Apprentissage continu et étiquetage automatique de données pour améliorer un réseau de neurones incertain

Quentin Christoffel, Ali Ayadi, Aline Deruyver, Anne Jeannin-Girardon

Apprentissage continu et étiquetage automatique de données pour améliorer un réseau de neurones incertain

Q. Christoffel[1], A. Ayadi[1], A. Deruyver[1], A. Jeannin-Girardon[1],

[1] Université de Strasbourg, Laboratoire ICube UMR 7357

{q.christoffel, ali.ayadi, aline.deruyver, anne.jeannin}@unistra.fr

Résumé

Les réseaux de neurones artificiels s'inspirent du fonctionnement du cerveau humain, mais sont encore très loin d'imiter le comportement humain. Cet article propose nos premières réflexions pour développer un modèle capable d'agir en toute autonomie dans le cadre de la classification d'images. L'approche proposée permet au réseau d'exprimer son incertitude, et de l'utiliser pour détecter les nouveautés qui lui sont présentées. En utilisant des graphes de connaissances, l'approche permettrait d'étiqueter automatiquement les nouveautés. Ces données étiquetées seront utilisées dans le cadre d'un apprentissage continu du modèle, afin de l'améliorer et de réduire son incertitude.

Mots-clés

Apprentissage continu, Incertitude, Détection de nouveautés, Vectorisation de connaissances, Graphes de Connaissance

Abstract

Artificial neural networks are inspired by the functionning of the human brain, but they are still far from imitating human behaviour. This paper proposes our first thoughts to develop a model capable of acting autonomously in the context of image classification. The proposed approach allows the network to express its uncertainty, and to use it to detect the novelties presented to it. By using Knowledge Graphs, the approach would allow novelties to be labelled automatically. This labelled data will be used in a continual learning setting of the model to improve it and reduce its uncertainty.

Keywords

Continual Learning, Uncertainty, Novelty Detection, Knowledge Embedding, Knowledge Graph

1 Introduction

L'être humain est capable d'apprendre pendant toute la durée de son existence. Dès le plus jeune âge, le cerveau humain fait appel à une grande partie de ses fonctions (vision, attention, mémoire, *etc.*) pour que nous puissions acquérir de nouveaux savoirs et savoir-faire. De manière simplifiée, l'apprentissage humain peut être vu comme un processus cognitif dynamique composé de deux étapes : l'acquisition de nouvelles connaissances à partir d'une succession d'expériences et leur stockage en mémoire sans les oublier [14]. Dans l'optique de simuler le comportement de l'apprentissage humain, des chercheurs ont présenté les réseaux de neurones artificiels [35], inspirés du fonctionnement des neurones humains. Les réseaux de neurones sont très efficaces pour apprendre une représentation des données et faire une prédiction à partir des données [2], mais ils n'ont pas l'autonomie d'un humain [20]. Dans ce domaine, différentes approches ont été proposées pour rendre l'apprentissage des réseaux de neurones plus proche de l'apprentissage humain :

1. Contrairement aux humains, un réseau de neurones est généralement entraîné à faire une classification parmi un nombre fixé de classes. Pour permettre à un modèle d'être capable d'apprendre à prédire de nouvelles classes, l'*apprentissage continu* [30] a été proposé. Cela permet à un modèle d'apprendre à résoudre de nouvelles tâches de manière séquentielle.
2. De plus, une personne est capable de se rendre compte qu'elle se trouve face à une nouveauté, qu'il faut donc analyser, apprendre et mémoriser une chose en plus. Pour qu'un modèle soit capable de détecter qu'il se trouve dans une situation inhabituelle, il est possible d'associer une mesure d'*incertitude* à ses prédictions [15, 29]. Pour détecter spécifiquement qu'une donnée est nouvelle, il est possible d'utiliser des méthodes de détection de nouveautés [4].
3. Dans le cas où une personne ignore quelque chose, elle peut demander de l'aide à quelqu'un de plus expérimenté pour apprendre. L'*apprentissage actif* [37] permet à un modèle de « demander » des informations concernant l'étiquette des données à un utilisateur, ou en utilisant d'autres sources d'information.

Dans ce contexte de grand manque d'autonomie de la part des réseaux de neurones, nous soulevons deux problématiques. La première vise à détecter lorsqu'un modèle « ne sait pas résoudre » une tâche afin d'améliorer ce modèle en lui apprenant davantage, tout en réduisant son incertitude. La deuxième problématique, porte sur la capacité à *étiqueter* automatiquement des données grâce aux connaissances structurées contenues dans des graphes de connaissances, en passant par une représentation commune entre les don-

nées et les connaissances.
L'approche que nous proposons dans cet article de positionnement consiste à, dans un premier temps, exploiter l'incertitude exprimée par un modèle pour détecter des données sur lesquelles le modèle « ne sait pas » faire une prédiction. Notre approche se base sur l'hypothèse que, si le modèle n'est pas totalement certain ni incertain face à des données, c'est qu'il a réussi à reconnaître des attributs dans ces données, mais qu'il n'a pas été capable de les exploiter pour faire une prédiction, donc nous considérons que le modèle est face à des nouveautés. Dans un second temps, l'incorporation de connaissances externes, provenant de graphes de connaissances, permettra d'étiqueter ces nouveautés. L'idée est de trouver une représentation commune entre les données et les connaissances afin de les comparer et trouver l'étiquette correcte. Ces nouvelles données annotées seraient ensuite utilisées dans le cadre d'un apprentissage continu pour permettre au modèle à la fois de réduire son incertitude, et aussi d'apprendre de nouvelles tâches.

2 État de l'art

2.1 Incertitude et apprentissage automatique

Pour donner à un modèle d'apprentissage automatique plus d'autonomie, en permettant au modèle de dire qu'il « ne sait pas résoudre » une tâche, une approche consiste à introduire la notion d'incertitude. Il faut d'abord distinguer deux concepts : l'*exactitude* et la *certitude* d'une prédiction. Si la prédiction faite par le modèle est celle qui est attendue, elle est exacte. Cependant, cette prédiction n'est généralement pas accompagnée d'une mesure de confiance : il se peut que le modèle ait fait une prédiction exacte, mais sans en être certain, d'où la nécessité de pouvoir mesurer l'incertitude d'une prédiction. L'incertitude peut ensuite être utilisée dans le cas où elle est trop élevée, pour donner au modèle la possibilité de dire qu'il « ne sait pas résoudre » le problème qui lui a été donné. Cette mesure d'incertitude peut donc être très utile dans des domaines où les prédictions peuvent avoir des conséquences sur la vie des personnes, comme en médecine ou dans le cas de la conduite autonome.
Il existe deux types d'incertitude [15]. L'incertitude *aléatoire* est inhérente aux données, elle peut être due par exemple à un capteur qui introduit une erreur de mesure. L'incertitude *épistémique* est causée par un manque d'informations ou de connaissances d'un domaine. Contrairement à l'incertitude aléatoire, l'incertitude épistémique peut généralement être réduite avec plus de données, si celles-ci couvrent mieux le domaine étudié.
Au vu des définitions ci-dessus, un objectif à atteindre est d'avoir une incertitude élevée lorsque des données hors distribution ou bruitées sont présentées au modèle, et une incertitude plus faible si le modèle a bien appris à traiter des données d'un domaine similaire.
Généralement, les modèles font une prédiction en un point, ce qui permet de prédire une valeur proche de la moyenne des vraies valeurs possibles, mais cela ne permet pas d'avoir une information sur l'incertitude du modèle. Au lieu de prédire uniquement un point, une solution consiste alors à obtenir une distribution de prédictions, qui exprime directement l'incertitude d'un modèle. On peut ensuite extraire de cette distribution différentes mesures statistiques (moyenne, médiane, *etc*). En particulier, la variance permet de quantifier l'incertitude du modèle, car elle mesure la dispersion des prédictions : plus la variance est faible, plus le modèle est certain, et plus elle est élevée, plus le modèle est incertain.
Il y a plusieurs méthodes pour obtenir une distribution de prédiction. Certaines se basent sur les statistiques Bayésiennes qui permettent, grâce au théorème de Bayes, de mettre à jour des probabilités à partir de connaissances antérieures et de données. Blundell *et al*. [7] ont présenté un modèle où chaque poids et chaque biais est représenté par une distribution, ce qui autorise une incertitude au niveau des paramètres. En théorie, l'apprentissage de ces distributions est possible grâce à l'inférence Bayésienne, mais en pratique, l'inférence Bayésienne exacte est insoluble à cause du nombre élevé de paramètres présent dans un réseau de neurones [7]. Les auteurs présentent leur algorithme *Bayes by Backprop* qui permet d'approximer l'inférence Bayésienne. Un avantage de leur approche est que, bien que le modèle requiert deux fois plus de paramètres, le résultat permet d'avoir un ensemble infini de modèles à travers l'échantillonnage de chaque distribution. Puisque les poids ne sont pas fixes, le modèle ne fait pas toujours la même prédiction sur une même entrée et ceci permet d'estimer à quel point il est certain ou non de sa prédiction. Il faut pour cela répéter plusieurs fois la prédiction sur les mêmes données et analyser la distribution des résultats prédits. Ce type de réseau avec des poids représentés par des distributions est appelé un *réseau de neurones Bayésien*.
Pour éviter d'entraîner un réseau de neurones Bayésien, qui a plus de paramètres, et s'entraîne plus lentement, et converge moins vite [10], Gal *et al*. [10] ont présenté une méthode permettant d'approximer un comportement Bayésien à partir d'un réseau de neurones. Leur approche consiste à ajouter du *dropout* [40] entre chaque couche du modèle. Le *dropout* est une méthode de régularisation utilisée pendant l'entraînement d'un modèle qui permet de désactiver des neurones dans les couches du réseau. La méthode proposée dans [10] consiste à continuer à utiliser le dropout après l'entraînement. Par conséquent, à chaque prédiction, des neurones différents sont utilisés et la sortie du modèle est non-déterministe comme dans le cas du réseau de neurones Bayésien.
Une autre méthode consiste à entraîner un ensemble de modèles profonds, où chaque modèle est entraîné sur les mêmes données [18]. Comme les modèles peuvent être entraînés en parallèle, cela ne demande pas plus de temps d'entraînement. L'utilisation de l'ensemble des modèles pour faire une prédiction permet d'avoir une distribution de résultats.
L'incertitude exprimée par un modèle à propos d'une prédiction nous permet de savoir s'il était certain de la classe prédite, ou s'il a prédit la classe par hasard. Dans ce dernier cas, nous voulons réduire l'incertitude du modèle en lui ap-

prenant à mieux connaître cette classe, voire à en apprendre une nouvelle. Nous cherchons donc à améliorer notre modèle, comme l'évoque notre première problématique. Pour améliorer le modèle, nous envisageons d'utiliser l'apprentissage continu, dont une définition et un état de l'art sont présentés dans la section suivante.

2.2 Apprentissage continu

L'apprentissage continu a pour but d'entraîner un modèle de manière à ce qu'il puisse résoudre des tâches qui lui seront présentées de manière séquentielle. Pour citer un exemple basique d'un tel problème, le jeu de données *MNIST* [19] peut être découpé en différentes tâches à apprendre : d'abord les classes $\{0, 1\}$ puis $\{2, 3\}$, $\{4, 5\}$, *etc*. Le modèle se voit donc présenté uniquement les données de la première tâche à apprendre, puis les données de la deuxième tâche et ainsi de suite pour toutes les tâches. L'objectif est d'obtenir un modèle qui arrive à apprendre les nouvelles tâches et qui parvient toujours à résoudre les tâches précédentes. La difficulté de cette approche se situe dans l'ajout de connaissances au modèle pour qu'il résolve les nouvelles tâches sans oublier ce qui a déjà été appris précédemment. Le problème étant que si on ne veille pas à conserver ce que le modèle a appris, il peut se produire de l'*oubli catastrophique* [16], qui est une perte soudaine de performance sur les tâches déjà apprises lors de l'apprentissage de la tâche courante. On parle ici de compromis *plasticité/stabilité* [22], définissant la capacité de s'adapter à de nouvelles tâches (plasticité) tout en conservant les performances atteintes sur d'autres tâches (stabilité).

Plusieurs objectifs peuvent être atteints avec l'apprentissage continu [31] : le principal objectif est qu'on ne veut pas que le modèle oublie ce qu'il a appris. Il est, de plus, souhaitable que l'utilisation de la mémoire et des ressources de calcul par le modèle soit fixe ou augmente légèrement lorsque de nouvelles tâches sont apprises. Il faut aussi faire un choix concernant deux points importants : l'autorisation (ou non) d'avoir du *Forward Transfer* ou du *Backward Transfer* dans le modèle. Imaginons que nous avons un modèle déjà entraîné sur les tâches 0 à $t-1$ et qu'on veut apprendre la tâche t. Le Forward Transfer est le fait que l'apprentissage des tâches précédentes ait une influence sur l'apprentissage de la tâche t. Cela peut avoir un effet positif dans le cas où le modèle présente de meilleures performances sur la tâche t, et un effet négatif si le modèle n'arrive pas à bien apprendre la tâche t. Le Backward Transfer représente la situation inverse : on considère ici l'influence de l'apprentissage de la tâche t sur les performances du modèle sur les tâches précédentes. L'effet est positif si les performances sont améliorées. Un dernier objectif consiste à ne pas conserver les données des tâches précédentes, ou du moins aussi peu que possible. Dans le cas d'une application réelle de l'apprentissage continu qui traiterait un flux de données contenant de nouvelles tâches à apprendre, il pourrait être impossible de conserver toutes les données pour des raisons d'espace de stockage ou des raisons légales [1].

Pour atteindre certains de ces objectifs, plusieurs méthodes ont été proposées. Par exemple, la méthode de *Model Growing* proposée par [36] consiste dans un premier temps à définir une structure de base pour le réseau qui sera capable de traiter une tâche. Puis, à chaque nouvelle tâche, la capacité du modèle est augmentée en ajoutant cette structure de base au réseau déjà existant. Cette méthode permet d'utiliser du Forward Transfer en liant les neurones de l'ancienne structure à la nouvelle, mais pas de Backward Transfer, car les neurones de la nouvelle structure ne sont pas connectés à l'ancienne. D'autres méthodes existent comme l'*isolation de paramètres* [25], la *régularisation* [16], et la *distillation de connaissances* [21]. Il existe aussi des méthodes basées sur la *répétition* des données, qui consistent à stocker ou générer un certain nombre de données pour chaque tâche [23, 38]. Ces méthodes permettent de limiter l'oubli des tâches précédentes en les rappelant au modèle.

L'apprentissage continu est la méthode qui nous permettra d'améliorer notre modèle. Il faut pour cela avoir de nouvelles données avec leur étiquette pour les fournir à notre modèle. Notre deuxième problématique porte donc sur le fait d'étiqueter automatiquement des données grâce à des connaissances contenues dans des graphes de connaissances, en utilisant une représentation commune entre les données et les connaissances. La section suivante présente les graphes de connaissances et différentes méthodes pour changer leur représentation.

2.3 Vectorisation des connaissances

Bien que les graphes de connaissances ne soient pas récents, ils ont gagné en popularité et sont désormais un élément clé dans de nombreuses applications d'intelligence artificielle liées à la recherche rapide et contextuelle d'information ainsi qu'à la prise de décision, citons à titre d'exemples DBpedia [33], Google Knowledge Graph [27], Wikidata [41], etc. Comme les ontologies, ces graphes fournissent une représentation de la connaissance relative à un domaine particulier sous une forme facilement exploitable par la machine. Ils représentent un ensemble d'entités reliées entre elles, composés de nœuds qui décrivent les entités (objets et concepts), et d'arcs modélisant les relations entre ces entités. Les relations peuvent être enrichies par des attributs ou des valeurs quantitatives représentant le poids de la relation. Les graphes de connaissances sont relativement faciles à développer et à interpréter, ce qui en fait un important outil pour décrire la sémantique de grands volumes de données issus de multiples sources, hétérogènes ou incomplètes [9].

Une représentation commune permettrait d'établir un lien entre des données et les connaissances stockées dans les graphes de connaissances. On peut par exemple utiliser une représentation sous forme vectorielle. Pour cela, il existe différentes méthodes qui se basent sur un même principe : il faut dans un premier temps transformer le graphe de connaissance en une séquence de mots qui pourra ensuite être vectorisée avec des méthodes comme Word2Vec [12]. Pour générer ces séquences, la technique *graph walks* consiste à parcourir un graphe en partant de chaque sommet et en parcourant tous les chemins qui y sont liés avec une profondeur définie. Chaque sommet et chaque relation

parcourus permettent de former une phrase. Par exemple, avec les sommets « chat » et « félin » reliés par la propriété « est un type de » allant de « chat » vers « félin » la phrase générée serait « un chat est un type de félin ». D'autres algorithmes de parcours de graphe comme la méthode de *Weisfeler Lehman Subtree RDF Graph Kernel* [8] permettent de se focaliser sur des sous-graphes particuliers. Ces deux méthodes de transformation de graphe en séquences de mots ont été utilisées par Ristoki et Paulheim pour créer l'outil RDF2Vec [34] permettant de transformer des entités et des relations en une représentation vectorielle. Un autre moyen d'obtenir des séquences de mots à partir d'un graphe est d'utiliser un raisonneur pour inférer de nouveaux axiomes logiques. Cette méthode a été utilisée par Smaili *et al*. dans leur approche Onto2Vec [39].

Après avoir transformé ces structures de connaissances en représentations vectorielles, se pose la question de l'évaluation de leur qualité. Initialement, l'évaluation était extrinsèque, utilisant la vectorisation générée comme attributs d'entrée d'un modèle. Ainsi, la qualité des représentations vectorielles est estimée en fonction de la qualité des résultats du modèle. Si les résultats ne sont pas satisfaisants, la représentation ne l'est pas non plus. Alshargi *et al*. [5] ont présenté des métriques pour évaluer la qualité de ces vectorisations intrinsèquement, sans devoir les utiliser en entrée d'un modèle. Une première métrique consiste à mesurer la *catégorisation* des entités par rapport à leur concept, en calculant la moyenne des représentations de toutes les entités typées par un concept, puis en mesurant la distance entre cette moyenne et la représentation du concept. Une distance faible reflète une représentation de bonne qualité. Une seconde métrique, qui a cette fois pour but d'évaluer la conservation du comportement hiérarchique, est l'*erreur sémantique absolue*. Le calcul de cette métrique nécessite d'avoir une mesure de similarité entre *deux concepts*. On calcule ensuite la similarité entre *les représentations de ces concepts*. Avec une représentation correcte, on peut s'attendre à ce qu'il y ait une corrélation entre ces deux mesures. Alshargi *et al*. [5] ont notamment évalué l'approche RDF2Vec [34] et conclu que les méthodes de vectorisation actuelles des graphes de connaissances ne permettent pas d'obtenir une vectorisation d'aussi bonne qualité qu'une vectorisation effectuée sur un corpus de texte, permettant, elle, d'obtenir un contexte plus riche et donc une meilleure vectorisation. De plus, les auteurs montrent qu'il n'y a pas de méthode de vectorisation meilleure que toutes les autres : chaque méthode capture des éléments spécifiques des concepts et il faut de ce fait choisir la méthode appropriée en fonction des besoins de la tâche extrinsèque qui utilisera les représentations.

Ce tour d'horizon de différents domaines et méthodes donne un aperçu des notions sur lesquelles notre approche s'appuie.

3 Approche proposée

Dans cette section, nous présentons nos idées et réflexions concernant une approche répondant aux deux problématiques présentées en introduction. La première portant sur la détection des cas où un modèle ne sait pas résoudre une tâche et la deuxième sur l'étiquetage de données en utilisant des connaissances issues de graphes de connaissances.

Lors de la conférence *Computer Vision and Pattern Recognition* (CVPR) 2021, Aljundi [4] a présenté une vision d'ensemble de la mise en place d'un agent autonome. Cela consiste dans un premier temps à entraîner un modèle, puis le déployer dans un environnement où il sera face à des situations changeantes qui nécessitent une adaptation du modèle. Pour que le modèle puisse s'adapter, il faut détecter quand il est confronté à des nouveautés. Il y a ensuite une sélection des nouveautés, pour garder et annoter celles qui permettront au modèle d'être amélioré grâce à l'apprentissage continu.

Nous nous positionnons dans un cadre similaire, mais alors qu'Aljundi présente une approche générique, nous discutons d'une approche plus bas niveau, en faisant des choix d'implémentation guidés par nos problématiques et une application dans un domaine précis : la classification d'images. Notre approche s'organise en 4 modules interconnectés comme l'illustre la figure 1. Ces modules sont détaillés dans les sections suivantes, mais n'ont pas encore été implémentés.

3.1 Modèle incertain

Notre module de base est un modèle qui est capable d'exprimer son incertitude. Cette particularité nous permet d'accorder plus de confiance aux prédictions de notre modèle et de voir ses faiblesses. Nous prévoyons aussi d'utiliser l'incertitude pour détecter les nouveautés (voir Section 3.2).

Ovadia *et al*. [29] ont évalué le comportement de différentes méthodes de prédiction d'incertitude pour vérifier si on pouvait effectivement *faire confiance à l'incertitude du modèle*. La méthode ayant obtenu les meilleurs résultats sur la plupart des métriques est celle utilisant les ensembles de modèles profonds [18]. De plus, ils ont montré que de bons résultats pouvaient être obtenus avec un petit ensemble de cinq réseaux. Leurs expériences ont aussi montré que les réseaux de neurones Bayésiens (SVI dans l'article, pour *Stochastic Variational Inference*) sont prometteurs sur des petits jeux de données comme MNIST, mais plus difficiles à utiliser avec des jeux de données comme ImageNet et des architectures complexes comme les LSTM. Dans notre cas, nous choisissons d'utiliser un réseau de neurones Bayésien, pour développer leur utilisation, car ils permettent en théorie d'entraîner un ensemble infini de modèles.

Dans les réseaux de neurones Bayésiens, les poids et les biais du modèle sont représentés par des distributions. Cette particularité permet de faire des prédictions non déterministes. Ainsi, en faisant plusieurs prédictions sur une même donnée, le résultat correspond à une distribution des valeurs en sortie pour chaque classe. Si le modèle est certain de sa prédiction, la distribution de la classe prédite a une faible variance, car le modèle prédit toujours qu'il s'agit de cette classe. Dans le cas où le modèle est incertain, toutes les distributions des classes ont une variance élevée. Pour déterminer quelle classe a été prédite par le réseau, la médiane

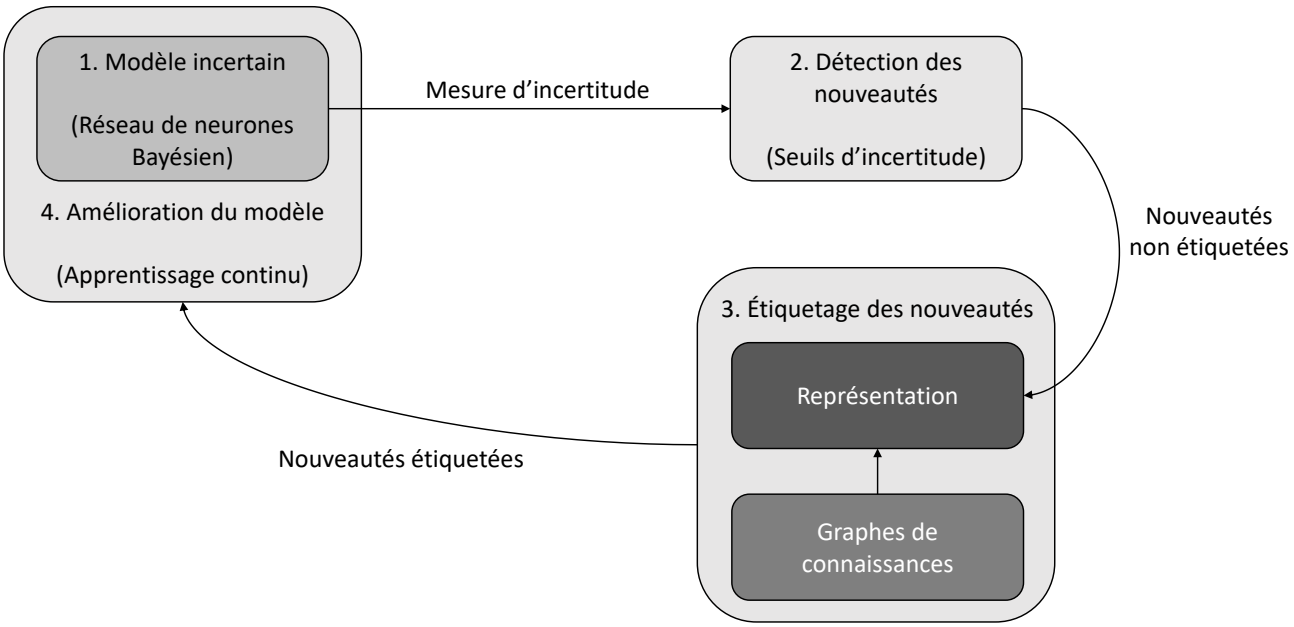

FIGURE 1 – Aperçu de l'approche proposée découpée en différents modules interconnectés.

de la distribution de chaque classe est calculée. Nous privilégions l'utilisation de la médiane puisqu'elle considère les observations et est robuste face à des valeurs extrêmes. La classe avec la médiane la plus élevée représente la prédiction finale du modèle. La médiane nous donne aussi une information sur l'incertitude, car si le modèle ne fait pas la même prédiction assez souvent, cela se reflétera sur la médiane qui sera plus basse.

3.2 Détection de nouveautés

Après l'obtention d'un modèle capable d'exprimer son incertitude, nous souhaitons détecter si des données d'une classe inconnue du modèle sont passées en entrée de celui-ci. Pour cela, nous prévoyons initialement d'utiliser l'incertitude pour détecter les nouveautés, ce qui a déjà été expérimenté dans de nombreux articles dont [32]. Nous utilisons l'incertitude exprimée par le modèle pour permettre au modèle de dire qu'il « ne sait pas » à quelle classe correspond l'image qu'il a eu en entrée si l'incertitude est trop élevée.
Nous proposons de fixer deux seuils auxquels comparer la médiane de la classe prédite par le modèle. Un premier seuil élevé, qui permet de déterminer les cas où le modèle reconnaît effectivement les données. Ainsi, si la médiane est supérieure à ce seuil, le modèle a su classifier les données. Dans le cas contraire, on considère que le modèle répond qu'il « ne sait pas » de quelle classe il s'agit. Un deuxième seuil, plus faible, permet de déterminer si le modèle est complètement incertain face aux données. Si la médiane est inférieure à ce seuil, c'est que le modèle n'a pas su faire de prédiction sur les données et on les considère comme étant hors distribution. Nous rappelons que notre hypothèse principale est que si le modèle n'est pas totalement incertain et n'est pas certain non plus, il a pu reconnaître des attributs dans les données, mais n'a pas réussi à les exploiter pour prédire une classe. Nous portons par conséquent un intérêt particulier aux données qui ont été prédites avec une médiane entre ces deux seuils et nous considérons que le modèle devrait apprendre à faire des prédictions sur ces données par l'apprentissage d'une nouvelle classe. Les données détectées comme étant hors distribution ne seront pas utilisées, bien qu'elles pourraient contenir de nouvelles classes qui n'avaient tout simplement pas de rapport avec les classes que le modèle connaissait déjà. Ces données rejetées pourraient être classées de manière non-supervisée avant d'être présentées à un expert chargé de déterminer leur importance. Ces seuils seront dans un premier temps recherchés de manière empirique avec des jeux de données appropriés aux différentes catégories que l'on veut séparer : des données connues du modèle, des données qu'il devra apprendre et des données hors distribution. Il est possible que ces seuils soient dépendants du nombre de classes, par exemple, lors d'une classification entre 1000 classes, une médiane autour de 0.5 aura plus d'importance que lors d'une classification avec 10 classes. Par la suite, nous réfléchirons à une alternative utilisant la variance de la distribution de la classe prédite, en plus de la médiane.
Dans le cas où l'utilisation de l'incertitude ne serait pas suffisante pour détecter les nouveautés, il serait possible d'utiliser d'autres méthodes. Dans un contexte similaire à ce que l'on propose, Aljundi *et al.* [3] ont étudié différentes méthodes de détection de nouveautés appliquées à un modèle entraîné grâce à l'apprentissage continu.

3.3 Étiquetage des nouveautés

Une fois que le modèle a détecté des données inconnues (mais pas complètement hors distribution) grâce à l'incertitude, l'objectif est de réussir à identifier ces données grâce à des connaissances contenues dans des graphes de connais-

sances. Par souci de clarté, dans la suite le mot « données » est utilisé pour parler des données inconnues du modèle et le mot « connaissances » est utilisé pour parler des graphes de connaissances. C'est ici que se trouve notre principale problématique : pour identifier ces données, nous devons d'abord déterminer une représentation commune entre les données et les connaissances. Plusieurs possibilités existent : dans un premier temps, il est possible de transformer à la fois les données et les connaissances sous une représentation vectorielle afin de les comparer. Pour cela, nous proposons de générer des représentations vectorielles des connaissances à l'aide des approches de transformation de graphes comme RDF2Vec [34] ou Onto2Vec [39]. Ces représentations seront évaluées grâce aux métriques intrinsèques présentées par [5]. Étant donné que nous nous intéressons à des images, il peut être envisageable de générer une description textuelle de l'image [13], au coût d'un modèle supplémentaire. Cette description pourra ensuite être vectorisée pour être comparée aux connaissances vectorisées. Le modèle de description doit être générique au sens où il doit être capable de décrire les éléments de base présents sur l'image et leur organisation. Si ce modèle pouvait décrire des éléments de haut niveau, cela voudrait dire qu'il a déjà connaissance de la nouvelle classe qu'on recherche, ce qui rendrait caduque la démarche de découverte de nouvelles connaissances. L'espace de représentation latent de notre modèle pourrait aussi être comparé aux représentations vectorielles des connaissances, mais comme notre modèle est non déterministe, il faudra tenir compte du fait que la représentation latente des données ne sera pas toujours la même.

La deuxième méthode envisagée pour arriver à une représentation commune consiste à transformer les images sous forme d'un graphe [6], les connaissances étant déjà stockées sous forme de graphe. Une piste à explorer consiste à voir s'il est possible d'appliquer cette approche de transformation en graphe pour d'autres types de données.

Initialement, nous supposerons que les nouvelles classes à ajouter au modèle sont présentes et décrites dans les graphes de connaissances utilisés. Mais dans une application réelle, il est possible que les graphes de connaissances ne contiennent pas d'informations concernant les nouvelles données. Dans ce cas, une piste envisagée est un apprentissage conjoint entre le modèle et les structures de connaissances, ce qui peut également nécessiter de transformer les données en un graphe. Dans un dernier recours, il reste toujours la possibilité de faire intervenir un expert pour identifier les données ou enrichir les graphes de connaissances.

Après avoir transformé les données et les connaissances dans une représentation commune, il faudra trouver une manière de faire un lien entre les représentations dans le but d'attribuer une étiquette aux données. On peut par exemple penser à une mesure de similarité entre deux représentations vectorielles, en utilisant la similarité cosinus ou en passant par un modèle supplémentaire chargé d'apprendre la similarité [26]. Il existe aussi des méthodes de mesure de similarité entre graphes [24].

3.4 Amélioration du modèle

Une fois que les données ont été annotées, nous voulons ajouter cette connaissance acquise dans le modèle : si les données correspondent à une nouvelle classe, il faudra que le modèle l'apprenne, si la classe était déjà connue, cela permettra au modèle de renforcer sa connaissance. L'objectif de cet ajout de connaissances est de réduire l'incertitude du modèle quand il sera confronté à des données de cette classe. Dans cette optique d'ajout de connaissance, il est possible de se limiter à une modification du modèle seulement si un nombre suffisant de données ont préalablement été annotées. Le cas opposé, où seul un petit ensemble de données annotées permet la modification du modèle revient à faire de l'apprentissage en *few-shot* [42]. Pour améliorer le modèle, une approche d'apprentissage continu peut être mise en place. En particulier, utiliser une méthode qui permet le Forward Transfer est important, puisque notre approche se base sur l'hypothèse que le modèle connaît déjà des attributs présents dans les données, il faut que ce que le modèle a déjà appris puisse avoir une influence positive sur la nouvelle tâche qu'il va apprendre. Il faudra étudier dans quelle mesure le Backward Transfer pourra être utilisé, car les données annotées grâce aux connaissances peuvent introduire des erreurs dans le modèle et on ne veut pas que la nouvelle tâche réduise les performances du modèle sur les tâches précédemment apprises. Les approches de type *model growing* seront potentiellement à éviter, car dans un scénario réel le nombre de tâches n'est pas censé être connu à l'avance, donc la taille du modèle pourrait fortement augmenter. Dans le contexte que nous présentons, il est possible qu'il y ait naturellement de la *répétition* des données des tâches précédentes : si le modèle est devenu trop incertain dans ses prédictions d'une classe (s'il y a eu de l'oubli), alors une donnée de cette classe avec son étiquette pourra tout à fait être présentée au modèle. Nguyen *et al*. [28] présentent une méthode d'apprentissage continu appliquée au cas particulier des réseaux de neurones bayésiens (dont les poids sont représentés par des distributions), en utilisant les poids appris après chaque tâche comme distribution antérieure des poids de la tâche suivante.

D'autres approches [17], travaillent plutôt sur l'injection de connaissances d'un domaine à partir de graphes de connaissances directement dans les couches du modèle. Ces approches, connues sous le nom de « *Knowledge Infused Learning* » [11], permettraient de relier des caractéristiques couvrant différentes dimensions contextuelles d'un problème, en relevant les défis lexicaux et sémantiques spécifiques au domaine, tels que la rareté, l'ambiguïté et le bruit pour la classification selon une échelle d'évaluation informée par des experts du domaine. Ces travaux s'intéressent à deux questions. La première, comment décide-t-on d'injecter ou non des connaissances à un stade particulier de l'apprentissage, et comment mesurer cette injection de connaissances. La deuxième porte sur la manière dont on peut combiner les représentations de données situées entre les couches du modèle avec des représentations de connaissances externes issues de graphe de connaissances.

D'après les auteurs, cette approche d'injection de connaissances aborde des défis fondamentaux de l'IA, à savoir la réduction des données volumineuses, renforcer l'explicabilité des décisions du modèle, et améliorer la couverture des données et connaissances spécifiques à un domaine qui ne seraient pas considérées autrement.

4 Conclusion

Dans cet article, nous nous positionnons sur une approche divisée en plusieurs modules, permettant d'obtenir un modèle capable d'agir en toute autonomie dans le cadre de la classification d'images. Nous proposons d'utiliser un réseau de neurones Bayésien afin d'avoir un modèle capable d'exprimer son incertitude lors d'une prédiction. Cette incertitude est ensuite utilisée pour faire de la détection de nouveautés. Nous considérons que si le modèle n'est pas totalement certain, ni totalement incertain lors d'une prédiction, c'est qu'il a reconnu des attributs dans les données et qu'il s'agit donc de nouveautés qu'il peut être utile d'apprendre. Ces nouveautés doivent d'abord être étiquetées automatiquement en passant par une représentation commune aux données et aux connaissances stockées dans des graphes de connaissances. L'étape finale consiste à améliorer le modèle par apprentissage continu, grâce à l'utilisation des données étiquetées, ce qui permettra au modèle de réduire son incertitude.

Nos objectifs futurs consistent à mettre en place les différentes étapes de cette approche, en commençant par évaluer l'influence de l'apprentissage continu sur la mesure d'incertitude. La problématique de l'annotation automatique des données est le point le plus important, qui nécessitera des travaux approfondis.

Dans ce processus autonome qui se rapproche du raisonnement humain, il pourrait être intéressant d'intégrer un module supplémentaire, qui ajouterait de l'explicabilité au niveau des différents modules. Cela permettrait d'améliorer l'approche grâce à une meilleure compréhension, et donnerait encore plus de transparence et de confiance au processus.

Références

[1] General Data Protection Regulation (GDPR) – Official Legal Text. https ://gdpr-info.eu/.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications : A survey. *Heliyon*, 4(11) :e00938, November 2018.

[3] Rahaf Aljundi. Continual learning : A story line and wider view. Conference on Computer Vision and Pattern Recognition, 2021. page 18.

[4] Rahaf Aljundi, Daniel Olmeda Reino, Nikolay Chumerin, and Richard E. Turner. Continual Novelty Detection. *arXiv :2106.12964 [cs]*, June 2021.

[5] Faisal Alshargi, Saeedeh Shekarpour, Tommaso Soru, Amit P. Sheth, and Uwe Quasthoff. Concept2vec : Metrics for evaluating quality of embeddings for ontological concepts. *CoRR*, abs/1803.04488, 2018.

[6] Pedro H. C. Avelar, Anderson R. Tavares, Thiago L. T. da Silveira, Cláudio R. Jung, and Luís C. Lamb. Superpixel Image Classification with Graph Attention Networks. *arXiv :2002.05544 [cs, stat]*, November 2020.

[7] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[8] Gerben Klaas Dirk De Vries and Steven De Rooij. Substructure counting graph kernels for machine learning from rdf data. *Journal of Web Semantics*, 35 :71–84, 2015.

[9] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4) :2, 2016.

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning. *arXiv :1506.02142 [cs, stat]*, October 2016.

[11] Manas Gaur, Ugur Kursuncu, Amit Sheth, Ruwan Wickramarachchi, and Shweta Yadav. Knowledge-infused deep learning. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, pages 309–310, 2020.

[12] Martin Grohe. Word2vec, node2vec, graph2vec, x2vec : Towards a theory of vector embeddings of structured data. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, pages 1–16, New York, NY, USA, 2020. Association for Computing Machinery.

[13] MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CsUR)*, 51(6) :1–36, 2019.

[14] Knud Illeris. *Contemporary Theories of Learning : Learning Theorists... in Their Own Words*. Routledge, 2009.

[15] H. M. Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural Network-Based Uncertainty Quantification : A Survey of Methodologies and Applications. *IEEE Access*, 6 :36218–36234, 2018.

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13) :3521–3526, March 2017.

[17] Ugur Kursuncu, Manas Gaur, and Amit Sheth. Knowledge infused learning (k-il) : Towards deep incorporation of knowledge in deep learning, 2020.

[18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[20] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics : Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58 :52–68, 2020.

[21] Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12) :2935–2947, December 2018.

[22] Guoliang Lin, Hanglu Chu, and Hanjiang Lai. Towards better plasticity-stability trade-off in incremental learning : A simple linear connector. *CoRR*, abs/2110.07905, 2021.

[23] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and Philip S Yu. Deep graph similarity learning : A survey. *Data Mining and Knowledge Discovery*, 35(3) :688–725, 2021.

[25] Arun Mallya and Svetlana Lazebnik. PackNet : Adding Multiple Tasks to a Single Network by Iterative Pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, Salt Lake City, UT, June 2018. IEEE.

[26] Stefano Melacci, Lorenzo Sarti, Marco Maggini, and Monica Bianchini. A neural network approach to similarity learning. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 133–136. Springer, 2008.

[27] Casey Newton. Google's knowledge graph tripled in size in seven months. *CNET. CBS Interactive*, 2012.

[28] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv :1710.10628*, 2017.

[29] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty ? Evaluating Predictive Uncertainty Under Dataset Shift. *arXiv :1906.02530 [cs, stat]*, December 2019. Comment : Advances in Neural Information Processing Systems, 2019.

[30] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks : A review. *Neural Networks*, 113 :54–71, May 2019.

[31] Razvan Pascanu. Continual learning the challenge. CVPR, 2021.

[32] Xuming Ran, Mingkun Xu, Lingrui Mei, Qi Xu, and Quanying Liu. Detecting out-of-distribution samples via variational auto-encoder with reliable uncertainty estimation. *Neural Networks*, 145 :199–208, January 2022.

[33] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all ? analyzing the differences between dbpedia, yago, wikidata & co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 366–372. Springer, 2017.

[34] Petar Ristoski and Heiko Paulheim. Rdf2vec : Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016.

[35] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408, 1958.

[36] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv :1606.04671*, 2016.

[37] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[38] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[39] Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Onto2vec : Joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics (Oxford, England)*, 34(13) :i52–i60, 2018.

[40] 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.

[41] Andra Waagmeester, Gregory Stupp, Sebastian Burgstaller-Muehlbacher, Benjamin M Good, Malachi Griffith, Obi L Griffith, Kristina Hanspers, Henning Hermjakob, Toby S Hudson, Kevin Hybiske, et al. Science forum : Wikidata as a knowledge graph for the life sciences. *Elife*, 9 :e52614, 2020.

[42] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples : A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3) :1–34, 2020.