérations, l'état mis à jour de chaque nœud concept $h_{c_1}^{(T)}, \ldots, h_{c_o}^{(T)}$, de chaque nœud de classe $h_{y_1}^{(T)}, \ldots, h_{y_k}^{(T)}$, ainsi que de chaque nœud concept intermédiaire $h_{ci_1}^{(T)}, \ldots, h_{ci_{|V|-(o+k)}}^{(T)}$ contient des informations sur l'état du concept associé au nœud dans l'image d'entrée. Tous les nœuds mis à jour dépendent de ℓ, mais pour plus de simplicité, nous omettons ℓ dans la notation de cette section.

Chaque état mis à jour d'un nœud concept est encouragé à être aligné avec le concept de vérité terrain $\hat{\mathbf{c}}_i$ en utilisant la fonction de score $s_2 : \mathbb{R}^d \rightarrow [0, 1]$, où s_2 est entraînée pour prédire la probabilité que le concept $\mathbf{c}_i$ soit actif à partir de l'espace des états des nœuds :

$$p_i^*(\ell) = \sigma(W_{s_2} h_{c_i}^{(T)} + b_{s_2}) \tag{8.6}$$

où les paramètres d'apprentissage W_{s_2} et b_{s_2} sont partagés entre tous les concepts.

8.3.3 Prédiction de la tâche

La tâche finale est apprise à partir des plongements de concepts obtenus à la fois par l'*embedding generator* et le *concept reasoner*. Tous les plongements de concepts $\hat{w}_i$ issus de l'*embedding generator* sont concaténés puis passés dans une couche linéaire f_{CNN} afin d'obtenir une distribution de probabilité pour chaque classe $\hat{y}_{\mathrm{CNN}}$:

$$\hat{y}_{\mathrm{CNN}}(\ell) = f_{\mathrm{CNN}}(\hat{w}(\ell)) \tag{8.7}$$

où $\widehat{w(\ell)} = [\hat{w}_1(\ell), \ldots, \hat{w}_o(\ell)]$ représente la concaténation des plongements de concepts issus de l'*embedding generator*.

De manière similaire, nous concaténons les plongements de concepts $h_{c_i}^{(T)}$ obtenus à partir de l'*embedding generator* ainsi que les k plongements de classes $h_{y_i}^{(T)}$ issus du *concept reasoner* afin d'entraîner un prédicteur de classe f_{GNN}, permettant d'obtenir la distribution des probabilités de classe $\hat{y}_{GNN}$ pour l'image en entrée :

$$\hat{y}_{GNN}(\ell) = f_{GNN}([h_{c_1}^{(T)}, \ldots h_{c_o}^{(T)}, h_{y_1}^{(T)}, \ldots, h_{y_k}^{(T)}]) \tag{8.8}$$

où $[h_{c_1}^{(T)}, \ldots h_{c_o}^{(T)}, h_{y_1}^{(T)}, \ldots, h_{y_k}^{(T)}]$ représente la concaténation des plongements de concepts et de classes issus de l'*embedding generator*.

Dans la section expérimentale, nous utilisons les sorties de l'*embedding generator* et du *concept reasoner* afin d'évaluer l'efficacité du raisonnement basé sur les règles.

8.3.4 Fonction de perte

CRM est entraîné pour minimiser une somme pondérée de deux fonctions de perte : l'une pour la prédiction des concepts et l'autre pour la prédiction de la tâche. La perte associée à la prédiction des concepts pour la i-ième image d'entrée est :

$$\mathcal{L}_{concept} = \mathcal{L}_1(c_i, \hat{p}(\ell)) + \mathcal{L}_2(c_i, \hat{p}^*(\ell)) \tag{8.9}$$

où $\hat{p}(\ell) = [p_1, ..., p_o]$ est un vecteur de probabilités des concepts obtenu après l'*embedding generator* pour la i-ième image, et $\hat{P}^*(\ell) = [p_1^*, ..., p_o^*]$ est un vecteur de probabilités des concepts obtenu après le *concept reasoner* pour la i-ième image. $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des entropies croisées binaires. De même, la perte associée à la prédiction de la tâche pour la i-ième image d'entrée est :

$$\mathcal{L}_{Tâche} = \mathcal{L}_3(y_i, \hat{y}_{CNN}) + \mathcal{L}_4(y_i, \hat{y}_{GNN}) \tag{8.10}$$

où $\mathcal{L}_3$ et $\mathcal{L}_4$ sont des entropies croisées. La perte finale est donnée par :

$$\mathcal{L} = \alpha_1 \mathcal{L}_{concept} + \alpha_2 \mathcal{L}_{Tâche} \tag{8.11}$$

8.3.5 Intervention sur les concepts

CRM prend en charge les interventions sur les concepts au moment du test. Pour intervenir sur un concept $\mathbf{c}_i$, $\hat{w}_i$ est mis à jour en remplaçant la prédiction de sortie p_i par le label de vérité terrain $\hat{\mathbf{c}}_i$.

L'avantage de notre modèle CRM est que les interventions peuvent influencer non seulement le concept ciblé $\hat{w}_i$, mais aussi mettre à jour l'état d'autres concepts associés grâce au *concept reasoner*.

À l'instar de CEM (Zarlenga et al. 2022), nous utilisons une stratégie de régularisation *RandInt*, qui permet d'effectuer des interventions sur les concepts pendant l'entraînement afin d'améliorer l'efficacité du modèle final. *RandInt* réalise aléatoirement des interventions indépendantes sur les concepts pendant l'entraînement avec une probabilité p_{int} (c'est-à-dire que p_i est remplacé par $\hat{\mathbf{c}}_i$ pour le concept $\mathbf{c}_i$ avec une probabilité p_{int}). En d'autres termes, pour tous les concepts $\mathbf{c}_i$, le plongement $\hat{w}_i$ est calculé pendant l'entraînement comme suit :

$$\hat{w}_i = \begin{cases} (\hat{\mathbf{c}}_i \phi_i) \text{ avec la probabilité } p_{int} \\ (p_i \phi_i) \text{ avec la probabilité } (1 - p_{int}) \end{cases} \tag{8.12}$$

tandis qu'au moment du test, nous utilisons toujours les probabilités prédites p_i pour calculer $\hat{w}_i$.

8.4 Protocoles Expérimentaux

Dans cette section, nous présentons les jeux de données utilisés dans nos expériences et décrivons notre cadre expérimental. Notre objectif est de démontrer empiriquement que notre méthode atteint des performances comparables aux meilleurs modèles actuels basés sur des concepts tout en maintenant un haut niveau d'explicabilité. Nous montrons également que l'utilisation du *concept reasoner* améliore les performances du modèle sur les deux sorties de classification (avant et après le *concept reasoner*).

8.4.1 Jeux de données

Pour évaluer notre méthode, nous avons utilisé les jeux de données Synthetic, COCO@11, COCO@24, COCO@48, CelebA, CUB-Simp, VG12, *SunIn* et *SunOut*. Concernant les jeux de données Synthetic, CelebA, VG12 et COCO@11, l'ensemble des règles de classe et de concepts disponibles a été considéré, comme présenté dans la Section 6.4. Pour les jeux de données COCO@24 et VG48, nous avons retenu toutes les règles de concepts ainsi que les règles de classe ayant une confiance minimale de 0.08 et 0.1 sur les données d'apprentissage, respectivement pour COCO@24 et VG48. Nous obtenons ainsi un total de 63 règles de classe pour COCO@24 et 108 règles de classe pour VG48. Pour les jeux de données *SunIn* et *SunOut*, nous avons également utilisé l'intégralité des règles de concepts, tout en sélectionnant uniquement les règles de classe présentant une confiance minimale de 0.2, aboutissant respectivement à 132 règles pour *SunIn* et 183 règles pour *SunOut*. Enfin, pour le jeu de données CUB-Simp, l'ensemble des règles de concepts a été utilisé, ainsi que les règles de classe ayant une confiance égale à 1 et contenant au maximum deux atomes dans la prémisse pour les règles du type $\bigwedge_{j=1}^{\|\mathcal{C}'\|} \mathbf{c}'_j \rightarrow y_i$, totalisant ainsi 413 règles de classe. Un récapitulatif des propriétés de chaque jeu de données est présenté dans la Table 8.1.

Jeu de données	Train	Test	Entrée	Concepts	Groupes	Classes	Règles classe	Règles concepts
Synthetic	236	199	[3, 299, 299]	16	4	8	157	33
CUB-Simp	5 994	5,794	[3, 299, 299]	112	28	200	413	598
COCO@11	1 692	771	[3, 299, 299]	24	24	11	487	650
COCO@24	5 879	2 722	[3, 299, 299]	36	36	24	63	36
COCO@48	17 278	8 190	[3, 299, 299]	59	59	48	99	17
VG12	830	277	[3, 299, 299]	17	17	12	24	221
VG48	7 872	3 374	[3, 299, 299]	480	480	48	108	101
SunIn	1 092	468	[3, 299, 299]	68	68	78	132	125
CelebA	13 507	3 376	[3, 64, 64]	6	6	256	183	55

TABLE 8.1 – Détails sur tous les jeux de données utilisés pour expérimenter CRM.

8.4.2 Détails de l'entraînement

Pour l'ensemble des jeux de données, nous utilisons un modèle ResNet-34 pré-entraîné (He et al. 2016) comme architecture CNN ψ, pour laquelle nous avons modifié la dernière couche pour produire 512 activations. Nous utilisons $d = 16$ activations pour l'apprentissage des plongements. Afin de garantir une comparaison équitable, nous utilisons une architecture identique pour tous les modèles. Un réseau de neurones entièrement connecté est utilisé pour les prédicteurs de classes f_{CNN} et f_{GNN}. Lorsque nous appliquons la régularisation *RandInt*, nous fixons $p_{\text{int}} = 0.25$, comme utilisé dans Espinosa Zarlenga et al. (2022).

Le choix des paramètres α_1 et α_2 sont définis en s'inspirant des travaux de Espinosa Zarlenga et al. (2022). Dans les jeux de données Synthetic, CUB-Simp, COCO@11, COCO@24 et COCO@48, nous fixons le poids de la perte associée aux concepts à $\alpha_1 = 3$ et celui de la perte de classification à $\alpha_2 = 1$. Nous maintenons le poids de la perte des concepts à $\alpha = 5$ pour toutes les méthodes concurrentes et utilisons une perte d'entropie croisée pondérée afin de corriger les déséquilibres dans les annotations des concepts. L'optimisation est effectuée avec l'algorithme SGD, en utilisant un momentum de 0.9 et un taux d'apprentissage initial de 10^{-2}. Les modèles sont entraînés pendant 300 époques avec une taille de lot de 96 pour Synthetic, 256 pour CUB-Simp et 192 pour COCO@11, COCO@24 et COCO@48.

Pour la tâche CelebA, nous appliquons les mêmes réglages que dans Espinosa Zarlenga et al. (2022), en fixant le poids des pertes à $\alpha_1 = \alpha_2 = 1$ et en utilisant une perte d'entropie croisée pondérée pour atténuer les déséquilibres dans les annotations des concepts. Les modèles sont entraînés sur 200 époques avec une taille de lot de 512 et un optimiseur SGD avec un momentum de 0.9 et un taux d'apprentissage de 5×10^{-3}.

Dans toutes les expériences, nous appliquons un facteur de décroissance de $4e-05$ et ajustons dynamiquement le taux d'apprentissage en le réduisant d'un facteur de 0.1 si aucune amélioration de la perte de validation n'est observée après 10 époques. Pour les jeux de données CUB-Simp et CelebA, un mécanisme d'arrêt anticipé est utilisé : l'entraînement est interrompu si la perte de validation ne s'améliore pas sur une période de 15 époques consécutives.

8.4.3 Méthodes comparatives

Nous comparons CRM à des modèles basés sur des concepts, qui permettent des interventions sur les concepts lors de la phase de test. Parmi ces modèles, nous incluons CEM (Zarlenga et al. 2022), ProbCBM (Kim et al. 2023), IntCEM (Espinosa Zarlenga et al. 2023) avec $\lambda_{\text{roll}} = 1$, ainsi que CBM (Koh et al. 2020), tous présentés dans la Section 5.2. Nous comparons CRM avec la version jointe de CBM qui apprend à la fois à prédire les concepts et les classes. CBM joint est utilisé sous différentes variantes : celle avec un goulet d'étranglement sigmoïdal (Joint CBM-Sigmoid) et celle basée sur les logits (Joint CBM-Logit). Dans nos expériences, nous effectuons un apprentissage de CRM sans utiliser le concet reasoner, et nous appelons ce modèle CRM-W. Aussi, nous entraînons CRM en utilisant le *concept reasoner*, et nous évaluons les sorties de l'*embedding generator* (avant le raisonnement sur les concepts avec les règles) et du *concept reasoner* (après le raisonnement sur les concepts). Nous appelons la sortie de l'*embedding generator* CRM-CNN et nous appelons la sortie du *concept reasoner* CRM-GNN. Toutes les métriques d'intérêt (précision sur la tâche et précision sur les concepts) ont été moyennées sur cinq initialisations aléatoires différentes pour chaque jeu de données et chaque méthode. Afin d'assurer une comparaison équitable, nous avons utilisé la même architecture pour l'encodeur de concepts et le prédicteur de classes dans toutes les méthodes de référence. Lors de l'entraînement, nous avons fixé la probabilité d'intervention *RandInt* à $p_{\text{int}} = 0.25$. Contrairement aux autres méthodes qui utilisent Leaky-ReLU comme fonction d'activation non supervisée dans les générateurs de plongements, nous avons opté pour ReLU afin de garantir que la valeur minimale d'activation soit 0.

Enfin, nous avons évalué différentes politiques d'intervention (Random, CooP (Chauhan et al. 2023), UCP (Shin et al. 2022), CVA (Koh et al. 2020), CVI (Chauhan et al. 2023), Skyline) afin d'analyser les performances de notre méthode lors de la phase de test, en comparaison aux variantes les plus avancées de l'état de l'art.

8.5 Résultats et discussions

8.5.1 Performances sur la tâche et les concepts

Performance sur la tâche

Les performances sur la tâche sont présentés dans la première moitié de la Table 8.2, ainsi que sur l'axe des abscisses de la Figure 8.4. Les résultats expérimentaux indiquent que la méthode CRM offre une précision de classification compétitive, surpassant ou égalant les approches concurrentes sur l'ensemble des jeux de données évalués. Malgré une variabilité des performances selon les jeux de données, CRM maintient une compétitivité constante dans la plupart des scénarios. Sur les jeux de données Synthetic, COCO@24, CelebA et VG12, CRM surpasse nettement les autres méthodes, avec des écarts parfois significatifs (1.59% sur CelebA, 1.1% sur VG12, 0.3% sur Synthetic et 0.24% sur COCO@24). Sur les jeux de données COCO@11, COCO@48 et CUB-Simp, CRM atteint la deuxième meilleure performance avec une différence par rapport aux meilleures performances de 0.21% sur

CUB-Simp, 0.16% sur COCO@11 et 0.02% sur COCO@48. Il est intéressant de remarquer que les jeux de données COCO@11 et CUB-Simp ont un total de règles (règles de classe et règles de concepts) élevé par rapport aux autres. Particulièrement, sur les jeux de données SunIn et SunOut, CRM obtient des performances très inférieures par rapport au autres méthodes. Cette disparité peut s'expliquer par le fait que les concepts disponibles dans ces jeux de données ne permettent pas de séparer les classes, tel que présenté dans la Figure 8.3. Sur le jeu de données CelebA où le modèle ne dispose pas d'un ensemble complet d'annotations sur les concepts, CRM se distingue par de meilleures résultats par rapport aux autres approches.

Performance sur les concepts

Les performances sur les concepts sont présentés dans la seconde moitié de la Table 8.2 ainsi que sur l'axe des ordonnées de la Figure 8.4. Nos expériences montrent la compétitivité de CRM par rapport aux méthodes concurrentes. Sur les jeux de données Synthetic, CelebA et VG12, CRM se classe parmi les deux meilleures performances, démontrant la pertinence de l'utilisation des connaissances. Sur les jeux de données COCO@11, COCO@24, COCO@48 et CUB-Simp, CRM affiche une performance légèrement inférieure que les méthodes concurrentes, bien que l'écart ne soit pas significatif. Plus particulièrement, sur CelebA où le modèle ne dispose pas d'un ensemble complet d'annotations sur les concepts, CRM montre une amélioration de performance plus marquée que sur les autres jeux de données.

Étude d'ablation

Il convient de rappeler que CRM-W fait référence à la sortie du générateur de plongements (*embedding generator*) du modèle CRM entraîné sans module de raisonnement sur les concepts. À l'inverse, CRM-CNN et CRM-GNN correspondent respectivement à la sortie du générateur d'embeddings et à celle du module de raisonnement sur les concepts (*concept reasoner*), lorsque le modèle est entraîné de bout en bout avec raisonnement. Comme le montre la Table 8.2, l'ensemble des résultats expérimentaux indique une amélioration systématique des performances lorsque le raisonnement est intégré, en comparaison avec la version sans raisonnement. Ces résultats confirment l'hypothèse selon laquelle l'intégration de règles permet au modèle de bénéficier d'informations supplémentaires pertinentes, facilitant une meilleure interprétation et une prise en compte plus fine des concepts présents dans les images. Par ailleurs, les expériences mettent en évidence l'influence du module de raisonnement (CRM-GNN) sur la qualité des représentations générées par le générateur d'embeddings (CRM-CNN), cette influence étant transmise par rétropropagation des gradients lors de l'entraînement.

	Dataset	CEM	Joint CBM-Sigmoid	Joint CBM-Logit	Prob-CBM	IntCEM($\lambda_{\text{roll}} = 1$)	CRM-W	CRM-CNN	CRM-GNN
Tâche	Synthetic	56.58 ± 2.22	56.38 ± 1.07	56.28 ± 2.93	51.46 ± 1.03	59.90 ± 1.51	56.18 ± 2.09	**60.20 ± 2.16**	59.79 ± 2.12
	COCO@11	51.75 ± 2.23	53.85 ± 2.18	53.49 ± 1.49	**55.06 ± 0.66**	53.59 ± 1.07	54.15 ± 0.56	54.70 ± 1.14	54.90 ± 0.56
	COCO@24	51.14 ± 1.14	51.36 ± 1.62	50.22 ± 2.44	52.10 ± 0.26	51.93 ± 0.76	50.67 ± 2.17	52.15 ± 0.57	**52.34 ± 0.43**
	COCO@48	48.11 ± 1.17	28.36 ± 6.41	34.16 ± 2.63	**51.32 ± 0.30**	50.77 ± 0.50	50.50 ± 0.50	50.35 ± 0.28	51.30 ± 1.23
	CelebA	30.69 ± 0.21	24.15 ± 0.28	23.28 ± 1.05	29.84 ± 0.41	32.43 ± 0.71	31.09 ± 0.63	34.02 ± 0.80	**34.02 ± 0.71**
	CUB-Simp	**78.87 ± 0.49**	73.87 ± 0.97	77.68 ± 0.24	72.99 ± 0.34	77.81 ± 0.40	78.30 ± 0.24	78.00 ± 0.36	78.66 ± 0.52
	VG12	37.55 ± 1.31	41.88 ± 1.50	36.97 ± 1.45	40.29 ± 1.69	38.84 ± 1.22	37.40 ± 1.16	**42.98 ± 2.08**	42.98 ± 2.26
	SunIn	**49.15 ± 1.38**	14.89 ± 8.61	34.97 ± 1.79	39.15 ± 1.24	48.50 ± 2.71	45.94 ± 1.55	47.67 ± 3.50	46.74 ± 2.89
	SunOut	**45.58 ± 1.56**	21.86 ± 2.92	34.59 ± 0.81	36.05 ± 1.96	44.68 ± 2.12	40.02 ± 1.56	43.92 ± 2.83	43.20 ± 1.55
Concept	Synthetic	86.38 ± 0.81	**87.22 ± 0.46**	86.84 ± 0.90	85.68 ± 0.34	86.24 ± 0.46	86.36 ± 0.65	87.05 ± 0.43	86.94 ± 0.58
	COCO@11	87.48 ± 0.48	87.98 ± 0.40	88.03 ± 0.19	**88.09 ± 0.33**	87.59 ± 0.55	87.93 ± 0.20	87.96 ± 0.12	88.02 ± 0.05
	COCO@24	91.90 ± 0.16	91.91 ± 0.36	91.94 ± 0.41	**92.28 ± 0.09**	92.07 ± 0.07	91.73 ± 0.46	92.04 ± 0.05	91.84 ± 0.05
	COCO@48	94.85 ± 0.14	93.65 ± 0.39	93.88 ± 0.15	**95.13 ± 0.01**	95.09 ± 0.01	95.01 ± 0.03	94.91 ± 0.26	94.70 ± 0.23
	CelebA	85.71 ± 0.18	81.36 ± 0.32	82.91 ± 0.48	85.26 ± 0.36	86.39 ± 0.17	85.99 ± 0.22	87.26 ± 0.12	**87.40 ± 0.12**
	CUB-Simp	**96.35 ± 0.03**	95.82 ± 0.08	96.01 ± 0.10	95.57 ± 0.14	95.44 ± 0.09	96.30 ± 0.13	96.26 ± 0.10	96.22 ± 0.21
	VG12	85.94 ± 0.26	86.20 ± 0.19	86.44 ± 0.28	86.33 ± 0.38	**86.74 ± 0.24**	86.17 ± 0.24	86.63 ± 0.25	86.23 ± 0.29
	SunIn	84.45 ± 0.10	85.07 ± 0.21	**86.10 ± 0.09**	85.81 ± 0.10	85.86 ± 0.34	84.39 ± 0.07	85.73 ± 0.36	85.35 ± 0.17
	SunOut	84.36 ± 0.28	85.46 ± 0.15	86.05 ± 0.21	**86.47 ± 0.06**	86.14 ± 0.25	84.24 ± 0.09	85.95 ± 0.22	84.91 ± 0.17

TABLE 8.2 – Performance sur la tâche (accuracy en %) et sur les concepts (accuracy en %) pour tous les jeux de données et méthodes. La première moitié du tableau représente la performance sur la tâche, tandis que la seconde moitié représente la performance sur les concepts. La meilleure performance pour chaque ligne est mise en gras, et la deuxième meilleure performance est soulignée.

Jeu de données	Synthetic	COCO@11	COCO@24	COCO@48	CelebA	CUB-Simp	VG12	SunIn	SunOut
Arbre de décision	97.99	97.15	97.47	96.83	91.20	100.0	97.83	18.38	14.16
Forêt aléatoire	99.00	99.22	98.73	99.38	91.32	100.0	97.83	31.41	24.73
SVM	100.0	98.70	97.61	98.60	91.32	100.0	98.19	30.13	28.14

TABLE 8.3 – Évaluation de la capacité des concepts à séparer les classes dans les différents jeux de données.

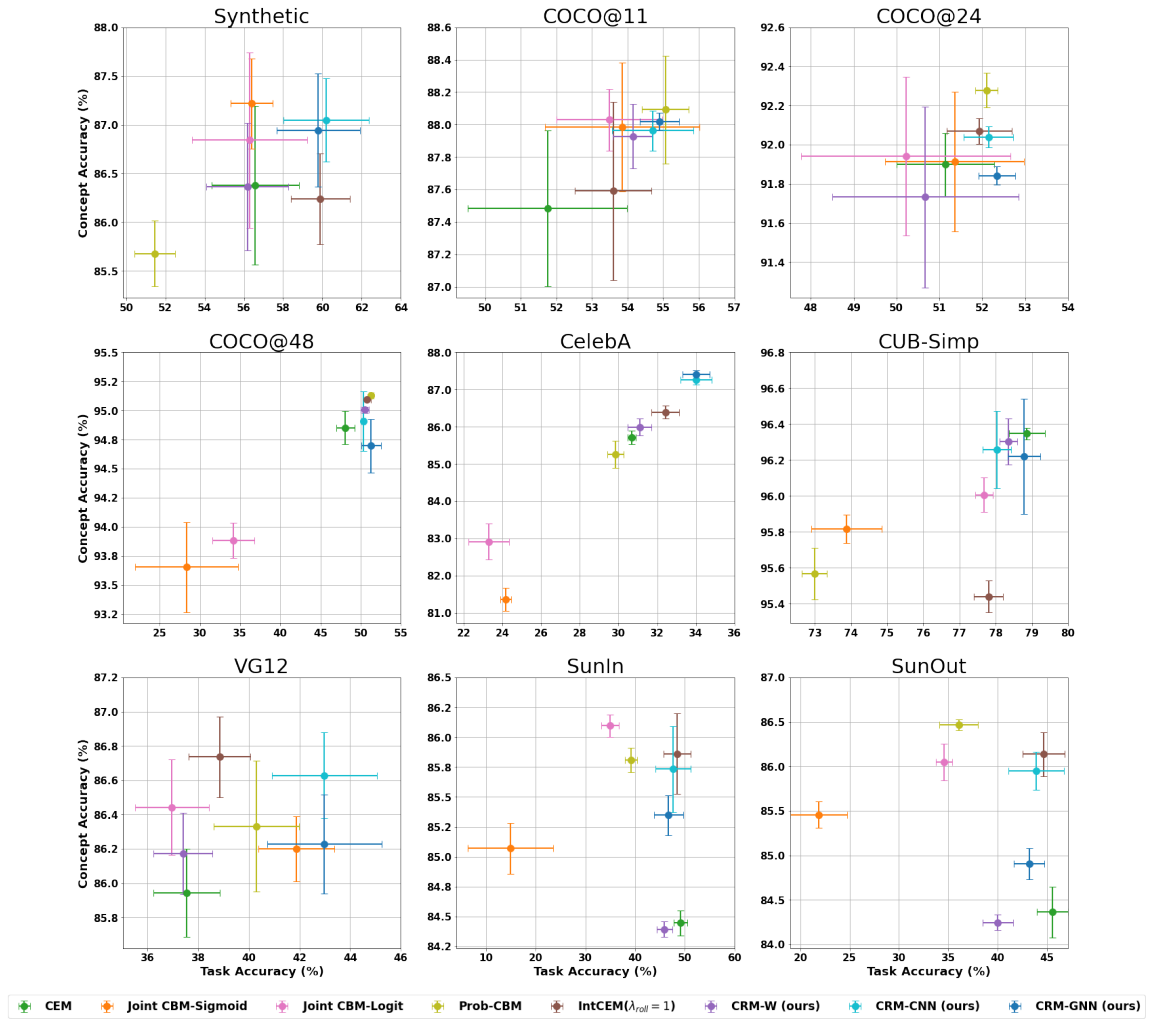


FIGURE 8.4 – Performances sur la tâche et les concepts pour différents modèles de référence. L'axe des abscisses représente les performances sur la tâche et l'axe des ordonnées représente les performances sur les concepts.

8.5.2 Performances des interventions

Performances sur la tâche

Dans cette section, nous analysons l'impact des interventions en phase de test sur la performance des tâches avec CRM, en comparaison aux autres modèles de référence. La Figure 8.5 présente les résultats des interventions sur tous les jeux de données, résumés dans la Table 8.4. Pour rappel, une intervention consiste à corriger une mauvaise prédiction faite par le modèle sur la présence d'un concept dans une image. L'objectif ici est de mesurer l'impact de cette intervention sur la prédiction de la tâche par le modèle. Les résultats montrent que, après interventions sur les concepts, la précision sur la tâche avec CRM est compétitive à celle des modèles concurrents, généralement entre les deux meilleurs résultats. CRM obtient généralement la meilleure précision après des interventions sur moins de 50% des concepts. C'est particulièrement le cas pour les jeux de données Synthetic, COCO@24, CelebA, CUB-Simp et VG12. Pour des interventions sur plus de 50% de concepts, IntCEM obtient généralement de meilleures performance, suivi par CEM qui obtient généralement

les deuxièmes meilleures performances. Il est toutefois important de rappeler que IntCEM, contrairement à CEM et aux autres méthodes, est optimisé pour obtenir de meilleures résultats après une intervention. Particulièrement pour les jeux de données SunIn et SunOut, nous pouvons constater que les performances sur la tâche ne sont pas impactées par les interventions sur les concepts. Ces résultats s'expliquent par le fait que les concepts ne sont pas discriminant pour la tâche, comme illustré dans la Table 8.3.

Les résultats démontrent également la pertinence d'intégration des connaissances dans CRM. En effet, nous observons que les interventions appliquées avec utilisation du module de raisonnement (CRM-CNN et CRM-GNN) obtiennent toujours des performances supérieures par rapport à la non utilisation du module de raisonnement (CRM-W), quelque soit le nombre de concepts ou le jeux de données. De plus, nous pouvons remarquer une fois de plus l'impact sur CRM-CNN du *concept reasoner* sur les performances de l'*embedding generator* grâce à la rétro-propagation.

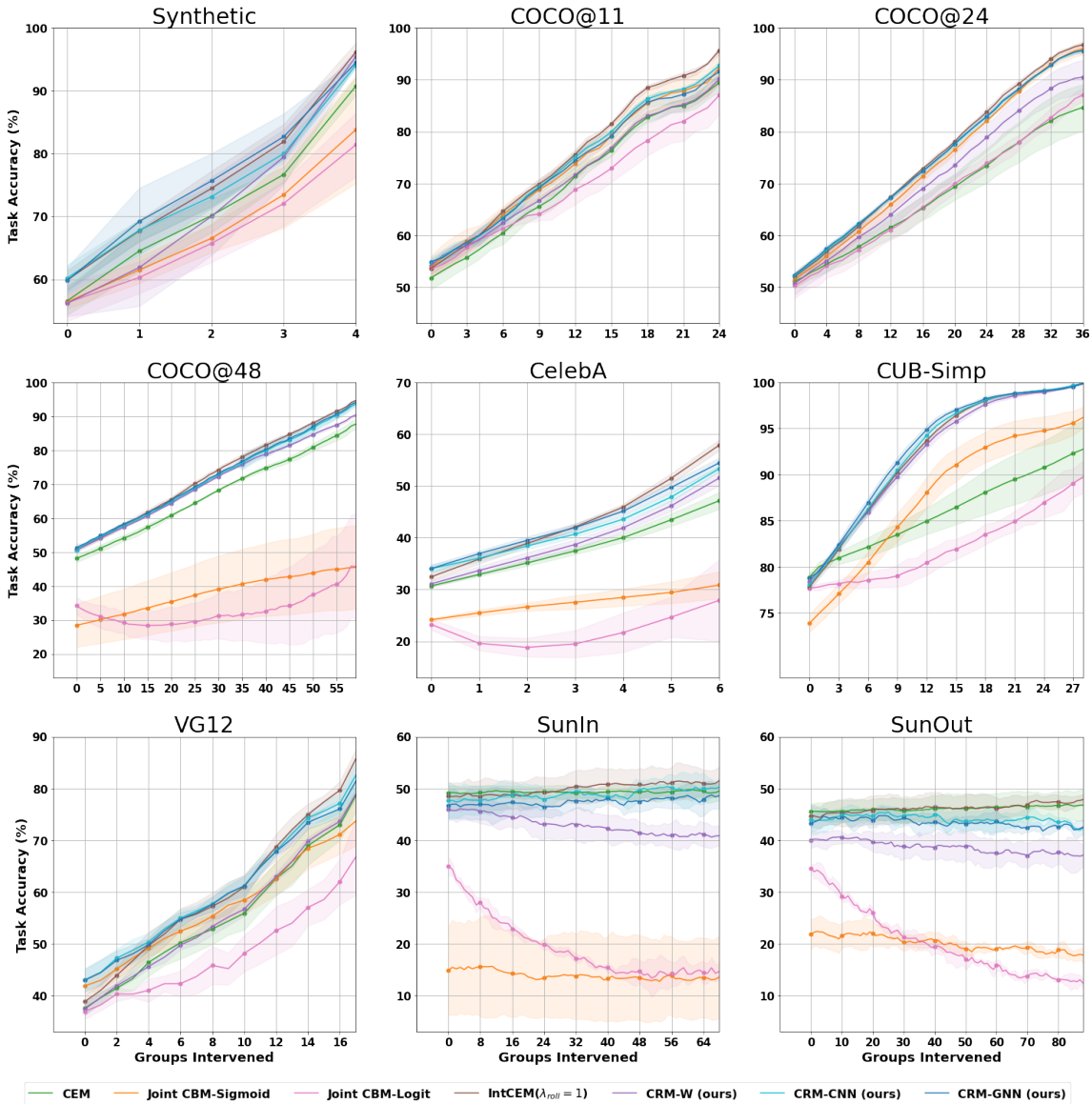


FIGURE 8.5 – Performances sur la tâche pour tous les modèles de référence après un nombre variable d'interventions où les concepts sont sélectionnées aléatoirement.

	Dataset	CEM	Joint CBM-Sigmoid	Joint CBM-Logit	IntCEM($\lambda_{\text{roll}} = 1$)	CRM-W	CRM-CNN	CRM-GNN
25%	Synthetic	64.52 ± 3.22	61.51 ± 2.12	60.30 ± 2.52	67.74 ± 1.75	61.89 ± 6.16	67.84 ± 1.71	**69.23 ± 5.33**
	COCO@11	60.42 ± 2.13	63.92 ± 1.67	61.32 ± 0.89	**64.64 ± 0.50**	62.41 ± 0.78	63.43 ± 1.28	63.25 ± 1.24
	COCO@24	58.67 ± 1.80	62.05 ± 1.32	58.14 ± 2.17	63.23 ± 0.68	60.59 ± 2.01	63.37 ± 0.47	**63.45 ± 0.30**
	COCO@48	56.60 ± 0.91	33.24 ± 7.95	28.44 ± 4.08	**60.95 ± 0.93**	59.93 ± 0.10	60.34 ± 0.03	60.83 ± 0.06
	CelebA	32.89 ± 0.35	25.49 ± 0.59	19.61 ± 1.37	35.85 ± 0.92	33.67 ± 0.80	36.03 ± 0.63	**36.94 ± 0.57**
	CUB-Simp	82.63 ± 1.13	81.82 ± 1.23	78.73 ± 0.91	87.53 ± 0.44	87.27 ± 0.47	87.73 ± 0.60	**88.51 ± 0.57**
	VG12	46.43 ± 2.67	49.10 ± 1.48	41.01 ± 2.22	49.68 ± 2.65	45.63 ± 0.43	**50.35 ± 1.72**	49.84 ± 1.22
	SunIn	**49.36 ± 1.22**	14.17 ± 8.28	22.58 ± 0.81	48.72 ± 2.46	44.23 ± 2.01	49.02 ± 2.18	47.17 ± 2.24
	SunOut	**46.00 ± 2.14**	21.74 ± 2.14	24.07 ± 0.69	45.94 ± 2.79	39.55 ± 1.91	45.35 ± 1.47	44.76 ± 1.58
50%	Synthetic	70.15 ± 2.67	66.53 ± 2.26	65.73 ± 3.01	74.47 ± 2.80	70.03 ± 6.21	73.17 ± 1.70	**75.67 ± 4.37**
	COCO@11	71.36 ± 1.79	73.77 ± 2.46	68.82 ± 2.85	**75.59 ± 1.00**	71.62 ± 0.69	75.10 ± 1.13	74.43 ± 1.40
	COCO@24	67.41 ± 2.49	74.01 ± 0.91	67.74 ± 1.80	**75.49 ± 0.64**	71.47 ± 2.52	74.89 ± 0.53	74.97 ± 0.39
	COCO@48	67.51 ± 0.19	38.70 ± 9.71	30.77 ± 6.36	**73.39 ± 0.75**	71.51 ± 0.11	71.93 ± 0.05	72.35 ± 0.11
	CelebA	37.46 ± 0.68	27.55 ± 1.34	19.50 ± 2.61	**42.09 ± 0.61**	38.70 ± 0.95	40.70 ± 0.65	41.95 ± 0.76
	CUB-Simp	85.96 ± 1.72	90.40 ± 1.80	81.62 ± 1.25	95.62 ± 0.21	95.10 ± 0.40	96.02 ± 0.23	**96.51 ± 0.32**
	VG12	52.85 ± 2.95	55.31 ± 2.41	45.85 ± 3.68	57.26 ± 2.65	53.36 ± 1.64	**57.71 ± 1.44**	57.64 ± 1.43
	SunIn	49.29 ± 1.14	13.96 ± 8.52	17.38 ± 0.10	**50.28 ± 3.13**	42.66 ± 3.29	48.74 ± 1.22	47.45 ± 1.81
	SunOut	46.18 ± 2.96	19.95 ± 1.22	18.34 ± 2.12	**46.24 ± 2.44**	38.83 ± 2.72	43.38 ± 2.28	42.73 ± 1.19
75%	Synthetic	76.68 ± 1.21	73.47 ± 5.29	72.06 ± 3.52	81.91 ± 2.79	79.48 ± 4.36	80.00 ± 1.40	**82.70 ± 3.67**
	COCO@11	82.72 ± 1.16	85.53 ± 2.47	78.26 ± 2.90	**88.46 ± 0.53**	83.01 ± 1.12	86.31 ± 0.60	85.66 ± 1.05
	COCO@24	76.99 ± 3.72	86.47 ± 0.43	76.79 ± 1.70	**88.13 ± 0.70**	82.87 ± 2.67	86.97 ± 0.50	87.10 ± 0.44
	COCO@48	76.73 ± 0.84	42.59 ± 11.07	34.07 ± 11.01	**84.03 ± 0.56**	80.90 ± 0.32	82.43 ± 1.72	82.67 ± 1.70
	CelebA	40.02 ± 0.85	28.48 ± 1.64	21.70 ± 3.81	**45.94 ± 0.60**	41.93 ± 0.82	43.58 ± 0.52	45.12 ± 0.79
	CUB-Simp	89.48 ± 2.71	94.20 ± 1.63	84.92 ± 1.13	98.78 ± 0.10	98.55 ± 0.20	**98.81 ± 0.13**	98.81 ± 0.15
	VG12	62.60 ± 3.23	62.60 ± 3.15	52.56 ± 4.65	**68.74 ± 2.51**	62.96 ± 0.78	67.89 ± 0.81	67.82 ± 0.67
	SunIn	49.22 ± 1.31	12.89 ± 7.91	13.39 ± 3.70	**51.00 ± 3.44**	41.38 ± 1.69	49.95 ± 2.10	47.95 ± 1.85
	SunOut	46.65 ± 2.75	19.59 ± 0.47	13.98 ± 0.96	**46.71 ± 2.00**	37.63 ± 1.83	43.80 ± 1.44	42.67 ± 1.31
100%	Synthetic	90.75 ± 1.41	83.82 ± 8.56	81.41 ± 5.08	**96.18 ± 1.48**	95.48 ± 0.95	94.07 ± 0.38	94.47 ± 0.78
	COCO@11	89.49 ± 1.75	92.37 ± 2.95	87.08 ± 3.65	**95.69 ± 0.67**	90.17 ± 0.97	92.79 ± 0.83	91.57 ± 1.02
	COCO@24	84.68 ± 4.42	95.98 ± 1.19	87.08 ± 1.32	**96.79 ± 0.52**	90.51 ± 3.34	95.61 ± 0.40	95.66 ± 0.51
	COCO@48	87.69 ± 1.52	45.48 ± 12.34	45.74 ± 15.64	**94.62 ± 0.27**	90.35 ± 0.44	93.61 ± 0.92	94.04 ± 0.96
	CelebA	47.16 ± 1.64	30.86 ± 2.62	27.94 ± 7.86	**57.91 ± 1.05**	51.55 ± 1.43	53.34 ± 0.59	54.47 ± 0.99
	CUB-Simp	92.75 ± 3.13	96.19 ± 1.15	89.74 ± 1.77	99.89 ± 0.03	99.84 ± 0.04	**99.92 ± 0.02**	99.86 ± 0.05
	VG12	78.63 ± 3.07	73.72 ± 4.80	66.79 ± 7.31	**85.70 ± 1.84**	78.92 ± 0.90	82.55 ± 2.72	81.40 ± 2.02
	SunIn	49.50 ± 1.76	13.60 ± 7.96	14.67 ± 1.62	**51.50 ± 2.77**	40.88 ± 1.96	50.23 ± 1.84	48.59 ± 2.53
	SunOut	46.77 ± 2.99	17.80 ± 1.48	12.43 ± 0.88	**47.85 ± 1.02**	37.10 ± 2.19	42.37 ± 0.22	42.55 ± 1.44

TABLE 8.4 – Performance prédictive sur la tâche (en %) pour les jeux de données et les des méthodes de référence, après interventions sur un pourcentage de concepts au moment du test (indiquée à gauche des noms des jeux de données). Toutes les interventions utilisent une politique d'intervention aléatoire (Random policy). Les meilleures performances sont indiquées en gras et les deuxièmes meilleures performances sont soulignées

Étude d'ablation

La Figure 8.6 illustre les performances sur la tâche pour les jeux de données Synthetic, COCO@11, CelebA et CUB-Simp, dans le cadre d'interventions effectuées en phase de test sur le modèle CRM, selon différentes politiques d'intervention. Les deux dernières colonnes correspondent aux deux sortie de CRM utilisant le module de raisonnement sur les concepts (CRM-GNN et CRM-GCN), tandis que la première colonne présente la version sans raisonnement (CRM-W). Les Tables 8.5, 8.6, 8.7 et 8.8 fournissent un résumé détaillé des résultats obtenus pour chaque jeu de données.

Sur le jeu de données Synthetic (Table 8.5), CRM obtient généralement les meilleures performances après intervention jusqu'à 75% de concepts utilisés. Au delà de 75%, CRM-W obtient de meilleures performances. Il est toutefois intéressant de noter que l'impact du raisonnement sur la performance est d'autant plus marqué que le nombre de concepts intervenus est faible. Par exemple, lorsqu'une intervention porte sur seulement 25% des concepts, l'utilisation du raisonnement améliore la performance en moyenne de 10%, contre seulement 2% lorsque 75% des concepts sont modifiés. Sur les jeux de données COCO@11 (Table 8.6), CelebA (Table 8.7) et CUB-Simp (Table 8.8), CRM obtient toujours les meilleures résultats après intervention sur les concepts. Sur COCO@11, les meilleures résultats sont presque toujours obtenus à la sortie de l'*embedding generator*, alors que sur CelebA, les meilleures résultats sont toujours obtenus à la sortie de *concept reasoner*. Tout comme pour le Synthetic, l'impact du raisonnement sur la performance est d'autant plus marqué que le nombre de concepts intervenus est faible pour chacun des jeux de données.

La comparaison des différentes politiques d'intervention révèle que les politiques Skyline (oracle) et CooP permettent d'obtenir les meilleures performances sur la majorité des jeux de données. Il est toutefois important de noter que la politique Skyline, bien qu'idéale en théorie, entraîne une dégradation significative des performances lorsque le modèle CRM est utilisé sans module de raisonnement sur les concepts. Ce constat souligne le rôle essentiel du raisonnement sur les concepts dans la capacité du modèle à exploiter efficacement les interventions.

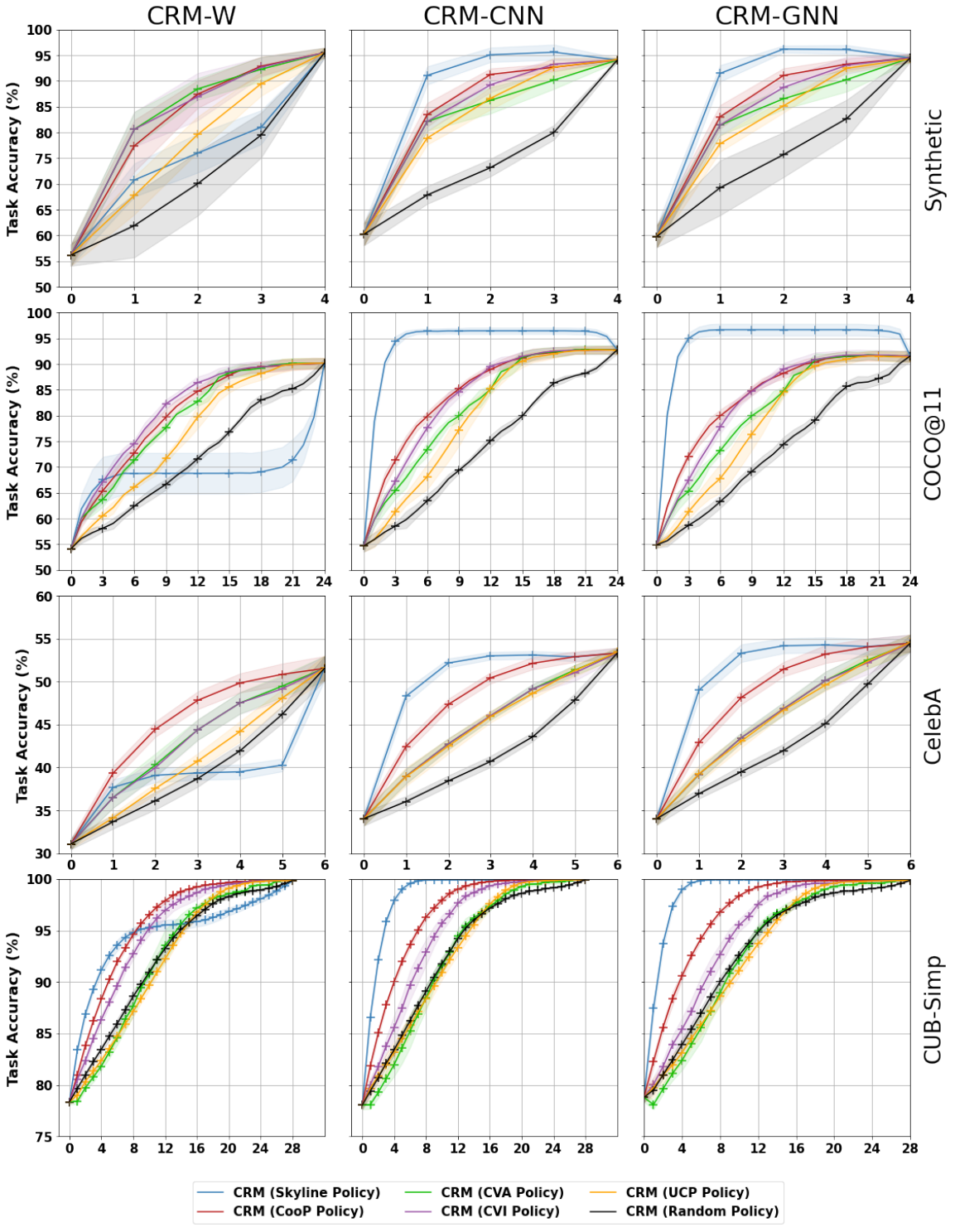


FIGURE 8.6 – Résultats des interventions sur la performance de la tâche du modèle CRM avec et sans le module de raisonnement sur les concepts. Les interventions, faites en phase de test, ont été effectuées en utilisant différentes politiques d'intervention.

	Polices	CRM-W	CRM-CNN	CRM-GNN
25%	Skyline Policy	71.36 ± 3.51	90.45 ± 1.88	**91.46 ± 2.05**
	CooP Policy	78.56 ± 2.79	**83.42 ± 2.87**	82.75 ± 2.88
	CVA Policy	81.74 ± 1.44	**82.24 ± 2.26**	81.41 ± 2.50
	CVI Policy	81.74 ± 1.44	**82.24 ± 2.26**	81.41 ± 2.50
	UCP Policy	67.34 ± 2.87	**78.89 ± 1.09**	77.39 ± 1.42
	Random Policy	60.94 ± 7.63	68.68 ± 1.66	**71.34 ± 6.02**
50%	Skyline Policy	75.88 ± 4.66	94.81 ± 1.25	**96.31 ± 0.85**
	CooP Policy	88.11 ± 0.24	**91.46 ± 1.48**	91.12 ± 1.32
	CVA Policy	**88.27 ± 1.94**	85.93 ± 3.20	86.93 ± 2.56
	CVI Policy	88.27 ± 2.51	**89.78 ± 1.44**	88.94 ± 0.82
	UCP Policy	80.07 ± 2.40	**87.60 ± 0.63**	85.59 ± 0.85
	Random Policy	69.32 ± 7.87	73.53 ± 1.85	**78.20 ± 3.39**
75%	Skyline Policy	80.74 ± 4.21	**95.31 ± 0.85**	**95.98 ± 1.09**
	CooP Policy	**93.47 ± 0.82**	92.13 ± 0.47	92.96 ± 1.23
	CVA Policy	**92.80 ± 0.63**	89.95 ± 0.71	**90.45 ± 0.71**
	CVI Policy	92.13 ± 1.94	**92.96 ± 1.09**	92.80 ± 2.11
	UCP Policy	90.12 ± 1.94	**92.96 ± 0.41**	92.46 ± 0.41
	Random Policy	**81.57 ± 0.24**	79.56 ± 1.55	80.74 ± 1.85
100%	Skyline Policy	**95.64 ± 0.63**	94.14 ± 0.24	94.30 ± 0.95
	CooP Policy	**95.64 ± 0.63**	94.14 ± 0.24	94.30 ± 0.95
	CVA Policy	**95.64 ± 0.63**	94.14 ± 0.24	94.30 ± 0.95
	CVI Policy	**95.64 ± 0.63**	94.14 ± 0.24	94.30 ± 0.95
	UCP Policy	**95.64 ± 0.63**	94.14 ± 0.24	94.30 ± 0.63
	Random Policy	**95.64 ± 0.63**	94.14 ± 0.24	94.30 ± 0.95

TABLE 8.5 – Performances après intervention sur la tâche du modèle CRM avec et sans le module de raisonnement sur les concepts. Les interventions en phase de test ont été effectuées en utilisant différentes politiques d'intervention sur le jeu de données Synthetic. La meilleure précision pour chaque ligne est mise en gras, et la deuxième meilleure précision est soulignée.

	Polices	CRM-W	CRM-CNN	CRM-GNN
25%	Skyline Policy	68.74 ± 4.09	96.39 ± 0.48	**96.68 ± 1.15**
	CooP Policy	72.46 ± 0.96	79.64 ± 0.59	**79.68 ± 0.22**
	CVA Policy	71.73 ± 0.28	**74.41 ± 0.43**	73.54 ± 0.38
	CVI Policy	74.66 ± 0.69	**77.39 ± 0.98**	77.39 ± 1.44
	UCP Policy	65.93 ± 0.43	**68.48 ± 2.75**	68.22 ± 3.21
	Random Policy	61.87 ± 0.21	**63.85 ± 1.40**	63.59 ± 1.45
50%	Skyline Policy	68.79 ± 3.97	96.47 ± 0.62	**96.68 ± 1.15**
	CooP Policy	84.91 ± 0.52	**89.28 ± 0.65**	88.37 ± 0.86
	CVA Policy	82.53 ± 0.74	**85.60 ± 0.28**	85.21 ± 0.37
	CVI Policy	86.34 ± 1.28	**90.10 ± 0.48**	89.36 ± 0.64
	UCP Policy	80.24 ± 0.98	**85.65 ± 1.26**	84.82 ± 1.22
	Random Policy	71.12 ± 0.27	**74.66 ± 1.07**	73.97 ± 1.59
75%	Skyline Policy	69.05 ± 3.92	96.47 ± 0.62	**96.68 ± 1.15**
	CooP Policy	89.32 ± 1.17	**92.78 ± 0.86**	91.79 ± 0.48
	CVA Policy	88.98 ± 0.76	**92.95 ± 0.64**	91.92 ± 0.34
	CVI Policy	89.23 ± 0.97	**92.95 ± 0.54**	92.22 ± 0.21
	UCP Policy	87.68 ± 0.69	**92.52 ± 0.44**	90.96 ± 1.02
	Random Policy	82.36 ± 0.84	**86.29 ± 0.44**	85.42 ± 0.69
100%	Skyline Policy	90.17 ± 0.97	**92.79 ± 0.83**	91.57 ± 1.02
	CooP Policy	89.93 ± 1.06	**93.34 ± 0.54**	91.83 ± 0.46
	CVA Policy	89.93 ± 1.06	**93.34 ± 0.54**	91.83 ± 0.46
	CVI Policy	89.93 ± 1.06	**93.34 ± 0.54**	91.83 ± 0.46
	UCP Policy	89.97 ± 0.91	**93.17 ± 0.40**	91.57 ± 0.66
	Random Policy	89.93 ± 1.06	**93.34 ± 0.54**	91.83 ± 0.46

TABLE 8.6 – Performances après intervention sur la tâche du modèle CRM avec et sans le module de raisonnement sur les concepts. Les interventions en phase de test ont été effectuées en utilisant différentes politiques d'intervention sur le jeu de données COCO@11. La meilleure précision pour chaque ligne est mise en gras, et la deuxième meilleure précision est soulignée.

	Polices	CRM-W	CRM-CNN	CRM-GNN
25%	Skyline Policy	37.34 ± 0.85	48.35 ± 0.96	**49.34 ± 0.88**
	CooP Policy	38.77 ± 0.13	42.45 ± 0.91	**42.98 ± 0.80**
	CVA Policy	35.79 ± 0.13	38.50 ± 0.75	**38.93 ± 0.75**
	CVI Policy	35.79 ± 0.13	38.50 ± 0.75	**38.93 ± 0.75**
	UCP Policy	33.78 ± 0.37	38.48 ± 1.05	**38.87 ± 0.99**
	Random Policy	33.18 ± 0.21	35.91 ± 0.72	**36.86 ± 0.59**
50%	Skyline Policy	39.04 ± 0.83	53.10 ± 0.56	**54.51 ± 0.69**
	CooP Policy	47.27 ± 0.48	50.49 ± 0.83	**51.65 ± 0.81**
	CVA Policy	43.74 ± 0.54	46.25 ± 0.41	**47.29 ± 0.29**
	CVI Policy	43.74 ± 0.54	46.08 ± 0.24	**47.26 ± 0.23**
	UCP Policy	40.63 ± 1.10	45.65 ± 0.76	**46.70 ± 0.73**
	Random Policy	38.18 ± 0.44	40.82 ± 0.76	**42.10 ± 0.72**
75%	Skyline Policy	39.16 ± 0.84	53.21 ± 0.56	**54.59 ± 0.69**
	CooP Policy	49.27 ± 0.74	52.31 ± 0.89	**53.52 ± 0.96**
	CVA Policy	46.92 ± 0.95	49.33 ± 0.53	**50.60 ± 0.61**
	CVI Policy	46.92 ± 0.95	49.33 ± 0.53	**50.60 ± 0.61**
	UCP Policy	44.05 ± 1.50	48.60 ± 0.67	**49.77 ± 0.80**
	Random Policy	41.57 ± 0.59	43.79 ± 0.55	**45.52 ± 0.57**
100%	Skyline Policy	50.75 ± 0.94	53.54 ± 0.69	**54.86 ± 0.80**
	CooP Policy	50.75 ± 0.94	53.53 ± 0.70	**54.85 ± 0.81**
	CVA Policy	50.75 ± 0.94	53.54 ± 0.69	**54.86 ± 0.80**
	CVI Policy	50.75 ± 0.94	53.54 ± 0.69	**54.86 ± 0.80**
	UCP Policy	50.88 ± 1.04	53.61 ± 0.66	**54.97 ± 0.80**
	Random Policy	50.75 ± 0.94	53.54 ± 0.69	**54.86 ± 0.80**

TABLE 8.7 – Performances après intervention sur la tâche du modèle CRM avec et sans le module de raisonnement sur les concepts. Les interventions en phase de test ont été effectuées en utilisant différentes politiques d'intervention sur le jeu de données CelebA. La meilleure précision pour chaque ligne est mise en gras, et la deuxième meilleure précision est soulignée.

	Polices	CRM-W	CRM-CNN	CRM-GNN
25%	Skyline Policy	94.35 ± 0.52	99.80 ± 0.04	**99.92 ± 0.03**
	CooP Policy	93.22 ± 0.24	94.99 ± 0.32	**95.59 ± 0.12**
	CVA Policy	86.57 ± 1.36	87.00 ± 1.47	**87.08 ± 1.25**
	CVI Policy	91.70 ± 0.21	**91.78 ± 1.57**	91.51 ± 1.23
	UCP Policy	85.78 ± 0.29	87.08 ± 0.93	**87.18 ± 1.07**
	Random Policy	87.57 ± 0.35	87.68 ± 0.66	**88.38 ± 0.68**
50%	Skyline Policy	95.64 ± 0.49	**99.94 ± 0.02**	99.92 ± 0.03
	CooP Policy	98.78 ± 0.12	99.51 ± 0.14	**99.62 ± 0.06**
	CVA Policy	96.32 ± 0.58	96.08 ± 0.39	**96.47 ± 0.27**
	CVI Policy	98.04 ± 0.82	98.84 ± 0.41	**98.86 ± 0.29**
	UCP Policy	94.83 ± 0.52	95.67 ± 0.23	**96.28 ± 0.46**
	Random Policy	95.26 ± 0.46	96.10 ± 0.27	**96.41 ± 0.38**
75%	Skyline Policy	97.10 ± 0.36	**99.94 ± 0.01**	99.92 ± 0.03
	CooP Policy	99.74 ± 0.02	**99.92 ± 0.04**	99.88 ± 0.06
	CVA Policy	98.66 ± 0.33	**99.59 ± 0.18**	99.52 ± 0.10
	CVI Policy	99.46 ± 0.43	**99.87 ± 0.02**	99.79 ± 0.06
	UCP Policy	99.42 ± 0.23	**99.75 ± 0.02**	99.71 ± 0.08
	Random Policy	98.62 ± 0.22	**98.80 ± 0.15**	98.75 ± 0.16
100%	Skyline Policy	99.84 ± 0.04	**99.92 ± 0.02**	99.86 ± 0.05
	CooP Policy	99.86 ± 0.01	**99.93 ± 0.03**	99.88 ± 0.05
	CVA Policy	99.86 ± 0.01	**99.93 ± 0.03**	99.88 ± 0.05
	CVI Policy	99.86 ± 0.01	**99.93 ± 0.03**	99.88 ± 0.05
	UCP Policy	99.86 ± 0.01	**99.93 ± 0.03**	99.88 ± 0.05
	Random Policy	99.86 ± 0.01	**99.93 ± 0.03**	99.88 ± 0.05

TABLE 8.8 – Performances après intervention sur la tâche du modèle CRM avec et sans le module de raisonnement sur les concepts. Les interventions en phase de test ont été effectuées en utilisant différentes politiques d'intervention sur le jeu de données SUB-Simp. La meilleure précision pour chaque ligne est mise en gras, et la deuxième meilleure précision est soulignée.

		1	2	4	8	16	32	64	128
Tâche	Synthetic	41.21 ± 16.0	53.94 ± 1.32	55.95 ± 0.24	59.80 ± 2.17	59.76 ± 2.73	**60.97 ± 1.85**	57.12 ± 2.4	56.11 ± 2.02
	COCO@11	32.23 ± 21.21	52.14 ± 0.00	51.43 ± 0.06	51.75 ± 0.00	**54.78 ± 0.32**	54.41 ± 0.32	51.05 ± 0.08	47.37 ± 0.16
	CelabA	25.21 ± 9.85	31.01 ± 1.42	33.98 ± 0.12	35.24 ± 0.52	33.73 ± 0.74	**36.16 ± 0.67**	34.89 ± 0.67	35.13 ± 0.23
Concept	Synthetic	75.17 ± 5.93	77.04 ± 1.03	83.63 ± 0.12	83.73 ± 1.77	86.63 ± 0.56	85.24 ± 1.54	84.56 ± 1.96	**86.68 ± 0.89**
	COCO@11	78.65 ± 7.59	87.88 ± 0.09	87.48 ± 0.02	87.54 ± 0.07	**88.06 ± 0.05**	87.59 ± 0.15	87.61 ± 0.22	86.81 ± 0.23
	CelabA	75.91 ± 15.96	87.20 ± 0.11	87.44 ± 0.16	87.62 ± 0.16	87.36 ± 0.09	**87.65 ± 0.14**	87.27 ± 0.18	87.56 ± 0.20

TABLE 8.9 – Étude de l'impact de la taille des plongements des concepts sur la performance des modèles pendant l'entraînement de CRM. Les jeux de données utiilsés sont Synthetic, COCO@11 et CelebA. Pour chaque ligne, le meilleur résultat est en gras et le deuxième meilleur résultat est souligné.

		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Tâche	Synthetic	54.61 ± 2.67	54.94 ± 0.63	56.95 ± 1.25	59.13 ± 1.18	57.62 ± 1.25	**60.97 ± 1.03**	59.30 ± 0.41	58.46 ± 2.40	58.96 ± 0.85	58.96 ± 0.85	56.11 ± 1.71
	COCO@11	52.40 ± 0.39	52.58 ± 0.45	54.45 ± 0.97	53.3 ± 0.63	53.63 ± 0.45	55.06 ± 1.49	54.86 ± 1.30	55.12 ± 0.39	**55.51 ± 0.13**	**55.51 ± 0.13**	54.93 ± 0.97
	CelabA	35.14 ± 0.37	34.92 ± 0.04	34.90 ± 1.22	35.45 ± 0.63	**35.67 ± 0.53**	35.26 ± 0.55	34.7 ± 0.66	34.87 ± 0.69	35.04 ± 0.51	35.04 ± 0.51	30.81 ± 0.15
Concept	Synthetic	82.71 ± 0.72	85.78 ± 0.29	86.06 ± 0.84	85.27 ± 0.86	85.40 ± 1.46	85.70 ± 1.24	85.17 ± 1.43	83.13 ± 3.83	83.89 ± 2.36	**86.49 ± 0.09**	85.61 ± 0.58
	COCO@11	87.29 ± 0.01	87.94 ± 0.14	88.01 ± 0.21	87.42 ± 0.50	87.79 ± 0.24	87.74 ± 0.01	88.00 ± 0.36	88.11 ± 0.06	**88.31 ± 0.19**	88.01 ± 0.02	87.34 ± 0.61
	CelabA	86.73 ± 0.12	86.54 ± 0.17	86.57 ± 0.30	86.61 ± 0.21	**86.75 ± 0.14**	86.51 ± 0.04	86.43 ± 0.10	86.62 ± 0.18	86.44 ± 0.10	86.59 ± 0.13	83.96 ± 0.30

TABLE 8.10 – Étude de l'impact de la probabilité RandInt (p_{int}) sur la performance de CRM. Les jeux de données utilisés sont Synthetic, COCO@11 et CelabA. Pour chaque ligne, le meilleur résultat est en gras et le deuxième meilleur résultat est souligné.

8.5.3 Évaluation de la taille des plongements

Cette section analyse l'effet de la taille d des plongements sur les performances du modèle CRM, en comparant différentes configurations. Les résultats pour les jeux de données Synthetic, COCO@11 et CelebA sont présentés dans la Figure 8.7. Les deux premières colonnes illustrent la précision sur la tâche et sur les concepts selon les tailles de représentation, tandis que la dernière colonne montre la précision sur la tâche lors d'interventions aléatoires (police Fandom), en fonction du nombre croissant de groupes de concepts et de la taille des plongements. Les résultats, présentés dans la Table 8.9, correspondent à une moyenne sur trois initialisations.

Les meilleures performances sur la tâche sont obtenues avec des tailles de plongement $d = 16$ (COCO@11) et $d = 32$ (Synthetic et CelebA) Les précisions sont de 60.97%, 54.78% et 36.16% respectivement pour Synthetic, (COCO@11 et CelebA. En ce qui concerne la précision sur les concepts, l'effet de la taille du plongement reste limité pour COCO@11 et CelebA, avec un léger avantage pour $d = 16$ sur COCO@11 et $d = 32$ sur CelebA. Le jeu de données Synthetic, en revanche, montre une sensibilité plus marquée, avec des performances optimales pour $d = 128$. Dans l'ensemble, CRM parvient à maintenir de bonnes performances avec différentes tailles de plongement, tant que celles-ci ne sont pas trop faibles.

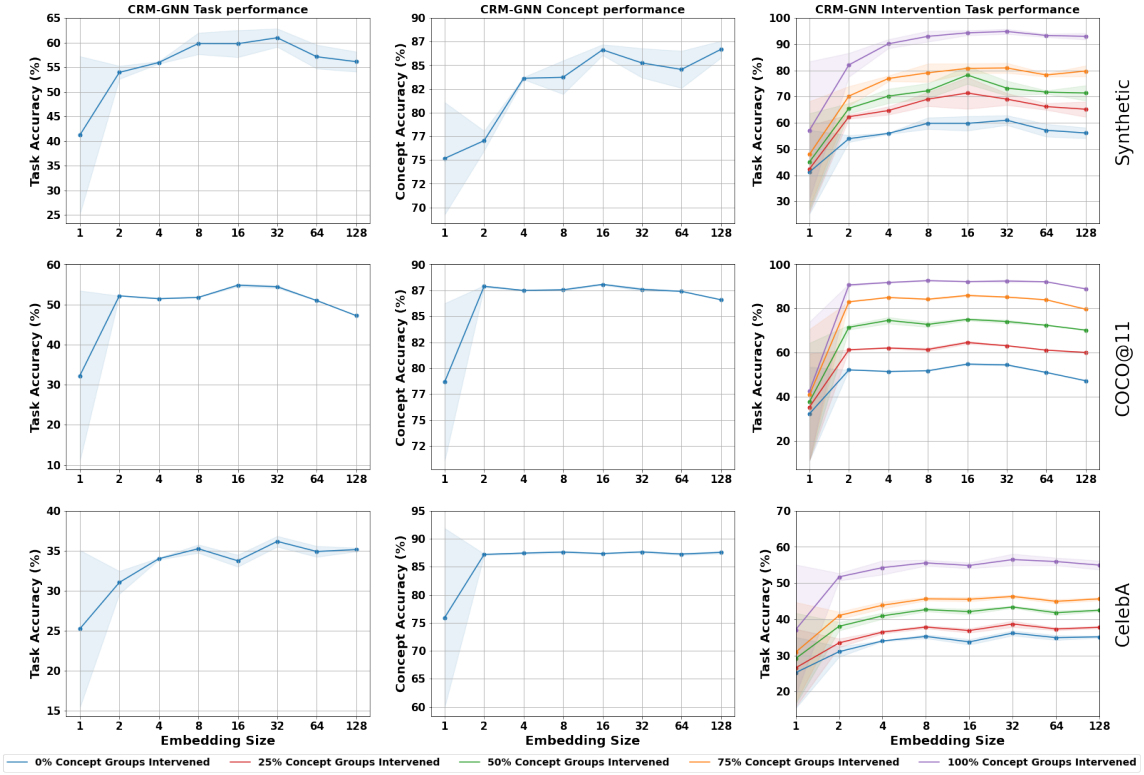


FIGURE 8.7 – Étude de l'impact de la taille des plongements des concepts sur la performance des modèles pendant l'entraînement de CRM. Les jeux de données utiilsés sont Synthetic, COCO@11 et CelebA.

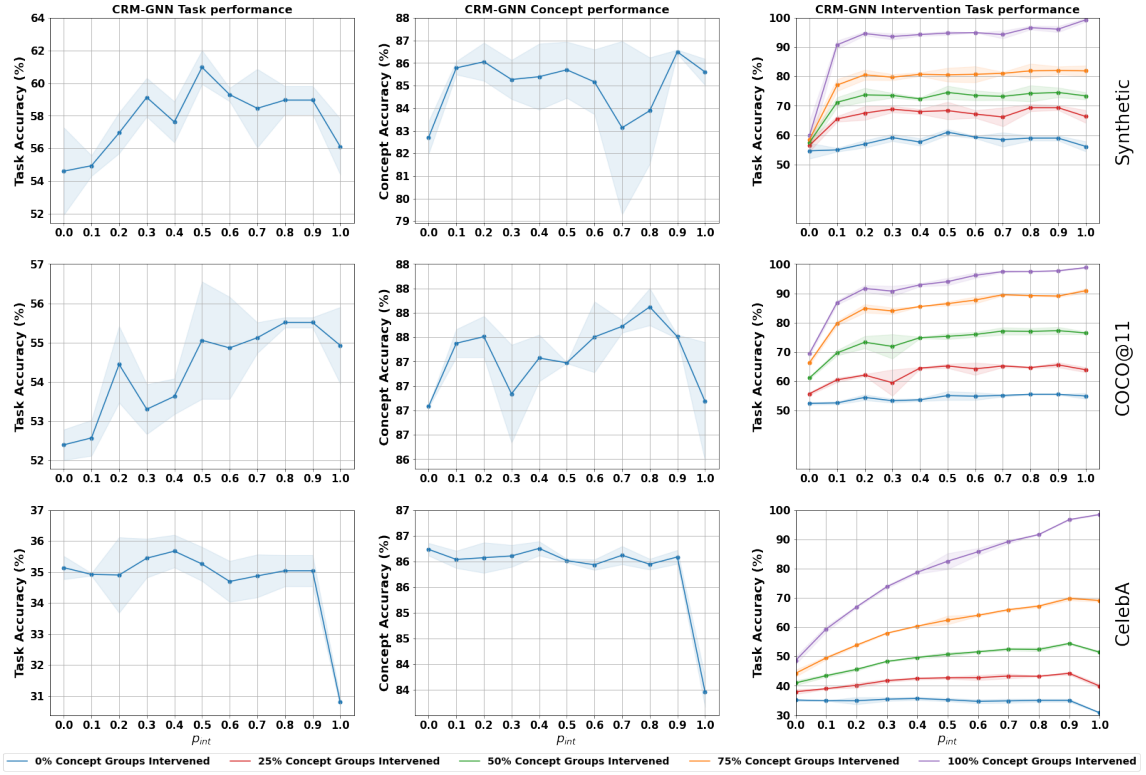


FIGURE 8.8 – Étude de l'impact de la probabilité RandInt (p_{int}) sur la performance de CRM. Les jeux de données utilisés sont Synthetic, COCO@11 et CelebA.

8.5.4 Étude de la probabilité *RandInt*

La Figure 8.8 présente les résultats obtenus en faisant varier la probabilité p_{int} pendant la phase d'apprentissage de CRM, entraîné sur les jeux de données Synthetic, COCO@11 et CelebA. Les résultats obtenus présentent les moyennes obtenus après trois initialisations. Les deux premières colonnes de la figure illustrent la précision sur la tâche et sur les concepts à la sortie du *concept reasoner* (CRM-GNN) entraîné avec différentes valeurs de p_{int}, tandis que la dernière colonne montre la précision sur la tâche lors d'interventions sur un nombre croissant de groupes de concepts, en fonction des différentes valeurs de la probabilité d'intervention p_{int}. La Table 8.10 présente les résultats sur la tâche et sur les concepts. Sur les jeux de données Synthetic et COCO@11, nous observons que la précision sur la tâche augmente progressivement avant de décliner. Les meilleures performances sont obtenues pour $p_{int} = 0.5$ sur Synthetic (60.97% de précision), et pour $p_{int} = 0.8$ sur COCO@11 (55.51% de précision). Sur le jeu de données CelebA, les performances sur la tâche ne fluctuent pas beaucoup pour $0.0 < p_{int} < 0.9$. Par contre, elle sont particulièrement mauvaise pour $p_{int} = 1.0$

Concernant la performance sur les concepts, les performances suivent à peu près la même dépendance sur p_{int} que les performances sur la tâche, mais avec les meilleures performances obtenues pour des valeur de p_{int} plus élevées. Sur le jeu de données CelebA, les meilleures performances sur les concepts sont obtenues pour $p_{int} = 0.4$ (86.75% de précision), avec une chute considérable pour $p_{int} = 1.0$ (83.96% de précision). Pour les jeux de données Synthetic et COCO@11, les meilleures performances sont obtenues pour $p_{int} = 0.9$ (86.49% de précision) et $p_{int} = 0.8$ (88.31% de précision) respectivement.

La dernière colonne de la Figure 8.8 montre que les performances après interventions sur les concepts sont d'autant plus bonne que la probabilité p_{int} est élevée. Cette remarque est plus perceptible lorsque l'on intervient sur 100% de concept, avec une performance sur la tâche qui approche les 100%. Dans nos expériences, nous avons utilisé $p_{int} = 0.25$ afin d'assurer une comparaison équitable avec les autres méthodes. Toutefois, les résultats de la Figure 8.8 suggèrent que l'utilisation de différentes valeurs de p_{int} pourrait permettre d'obtenir des performances nettement meilleures.

8.5.5 Étude de l'impact de la confiance des règles dans l'échantillon de test

La pertinence de la confiance des règles a été évaluée sur l'échantillon de test du jeu de données Synthetic, qui comprend 157 règles de classe (dont seulement 13 ont une confiance égale à 1) et 33 règles de concepts (dont 32 avec une confiance de 1). À partir de ce jeu de données, 11 versions modifiées ont été créées : les images restent inchangées, mais l'ensemble des règles est ajusté. Dans le premier jeu, aucune règle n'a une confiance de 1 dans l'échantillon de test (0%), puis cette proportion est augmentée progressivement par paliers de 10% jusqu'à atteindre 100% de règles dans la onzième version. À chaque étape, les règles (qu'elles concernent les classes ou les concepts) sont sélectionnées aléatoirement. La Figure 8.9 illustre les résultats obtenus après cinq initialisations pour chaque configuration. Les deux premières colonnes montrent respectivement la précision sur la tâche principale et sur les prédictions de concepts, tandis que la troisième colonne représente la précision sur la tâche après intervention sur les concepts, avec un nombre croissant de groupes de concepts modifiés. La Table 8.12 retranscrit ces résultats de façon numérique. Les résultats révèlent que plus le pourcentage de règles ayant une confiance de 1 augmente, plus la précision sur la tâche s'améliore (première de la Figure 8.9). En revanche, cette corrélation n'est pas aussi marquée pour la précision sur les concepts (deuxième de la Figure 8.9), qui atteint un pic lorsque 50% à 70% des règles ont une confiance de 1 dans l'échantillon de test.

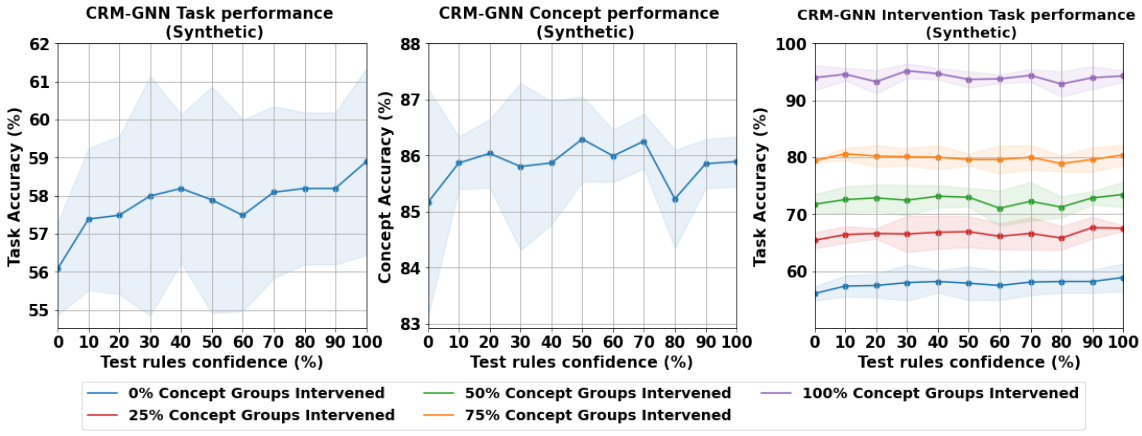


FIGURE 8.9 – Étude de l'impact de la confiance des règles sur l'échantillon de test du jeu de données Synthetic.

L'impact des paramètres `minsup` (support minimal) et `minconf` (confiance minimale) sur la génération des règles de concepts (présenté dans le Chapitre 6) ainsi que leur influence sur les performances du modèle a été analysé. Les résultats obtenus sur le jeu de données Synthetic sont illustrés dans la Figure 8.10, où `minsup` est représenté sur l'axe vertical (en

pourcentage) et minconf sur l'axe horizontal (en pourcentage). La Table 8.11 précise le nombre de règles de concepts générées pour chaque combinaison de ces paramètres. Les modèles ont été entraînés avec l'intégration du module de raisonnement sur les concepts et évalués avec trois initialisations aléatoires différentes.

La Figure 8.10 est divisée en deux colonnes par niveau de sortie : la précision sur les concepts (colonne de gauche, en bleu) et la précision sur la tâche (colonne de droite, en rouge). Les deux lignes de la figure différencient les performances avant (CRM-CNN) et après le raisonnement sur les concepts (CRM-GNN). L'un des constats majeurs est que, quelle que soit la configuration des paramètres, l'utilisation du *concept reasoner* améliore systématiquement les performances en comparaison avec l'absence de raisonnement (CRM-W), ce qui valide son efficacité sur la précision des concepts et des tâches.

L'analyse détaillée révèle que la précision sur les concepts reste relativement stable malgré les variations de minsup et minconf, indiquant que les prédictions sur les concepts sont peu sensibles à ces seuils. En revanche, la précision sur la tâche est plus affectée par ces paramètres. Le modèle CRM-CNN affiche de meilleures performances lorsque minsup est faible et minconf élevé. Le modèle CRM-GNN suit une tendance similaire, tout en montrant des performances plus stables, avec moins de fluctuations. Ces résultats soulignent l'importance de sélectionner des règles pertinentes avec un bon équilibre entre quantité et qualité, afin d'optimiser les performances du modèle.

		minconf				
		1%	5%	10%	15%	20%
minsup	10%	1 729	431	124	38	32
	20%	1 129	341	114	38	22
	30%	725	214	95	34	21
	40%	478	141	67	25	16
	50%	363	101	44	17	11
	60%	273	68	30	16	10
	70%	212	46	17	7	3
	80%	190	39	15	5	2
	90%	162	32	10	3	2
	100%	159	29	9	3	2

TABLE 8.11 – Nombre de règles de concepts générées en fonction des paramètres minsup et minconf sur le jeu de données Synthetic.

	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Tâche	56.08 ± 1.21	57.39 ± 1.86	57.49 ± 2.07	57.99 ± 3.15	58.19 ± 1.96	57.89 ± 2.97	57.48 ± 2.51	58.09 ± 2.26	<u>58.19 ± 1.95</u>	58.19 ± 1.99	**58.89 ± 2.45**
Concept	85.16 ± 2.03	85.87 ± 0.47	86.04 ± 0.61	85.80 ± 1.49	85.87 ± 1.10	**86.29 ± 0.76**	85.99 ± 0.47	<u>86.26 ± 0.49</u>	85.23 ± 0.87	85.85 ± 0.44	85.89 ± 0.45

TABLE 8.12 – Étude de l'impact de la confiance (en %) des règles sur l'échantillon de test du jeu de données Synthetic. Pour la tâche et pour les concepts, la meilleure performance est en gras et la deuxième meilleure performance est soulignée.

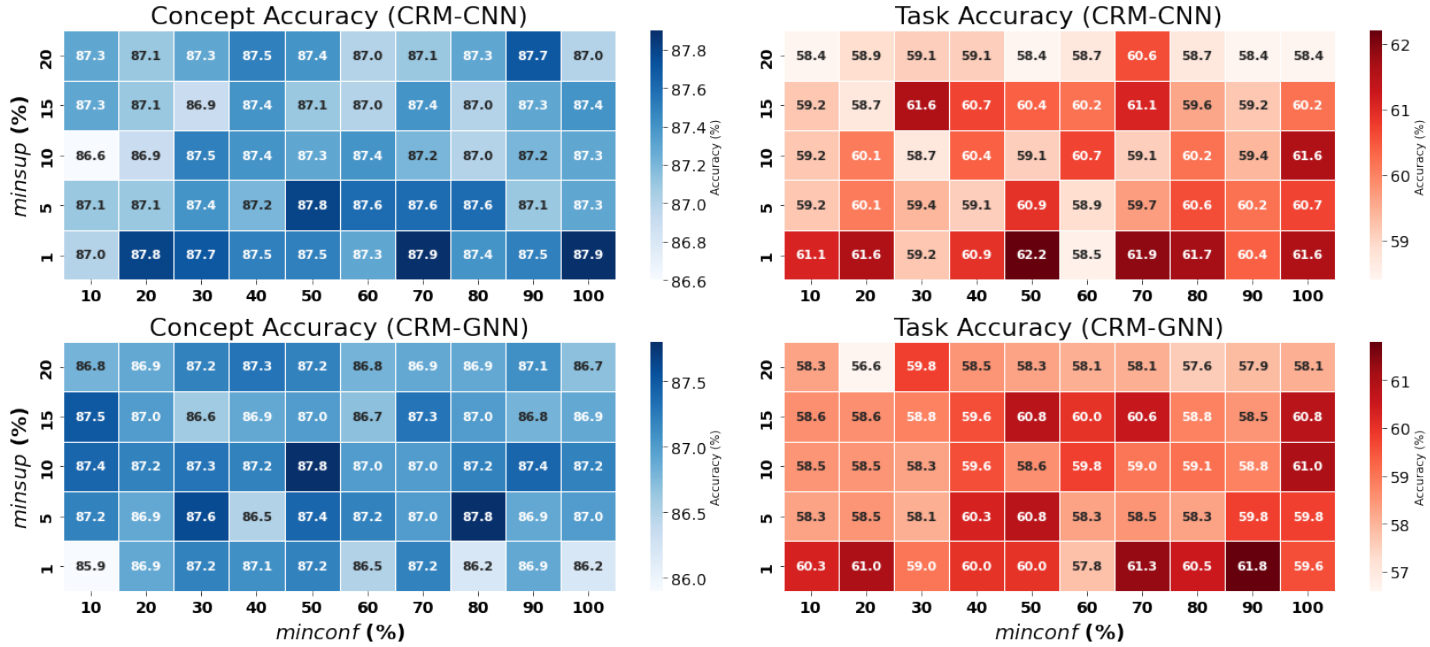


FIGURE 8.10 – Évaluation de l'influence des paramètres `minsup` et `minconf` sur la génération des règles de concepts sur le jeu de données Synthetic. La colonne de gauche (en bleu) indique la précision sur les concepts, tandis que la colonne de droite (en rouge) indique la précision sur la tâche. La première ligne correspond aux sorties de CRM-CNN, et la seconde ligne à celles de CRM-GNN.

8.5.6 Consommation des ressources de calcul

Dans la Table 8.13, les temps d'exécution pour une seule époque d'entraînement de chaque modèle sont présentés, obtenus en entraînant cinq modèles initialisés différemment, ainsi que le nombre de paramètres pour chaque modèle. Les résultats montrent qu'en comparaison avec les autres modèles, CRM prend légèrement plus de temps par époque d'entraînement, en fonction de la taille du graphe créé à partir des règles. Cela s'explique par le fait que le nombre de paramètres dans le raisonneur de concepts dépend du nombre d'arcs dans le graphe. Par conséquent, à mesure que la taille du graphe augmente, le nombre de paramètres augmente par rapport aux méthodes de référence. Cependant, les temps d'entraînement sont fortement variables et dépendent de manière significative de l'implémentation et du matériel, y compris des facteurs tels que la taille des lots et les spécifications de la machine. Cette variabilité suggère que les différences observées dans le temps d'entraînement sont peu susceptibles d'être statistiquement significatives. La Table 8.13 montre également que CRM n'est généralement pas le modèle avec le plus grand nombre de paramètres.

	Modèle	Temps d'entraînement par époque (s)	Nombre de Paramètres
Synthetic	CEM	12.80 ± 0.58	2.23×10^7
	Joint CBM-Sigmoid	12.85 ± 0.57	2.13×10^7
	Joint CBM-Logit	13.00 ± 0.49	2.13×10^7
	IntCEM	12.41 ± 0.16	2.24×10^7
	CRM	13.09 ± 0.53	2.20×10^7
COCO@11	CEM	122.73 ± 2.04	2.26×10^7
	Joint CBM-Sigmoid	122.40 ± 1.93	2.20×10^7
	Joint CBM-Logit	122.49 ± 2.12	2.13×10^7
	IntCEM	122.70 ± 2.17	2.27×10^7
	CRM	125.36 ± 1.2	2.22×10^7
COCO@24	CEM	61.91 ± 0.25	2.30×10^7
	Joint CBM-Sigmoid	61.35 ± 1.17	2.13×10^7
	Joint CBM-Logit	60.13 ± 2.97	2.13×10^7
	IntCEM	50.3 ± 0.51	2.31×10^7
	CRM	63.73 ± 0.47	2.24×10^7
COCO@48	CEM	154.02 ± 8.67	2.37×10^7
	Joint CBM-Sigmoid	153.82 ± 10.84	2.13×10^7
	Joint CBM-Logit	152.32 ± 8.85	2.13×10^7
	IntCEM	149.19 ± 11.48	2.39×10^7
	CRM	157.83 ± 2.45	2.29×10^7
CelebA	CEM	5.03 ± 0.39	2.20×10^7
	Joint CBM-Sigmoid	4.99 ± 0.34	2.13×10^7
	Joint CBM-Logit	4.99 ± 0.35	2.13×10^7
	IntCEM	7.21 ± 0.43	2.21×10^7
	CRM	5.87 ± 0.55	2.28×10^7
CUB	CEM	264.71 ± 2.42	2.57×10^7
	Joint CBM-Sigmoid	264.55 ± 2.99	2.14×10^7
	Joint CBM-Logit	262.46 ± 3.25	2.14×10^7
	IntCEM	261.9 ± 2.9	2.60×10^7
	CRM	267.45 ± 0.79	2.50×10^7

TABLE 8.13 – Temps d'entraînement moyen par époque et nombre de paramètres pour toutes les méthodes.

8.6 Conclusion

L'apprentissage profond a réalisé des avancées significatives dans la classification supervisée d'images, mais l'explicabilité reste un défi majeur pour établir la confiance, faciliter le dépannage et garantir la conformité aux réglementations. Les modèles basés sur les concepts offrent une solution intéressante en associant les prédictions des modèles profonds à des concepts de haut niveau compréhensibles par l'humain, permettant ainsi des interventions sur les concepts mal classifiés afin d'améliorer potentiellement les performances sur la tâche. Le *Concept-Based Reasoning Model* (CRM) présenté dans cet article intègre des connaissances basées sur des règles à travers une couche d'inférence, renforçant ainsi les capacités de généralisation des modèles basés sur les concepts. Nos résultats, validés sur des jeux de données synthétiques et réels, montrent que CRM atteint des performances compétitives en termes de précision des tâches et d'efficacité sur les concepts. De plus, il permet des interventions humaines en phase de test, affinant ainsi les prédictions grâce aux retours d'experts.

Chapitre 9

Conclusion

Dans ce chapitre, nous revenons sur l'ensemble des travaux présentés au cours de cette thèse. Nous en proposons une synthèse globale, en mettant en lumière les contributions principales ainsi que les enseignements tirés. Nous discutons également des limites rencontrées, avant de suggérer plusieurs perspectives de recherche envisageables à la lumière des résultats obtenus.

Contents

Cette thèse s'inscrit dans le contexte de la vision par ordinateur et de l'apprentissage profond, deux domaines qui ont démontré une efficacité remarquable pour résoudre des tâches complexes telles que la classification d'images. Toutefois, malgré ces avancées, plusieurs limitations subsistent, notamment en matière de généralisation, d'explicabilité et de robustesse face à des contextes de données rares ou imprécises. Ces limites s'expliquent en partie par la nature strictement statistique des modèles profonds, qui exploitent les corrélations des données sans en comprendre réellement le sens. C'est dans ce cadre que s'est inscrite notre problématique, qui s'intéresse à intégrer des connaissances a priori dans des architectures d'apprentissage profond afin d'en améliorer les performances, la robustesse et l'explicabilité, en particulier pour des tâches de classification d'images difficiles comme la classification à grain fin.

Le travail de recherche que nous avons déroulé dans cette thèse a d'abord posé le contexte général de la vision par ordinateur et a exposé les motivations scientifiques et applicatives qui justifient l'intérêt pour l'intégration de connaissances dans les architectures profondes. Nous avons ensuite défini les fondements nécessaires en rappelant les principes de l'apprentissage profond, les approches de compilation de connaissances et les approches de plongement de graphes de connaissances. Une attention particulière a été portée aux défis posés par la classification à grain fin, et à l'intégration de connaissances dans les architectures profondes. Enfin, les chapitres suivants ont été consacrés à nos contributions originales, incluant la construction de jeux de données enrichis de connaissances, le développement d'une méthode d'intégration de graphes de connaissances pour la classification d'images, et la proposition d'un cadre de classification basé sur les concepts visant à améliorer l'explicabilité tout en préservant les performances.

9.1 Synthèse des contributions

Les contributions présentées dans le cadre de cette thèse s'articulent autour de trois axes majeurs, chacun développé dans un chapitre dédié. Nous avons tout d'abord proposé une méthodologie générique pour la création de jeux de données innovants dans le domaine de la classification d'images, associés à des connaissances, conçus spécifiquement pour permettre l'intégration de connaissances a priori dans les architectures de classifications. Ces jeux de données, construits à partir de bases existantes utilisées pour des tâches telles que la classification multi-étiquette ou la détection d'objets, ont été enrichis par des règles symboliques, facilitant ainsi leur exploitation dans des approches hybrides combinant apprentissage statistique et raisonnement symbolique. Ces jeux de données nouvellement créés ont par ailleurs été exploités pour évaluer les contributions suivantes de la thèse.

Dans un second temps, nous avons introduit une méthode d'apprentissage profond, conçue pour intégrer explicitement des connaissances a priori sous forme de graphes de connaissances, lesquels peuvent être construits à partir d'attributs sémantiques. Cette méthode, générique parce qu'elle peut être utilisée dans n'importe quelle architecture profonde, repose sur une fonction de perte hybride, combinant l'entropie croisée couramment utilisée pour la classification supervisée, avec un nouveau terme de régularisation fondée sur la proximité dans l'espace des représentations obtenues après le plongement du graphe. Cette formulation vise à informer l'architecture profonde sur la difficulté de créer des frontières

de décision entre des classes. Les résultats expérimentaux obtenus sur plusieurs jeux de données démontrent l'efficacité de cette approche, avec une amélioration parfois significative des performances de classification par rapport à des modèles concurrents. En outre, cette méthode présente l'avantage de ne pas complexifier le processus d'apprentissage en termes de temps de calcul ou de ressources.

Enfin, la dernière contribution de cette thèse porte sur l'amélioration du pouvoir de généralisation des modèles à travers un cadre basé sur les concepts. En partant du constat que ces approches souffrent généralement d'une baisse de performance au profit de la transparence, nous avons conçu une solution qui tire parti de graphes de connaissances comme outils de raisonnement, exploitant des règles entre concepts et classes pour enrichir les représentations apprises. Cette approche permet de réconcilier explicabilité et efficacité, en renforçant l'explicabilité des prédictions sans compromettre les résultats en termes de classification. L'ensemble de ces contributions s'inscrit dans une volonté de rendre les modèles d'apprentissage profond à la fois plus compréhensibles et plus robustes, en les dotant de mécanismes de raisonnement inspirés des approches symboliques. Les résultats obtenus sur les différents jeux de données démontrent la pertinence de notre approche non seulement sur la tâche de classification, mais aussi dans les interventions pendant la phase de test. En particulier, le modèle montre une capacité à affiner ses décisions lorsque des retours experts sont fournis sur certains concepts, soulignant l'intérêt d'une collaboration homme–machine facilitée par la transparence et le raisonnement. Cependant, la comparaison avec les performances obtenues dans la contribution précédente met en évidence l'écart persistant entre les résultats atteints avec ou sans l'introduction d'un goulot d'étranglement pour les concepts. Les modèles basés sur les concepts, bien qu'ils favorisent l'explicabilité en imposant une contrainte sur la représentation intermédiaire, peut limiter la capacité du modèle à capturer certaines subtilités des données. Néanmoins, un cas particulier mérite d'être souligné sur le jeu de données Synthetic, où les performances s'avèrent supérieures lorsqu'on utilise une représentation explicite par les concepts. Cela suggère que, dans des contextes où les relations entre concepts sont bien définies et peu ambiguës, le recours à une couche de concepts peut non seulement améliorer l'explicabilité, mais aussi renforcer l'efficacité du modèle. Ce constat ouvre la voie à une réflexion plus fine sur l'adéquation entre la structure des données et le niveau de contrainte imposé par les approches basées sur les concepts.

9.2 Perspectives

9.2.1 Graphe de connaissances pour la classification d'images

Un élément central de notre méthode réside dans la construction et l'intégration d'un graphe de connaissances au sein de l'architecture d'apprentissage profond. Dans le cadre de ce travail, nous avons opté pour des graphe non dirigés et pondérés, ce qui permet de capturer des relations de similarité entre les nœuds de manière symétrique. Toutefois, cette symétrie pourrait constituer une limitation, notamment pour l'ajout d'informations hiérarchiques. L'introduction d'arêtes orientées dans le graphe pourrait donc permettre de mieux modéliser la structure entre les différents types de nœuds, et ainsi améliorer la qualité des représentations apprises. Nous prévoyons d'explorer cette piste dans nos futurs travaux,

notamment à travers l'utilisation de graphes orientés.

Un autre enjeu important est celui de l'évolutivité. Dans notre approche actuelle, le graphe est construit à partir de l'ensemble des images d'apprentissage, ce qui fonctionne bien sur des jeux de données de taille modérés, mais devient difficilement tenable à grande échelle. Pour faire face à cette limitation, nous envisageons de développer des stratégies de compression du graphe, notamment par la sélection d'images prototypes représentant efficacement la diversité des classes et des images. Une telle réduction permettrait de conserver les bénéfices de la structure sémantique tout en réduisant considérablement les coûts de calcul et la mémoire nécessaire à l'apprentissage.

Par ailleurs, dans nos expérimentations, nous avons utilisé une taille de plongement fixée à 128 dimensions pour représenter les nœuds du graphe dans l'espace de représentation. Si ce choix s'est révélé efficace pour nos expérimentations, il reste arbitraire et mériterait une étude plus approfondie. Il serait pertinent d'évaluer l'impact de différentes tailles de plongement sur la qualité des représentations, les performances de classification et la capacité de généralisation. Une telle analyse permettrait non seulement d'optimiser les performances de l'approche, mais également de mieux comprendre les compromis entre complexité, expressivité et robustesse des représentations intégrant les connaissances.

Enfin, une perspective à plus long terme consisterait à rendre la construction du graphe de connaissances dynamique et adaptative. Plutôt que de le définir manuellement ou en amont, le graphe pourrait évoluer au cours de l'apprentissage, en fonction des erreurs du modèle ou de l'incertitude des prédictions. Cette évolution du graphe et du réseau de neurones ouvrirait la voie à des systèmes capables de structurer eux-mêmes leurs connaissances pour améliorer leur performance, et peut-être ouvrir la voix vers une approche explicable.

9.2.2 Classification basée sur des concepts intégrant des règles

Malgré le compromis entre précision et explicabilité offert par le modèle CRM, certaines limitations importantes persistent, tant au niveau de la représentation que sur le raisonnement sur les concepts. Dans sa version actuelle, le modèle encode la présence des concepts via les modules d'*embedding generator*, qui produisent une représentation vectorielle transmise ensuite au *concept reasoner*. Ce dernier s'appuie sur un raisonnement logique fondé sur la logique floue, en supposant que plus la valeur d'un plongement est élevée, plus il est probable que le concept correspondant soit effectivement présent dans l'image. Cette hypothèse de correspondance directe entre activation numérique et présence de concept fonctionne relativement bien dans certains cas, mais elle reste fragile, notamment lorsque l'encodeur utilise des valeurs élevées pour potentiellement encoder l'absence du concept. Pour pallier cette limitation, nous envisageons dans des travaux futurs de concevoir une nouvelle fonction de perte spécifiquement adaptée à l'encodage des concepts, visant à séparer de manière plus guidée les représentations des concepts présents et absents. Cette fonction devra guider les générateurs de plongement à produire des vecteurs sémantiquement cohérents avec le raisonnement fait par le *concept reasoner*.

Aussi, contrairement aux travaux antérieurs qui modélisent à la fois la présence et la non-présence des concepts dans les images, notre approche se concentre uniquement sur la présence des concepts. Ce choix est motivé par le fait que notre mécanisme de raisonnement

repose exclusivement sur la validation de règles logiques exprimant la co-occurrence de concepts dans les images. Cependant, cette focalisation sur la seule présence constitue également une limite. En effet, la non-présence de certains concepts dans une image peut s'avérer tout aussi informative pour décider de sa classe, ou même de la présence (ou de la non-présence) d'autres concepts dans l'image. Une piste d'amélioration importante serait donc d'enrichir notre approche pour représenter également la non-présence explicite des concepts, et d'introduire des négations dans les règles de raisonnement, permettant au modèle d'étendre son champ de raisonnement logique.

Enfin, pour évaluer la robustesse et la capacité de passage à l'échelle de notre approche, nous prévoyons de l'appliquer à des jeux de données réels de grande taille. Une telle évaluation nous permettrait non seulement de tester les limites de CRM dans un cadre plus réaliste, mais aussi de mieux comprendre ses performances dans d'autres conditions d'apprentissage.

Bibliographie

Reza Abbasi-Asl and Bin Yu. Structural compression of convolutional neural networks. arXiv preprint arXiv :1705.07356, 2017.

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop : Higher-order graph convolutional architectures via sparsified neighborhood mixing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 21–29. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/abu-el-haija19a.html.

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. Social Networks, 25(3) :211–230, 2003. ISSN 0378-8733. doi : https://doi.org/10.1016/S0378-8733(03)00009-1. URL https://www.sciencedirect.com/science/article/pii/S0378873303000091.

Abien Fred Agarap. Deep learning using rectified linear units (relu). ArXiv, abs/1803.08375, 2018.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE, 10(7) :1–46, 07 2015. doi : 10.1371/journal.pone.0130140. URL https://doi.org/10.1371/journal.pone.0130140.

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lió, Frederic Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In International Conference on Machine Learning, pages 1801–1825. PMLR, 2023.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection : Quantifying interpretability of deep visual representations. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6541–6549, 2017.

Y. Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. volume 60, page 6, 01 2009. doi : 10.1145/1553374.1553380.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

Samuele Bortolotti, Emanuele Marconato, Tommaso Carraro, Paolo Morettin, Emile van Krieken, Antonio Vergari, Stefano Teso, and Andrea Passerini. A neuro-symbolic benchmark suite for concept quality and reasoning shortcuts. Advances in neural information processing systems, 37 :115861–115905, 2024.

Grégory Bourguin and Arnaud Lewandowski. Pizzaïolo dataset : Des images synthétiques ontologiquement explicables. In Atelier Explain'AI à EGC'2024, 2024.

Mohammed Brahimi, Saïd Mahmoudi, Kamel Boukhalfa, and Abdelouahab Moussaoui. Deep interpretable architecture for plant diseases classification. 2019 Signal Processing : Algorithms, Architectures, Arrangements, and Applications (SPA), pages 111–116, 2019. URL https://api.semanticscholar.org/CorpusID:173188061.

Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. arXiv preprint arXiv :1406.2952, 2014.

Sijia Cai, Wangmeng Zuo, and Lei Zhang. Higher-Order Integration of Hierarchical Convolutional Activations for Fine-Grained Visual Categorization . In 2017 IEEE International Conference on Computer Vision (ICCV), pages 511–520, Los Alamitos, CA, USA, October 2017. IEEE Computer Society. doi : 10.1109/ICCV.2017.63. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.63.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep : Learning graph representations with global structural information. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, page 891–900, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337946. doi : 10.1145/2806416.2806512. URL https://doi.org/10.1145/2806416.2806512.

Chetna Chand, Amit Thakkar, and Amit Ganatra. Sequential pattern mining : Survey and current research challenges. International Journal of Soft Computing and Engineering, 2(1) :185–193, 2012.

Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++ : Generalized gradient-based visual explanations for deep convolutional networks. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 839–847, 2017.

Kushal Chauhan, Rishabh Tiwari, Jan Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pages 5948–5955, 2023. ISBN 978-1-57735-880-0.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that : Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf.

Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp : Hierarchical representation learning for networks. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1), Apr. 2018a. doi : 10.1609/aaai.v32i1.11849. URL https://ojs.aaai.org/index.php/AAAI/article/view/11849.

Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of image classification algorithms based on convolutional neural networks. Remote

Sensing, 13(22), 2021. ISSN 2072-4292. doi : 10.3390/rs13224712. URL https://www.mdpi.com/2072-4292/13/22/4712.

Riquan Chen, Tianshui Chen, Xiaolu Hui, Hefeng Wu, Guanbin Li, and Liang Lin. Knowledge graph transfer network for few-shot recognition. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 10575–10582, 2020a.

Shuangshuang Chen and Wei Guo. Auto-encoders in deep learning—a review with new perspectives. Mathematics, 11(8), 2023. ISSN 2227-7390. doi : 10.3390/math11081777. URL https://www.mdpi.com/2227-7390/11/8/1777.

Tianshui Chen, Liang Lin, Riquan Chen, Yang Wu, and Xiaonan Luo. Knowledge-embedded representation learning for fine-grained image recognition. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, page 627–634. AAAI Press, 2018b. ISBN 9780999241127.

Yue Chen, Yalong Bai, Wei Zhang, and Tao Mei. Destruction and construction learning for fine-grained image recognition. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5152–5161, 2019b.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. Nature Machine Intelligence, 2(12) :772–782, 2020b.

Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1153–1162, 2016.

Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge J. Belongie. Kernel pooling for convolutional neural networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3049–3058, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, 2009.

Jia Deng, Jonathan Krause, Michael Stark, and Li Fei-Fei. Leveraging the wisdom of the crowd for fine-grained recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38 :666–676, 2016.

Luc Devroye. Sample-based non-uniform random variate generation. In Proceedings of the 18th conference on Winter simulation, pages 260–265, 1986.

Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. Artificial Intelligence, 244 :143–165, 2017.

Yao Ding, Yanzhao Zhou, Yi Zhu, Qixiang Ye, and Jianbin Jiao. Selective sparse sampling for fine-grained image recognition. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 6598–6607, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words : Transformers for image recognition at scale. ArXiv, abs/2010.11929, 2020.

Ruoyi Du, Dongliang Chang, Ayan Kumar Bhunia, Jiyang Xie, Yi-Zhe Song, Zhanyu Ma, and Jun Guo. Fine-grained visual classification via progressive multi-granularity training of jigsaw patches. In European Conference on Computer Vision, 2020.

Abhimanyu Dubey, Otkrist Gupta, Pei Guo, Ramesh Raskar, Ryan Farrell, and Nikhil Naik. Pairwise confusion for fine-grained visual classification. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018a.

Abhimanyu Dubey, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Maximum-entropy fine grained classification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/0c74b7f78409a4022a2c4c5a5ca3ee19-Paper.pdf.

Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning : A comprehensive survey and benchmark. Neurocomputing, 503 :92–108, 2022.

Nicolas Durand and Bruno Crémilleux. Ecclat : a new approach of clusters discovery in categorical data. In Max Bramer, Alun Preece, and Frans Coenen, editors, Research and Development in Intelligent Systems XIX, pages 177–190, London, 2003. Springer London. ISBN 978-1-4471-0651-7.

Jessica D'souza, PK Aleema, S Dhanyashree, Clita Fernandes, KM Kavitha, and Chandra Naik. Knowledge-based scene graph generation in medical field. In 2023 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), pages 232–237. IEEE, 2023.

Melih Engin, Lei Wang, Luping Zhou, and Xinwang Liu. Deepkspd : Learning kernel-matrix-based spd representation for fine-grained image recognition. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, and Mateja Jamnik. Concept embedding models : Beyond the accuracy-explainability trade-off. Advances in Neural Information Processing Systems, 35, 2022.

Mateo Espinosa Zarlenga, Katie Collins, Krishnamurthy Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help : Intervention-aware concept embedding models. Advances in Neural Information Processing Systems, 36, 2023.

Yuan Fang, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. Object detection meets knowledge graphs. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pages 1661–1667, 2017. doi : 10.24963/ijcai.2017/230. URL https://doi.org/10.24963/ijcai.2017/230.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In Proceedings of the IEEE international conference on computer vision, pages 3429–3437, 2017.

Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. A survey of sequential pattern mining. Data Science and Pattern Recognition, 1(1) :54–77, 2017.

Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better : Recurrent attention convolutional neural network for fine-grained image recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4476–4484, 2017.

Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors. Formal Concept Analysis : foundations and applications. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 3540278915.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 317–326, 2015.

Yu Gao, Xintong Han, Xun Wang, Weilin Huang, and Matthew R. Scott. Channel interaction networks for fine-grained image categorization. ArXiv, abs/2003.05235, 2020.

Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3029–3038, 2019.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

Chen Gong, Dacheng Tao, Jie Yang, and Keren Fu. Signed laplacian embedding for supervised dimension reduction. Proceedings of the AAAI Conference on Artificial Intelligence, 28(1), Jun. 2014. doi : 10.1609/aaai.v28i1.8954. URL https://ojs.aaai.org/index.php/AAAI/article/view/8954.

Ke Gong, Xiaodan Liang, Dongyu Zhang, Xiaohui Shen, and Liang Lin. Look into person : Self-supervised structure-sensitive learning and a new benchmark for human parsing. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 932–940, 2017.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance : A survey. Knowledge-Based Systems, 151 :78–94, 2018.

Alex Graves and Alex Graves. Long short-term memory. Supervised sequence labelling with recurrent neural networks, pages 37–45, 2012.

A. Grover and J. Leskovec. node2vec : Scalable feature learning for networks. In ACM SIGKDD, pages 855–864, 2016.

Michal K Grzeszczyk, Szymon Płotka, Beata Rebizant, Katarzyna Kosińska-Kaczyńska, Michał Lipa, Robert Brawura-Biskupski-Samaha, Przemysław Korzeniowski, Tomasz Trzciński, and Arkadiusz Sitek. Tabattention : Learning attention conditionally on tabular data. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 347–357. Springer, 2023.

Qingji Guan, Yaping Huang, Zhun Zhong, Zhedong Zheng, Liang Zheng, and Yi Yang. Diagnose like a radiologist : Attention guided convolutional neural network for thorax disease classification. ArXiv, abs/1801.09927, 2018.

Christoph Haarburger, Michael Baumgartner, Daniel Truhn, Mirjam Broeckmann, Hannah Schneider, Simone Schrading, Christiane Kuhl, and Dorit Merhof. Multi scale curriculum cnn for context-aware breast mri malignancy classification. Medical Image Computing and Computer Assisted Intervention – MICCAI 2019, page 495–503, 2019. ISSN 1611-3349. doi : 10.1007/978-3-030-32251-9_54. URL http://dx.doi.org/10.1007/978-3-030-32251-9_54.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. Interpretable image recognition with hierarchical prototypes. In Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, volume 7, pages 32–40, 2019.

Ju He, Jie-Neng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, and Changhu Wang. Transfg : A transformer architecture for fine-grained recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 852–860, 2022.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. doi : 10.1109/CVPR.2016.90.

Xiangteng He and Yuxin Peng. Fine-Grained Image Classification via Combining Vision and Language . In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7332–7340, Los Alamitos, CA, USA, July 2017a. IEEE Computer Society. doi : 10.1109/CVPR.2017.775. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.775.

Xiangteng He and Yuxin Peng. Weakly supervised learning of part selection model with spatial constraints for fine-grained image classification. In AAAI Conference on Artificial Intelligence, 2017b.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2261–2269, 2016.

Chae Hwa Yoo, Nayoung Kim, and Je-Won Kang. Relevance regularization of convolutional neural network for interpretable classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization : accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, page 448–456. JMLR.org, 2015.

Naeem Ul Islam and Sukhan Lee. Interpretation of deep cnn based on learning feature reconstruction with feedback weights. IEEE Access, 7 :25195–25208, 2019.

Nikita Jain and Vishal Srivastava. Data mining techniques : a survey paper. IJRET : International Journal of Research in Engineering and Technology, 2(11) :2319–1163, 2013.

Andrew Jesson, Nicolas Guizard, Sina Hamidi Ghalehjegh, Damien Goblot, Florian Soudan, and Nicolas Chapados. Cased : Curriculum adaptive sampling for extreme data imbalance. Lecture Notes in Computer Science, page 639–646, 2017. ISSN 1611-3349. doi : 10.1007/978-3-319-66179-7_73. URL http://dx.doi.org/10.1007/978-3-319-66179-7_73.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In Chengqing Zong and Michael Strube, editors, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers), pages 687–696, Beijing, China, July 2015. Association for Computational Linguistics. doi : 10.3115/v1/P15-1067. URL https://aclanthology.org/P15-1067/.

Ruyi Ji, Longyin Wen, Libo Zhang, Dawei Du, Ynajun Wu, Chen Zhao, Xianglong Liu, and Feiyue Huang. Attention convolutional binary neural tree for fine-grained visual categorization. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10465–10474, 2019.

Chenhan Jiang, Hang Xu, Xiaodan Liang, and Liang Lin. Hybrid knowledge routed modules for large-scale object detection. Advances in Neural Information Processing Systems, 31, 2018.

Amelia Jiménez-Sánchez, Diana Mateus, Sonja Kirchhoff, Chlodwig Kirchhoff, Peter Biberthaler, Nassir Navab, Miguel A. González Ballester, and Gemma Piella. Medical-based deep curriculum learning for improved fracture classification. Medical Image Computing and Computer Assisted Intervention – MICCAI 2019, page 694–702, 2019. ISSN 1611-3349. doi : 10.1007/978-3-030-32226-7_77. URL http://dx.doi.org/10.1007/978-3-030-32226-7_77.

Leo Katz. A new status index derived from sociometric analysis. Psychometrika, 18 :39–43, 1953. URL https://api.semanticscholar.org/CorpusID:121768822.

Dmitry Kazhdan, Botty Dimanov, Mateja Jamnik, Pietro Liò, and Adrian Weller. Now you see me (cme) : concept-based model extraction. arXiv preprint arXiv :2010.13233, 2020.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning : Generalization gap and sharp minima, 2017. URL https://arxiv.org/abs/1609.04836.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision : A survey. ACM computing surveys (CSUR), 54(10s) :1–41, 2022.

Eunji Kim, Dahuin Jung, Sangha Park, Siwon Kim, and Sungroh Yoon. Probabilistic concept bottleneck models. In Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org, 2023.

Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. CoRR, abs/1412.6980, 2014.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL https://arxiv.org/abs/1609.02907.

Reinhard Klette. Concise Computer Vision - An Introduction into Theory and Algorithms. 01 2014. ISBN 978-1-4471-6319-0. doi : 10.1007/978-1-4471-6320-6.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 5338–5348. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/koh20a.html.

Shu Kong and Charless Fowlkes. Low-Rank Bilinear Pooling for Fine-Grained Classification . In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7025–7034, Los Alamitos, CA, USA, July 2017. IEEE Computer Society. doi : 10.1109/CVPR.2017.743. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.743.

Omar Krichen, Simon Corbillé, Éric Anquetil, Nathalie Girard, Élisa Fromont, and Pauline Nerdeux. Combination of explicit segmentation with seq2seq recognition for fine analysis of children handwriting. International journal on document analysis and recognition (IJDAR), 25(4) :339–350, 2022.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome : Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision, 123(1) :32–73, May 2017. ISSN 1573-1405. doi : 10.1007/s11263-016-0981-7. URL https://doi.org/10.1007/s11263-016-0981-7.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Neeraj Kumar, Alexander Berg, Peter Belhumeur, and Shree Nayar. Attribute and simile classifiers for face verification. pages 365 – 372, 11 2009. doi : 10.1109/ICCV.2009.5459250.

Christoph Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. 06 2009. doi : 10.1109/CVPR.2009.5206594.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1 (4) :541–551, 1989. doi : 10.1162/neco.1989.1.4.541.

Arthur Ledaguenel, Céline Hudelot, and Mostepha Khouadjia. Improving neural classification with logical prior knowledge. In Workshop on Composite AI (CompAI), European Conference on Artificial Intelligence (ECAI), 2024.

Liu Li, Mai Xu, Xiaofei Wang, Lai Jiang, and Hanruo Liu. Attention based glaucoma detection : A large-scale database and cnn model. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10571–10580, 2019.

P. Li, Jiangtao Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition ? 2017 IEEE International Conference on Computer Vision (ICCV), pages 2089–2097, 2017.

Yikai Li, CL Philip Chen, and Tong Zhang. A survey on siamese network : Methodologies, applications, and opportunities. IEEE Transactions on artificial intelligence, 3(6) :994–1014, 2022.

Zenan Li, Yunpeng Huang, Zhaoyu Li, Yuan Yao, Jingwei Xu, Taolue Chen, Xiaoxing Ma, and Jian Lü. Neuro-symbolic learning yielding logical constraints. In Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

Haoyu Liang, Zhihao Ouyang, Yuyuan Zeng, Hang Su, Zihao He, Shu-Tao Xia, Jun Zhu, and Bo Zhang. Training interpretable convolutional neural networks by differentiating class-specific filters. In Computer Vision – ECCV 2020 : 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II, page 622–638, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58535-8. doi : 10.1007/978-3-030-58536-5_37. URL https://doi.org/10.1007/978-3-030-58536-5_37.

Di Lin, Xiaoyong Shen, Cewu Lu, and Jiaya Jia. Deep lac : Deep localization, alignment and classification for fine-grained recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1666–1674, 2015a.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. CoRR, abs/1312.4400, 2013.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO : common objects in context. CoRR, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. 2015 IEEE International Conference on Computer Vision (ICCV), pages 1449–1457, 2015b.

Drew Linsley, Dan Shiebler, Sven Eberhardt, and Thomas Serre. Learning what and where to attend, 2019. URL https://arxiv.org/abs/1805.08819.

Chuanbin Liu, Hongtao Xie, Zheng-Jun Zha, Lingfeng Ma, Lingyun Yu, and Yongdong Zhang. Filtration and distillation : Enhancing region attention for fine-grained visual categorization. Proceedings of the AAAI Conference on Artificial Intelligence, 34(07) : 11555–11562, Apr. 2020. doi : 10.1609/aaai.v34i07.6822. URL https://ojs.aaai.org/index.php/AAAI/article/view/6822.

Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. The treasure beneath convolutional layers : Cross-convolutional-layer pooling for image classification. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4749–4757, 2014.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

Max Maria Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output : Semantic bottleneck networks. ArXiv, abs/1907.10882, 2019.

Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Lio'. Gcexplainer : Human-in-the-loop concept-based explanations for graph neural networks. ArXiv, abs/2107.11889, 2021.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. arXiv preprint arXiv :2106.13314, 2021.

Gabriel Maicas, Andrew P. Bradley, Jacinto C. Nascimento, Ian Reid, and Gustavo Carneiro. Training medical image analysis systems like radiologists, 2019.

Gary Marcus. Deep learning : A critical appraisal. arXiv preprint arXiv :1801.00631, 2018.

Franck Anaël Mbiaya, Christel Vrain, Frédéric Ros, Yves Lucas, et al. Dataset for image classification with knowledge. Data in Brief, 57 :110893, 2024a.

Franck Anaël Mbiaya, Christel Vrain, Frédéric Ros, and Yves Lucas. Kgic : Intégration de graphe de connaissances pour la classification d'images. Revue des Nouvelles Technologies de l'Information, Extraction et Gestion des Connaissances, RNTI-E-39 : 259–272, 2023.

Franck Anaël Mbiaya, Christel Vrain, Frédéric Ros, Thi-Bich-Hanh Dao, and Yves Lucas. Knowledge graph-based image classification. Data & Knowledge Engineering, 151 :102285, 2024b. ISSN 0169-023X. doi : https://doi.org/10.1016/j.datak.2024.102285. URL https://www.sciencedirect.com/science/article/pii/S0169023X24000090.

Carlos E. Mendoza-Ramírez, Juan C. Tudon-Martinez, Luis C. Félix-Herrán, Jorge de J. Lozoya-Santos, and Adriana Vargas-Martínez. Augmented reality : Survey. Applied Sciences, 13(18), 2023. ISSN 2076-3417. doi : 10.3390/app131810491. URL https://www.mdpi.com/2076-3417/13/18/10491.

Cyril Meurie, Olivier Lézoray, Christophe Coniglio, and Marion Berbineau. Graph-based people segmentation using a genetically optimized combination of classifiers. Journal of Electronic Imaging, 27(5) :051210–051210, 2018.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Marvin Minsky and Seymour Papert. An introduction to computational geometry. Cambridge tiass., HIT, 479(480) :104, 1969.

Masahiro Mitsuhara, Hiroshi Fukui, Yusuke Sakashita, Takanori Ogata, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. Embedding human knowledge into deep neural network via attention map. In VISIGRAPP, 2019.

Pietro Morbidelli, Diego Carrera, Beatrice Rossi, Pasqualina Fragneto, and Giacomo Boracchi. Augmented grad-cam : Heat-maps super resolution through augmentation. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4067–4071. IEEE, 2020.

W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance : Contextual decomposition to extract interactions from lstms. arXiv preprint arXiv :1801.05453, 2018.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, page 3395–3403, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldemariam. Smooth grad-cam++ : An enhanced inference level visualization technique for deep convolutional neural network models. ArXiv, abs/1908.01224, 2019.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 1105–1114, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi : 10.1145/2939672.2939751. URL https://doi.org/10.1145/2939672.2939751.

Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors : Towards confident, interpretable and robust deep learning. arXiv preprint arXiv :1803.04765, 2018.

Darsh Parekh, Nishi Poddar, Aakash Rajpurkar, Manisha Chahal, Neeraj Kumar, Gyanendra Prasad Joshi, and Woong Cho. A review on autonomous vehicles : Progress, methods and challenges. Electronics, 11(14), 2022. ISSN 2079-9292. doi : 10.3390/electronics11142162. URL https://www.mdpi.com/2079-9292/11/14/2162.

Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The sun attribute database : Beyond categories for deeper scene understanding. International Journal of Computer Vision, 108(1) :59–81, May 2014. ISSN 1573-1405. doi : 10.1007/s11263-013-0695-z. URL https://doi.org/10.1007/s11263-013-0695-z.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk : online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, page 701–710. ACM, August 2014. doi : 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732.

Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence : A survey. arXiv preprint arXiv :2312.12936, 2023.

Harish Guruprasad Ramaswamy et al. Ablation-cam : Visual explanations for deep convolutional network via gradient-free localization. In proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 983–991, 2020.

Scott E. Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 49–58, 2016.

Shaoqing Ren. Faster r-cnn : Towards real-time object detection with region proposal networks. arXiv preprint arXiv :1506.01497, 2015.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you ?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.

Laura Rieger, Chandan Singh, W. James Murdoch, and Bin Yu. Interpretations are useful : penalizing explanations to align neural networks with prior knowledge. In Proceedings of the 37th International Conference on Machine Learning, ICML'20. JMLR.org, 2020.

Mattia Rigotti, Christoph Miksovic, Ioana Giurgiu, Thomas Gschwind, and Paolo Scotton. Attention-based interpretability with concept transformers. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=kAa9eDS0RdO.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. Psychological review, 65 6 :386–408, 1958.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. Nature, 323(6088) :533–536, Oct 1986. ISSN 1476-4687. doi : 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

Lars Schmarje, Monty Santarossa, Simon-Martin Schröder, and Reinhard Koch. A survey on semi-, self-and unsupervised learning for image classification. IEEE Access, 9 : 82146–82168, 2021.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam : Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision, pages 618–626, 2017.

Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure of concept bottleneck models. In Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022, 2022. URL https://openreview.net/forum?id=PUspzfGsgY.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, page 3145–3153. JMLR.org, 2017.

Marcel Simon and Erik Rodner. Neural activation constellations : Unsupervised part model discovery with convolutional networks. 2015 IEEE International Conference on Computer Vision (ICCV), pages 1143–1151, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

Junho Song, Jiwon Son, Dong-hyuk Seo, Kyungsik Han, Namhyuk Kim, and Sang-Wook Kim. St-gat : A spatio-temporal graph attention network for accurate traffic speed prediction. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM '22, page 4500–4504, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi : 10.1145/3511808.3557705. URL https://doi.org/10.1145/3511808.3557705.

Kaitao Song, Xiu-Shen Wei, Xiangbo Shu, Ren-Jie Song, and Jianfeng Lu. Bi-modal progressive mask attention for fine-grained recognition. IEEE Transactions on Image Processing, 29 :7006–7018, 2020.

Yisheng Song, Ting Wang, Puyu Cai, Subrota K Mondal, and Jyoti Prakash Sahoo. A comprehensive survey of few-shot learning : Evolution, applications, challenges, and opportunities. ACM Computing Surveys, 55(13s) :1–40, 2023.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity : The all convolutional net. CoRR, abs/1412.6806, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1) :1929–1958, 2014.

Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In AAAI Conference on Artificial Intelligence, 2016.

Hongbo Sun, Xiangteng He, and Yuxin Peng. Sim-trans : Structure information modeling transformer for fine-grained visual categorization. In Proceedings of the 30th ACM International Conference on Multimedia, MM '22, page 5853–5861, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392037. doi : 10.1145/3503161.3548308. URL https://doi.org/10.1145/3503161.3548308.

Ming Sun, Yuchen Yuan, Feng Zhou, and Errui Ding. Multi-attention multi-class constraint for fine-grained image recognition. ArXiv, abs/1806.05372, 2018a.

Zhiqing Sun, Zhihong Deng, Jian-Yun Nie, and Jian Tang. Rotate : Knowledge graph embedding by relational rotation in complex space. ArXiv, abs/1902.10197, 2018b.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In International conference on machine learning, pages 3319–3328. PMLR, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, Proceedings of the 30th International Conference on Machine Learning, volume 28 of Proceedings of Machine Learning Research, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

Laszlo Szathmary. Symbolic Data Mining Methods with the Coron Platform. (Méthodes symboliques de fouille de données avec la plate-forme Coron). PhD thesis, Henri Poincaré University, Nancy, France, 2006a. URL https://tel.archives-ouvertes.fr/tel-00336374.

Laszlo Szathmary. Symbolic Data Mining Methods with the Coron Platform. Theses, Université Henri Poincaré - Nancy 1, November 2006b. URL https://theses.hal.science/tel-01754284.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, D. Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, 2014.

Martin Tamajka, Wanda Benesova, and Matej Kompanek. Transforming convolutional neural network to an interpretable classifier. 2019 International Conference on Systems, Signals and Image Processing (IWSSIP), pages 255–259, 2019.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In Artificial Neural Networks and Machine Learning–ICANN 2018 : 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27, pages 270–279. Springer, 2018.

Amirhossein Tavanaei. Embedded encoder-decoder in convolutional networks towards explainable ai. ArXiv, abs/2007.06712, 2020.

Hans Thisanke, Chamli Deshan, Kavindu Chamith, Sachith Seneviratne, Rajith Vidanaarachchi, and Damayanthi Herath. Semantic segmentation using vision transformers : A survey. Engineering Applications of Artificial Intelligence, 126 :106669, 2023. ISSN 0952-1976. doi : https://doi.org/10.1016/j.engappai.2023.106669. URL https://www.sciencedirect.com/science/article/pii/S0952197623008539.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, page 2071–2080. JMLR.org, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

Francesco Ventura and Tania Cerquitelli. What's in the box ? explaining the black-box model through an evaluation of its interpretable features. ArXiv, abs/1908.04348, 2019.

Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Transactions on Knowledge and Data Engineering, 35(1) : 614–633, 2021.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. California Institute of Technology, 2011.

Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3156–3164, 2017a.

Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM : Score-Weighted Visual Explanations for Convolutional Neural Networks . In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 111–119, Los Alamitos, CA, USA, June 2020a. IEEE Computer Society. doi : 10.1109/CVPRW50498.2020.00020. URL https://doi.ieeecomputersociety.org/10.1109/CVPRW50498.2020.00020.

Jian Wang, Hengde Zhu, Shui-Hua Wang, and Yu-Dong Zhang. A review of deep learning on medical image analysis. Mobile Networks and Applications, 26(1) :351–380, Feb 2021a. ISSN 1572-8153. doi : 10.1007/s11036-020-01672-7. URL https://doi.org/10.1007/s11036-020-01672-7.

Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn : A unified framework for multi-label image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016a.

Jun Wang, Xiaohan Yu, and Yongsheng Gao. Feature fusion vision transformer for fine-grained visual categorization. In British Machine Vision Conference, 2021b.

Kun Wang, Xiaohong Zhang, Sheng Huang, Feiyu Chen, Xiangbo Zhang, and Luwen Huangfu. Learning to recognize thoracic disease in chest x-rays with knowledge-guided deep zoom neural networks. IEEE Access, 8 :159790–159805, 2020b.

Qilong Wang, P. Li, and Lei Zhang. G2denet : Global gaussian distribution embedding network and its application to visual recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6507–6516, 2017b.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, and Ronald M. Summers. Tienet : Text-image embedding network for common thorax disease classification and reporting in chest x-rays. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9049–9058, 2018.

Yaming Wang, Vlad I. Morariu, and Larry S. Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4148–4157, 2016b.

Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Interpret neural networks by extracting critical subnetworks. IEEE Transactions on Image Processing, 29 :6707–6720, 2020c.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. AAAI'14, page 1112–1119. AAAI Press, 2014.

Zhuhui Wang, Shijie Wang, Haojie Li, Zhi Dou, and Jianjun Li. Graph-propagation based correlation learning for weakly supervised fine-grained image classification. Proceedings of the AAAI Conference on Artificial Intelligence, 34(07) :12289–12296, Apr. 2020d. doi : 10.1609/aaai.v34i07.6912. URL https://ojs.aaai.org/index.php/AAAI/article/view/6912.

Jerry Wei, Arief Suriawinata, Bing Ren, Xiaoying Liu, Mikhail Lisovsky, Louis Vaickus, Charles Brown, Michael Baker, Mustafa Nasir-Moin, Naofumi Tomita, Lorenzo Torresani, Jason Wei, and Saeed Hassanpour. Learn like a pathologist : Curriculum learning by annotator agreement for histopathology image classification. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pages 2473–2483, January 2021.

Xiu-Shen Wei, Jianxin Wu, and Quan Cui. Deep learning for fine-grained image analysis : A survey. arXiv preprint arXiv :1907.03069, 2019.

Xiu-Shen Wei, Yi-Zhe Song, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, and Serge Belongie. Fine-grained image analysis with deep learning : A survey. IEEE transactions on pattern analysis and machine intelligence, 44(12) :8927–8948, 2022. ISSN 0162-8828.

Hao Wu, Jun Sun, and Qi You. Semi-supervised learning for medical image classification based on anti-curriculum learning. Mathematics, 11(6), 2023. ISSN 2227-7390. doi : 10.3390/math11061306. URL https://www.mdpi.com/2227-7390/11/6/1306.

Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 842–850, 2014.

Yuxiang Xie, Quanzhi Gong, Xidao Luan, Jie Yan, and Jiahui Zhang. A survey of fine-grained visual categorization based on deep learning. Journal of Systems Engineering and Electronics, 35(6) :1337–1356, 2023.

Huapeng Xu, Guilin Qi, Jingjing Li, Meng Wang, Kang Xu, and Huan Gao. Fine-grained image classification by visual-semantic embedding. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 1043–1049. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi : 10.24963/ijcai.2018/145. URL https://doi.org/10.24963/ijcai.2018/145.

Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification. In Computer Vision – ECCV 2018 : 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XIV, page 438–454, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-01263-2. doi : 10.1007/978-3-030-01264-9_26. URL https://doi.org/10.1007/978-3-030-01264-9_26.

Zijiang Yang, Reda Al-Bahrani, Andrew C. E. Reid, Stefanos Papanikolaou, Surya R. Kalidindi, Wei-keng Liao, Alok Choudhary, and Ankit Agrawal. Deep learning based domain knowledge integration for small datasets : Illustrative applications in materials informatics. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2019. doi : 10.1109/IJCNN.2019.8852162.

Bangjie Yin, Luan Tran, Haoxiang Li, Xiaohui Shen, and Xiaoming Liu. Towards interpretable face recognition. In In Proceeding of International Conference on Computer Vision, Seoul, South Korea, October 2019.

Ming Yin, Junbin Gao, and Zhouchen Lin. Laplacian regularized low-rank representation and its applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(3) :504–517, 2016. doi : 10.1109/TPAMI.2015.2462360.

Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In Computer Vision – ECCV 2018 : 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI, page 595–610, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-01269-4. doi : 10.1007/978-3-030-01270-0_35. URL https://doi.org/10.1007/978-3-030-01270-0_35.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. arXiv preprint arXiv :2205.15480, 2022.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks, 2020. URL https://arxiv.org/abs/1911.06455.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, et al. Concept embedding models. In NeurIPS 2022-36th Conference on Neural Information Processing Systems, 2022.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Metagraph2vec : Complex semantic path augmented heterogeneous network embedding. ArXiv, abs/1803.02533, 2018.

Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 834–849, Cham, 2014. Springer International Publishing.

Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning : a survey. Frontiers of Information Technology & Electronic Engineering, 19(1) :27–39, 2018.

Quanshi Zhang, Ruiming Cao, Ying Nian Wu, and Song-Chun Zhu. Growing interpretable part graphs on convnets via multi-shot learning. In AAAI Conference on Artificial Intelligence, 2016a.

Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting cnns via decision trees. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 6261–6270, 2019.

Quanshi Zhang, Xin Wang, Ying Nian Wu, Huilin Zhou, and Song-Chun Zhu. Interpretable CNNs for Object Classification . IEEE Transactions on Pattern Analysis & Machine Intelligence, 43(10) :3416–3431, October 2021a. ISSN 1939-3539. doi : 10.1109/TPAMI.2020.2982882. URL https://doi.ieeecomputersociety.org/10.1109/TPAMI.2020.2982882.

Shiwen Zhang, Sheng Guo, Limin Wang, Weilin Huang, and Matthew Scott. Knowledge integration networks for action recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 12862–12869, 2020.

Yu Zhang, Xiu-Shen Wei, Jianxin Wu, Jianfei Cai, Jiangbo Lu, Viet-Anh Nguyen, and Minh N. Do. Weakly supervised fine-grained categorization with part-based image representation. Trans. Img. Proc., 25(4) :1713–1725, April 2016b. ISSN 1057-7149. doi : 10.1109/TIP.2016.2531289. URL https://doi.org/10.1109/TIP.2016.2531289.

Zhenghang Zhang, Jinlu Jia, Yalin Wan, Yang Zhou, Yuting Kong, Yurong Qian, and Jun Long. Transr* : Representation learning model by flexible translation and relation matrix projection. J. Intell. Fuzzy Syst., 40(5) :10251–10259, January 2021b. ISSN 1064-1246. doi : 10.3233/JIFS-202177. URL https://doi.org/10.3233/JIFS-202177.

Zizhao Zhang, Pingjun Chen, Manish Sapkota, and L. Yang. Tandemnet : Distilling knowledge from medical images using diagnostic reports as optional semantic references. ArXiv, abs/1708.03070, 2017.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. Representation learning for measuring entity relatedness with rich information. In Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, page 1412–1418. AAAI Press, 2015. ISBN 9781577357384.

Heliang Zheng, Jianlong Fu, Zhengjun Zha, and Jiebo Luo. Looking for the devil in the details : Learning trilinear attention sampling network for fine-grained image recognition. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5007–5016, 2019.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2921–2929, 2015.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places : A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence, 40(6) :1452–1464, 2017.

Feng Zhou and Yuanqing Lin. Fine-grained image classification by exploring bipartite-graph labels. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1124–1133, 2015.

Peiqin Zhuang, Yali Wang, and Yu Qiao. Learning attentive pairwise interaction for fine-grained classification. ArXiv, abs/2002.10191, 2020.

Franck Anaël MBIAYA KWUITE

Intégration de connaissances dans des architectures profondes pour la classification d'images

Résumé : Les architectures d'apprentissage profond ont profondément transformé l'intelligence artificielle, en particulier dans le domaine de la vision par ordinateur. Elles permettent d'extraire des représentations complexes à partir d'images, avec des performances souvent remarquables. Néanmoins, ces modèles soulèvent encore plusieurs défis : leur manque d'explicabilité, leur dépendance à de grandes quantités de données annotées, ainsi que leur difficulté à intégrer des connaissances explicites. Dans ce contexte, cette thèse explore l'hybridation entre réseaux de neurones profonds et connaissances a priori, afin d'améliorer la capacité de généralisation des modèles de classification d'images. Elle s'articule autour de trois contributions principales. La première contribution consiste à proposer une méthodologie pour, à partir d'informations complémentaires sur les images, étiqueter des jeux de données et les enrichir avec des connaissances sous forme de règles. Ces ressources visent à faciliter l'évaluation de modèles exploitant des connaissances explicites, un besoin encore peu couvert dans la communauté. La deuxième contribution introduit une méthode d'apprentissage qui intègre des graphes de connaissances dans une architecture profonde. Elle repose sur une fonction de perte hybride combinant entropie croisée et distance dans un espace latent structuré par les connaissances. Ce cadre permet au modèle de se concentrer sur les exemples difficiles à classer en tirant parti de la structure sémantique du graphe. Les expériences menées montrent une amélioration des performances sur plusieurs jeux de données. Enfin, la troisième contribution propose un cadre de classification basé sur des concepts. Celui-ci s'appuie sur un graphe de connaissances pour d'une part modéliser les relations entre concepts et classes, et d'autre part propager des informations à travers les nœuds de ce graphe pour raisonner sur les concepts. Ce modèle vise à concilier capacité d'explication et performance, en combinant raisonnement symbolique et apprentissage profond.

Mots clés : Apprentissage profond, Vision par ordinateur, Intégration de connaissances, Classification d'images, Explicabilité

Knowledge Integration in Deep Learning for Image Classification

Abstract : Deep learning architectures have greatly transformed the field of artificial intelligence, particularly in computer vision. These models are capable of extracting rich and complex representations from raw images, often achieving impressive performance. However, several challenges persist, including explainability, strong dependence on large amounts of annotated data, and difficulty in integrating explicit prior knowledge. In this context, this thesis explores the hybridization of deep neural networks with prior knowledge, with the aim of improving the generalization ability of image classification models. It is structured around three main contributions. The first contribution presents a generic methodology for annotating image datasets by using additional semantic information. Based on complementary data, we introduce a labeling procedure and generate logical rules to encode domain knowledge. These enriched datasets aim to support the evaluation of models that incorporate explicit knowledge. The second contribution introduces a framework that integrates knowledge graphs into deep architectures. This approach is built based on a hybrid loss function that combines cross-entropy with a distance measure in a latent space structured by the knowledge graph. This enables the model to focus learning on the most challenging examples, while leveraging the semantic relationships encoded in the graph. Experimental results show improved performance across multiple datasets, demonstrating the effectiveness of the method. The third contribution presents a concept-based classification framework. This model uses a knowledge graph both to represent relationships between concepts and classes, and to propagate information through the graph nodes by reasoning. By combining symbolic reasoning with deep learning, this framework achieves a balance between explainability and accuracy.

Keywords : Deep Learning, Computer vision, Knowledge Integration, Image classification, Explainability

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

- Laboratoire d'Informatique Fondamentale d'Orléans (LIFO) -
6 Rue Léonard de Vinci, 45067 Orléans, France

PRISME Laboratoire Pluridisciplinaire de Recherche Ingénierie des Systèmes, Mécanique, Énergétique