Grégory Bourguin, Arnaud Lewandowski, Mourad Bouneffa

Hybridation de modèles d'IA avec des classifieurs ontologiquement explicables

Volume 6, nº 1-2 (2025), p. 35-57.

https://doi.org/10.5802/roia.92

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Volume 6, nº 1-2, 2025, 35-57

Hybridation de modèles d'IA avec des classifieurs ontologiquement explicables

Grégory Bourguin[a], Arnaud Lewandowski[a], Mourad Bouneffa[a]

[a] LISIC 50 rue Ferdinand Buisson Calais, 62228 (France)
E-mail : gregory.bourguin@univ-littoral.fr, arnaud.lewandowski@univ-littoral.fr, mourad.bouneffa@univ-littoral.fr
URL *:* https://lisic-prod.univ-littoral.fr/.

Résumé. — La recherche en Intelligence Artificielle Explicable (XAI) a souligné la nécessité de créer des modèles basés sur les connaissances du domaine pour qu'ils soient explicables du point de vue de leurs utilisateurs. Une part importante des travaux actuels se concentre sur la conception de modèles d'IA qui intègrent la sémantique et le mode de raisonnement des experts du domaine. S'inspirant des Modèles Basés sur les Concepts et des approches Neuro-Symboliques, nous proposons une architecture hybride pour construire des pipelines d'IA qui utilisent l'Apprentissage Automatique pour extraire les concepts du domaine et un raisonnement symbolique pour prédire une classification explicable. Le cœur de cette proposition est l'OntoClassifier, un module qui utilise des ontologies de domaine pour générer automatiquement des classifieurs ontologiquement explicables. Nous décrivons l'approche et l'architecture proposées en détaillant l'implémentation et les capacités de l'OntoClassifier. La solution est appliquée en Vision par Ordinateur et est illustrée à l'aide du Pizzaïolo Dataset.

Mots-clés. — IA explicable, modèle hybride, système à base de connaissance, ontologie, vision par ordinateur.

1. Introduction

Ces dernières années ont été marquées par une large démocratisation de solutions basées sur l'Apprentissage Automatique (ML, Machine Learning), en particulier l'Apprentissage Profond (DL, Deep Learning). Si la prolifération de ces nouveaux outils destinés à supporter des utilisateurs dans des tâches très diverses a démontré leur grande utilité, elle s'est aussi accompagnée de questionnements concernant la confiance que l'on peut leur accorder. De nombreux travaux de recherche ont ainsi souligné le problème de l'opacité des modèles issus du ML, et le besoin prégnant envers de nouvelles solutions permettant d'ouvrir ces boites noires. Cette problématique est au cœur du mouvement XAI [3] (eXplainable AI). Les exigences sont multiples : les développeurs ont besoin d'explications pour le débogage, les experts du domaine doivent pouvoir comprendre le fonctionnement de leurs outils pour les accepter, et la société dans son ensemble a besoin de preuves de fiabilité et d'équité. Face à ces

exigences, l'un des principaux défis est de fournir des explications appropriées à des utilisateurs spécifiques pour des tâches spécifiques.

Cet article se concentre sur l'XAI en Vision par Ordinateur (CV, Computer Vision). Au cours des dernières années, de nombreux travaux de recherche ont proposé d'expliquer des modèles d'IA sans définir clairement à qui ces explications sont destinées. Cependant, des recherches récentes [2] ont montré que si certaines explications comme les cartes de saillance peuvent aider les spécialistes de l'IA, ces informations ne devraient pas servir d'explication pour tous les utilisateurs. En raison de leur culture, de leurs connaissances et de leurs besoins différents, nous ne pouvons pas offrir les mêmes explications aux spécialistes de l'IA et à des utilisateurs qui ne sont pas des spécialistes de l'IA, mais des experts dans leur domaine d'application. Les travaux présentés dans cet article s'intéressent à la création d'XAI pour les experts du domaine.

Pour ce faire, il est nécessaire que la solution proposée soit capable de fournir des explications quant à ses décisions, mais aussi que ces explications soient compréhensibles par l'utilisateur, c'est-à-dire qu'elles soient en adéquation avec ses connaissances. Pour que l'explication soit efficace, il est crucial que l'utilisateur puisse comprendre : 1) la sémantique des entités qui sont manipulées, et aussi 2) ce qui est fait avec ces entités pour faire une prédiction, c'est-à-dire le raisonnement sous-jacent. Le défi pour les concepteurs est alors de créer des modèles explicables capables de combler l'écart sémantique entre les entités manipulées par les algorithmes, et celles permettant d'expliquer leurs décisions. En réponse, une partie significative des propositions actuelles en XAI vise à intégrer les connaissances des utilisateurs dans les explications fournies, voire dans les modèles d'IA eux-mêmes, et ce dès leur conception.

Parmi ces propositions, les Modèles Basés sur des Concepts (CBM, Concept Based Models) [21, 32] et les approches Neuro-Symboliques [4, 22] tentent de réifier les concepts et le raisonnement des utilisateurs pour produire des modèles d'IA intrinsèquement plus explicables. En suivant ces travaux, nos propositions s'inscrivent dans la tendance qui vise à tirer parti des ontologies de domaine [7, 5]. En effet, une ontologie permet de réifier la sémantique des utilisateurs ainsi que leur manière de raisonner sous la forme de symboles et de règles logiques exprimées à travers une logique de description. Il est alors possible de marier les modèles issus du ML avec des raisonneurs ontologiques pour créer des pipelines d'IA dont l'inférence est basée sur les connaissances du domaine [11]. Cette approche est particulièrement intéressante du point de vue de l'explicabilité du fait que l'inférence ontologique est un processus déductif qui peut être expliqué. Toutefois, la littérature souligne que cette inférence coûte cher [13], et ce type de processus n'est généralement mis en œuvre que lorsque la tâche de classification est trop complexe pour les algorithmes de ML (incluant le DL).

La solution que nous avons proposée dans [7] a pour but d'explorer cette approche CBM, Neuro-Symbolique et ontologique, et de pallier ses problèmes. Le papier présent offre nos plus récentes avancées concernant la méthode, l'architecture, et les outils que nous proposons pour créer des IA capables de fournir des explications fondées sur une ontologie, ainsi que nos dernières expérimentations. Nous ne focalisons pas ici sur les moyens permettant d'améliorer la classification, mais sur les apports d'une ontologie

pour l'explicabilité. Nous explorons la faisabilité de l'approche en utilisant des outils classiques du DL, et proposons une solution qui permet de constituer des pipelines d'IA intégrant des modèles Pytorch directement générés à partir d'expressions de classes décrites en OWL2 (Web Ontology Language). Ces modèles sont instances d'un module nommé OntoClassifier. Se basant sur plusieurs hypothèses en adéquation avec les tâches traitées, le raisonnement ontlogique fourni s'exécute beaucoup plus rapidement qu'avec des raisonneurs classiques comme Hermit ou Pellet. Tout en partageant le même framework, ces classifieurs ontologiquement explicables peuvent être combinés de manière transparente avec des extracteurs de concepts basés sur PyTorch.

La partie 2 de cet article détaille notre analyse de l'état de l'art de l'XAI qui nous amène à définir l'approche des Classifieurs Ontologiquement Explicables. La partie 3 décrit l'architecture proposée : nous illustrons son utilisation en CV sur une tâche de classification multi-étiquettes tout en fournissant le Pizzaïolo Dataset [6] que nous avons créé à cet effet. La partie 4 se concentre sur l'implémentation du module de classification ontologique OntoClassifier. La partie 6 analyse l'approche et les résultats actuels permettant d'envisager les futurs développements.

2. Explicabilité

Les systèmes basés sur le ML sont de plus en plus performants, mais deviennent en corollaire de plus en plus complexes et opaques. Ils apparaissent comme des boites noires [15], rendant très problématique l'intervention humaine pour le contrôle de leur exécution, leur déploiement, la compréhension de leurs décisions et leur évolution [3]. En conséquence, le besoin de transparence et surtout d'explicabilité des IA s'est révélé crucial avec des aspects liés à la confiance qu'un utilisateur peut leur accorder en matière de sureté de fonctionnement de systèmes critiques pilotés par l'IA, mais également en matière d'éthique et du respect de règles légales et sociétales telles que la non-ségrégation et le respect de la vie privée [18]. Dans ce papier, nous nous intéressons à l'explicabilité de l'IA en tant que justifications des décisions compréhensibles par les experts du domaine. Nous envisageons de plus les explications dites locales, c.a.d. celles qui veulent expliquer les décisions d'un modèle sur une prédiction particulière.

La nécessité de fournir des explications est un besoin ancien étant apparu dès les premières implémentations de systèmes experts [30]. Les systèmes à base de règles, ainsi que les algorithmes de ML réputés plus interprétables, comme la régression linéaire et les arbres de décision, ont également besoin d'outils simplifiant, résumant, et expliquant leurs prédictions. Cela peut se traduire par une restitution de la trace d'exécution des règles appliquées pour aboutir à une décision, ou encore par la simplification d'un arbre de décision en remplaçant des nœuds et arcs de niveaux de granularité fine par des concepts plus généraux issus de la terminologie du domaine de l'utilisateur [3]. Cependant une grande partie des systèmes actuels tels que ceux utilisés en CV nécessitent des modèles complexes issus du DL qui génèrent des modèles non interprétables pour lesquels l'application de ces techniques s'avère très difficile, voire impossible.

De ce fait, fournir des explications adaptées aux utilisateurs s'avère être un véritable défi. Comme l'a montré [17], la sémantique des configurations de paramètres

émergeant de l'entraînement de ces modèles ne correspond généralement pas à la compréhension humaine. De plus, même si des travaux récents réalisés dans un environnement contrôlé ont montré qu'il semble possible de relier certaines parties d'un modèle issu du DL à des concepts humains [12], les mêmes auteurs soulignent qu'il n'y a aucune garantie sur le fait que ces concepts soient utilisés par le modèle selon un raisonnement similaire à celui des humains pour effectuer des prédictions. Dans le contexte XAI, le problème est donc double : les entités manipulées à l'intérieur des modèles issus du DL ne correspondent très probablement pas à la sémantique des utilisateurs, et les calculs résultants sont si complexes qu'ils empêchent même les spécialistes de l'IA de comprendre exactement ce qu'ils font [15].

2.1. Les Méthodes Locales Post-Hoc

2.1.1. *Point de Vue Général*

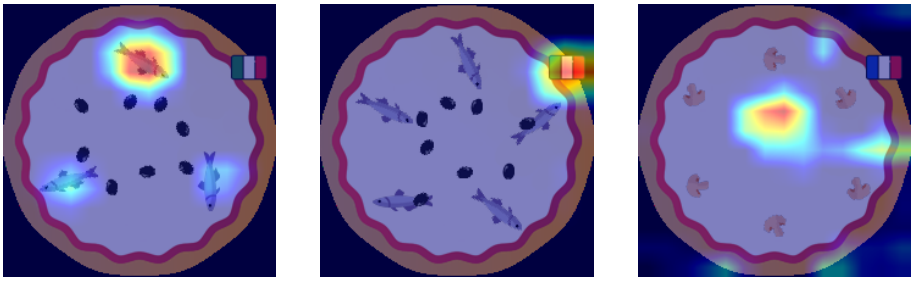
Les outils pour l'explicabilité locale des modèles issus du DL mettent majoritairement en œuvre des approches dites *post-hoc* permettant de fournir des explications sur des modèles préexistants. La plupart de ces méthodes sont aussi dites *agnostiques* du fait qu'elles sont applicables à tous types de modèles. Ces techniques utilisent en majorité les facteurs d'importance des caractéristiques d'entrée et reposent sur l'idée d'associer à chacune une valeur traduisant l'importance de son rôle dans la prédiction.

En CV, les caractéristiques correspondent aux pixels d'une image fournie en entrée du modèle. Les propositions consistent à mettre en correspondance les prédictions et les pixels qui ont conduit à une classification. Pour ce faire, diverses approches ont été adoptées avec en particulier les travaux consistant à explorer l'architecture des réseaux pour déterminer comment les couches intermédiaires perçoivent le monde extérieur [33]. Une des méthodes les plus représentatives de l'XAI en CV est la méthode Grad-CAM (Gradient-Weighted Class Activation Mapping) [27] (et ses dérivées) qui utilise le gradient d'une classe cible pour produire des cartes de saillance identifiant les régions de l'image qui ont le plus participé à sa classification. Une autre méthode très utilisée est LIME (Local Interpretable Model-Agnostic Explanations) [25] qui utilise des modèles de substitution interprétables sur des sous-parties du système global. Ces techniques et outils se sont avérés très utiles dans de nombreux travaux pour fournir des explications sur le fonctionnement de modèles d'IA. Cependant, comme le souligne [21] et comme nous allons le voir ci-après, ces approches ne garantissent généralement aucunement que les explications fournies soient compréhensibles par les utilisateurs.

2.1.2. *Problèmes des Approches Post-Hoc*

Pour illustrer les problèmes liés à ces approches post-hoc nous avons réalisé une expérience utilisant le Pizzaïolo Dataset [6]. Ce jeu de données, qui sera détaillé dans la partie 3.1, fournit entre autres des images synthétiques de pizzas et des étiquettes correspondant à leur nom (classe). Ce nom est en corrélation avec la « recette » de la pizza qui détermine les types d'éléments qui la constituent, c'est-à-dire : la base

(le bord de la pizza), les garnitures (olives, anchois, etc.), et l'origine (le drapeau du pays). Dans notre expérience, la tâche considérée est un problème de classification multi-classes (les noms) d'images synthétiques de pizzas. Nous avons construit et entrainé un classifieur tout à fait « classique » basé sur une architecture VGG19 [28] pré-entrainé sur Imagenet. Les données étant simples, ce classifieur a pu être entrainé pour atteindre une précision de 100 % sur l'ensemble de test du jeu de données constitué de 20 % des échantillons. Nous avons ensuite utilisé les outils dérivés de Grad-CAM [27] qui expliquent le résultat d'une classification en générant une carte de saillance des pixels qui ont le plus participé à la prédiction. Les images du Pizzaïolo Dataset étant constituées de manière à ce que les seules différences entre les pizzas soient les éléments spécifiés dans leur recette, en particulier les garnitures, on peut espérer que les cartes de saillance focalisent sur les pixels qui y correspondent pour les différencier.



(a) pizza_1 *(Napoletana)* (b) pizza_2 *(Napoletana)* (c) pizza_3 *(Mushrooms)*

FIGURE 2.1 – Trois images de test du Pizzaïolo Dataset classifiées par une architecture VGG19 (exactitude de 100 %) et exliquées par Grad-CAM.

La figure 2.1 montre les résultats obtenus avec Grad-CAM pour la classification de trois images de test. Il est à noter que comme dans la plupart des techniques à base de cartes de saillances, Grad-CAM souligne les pixels qui expliquent la classification sans les associer à une sémantique de plus haut niveau d'abstraction (ex. anchois, olive, etc.). L'interprétation dans le contexte du domaine est donc laissée à l'utilisateur. Les images (a) et (b) ont à juste titre été classifées *Napoletana*, qui est par construction constituée uniquement d'une pâte fine, d'olives et d'anchois, et d'une origine italienne. Pour l'image (a), on peut interpréter que l'IA a classifié cette image du fait de la présence d'anchois. Pourtant, le jeu de données contient d'autre types de pizzas contenant les mêmes anchois (*Capricciosa, FourSeasons, Siciliana*) : leur seule présence ne peut donc expliquer la classification. On peut d'ailleurs noter que les olives semblent totalement ignorées par l'IA, alors que du point de vue de la recette des « experts en pizzas », elles sont indispensables à une *Napoletana*. L'image (b) a elle aussi à juste titre été classifiée en *Napoletana*. Cependant, l'explication fournie laisse penser que l'IA a cette fois utilisé le drapeau italien et la base pour classifier. Une nouvelle fois, cette combinaison d'éléments peut être trouvée sur d'autres types de pizzas (*Fiorentina, Siciliana, SloppyGiuzeppe, Venizia*). Bien entendu, cette définition « pâte

fine italienne » ne correspond pas à la recette spécifique et attendue d'une *Napoletana*. On peut de plus noter que cette explication pour l'image (b) n'est pas cohérente avec celle de l'image (a) alors que la classe prédite est la même. Enfin, l'image (c) a justement été classifiée en *Mushrooms* dont la recette indique une pizza française à pâte fine contenant des champignons. L'explication indique que cette pizza est une *Mushrooms* parce qu'elle est majoritairement vide, ce qui ne correspond bien entendu pas à la définition attendue.

Toutefois, l'étude du Pizzaïolo Dataset permet de comprendre une partie de ces phénomènes. Par exemple, l'analyse du jeu de données montre que les anchois apparaissent systématiquement avec des olives. Par contre, les olives apparaissent fréquemment avec d'autres garnitures. De fait, en tant qu'expert en IA, on peut comprendre pourquoi sur une *Napoletana* constituée uniquement d'anchois et d'olives, un discriminant majeur choisi par le modèle est la présence d'anchois. De même, pour le cas (c), l'étude des recettes révèle que la *Mushrooms* est celle qui possède le moins de garnitures, ce qui peut expliquer pourquoi le modèle a considéré le vide comme un discriminant. Enfin, le cas (b) est plus intriguant puisque d'autres recettes indiquent une pizza italienne à pâte fine. Il est probable que le modèle ait arbitrairement choisi le drapeau comme fort discriminant pour des arrangements similaires (répartition) de garnitures sur la *Napoletana*, et utilise les autres pixels avec moins d'intensité pour prédire sa classification. Ceci ne peut être compris que lorsqu'on sait que Grad-CAM met en exergue les pixels qui ont le plus participé, ce qui ne signifie pas que les autres pixels de l'image entière soient totalement ignorés par le modèle.

L'expérience que nous venons de décrire a été réalisée avec d'autres architectures (ResNet50, EfficientNet, InceptionV3) et d'autres outils d'explication (Grad-CAM++ [8], Smooth Grad-CAM++ [24], LIME [25]), avec des résultats similaires. Ces remarques n'ont aucunement pour but de discréditer ces outils. Comme nous venons de le montrer, ils peuvent se révéler très utiles pour expliquer comment fonctionne un classifieur. Cependant, ces explications ne peuvent généralement être interprétées que par des spécialistes en IA capables d'analyser le jeu de données, de comprendre l'architecture du modèle ainsi que les algorithmes d'optimisation. Il est aussi préférable d'être au fait de la méthode mise en œuvre par l'outil d'explication : en effet, chaque technique ne fournit en général pas les mêmes explications sur le même modèle et pour les mêmes exemples [29]. Pour toutes ces raisons, ces approches post-hoc ne nous paraissent pas les plus indiquées pour fournir des explications aux utilisateurs.

Pour pallier ces problèmes, les travaux de recherche en XAI actuels explorent différentes stratégies pour adapter les explications aux connaissances des utilisateurs. Dans la littérature, l'explication de modèle post-hoc présente une évolution qui dirige les techniques d'attention sur les caractéristiques de bas niveau vers des cartes de saillance voulant faire émerger des concepts du domaine [16]. Les connaissances des experts du domaine ont également été introduites pour générer de meilleures explications post-hoc avec des modèles de substitution [10]. Cependant, ces dernières années, ces approches post-hoc ont aussi été critiquées pour d'autres raisons. En particulier, [26] souligne la question de la fiabilité des modèles de substitution censés

imiter les résultats des modèles boîte noire. Cet article souligne également que ce type d'approche n'explique en réalité pas le modèle ciblé, mais le modèle de substitution.

De notre point de vue, bien que les approches post-hoc aient produit des résultats intéressants, nous ne pensons pas qu'elles puissent produire des explications qui correspondent exactement au raisonnement des utilisateurs. En effet, les modèles issus du ML sont entraînés dans un monde limité défini par un jeu de données qui ne représente pas l'ensemble des connaissances des utilisateurs sur leur domaine. Un autre point est que les modèles issus du ML résultent d'un mécanisme d'optimisation qui ne correspond généralement pas à la manière de penser des utilisateurs. En d'autres termes, il est difficile d'extraire *a posteriori* la sémantique et le raisonnement utilisateur désirés d'un modèle ML qui a capturé ses propres connaissances du domaine d'application en opérant dans un monde et avec un paradigme différents. C'est pourquoi nous pensons que les connaissances des utilisateurs doivent être explicitement introduites dès la phase de conception du système d'IA.

2.2. Les approches par Conception

Parmi les approches qui veulent introduire la sémantique, voire le raisonnement utilisateur dès la conception des modèles d'IA, nous sommes particulièrement influencés par les Modèles Basés sur les Concepts (CBM) [21, 32]. Le principe sous-jacent est de concevoir les modèles en les obligeant à manipuler les entités sémantiques ou « concepts » du domaine. Un extracteur de concepts est formé pour identifier ces entités dans les données brutes. Les concepts détectés sont alors utilisées dans un classifieur qui est dédié aux prédictions. Les auteurs ont montré que ces approches peuvent aider à fournir de meilleures explications en soulignant l'activation des concepts qui correspondent aux connaissances des utilisateurs. Cependant, lorsque le classifieur reste un modèle « classique » issu du ML (sans hybridation symbolique), il n'y a aucune garantie sur la manière dont il utilise les caractéristiques sémantiques extraites dans ses calculs, et donc sur le fait qu'ils soient en adéquation avec le raisonnement des utilisateurs [1].

Comme indiqué dans [14], nous pensons que l'IA réellement explicable du point de vue des utilisateurs ne peut être atteinte qu'en utilisant un raisonnement logique. En réponse, une tendance émergente est l'IA Neuro-Symbolique qui propose de concevoir des IA hybrides en associant les modèles ML avec du raisonnement symbolique. Le raisonnement symbolique met en œuvre des calculs exprimables sous forme de règles logiques qui correspondent mieux au raisonnement humain. Une grande partie de la recherche en IA Neuro-Symbolique tente de produire automatiquement ces règles à partir du ML. Ces travaux sont très intéressants mais il n'y a actuellement pas de garantie sur le fait que la logique produite corresponde à celle des utilisateurs [23]. Pour que la logique mise en œuvre corresponde à celle du domaine, d'autres travaux proposent d'intégrer directement les règles réifiées au sein des modèles. C'est en particulier le cas des travaux qui combinent le DL avec la logique probabiliste [22]. Leur potentiel est prometteur, mais des questions subsistent quant à la formation des concepts au sein des modèles [23], et quant à l'interprétabilité d'explications exprimées en logique probabiliste pour des utilisateurs non spécialistes de l'IA [31].

Les travaux que nous présentons dans cet article explorent une voie alternative en concevant des IA hybrides qui associent les modèles du DL à des raisonneurs ontologiques [7, 5]. Ces approches visent à tirer parti du développement des ontologies de domaine dont le rôle est de plus en plus pregnant en Ingénierie des Connaissances et dans le Web Sémantique. Une ontologie permet de réifier la sémantique du domaine en entités conceptuelles ontologiques, et elle peut également réifier le raisonnement des utilisateurs sous la forme de règles exprimées à travers la richesse d'une logique de description. S'inscrivant dans une approche CBM, Neuro-Symbolique et ontologique, l'extracteur de concepts a alors pour but d'identifier dans les données d'entrée les entités qui correspondent aux concepts de l'ontologie, et les caractéristiques extraites sont transmises à un raisonneur ontologique qui infère la classification ciblée selon les règles de l'ontologie. Il faut noter que les travaux qui ont exploré cette voie ont utilisé des raisonneurs classiques comme Hermit ou Pellet pour effectuer l'inférence ontologique [11, 5], et qu'ils ont tous souligné les problèmes liés à leur lenteur. De notre point de vue, bien que très puissants, ces outils s'intègrent mal avec les solutions actuellement utilisées pour construire les modèles du DL comme Pytorch ou Tensorflow. En conséquence, nos travaux visent à produire une solution de classification ontologique qui pallie ces problèmes en se combinant harmonieusement avec les modèles issus du DL et en se basant sur les mêmes frameworks. En d'autres termes, nous visons à hybrider les pipelines d'IA avec des classifieurs ontologiquement explicables.

3. Conception de Pipelines d'IA Ontologiquement Explicables

3.1. Le Pizzaïolo Dataset

Les études expérimentant la création d'IA en utilisant des approches CBM (Concept Based Models) et Neuro-Symboliques ont souligné l'intérêt de travailler avec des jeux de données synthétiques. Ces jeux de données proposent des représentations visuelles simples et mettent en œuvre des concepts explicites non ambigus ainsi que des raisonnements de complexité variable [4]. Des travaux récents ont de plus souligné l'intérêt de travailler avec un jeu de données construit à partir d'une ontologie [12, 1]. En effet, les ontologies permettent de réifier les connaissances du domaine et facilitent donc la génération d'un ensemble de données contrôlé qui représente explicitement les concepts et raisonnements souhaités. C'est pourquoi nous avons créé le Pizzaïolo Dataset[(1)] [6] constitué de 4 800 images synthétiques de pizzas annotées (224×224 pixels) générées à partir de l'ontologie Pizzaïolo dédiée. La Figure 3.1 montre quelques-unes de ses images et étiquettes d'échantillon.

L'ontologie Pizzaïolo s'inspire de l'ontologie des pizzas de l'Université de Manchester [19] et définit 28 sous-classes de *Pizza* (*Napoletana*, *SpicyPizza*, ...) en utilisant :

- 2 sous-classes de *PizzaBase* (*ThinAndCrispyBase* et *DeepPanBase*)
- 16 sous-classes de *PizzaTopping* (*OliveTopping*, *JalapenoPepperTopping*, ...) qui appartiennent à 6 catégories principales (*MeatTopping*, *SpicyTopping*, ...)
- 4 instances de *Country* (*America*, *England*, *France*, *Italy*)

(1) `https://www.kaggle.com/dsv/8132725`

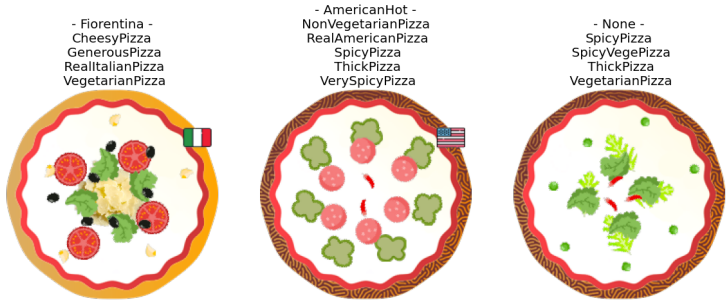


FIGURE 3.1 – Échantillons du Pizzaïolo Dataset. Les étiquettes correspondent aux expressions de classes ontologiques basées sur les concepts présents (base de pizza, garniture, drapeau) : la première étiquette indique le nom (recette) de la pizza, les suivantes en soulignent diverses caractéristiques. Les deux premiers échantillons sont des instances de « recettes » (*Fiorentina*, *AmericanHot*), tandis que le troisième résulte d'une génération de concepts aléatoire (étiquetée *None*).

L'ontologie décrit diverses recettes sous la forme d'expressions de classes comme :

Napoletana ≡ Pizza
⊓ (∃ hasTopping.AnchovyTopping) ⊓ (∃ hasTopping.OliveTopping)
⊓ (∀ hasTopping.(AnchovyTopping ⊔ OliveTopping))
⊓ (∀ hasCountryOfOrigin.Italy)

Elle fournit également des expressions de classes qui soulignent des caractéristiques générales telles que la *VegetarianPizza* :

VegetarianPizza ≡ Pizza
⊓ ¬ (∃ hasTopping.SeafoodTopping) ⊓ ¬ (∃ hasTopping.MeatTopping)

Avec ce jeu de données, notre objectif est cette fois une classification multi-étiquettes qui prend une image de pizza en entrée, qui prédit la ou les classe(s) correspondante(s), et qui est capable d'expliquer aux utilisateurs du domaine (des experts en pizzas) le raisonnement mis en œuvre par l'IA pour obtenir cette classification.

Inspirée par les approches CBM, Neuro-Symboliques et ontologiques, la figure 3.2 présente notre architecture hybride pour la classification ontologiquement explicable. Le pipeline d'IA utilise des modèles issus du ML comme extracteurs de concepts, et introduit un raisonneur ontologique nommé OntoClassifier. L'OntoClassifier est implémenté sous la forme d'un module Python qui est alimenté avec des ontologies de domaine pour générer automatiquement des modèles de classifieur Pytorch explicables. Les sections suivantes détaillent le pipeline proposé et sa mise en œuvre sur la tâche de classification multi-étiquettes du Pizzaïolo Dataset.

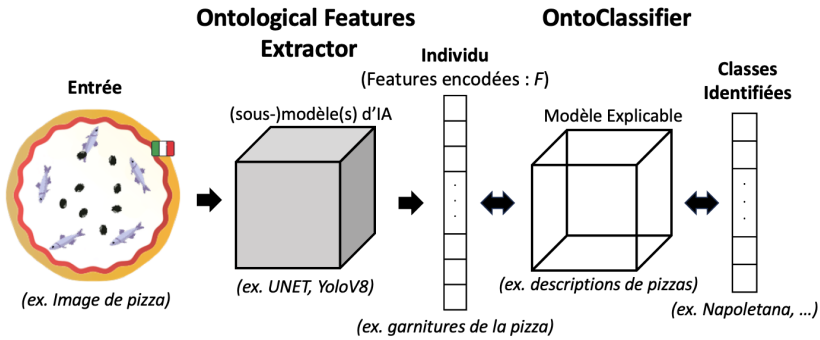


FIGURE 3.2 – Pipeline d'IA Hybride pour la Classification Explicable.

3.2. APPROCHE PROPOSÉE

3.2.1. *Extracteur de Propriétés Ontologiques*

Le rôle de l'extracteur de caractéristiques ontologiques (OFE, Ontological Features Extractor) (fig. 3.2 & 3.3) est d'identifier les instances de concepts dans les données d'entrée. Il génère des représentations d'individus (pizzas) en utilisant les propriétés ontologiques impliquées dans les expressions de classe ciblées. Notre tâche a besoin d'un sous-modèle pour caractériser la propriété *hasTopping*, c'est-à-dire un modèle qui identifie les pixels correspondant aux différentes garnitures. L'ensemble des classes de concepts ciblé est fourni par le co-domaine R_P de la propriété P. La propriété *hasTopping* a pour co-domaine *PizzaTopping*. Nous avons donc construit un modèle pour identifier les instances de *PizzaTopping*. Cette caractérisation peut être réalisée en

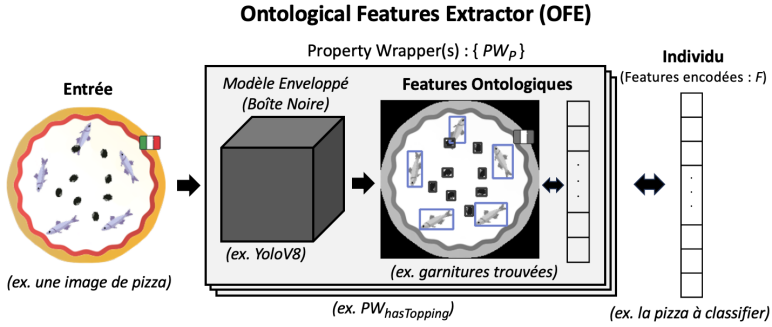


FIGURE 3.3 – L'extracteur de caractéristiques ontologiques OFE (Ontological Features Extractor). Ce modèle peut combiner différents Property Wrappers. Chaque Property Wrappers PW_P enveloppe un modèle d'IA sélectionné pour caractériser une propriété P. Le modèle enveloppé peut être partagé par différents PW_P. La sortie de l'OFE est un individu encodé F, *i.e.* la concaténation de toutes les caractéristiques F_P extraites par les différents PW_P impliqués.

combinant n'importe quels modèles d'IA capables d'extraire les caractéristiques souhaitées. Par exemple, la figure 3.3 montre que pour caractériser *hasTopping*, nous avons utilisé un détecteur d'objets YoloV8 entraîné pour détecter les garnitures de pizza. Ces objets (les boîtes englobantes et la classe de garniture associée) sont considérés comme des représentations d'entités *PizzaTopping*, et seront utilisés pour la classification.

La figure 3.3 montre également que nous introduisons le concept de Property Wrapper (*PW*). Un Property Wrapper PW_P enveloppe un modèle d'IA (par exemple, YoloV8) sélectionné pour caractériser une propriété P (par exemple, *hasTopping*) tout en maintenant les liens entre les données d'entrée (les pixels) et les caractéristiques extraites (par exemple, les entités *PizzaTopping*). Cette connexion bidirectionnelle sera utile pour fournir des explications mettant en évidence les éléments qui ont conduit à une classification (cf. 5.2). Le rôle d'un PW_P est également d'encoder la sortie de son modèle sous forme de caractéristiques ontologiques F_P. En effet, YoloV8, Mask R-CNN ou d'autres modèles d'IA ne fournissent pas le même format de sortie : c'est pourquoi chaque PW_P doit générer un encodage standardisé F_P des caractéristiques ontologiques extraites. Cet encodage sera détaillé dans la partie 4.1.

Un OFE peut utiliser conjointement plusieurs PW_P. Par exemple, notre expérience avec le Pizzaïolo Dataset utilise 2 PW_P : un pour *P=hasTopping*, et un autre pour *P=hasCountryOfOrigin*. Alors que nous utilisons YoloV8 pour détecter les entités *PizzaTopping*, nous pourrions utiliser un autre modèle (comme Mask R-CNN, ...) pour identifier la *Country* (le drapeau). Cependant, différents PW_P peuvent également envelopper un même modèle : dans notre exemple, nous avons finalement entraîné un unique YoloV8 pour détecter les entités *PizzaTopping* et *Country*, réutilisant ainsi le même modèle dans différents PW_P. Une dernière remarque concernant les PW_P est que ce choix d'architecture reflète notre volonté d'introduire une approche par composants dans laquelle divers modèles d'IA (préexistants) peuvent être combinés et participer à un raisonnement ontologique complexe.

Enfin, la figure 3.3 montre que la sortie d'un OFE est désignée comme un Individu. Cet individu est encodé sous la forme F, c'est-à-dire la concaténation de toutes les caractéristiques ontologiques F_P fournies (encodées) par tous les PW_P impliqués. Il constitue l'entrée du modèle OntoClassifier.

3.2.2. *Génération de Modèle OntoClassifier*

La tâche de l'OntoClassifier est la vérification d'appartenance des individus à des classes ontologiques afin de fournir une classification explicable. Le processus de génération utilise une ontologie pour créer automatiquement un ensemble de couches Pytorch directement capables de traiter les expressions de classe *OWL2*. Bien qu'implémentées en Pytorch, ces couches sont statiques et n'ont pas vocation à être entrainées. L'ontoclassifier ne vise pas à remplacer les raisonneurs ontologiques bien connus tels que Hermit ou Pellet dans d'autres contextes : il ne pourra par exemple pas vérifier la consistance de l'ontologie. L'OntoClassifier vise à fournir une classification ciblée et basée sur des expressions de classe en palliant la lenteur de ces raisonneurs, ainsi que leur manque d'intégration dans des pipelines utilisant des modèles basés

sur les frameworks du DL. Nous voulons en particulier qu'il soit capable de traiter rapidement des lots d'individus : en effet, l'une de nos applications visées (non détaillée ici) est de classer des milliers d'oiseaux marins dans des images provenant de campagnes d'études environnementales pour aider les ornithologues sur un jeu de données en constitution dans le projet TRIDA (financé par l'Office Français pour la Biodiversité).

La solution que nous avons choisie est la génération automatique de modèles effectuant des calculs matriciels qui tirent parti de la puissance de Pytorch. On peut noter que si notre implémentation actuelle repose sur Pytorch, elle peut être transposée en Tensorflow. L'idée principale est de générer des modèles de classification ontologique aisément intégrables dans des pipelines d'IA basés sur la même technologie. Pour cela, nous formulons quelques hypothèses sur les ontologies fournies en OWL 2 DL :

- Toutes les propriétés P impliquées dans les expressions de classes ciblées peuvent être caractérisées par un Property Wrapper PW_P.
- Les faits représentés dans les caractéristiques ontologiques encodées F sont considérés comme fermés (raisonnement local en monde clos).
- Par défaut, les classes appartenant au co-domaine R_P d'une propriété P sont considérées comme disjointes.
- Les individus sont classifiés séparément, *i.e.* dans des A Box différentes.

Ainsi, les expressions de classe gérées par les modèles OntoClassifier peuvent mélanger les restrictions de propriétés, comme par exemple dans :

Napoletana ≡ Pizza
⊓ (∃ hasTopping.AnchovyTopping) ⊓ (∃ hasTopping.OliveTopping)
⊓ (∀ hasTopping.(AnchovyTopping ⊔ OliveTopping))
⊓ (∀ hasCountryOfOrigin.Italy)

Elles peuvent utiliser des superclasses :

AnchovyTopping ⊑ SeafoodTopping
BaconTopping ⊑ MeatTopping
VegetarianPizza ≡ Pizza
⊓ ¬ (∃ hasTopping.SeafoodTopping)
⊓ ¬ (∃ hasTopping.MeatTopping)

Elles peuvent utiliser d'autres expressions de classes :

SpicyTopping ≡ PizzaTopping ⊓ (∃ hasSpiciness.(Hot ⊔ Mild))
SpicyVegePizza ≡ VegetarianPizza ⊓ hasTopping.SpicyTopping

Elles peuvent utiliser des restrictions de cardinalité qualifiées :

GenerousPizza ≡ Pizza ⊓ (⩾ 10 hasTopping.OliveTopping)

En pratique, le module OntoClassifier est paramétré par une ontologie OWL2 (par exemple, l'ontologie Pizzaïolo) et par la liste des classes ciblées par le futur classifieur (par exemple, *Napoletana*, *VegetarianPizza*, *SpicyVegePizza*, etc.). Le module génère alors automatiquement le modèle OntoClassifier correspondant.

4. Implémentation de l'OntoClassifier

L'implémentation actuelle de l'OntoClassifier[(2)] est réalisée en Python et est basée sur Pytorch. En tant que tel, un modèle OntoClassifier est capable de traiter des lots d'individus avec des calculs rapides et parallèles effectués sur GPU (cf. 5.1).

4.1. Les Entrées de l'OntoClassifier

Pour comprendre les couches qui constituent un modèle OntoClassifier généré, il est important de comprendre l'entrée que reçoit ce modèle à partir d'un OFE. Un OFE est composé d'un ou plusieurs Property Wrapper(s) PW_P où P est une propriété ontologique (par exemple, *hasTopping*, *hasCountryOfOrigin*, ...).

Soit $R_P = \{ r_i \mid i \in \mathbb{N} \}$, où R_P est le co-domaine de la propriété P.

Soit $E_P = \{ e_j \mid j \in \mathbb{N} \}$, où E_P est l'ensemble des entités détectées pour P.

PW_P génère F_P ($|R_P|$ x $|E_P|$) avec :

$$F_P = (f_{ij}) \text{ tel que } f_{ij} = \begin{cases} 1 & \text{si } e_j \text{ est instance de } r_i \\ 0 & \text{sinon} \end{cases}$$

Il ne faut pas oublier qu'en raison de la possibilité d'héritage multiple, les entités ontologiques peuvent être instance de plusieurs classes en même temps. De ce fait, une colonne f_{*j} peut contenir plusieurs 1. La figure 4.1 présente un exemple dans lequel $PW_{hasTopping}$ encode une image de pizza simple.

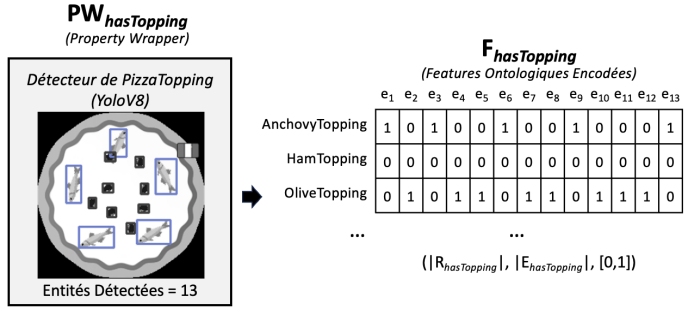


Figure 4.1 – $PW_{hasTopping}$ encodant une image de pizza. Dans cet exemple simplifié, $R_{hasTopping} = \{AnchovyTopping, HamTopping, OliveTopping, ...\}$. Ici, $PW_{hasTopping}$ utilise un modèle de détection d'objets YoloV8 pour générer la matrice binaire $F_{hasTopping}$ où les colonnes représentent $E_{hasTopping}$ (les entités détectées), et les lignes représentent $R_{hasTopping}$ (les classes de garnitures). Dans $F_{hasTopping}$, f_{ij} contient 1 si e_j est une instance de r_i, 0 sinon.

[(2)] https://github.com/SysReIC/ontoclassifier

Comme cela a été montré dans les figures 3.2 et 3.3, l'entrée du modèle OntoClassifier correspond à des individus encodés sous la forme F qui est la concaténation des différents F_P fournis par l'ensemble des PW_P impliqués.

4.2. Parsing des Expressions de Classes

La génération d'un modèle OntoClassifier est réalisée en parcourant récursivement les expressions de classe décrivant chaque classe ontologique ciblée. Ce parcours est implémenté avec Owlready2 [20]. Une expression de classe correspond à un arbre dans lequel les nœuds sont des opérateurs logiques, des classes nommées et/ou des restrictions de propriétés. L'algorithme génère un modèle qui combine un ensemble de couches Pytorch spécifiques. Pour simplifier la démonstration, nous nous concentrons ici sur la génération de couches correspondant aux opérateurs logiques et aux restrictions de propriétés. Chaque type d'opérateur logique et de restriction de propriété (objet et type de données, y compris les restrictions de cardinalité qualifiées), a une classe de couche Pytorch « ontologique » correspondante. Les noms de ces classes de couche informent sur le type de nœud correspondant : AndLayer, OrLayer, NotLayer, SomeLayer, OnlyLayer, ValueLayer, MinLayer, MaxLayer, ExactlyLayer, DataLTLayer, DataLELayer, DataGTLayer, DataGELayer et DataEQLayer. À titre d'exemple, nous pouvons nous concentrer sur le sous-modèle généré pour l'expression de classe ontologique *Napoletana* qui a été décrite en 3.2.2 :

Sous-modèle pour *Napoletana* :

AndLayer(*// Napoletana class expression ; opérateurs de 1er niveau :* $\sqcap$
  SomeLayer(*mask*(hasTopping.AnchovyTopping)) *// opérateur :* $\exists$
  SomeLayer(*mask*(hasTopping.OliveTopping)) *// opérateur :* $\exists$
  OnlyLayer(*// opérateur :* $\forall$
    mask(hasTopping.(AnchovyTopping $\sqcup$ OliveTopping))))
  OnlyLayer(*mask*(hasCountryOfOrigin.Italy)) *// opérateur :* $\forall$

Les classes de couche Pytorch calculant une restriction de propriété (SomeLayer, OnlyLayer, etc.) sont notées $L_{P.C}$, où L est un type de couche, P une propriété, et C une classe ontologique (ex. $SomeLayer_{hasTopping.OliveTopping}$). Chaque $L_{P.C}$ est paramétrée avec un masque $M_{P.C}$ pré-calculé lors du processus d'analyse avec la fonction *mask(P.C)*. Cette fonction crée une projection de C dans le co-domaine R_P. Comme C peut elle-même être une expression de classe, nous utilisons le raisonneur Hermit (uniquement pendant l'analyse) pour produire $M_{P.C}$ qui est défini comme suit :

$$M_{P.C} = (m_i) \text{ tel que } m_i = \begin{cases} 1 & \text{si } C \sqsubseteq r_i,\ r_i \in R_P \\ 0 & \text{sinon} \end{cases}$$

Dans le cas où C est une conjonction ($\sqcap$) entre un ensemble de classes $c_1, \ldots, c_n$, nous recherchons des entités appartenant à plusieurs classes (en même temps) :

$$M_{P.C} = M_{P.c1} \vee \cdots \vee M_{P.cn}$$

Dans le cas où C est une disjonction ($\sqcup$) entre un ensemble de classes $c_1, \ldots, c_n$, nous recherchons des entités appartenant à (au moins) l'une de ces classes. Puisque nous

voulons garder la trace de chaque c_k impliquée à des fins d'explication (révéler celles qui ont correspondu ou non), nous concaténons les masques $M_{P.ck}$ et définissons :

$$M_{P.C} = \begin{bmatrix} M_{P.c1} & \dots & M_{P.cn} \end{bmatrix}^T$$

Le résultat global du modèle OntoClassifier correspond à la classification multi-étiquettes. Il résulte de la concaténation de tous les sous-modèles générés lors de ce processus de parsing, chaque sous-modèle correspondant à une expression de classe ontologique ciblée. Nous verrons dans la partie 5.2 que chaque couche Pytorch du modèle fournit également des moyens pour générer des explications.

4.3. Raisonnement

Un modèle OntoClassifier prend en entrée un lot d'individus (cf. 4.1), et calcule leur classification multi-étiquettes respective. Nous avons précédemment montré que le calcul pour chaque expression de classe est effectué par un (sous-)modèle dédié et composé de nos couches « ontologiques ». Chaque couche est elle-même un modèle qui reçoit le lot complet en entrée, et calcule l'appartenance à sa (sous-)expression de classe en sortie. Nous voulons que nos couches utilisent la puissance de Pytorch, leur mode opératoire a donc été conçu comme un calcul matriciel. Pour illustrer l'approche, nous nous concentrons ici sur SomeLayer, OnlyLayer et MinLayer. Pour simplifier, les définitions suivantes sont présentées en omettant volontairement la dimension du lot.

Nous considérons un individu *ind* pour lequel les caractéristiques ontologiques extraites sont encodées comme F, c'est-à-dire la concaténation de toutes les caractéristiques encodées F_P extraites par chaque PW_P. Chaque couche ontologique $L_{P.C}$ effectue son calcul en utilisant son masque précalculé $M_{P.C}$ (cf. 4.2).

Soient les fonctions de base δ et θ :

$$\delta_{A\geqslant 0} = D = (d_{ij}) \text{ tel que } d_{ij} = \begin{cases} 1 & \text{si } a_{ij} \geqslant 0 \\ 0 & \text{sinon} \end{cases} \qquad \theta_{x\geqslant n} = \begin{cases} 1 & \text{si } x \geqslant n \\ 0 & \text{sinon} \end{cases}$$

La couche SomeLayer correspondant à $\exists\, P.C$ est alors définie par :

$$\text{SomeLayer}_{P.C}\,(ind) = \theta_{x\geqslant 1}\Big(\sum_j \sum_i \delta_{A\geqslant 0}\big(M_{P.C}\,(F_P - 1)\big)_{ij}\Big)$$

La couche OnlyLayer correspondant à $\forall\, P.C$ est définie par :

$$\text{OnlyLayer}_{P.C}\,(ind) = \theta_{x\geqslant 1}\Big(\prod_j \sum_i \delta_{A\geqslant 0}\big(M_{P.C}\,(F_P - 1)\big)_{ij}\Big)$$

La couche MinLayer correspondant à $\geqslant n\ P.C$ est définie par :

$$\text{MinLayer}_{P.C}\,(ind, n) = \theta_{x\geqslant n}\Big(\sum_j \sum_i \delta_{A\geqslant 0}\big(M_{P.C}\,(F_P - 1)\big)_{ij}\Big)$$

L'implémentation des autres $L_{P.C}$ fonctionne de manière similaire. Les couches calculant les propriétés de type de données et les opérateurs logiques effectuent des tests « simples » ou des opérations logiques entre les résultats fournis par d'autres couches.

5. Expérimentations

5.1. Classification

Nous avons évalué notre approche sur le Pizzaïolo Dataset, ainsi que sur un sous-ensemble aléatoire et équilibré du jeu de données XTrains [12]. XTrains contient des images synthétiques de trains pour une tâche de classification multi-étiquettes dont les classes sont définies dans une ontologie qui décrit différents types de trains (ex. *PassengerTrain*) à partir de différents types de wagons (ex. *PassengerCar*, *LongWagon*).

Pour le Pizzaïolo Dataset, l'OFE contient un unique modèle YoloV8 enveloppé dans $PW_{hasTopping}$ et $PW_{hasCountryOfOrigin}$. L'OntoClassifier a été généré directement à partir de l'ontologie pour la tâche de classification multi-étiquettes. Pour XTrains, l'ontologie n'implique qu'une seule propriété *has*. De fait, l'OFE pour XTrains utilise un unique YoloV8 (détectant les types de wagons) enveloppé dans PW_{has}, et nous avons généré un OntoClassifier directement à partir de l'ontologie fournie. Le tableau 5.1 montre que chacun des pipelines hybrides produits atteint d'excellents résultats.

Table 5.1 – Détails des Expérimentations de Pipelines d'IA Hybrides

	Pizzaïolo Dataset	XTrains Dataset
Classes ciblées en multi-étiquettes	28	11
Taille du jeu de données d'entrainement	2880	300 000
Taille du jeu de données de test	960	100 000
Exactitude (Accuracy)	0,994	0,975
Précision (Precision)	0,999	0,976
Rappel (Recall)	0,998	0,975
Score F1 (F1 Score)	0,999	0,975

Pour évaluer la vitesse de classification d'un OntoClassifier, nous avons utilisé le Pizzaïolo Dataset (du fait qu'il présente l'ontologie la plus complexe) et mesuré le temps nécessaire pour classifier différentes tailles de lots d'individus (en excluant les temps de traitement de l'OFE). Dans le Pizzaïolo Dataset, le nombre d'entités (principalement des garnitures) constituant un individu varie entre 5 et 30. La vitesse de classification pouvant varier en fonction de la complexité d'un individu, nous avons répété chaque expérience 100 fois avec des lots différents et pour chaque taille de lot considérée, puis nous avons conservé les temps de traitement moyens. Les mêmes expériences ont été répétées pour Hermit et Pellet avec Owlready2 [20] pour comparaison. On notera que nous avons interrompu les expériences pour lesquelles la durée de calcul d'un seul lot dépassait 300 secondes. Les résultats sont présentés dans le tableau 5.2. Il convient de rappeler que l'OntoClassifier ne permet pas le raisonnement ontologique « global » d'Hermit ou de Pellet. De plus, le traitement d'un lot d'individus dans Hermit et Pellet

signifie peupler l'ontologie, c'est-à-dire mettre les individus dans la même A-Box, alors qu'un OntoClassifier traite les individus séparément, c'est-à-dire dans des A-Box différentes. Ceci étant dit, le tableau 5.2 montre que la classification ontologique ciblée réalisée par l'OntoClassifier s'effectue beaucoup plus rapidement.

TABLE 5.2 – Temps Moyens pour Classifier un Lot d'Individus

	Hermit		Pellet		OC (CPU)		OC (GPU)	
Taille du Lot	b	i	b	i	b	i	b	i
1	11,00	11,001	0,74	0,743	0,023	**0,0232**	0,051	0,0507
5	62,58	12,517	5,22	1,043	0,029	**0,0059**	0,052	0,0103
10	151,99	15,199	32,94	3,294	0,034	**0,0034**	0,052	0,0052
80	-	-	-	-	0,0506	**0,00063**	0,0527	0,00066
90	-	-	-	-	0,0560	0,00062	0,0542	**0,00060**
1000	-	-	-	-	0,6769	0,00068	0,1090	**0,00011**

OC= « OntoClassifier » ; b = « secondes/lot » ; i = « secondes/individu »
(CPU : I9-10850K 3,6GHz / 32Go DDR4 3200MHz ; GPU : RTX3080)

5.2. EXPLICATIONS

Notre approche a été conçue pour fournir une classification avec des explications locales. Les exemples suivants ont été produits avec un module d'explication simple construit pour illustrer les capacités d'introspection de l'OntoClassifier.

La figure 5.1 montre une image classifiée avec 4 classes. Elle affiche les explications pour la classe *Napoletana*. Ces explications reflètent exactement ce qui se passe dans le modèle : chaque partie est fournie par le (sous-)modèle correspondant et qui a été pré-généré lors de l'analyse de l'ontologie (cf. 4.2). En effet, l'implémentation de chaque couche ontologique L a une méthode $explain(F)$ qui prend un individu encodé F en

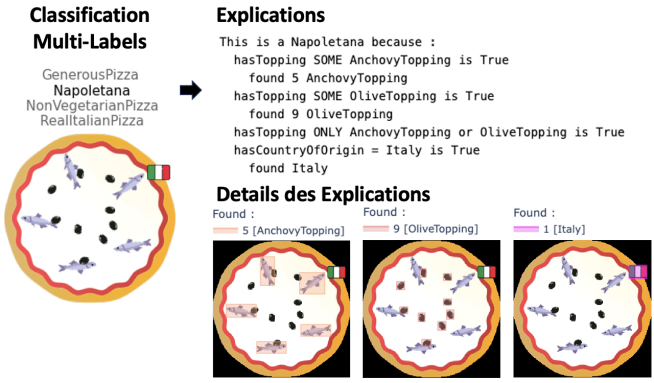


FIGURE 5.1 – Génération d'une explication pour une classe ciblée.

entrée, et retourne X : l'ensemble des caractéristiques spécifiques de l'individu qui expliquent pourquoi il est instance (ou non) de l'expression de classe correspondante.

L'OntoClassifier peut également expliquer pourquoi une image *n'a pas* été prédite comme étant une instance d'une classe particulière. Par exemple, dans la figure 5.2, nous demandons si une pizza est instance de *SpicyVegePizza*. Le module explique que ce n'est pas le cas et souligne (uniquement) les restrictions en conflit avec l'expression de classe correspondante, tout en mettant en évidence les entités problématiques. Ce mécanisme est utile pour construire des explications contrastives ou contrefactuelles. On pourra enfin noter que les explications utilisent la richesse d'OWL 2 DL et en particulier ici la notion d'héritage en indiquant que cette pizza n'est pas une *VegetarianPizza* car elle contient des *MeatTopping* qui sont en réalité 4 instances de *HamTopping*.

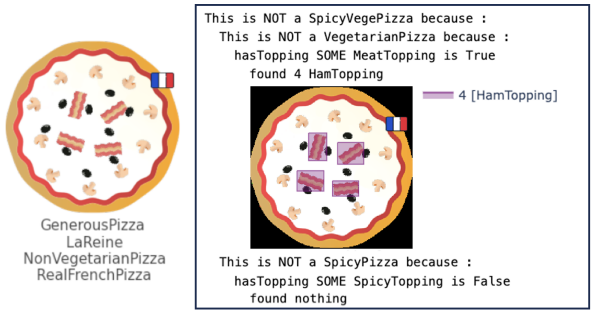


FIGURE 5.2 – Expliquer pourquoi un individu *n'est pas* instance d'une classe.

6. DISCUSSION

Comme nous l'avons montré, notre proposition peut aider à créer des IA efficaces et ontologiquement explicables. Cependant, il faut souligner que le niveau d'abstraction des explications est intrinsèquement lié au niveau d'abstraction des concepts extraits. Dans notre exemple, s'il est possible d'expliquer qu'une pizza est une *Napoletana* parce qu'elle contient des entités *AnchovyTopping* et *OliveTopping*, le classifieur ne peut pas expliquer pourquoi une région de l'image a été identifiée comme représentant un concept spécifique. Il serait possible de créer un pipeline plus complexe utilisant une ontologie plus détaillée, avec des expressions pour chaque sous-classe de *PizzaTopping*, et d'utiliser des modèles permettant d'extraire des caractéristiques ontologiques plus basses. Cependant, toute approche d'XAI pour les utilisateurs doit envisager qu'au-delà d'un certain niveau d'abstraction, il n'est plus possible de lui fournir des explications. Ainsi, nous pensons que notre solution est utile pour fournir des explications aux utilisateurs jusqu'à un certain niveau défini par l'ontologie, et nous réservons les explications plus profondes aux spécialistes de l'IA avec d'autres techniques de l'XAI.

Les expériences décrites dans ce papier utilisent des jeux de données synthétiques. La recherche en IA Neuro-Symbolique repose souvent de ce type de jeux de données car ils aident à développer des modèles travaillant sur des exemples complexes mais

également contrôlés. C'est pourquoi nous avons publié le Pizzaïolo Dataset : il offre de nouvelles opportunités pour des expériences avec des images multi-étiquettes représentant des classes ontologiques de complexité variable. Il convient de noter que notre approche s'applique aussi sur des tâches de classification d'images réelles. Nous n'avons pas pour l'instant pu trouver de jeu de données de référence composé d'images réelles annotées avec des concepts et des classes définis dans une ontologie. Néanmoins, aux fins de démonstration, nous avons réutilisé un jeu de données public d'images de cartes à jouer[3] pour entrainer un extracteur de concepts, créé une ontologie des Mains de Poker utilisant ces concepts, et généré l'OntoClassifier correspondant. La figure 6.1 montre comment le pipeline résultant permet de classifier les 9 catégories de mains de poker (*RoyalFlush*, etc.) dans un flux vidéo en affichant des explications générées en temps réel. Le code et la vidéo sont disponibles sur le github du projet OntoClassifier.

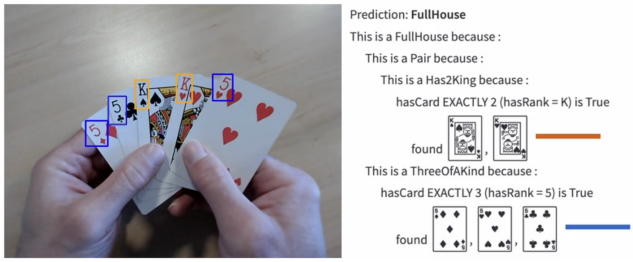


FIGURE 6.1 – Explications en temps réel dans des vidéos de mains de poker. (MacBook Pro 2021 / M1 Pro / 16 Go ; Images : 640×480 ; Vitesse : 22 fps).

Les approches CBM sont par essence sensibles à l'extraction de concepts, et une détection erronnée peut entraîner une mauvaise classification ou une explication incorrecte. L'approche que nous avons présentée y est d'autant plus sensible qu'elle utilise le raisonnement ontologique qui est déterministe et repose sur les concepts détectés. Nous avons montré que notre approche peut fournir de très bons résultats, y compris sur des images réelles, mais à condition d'être capable de créer un OFE performant. De fait, si un concept est manquant, ou si un concept non désiré a faussement été détecté, l'OntoClassifier ne peut pas fournir la classification espérée. Par contre, grâce à ses explications contrastives (cf. 5.2), il peut en expliquer les raisons, ce qui est utile pour identifier précisément les problèmes et améliorer le pipeline. Néanmoins, on ne peut ignorer que même s'ils peuvent être très puissants, les modèles issus du DL qui constituent un OFE ne sont pas parfaits, et qu'il est nécessaire d'intégrer une gestion de l'incertitude. Une piste naturelle déjà explorée dans la littérature est celle des logiques probabilistes [22]. Cependant, nous avons expliqué en 2.2 les raisons de notre préférence envers une approche ontologique, et avons donc pour l'instant choisi de ne pas suivre cette voie. Nos travaux actuels se tournent vers le calcul de similarité sémantique permettant, en cas d'incertitude, de proposer une classification candidate tout en étant capables, grâce aux outils de l'OntoClasifier, d'en expliquer les

[3] `https://github.com/geaxgx/playing-card-detection`

raisons à l'utilisateur concerné. Ces travaux nécéssitent non seulement d'explorer la notion de similarité sémantique au sein des ontologies, mais également de développer les Interactions Humain-Machine (IHM) qui permettront aux utilisateurs d'explorer la classification et les explications fournies. Les explications « brutes » présentées dans ce papier avaient pour but d'illustrer les capacités d'introspection de l'OntoClassifier. Il est évident que nous devons encore inventer de nouvelles IHM pour mieux les présenter aux utilisateurs. Nous sommes convaincus que la recherche en XAI doit aujourd'hui participer à l'effort important et complémentaire des travaux en IHM sur l'XUI [9].

7. Conclusion

En nous engageant dans la recherche en XAI pour créer des modèles d'IA explicables du point de vue de leurs utilisateurs, nous avons proposé une approche, une architecture et des outils pour hybrider les pipelines d'IA avec des classifieurs ontologiquement explicables. L'approche proposée est conçue pour intégrer la sémantique et le mode de raisonnement des utilisateurs riéfiés dans une ontologie. L'architecture proposée utilise des modèles issus du DL pour extraire les concepts du domaine, et un raisonnement symbolique effectué par l'OntoClassifier pour fournir une classification explicable. L'OntoClassifier a été créé pour pallier la lenteur et le manque d'intégration des raisonneurs ontologiques classiques dans les pipelines d'IA. Il est capable de parser les ontologies du domaine en OWL2 pour générer automatiquement des classifieurs ontologiquement explicables sous la forme de modèles Pytorch. Nous avons créé et utilisé le Pizzaïolo Dataset pour illustrer l'approche proposée tout en détaillant la mise en œuvre de l'OntoClassifier sur une tâche de classification multi-étiquettes en CV. Nous avons testé ses capacités en termes de performances et de vitesse de classification, et présenté ses mécanismes d'introspection qui permettent la génération d'explications. Ces expériences ont montré qu'il permet de créer des IA hybrides performantes, rapides, et ontologiquement explicables. Enfin, nous avons discuté des limites de notre approche, des pistes de recherche futures, ainsi que des travaux en cours pour améliorer cette solution en y intégrant la gestion des incertitudes dans un cadre ontologique, et en inventant les futures XUI qui permettront aux utilisateurs d'interagir avec leurs IA au travers d'explications.

Bibliographie

[1] A. Agafonov & A. Ponomarev, « An Experiment on Localization of Ontology Concepts in Deep Convolutional Neural Networks », in *The 11th International Symposium on Information and Communication Technology, SoICT 2022, Hanoi, Vietnam, December 1-3, 2022*, ACM, 2022, p. 82-87.

[2] A. R. Akula & S.-C. Zhu, « Attention Cannot Be an Explanation », `https://arxiv.org/abs/2201.11194`, 2022.

[3] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila & F. Herrera, « Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI », *Inf. Fusion* **58** (2020), p. 82-115.

[4] P. Barbiero, G. Ciravegna, F. Giannini, M. E. Zarlenga, L. C. Magister, A. Tonda, P. Lio, F. Precioso, M. Jamnik & G. Marra, « Interpretable Neural-Symbolic Concept Reasoning », in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii,*

USA (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato & J. Scarlett, éds.), Proceedings of Machine Learning Research, vol. 202, PMLR, 2023, p. 1801-1825.

[5] M. Bellucci, N. Delestre, N. Malandain & C. Zanni-Merk, « Combining an Explainable Model Based on Ontologies with an Explanation Interface to Classify Images », in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES-2022, Verona, Italy and Virtual Event, 7-9 September 2022* (M. Cristani, C. Toro, C. Zanni-Merk, R. J. Howlett & L. C. Jain, éds.), Procedia Computer Science, vol. 207, Elsevier, 2022, p. 2395-2403.

[6] G. Bourguin & A. Lewandowski, « Pizzaïolo Dataset: Des Images Synthétiques Ontologiquement Explicables », in *Explain'AI Workshop, at 24e Conférence Francophone Sur l'Extraction et La Gestion Des Connaissances, EGC 2024.*, 2024.

[7] G. Bourguin, A. Lewandowski, M. Bouneffa & A. Ahmad, « Vers Des Classifieurs Ontologiquement Explicables », in *IC 2021 : 32e Journées Francophones d'Ingénierie Des Connaissances (Proceedings of the 32nd French Knowledge Engineering Conference), Bordeaux, France, June 30 – July 2, 2021* (M. Lefrançois, éd.), 2021, p. 89-97.

[8] A. Chattopadhay, A. Sarkar, P. Howlader & V. N. Balasubramanian, « Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks », in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, p. 839-847.

[9] M. Chromik & A. Butz, « Human-XAI Interaction: A Review and Design Principles for Explanation User Interfaces », in *Human-Computer Interaction – INTERACT 2021 – 18th IFIP TC 13 International Conference, Bari, Italy, August 30 – September 3, 2021, Proceedings, Part II* (C. Ardito, R. Lanzilotti, A. Malizia, H. Petrie, A. Piccinno, G. Desolda & K. Inkpen, éds.), Lecture Notes in Computer Science, vol. 12933, Springer, 2021, p. 619-640.

[10] R. Confalonieri, T. Weyde, T. R. Besold & F. Moscoso del Prado Martín, « Trepan Reloaded: A Knowledge-driven Approach to Explaining Artificial Neural Networks », in *24th European Conference on Artificial Intelligence (ECAI 2020)*, vol. 325, IOS Press, 2020, p. 2457-2464.

[11] D. Conigliaro, R. Ferrario, C. Hudelot & D. Porello, « Integrating Computer Vision Algorithms and Ontologies for Spectator Crowd Behavior Analysis », in *Group and Crowd Behavior for Computer Vision, 1st Edition* (V. Murino, M. Cristani, S. Shah & S. Savarese, éds.), Academic Press, 2017, p. 297-319.

[12] M. de Sousa Ribeiro & J. Leite, « Aligning Artificial Neural Networks and Ontologies towards Explainable AI », in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, AAAI Press, 2021, p. 4932-4940.

[13] Z. Ding, L. Yao, B. Liu & J. Wu, « Review of the Application of Ontology in the Field of Image Object Recognition », in *Proceedings of the 11th International Conference on Computer Modeling and Simulation, ICCMS 2019, North Rockhampton, QLD, Australia, January 16-19, 2019*, ACM, 2019, p. 142-146.

[14] D. Doran, S. Schulz & T. R. Besold, « What Does Explainable AI Really Mean? A New Conceptualization of Perspectives », in *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 Co-Located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017* (T. R. Besold & O. Kutz, éds.), CEUR Workshop Proceedings, vol. 2071, CEUR-WS.org, 2017.

[15] F. K. Dosilovic, M. Brcic & N. Hlupic, « Explainable Artificial Intelligence: A Survey », in *41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018, Opatija, Croatia, May 21-25, 2018* (K. Skala, M. Koricic, T. G. Grbac, M. Cicin-Sain, V. Sruk, S. Ribaric, S. Gros, B. Vrdoljak, M. Mauher, E. Tijan, P. Pale & M. Janjic, éds.), IEEE, 2018, p. 210-215.

[16] T. Fel, A. Picard, L. Béthune, T. Boissin, D. Vigouroux, J. Colin, R. Cadène & T. Serre, « CRAFT: Concept Recursive Activation FacTorization for Explainability », in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, IEEE, 2023, p. 2711-2721.

[17] A. Gonzalez-Garcia, D. Modolo & V. Ferrari, « Do Semantic Parts Emerge in Convolutional Neural Networks? », *Int. J. Comput. Vis.* **126** (2018), no 5, p. 476-494.

[18] B. Goodman & S. R. Flaxman, « European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation" », *AI Mag.* **38** (2017), nº 3, p. 50-57.

[19] M. Horridge, « Protégé OWL Tutorial | OWL Research at the University of Manchester », `http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/`, 2011.

[20] J.-B. Lamy, *Ontologies with Python: Programming OWL 2.0 Ontologies with Python and Owlready2*, 1st ed. édition éd., Apress, 2020.

[21] M. M. Losch, M. Fritz & B. Schiele, « Interpretability Beyond Classification Output: Semantic Bottleneck Networks », `https://arxiv.org/abs/1907.10882`, 2019.

[22] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester & L. D. Raedt, « DeepProbLog: Neural Probabilistic Logic Programming », in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi & R. Garnett, éds.), 2018, p. 3753-3763.

[23] E. Marconato, S. Teso, A. Vergari & A. Passerini, « Not All Neuro-Symbolic Concepts Are Created Equal: Analysis and Mitigation of Reasoning Shortcuts », in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt & S. Levine, éds.), 2023.

[24] D. Omeiza, S. Speakman, C. Cintas & K. Weldemariam, « Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models », `https://arxiv.org/abs/1908.01224`, 2019.

[25] M. T. Ribeiro, S. Singh & C. 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, San Francisco, CA, USA, August 13-17, 2016* (B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen & R. Rastogi, éds.), ACM, 2016, p. 1135-1144.

[26] C. Rudin, « Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead », *Nat. Mach. Intell.* **1** (2019), nº 5, p. 206-215.

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh & D. Batra, « Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization », *Int. J. Comput. Vis.* **128** (2020), nº 2, p. 336-359.

[28] K. Simonyan & A. Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition », in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio & Y. LeCun, éds.), 2015.

[29] J. Stodt, M. Madan, C. Reich, L. Filipovic & T. Ilijas, « A Study on the Reliability of Visual XAI Methods for X-Ray Images », in *Healthcare Transformation with Informatics and Artificial Intelligence, ICIMTH 2023, 21st International Conference on Informatics, Management, and Technology in Healthcare, Athens, Greece, from 1-3 July 2023* (J. Mantas, P. Gallos, E. Zoulias, A. Hasman, M. S. Househ, M. Charalampidou & A. Magdalinou, éds.), Studies in Health Technology and Informatics, vol. 305, IOS Press, 2023, p. 32-35.

[30] W. R. Swartout, C. Paris & J. D. Moore, « Explanations in Knowledge Systems: Design for Explainable Expert Systems », *IEEE Expert* **6** (1991), nº 3, p. 58-64.

[31] G. Vidal, « Explaining Explanations in Probabilistic Logic Programming », `https://arxiv.org/abs/2401.17045`, 2024.

[32] M. E. Zarlenga, P. Barbiero, G. Ciravegna, G. Marra, F. Giannini, M. Diligenti, Z. Shams, F. Precioso, S. Melacci, A. Weller, P. Lió & M. Jamnik, « Concept Embedding Models: Beyond the Accuracy-Explainability Trade-Off », in *NeurIPS*, 2022.

[33] Q. Zhang, Y. N. Wu & S.-C. Zhu, « Interpretable Convolutional Neural Networks », in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, Computer Vision Foundation / IEEE Computer Society, 2018, p. 8827-8836.

ABSTRACT. — The research in eXplainable Artificial Intelligence (XAI) has emphasized the need to create models based on domain knowledge to make them explainable from their users' perspective. A significant portion of current work focuses on designing AI models that integrate the domain experts' semantics and way of reasoning. Drawing inspiration from Concept Based Models and Neuro-Symbolic AI, we propose a hybrid architecture for constructing AI pipelines that utilize Machine Learning to extract domain concepts and symbolic reasoning to predict an explainable classification. The core of this proposal is the OntoClassifier, a module that uses domain ontologies to automatically generate ontologically explainable classifiers. We describe the proposed approach and architecture, detailing the implementation and capabilities of the OntoClassifier. The solution is applied in Computer Vision and is illustrated using the Pizzaïolo Dataset.

KEYWORDS. — XAI, hybrid model, knowledge based system, ontology, computer vision.