Vers l'amélioration des caractéristiques des réseaux de neurones profonds : une approche de méta-modélisation

Aïcha Choutri[1,*], Angham Boukhari[2], Faiza Belala[1] et Ahmed Hadj Kacem[2]

[1]*Université de Constantine 2-Abdelhamid Mehri, Laboratoire LIRE, BP : 67A, Constantine, Algérie*
[2]*ReDCAD Laboratory, ENIS, University of Sfax, Tunisia*

Résumé

L'apprentissage profond ou Deep Learning (DL) est un type de machine learning géré par des réseaux de neurones dits profonds ou Deep Neural Networks (DNN). Son développement rapide a ouvert de nombreuses opportunités conduisant à des avancées majeures dans plusieurs domaines d'application de l'IA (Intelligence Artificielle). Il a permis d'atteindre des performances record dans beaucoup de tâches, dépassant souvent celles des méthodes traditionnelles. Cependant, il présente également des défis et des questions éthiques dont les raisons sont étroitement liées à la diversification des DNNs existants et des méthodes appliquées à leur configuration architecturale et à leur apprentissage ainsi qu'aux difficultés rencontrés par les spécialistes en science des données dans le processus fastidieux et coûteux d'amélioration de leurs performances. Dans notre travail, nous nous intéressons à l'application des méthodes formelles au DL afin de prendre en charge un certain nombre de ces défis, en relation avec la fiabilité, la sécurité, la robustesse et la transparence des modèles. L'objectif majeur de ce papier est de détailler les résultats de la première étape de notre approche. Il s'agit d'un méta-modèle défini et élaboré selon une technique de l'IDM (Ingénierie Dirigée par les Modèles) qui contribue à la transparence des modèles considérés, les DNNs dans notre cas, leur évolutivité, leur réutilisation et leur compréhension. L'originalité de notre approche réside dans le fait que les modèles soient au centre du processus du DL en général, ce qui permettra d'améliorer en particulier, les performances de ces réseaux ou d'associer naturellement des modèles formels ayant des sémantiques rigoureuses à ce type de réseaux assez complexes dans leur comportement.

Mots-clés

Réseaux de neurones profonds (DNNs), Apprentissage profond (DL), Méta-modélisation, Ingénierie dirigée par les modèles (IDM), Méthodes Formelles

1. Introduction

L'apprentissage profond (DL : Deep Learning) est l'une des branches les plus importantes et les plus exploitées de l'apprentissage automatique (ML : Machine learning). Il se caractérise par la capacité d'acquérir des connaissances en s'entrainant tout en imitant les êtres humains dans leur façon de raisonner et d'apprendre. Il est un élément clé de l'IA (Intelligence Artificielle) et se base sur des réseaux de neurones artificiels appelés aussi réseaux de neurones profonds (DNNs:Deep Neural Networks). Son développement rapide a ouvert de nombreuses opportunités, conduisant à des avancées majeures dans des domaines tels que la vision par ordinateur, la reconnaissance vocale, la traduction automatique et bien d'autres. Grâce à son succès, il a réalisé un saut quantique dans le domaine de l'IA pour devenir le domaine le plus important dans le monde de l'informatique. Ainsi, il présente des avantages significatifs, tels que :

- Amélioration des performances dans plusieurs domaines.
- Automatisation des tâches complexes avec économie de temps et réduction des coûts.
- Capacité d'adaptation des modèles de DL à des domaines variés.
- Soutien et facilité de la recherche dans différents domaines tels que la biologie, la médecine, la climatologie, etc.
- Développement de systèmes d'IA plus puissants.

TACC 2023: Tunisian-Algerian Joint Conference on Applied Computing, November 6 - 8, Sousse, Tunisia

[*] Corresponding author.

aicha.choutri@univ-constantine2.dz (A. Choutri); anghamboukhari@fsegs.u-sfax.tn (A. Boukhari); faiza.belala@univ-constantine2.dz (F. Belala); ahmed.hadjkacem@fsegs.usf.tn (A. Hadj Kacem)

Cependant, il est important de noter que le développement rapide de DL et implicitement des DNNs, présente également des défis et des questions éthiques, tels que la nécessité de garantir la transparence des modèles, de résoudre les problèmes de biais algorithmique et de préserver la vie privée des individus. Ces défis doivent être pris en compte pour assurer un développement responsable de cette technologie. En outre, malgré l'existence de nombreux types de DNNs et malgré leur succès remarquable, les DNNs demeurent difficiles à concevoir et à mettre en œuvre. En effet, les modèles DNNs qui conviennent à une application peuvent ne pas convenir à une autre. De même, les performances des modèles DNNs varient d'une plate-forme matérielle à l'autre [1]. Ceci d'une part, d'autre part, les spécialistes en science des données sont confrontés à la tâche complexe d'évaluer différents algorithmes d'apprentissage et d'ajuster les paramètres pour obtenir de meilleures performances. Ce processus manuel est fastidieux, coûteux en termes de temps et de ressources, et incertain quant à la création de modèles adaptés à des problèmes spécifiques [2]. Par ailleurs, l'utilisation croissante des DNN dans une variété d'applications, y compris certaines critiques pour la sécurité, a suscité un regain d'intérêt pour la question de la vérification des réseaux de neurones. Cependant, la vérification est plus significative lorsqu'ils sont exécutés avec des spécifications formelles de haute qualité [3]. Notre étude aborde cette problématique et propose comme solution, une approche de modélisation formelle nouvelle des DNNs afin de prendre en charge l'amélioration de leurs caractéristiques. Nous y exploitons : (1) les techniques et les outils de l'IDM (Ingénierie dirigée par les Modèles) [4] et (2) le langage formel Maude et ses outils [5]. Cette approche définie selon un modèle en cascade et représentée par un diagramme d'activité UML (Unified Modeling language) [6], comporte quatre étapes principales (voir figure 1) et se traduit en fait par un processus automatisable. Chaque étape est décomposée en plusieurs étapes réalisant les tâches en question. Les résultats fournis (objet détaché) par l'activité de chacune, sont utilisés par l'activité de l'étape suivante.

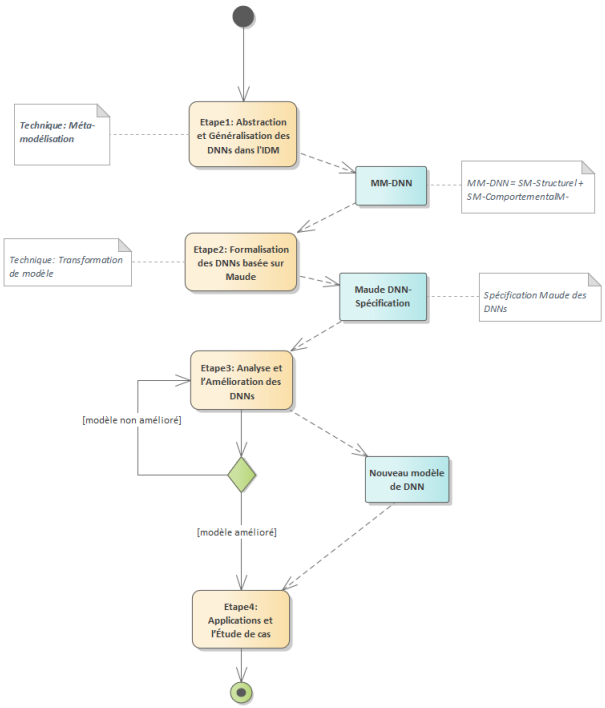


Figure 1: Processus général de modélisation formelle et amélioration des DNNs

Comme chaque étape est un processus en soit, nous ne présenterons dans ce papier que la première où les modèles sont au centre de DL. Cette étape de méta-modélisation peut simplifier la maintenance à long terme des DNNs car les modifications peuvent être apportées au niveau

du méta-modèle, évitant ainsi des changements pénibles et répétitifs sur de nombreux modèles. Cette étape, comme indiquée sur la figure 1, aide à gérer la complexité des DNNs en les décomposant en éléments plus simples et en établissant des relations claires entre eux. Elle permet de modéliser non seulement la structure statique des DNNs, mais aussi leur comportement dynamique. Cela facilite leur simulation et l'analyse de leur dynamique. Le méta-modèle MM-DNN proposé est d'une grande utilité pour la documentation et l'évolution des DNNs, notamment lorsque les besoins ou les spécifications changent. De plus, il constitue un premier pas facilitant le passage vers la définition de modèles formels. Nous montrerons dans un travail futur comment ce méta modèle servira de base pour associer une spécification Maude exécutable [5] à ce type de réseaux. Evidemment, cette approche formelle à travers toutes ses étapes permet de garantir la fiabilité et la sécurité des algorithmes d'apprentissage profond, en identifiant et en corrigeant les erreurs potentielles avant qu'elles ne conduisent à des résultats incorrects ou indésirables. Cela est particulièrement important dans les domaines critiques. De plus, elle offrira un moyen pour spécifier et vérifier formellement des propriétés essentielles des DNNs, telles que la convergence, la stabilité, l'invariance, l'équité, ou la robustesse.
La suite de ce papier est organisée comme suit : La Section 2 introduit les concepts fondamentaux des DNNs et de l'IDM sur lesquels repose notre travail. La Section 3 est ensuite consacrée à la présentation de notre méta-modèle MM-DNN résultat de l'étape 1 de la figure 1.pour la définition abstraite et générale des DNNs, selon une approche de méta-modélisation dans l'IDM tout en détaillant la description des différents sous-modèles. La Section 4 examine les travaux connexes pour situer le nôtre dans un contexte plus large et souligner sa pertinence. Enfin, la Section 5 conclut l'article en formulant quelques commentaires et en esquissant les orientations futures de notre recherche.

2. Concepts fondamentaux

Dans cette section, nous rappelons les définitions clés des DNNs et de l'IDM en particulier, le principe de l'Architecture Dirigée par les Modèles connue sous le nom approche MDA (Model Driven Architechture).

2.1. DNNs

Les DNNs constituent une classe spéciale de réseaux de neurones artificiels, caractérisée par leur architecture en couches. Un DNN agit comme une fonction mathématique qui mappe certains ensembles de valeurs d'entrée à des valeurs de sortie. Cette fonction est une composition de fonctions plus simples, permettant ainsi au réseau de traiter des tâches de plus en plus complexes grâce à cette hiérarchie de couches de neurones [7]. Une caractéristique clé des DNNs est leur capacité à apprendre de manière automatique des représentations significatives à partir des données d'entraînement.

2.1.1. Architectures et types de DNN

DNN est l'association de plusieurs neurones groupés en couches reliées par des connexions pondérées comme le montre la figure 2 [8]. Son architecture détermine la manière dont les neurones sont ordonnés et connectés au sein d'un même réseau. En général, un DNN est composé de plusieurs couches de neurones successives : les entrées (x), les couches cachées (z, qui ne sont pas accessibles en dehors du réseau) jusqu'à la couche de sortie (y). La "profondeur" d'un DNN est déterminée par le nombre de ses couches cachées. Une des architectures les plus basiques est le perceptron multicouches ou DNN à propagation avant (feed-forward) [9], qui transmet l'information de l'entrée à la sortie [8], comme représenté dans la figure 2.

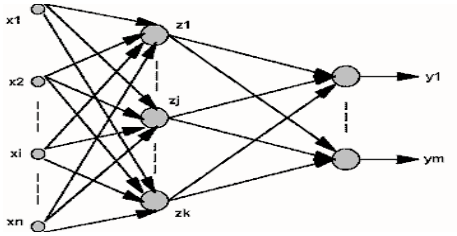


Figure 2: Représentation schématique d'un DNN avec ses couches et connexions

Les DNNs se déclinent en diverses architectures en fonction de leur structure et de leur fonctionnement. Le tableau 1 présente différents types de DNN ainsi que leurs principales applications:

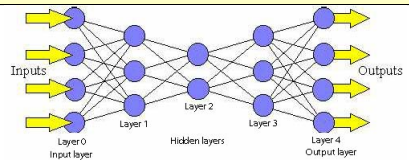
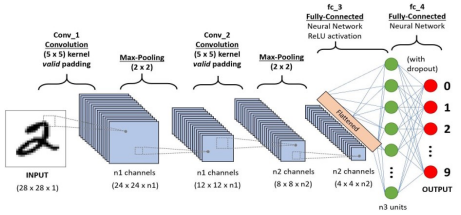
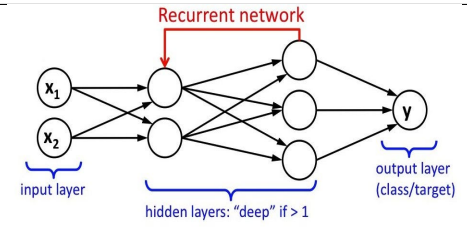
Types de DNN	Définition	Architecture	Domaines d'Application
Feedforward	Type acyclique, Information circule de l'entrée à la sortie		Classification, régression, autres tâches de prédiction
CNN	Conçu pour tâches de vision, Couches de convolutions et pooling		Vision par ordinateur, reconnaissance d'images, détection d'objets
RNN	Conçu pour traiter des séquences, Composant "mémoire" pour circulation cyclique		Traitement du langage naturel, modélisation de langage, traduction automatique

Tableau 1: Types de DNN et leurs Domaines d'Application

2.1.2. Fonctionnement

Pour la mise en place d'un DNN et avant le lancement de son fonctionnement et de l'apprentissage, des paramètres essentiels requis tels que le nombre de couches, le nombre de neurones par couche, les poids des connexions, la fonction d'activation, etc., doivent être initialisés. Le fonctionnement d'un DNN s'articule autour de la propagation des données, de l'entrée jusqu'à la sortie. Les neurones de la couche d'entrée reçoivent les données d'entrée et les transmettent aux neurones de la couche cachée suivante. Les neurones d'une couche cachée reçoivent les informations soit des neurones d'entrée, soit des autres neurones cachés. Ils combinent linéairement les sorties de la couche précédente, ajustées par les poids des liens, puis appliquent la fonction d'activation. Cette dernière (comme la sigmoïde, tanh, ReLU, etc.), permet de déterminer l'information transmise à la prochaine couche. Les neurones de la couche de sortie délivrent la sortie finale du réseau après avoir traité les informations des neurones cachés. Afin de mieux comprendre le fonctionnement des DNN et leur processus

d'apprentissage, il est essentiel d'examiner en détail le fonctionnement d'un seul neurone, comme illustré dans la Figure 3.

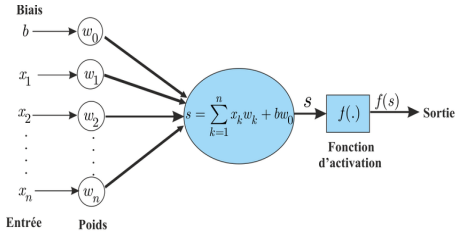


Figure 3: Structure d'un neurone artificiel [10]

Le neurone effectue une simple sommation mathématique des poids multipliés par la valeur d'entrée et ajoute un biais. Le produit de ces opérations passe par une fonction d'activation non linéaire et la sortie de la fonction d'activation est la sortie du neurone [11]. La sortie du réseau de neurones est donnée par l'expression de l'équation (1)

$$y = f(s) = f\left(\sum_{k=1}^{n} x_k \cdot w_k + b \cdot W_0\right) \tag{1}$$

Une fois que la sortie du réseau y est obtenue, l'erreur entre la sortie attendue et la sortie réelle est calculée à l'aide de la "fonction de coût".

Optimisation du réseau par l'algorithme "Rétropropagation du gradient":
L'apprentissage consiste à calculer le gradient des paramètres de la dernière couche, puis propager le gradient vers les entrées de la dernière couche (ce qui correspond au gradient de la sortie de la couche précédente) [12] dans le but de la minimisation des erreurs. Ainsi, le gradient des paramètres de l'avant-dernière couche, puis le gradient de son entrée seront calculés. Et ainsi de suite, couche à couche, de la dernière à la première. C'est ce qu'on appelle la rétropropagation du gradient [13]. A la fin, les poids et les biais seront mis à jour avec les relations suivantes :

$$W_{ji}^{l} = W_{ji}^{l} - \alpha dW_{ji}^{l} \tag{2}$$

$$b^{l} = b^{l} - \alpha db^{l} \tag{3}$$

α: représente le pas d'apprentissage.

2.1. IDM et approche MDA

L'Ingénierie Dirigée par les Modèles (IDM) également connue sous le nom de Model Driven Engineering (MDE) en anglais, est une forme d'ingénierie générative. Cette approche met l'accent sur l'utilisation de modèles comme des abstractions simplifiées pour mieux comprendre les systèmes. Ces modèles sont ensuite transformés en artefacts opérationnels à l'aide d'algorithmes spécifiques. La création de Langages de Modélisation Spécifiques au Domaine (DSML) et leur métamodélisation sont des éléments essentiels dans cette approche [14].

2.1.1. Notions de modèle et méta-modèle

Un modèle est une représentation simplifiée d'un système, conçue dans un but précis. Il doit permettre de répondre à des questions concernant le système qu'il décrit.
Les modèles servent d'entrées à des algorithmes de transformation qui génèrent des artefacts. La notion centrale de l'IDM est donc le modèle. La première relation majeure de l'IDM, entre le modèle et le système qu'il représente, *« représentationDe »*, nommée dans [15] par μ et représentée sur la Figure 4. Bien que définir ce qu'est un bon modèle reste une tâche complexe. il est tout de même évident qu'un bon modèle doit être une abstraction pertinente du système

réel, tout en étant suffisamment précis pour répondre à certaines questions à la place du système lui-même. Autrement dit, le modèle devrait fournir des réponses qui reflètent fidèlement celles du système qu'il représente. Dans l'IDM, le concept de modèle fait explicitement référence à un langage clairement défini. Pour qu'un modèle soit utile, il doit être interprétable par une machine. Le langage dans lequel ce modèle est formulé doit donc être précisément défini. Ainsi, naturellement, la définition d'un langage de modélisation a pris la forme d'un modèle, que l'on appelle métamodèle. Ce concept a mené à l'identification de la seconde relation, liant le modèle au métamodèle, appelée *« conformeA »*, et nommée dans [15] par χ et représentée sur la Figure 4.

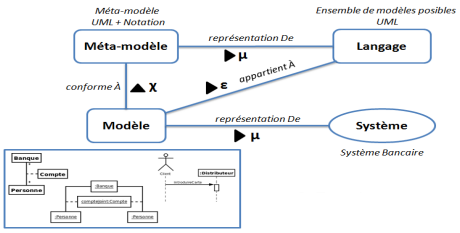


Figure 4: Relations entre système, modèle, métamodèle et langage [10]

2.1.2. Approche MDA

MDA est une méthode au cœur de l'IDM. Proposée par l'OMG (Object Management Group) en 2000, elle met l'accent sur la séparation des préoccupations, permettant d'aborder distinctement les aspects métiers et techniques d'une application à travers la modélisation. Elle automatise la génération de code à partir de modèles analytiques et conceptuels [16]. Les modèles ne sont plus seulement un élément visuel ou de communication, mais sont, dans l'approche MDA, un élément productif et le pivot du processus MDA. L'architecture MDA (figure 5) possède une structure pyramidale composée de quatre niveaux : le monde réel (M0), les modèles du système (M1), les métamodèles (M2) et le métamétamodèle (M3) [17, 18].

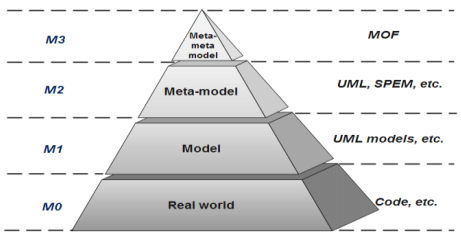


Figure 5: Pyramide de modélisation de l'OMG

3. Contribution

Dans l'Introduction du papier, nous avons bien expliqué que dans le contexte de l'avancée rapide de DL qui s'avère être un levier dans de multiples secteurs, la nécessité d'une approche systématique pour le développement de cette technologie devient de plus en plus nécessaire. L'IDM, spécifiquement dans sa forme MDA, fournit une réponse à ce défi. En effet, elle permet de créer des modèles initialement d'analyse puis de conception qui deviennent du code exécutable via des transformations, dérivations et enrichissements successifs. Ces règles de transformation sont définies en utilisant un langage de transformation QVT(«Query/View/Transformation») qui est un langage déclaratif standardisé par l'OMG [19]. L'alliance de l'IDM avec le DL permettrait alors des développements plus rapides, robustes et maintenables. Dans cette section,

nous présentons et expliquons les résultats de cette fusion qui correspond à l'étape 1 de notre approche de modélisation formelle et amélioration des DNNs (figure 1), en proposant un méta-modèle MM-DNN. Ce dernier est en fait, une nouvelle définition des DNNs de manière abstraite et générale mais concise et complète. En outre, il capitalise sur les atouts de l'IDM et de ce fait, il a l'avantage de renforcer la compréhension et la réutilisation des composants DNN, tout en offrant une extensibilité optimale.

3.1. Principe général

Le principe général de notre contribution repose sur la considération des DNNs comme des systèmes ayant une architecture logicielle en distinguant bien une structure et un comportement modélisables dans l'IDM tout en tenant compte des caractéristiques spécifiques de ce type de réseaux. Le concept de DNN est l'élément clé de cette perspective. L'idée principale est de définir un langage commun en terme de méta-modèle MM-DNN (figure 6) dans lequel tous les types de DNN peuvent être définis, manipulés et transformés facilement dans 'd'autres langages dans un but donné. En utilisant comme outils, le langage ECORE [20] et l'IDE Eclipse, l'élaboration de MM-DNN consiste donc à distinguer deux sous-modèles complémentaires : SM-Structurel pour modéliser les éléments architecturaux de base de la structure hiérarchique (couches, neurones et liens) et la topologie de leurs connexions ; SM-Comportemental englobant tous les aspects liés au fonctionnement d'un DNN (Initialisation de paramètres, Traitement + Propagation des données et Apprentissage). Comme le montre la figure 6, les éléments de modélisation utilisés dans MM-DNN sont principalement: les Méta-classes et les relations (spécialisations, associations et compositions). Les cardinalités des associations et des compositions traduisent des règles de configuration d'un DNN. Rappelons que MM-DNN est élaboré conformément à la définition informelle de la structure hiérarchique et du comportement d'un DNN avec toutes ses caractéristiques clés.

- Les Méta-classes incarnent les éléments clés des DNNs. Pour simplifier notre interprétation du méta-modèle, nous réduirons le terme « méta-classe » à « classe » dans la suite de cette section.
- Les spécialisations permettent de spécifier par une relation d'héritage, les différents sous-types possibles d'un élément donné en termes de sous-classes de la classe définie pour cet élément.
- Les associations sont des relations binaires entre des éléments du DNN et modélisés par des classes.
- Les compositions sont des associations fortes qui servent à définir les éléments composant physiquement un autre élément du DNN en spécifiant leur nombre par une cardinalité.

La classe DNN (colorée en violet) est le noyau de MM-DNN. Ses sous-classes FeedForward, RNN, CNN, Autoencodeur, GAN, DBN, et RBM modélisent les différents types de DNN qui bien évidemment partagent les caractéristiques clés de DNN.

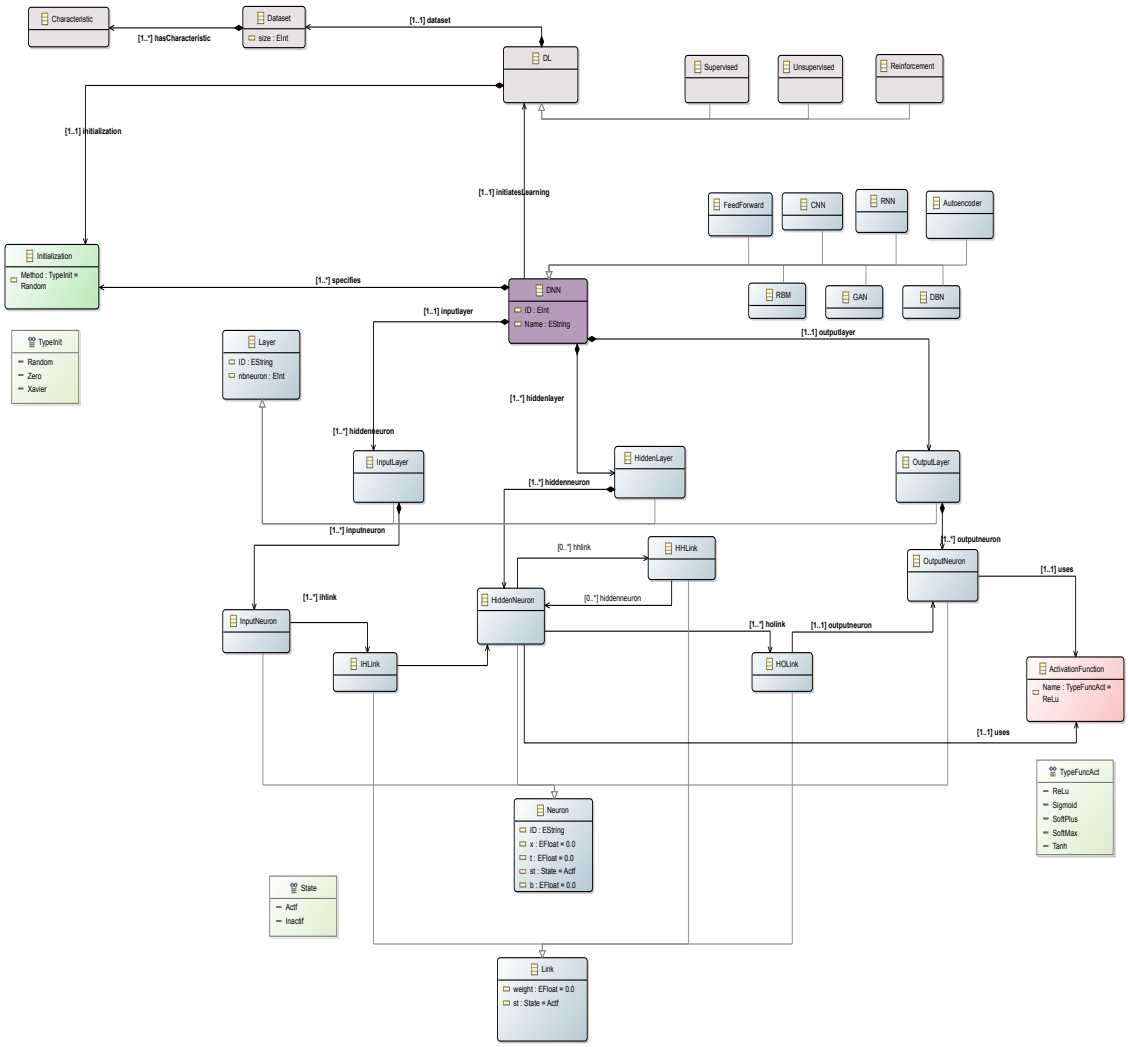


Figure 6: Méta-Modèle MM-DNN

3.2. Description du SM-Structurel

Le SM-Structurel est la partie du MM-DNN colorée en bleu. Nous y identifions les trois types d'éléments de base composant DNN, les catégories de chaque type d'élément et leurs connexions. Une instance de DNN (i.e. un modèle de DNN donné) qui peut être d'un type donné i.e. appartient à la sous-classe associée (FeedForward ou RNN, etc.), se compose d'une seule instance de InputLayer (i.e. une seule couche d'entrée), d'une ou plusieurs instance(s) de HiddenLayer(i.e. Une ou plusieurs couches cachées) et d'une seule instance de OutputLayer (i.e. une seule couche de sortie). Ces instances sont reliées de manière hiérarchique via (1) des instances des sous-classes InputNeuron, HiddenNeuron et outputNeuron de la classe Neuron et (2) des instances des sous-classes IHLink, HHLink et HOLink. de la classe Link selon une topologie de connexion décrite dans la sous-section 3.2.2.

3.2.1. Eléments de base

Les trois types d'éléments de base Couche, Neurone et Lien sont modélisés respectivement par les classes **Layer**, **Neuron** et **Link**.

La classe **Layer** a comme attribut **nblaye**r, le numéro de couche. Elle est spécialisée par trois sous-classes **InputLayer**, **HiddenLayer** et **OutputLayer** qui modélisent respectivement les

trois catégories couche d'entrée, couche cachée et couche de sortie de l'élément Couche.
La classe Neuron incarne les unités de calcul individuelles au sein d'une couche : la valeur d'activation actuelle (attribut **x**), le seuil sur lequel cette valeur est basée (attribut **t**) et l'état (attribut **st**) qui sera utilisé pour déterminer si le neurone a déjà été activé ou non. Ses trois sous-classes **InputNeuron**, **HiddenNeuron** et **OutputNeuron** modélisent respectivement les trois catégories (neurone de la couche d'entrée, neurone d'une couche cachée et neurone de la couche de sortie).
La classe **Link** représente les connexions entre les neurones de différentes couches. Chaque lien (instance de Link) stocke son poids numérique (attribut Weight) et contient une valeur (attribut st) pour indiquer si une valeur a déjà traversé le lien ou non. Toute instance d'une sous-classe de Layer (InputLayer ou HiddenLayer ou OutputLayer) se compose respectivement d'une ou de plusieurs instance(s) de la sous-classe InputNeuron ou HiddenNeuron ou OutputNeuron). Les trois sous-classes **IHLink**, **HHLink** et **HOLink** modélisent respectivement les liens entre neurone d'entrée et neurone de la première couche cachée, neurone d'une couche cachée et neurone de la couche cachée suivante dans la hiérarchie, neurone de la dernière couche cachée et neurone de la couche de sortie.

3.2.2. Topologie de connexion

La topologie de connexion est modélisée par les associations entre les différentes sous-classes de Neuron ainsi que leurs cardinalités respectives. Les règles qui la régissent selon la structure hiérarchique de DNN sont méticuleusement considérées dans MM-DNN et facilement déductibles à partir d'une interprétation correcte du SM-Structurel : Chaque instance de InputNeuron composant une instance de InputLayer est connectée à chaque instance de HiddenNeuron composant l'instance de HiddenLayer dont nblayer correpond à la première couche cachée, via une instance de IHLink. Chaque instance de HiddenNeuron composant une instance de HiddenLayer est connectée :
- soit à chaque instance de HiddenNeuron composant l'instance suivante de HiddenLayer via une instance de HHLink ;
- soit, s'il s'agit de la dernière couche cachée, à chaque instance de OutputNeuron composant l'instance de OutputLayer via une instance de HOLink.

3.3. Description du SM-Comportemental

Ce Sous-modèle du MM-DNN, comme nous le constatons clairement dans la figure 6, comporte trois parties, décrites ci-après, définissant le comportement dynamique des DNN. Cette sous-section le décrit en soulignant comment MM-DNN supporte la modélisation de l'initialisation de paramètres requis, du traitement des données avec leur propagation et l'apprentissage.

3.3.1. Modélisation de l'initialisation

La classe **Initialisation** (colorée en vert) considérée, grâce à une relation de composition, comme une composante de la classe DL mais indirectement reliée à la classe DNN pour supporter la procédure d'initialisation des différents paramètres nécessaires (voir attributs de la classe) allant de l'assignation des poids initiaux, à la structuration des différentes couches comprenant le nombre de couches, la valeur d'activation actuelle (attribut x), le seuil sur lequel cette valeur est basée (attribut t) et l'état (attribut st), les fonctions d'activation (telles que ReLU, sigmoid, tanh, etc.), et à la préparation optimale du réseau pour son fonctionnement ; en particulier son apprentissage.

3.3.2. Modélisation du traitement et de la propagation des données

Le fonctionnement d'un réseau de neurones s'articule autour de la propagation des données traitées de l'entrée jusqu'à la sortie. Etant à un haut niveau d'abstraction, cette tâche est

supportée dans MM-DNN par les différentes sous-classes de Neuron (**InputNeuron**, **HiddenNeuron** et **OutputNeuron**), les différentes relations d'association définies dans le sous-modèle structurel de MM-DNN et la classe **ActivationFunction** (colorée en rose). Les détails du fonctionnement qui en réalité est un processus, ne peuvent pas à notre sens, être à ce niveau. C'est plutôt à un niveau plus bas (niveau modèle) que l'on pourrait le faire avec un diagramme UML de comportement notamment le diagramme d'activité.

3.3.3. Modélisation de l'apprentissage

Pour la définition abstraite de l'apprentissage dans MM-DNN, nous avons proposé l'association de la classe **DL** (colorée en gris) à la classe DNN avec la cardinalité [1..1] pour les deux extrémités. Pour entraîner un modèle de DNN, plusieurs étapes sont nécessaires :

- L'initialisation du modèle est l'étape primordiale ; raison pour laquelle nous avons défini la classe Initialisation comme une composante de la classe DL. Sa coloration en vert s'explique par le fait qu'elle contribue non seulement à la modélisation de l'Initialisation du DNN, mais aussi à la modélisation de l'apprentissage.
- Ensuite, la préparation de données est indispensable pour lancer les processus de fonctionnement et d'apprentissage. Cette étape est supportée par la classe **Dataset** (colorée en gris) comme deuxième composante de la classe DL avec la cardinalité [1..1]. Comme les données peuvent avoir différents formats (textes, chiffres, images, vidéos, etc.) et des caractéristiques spécifiques, nous avons défini une autre classe appelée **Characteristic** (colorée en gris aussi) comme composante de la classe Dataset avec la cardinalité [1..*].

Comme pour le cas du processus de traitement et propagation des données, les détails de l'apprentissage ne peuvent être décrits qu'au niveau modèle avec UML de comportement tel que le diagramme d'activité.

3.4. Illustration

Faute de limitation du nombre de pages, notre illustration consiste à considérer le sous-modèle structurel de MM-DNN pour donner un exemple de son instanciation en proposant un modèle de DNN de type Feesdforward et un exemple concret d'instance de ce modèle. A travers ces deux exemples représentés dans la figure 7 :

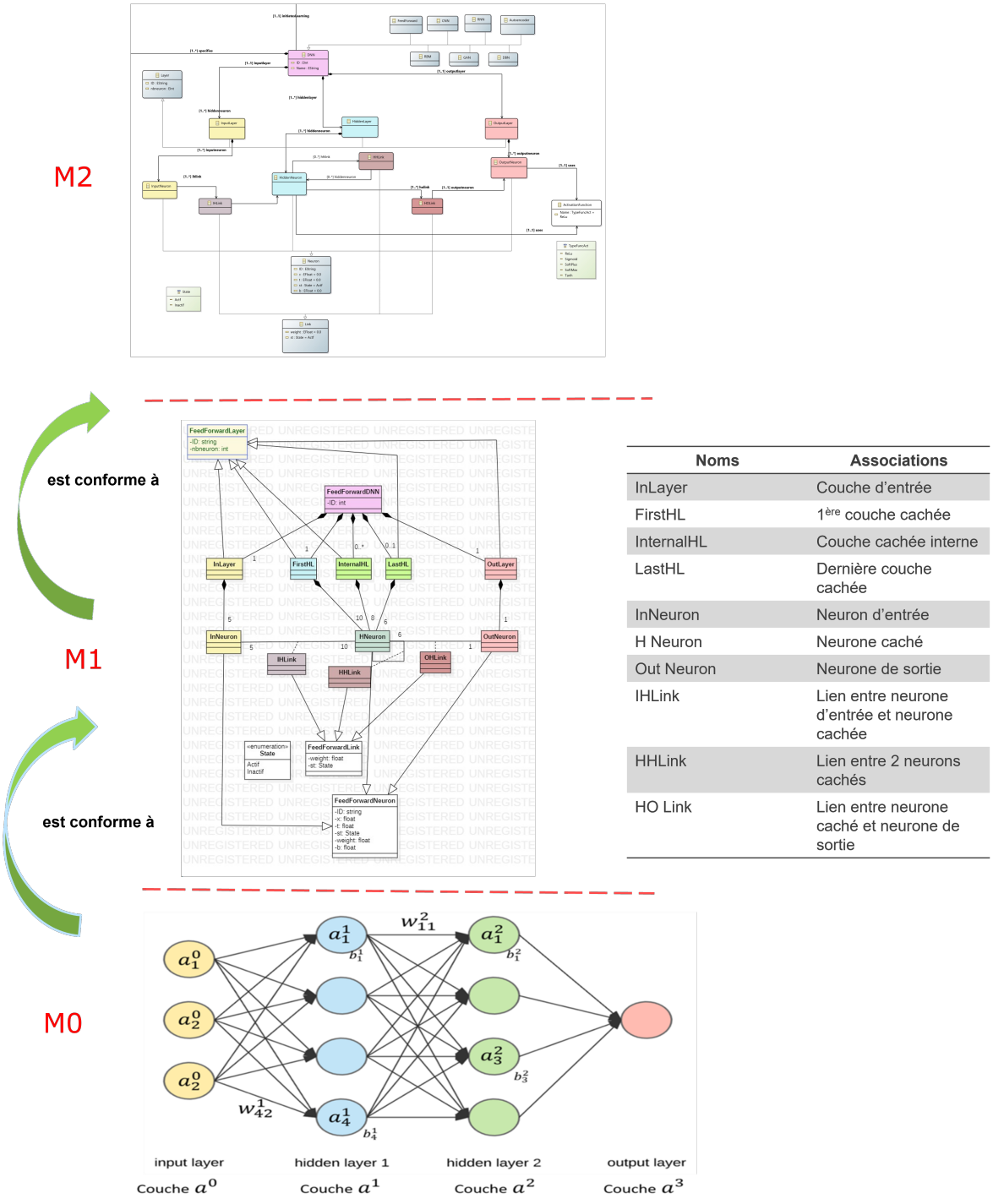
(1) Nous adaptons l'approche MDA en respectant sa structure en focalisant sur les niveaux de modélisation M2, M1 et M0. A ces niveaux, nous plaçons respectivement, le SM-Structurel de MM-DNN, le modèle-Feedforward-DNN sous forme d'un diagramme de classes UML et l'exemple concret de DNN sous une forme graphique, correspondant en fait à un diagramme d'objets, pour refléter la représentation (des ANNs) communément utilisée dans l.a littérature.

(2) Nous identifions explicitement la conformance entre les éléments des différents niveaux. Cette conformance est représentée par des flèches reliant les éléments d'un niveau ($i, 0 \leq i \leq 1$) avec les éléments du niveau $i + 1$ dont ils sont des instances.

En observant la Figure 7, nous voyons bien que :

- Le modèle-Feedforward-DNN au niveau M1 modélisant l'architecture des DNNs de type Feedforward telle que décrite dans la première ligne du tableau1 et il est bien conforme au SM-structurel (niveau M2).

- Le DNN au niveau M0, défini conformément au modèle-Feedforward-DNN, se compose d'une couche d'entrée composée de 3 neurones d'entrée, de deux couches cachées composée chacune de 4 neurones cachées et d'une couche de sortie composée de 1 neurone de sortie.



Noms	Associations
InLayer	Couche d'entrée
FirstHL	1ère couche cachée
InternalHL	Couche cachée interne
LastHL	Dernière couche cachée
InNeuron	Neuron d'entrée
H Neuron	Neurone caché
Out Neuron	Neurone de sortie
IHLink	Lien entre neurone d'entrée et neurone cachée
HHLink	Lien entre 2 neurons cachés
HO Link	Lien entre neurone caché et neurone de sortie

Figure 7: Illustration de l'instanciation du SM-Structurel de MM-DNN

4. Travaux Connexes

L'avènement des DNNs a conduit à une amélioration spectaculaire de l'efficacité et de l'efficience des algorithmes dans la résolution de problèmes complexes dans divers domaines d'application, allant des technologies de l'information à la biologie computationnelle. Toutefois, la conception et la mise en œuvre de ces réseaux demeurent complexes et exigent des connaissances spécial-

isées. Simplifier cette tâche et maîtriser particulièrement sa complexité tout en garantissant l'amélioration des performances est donc d'une grande nécessité. Une solution préconisée est l'abstraction en se basant sur la notion de modèles (formels) faciles à traiter automatiquement et l'IDM en est le meilleur cadre. Dans la littérature, il n'y a pas beaucoup de travaux dans ce contexte. Dans cette section, nous identifions et discutons uniquement deux travaux majeurs, ceux de Thomas Hartmann et al. [2] et de Lechevalier et al. [21], par rapport auxquels nous essayons de situer notre contribution. Les auteurs du premier article, ont conçu un méta-modèle pour l'apprentissage automatique qui sert de solution à un problème de recherche, approximant une fonction de mapping sous-jacente inconnue d'entrées à des sorties. Leur travail se concentre sur l'objectif, l'algorithme d'apprentissage, l'optimiseur et les métadonnées de l'ensemble de données. Leur approche abstraite permet une application à divers algorithmes d'apprentissage automatique, tels que les réseaux de neurones, le gradient boosting et la régression linéaire. Toutefois, leur approche demeure générale et ne traite pas en détail des spécificités de DNN. De plus, ils n'abordent pas explicitement certains éléments clés des DNNs, tels que la topologie, l'apprentissage et le fonctionnement, qui sont essentiels pour comprendre et optimiser ces systèmes. Dans le second article, les auteurs ont proposé un méta-modèle de réseau de neurones pour une application de fabrication additive où la représentation et la compréhension de tel réseau sont cruciales pour optimiser les processus de production. Leur travail a introduit le concept d'un "modèle méta-neuronal" (Neural Network Meta-Model, NNMM), une méthode pour représenter divers types de réseaux de neurones dans l'Environnement de Modélisation Générique (Generic Modeling Environment, GME). Bien que leur étude offre une représentation valable pour les réseaux de neurones feedforward (FNN), ils mentionnent l'inclusion des réseaux de neurones récurrents (RNNs) comme une étape future pour la représentation de tous les types de réseaux de neurones. Cependant, leur approche, bien qu'utile pour comprendre la représentation des différents types de réseaux de neurones, ne se concentre pas spécifiquement sur les DNNs qui sont l'objectif central de notre travail. De plus, leur méta-modèle, bien qu'il mette l'accent sur la facilité d'utilisation et de compréhension, ne traite pas en profondeur de la complexité inhérente aux DNNs. Nous déduisons donc que les propos des deux travaux ainsi présentés et argumentés partagent l'idée d'adopter la méta-modélisation pour simplifier l'utilisation des réseaux de neurones en général mais sans couvrir explicitement des éléments et des caractéristiques spécifiques aux DNNs, Dans un contexte similaire, notre travail propose aussi un méta-modèle mais spécifiquement adapté aux DNNs, en tenant compte des spécificités à la fois de leur topologie, de leur apprentissage et de leur fonctionnement. Le tableau 2 récapitule cette comparaison et montre clairement la particularité de notre propos présenté dans ce papier. Nous espérons que notre travail contribuera à optimiser l'exploitation du potentiel immense des DNNs dans divers domaines d'application.

5. Conclusion et Perspectives

Notre étude du DL, et plus spécifiquement des DNNs, nous a permis de recenser certains défis associés généralement à leur conception et à leur mise en œuvre ou encore à leur exploitation, et de confirmer que le succès ou l'échec de cette technologie en plein essor dépend principalement des méthodes appliquées à la configuration architecturale des DNNs. Nous avons donc déduit que les DNNs nécessitent un développement systématique, basé sur l'abstraction, à un haut niveau, dans une première étape ; puis, l'utilisation de méthodes formelles tout en supportant toutes leurs caractéristiques structurelles et comportementales. Cette exigence est en fait une solution prometteuse pour maîtriser leur complexité en croissance continue, et faciliter leur conception. Le résultat de cette abstraction devra en conséquence, garantir une mise en œuvre de hautes performances et une exploitation aisée, non fastidieuse et à moindre coût. A la lumière de cette déduction, nous avons alors établi l'objectif de notre contribution présentée dans ce papier, dans le même contexte. C'est la réalisation de la première étape d'une approche de modélisation

Aspect	[2]	[21]	Notre Méta-modèle
Focus sur les DNN	✗	✗	✓
Modélisation de la topologie	✗	✗	✓
Fonctionnement et Apprentissage	✗	✗	✓
Algorithmes d'apprentissage automatique	✓	✗	✓
Application spécifique	✗	—	—
Objectif majeur	Concevoir un méta-modèle pour l'apprentissage automatique afin de simplifier la solution à des problèmes de recherche, avec un focus sur l'optimisation des algorithmes d'apprentissage et des métadonnées de l'ensemble de données.	Créer un méta-modèle de réseaux de neurones pour une application spécifique dans le domaine de la fabrication, avec un focus sur les réseaux feedforward.	Développer une nouvelle approche de modélisation des DNNs afin de prendre en charge l'amélioration de leurs caractéristiques.

Tableau 2: Aperçu des Méta-modèles : Notre Approche vs. Travaux Connexes

formelle des DNNs pour l'amélioration de leurs caractéristiques en utilisant la technique de méta-modélisation de l'IDM selon son approche MDA. Le fruit de notre travail est la proposition du Méta-Modèle MM-DNN comme nouvelle définition (formelle) des DNNs. Comparé à d'autres méta-modèles de réseaux de neurones proposés dans des travaux antérieurs et dont la perspective est soit trop générale, soit trop limitée, MM-DNN se distingue par sa spécificité exclusive pour les DNNs couvrant à la fois structure et comportement (Fonctionnement + apprentissage). Outre cette définition abstraite concise et complète des DNNs, MM-DNN à l'avantage d'être extensible et facile à devenir automatiquement du code exécutable ou à transcrire dans d'autres langages plus formels, tel que Maude, dotés d'outils performants d'analyse et de vérification. Aussi, MM-DNN se concentre sur les particularités des DNNs tout en maintenant la flexibilité nécessaire pour s'adapter à une multitude d'applications. En outre, notre attention à la complexité inhérente des DNNs, permet une compréhension plus approfondie et nuancée, essentielle pour les chercheurs et les praticiens du domaine. Cependant, comme tout travail de recherche, le nôtre n'est pas exempt de limitations. Bien que notre méta-modèle ait été conçu avec une attention particulière à la pratique et aux défis courants des DNNs, il n'a pas encore été mis en œuvre dans des applications concrètes. Cela ouvre la porte à de futures recherches et améliorations. Dans les perspectives futures, nous envisageons de le tester dans des scénarios d'application réels pour évaluer sa robustesse et son efficacité puis l'utiliser comme point de départ pour poursuivre la réalisation de notre approche de modélisation formelle et amélioration des DNNs Dans un autre volet de recherche, MM-DNN pourra facilement être considéré comme une syntaxe abstraite d'un langage de modélisation spécifique aux DNNs i.e. un DMSL. De plus, à mesure que le domaine des DNNs évolue, notre MM-DNN pourrait nécessiter des mises à jour pour intégrer les dernières avancées et techniques. En conclusion, notre travail représente une étape importante dans la quête d'une meilleure compréhension, conception et mise en oeuvre des DNNs. Nous espérons que notre méta-modèle servira de base solide pour de futures recherches et applications dans ce domaine passionnant.

Remerciements

Ce travail a été partiellement soutenu par le projet LABEX-TA MeFoGL : "Méthodes Formelles pour le Génie Logiciel".

Références

[1] P. Chaber, M. Ławryńczuk, Pruning of recurrent neural models: an optimal brain damage approach, Nonlinear Dynamics 92 (2018) 763–780.

[2] T. Hartmann, A. Moawad, C. Schockaert, F. Fouquet, Y. Le Traon, Meta-modelling meta-learning, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), 2019, pp. 300–305. doi:10.1109/MODELS.2019.00014.

[3] S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, X. Yue, Formal specification for deep neural networks, in: Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings 16, Springer, 2018, pp. 20–34.

[4] D. C. Schmidt, et al., Model-driven engineering, Computer-IEEE Computer Society- 39 (2006) 25.

[5] M. Clavel, S. Eker, P. Lincoln, J. Meseguer, Principles of maude, Electronic Notes in Theoretical Computer Science 4 (1996) 65–89.

[6] M. Seidl, M. Scholz, C. Huemer, G. Kappel, M. Seidl, M. Scholz, C. Huemer, G. Kappel, The activity diagram, UML@ Classroom: An Introduction to Object-Oriented Modeling (2015) 141–166.

[7] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[8] E. Simonnet, Réseaux de neurones profonds appliqués à la compréhension de la parole, Ph.D. thesis, Le Mans Université, 2019.

[9] M. Minsky, S. Papert, 1990. perceptrons: An introduction to computational geometry, 1969.

[10] P. H. Truong, Optimisation des performances de la machine synchrone à réluctance variable: approches par la conception et par la commande, Ph.D. thesis, Mulhouse, 2016.

[11] H. M. SEKKIL, M. MEBROUKI, Etude comparative entre les différentes architectures des réseaux de neurones convolutifs (CNNs) pour la détection de la fatigue du conducteur., Ph.D. thesis, Directeur: Melle. Imane NEDAJR/Co-Directeur: M. MEGNAFI Hichem, 2021.

[12] OpenClassrooms, Explorez les réseaux de neurones en couches, 2021. URL: https://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning/5814616-explorez-les-reseaux-de-neurones-en-couches.

[13] S. Baamara, Classification des arythmies cardiaquesen utilisantles réseaux de neurones profonds, Ph.D. thesis, 2020.

[14] J. Bézivin, On the unification power of models, Softw Syst Model 4 (2005) 171–188. URL: http://link.springer.com/10.1007/s10270-005-0079-0. doi:10.1007/s10270-005-0079-0.

[15] J. Bézivin, M. Blay, M. Bouzhegoub, J. Estublier, J.-M. Favre, S. Gérard, J. M. Jézéquel, Rapport de synthèse de l'as cnrs sur le mda (model driven architecture), CNRS, novembre 21 (2004).

[16] L. Gaouar, Généralités sur l'approche MDA, Developpez.com (2013). URL: https://laine.developpez.com/tutoriels/alm/mda/generalites-approche-mda/.

[17] MDA, Model Driven Architecture (MDA) – Projet de fin d'etudes, Rapport-gratuit.com, 2019. URL: https://www.rapport-gratuit.com/model-driven-architecture-mda/.

[18] B. Combemale, Ingénierie dirigée par les modèles (idm)–état de l'art (2008).

[19] ObjectManagementGroup, MetaObjectFacility(MOF)2.0, Technicalreport, 2009. URL: http://www.omg.org/spec/QVT/1.1/Beta2/.
[20] F. Budinsky, Eclipse modeling framework: a developer's guide, Addison-Wesley Professional, 2004.
[21] D. Lechevalier, S. Hudak, R. Ak, Y. T. Lee, S. Foufou, A neural network meta-model and its application for manufacturing, in: 2015 IEEE International Conference on Big Data (Big Data), IEEE, 2015, pp. 1428–1435.