REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Informatique

Par :

BOUCHELIL Narimane
MOUKLI Sara

Sur le thème

L'extraction des règles de prédiction dans le domaine de l'intelligence ambiante

---

Soutenu publiquement le 11 / 07 / 2023 à Tiaret devant le jury composé de :

Mr ALEM Abdelkader	MAA Université Tiaret	Président
Mr MOKHTARI Ahmed	MAA Université Tiaret	Encadreur
Mr ABID Khaled	MAA Université Tiaret	Examinateur

2022-2023

Dédicace

Je dédie humblement ce manuscrit à :
A celui qui m'a toujours ouvert ses bras et soutenu dans tout ce que j'ai entrepris ; celui qui a su être bon, gentil et compréhensif avec moi ; celui qui a toujours su trouver les mots pour me redonner la force de continuer et d'aller au bout de cette aventure, **mon très cher père**.
A celle qui s'est toujours dévouée et sacrifiée pour moi ; celle qui m'a aidée du mieux qu'elle pouvait pour réussir ; celle qui m'a accompagnée dans mon cœur tout au long, ma très chère mère, que Dieu ait pitié de toi, **ma mère**.
A celles qui m'ont toujours aidé, écouté, soutenu et encouragé tout au long de mon parcours ; celles qui ont toujours été présentes pour moi, mes très chères sœurs et mon très cher frère Ilyes.
A mes grands-mères et mes très chers oncles, tantes surtout mon oncle Djilali et ma tante Oum Houssine.
A toutes ma famille, mes cousins, mes cousines et à la famille Braiki.
A mes petits anges Jana, Ayoub, Younes, Saja, Line, Lina, Layen, Aya et Souad.
A celle qui m'a toléré pour continuer ce parcours ensemble, ma binôme Sara et toute sa famille.
A celles qui m'ont supporté mes chères amies Khadidja, Hakima, Intissar, Rihem.
A tous ceux qui je n'ai pas cité bien sûr ne croyait pas que je vous ai oublié, je vous porte toujours dans mon cœur.

Bouchelil Narimane
Merci.

Dédicace

À mes chers parents

En ce jour de réalisation et de réussite, je tiens à vous adresser cette dédicace de mémoire empreinte d'amour et de gratitude. Votre soutien indéfectible, votre amour inconditionnel et vos sacrifices ont été les piliers de mon parcours académique.

À toi, **papa** qui m'as toujours encouragé à poursuivre mes rêves, à croire en moi et à viser l'excellence. Ta force, ta sagesse et ta bienveillance ont été une source d'inspiration inépuisable. Merci de m'avoir transmis les valeurs de persévérance et de détermination qui ont guidé chacun de mes pas.

À toi, **maman** qui as toujours été mon roc, ma confidente et ma première enseignante. Ton amour inconditionnel, ta patience et ton soutien constant ont été des moteurs essentiels dans ma quête de connaissances. Merci de m'avoir inculqué le goût de l'apprentissage et la volonté d'aller au-delà des limites.

A mes petits anges : Tadj et Siradj ;

Mes chères sœurs : Naima, Fatima et Djamila ;

Mes chers frères : ABD Elkader et Amine ;

Mes meilleurs amis et mes collègues : Sara, Rihem, Intissar, Khadidja ;

À ma chère binôme Narimane qui m'a toléré pour continuer ce parcours ensemble et toute sa famille ;

L.Zaki ; Pour tout et chaque instant tu me soutiens et m'encourage ;

A tous ceux que je n'ai pas cités, bien sûr, ne croyez pas que je vous ai oubliés, je vous porte toujours dans mon cœur.

Moukli Sara

Merci.

Remerciements

Tout d'abord, nous tenons à remercier Dieu de nous avoir accordé toute la détermination, la volonté et la force pour que nous puissions réaliser ce modeste travail.

Nous tenons à exprimer notre profonde gratitude envers **Mokhtari Ahmed** pour son soutien infaillible tout au long de ce parcours. Ses encouragements, ses conseils éclairés et sa confiance en nos capacités ont été déterminants pour notre réussite.

Nos remerciements les plus sincères vont à **Alem Abdelkader** et **Abid Khaled**, membres de notre jury de mémoire, pour leurs expertises, leurs précieux commentaires et leurs évaluations attentives.

Nous souhaitons également exprimer notre reconnaissance envers tous les professeurs et enseignants qui nous ont transmis leurs connaissances et leur passion pour la discipline. Nos remerciements les plus sincères vont à **phd.étudiant Lebani Ali Zakaria** et **phd.étudiant Djenane Mouloud** pour leur aide et leurs conseils afin de nous guider chaque fois que nous avons besoin d'eux.

Enfin, un grand merci à notre famille, nos amis et tous ceux qui nous ont soutenus de près ou de loin. Votre amour, vos encouragements et votre présence ont été une source de motivation constante.

Ce mémoire est le fruit d'un travail collectif et nous sommes reconnaissants envers chacune des personnes qui y ont contribué d'une manière ou d'une autre.

Nos plus sincères remerciements.

Résumé

Ce projet de thèse de maîtrise porte sur l'utilisation de techniques d'apprentissage profond et d'exploration de données, en particulier LSTM, GRU et MLP, pour extraire les règles de prédiction des données fournies par les objets connectés dans le domaine de l'intelligence ambiante. En utilisant ces techniques, il devient possible de prédire le comportement des utilisateurs et de permettre une automatisation intelligente, ouvrant ainsi de nouvelles perspectives pour des applications personnalisées adaptées aux besoins des utilisateurs. La recherche utilise l'ensemble de données "Aruba" du projet "CASAS Smart Home" et les résultats obtenus pour LSTM, GRU et MLP sont de 0,84, 0,84 et 0,41, respectivement. Ces résultats devraient avoir un impact important sur le développement de l'intelligence ambiante, faire progresser le domaine et ouvrir la voie à une expérience utilisateur améliorée.

Mots clés : Intelligence ambiante, prédiction de comportement, apprentissage automatique, apprentissage profond.

Abstract

This master's thesis project focuses on utilizing deep learning and data mining techniques, specifically LSTM, GRU, and MLP, to extract prediction rules from data provided by connected objects in the field of ambient intelligence. By leveraging these techniques, it becomes possible to predict user behavior and enable intelligent automation, thereby opening new perspectives for customized applications tailored to user needs. The research utilizes the "Aruba" dataset from the "CASAS Smart Home" project, and the obtained results for LSTM, GRU, and MLP are 0.84, 0.84, and 0.41, respectively. These findings are expected to have a significant impact on the development of ambient intelligence, advancing the field and paving the way for enhanced user experiences.

Key words : Ambient Intelligence, behavoir prediction, Machine Learning, Deep Learning.

ملخص

يركز مشروع أطروحة الماجستير هذا على استخدام تقنيات التعلم العميق واستخراج البيانات، وتحديدا LSTM و GRU و MLP، لاستخراج قواعد التنبؤ من البيانات المقدمة من الكائنات المتصلة في مجال الذكاء المحيط. من خلال الاستفادة من هذه التقنيات، يصبح من الممكن التنبؤ بسلوك المستخدم وتمكين الأتمتة الذكية، وبالتالي فتح وجهات نظر جديدة للتطبيقات المخصصة المصممة خصيصًا لاحتياجات المستخدم. يستخدم البحث مجموعة بيانات «Aruba» من مشروع CASAS Smart» «Home، والنتائج التي تم الحصول عليها لـ LSTM و GRU و MLP هي 0.84 و 0.84 و 0.41 على التوالي. من المتوقع أن يكون لهذه النتائج تأثير كبير على تطوير الذكاء المحيط، والنهوض بالميدان وتمهيد الطريق لتجارب المستخدمين المعززة.

Table des matières

Table des figures

Liste des tableaux

Listes des Acronymes

AmI : Ambient Intelligence

ANN : Artificiel Neural Network

CNN : Convolutional Neural Network

CSV : Comma-Separated Values

DL : Deep Learning

GPU : Graphics Processing Unit

GRU : Gated Recurrent Unit

IA : Intelligence Artificielle

IBM : International Business Machines

ICT : Information and Communication Technology

LSTM : Long Short Term Memory

ML : Machine Learning

MLP : Multi layer perceptron

ReLU : Rectified Linear Unit

RNN : Recurrent Neural Network

TDE : Trusted Data Environment

TPU : Tensor Processing Unit

TXT : Text File

Introduction générale

Initialement appelé « Ubiquitous Computing » [1], il s'agissait de mettre en évidence l'intégration voir la fusion de l'informatique au cœur de nos activités quotidiennes. Puis, sont apparus les termes de « pervasive computing » dont un des objectifs étaient de mettre en avant les aspects techniques liés à cette évolution technologique. Le terme « Ambiant Intelligence » est l'un des enjeux de stimuler la créativité pour la conception de nouveaux services, donc ouvrant la voie à de nombreuses applications dans divers domaines tels que la domotique, la santé, les transports et bien d'autres.

Selon [2], les auteurs définissent l'intelligence ambiante de la manière suivante : « *Milieu ayant la faculté de percevoir, de raisonner, d'agir et d'interagir afin de fournir des services améliorant la qualité de vie des êtres vivants et notamment des personnes* ». Dans ce contexte, la prédiction du comportement des utilisateurs devient essentielle pour offrir des automatisations intelligentes et adaptées à leurs besoins.

Notre problématique est : comment prédire les comportements des utilisateurs dans l'intelligence ambiante ? En se basant sur les données fournies par les objets connectés, cette question soulève des enjeux importants, car une prédiction précise du comportement permettrait d'anticiper les besoins des utilisateurs, d'optimiser les ressources disponibles et de proposer des services personnalisés.

L'objectif de ce travail de recherche est de développer une méthodologie d'extraction de règles de prédiction à partir des données collectées par les objets connectés. En analysant ces données, il est possible de déceler des schémas et des corrélations qui permettront de formuler des règles prédictives. Ces règles pourront ensuite être utilisées pour automatiser certaines actions ou prises de décision dans l'environnement d'intelligence ambiante.

Pour atteindre cet objectif, différentes techniques d'apprentissage profond seront explorées. Des algorithmes de prédiction seront mis en place et évalués en utilisant des jeux de données réels provenant d'environnements d'intelligence ambiante. L'accent sera mis sur la précision, la robustesse et l'efficacité des règles de prédiction extraites, afin d'assurer leur fiabilité et leur applicabilité dans des situations réelles.

Ce projet de fin d'études contribuera à l'avancement de l'intelligence ambiante en proposant des modèles pour la prédiction du comportement des utilisateurs. Les résultats obtenus permettront de mieux comprendre les interactions entre les individus et leur environnement connecté, ouvrant ainsi la voie à de nouvelles applications et à une amélioration significative de l'expérience utilisateur.

Quelques travaux liés :
Les auteurs de [3] proposent un système d'extraction de règles de base sur une séquence d'atomes, où un atome en prédit un autre. Ces atomes représentent de simples variations des données boursières. Il peut également rechercher des règles plus complexes, où la condition est une séquence. Ce système permet donc de creuser des règles de prédiction sur une série temporelle. Cependant, il recherche des règles de prédiction entièrement ordonnées, plutôt que des règles partiellement ordonnées. De plus, la partie prédiction des règles est limitée à un seul atome, une limitation que nous voulons éviter dans notre système AmI.

Par contre dans [4] on peut considérer une amélioration par rapport à [3], car ce système recherche des règles où la prédiction n'est pas limitée à un seul atome. Toutefois, étant donné qu'il recherche des règles entièrement ordonnées, ce système ne peut pas être appliqué dans notre cas, car nos règles ne sont pas ordonnées mais plutôt aléatoires.

En revanche les auteurs dans [5] introduisent un concept de support pour une série temporelle, via une fenêtre glissante de durée déterminée. Le support d'un atome, d'un ensemble d'atomes ou d'une règle est le nombre de fenêtres dans lesquelles cet atome, cet ensemble ou cette règle apparaît. Cet algorithme trouve des règles partiellement ordonnées, en trouvant d'abord des séquences d'atomes fréquentes, puis en divisant ces séquences en deux sous-séquences de toutes les manières possibles pour déterminer si elles forment ou non une règle de prédiction.

D'autres algorithmes utilisent la notion de [5], notamment [6] qui trouve des règles dont la prédiction est composée d'un seul atome.

Dans [7] l'auteur traite la même problématique qu'est la prédiction de comportement, mais on utilise les techniques d'apprentissage automatique.

Ce mémoire est composé de trois chapitres. Le premier chapitre propose une définition de l'intelligence ambiante et introduit les différentes fonctionnalités, applications et facteurs en jeu de ce domaine. Une approche axée sur l'IA et l'apprentissage est privilégiée pour présenter en détail ce domaine.

Le deuxième chapitre rappelle les concepts clés de l'intelligence artificielle (IA) et de l'apprentissage automatique (ML). Nous explorerons les différents types d'apprentissage automatique et nous plongerons plus profond dans les réseaux de neurones. Nous étudierons l'inspiration derrière les réseaux de neurones artificiels et comprendrons le fonctionnement d'un réseau de neurones. Enfin, nous examinerons différents modèles de Deep Learning tels que le MLP, le CNN, le RNN, le LSTM et le GRU.

Le troisième chapitre de notre travail aborde nos contributions en mettant en place trois modèles d'extraction des règles de prédiction et en les comparant entre eux. Nous structurons ce chapitre sous l'intitulé " Conception et implémentation".

Chapitre 1

Intelligence ambiante

1.1 Introduction

Dans ce chapitre nous commençons par présenter le domaine de l'intelligence ambiante, ses différentes fonctionnalités, ses applications, ses facteurs en jeu. Nous nous concentrons sur une approche orientée avant tout vers l'IA et l'apprentissage.

1.2 Historique

Le terme « ubiquitous computing » (informatique ubiquitaire) a été introduit par Mark Weiser, chercheur à Xerox PARC3, pour désigner sa vision de l'ordinateur du 21ième siècle. Weiser constate que les technologies ancrées dans nos activités quotidiennes sont celles qui savent s'y fondre, jusqu'à disparaître. Il illustre son propos avec l'exemple de l'écriture : omniprésente dans nos sociétés modernes, chacun l'utilise quotidiennement sans même y prêter attention. Comparé à l'écriture, l'ordinateur actuel, bien que très répandu, est loin d'être aussi intégré dans le tissu de nos activités [8].

Selon Weiser, l'ordinateur d'aujourd'hui ne constitue qu'une étape vers l'informatique ubiquitaire dont l'ancrage dans le quotidien serait tel qu'il rendrait continuellement des services indispensables sans que personne ne remarque sa présence [8].

« Pervasive Computing » (informatique diffuse), apparu au milieu des années 90, est une initiative essentiellement industrielle marquée par l'implication d'IBM. L'accent est plutôt mis sur les aspects techniques avec notamment, le développement des supports matériels et logiciels nécessaires à la concrétisation de la vision de Weiser. Les termes « Ambient Intelligence » et « Disappearing Computer » apparaissent avec le 5ème PRCD de la communauté européenne. Cette période est fortement marquée par l'initiative de Philips Research qui lance alors le projet « Vision of the Future » que concrétise un laboratoire d'expérimentations : le Philips Homelab. Il s'agit de stimuler la créativité, d'explorer de nouvelles opportunités par la convergence de technologies, d'identifier la signification socioculturelle de ces inventions ; en somme, rendre les concepts tangibles, utiles et accessibles à tous [8].

1.3 Définition

L'expression intelligence ambiante (en Anglais Ambient Intelligent (AmI)) décrit les contextes où les composants technologiques interagissent pour offrir aux occupants divers avantages découlant de cette interaction. Cette idée est généralement liée à une attente de l'endroit où nous pourrions vivre dans un avenir plus ou moins proche. AmI est un paradigme qui résulte naturellement de l'interaction de deux éléments, à savoir l'avancement de l'intelligence artificielle (IA) et la réalisation progressive de l'informatique omniprésente comme décrit par [1]. Bien que l'infor-

matique omniprésente ne puisse pas encore être considérée comme pleinement réalisée, les progrès technologiques (capteurs, actionneurs, microprocesseurs, communications sans fil) et la baisse du prix de l'électronique ont permis de multiplier et de réduire la taille des dispositifs de communication (Figure 1.1). La diffusion du matériel informatique s'accompagne d'une recherche continue d'interfaces homme-machine plus conviviales qui place l'utilisateur au centre du système.

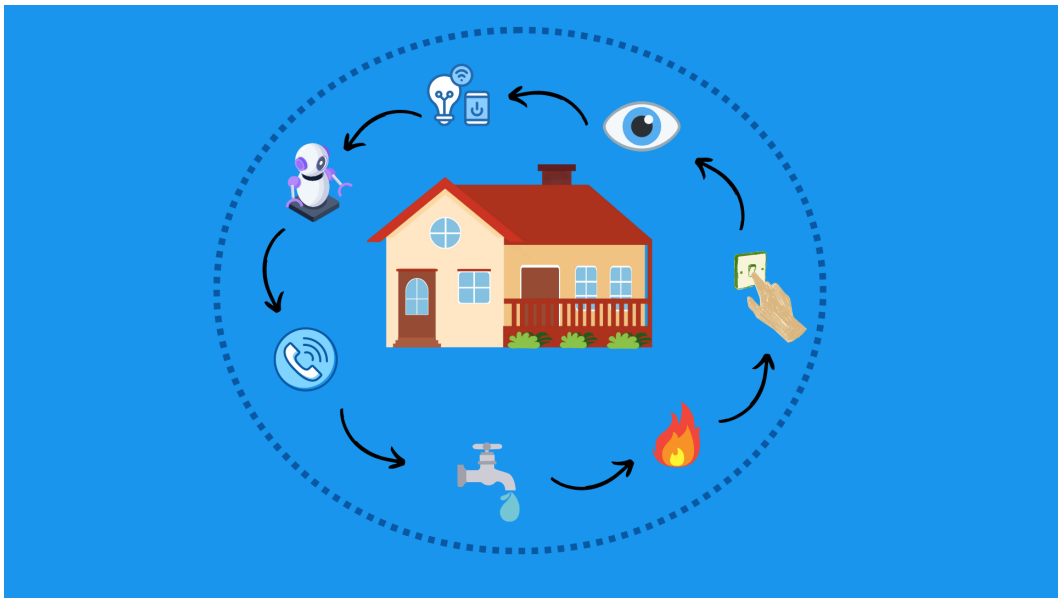

FIGURE 1.1 – Un système d'intelligence ambiante est composé d'un ensemble d'appareils communicants disséminés dans l'environnement

AmI est une idée très nouvelle et abstraite qui est continuellement développée. De ce fait, il est parfois utilisé de manière nuancée et est parfois confondu avec des termes comme « domotique », « informatique ubiquitaire », « environnements pervasifs », « Internet des objets », etc. L'apport de l'intelligence artificielle pour coordonner les éléments de l'environnement pour qu'il devienne adaptable et proactif, dans le but d'améliorer l'expérience de l'occupant, sépare AmI de ces conceptions antérieures. C'est pourquoi les expressions "environnements intelligents", fréquemment utilisées, correspondent davantage à l'idée d'intelligence ambiante. [9] fournit une étude approfondie de cette idée et de ses définitions potentielles tout en proposant une architecture générique pour représenter un système AmI. Afin d'améliorer l'expérience de ses habitants, l'intelligence ambiante est décrite comme l'application de l'intelligence artificielle à un environnement qui incarne le concept d'informatique ubiquitaire. Les connexions entre l'AmI et ses aires associées sont illustrées à la figure 1.2.

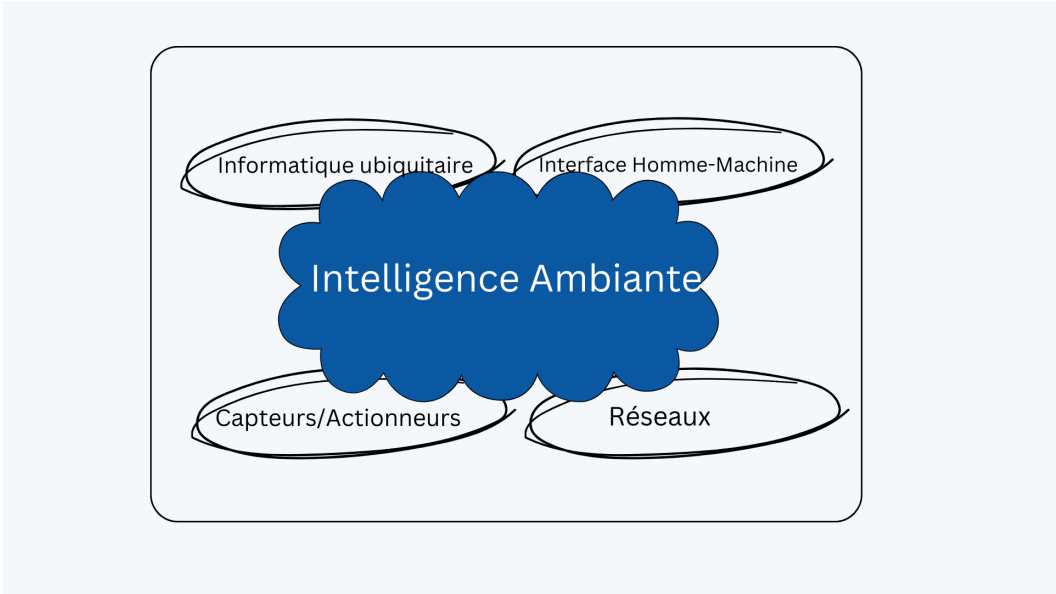

FIGURE 1.2 – Les domaines de l'intelligence ambiante

1.4 Les différentes fonctionnalités de l'AmI

Un système d'intelligence ambiante a des fonctionnalités suivantes qui sont illustrées dans la figure 1.3 [10] :

1. Détection

   Capteurs.

   Ambiance ou corps.

2. Raisonnement

   Nécessaire pour apporter réactivité et adaptabilité.

   Interpréter et reconnaître le contexte et l'activité.

   Suivi de la mobilité.

   Reconnaissance d'activité, prédiction d'activité.

   La prise de décision.

3. Agissant

   Systèmes domotiques (lumières, portes, fenêtres, température...).

   Interface utilisateur ou appareils portables (notification, information, alerte...).

   Robots.

4. Interagir avec les utilisateurs

   Interfaces utilisateurs traditionnelles : web, mobiles.

   Luminaires de la maison.

   Interfaces utilisateur naturelles :

•Parole, geste, traçage des mouvements corporels, émotions, expressions faciales, attention ...

•L'interaction contourne l'équipement ICT ("ordinateur en voie de disparition").

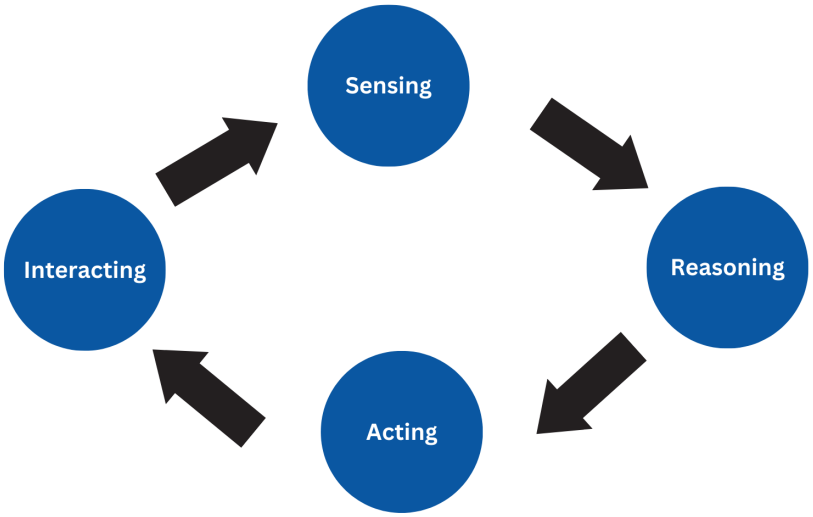

FIGURE 1.3 – Les différentes fonctionnalités de l'AmI

1.5 Les applications de l'AmI

Parmi les domaines d'applications de l'AmI, on cite les plus importants [11] :

1.5.1 Éducation

La relation entre les enseignants et les élèves a considérablement changé en raison du développement de la technologie. Cet engagement ne se produit plus en face à face et peut plutôt s'appuyer sur des outils de communication, qui ne remplacent jamais efficacement nos processus de communication complexes.

Les conversations en face à face entre enseignants et élèves ont pour la plupart été remplacées par des forums en ligne et des débats textuels, ce qui aggrave considérablement la communication.

1.5.2 Robotique

La vision de l'intelligence artificielle a toujours inclus les robots. Chaque personne aurait sous peu un compagnon robotique, très probablement sous forme humanoïde, qui servirait d'assistant personnel aux multiples facettes et prendrait soin de tous nos besoins. La direction de la recherche

et du développement technologique semble changer.

1.5.3 Santé

Actuellement, l'un des secteurs avec la technologie la plus avancée dans le domaine de la santé. Par conséquent, il n'est pas surprenant que cette catégorie comprenne un large éventail d'initiatives. La prestation de services de santé à domicile et le soutien des soins de santé dans les hôpitaux sont deux tendances importantes à prendre en compte.

1.5.4 Transport

Le transport est une autre industrie où l'adoption de l'intelligence ambiante peut présenter divers avantages étant donné que nous passons une grande partie de notre vie à faire des allers retours.

Les bus et autres véhicules peuvent être équipés d'une technologie qui peut fournir des informations cruciales sur le fonctionnement du système et identifier les ajustements potentiels en utilisant plus efficacement le système, ce qui peut améliorer l'expérience des personnes.

1.5.5 Lieu de travail

Les accidents du travail, y compris sur les chantiers de construction, pourraient être considérablement réduits grâce à l'intelligence ambiante. Le non-respect des règles de sécurité est à l'origine de nombreux accidents.

Les moniteurs humains ne peuvent pas être en service en permanence. AmI, d'autre part, a toujours une vue d'ensemble de l'environnement et des personnes qui s'y trouvent. Si une règle de sécurité est enfreinte, elle peut rapidement avertir la direction ou l'employé.

1.6 Facteurs en jeu

L'évolution technologique permet de fabriquer des ordinateurs et composants informatiques minuscules, des capteurs et senseurs qui pourront être omniprésents et communiquer entre eux et avec différents réseaux. Elle ouvre presque à tous les objets de la vie courante, la capacité à déclencher un échange spontané d'informations, sans interaction avec leur utilisateur [12] .

Les concepts informatiques évoluent vers des systèmes complexes en réseaux, fondamentalement différents des systèmes informatiques du 20ième siècle et de la notion d'ordinateur (disque dur, mémoire vive, interface par clavier, écran et souris, etc.) qui lui était couramment rattachée [12] .

Ces nouveaux concepts pourraient induire de profonds changements dans le monde social, culturel et de l'entreprise et s'introduire dans la vie quotidienne. De nombreux prospectivistes pensent qu'une évolution inéluctable des modes de vie est entamée, ainsi qu'une évolution capitale des activités et métiers informatiques [12] .

1.7 En intelligence ambiante

L'environnement physique est crucial pour l'intelligence ambiante. Dans la plupart des cas, les données des capteurs sont utilisées pour l'obtenir. Ces données sont créées conformément à la description des paramètres les plus adaptés à leur utilisation. Les informations sur les données des capteurs ont été affaiblies ou rendues moins précises mais plus adaptées.

Le modèle cite dans la figure 1.4 est proposé dans [13] et [14]. C'est un modèle de gestion de l'information contextuelle construit sur une hiérarchie à quatre niveaux d'abstraction (capture, transformation, reconnaissance et adaptation) et de support. Mécanismes de basculement, de sécurité, de confidentialité et de confiance.

FIGURE 1.4 – Pyramide de contexte

La couche de capture agit comme une interface avec les capteurs physiques au niveau d'abstraction le plus bas en masquant leur variabilité et en encapsulant les informations qu'ils fournissent sous forme d'observables.

Les observables numériques sont utilisées par la couche de transformation pour créer des observables symboliques. Une fois les entités détectées, identifiées et suivies, il entreprend de créer un réseau d'observables à partir duquel il peut déduire leurs relations et leurs responsabilités.

Les réseaux de contexte et de scénario doivent être traversés par la couche d'identification pour identifier le contexte et la situation actuels. Il est également chargé de détecter les changements dans le contexte et les réseaux de situation. Quelle est la situation actuelle ? Est une question d'abstraction de haut niveau à laquelle ce niveau peut répondre.

La couche d'adaptation, située au sommet de la pyramide, permet de résoudre toutes les incohérences entre l'infrastructure et les applications clientes, y compris l'adaptation de la structure des données, l'adaptation du protocole de communication, l'adaptation du formalisme et surtout la transformation d'une requête applicative en les termes attendus par l'infrastructure dans le niveau d'abstraction souhaité.

Ces conceptions ne précisent pas comment transmettre les informations de contexte, tout en étant suffisamment générales pour permettre toutes les applications ubiquitaires. Des ontologies ont été naturellement employées à cette fin dans plusieurs propositions [15] et [16] . Cependant, "haute température" n'est pas un descripteur absolu, et l'utilisation de l'infrastructure contextuelle dans l'intelligence ambiante reste souvent liée à une certaine application ou zone. Cela dépend si l'espace sera utilisé comme chambre ou comme sauna. Étant donné que le but des programmes sensibles au contexte est d'aider les utilisateurs dans le contexte, cette dépendance est appropriée.

En conséquence, AmI modifie souvent la caractérisation d'un contexte du point de vue d'une application, ce qui rend difficile pour les applications d'échanger des données contextuelles, c'est-à-dire de construire dynamiquement de nouvelles applications en utilisant la caractérisation du contexte créée pour des applications antérieures.

1.8 Le rôle de l'intelligence artificielle pour l'AmI

Nous avons vu qu'un système AmI doit être capable de faire fonctionner une variété de matériels, à savoir des capteurs et des actionneurs, afin d'atteindre certains objectifs. Donner à un utilisateur les outils pour définir et automatiser autant de paramètres d'environnement que possible est insuffisant pour cela. Il est essentiel que la technologie soit conviviale, discrète et

n'augmente pas la charge de travail de l'utilisateur. En effet, il a été constatés que les systèmes domotiques "classiques" n'ont pas connu un essor important car ils manquent d'adaptabilité et sont difficiles à exploiter pour un débutant. De ce fait, malgré l'intérêt écologique potentiel, la majorité des consommateurs potentiels ne voient pas beaucoup d'intérêt à acheter et utiliser de tels systèmes. En plus des problèmes d'optimisation, l'utilisation de l'intelligence artificielle pourrait libérer l'utilisateur du fardeau du contrôle de l'équipement, qui devient inévitablement plus sophistiqué au fur et à mesure que les objectifs sont poursuivis. Devant tenir compte de beaucoup de données et d'équipements, ainsi que de facteurs qui peuvent être hostiles, tels que le confort et la consommation d'énergie. Comme nous l'avons dit précédemment, pour que le système fonctionne bien, il doit être à la disposition du résident et ne pas l'obliger à exercer une activité particulière. Certains équipements, tels que les volets ou les radiateurs ne peuvent pas être totalement sous le contrôle de l'utilisateur dans certains systèmes domotiques. Les utilisateurs qui critiquent souvent les inefficacités de ces systèmes ont généralement une expérience très négative avec ce type de limitation. Cette méthode fonctionne comme un automate sans les facultés d'adaptation ou d'apprentissage nécessaires à la prise en compte de l'utilisateur humain et s'appuie fréquemment sur des modèles physiques a priori. Or, selon le paradigme AmI, c'est l'utilisateur, et non le concepteur qui lui impose son système, qui doit être au cœur du gadget. Il est a priori inconcevable de répondre à ces problématiques par une simple automatisation du fait de la variabilité des paramètres (caractéristiques thermiques du bâtiment, climat), des habitants (préférences, habitudes) et des équipements. Si les limites susmentionnées sont prises en compte, l'IA peut offrir des réponses pour créer des systèmes adaptables capables d'apprendre, d'anticiper les besoins et d'agir ou de fournir des informations sans interférer ni perturber les activités de l'utilisateur. Dans le cadre d'AmI, l'IA joue un rôle crucial mais aussi assez prospectif. Un système AmI devrait idéalement inclure des capacités d'adaptation et d'apprentissage étendues, telles que la capacité de reconnaître diverses circonstances environnementales, activités, événements, etc, et la capacité d'utiliser ces informations [25].

1.9 À quoi ressemble l'intelligence ambiante aujourd'hui ?

Tout composant ou surface d'une maison intelligente peut être utilisé comme exemple d'informatique ambiante. L'aide vocale, en revanche, est la technologie qui domine désormais le marché. L'intelligence artificielle est utilisée par des assistants et des haut-parleurs intelligents pour effectuer des demandes, comme Amazon Echo et Google Nest. Les haut-parleurs intelligents ont souvent des formes, des couleurs et des tailles simples pour mieux s'intégrer à leur environnement. Vous pouvez simplement parler à voix haute aux personnes autour pour terminer une tâche au lieu d'avoir à prendre le gadget et à lui donner des instructions. L'éventail des capacités de ces assistants vocaux ne cesse de s'élargir. Ces assistants vocaux se limitaient d'abord à jouer de la musique ou à répondre à des questions sur la météo. Ces gadgets peuvent désormais effectuer des tâches telles que le ménage, le contrôle de l'éclairage et la livraison à domicile.

«L'informatique ambiante comporte trois pièces de puzzle : un capteur qui sert de déclencheur, un flux de données et une unité centrale qui peut traiter ces données » Royston Seaward [18].

Outre les assistants vocaux, l'informatique ambiante utilise un large éventail de technologies contemporaines, notamment l'informatique en nuage, l'intelligence artificielle, les capteurs, la mise en réseau, etc. La puissance de calcul peut être dans le cloud et invisible au point d'utilisation, donc il n'est pas nécessaire de se concentrer sur des gadgets visibles. Le logiciel et l'utilisation de la technologie et non le matériel lui-même, sont ce qui rend l'informatique ambiante unique.

1.10 Conclusion

L'intelligence ambiante, cette vision futuriste qui a pour le but de faciliter notre vie, couvre différents domaines incluant des applications extrêmement diverses. Mais, elle subit encore des défis qu'on doit leur faire face pour qu'on puisse appliquer cette vision à tous les domaines de notre quotidien.

Chapitre 2

Contexte sur l'apprentissage profond

2.1 Introduction

L'intelligence artificielle (IA) est un domaine de l'informatique et des mathématiques rassemblant un ensemble de techniques algorithmiques et de théories permettant de réaliser des machines imitant l'intelligence humaine. Il cherche à créer l'intellect afin d'aborder des puzzles difficiles. Ce but implique de simuler le phénomène de l'intelligence humaine, comme on pourrait le faire dans les sciences de la physique, de la chimie ou de la biologie. Dans de nombreux domaines, y compris la théorie des probabilités, les neurosciences, la robotique, la théorie des jeux, la santé et les transports, l'intelligence artificielle est un domaine en croissance avec des applications.

L'apprentissage automatique est une branche de l'IA qui a le but de reproduire la façon dont les humains font leurs choix. L'apprentissage automatique permet aux ordinateurs de générer des modèles d'apprentissage à partir d'énormes ensembles de données sans avoir besoin de programmation. L'apprentissage profond, qui occupe la couche inférieure suivante, est l'un des nombreux moyens de l'apprentissage automatique qui ont été très efficaces au cours des cinq années précédentes.

Dans ce chapitre, nous allons d'abord présenter les techniques d'apprentissage automatique. Ensuite, nous décrivons comment l'apprentissage automatique a été déplacé vers l'apprentissage profond pour avoir des architectures plus prometteuses. Et enfin, nous discuterons l'application d'apprentissage automatique et l'apprentissage profond, pour l'extraction des règles de prédiction. Nous finissons le chapitre par une conclusion.

2.2 Intelligence artificielle

La notion d'intelligence artificielle est apparue dans les années 1950, coïncidant avec les progrès des premiers ordinateurs et les travaux de [19], considéré comme l'un des fondateurs de l'intelligence artificielle [20].

Dans son document Computing Machinery and Intelligence [19], il demande si une machine est capable de "penser". Une explication succincte du domaine serait : « *l'effort d'automatiser les tâches intellectuelles normalement effectuées par les humains* » Francois Chollet.

En tant que tel, l'IA est un vaste domaine qui couvre non seulement l'apprentissage automatique et l'apprentissage profond, mais aussi beaucoup d'autres techniques qui n'impliquent pas l'apprentissage [21].

De nombreux experts soutiennent depuis longtemps que l'intelligence artificielle au niveau

humain peut être atteinte en veillant à ce que les programmeurs fournissent un ensemble suffisamment complet de règles claires pour manipuler l'information à la main. Cette méthode est connue sous le nom d'IA symbolique, elle a été le paradigme dominant de l'IA entre les années 1950 et la fin des années 1980. Il était à son apogée de proéminence pendant le boom du système expert des années 1980 [21].

Bien que l'IA symbolique se soit révélée utile pour aborder des problèmes logiques bien définis tels que les échecs, trouver des règles explicites pour traiter des questions floues plus complexes telles que la classification des images, la reconnaissance vocale et la traduction s'est avérée problématique. L'apprentissage automatique a pris la place de l'IA symbolique [21].

2.3 L'apprentissage Automatique

L'apprentissage automatique «en Anglais Machine Learning (ML) » découle de cette question : un ordinateur pourrait-il aller au-delà de « ce que nous savons lui ordonner de faire » et apprendre par lui-même comment exécuter une tâche précise ? Un ordinateur pourrait-il nous surprendre ? Au lieu que les programmeurs élaborent des règles de traitement des données à la main, un ordinateur pourrait-il apprendre automatiquement ces règles en regardant les données ? Cette question ouvre la porte à un nouveau paradigme de programmation. Dans la programmation classique, le paradigme de l'IA symbolique, les règles d'entrée des humains (un programme) et les données à traiter selon ces règles, et en sortent des réponses (voir la figure 2.1). Avec l'apprentissage automatique, les humains entrent des données ainsi que les réponses attendues des données, et en sortent les règles. Ces règles peuvent ensuite être appliquées à de nouvelles données pour produire des réponses originales [21].

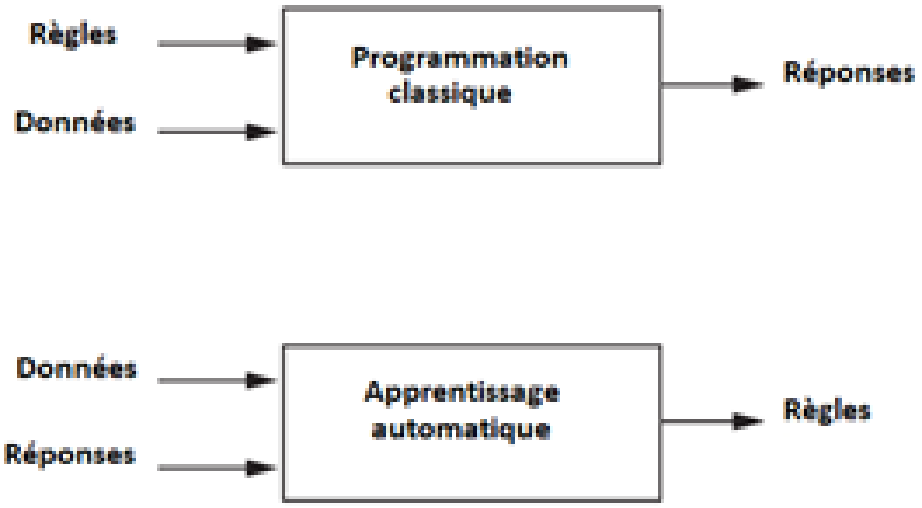

FIGURE 2.1 – Apprentissage automatique : un nouveau paradigme de programmation

Un système d'apprentissage automatique est formé plutôt qu'explicitement programmé. Il est présenté avec de nombreux exemples pertinents à une tâche, et il trouve la structure statistique dans ces exemples qui permet finalement au système de trouver des règles pour automatiser la tâche. Par exemple, si vous souhaitez automatiser la tâche d'étiqueter vos photos de vacances, vous pouvez présenter un système d'apprentissage automatique avec de nombreux exemples d'images déjà étiquetées par les humains, et le système apprendrait des règles statistiques pour associer des images spécifiques à des balises spécifiques [21].

Bien que l'apprentissage automatique n'ait commencé à prospérer que dans les années 1990, il est rapidement devenu le sous-domaine de l'IA le plus populaire et le plus réussi, une tendance stimulée par la disponibilité d'un matériel plus rapide et d'ensembles de données plus volumineux. L'apprentissage automatique est étroitement lié aux statistiques mathématiques, mais il diffère des statistiques de plusieurs manières importantes. Contrairement aux statistiques, l'apprentissage automatique a tendance à traiter de grands ensembles de données complexes (tels qu'un ensemble de données de millions d'images, chacune composée de dizaines de milliers de pixels) pour lesquels une analyse statistique classique telle que l'analyse bayésienne ne serait pas pratique. En conséquence, l'apprentissage automatique et en particulier l'apprentissage profond, présente relativement peu de théorie mathématique - peut-être trop peu - et est orienté vers l'ingénierie. C'est une discipline pratique dans laquelle les idées sont prouvées empiriquement plus souvent que théoriquement [21], la figure 2.2 illustrée le schéma général du l'apprentissage automatique.

FIGURE 2.2 – Schéma général du Machine Learning

Le processus d'apprentissage repose sur les étapes suivantes :

1. Alimenter un algorithme en données.
2. Utilisez ces données pour entraîner un modèle.
3. Tester et déployer le modèle.
4. Utiliser le modèle déployé pour effectuer une tâche prédictive automatisée.

2.4 Les types d'apprentissage automatique

Il existe plusieurs types d'apprentissage automatique. Les deux questions suivantes doivent être abordées afin de définir le type d'apprentissage, selon [30] :

o Est-ce que cet apprentissage compte sur la supervision humaine dans son entrainement et apprentissage ?

o Est-ce que ce type cet apprentissage utilise une base de données fournie par l'être humain ?

1. Nous avons supervisé l'apprentissage si la réponse aux deux questions est oui.
2. Parlons d'apprentissage non supervisé si la réponse à la première question est non et la réponse à la deuxième question est oui.
3. Si les réponses aux deux questions sont négatives, le renforcement est le type d'apprentissage.

Dans ce qui suit, nous définissons chacun de ces types : apprentissage supervisé, apprentissage non supervisé, et apprentissage par renforcement.

2.4.1 L'apprentissage supervisé

C'est une méthode d'apprentissage qui utilise des données étiquetées (voir la figure 2.3). L'environnement dispose d'un ensemble d'entrées et de sorties correspondantes pour les techniques DL supervisées. Il modifiera ensuite les paramètres du réseau de manière itérative pour obtenir une meilleure approximation des sorties prévues [23].

FIGURE 2.3 – Processus d'apprentissage supervisé

Classification

Dans ces types de problèmes, nous prédisons la réponse sous forme de classes spécifiques, telles que «oui» ou «non». Lorsque seulement 2 classes sont présentes, cela s'appelle une classification binaire. Pour plus de 2 valeurs de classe, cela s'appelle une classification multi-classes. Les valeurs de réponse prédites sont des valeurs discrètes. Par exemple, Est-ce l'image du soleil ou de la lune ? L'algorithme de classification sépare les données en classes.

Régression

Les problèmes de régression prédisent la réponse sous forme de valeurs continues telles que la prédiction d'une valeur comprise entre -infini et l'infini. Cela peut prendre plusieurs valeurs. Par exemple, l'algorithme de régression linéaire qui est appliqué, prédit le coût de la maison en fonction de nombreux paramètres tels que l'emplacement, l'aéroport à proximité, la taille de la maison, etc.

2.4.2 L'apprentissage non supervisé

Il s'agit d'un processus d'apprentissage qui se produit en l'absence d'étiquettes de données (voir la figure 2.4). Dans cette situation, le réseau découvre des liens non découverts ou une structure dans les données d'entrée par l'apprentissage de la représentation interne ou des propriétés essentielles. Les méthodes d'apprentissage non supervisées comprennent le regroupement, la réduction de la dimensionnalité et les algorithmes génératifs [23].

FIGURE 2.4 – Processus général de l'apprentissage non-supervisé

Il y a deux types d'apprentissage non supervisé, selon [30] :

Regroupement (clustering)

Stratégie d'analyse statistique utilisée pour classer les données brutes en silos homogènes, les données étant regroupées à l'intérieur de chaque cluster en fonction d'un attribut partagé.

Réduction des dimensions

L'objectif est de simplifier les données tout en conservant autant d'informations que possible, car en combinant de nombreux attributs en un seul caractère.

2.4.3 L'apprentissage par renforcement

L'apprentissage par renforcement est un type d'algorithme d'apprentissage automatique qui permet aux agents logiciels et aux machines d'évaluer automatiquement le comportement optimal dans un contexte ou un environnement particulier pour améliorer son efficacité [24]. En général, il est capable de percevoir et d'interpréter son environnement, de prendre des mesures et d'apprendre par essais et erreurs, c'est-à-dire une approche axée sur l'environnement. Dans lequel l'environnement est typiquement modélisé comme un processus de décision Markov et les décisions sont prises en utilisant une fonction de récompense.

2.5 l'apprentissage à base des réseaux de neurones artificiels

Les réseaux de neurones artificiels (en Anglais Artificiel Neural Network (ANN)) sont présentés comme la vague du futur dans le domaine de l'informatique et neuro-informatique. Ces processus d'auto-apprentissage permettent aux ordinateurs d'effectuer des tâches conformément à la méthodologie de l'intelligence artificielle [25].

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle l'information que chaque processeur élémentaire obtient est utilisée pour calculer une seule sortie. De toute évidence, un réseau est une structure hiérarchique [26].

2.5.1 Généralités sur les réseaux de neurones artificiels

Inspiration des réseaux de neurones artificiels

Il y a 10^{12} neurones liés dans le système neurologique. Bien que les neurones soient très variés, ils fonctionnent tous selon les mêmes principes [26], la figure 2.5 illustre le chemin du neurone

biologique vers le neurone artificiel.

Une cellule avec un corps cellulaire et un noyau est un neurone. Dendrites sont créées lorsque le corps cellulaire se divise. Parfois, il y a tellement d'entre eux que nous nous référons à des poils dendritiques ou arborisation dendritique. Les dendrites sont la voie par laquelle l'information passe de l'extérieur au soma, le corps du neurone.

L'axone est ensuite utilisé par le neurone pour transmettre l'information qu'il a traitée aux autres neurones. Il n'y a pas de communication directe entre deux neurones. L'axone du neurone afférent et les dendrites du neurone efférent ont vraiment une lacune intercellulaire de quelques dizaines d'Angströms. Le terme "synapse" désigne l'intersection de deux neurones [26].

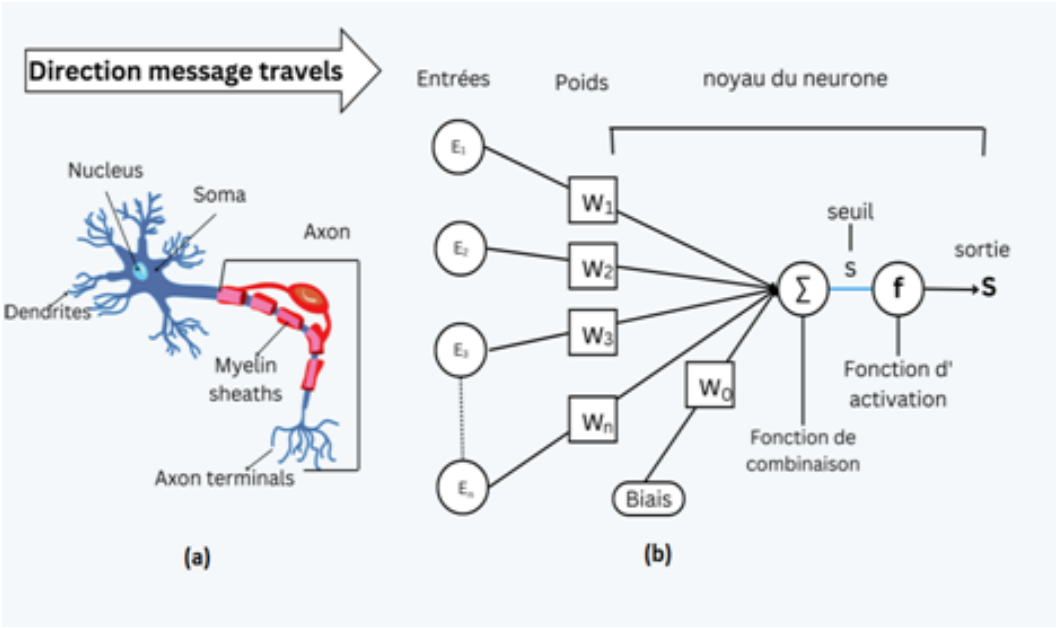

FIGURE 2.5 – (a) Neurone réel et (b) neurone artificiel

On pourra résumer une modélisation de tel neurone par le tableau 2.1 et la figure 2.6, qui nous permettra de voir clairement le passage du neurone biologique vers le neurone formel.

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions
Axones	Signal de sortie
Dendrite	Signal d'entrée
Somma	Fonction d'activation

TABLE 2.1 – Passage du neurone biologique vers le neurone formel

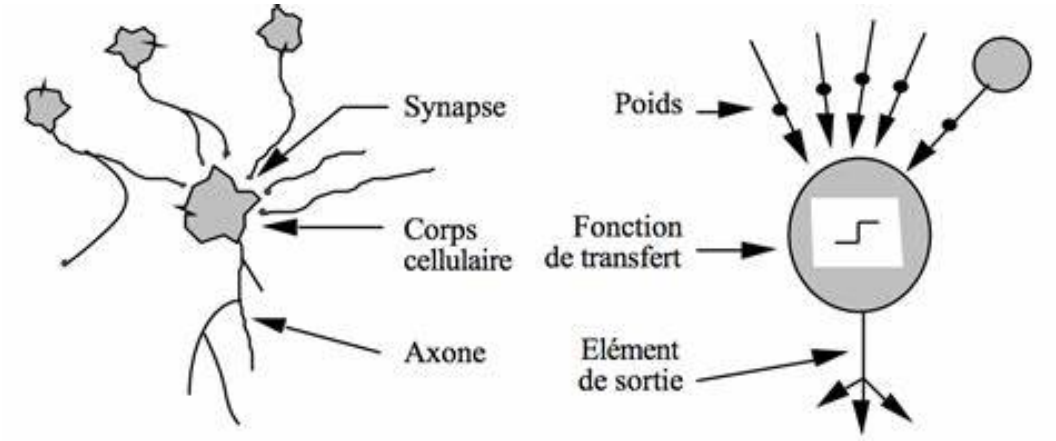

FIGURE 2.6 – Passage du neurone biologique vers le neurone formel

Les éléments constitutifs d'un neurone artificiel

Les éléments qui constituent un neurone artificiel sont les suivants [27] :

— Les entrées "E" du neurone proviennent soit d'autres éléments "processeurs", soit de l'environnement.

— Les poids "W" indiquent l'influence de chaque entrée.

— Entrée de biais (bais input) : une unité fictive dont le poids permet d'ajuster le seuil de déclenchement du neurone.

— La fonction de combinaison "s" combine les entrées "E" et les poids "W" comme suite :

$$s = Xni = 1WiEi \tag{2.1}$$

Wi : Poids de la connexion à l'entrée i.

Ei : Signal de l'entrée i.

— La Fonction d'activation"f" détermine l'´etat du neurone en sortie "S".

•Calcul de la sortie :

$$S = f(s) \tag{2.2}$$

•Ou encore :

$$S = f(Xni = 1WiEi) \tag{2.3}$$

— La fonction d'activation"f" calcule la sortie "S" du neurone en fonction de la combinaison "s" en entrée et peut avoir plusieurs formes :

La Fonction Relu : C'est une fonction d'activation très simple. Supposons que l'entrée est la valeur x et si x est positif, la sortie sera x sinon 0. La fonction Relu (unité linéaire rectifiée) est :

$$f(x) == max(0, x) \tag{2.4}$$

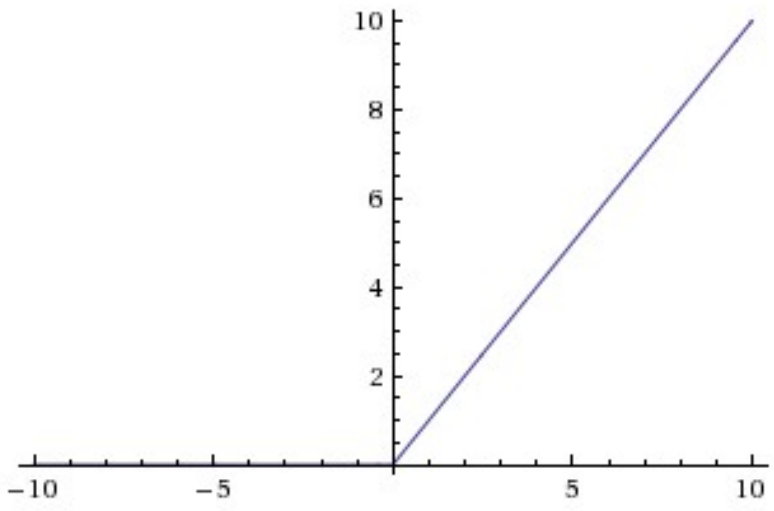

FIGURE 2.7 – Représentation graphique de la fonction Relu [28]

La Fonction sigmoïde : sont les fonctions les plus utilisées dans la création de réseaux neurones artificiels. Prend une entrée réelle et le réduit entre 0 et 1.

$$F(x) = 1/1 + e^{(} - x) \tag{2.5}$$

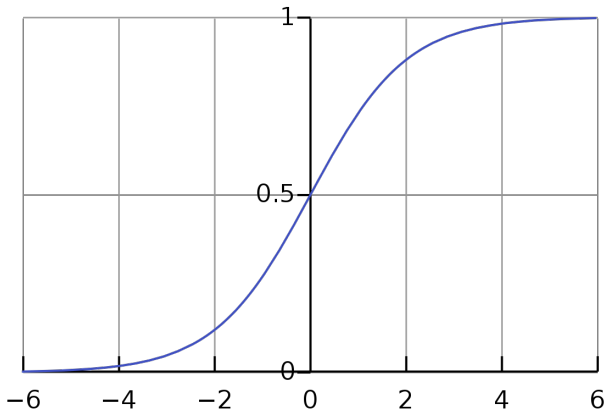

FIGURE 2.8 – Représentation du graphe de fonction sigmoïde [28]

La fonction Tangente hyperbolique : prend une entrée de valeur réelle et le réduit à une valeur dans [-1, 1].

$$F(x) = tanh(x) = 1 - e^{(} - 2x)/1 + e^{(} - 2x) \tag{2.6}$$

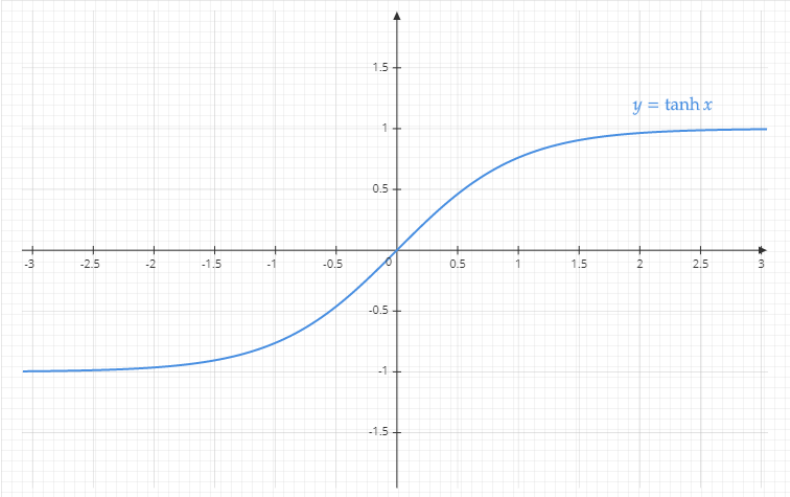

FIGURE 2.9 – Représentation du graphe de fonction Tangente hyperbolique [28]

2.5.2 Fonctionnement d'un réseau de neurones artificiel

Un réseau neuronal artificiel est composé de quelques dizaines, centaines, milliers, voire des millions de ces unités qui sont empilées dans une séquence de couches, dont chacune est connectée aux couches de chaque côté. Certaines d'entre elles, appelées unités d'entrée, sont utilisées pour obtenir divers types d'information du monde extérieur que le chercheur tenterait de comprendre, de reconnaître ou d'analyser. Ce sont les unités de production, qui sont situées à l'opposé du diagramme et qui montrent comment elles se rapportent aux connaissances qu'elles ont acquises [29].

Une ou plusieurs couches d'unités cachées, qui constituent collectivement la majorité du cerveau artificiel, sont positionnées entre les unités d'entrée et de sortie. La majorité des ré-ensembles neuronaux sont entièrement liés, ce qui implique que toutes les unités sont cachées et connectées à toutes les autres unités et couches des deux côtés. Une valeur de poids, qui peut être positive ou négative, est utilisée pour exprimer les connexions entre deux unités. Une unité en affecte une autre lorsque le poids est plus élevé [29].

Exemple d'un réseau de neurone

Parmi les architecteurs de réseau de neurones on a :

Perceptron simple

Le premier modèle de neurones suggéré était le perceptron monocouche(voir la figure 2.10). Un vecteur de poids constitue la mémoire locale du neurone. Une perception monocouche est calculée en additionnant les vecteurs d'entrée qui ont été multipliés par des éléments de vecteur de poids correspondants pour déterminer leurs valeurs. L'entrée d'une fonction d'activation sera la valeur présentée dans la sortie [30].

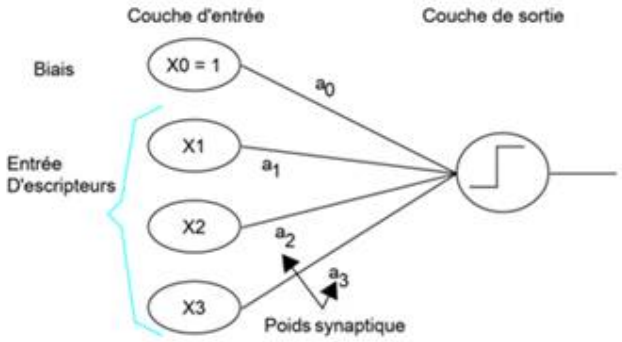

FIGURE 2.10 – Perceptron simple [31]

2.6 De l'apprentissage peu-profond à l'apprentissage profond

Nous ne pouvons pas différer sur l'efficacité de l'apprentissage automatique dans la résolution d'une variété de problèmes, mais il y a beaucoup de problèmes de ce type, dont les plus importants sont le temps, la vitesse et l'efficacité. Le développement de l'apprentissage profond a été motivé en partie par l'échec des algorithmes traditionnels, lors du travail avec des données à grande échelle, et en raison de sa structure de réseau neuronal artificiel, l'apprentissage profond excelle à identifier les modèles dans les données non structurées telles que les images, son, vidéo et texte. Pour cette raison, l'apprentissage profond transforme rapidement de nombreux secteurs, y compris la santé, l'énergie, les finances et les transports. Ces secteurs repensent actuellement leurs processus opérationnels traditionnels.

2.7 L'apprentissage profond « Deep Learning »

Les développements récents de l'intelligence artificielle ont été grandement facilités par un ensemble d'algorithmes d'apprentissage automatique appelés apprentissage profond. Dans ap-

prentissage automatique, un programme analyse un ensemble de données afin d'en déduire des règles qui permettront de tirer des conclusions sur de nouvelles données.

L'apprentissage profond est basé sur ce que l'on appelle des « réseaux de neurones artificiels », composés de milliers d'unités (les neurones) qui effectuent chacune de petites opérations simples.Une première couche de calculs de "neurones" fournit l'entrée pour les calculs d'une deuxième couche, et ainsi de suite.

Par exemple, les couches initiales d'unités pour la reconnaissance visuelle reconnaissent les lignes, les courbes et les angles... Les couches supérieures distinguent des formes, des combinaisons de formes, des objets, des contextes...

La croissance des bases de données massives (souvent appelées "big data") et l'expansion de la puissance de calcul ont notamment permis des avancées en matière d'apprentissage profond.

Voici la figure 2.11 qui nous montre la relation entre les trois (3) concepts cités, Intelligence Artificielle (AI), Machine Learning (ML) et Deep Learning (DL).

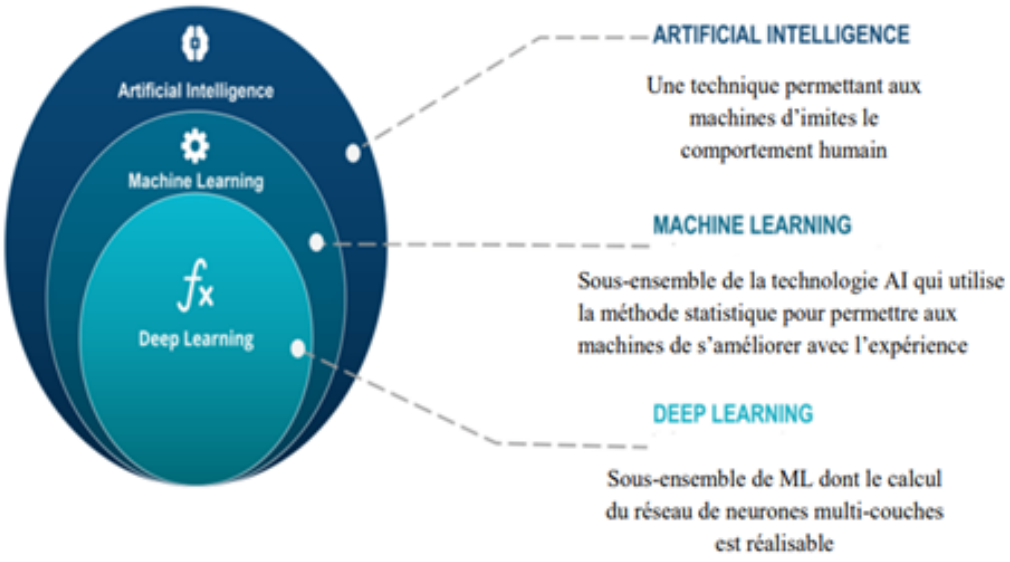

FIGURE 2.11 – La relation entre l'IA, ML et le Deep Learning [32].

2.8 Les différents modèles de Deep Learning

Il existe différents types de modèles de réseaux neuronaux qui ont été élaborés au fil des ans pour être utilisés dans les problèmes de classification et de régression, le traitement du langage,

la synthèse vocale et d'autres domaines pertinents. Trois grands modèles de réseaux neuronaux ont obtenu de bonnes performances sur divers problèmes.

2.8.1 Perceptron Multicouches

Perceptron multicouches (en Anglais Mutilayer Perceptron (MLP))est un type de réseau de neurone Feed-forward qui se caractérise par une couche d'entrée, un certain nombre de couches intermédiaires et une couche de sortie, qui sont entièrement connectées (voir la figure 2.12). Un MLP utilise back-propagation pour la formation. Le terme Feed-forward se réfère à l'architecture en couches dans le réseau, en particulier qu'il n'y a pas de cycles dans le réseau. La structure de la couche garantit qu'il n'existe aucun cycle, car les couches ne sont autorisées à avoir des poids que de la couche directement précédente. Le terme entièrement connecté se réfère au fait que dans MLP, tous les nœuds dans une couche donnée ont un poids à tous les nœuds dans la couche précédente [33].

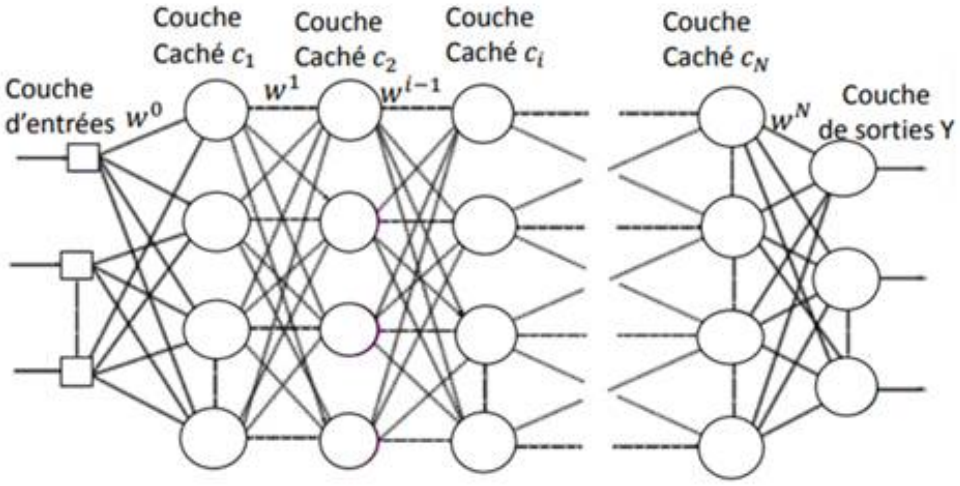

FIGURE 2.12 – Perceptron multicouches

2.8.2 Réseau de neurones convolutifs

Ces dernières années, le réseau de neurones convolutés (en Anglais convolutional Neural Network (CNN)) est devenu un important algorithme d'apprentissage profond de pointe. CNN est un réseau de neurones feedforward avec une structure profonde qui a une ou plusieurs couches convolutés, qui s'inspire de l'organisation du cortex visuel animal. Le réseau utilise une opération mathématique appelée convolution, d'où le nom utilisée pour l'extraction de caractéristiques. Cela peut être vu comme une multiplication par une matrice. Contrairement aux réseaux de neurones traditionnels entièrement connectés, les neurones CNN ont une connectivité locale (interactions

rares), donc ils n'ont pas besoin d'être connectés à toutes les sorties de la couche de neurones précédente. Les CNN sont conçus pour traiter des données qui se présentent sous la forme de plusieurs tableaux [34].

2.8.3 Réseaux neuronaux récurrents

Les réseaux neuronaux récurrents (en Anglais Recurrent Neural Network (RNN)) sont des réseaux de neurones qui possèdent des boucles de rétroaction dans leur structure de connexion. Ces boucles permettent au réseau de conserver et de transférer l'information à lui-même, ce qui modifie sa dynamique et lui permet de se maintenir. Ces modèles étaient souvent favorisés pour le traitement automatique de la parole et des séquences en général, car leurs caractéristiques leur permettent d'apprendre, de stocker et de tenir compte du contexte passé lors du traitement de l'information présente [35].

Par exemple, si nous traitons une phrase de 3 termes, le réseau sera déployé sous la forme d'un réseau neuronal à 3 couches, avec une couche pour chaque mot. La figure 2.13 illustre cette idée.

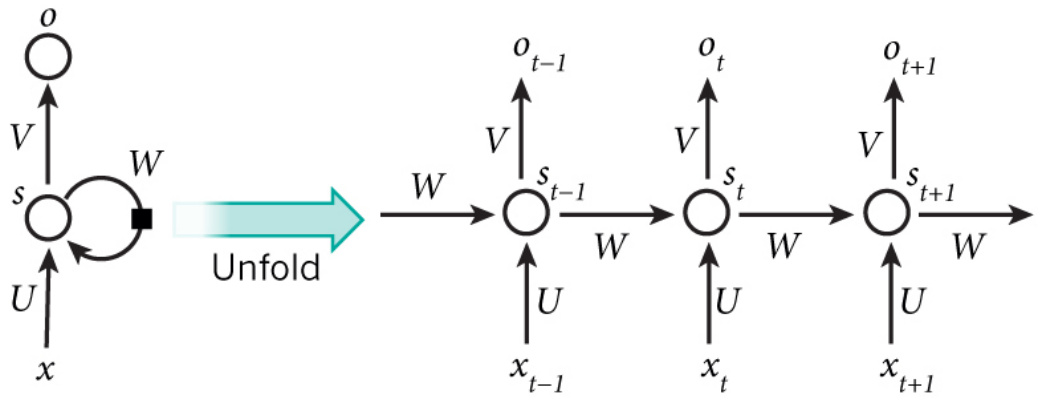

FIGURE 2.13 – Un exemple d'un réseau récurrent qui se déroule

2.8.4 Réseaux Long Short-Term Memory (LSTM)

Ils sont une forme de RNN qui peut apprendre les dépendances à long terme. Ils ont été initialement introduits par l'auteur de [36] et depuis lors, ils ont été améliorés et popularisés par plusieurs auteurs dans des études ultérieures. Les LSTM sont actuellement largement utilisés et se révèlent extrêmement efficaces pour résoudre diverses problématiques.

LSTM sont spécialement conçus pour éliminer la question de la dépendance à long terme. La mémoire à long terme est essentiellement leur comportement par défaut, ce n'est pas quelque chose qu'ils ont du mal à acquérir !

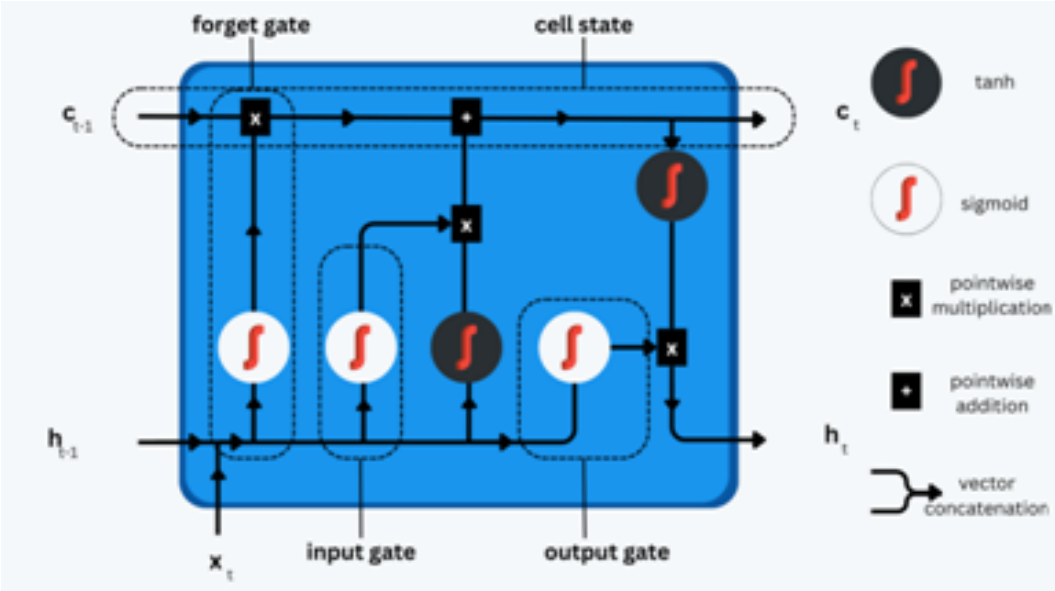

FIGURE 2.14 – Cellule LSTM et ses opérations

2.8.5 Unité récurrente à portes

Unité récurrente à portes (en Anglais Gated Recurrent Unit (GRU)) est un type de réseau neuronal récurrent qui a été introduit par les auteurs de [37] comme une alternative plus simple aux réseaux de mémoire à court terme (LSTM). Comme LSTM, GRU peut traiter des données séquentielles telles que le texte, la parole et les données de séries chronologiques.

L'idée de base derrière GRU est d'utiliser des mécanismes de synchronisation pour mettre à jour sélectivement l'état caché du réseau à chaque étape de temps. Les mécanismes de synchronisation sont utilisés pour contrôler le flux d'information entrant et sortant du réseau. Le GRU a deux mécanismes de synchronisation, appelés la porte reset et la porte update.

La porte reset détermine combien de l'état caché précédent doit être oublié, tandis que la porte update détermine combien de la nouvelle entrée doit être utilisée pour mettre à jour l'état caché. La sortie du GRU est calculée en fonction de l'état masqué mis à jour.

2.9 Conclusion

Nous avons consacré ce chapitre à la présentation des notions de base de l'apprentissage automatique et l'apprentissage profond.

Après les études nous pouvons dire que l'apprentissage profond est un apprentissage automatique, mais avec de plus grandes capacités et une approche de travail différente. Le choix de

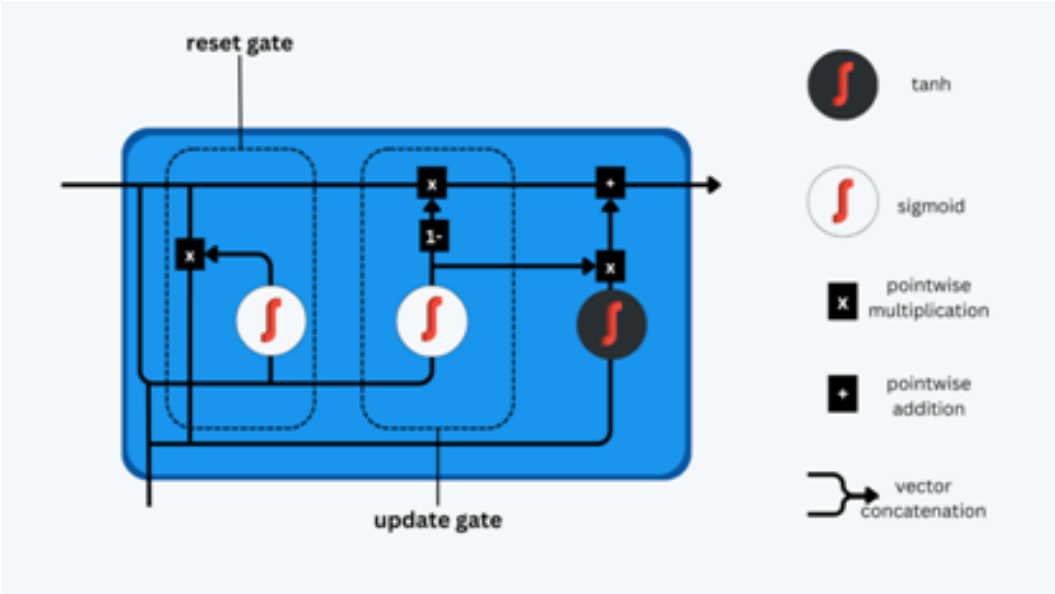

FIGURE 2.15 – Cellule GRU

l'un ou l'autre apprentissage pour résoudre un problème particulier dépend non seulement de la complexité du problème mais aussi de la quantité de données disponibles.

Le prochain chapitre, sera consacré à la présentation détaillée de notre approche proposée, qui a pour rôle la prédiction des règles, basée sur l'apprentissage profond supervisé.

Chapitre 3

Conception et Implémentation

3.1 Introduction

Ce chapitre se concentre sur les différentes étapes clés du projet. Nous commencerons par présenter le dataset utilisé, en décrivant sa composition et sa pertinence. Ensuite, nous aborderons le prétraitement du dataset, en détaillant les méthodes employées pour nettoyer et préparer les données. Nous passerons ensuite à la sélection du modèle et à la description de son architecture. Ensuite, nous décrirons l'implémentation du modèle et le processus d'entraînement, en détaillant les choix et les paramètres spécifiques. Enfin, nous discuterons des résultats obtenus et les comparerons à d'autres approches existantes. Ce chapitre offre une vue d'ensemble détaillée de la conception et de l'implémentation du projet, en fournissant des informations précises sur les différentes étapes et les choix effectués tout au long du processus.

3.2 Présentation du dataset

Dans cette étude, nous avons utilisé la base de données "Aruba" provenant du projet "CASAS Smart Home" du Centre d'études Avancées en Systems Adaptatifs de l'Université d'État de Washington. Les données ont été collectées sur une période de 7 mois auprès d'une personne âgée volontaire. Un plan d'accueil de la maison et l'emplacement des capteurs ont été fournis pour faciliter la compréhension de l'environnement (voir la figure 3.1) :

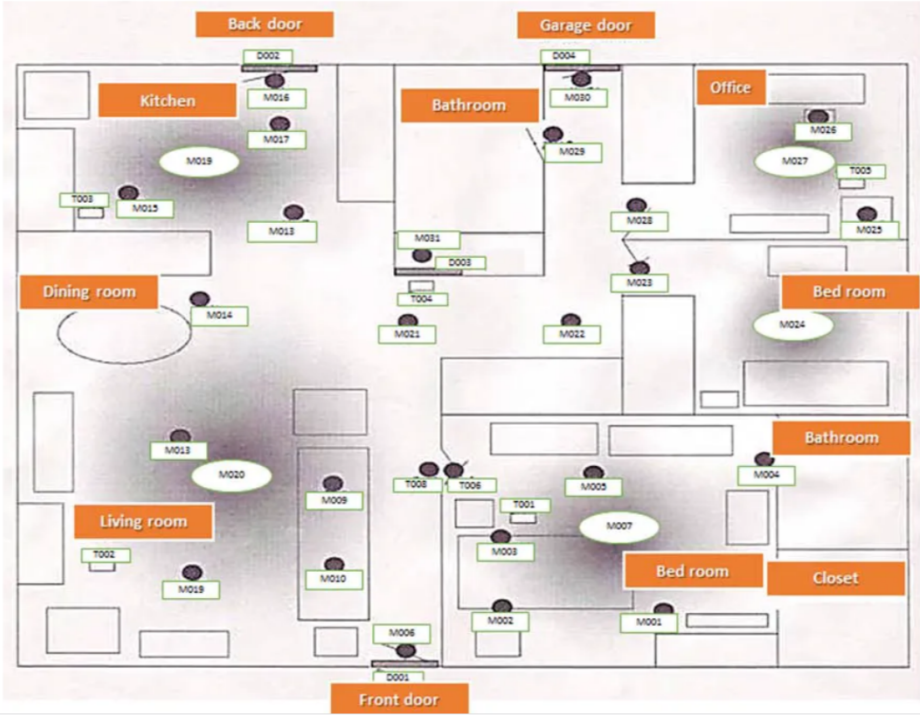

FIGURE 3.1 – Plan d'implantation des capteurs [38]

Le jeu de données comprend trois types de capteurs : les capteurs de mouvement (identifiés par des ID commençant par "M"), les capteurs de fermeture de porte (identifiés par des ID commençant par "D") et les capteurs de température (identifiés par des ID commençant par "T"). Les activations de ces capteurs ont été enregistrées dans le temps. Voici un exemple représentatif du jeu de données dans la figure suivante :

2010-11-04	05:40:51.303739	M004	ON	Bed_to_Toilet	begin
2010-11-04	05:40:52.342105	M005	OFF		
2010-11-04	05:40:57.176409	M007	OFF		
2010-11-04	05:40:57.941486	M004	OFF		
2010-11-04	05:43:24.021475	M004	ON		
2010-11-04	05:43:26.273181	M004	OFF		
2010-11-04	05:43:26.345503	M007	ON		
2010-11-04	05:43:26.793102	M004	ON		
2010-11-04	05:43:27.195347	M007	OFF		
2010-11-04	05:43:27.787437	M007	ON		
2010-11-04	05:43:29.711796	M005	ON		
2010-11-04	05:43:30.279021	M004	OFF	Bed_to_Toilet	end
2010-11-04	05:43:45.7324	M003	ON	Sleeping	begin
2010-11-04	05:43:52.044085	M003	OFF		
2010-11-04	05:43:53.185335	M002	ON		
2010-11-04	05:43:53.253809	M003	ON		
2010-11-04	05:43:59.493281	M002	OFF		
2010-11-04	05:44:04.048766	M003	OFF		
2010-11-04	05:44:06.14204	M003	ON		
2010-11-04	05:44:11.229146	M003	OFF		

FIGURE 3.2 – Exemple de data set [38]

Chaque ligne du tableau présente des informations sur la date, l'heure et un identifiant unique. Les deux dernières colonnes indiquent le nom de l'activité en cours et son statut (début ou fin). L'ensemble de données comprend un total de 11 activités distinctes, avec le nombre d'occurrences de chaque activité entre parenthèses : Préparation des repas (1606), Détente (2910), Repas (257), Travail (171), Sommeil (401), Faire la vaisselle (65), Allée des toilettes au lit (157), Entrer à la maison (431), Quitter la maison (431), Ménage (33) et Respire (6). Les activités ont été annotées manuellement par les chercheurs.

3.3 Problème du dataset

Le dataset que nous avons décrite présente plusieurs problèmes qui nécessitent une attention particulière :

3.3.1 Forme générale

Notre dataset présente des problèmes de structure en raison de la présence de valeurs manquantes. Plus précisément, les deux dernières colonnes contiennent des détails sur le nom de l'activité et son statut (début ou fin), et il y a de nombreuses cases vides entre chaque transition de début à fin ou vice versa. Cela crée une irrégularité dans la forme générale du dataset (voir la

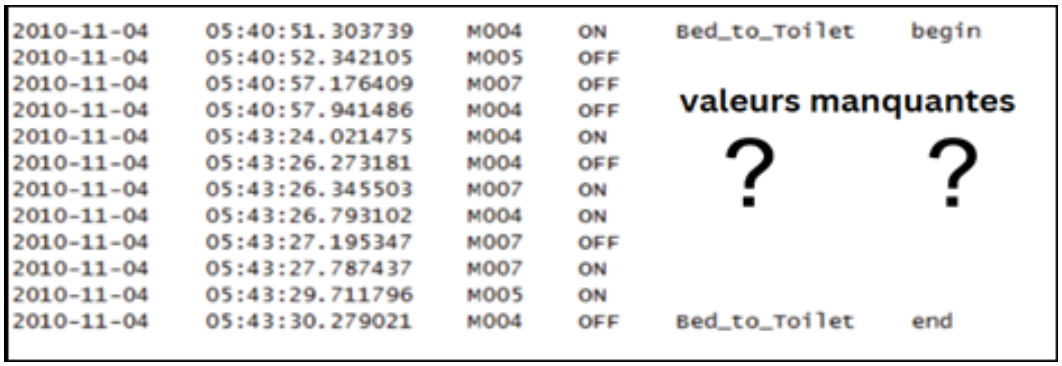

FIGURE 3.3 – Capture d'écran : "Aperçu du jeu de données - Problème de Forme générale"

figure 3.3).

3.3.2 Problème d'extension

Notre dataset est actuellement au format .txt, mais nous souhaitons la convertir en format .csv. L'extension .csv (Comma-Separated Values) est couramment utilisée pour les données tabulaires, ce qui facilite leur manipulation et leur analyse (voir la figure 3.4).

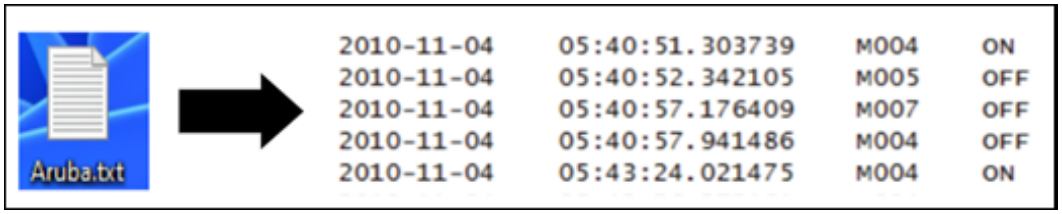

FIGURE 3.4 – Capture d'écran : "Aperçu du jeu de données - Problème d'extension"

3.3.3 Valeurs manquantes

Plusieurs cases dans plusieurs colonnes de notre dataset sont vides. Cela signifie que certaines informations n'ont pas été enregistrées ou sont indisponibles pour ces entrées. Les valeurs manquantes peuvent poser des problèmes lors de l'analyse et nécessitent souvent un traitement approprié pour éviter les erreurs ou les biais dans les résultats.

3.3.4 Conversion de valeurs textuelles en valeurs numériques

La plupart des cases de notre dataset contiennent des valeurs textuelles, mais nous souhaitons les convertir en valeurs numériques (voir la figure 3.5). Cette conversion peut être nécessaire pour effectuer des calculs, des analyses statistiques ou pour utiliser certains algorithmes d'apprentissage

automatique qui requièrent des données numériques.

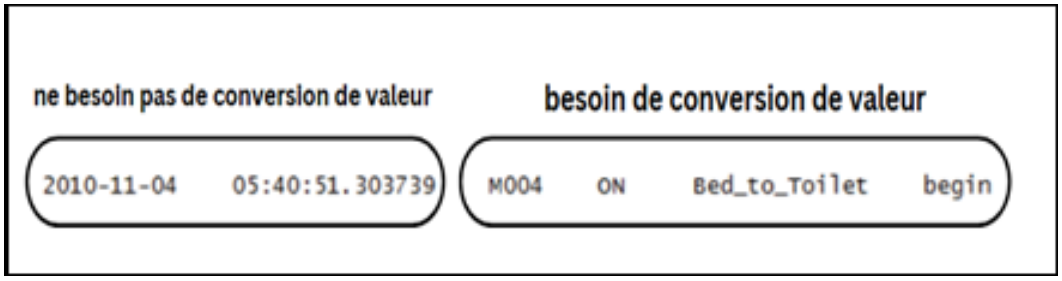

FIGURE 3.5 – Capture d'écran : "Aperçu du jeu de données - Problème de Conversion de valeurs"

3.3.5 Déséquilibre des classes

Notre jeu de données présente un déséquilibre au niveau des classes, ce qui signifie qu'il y a une variation dans le nombre de occurrences de chaque classe. Certains types d'activités peuvent être représentés de manière disproportionnée par rapport aux autres. Cela peut avoir un impact sur les performances des modèles d'apprentissage automatique utilisant ce jeu de données, car ils peuvent être biaisés en faveur des classes majoritaires. Le tableau ci-dessous interprète cette situation :

Nom activité	Détente	Préparation des repas	Repas	Travail	Sommeil	vaisselle	Toilettes	Entrer a la maison	Quitter la maison	Ménage	Respire
NB d'instances	1606	2910	257	171	401	65	157	431	431	33	6

TABLE 3.1 – Répartition déséquilibrée des classes dans le jeu de données

3.3.6 Types de capteurs et valeurs numériques

Dans la colonne "sensor", nous mentionnons qu'il existe un type de capteur température qui fournit des valeurs numériques, tandis que les autres capteurs indiquent simplement "on" ou "off" (voir la figure 3.6). Cela indique une différence de format entre les types de capteurs et nécessite une normalisation pour que toutes les valeurs soient dans un format cohérent (par exemple, en assignant des valeurs numériques spécifiques pour "on" et "off").

3.4 Prétraitement du dataset

Le prétraitement de notre jeu de données est une étape essentielle dans le processus d'analyse et de modélisation des données. Il vise à préparer les données brutes afin de les rendre exploitables et adaptées à l'objectif de notre étude. Le prétraitement comprend :

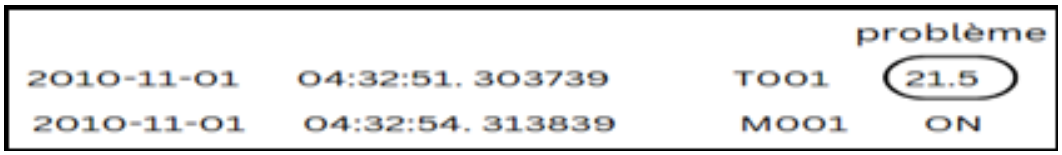

FIGURE 3.6 – Capture d'écran : "Aperçu du jeu de données - Problème de Conversion de types de capteurs"

3.4.1 Problème d'extension

Pour convertir notre fichier .txt en .csv à l'aide d'un script Python la méthode utilisée consiste à ouvrir le fichier .txt en mode lecture, puis à créer un nouveau fichier .csv en mode écriture. Ensuite, vous pouvez parcourir chaque ligne du fichier .txt et extraire les valeurs pertinentes en utilisant des opérations de manipulation de chaînes. Une fois les valeurs extraites, vous pouvez les écrire dans le fichier .csv nouvellement créé en les séparant par des virgules pour respecter le format CSV. Après avoir traité toutes les lignes du fichier .txt, vous fermez les deux fichiers pour finaliser la conversion. la résultat finale est dans la figure suivante :

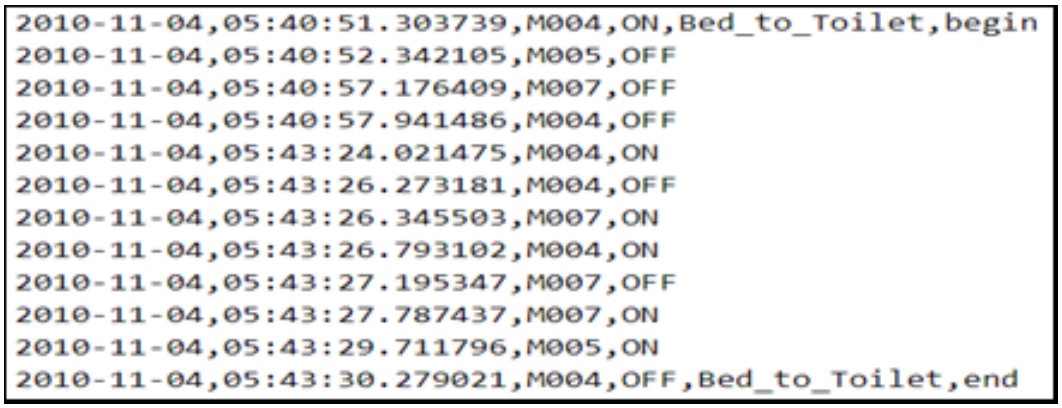

FIGURE 3.7 – Capture d'écran : Distribution des exemples après conversion l'extension

3.4.2 Forme générale

Nous employons un script Python afin d'éliminer la dernière colonne contenant les valeurs "begin, end" et de substituer, dans la colonne "activity_name", les valeurs manquantes par le nom de l'activité précédente. Le résultat final s'avère être dans la figure suivant :

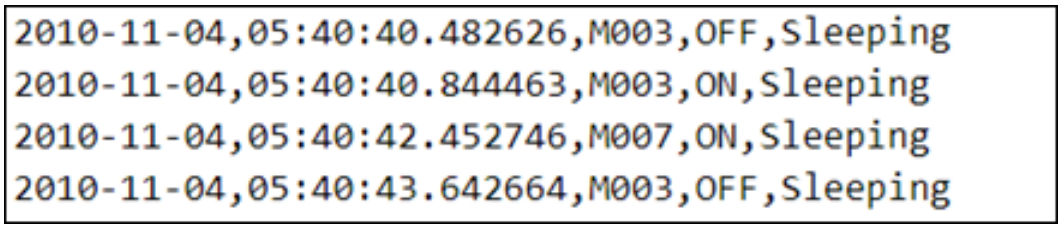

FIGURE 3.8 – Capture d'écran : Structure et colonnes du jeu de données après le prétraitement

3.4.3 Valeurs manquantes

Les valeurs manquantes numériques sont substituées par la moyenne ou la médiane des valeurs existantes dans la colonne correspondante. En ce qui concerne les variables catégorielles, les valeurs manquantes sont remplacées par le mode, c'est-à-dire la valeur la plus fréquente.

3.4.4 Conversion de valeurs textuelles en valeurs numériques

On utilise un scripte python pour permettre de transformer des informations représentées sous forme de texte en formats numériques exploitables par les ordinateurs (voir le tableau 3.2).

Activité	Détente	Préparation des repas	Repas	Travail	Sommeil	vaisselle	Toilettes	Entrer a la maison	Quitter la maison	Ménage	Respire
NB	0	1	2	3	4	5	6	7	8	9	10

TABLE 3.2 – Conversion des valeurs de classe après prétraitement en numéros

3.4.5 Types de capteurs et valeurs numériques

Nous procédons à la suppression de toutes les lignes du jeu de données qui contiennent, dans la colonne "sensor", un type de capteur de température, dans le but d'améliorer la qualité globale des données et de garantir que toutes les valeurs soient dans un format cohérent.

3.4.6 Déséquilibre des classes

Nous avons rencontré un problème de déséquilibre des données. Notre ensemble de données est composé de 11 classes, mais le nombre d'exemples dans chaque classe n'est pas homogène. Afin de remédier à ce problème, nous avons mis en place les étapes suivantes :

- Suppression des classes 5, 9 et 10, car elles contiennent un nombre très inférieur d'exemples par rapport aux autres classes.
- Réarrangement des classes restantes : la classe 6 devient la classe 5, la classe 7 devient la classe 6 et la classe 8 devient la classe 7.
- Division des données en deux parties : la partie 1 comprend les classes 0, 1 et 2, tandis

que la partie 2 comprend les classes 3, 4, 5, 6 et 7.

Cette approche permet de traiter le problème de déséquilibre des données en regroupant les classes similaires et en éliminant les classes avec un nombre d'exemples insuffisant. Ainsi, les deux parties obtenues présentent une distribution plus équilibrée des classes, ce qui facilitera le prétraitement et l'analyse ultérieure .

3.4.7 Transformation du dataset

Dans cette étape de prétraitement, la colonne 'Date' et la colonne 'Time' ont été supprimées afin de simplifier la structure du dataset. À la place, une nouvelle colonne appelée 'timestamp' a été créée en combinant les informations de date et d'heure. Ce nouveau format de timestamp permet de représenter le temps en millisecondes, ce qui est essentiel pour le parcours des modèles LSTM et GRU.

3.5 Modèles choisies et leurs architectures

3.5.1 Long Short-Term Memory(LSTM)

Les LSTM sont largement utilisés dans les séries temporelles en raison de leur capacité à gérer les dépendances à long terme, à modéliser des séquences de longueurs variables, à apprendre des motifs complexes. Ils offrent ainsi une solution puissante pour l'analyse et la prédiction des séries temporelles.

L'architecture d'un LSTM (Long Short-Term Memory) se compose de plusieurs composants clés qui lui permettent de capturer et de mémoriser les dépendances à long terme dans une séquence temporelle. Voici les principaux éléments d'un LSTM :

Cellule mémoire (Memory cell) : C'est le composant central d'un LSTM. La cellule mémoire maintient une mémoire à long terme et permet au modèle de mémoriser des informations importantes sur une séquence temporelle. La cellule mémoire est mise à jour à chaque pas de temps à l'aide de portes spéciales appelées portes d'entrée, d'oubli et de sortie.

Porte d'entrée (Input gate) :La porte d'entrée détermine quelles informations de l'entrée actuelle doivent être mises à jour dans la cellule mémoire. Elle prend en compte l'entrée actuelle et l'état caché précédent (hidden state) pour calculer quelles informations doivent être ajoutées à la cellule mémoire.

Porte d'oubli (Forget gate) :La porte d'oubli permet de décider quelles informations de la cellule mémoire précédente doivent être oubliées. Elle utilise l'entrée actuelle et l'état caché précédent pour calculer un vecteur de pondération qui contrôle quelles informations de la cellule

mémoire doivent être préservées et quelles informations doivent être oubliées.

Porte de sortie (Output gate) : La porte de sortie détermine quelles informations de la cellule mémoire doivent être exposées à l'étape de sortie. Elle prend en compte l'entrée actuelle et l'état caché précédent pour calculer quelles informations doivent être utilisées pour générer la sortie de l'étape de temps courante.

État caché (Hidden state) : L'état caché est la représentation résumée de l'information contenue dans la séquence temporelle jusqu'à l'étape de temps courante. Il est calculé en utilisant la cellule mémoire et la porte de sortie. L'état caché est ensuite utilisé comme entrée pour l'étape de temps suivante.

3.5.2 Gated Recurrent Unit (GRU)

Les GRU sont utilisées dans les séries temporelles en raison de leur capacité à modéliser les dépendances à long terme, à éviter le problème de disparition du gradient et à capturer des motifs complexes. La structure d'une unité récurrente avec portes (GRU) comprend plusieurs composants clés qui permettent le traitement et le contrôle des informations à travers les séquences temporelles. Voici une description générale de l'architecture d'une GRU :

Entrée (Input) :À chaque pas de temps de la séquence temporelle, un vecteur d'entrée est alimenté dans la GRU. Ce vecteur représente les caractéristiques ou les données à traiter à ce moment spécifique.

Porte de réinitialisation (Reset Gate) :La porte de réinitialisation est une couche de neurones qui détermine dans quelle mesure les informations passées doivent être oubliées ou réinitialisées. Elle prend en compte le vecteur d'entrée actuel ainsi que l'état caché précédent pour calculer une valeur entre 0 et 1 pour chaque dimension de l'état caché.

Porte d'oubli (Update Gate) :La porte d'oubli est une autre couche de neurones qui contrôle dans quelle mesure les nouvelles informations doivent être incorporées à l'état caché actuel. Elle utilise à la fois le vecteur d'entrée et l'état caché précédent pour générer une valeur de mise à jour entre 0 et 1 pour chaque dimension de l'état caché.

État caché (Hidden State) : L'état caché représente la mémoire interne de la GRU et contient des informations sur les dépendances temporelles apprises jusqu'à présent. Il est mis à jour en fonction des valeurs de la porte de réinitialisation, de la porte d'oubli et du vecteur d'entrée actuel.

Nouvel état caché (New Hidden State) : En utilisant la porte de réinitialisation, la porte d'oubli et le vecteur d'entrée actuel, le nouvel état caché est calculé en combinant les informa-

tions de l'état caché précédent et du vecteur d'entrée actuel. Cela permet de mettre à jour l'état caché pour le pas de temps actuel.

Sortie (Output) :En fonction de l'état caché actuel, une sortie peut être générée à chaque pas de temps, ou une sortie finale peut être produite à la fin de la séquence temporelle.

3.5.3 Perceptron Multicouche (MLP)

La démonstration simple de l'architecture MLP est la suivante :

Couche d'entrée : Cette couche reçoit les données d'entrée initiales et transmet les informations aux neurones de la couche suivante.

Couches cachées : Il peut y avoir une ou plusieurs couches cachées entre la couche d'entrée et la couche de sortie. Chaque couche cachée est composée de plusieurs neurones qui effectuent des calculs en utilisant des poids et des fonctions d'activation. Les valeurs de sortie des neurones de la couche précédente sont propagées aux neurones de la couche suivante.

Couche de sortie : C'est la dernière couche du MLP, où les valeurs de sortie finales sont calculées. Le nombre de neurones dans cette couche dépend de la tâche spécifique, par exemple, un neurone pour une classification binaire ou plusieurs neurones pour une classification multi-classes.

Connexions pondérées : Chaque connexion entre les neurones est associée à un poids, qui détermine l'importance de cette connexion dans le calcul des valeurs de sortie. Les poids sont ajustés lors de l'apprentissage pour minimiser l'erreur entre les prédictions du modèle et les valeurs réelles.

Fonctions d'activation : Chaque neurone utilise une fonction d'activation pour introduire de la non-linéarité dans le modèle. Des fonctions d'activation couramment utilisées sont la fonction sigmoïde, la fonction ReLU (Rectified Linear Unit) ou la fonction tanh.

3.6 Outils, langage et bibliothèques de développement

3.6.1 Outils

Google Colab : Plateforme de développement en ligne qui permet d'exécuter du code Python et d'accéder à des ressources informatiques telles que le GPU et le TPU.

Pycharm : Un environnement de développement intégré (IDE) pour Python, offrant des fonctionnalités avancées pour la programmation et le débogage.

Visual Studio : Un environnement de développement complet qui prend en charge plusieurs langages de programmation et offre des fonctionnalités avancées pour la création d'applications.

LaTeX : Un système de composition de documents largement utilisé pour la création de documents scientifiques et techniques de haute qualité.

Canva : Un outil de conception graphique en ligne qui permet de créer des graphiques, des infographies et d'autres éléments visuels de manière intuitive.

3.6.2 Langage

Python : Python est un langage de programmation polyvalent et populaire, souvent utilisé dans le domaine de l'apprentissage automatique et de l'analyse des données. Il est connu pour sa syntaxe simple et claire, sa grande lisibilité du code et sa vaste collection de bibliothèques.

3.6.3 Bibliothèques

TensorFlow : Une bibliothèque open-source d'apprentissage automatique développée par Google, utilisée pour la création et l'entraînement de modèles d'apprentissage automatique, y compris les réseaux de neurones [40].

Keras : Une interface de haut niveau construite sur TensorFlow qui simplifie la création et l'entraînement des modèles d'apprentissage automatique, notamment les réseaux de neurones [40].

NumPy : Une bibliothèque Python qui prend en charge le calcul numérique avec des tableaux multidimensionnels, des fonctions mathématiques avancées et des outils pour manipuler les données [41].

Scikit-learn (sklearn) :Une bibliothèque d'apprentissage automatique très utilisée qui offre une large gamme d'algorithmes et d'outils pour la classification, la régression, le regroupement et la préparation des données [39].

Pandas : Une bibliothèque Python destinée à la manipulation et à l'analyse de données structurées, offrant des structures de données puissantes et des fonctionnalités pour l'exploration et la

transformation des données [42].

Ces outils, langage et bibliothèques de développement sont largement utilisés dans le domaine de l'apprentissage automatique et de l'analyse des données en raison de leur efficacité, de leur convivialité et de leur vaste communauté de soutien. Ils fournissent des fonctionnalités essentielles pour la mise en œuvre, l'analyse et la visualisation des modèles d'apprentissage automatique.

3.7 Métriques d'évaluation

Précision (Precision) : La précision mesure la proportion de résultats positifs corrects parmi les résultats positifs prédits par le modèle.

Précision = True Positives / (True Positives + False Positives)

Rappel (Recall) : Le rappel mesure la proportion de résultats positifs corrects identifiés parmi tous les résultats positifs réels.

Rappel = True Positives / (True Positives + False Negatives)

F1-score : Le F1-score est une métrique qui combine la précision et le rappel en une seule valeur pour évaluer l'équilibre entre les deux.

F1-score = 2 * (Precision * Recall) / (Precision + Recall)

Exactitude (Accuracy) : L'exactitude mesure la proportion globale de prédictions correctes du modèle parmi toutes les prédictions effectuées.

Accuracy = (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives)

Perte (Loss) :La perte est une mesure de l'erreur du modèle lors de l'entraînement. Elle quantifie à quel point les prédictions du modèle diffèrent des valeurs réelles. L'objectif de l'entraînement est de minimiser la perte afin d'améliorer les performances du modèle.

Définitions supplémentaires :

- Faux positif (False Positive) : Un faux positif se produit lorsqu'une classe négative est prédite comme positive.

- Vrai positif (True Positive) : Un vrai positif se produit lorsqu'une classe positive est prédite comme positive.

- Faux négatif (False Negative) : Un faux négatif se produit lorsqu'une classe positive est prédite comme négative.

- Vrai négatif (True Negative) : Un vrai négatif se produit lorsqu'une classe négative est prédite comme négative.

Ces métriques et définitions sont essentielles pour évaluer les performances d'un modèle et comprendre ses capacités de prédiction et sa capacité à distinguer les différentes classes.

3.8 Implémentation et entraînement

3.8.1 Implémentation du modèle LSTM

Dimension du jeu de données :

Pour le jeu de données Activityname_DataSet_A contient :
- Classe 0 : 442,103 instances.
- Classe 1 : 286,322 instances.
- Classe 2 : 341,815 instances.

Pour le jeu de données Activity_name_DataSet_B contient :
- Classe 0 : 15,375 instances.
- Classe 1 : 16,321 instances.
- Classe 2 : 32,755 instances.
- Classe 3 : 10,378 instances.
- Classe 4 : 10,519 instances.

Implémentation des caractéristiques et des cibles :

- Caractéristiques (FEATURES) : 'Sensors', 'Status', 'timestamp'.

- Cible (TARGET) : 'Activity_name'.

Division de l'ensemble de données :

Les données sont divisées de manière à utiliser 80 % pour l'entraînement et réserver les 20 % restants pour les tests.

Reformation des données

Les données sont préparées dans un format approprié pour l'entraînement du modèle LSTM.

```
[ ]  X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
     X_test= np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1 ))
```

FIGURE 3.9 – Scripte python pour réformation des données pour model LSTM

Importation du modèle LSTM : Le modèle LSTM est importé à partir de la bibliothèque Keras.

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense, LSTM, BatchNormalization, Embedding, Dropout

     model = Sequential()
     model.add(LSTM(units=200, return_sequences=True, input_shape=(X_train.shape[1],1)))
     model.add(LSTM(units=125, return_sequences=True))
     model.add(LSTM(units=75, return_sequences=True))
     model.add(LSTM(units=50))
     model.add(BatchNormalization())
     model.add(Dense(units=25,activation='relu'))
     model.add(Dropout(0.25))
     model.add(Dense(units=3,activation='softmax'))

     model.compile(loss='categorical_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

FIGURE 3.10 – Scripte python pour importer le modèle LSTM

Entraînement du modèle :
Lors de l'entraînement, les hyperparamètres suivants sont utilisés :

- Pour DataSet A :
epochs=10, batch_size=1280, validation_split=0.2

- Pour DataSet B :
epochs=10, batch_size=512, validation_split=0.2

Prédiction et évaluation du modèle :

Une fois le modèle lstm entraîné, il peut être utilisé pour effectuer des prédictions sur de nouvelles données. Ces métriques permettent de mesurer la performance du modèle sur les ensembles de données de test :

- Pour DataSet A :
Précision (accuracy) : 0.76
Perte (loss) : 0.58

- Pour DataSet B :
Précision (accuracy) : 0.84
Perte (loss) : 0.50

Ces résultats indiquent la performance du modèle LSTM sur les différents ensembles de données Activity_name_DataSet_A et Activity_name_DataSet_B.

3.8.2 Implémentation du modèle GRU

Les étapes suivantes sont similaires à celles de l'implémentation du modèle LSTM :
Indiquer la dimension du jeu de données.
Mise en place des caractéristiques et des cibles.
Division de l'ensemble de données.

Différences spécifiques à l'implémentation du modèle GRU :

Reformation des données :
Avant d'appliquer le modèle GRU, il est important de reformater les données pour les adapter à l'architecture du réseau. Cela inclure la conversion des données en tenseurs, le redimensionnement ou le remodelage des dimensions, selon les exigences du modèle GRU.

```
[ ]  X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
     X_test= np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1 ))
```

FIGURE 3.11 – Scripte python pour réformation des données pour model GRU

Importation du modèle GRU
À cette étape, le modèle GRU est importé depuis une bibliothèque Keras.

Entraînement du modèle :

L'entraînement du modèle GRU se fait en spécifiant les hyperparamètres tels que le nombre d'époques (epochs), la taille du batch (batch_size) et le pourcentage de validation (validation_split).

```
[ ]  from keras import metrics
     from keras.callbacks import EarlyStopping, ModelCheckpoint,ReduceLROnPlateau
     from keras.models import Sequential
     from keras.layers import Dense, GRU, BatchNormalization, Embedding, Dropout

     model = Sequential()
     model.add(GRU(units=200, return_sequences=True, input_shape=(X_train.shape[1],1)))
     model.add(GRU(units=125, return_sequences=True))
     model.add(GRU(units=75, return_sequences=True))
     model.add(GRU(units=50))
     model.add(Dense(units=5,activation='softmax'))

     model.compile(loss='categorical_crossentropy',
                   optimizer='adam',
                   metrics=[metrics.CategoricalAccuracy()])
     early_stopping_monitor = EarlyStopping(
         monitor='categorical_accuracy',
         min_delta=0,
         patience=3,
         verbose=0,
         mode='max',
         baseline=None,
         restore_best_weights=True
     )
```

FIGURE 3.12 – Scripte python pour importer le modèle GRU

Ces paramètres peuvent être ajustés en fonction des besoins spécifiques du problème.

- DataSet A :
epochs=10, batch_size=512, validation_split=0.2

- DataSet B :
epochs=10, batch_size=512, validation_split=0.2

Prédiction et évaluation du modèle :

Une fois le modèle GRU entraîné, il peut être utilisé pour effectuer des prédictions sur de nouvelles données.Ces métriques permettent de mesurer la performance du modèle sur les ensembles de données de test.
Voici les résultats obtenus pour les ensembles de données A et B après l'entraînement du modèle GRU :
DataSet A :
Exactitude (accuracy) : 0.76
Perte (loss) : 0.58
DataSet B :
Exactitude (accuracy) : 0.84
Perte (loss) : 0.50

Ces résultats fournissent une indication de la performance du modèle GRU sur les deux ensembles de données évalués.

3.8.3 Implémentation du modèle MLP

Les étapes initiales de l'implémentation du modèle MLP sont similaires à celles du modèle LSTM et GRU. Elles comprennent :
Indiquer la dimension du jeu de données
Implémentation des caractéristiques et des cibles
Diviser l'ensemble de données

Les étapes suivantes sont spécifiques à l'implémentation du modèle MLP :

Importer le modèle MLP :
À cette étape, le modèle MLP est importé depuis une bibliothèque Keras.

```
[ ]  from keras import metrics
     from keras.callbacks import EarlyStopping, ModelCheckpoint,ReduceLROnPlateau
     from keras.models import Sequential
     from keras.layers import Dense, BatchNormalization, Embedding, Dropout

     model = Sequential()
     model.add(Dense(200, input_shape=(X_train.shape[1],), activation="relu"))
     model.add(Dense(125, activation="relu"))
     model.add(Dense(75, activation="relu"))
     model.add(Dense(50, activation="relu"))
     model.add(Dense(3,activation='softmax'))

     model.compile(loss='categorical_crossentropy',
                   optimizer='adam',
                   metrics=[metrics.CategoricalAccuracy()])
     early_stopping_monitor = EarlyStopping(
         monitor='categorical_accuracy',
         min_delta=0,
         patience=3,
         verbose=0,
         mode='max',
         baseline=None,
         restore_best_weights=True
     )
```

FIGURE 3.13 – Scripte python pour importer le modèle MLP

Entraîner le modèle :
Pendant la phase d'entraînement, les hyperparamètres sont :

- DataSet A :
epochs=10, batch_size=512, validation_split=0.2

- DataSet B :
epochs=10, batch_size=512, validation_split=0.2

Prédiction et évaluation du modèle :

Une fois l'entraînement terminé, utilisez le modèle MLP pour effectuer des prédictions sur les ensembles de test et évaluer ses performances. Calculez l'exactitude (accuracy) du modèle pour chaque ensemble de données.
- DataSet A :
Exactitude (accuracy) = 0.41
- DataSet B :
Exactitude (accuracy) = 0.38

Ces étapes d'implémentation permettent de mettre en œuvre le modèle MLP, de l'entraîner sur les données d'entraînement, de l'évaluer sur les données de test et d'obtenir les performances du modèle en termes d'exactitude.

3.9 Résultat et Discussion

Les résultats obtenus après l'entraînement des trois modèles (LSTM, GRU et MLP) sur les ensembles de données A et B sont présentés ci-dessous :

DataSet A

Modèle	Précision	Recall	F1-Score	**Accuracy**	Loss
LSTM	0.76	0.77	0.77	**0.76**	0.58
GRU	0.76	0.77	0.77	**0.76**	0.58
MLP	0.14	0.33	0.19	**0.41**	-

TABLE 3.3 – Performances des modèles (LSTM, GRU, MLP) sur l'ensemble de données A

DataSet B

Modèle	Précision	Recall	F1-Score	**Accuracy**	Loss
LSTM	0.80	0.78	0.78	**0.84**	0.5
GRU	0.80	0.78	0.78	**0.84**	0.5
MLP	0.80	0.20	0.11	**0.38**	-

TABLE 3.4 – Performances des modèles (LSTM, GRU, MLP) sur l'ensemble de données B

Les performances des modèles LSTM et GRU semblent être similaires sur les deux datasets, avec des valeurs de précision, de rappel (recall) et de F1-score relativement élevées, ainsi qu'une précision globale (accuracy) satisfaisante. Les valeurs de perte (loss) sont également similaires pour ces deux modèles.

En revanche, le modèle MLP présente des performances nettement inférieures aux modèles récurrents (LSTM et GRU) sur les deux datasets. Les scores de précision, de rappel et de F1-score sont considérablement plus bas pour le modèle MLP, ce qui indique que ce modèle a du mal à capturer les relations temporelles et séquentielles présentes dans les données.

Comparant les performances entre les datasets A et B, on peut observer que les performances des modèles LSTM et GRU restent relativement stables, tandis que le modèle MLP montre une baisse significative des performances sur le dataset B.

En conclusion, les résultats suggèrent que les modèles LSTM et GRU sont plus fiables et adaptés pour les données des deux datasets, en raison de leur capacité à capturer les dépendances temporelles et séquentielles. Le modèle MLP, quant à lui, présente des limitations dans la compréhension de ces dépendances et affiche des performances inférieures. Par conséquent, pour les datasets A et B, les modèles LSTM et GRU sont préférables pour des tâches d'apprentissage automatique nécessitant une analyse temporelle précise.

3.10 Conclusion

En conclusion, lors de l'implémentation des modèles LSTM, GRU et MLP sur le dataset Aruba, les modèles LSTM et GRU ont démontré des performances similaires et supérieures par rapport au modèle MLP. Les modèles LSTM et GRU ont réussi à capturer les dépendances temporelles et séquentielles dans les données, ce qui leur a permis d'obtenir des résultats plus fiables. Ainsi, pour l'analyse du dataset Aruba, les modèles LSTM et GRU sont recommandés en raison de leur capacité à traiter efficacement les données temporelles.

Conclusion Générale et Perspectives

En conclusion, ce travail de recherche se concentre sur l'intelligence ambiante et la prédiction du comportement des utilisateurs. L'intelligence ambiante vise à intégrer l'informatique au cœur de nos activités quotidiennes, ouvrant ainsi la voie à de nombreuses applications dans des domaines tels que la domotique, la santé et les transports. La prédiction précise du comportement des utilisateurs est essentielle pour offrir des automatisations intelligentes et des services personnalisés.

Ce mémoire est composé de trois chapitres. Le premier chapitre donne une définition de l'intelligence ambiante et présente les fonctionnalités, applications et facteurs en jeu dans ce domaine. Le deuxième chapitre rappelle les concepts clés de l'IA et de l'apprentissage automatique, en mettant l'accent sur les réseaux de neurones et le Deep Learning. Le troisième chapitre présente nos contributions, notamment la conception et l'implémentation d'un modèle d'extraction de règles de prédiction.

Dans l'ensemble, ce travail de recherche offre une vue d'ensemble approfondie de la conception, de l'implémentation et des applications potentielles de l'intelligence ambiante, en mettant en évidence l'importance de la prédiction du comportement des utilisateurs dans ce domaine en évolution constante.

Perspectives

• Nous avons la possibilité d'examiner la façon dont nous pouvons rassembler des données provenant de diverses sources telles que des capteurs, des appareils connectés,etc., afin d'améliorer les prédictives de nos modèles LSTM, GRU et MLP. • Nous cherchons à améliorer l'exactitude de nos trois modèles (LSTM, GRU, MLP) en utilisant des techniques de prétraitement des données telles que l'augmentation des données, l'extraction de caractéristiques et la normalisation. De plus, nous envisageons d'appliquer des techniques d'optimisation des hyperparamètres.

Bibliographie

[1] M. Weiser. The Computer for the Twenty-First Century. Scientific American 265, 3, Septembre 1991.

[2] J. Coutaz, J. Crowley, Plan "Intelligence Ambiante" : Défis et Opportunités, Document de réflexion conjoint du comité d'experts « Informatique Ambiante » du département ST2I du CNRS et du Groupe de Travail « Intelligence Ambiante » du Groupe de Concertation Sectoriel (GCS3) du Ministère de l'Enseignement Supérieur et de la Recherche, DGRI A3. iihm.imag.fr/en/publication/CC08b/

[3] Das, G., Lin, K.¬I., Mannila, H., Renganathan, G., and Smyth, P. (1998). Rulediscoveryfrom time series. In Proceedings of the Fourth International Conference on KnowledgeDiscovery and Data Mining, KDD'98, pages 16–22, New York, NY. AAAI Press.

[4] Schlüter, T. and Conrad, S. (2011). About the analysis of time serieswith temporal association rulemining. In 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pages 325–332.

[5] Mannila, H., Toivonen, H., and InkeriVerkamo, A. (1997). Discovery of Frequent Episodes in Event Sequences. Data Mining and KnowledgeDiscovery, 1(3) :259–289.

[6] Deogun, J. and Jiang, L. (2005). PredictionMining – An Approach to Mining Association Rules for Prediction. In Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Lecture Notes in Computer Science, pages 98–108. Springer Berlin Heidelberg.

[7] mémoire de BENAYADA Yassmine. titre. master d'Université -ibn khaldoun.TIARET-2020.

[8] Plan « intelligence ambiante » : défis et opportunités Document de réflexion conjoint du comité d'experts « Informatique Ambiante » du département ST2I du CNRS et du Groupe de Travail « Intelligence Ambiante » du Groupe de Concertation Sectoriel (GCS3) du Ministère de l'Enseignement Supérieur et de la Recherche, DGRI A3 Version 1.2 finale - 14 octobre 2008.

[9] J.C. Augusto. Ambient intelligence : the confluence of ubiquitous/pervasive computing and artificial intelligence. Intelligent Computing Everywhere, pages 213–234, 2007. xiii, 4.

[10] https ://fr.slideshare.net/ShifaliJindal/ambient-intelligence-made-by-shifali-jindal

[11] https ://www.analyticsinsight.net/top-5-applications-of-ambient-intelligence-ami/

[12] https ://fr.wikipedia.org/wiki/Intelligence-ambiante

[13] J. Crowley, J. Coutaz, G. Rey, and P. Reignier. Perceptual components for contextaware computing. In UbiComp '02 : Proceedings of the 4th international conference on Ubiquitous Computing, London, UK, 2002. Springer-Verlag.

[14] G. Rey. Contexte en Interaction Homme-Machine : le contexteur. Ph.D. thesis, Fdration IMAG – University Joseph Fourier - Grenoble I, 2005.

[15] Workshop ArtificialSocieties for Ambient Intelligence, http ://asami07.cs.rhul.ac.uk/

[16] Mark Weiser, ubiquitous computing, 1988

[17] Thèse de Sébastien MAZAC. APPROCHE DÉCENTRALISÉE DE L'APPRENTISSAGE CONSTRUCTIVISTE ET MODÉLISATION MULTI-AGENT DU PROBLÈME D'AMORÇAGE DE L'APPRENTISSAGE SENSORIMOTEUR EN ENVIRONNEMENT CONTINU. APPLICATION À L'INTELLIGENCE AMBIANTE.Doctorat d'Université de Lyon.2015

[18] https ://www.zdnet.fr/actualites/informatique-ambiante-intelligence-ambiante-tout-ce-que-vous-devez-savoir-sur-l-essor-de-la-technologie-invisible-39947430.htm

[19] Turing, A.M. (1950) Computing Machinery and Intelligence. Mind, 59, 433-460.

[20] Cooper, S. B. et Van Leeuwen, J. (Eds.). (2013). Alan Turing : His work and impact. Elsevier

[21] Francois Chollet et al. Deep learningwith Python, volume 361. Manning New York, 2018.

[22] Géron, A. (2017). Hands-On_Machine_Learning_with_Scikit-learn and trensflow. USA : O'Reilly Media.

[23] Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nas- rin, M. S., Van Esesn, B. C., Awwal, A. A. S., and Asari, V. K. (2018). The historybe- gan fromalexnet : A comprehensivesurvey on deeplearningapproaches. arXivpreprintarXiv :1803.01164.

[24] Kaelbling LP, Littman ML, Moore AW. Reinforcementlearning : a survey. J ArtifIntellRes. 1996,4 :237– 85.

[25] Dave Anderson and George McNeill. Artificial neural networks technology. Kaman Sciences Corporation, 258(6) :1-83, 1992.

[26] Claude Touzet, "Les reseaux de neurones artificiels, introduction au connexionnisme", 1992, Collection de l'EERIE, N. Giambiasi. hal-01338010

[27] Youcef Djeriri. Les Réseaux de Neurones Artificiels, 20 September 2017.

[28] Nikhil Buduma and Nicholas Locascio. Fundamentals of deeplearning : Designingnext-generation machine intelligence algorithms. " O'Reilly Media, Inc.", 2017.

[29] Chris Woodfordx. All rightsreserved, EXPLAINTHATSTUFF !,2011, 2020, disponible à l'adresse : https ://www.explainthatstuff.com/introduction-to-neural-networks.html Consulté : 03/11/2020.

[30] All RightsReserved, TUTORIALS POINT, 2020, disponible à l'adresse : https ://www.tutorialspoint.com/tensorflow/tensorflow single layer perc eptron.htm. Consulté : 20 mars 2020.

[31] RiccoRakotomalala. Perceptrons simples et multicouches, Université Lumière Lyon 2. Consulté : 21 mars 2020.

[32] Céline Deluzarche- Journaliste .intelligence artificielle et deeplearning, https ://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning17262.

[33] Benoît Virole. Réseaux de neurones et psychométrie. Editions du Centre de Psychologie Appliquée-ECPA, 2001.

[34] Aidlahcene, Djenane Mouloud Amine, Dehbi Abdelkader. Predicting life time and mechanical performance degradation of multilayerGreenhouse. ISSN 1560-0904, Polymer Science, Series B,2021.

[35] Dan ClaudiuCiresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In Twenty-second international joint conference on artificial intelligence, 2011

[36] Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9, 1735–1780.

[37] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv :1406.1078.

[38] https ://medium.com/@nickmal/real-time-activity-recognition-in-a-smart-home-using-binary-sensors-efd147ec694

[39] https ://scikit-learn.org/stable/index.html

[40] https ://www.tensorflow.org/ ?hl=fr

[41] https ://numpy.org/

[42] https ://pandas.pydata.org/