République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT : ELECTRONIQUE

Mémoire de fin d'études de Master Académique

Domaine : **Sciences et Technologie**

Filière : **Génie Electrique**

Spécialité : **Electronique des Systèmes Embarqués**

Présenté par : Mme . MALKI Fatima

Thème

Conception et évaluation d'un système d'IA embarqué basé sur les techniques d'apprentissage pour la carte NVIDIA Jetson Nano

Mémoire soutenu publiquement le /06/2024 devant le jury composé de :

Président :
M. H. ACHOUR
Professeur, Université MOULOUD MAMMERI de Tizi Ouzou,
Encadreur :

M. GANA Massine

Maître de conférences classe « B », Université BOUMERDES,
Examinateur :
M. Y. TRIKI
Maître de conférences classe « B », Université BOUMERDES,

Promotion : 2023/2024

Nous remercions tout d'abord Dieu tout puissant de nous avoir accordé la santé nécessaire pour mener à bien ce travail. C'est grâce à sa bienveillance que nous avons pu accomplir nos tâches avec diligence.

J'exprime ma profonde gratitude et mes sincères remerciements à mon promoteur, M. Gana Massine, pour son sérieux, sa rigueur, et son aide qui n'ont d'égale que ses connaissances et sa passion pour son métier.

Je tiens à remercier chaleureusement le président ainsi que tous les membres du jury, d'avoir honoré cette soutenance.

Nous remercions également M. ACHOUR, notre chef de département ainsi que tous les enseignants de la Faculté de Génie Electrique et Informatique pour leurs enseignements, leurs encouragements et leurs bienveillances.

J'exprime ma profonde gratitude envers mes parents, mon mari, mes enfants : Sarah, Yani , et Ilyane pour leur soutien indéfectible et leurs encouragements. Leur amour et leur confiance en nous ont été une source d'inspiration et de motivation inépuisable

Je remercie tous les camarades surtout Mme Gana Zakia et amis qui m'ont apporté une aide précieuse

Pour finir, je souhaite remercier toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

L'objectif de l'implémentation des réseaux de neurones et des SVM sur la Nvidia Jetson Nano est de tirer parti de ses capacités de calcul pour développer des solutions de machine learning embarquées et efficaces. Cela permet de réaliser des tâches complexes d'intelligence artificielle en temps réel, directement sur un dispositif compact et économe en énergie.

Pour ce faire, nous avons choisi d'utiliser un algorithme nommé réseaux de neuroneset l'autre Support Vector Machines (SVM).

Nous avons mené des tests pour évaluer les performances de chaque modèle et identifier celui qui répond le mieux à nos besoins. Les résultats obtenus avec les deux algorithmes se sont révélés très satisfaisants, démontrant que cette application permet de réaliser n'importe quel type d'apprentissage en ligne sur n'importe quelle base de données.

Mots _clés : les réseaux de neurones(ANN), les Support Vector Machines (SVM), la Nvidia Jetson Nano, Apprentissage Automatique.

The aim of implementing neural networks and SVMs on the Nvidia Jetson Nano is to take advantage of its computing capabilities to develop efficient embedded machine learning solutions. This makes it possible to perform complex artificial intelligence tasks in real time, directly on a compact, energy-efficient device.

To achieve this, we have chosen to use one algorithm called Neural Networks and the other Support Vector Machines (SVM).

We carried out tests to evaluate the performance of each model and identify the one best suited to our needs. The results obtained with both algorithms proved highly satisfactory, demonstrating that this application can perform any type of online learning on any database.
Keywords : neural networks (ANN), Support Vector Machines (SVM), Nvidia Jetson Nano, Machine Learning

IA : L'intelligence Artificielle

IAE : L'Intelligence Artificielle Embarquée

ANN : réseaux de neurones artificiels

CNN : Les réseaux de neurones convolutifs

RNN : Les réseaux de neurones récurrents

CPU : Central Processing Unit

GPU : Graphics Processing Unit ou Unité de Traitement Graphique

FPP : Full Parallel Processing

FPGA : Field Programmable Gate Arrays ou réseau de portes programmables sur site

DSP : Digital Signal Processor ou processeur de signal numérique

ASIC : Application-Specific Integrated Circuits ou des circuits intégrés personnalisés

FPP : Full Parallel Processing ou le calcul parallèle complet

SVM : Les "Support Vector Machines" ou Séparateurs à Vaste Marge

ReLU : Rectified Linear Unit ou Unité Linéaire Rectifiée

Table des matières

INTRODUCTION GENERALE

Depuis toujours, la curiosité de l'homme a été le moteur de la science, qui s'est révélée être l'un de ses plus grands bienfaits. Des figures emblématiques comme Isaac Newton et Einstein ont su nourrir cette soif de connaissance, en se posant des questions essentielles sur le monde qui les entourait. Aujourd'hui, cette même curiosité pousse les scientifiques à explorer de nouveaux horizons, notamment dans le domaine de l'intelligence artificielle.

Bien que certains grands penseurs comme Bill Gates et Stephen Hawking aient exprimé des craintes quant aux dangers potentiels de l'IA sur l'essence même de l'humanité, d'autres, à l'instar de Joël de Rosnay, considèrent qu'il faut davantage redouter "la stupidité naturelle" que les avancées de l'IA. Dans cette perspective, l'éducation, la formation et l'apprentissage continu apparaissent comme des éléments clés pour apprivoiser cette nouvelle technologie et en faire profiter l'humanité[1].

L'homme continue d'être guidé par sa curiosité, le poussant à explorer de nouvelles pistes de réflexion. Ainsi, il s'est inspiré du comportement des fourmis pour développer l'algorithme de colonie de fourmis, dans l'optique de résoudre des problèmes complexes. De même, l'ambition de créer des machines capables de penser comme les humains a conduit à l'invention des réseaux de neurones artificiels (ANNs). Le présent travail se concentre précisément sur la création, la manipulation et l'utilisation de ces réseaux de neurones artificiels, les SVMs implémentés sur Le carte NVIDIA Jetson.

Le cerveau humain a pour fonction essentielle de transformer les informations sensorielles en le connaissances organisées. Qu'il s'agisse d'identifier des lettres, de les assembler en mots et en phrases, ou d'en extraire un sens, ces activités nous semblent naturelles une fois que l'apprentissage nécessaire a été accompli. Les réseaux de neurones artificiels et les SVM visent à imiter cette capacité d'apprentissage par l'exemple, propre aux êtres vivants. Depuis quelques années, ces réseaux ont affiché des résultats remarquables dans des domaines très variés, de la vision par ordinateur à la médecine. Au-delà de leurs aspects techniques et applicatifs, ces systèmes inspirés des réseaux neuronaux biologiques, notamment ceux du cerveau humain, soulèvent des interrogations passionnantes sur nos propres capacités cognitives et sur la nature de l'intelligence artificielle.

Les Réseaux de neurones et les SVM offrent des solutions plus que satisfaisantes pour un grand nombre de problèmes, dont certains sont difficiles à traiter par les approches classiques (analytiques, numériques…)

L'implémentation des ANNs et les SVMs sont aujourd'hui beaucoup plus simple a mettre en œuvre vu l'avancée fulgurante de la technologie et des systèmes informatiques.

Les cartes NVIDIA Jetson, comme la Jetson Nano et la Jetson Xavier, sont des plateformes puissantes pour l'IA embarquée, idéales pour des tâches de vision par ordinateur et de robotique. Pour implémenter des réseaux de neurones (ANN) et des machines à vecteurs de support (SVM) sur une carte Jetson, nous commençons par installer des bibliothèques comme TensorFlow, PyTorch, et scikit-learn, ainsi que les outils NVIDIA tels que CUDA et cuDNN. Les réseaux de neurones peuvent être optimisés en utilisant des techniques comme la quantification, le pruning, et la distillation de connaissances, ainsi qu'en exploitant TensorRT pour l'inférence haute performance. Pour les SVM, l'optimisation des hyperparamètres, le choix du kernel, et la sélection de caractéristiques sont essentiels pour améliorer l'efficacité et la performance. En optimisant ces algorithmes en fonction des ressources limitées des dispositifs embarqués [2].

Pour cela, on a divisé notre travail en trois grandes parties :

- La première étant purement théorique, et se composera de l'apprentissage automatique et son rôle croissant dans divers domaines technologiques, soulignant la façon dont les machines évoluent et s'adaptent pour répondre aux défis contemporains.
- Dans la deuxième partie toujours dans l'aspect théorique, on décrira de manière simple et complète les différentes méthodes d'apprentissage en IA, notamment les réseaux de neurones et les (SVM).
- La troisième partie sera consacrée à la dimension pratique de notre travail. Nous y détaillerons les différentes étapes et les progrès successifs de nos résultats. Parallèlement, nous identifierons les défis et les difficultés rencontrés, et exposerons les solutions mises en place pour atteindre nos objectifs.

CHAPITRE I :
FONDEMENTS DE L'INTELLIGENCE ARTIFICIELLE

I.1 Introduction

L'intelligence artificielle (IA) est un domaine vaste et dynamique, englobant des disciplines telles que le traitement d'images, le traitement du langage naturel et les bases de données. Au cœur de l'IA se trouve l'apprentissage automatique, une méthode permettant aux machines d'apprendre à partir de données sans programmation explicite. Ce chapitre explore les bases de l'apprentissage automatique et son rôle croissant dans divers domaines technologiques, soulignant la façon dont les machines évoluent et s'adaptent pour répondre aux défis contemporains.

I.2 Définition de l'Intelligence Artificielle

L'Intelligence Artificielle (IA) se réfère à un domaine informatique qui cherche à développer des systèmes capables d'accomplir des tâches traditionnellement associées à l'intelligence humaine. Cela inclut des capacités telles que la reconnaissance de la parole, la vision par ordinateur, la prise de décision, et bien d'autres. Les techniques sous-jacentes à l'IA reposent sur l'utilisation de modèles mathématiques, statistiques et l'apprentissage à partir de données pour permettre aux systèmes de s'améliorer et de prendre des décisions autonomes [3].

John McCarthy, qui est l'un des pionniers de l'IA l'a défini comme "l'art de créer des machines intelligentes"[4].

L'Académie française la définit comme un "Ensemble de techniques informatiques qui visent à créer des machines capables de simuler l'intelligence humaine, notamment la faculté de raisonner, d'apprendre et de résoudre des problèmes" [5].

Selon Larousse, c'est une "Science et technique qui visent à créer des machines capables d'imiter les fonctions cognitives de l'homme, telles que l'apprentissage, la résolution de problèmes et la prise de décision"[6] .

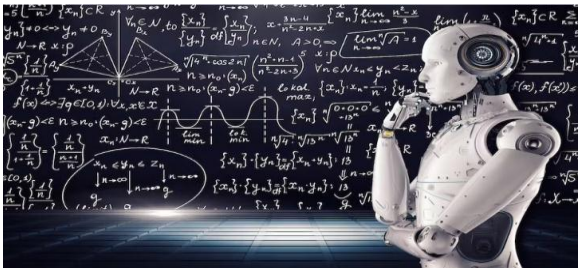


Figure I. 1 : Image sur l'Intelligence Artificielle

I.3 Histoire de l'Intelligence Artificielle

L'intelligence artificielle (IA) est un domaine fascinant qui a connu une évolution fulgurante. Cependant, son histoire remonte à des siècles avant l'ère numérique, avec des mythes et des légendes racontant la création d'êtres artificiels dotés d'intelligence, tels que le Golem de la tradition juive ou Pygmalion de la mythologie grecque. Ces récits illustrent le rêve humain de créer des machines à l'image de son propre esprit.

Le véritable essor de l'IA débute au XXe siècle, avec des progrès notables dans des domaines comme le traitement du langage naturel, la résolution de problèmes et les jeux. Cependant, cet enthousiasme initial est suivi d'un "hiver de l'IA" dans les années 1970 et 1980, en raison des limitations techniques et de la difficulté à reproduire l'intelligence humaine dans des machines.

L'arrivée de nouvelles technologies, telles que l'apprentissage automatique et les réseaux de neurones profonds, a permis à l'IA de connaître une renaissance à partir des années 1990, donnant naissance à des applications révolutionnaires dans de nombreux domaines.

Aujourd'hui, l'IA est omniprésente dans notre monde, transformant nos modes de vie et de travail. De la reconnaissance faciale aux assistants vocaux en passant par les véhicules autonomes, l'IA est devenue un élément incontournable de la société moderne. Le futur de l'IA est prometteur, avec des potentialités extraordinaires dans des domaines tels que la santé, l'éducation, l'exploration spatiale et la lutte contre le changement climatique [7] .

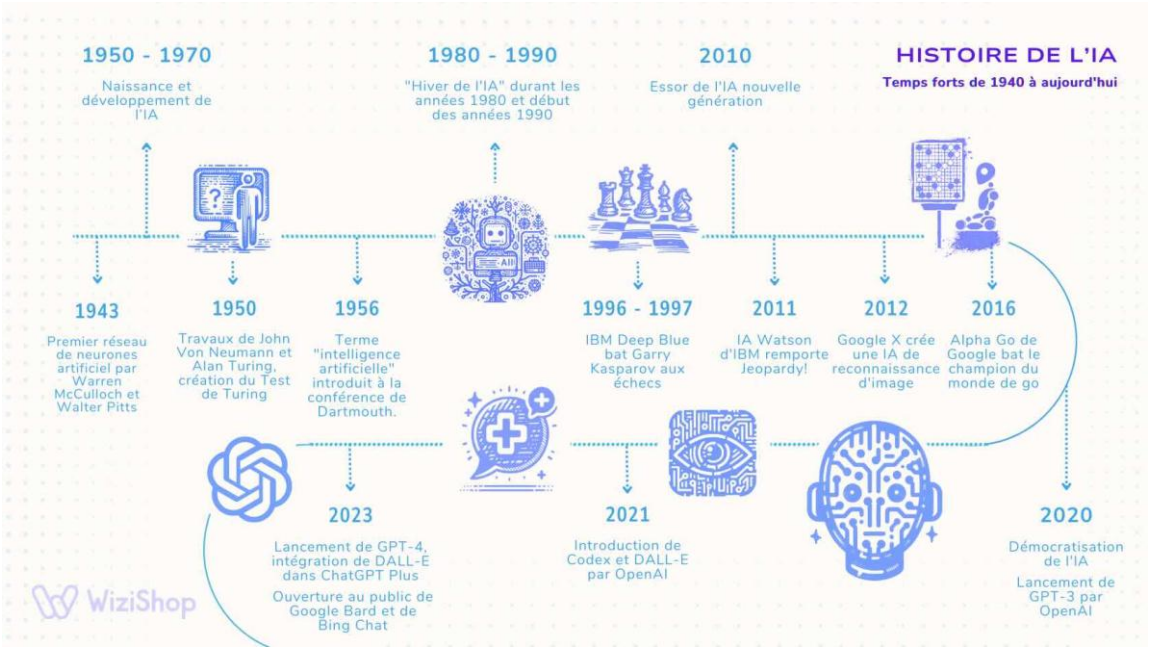


Figure I.2 : Histoire de l'Intelligence Artificielle

l.4 Domaines de l'intelligence Artificielle (IA)

L'intelligence artificielle (IA) est un domaine de l'informatique qui se concentre sur la création de machines capables d'exécuter des tâches qui nécessitent normalement l'intelligence humaine. Cela inclut des domaines tels que l'apprentissage automatique, le traitement du langage naturel, la vision par ordinateur, la robotique et la prise de décision automatisée. L'IA est utilisée dans de nombreux domaines, y compris la médecine, la finance, les transports, la fabrication et les services publics. Voici quelques domaines d'application :

I.4. 1 Apprentissage Automatique (Machine Learning)

Il s'agit d'une branche de l'IA qui se concentre sur la création de modèles et d'algorithmes capables d'apprendre à partir de données. Ces modèles peuvent être utilisés pour effectuer des prévisions, détecter des schémas ou prendre des décisions, et sont largement utilisés dans des applications telles que la recommandation de produits, la détection de fraude, la reconnaissance vocale et la traduction automatique[3].

I.4. 2 Traitement du langage naturel

Ce domaine vise à permettre aux machines de comprendre et de communiquer en utilisant un langage humain naturel. Les applications incluent les chatbots, les systèmes de traduction automatique, l'analyse de sentiment, la génération de texte automatique et la compréhension de documents.

I.4. 3 Vision par ordinateur

La vision par ordinateur consiste à permettre aux machines de comprendre, analyser et interpréter des images et des vidéos. Cela inclut des tâches telles que la reconnaissance faciale, la détection d'objets, la navigation autonome des véhicules, la surveillance vidéo et la réalité augmentée.

I.4. 4 Robotique

L'IA est utilisée pour développer des robots capables d'accomplir des tâches variées, allant de la production industrielle à l'assistance à la personne. Les robots utilisant l'IA peuvent apprendre de nouvelles tâches, s'adapter à des environnements changeants et interagir de manière plus naturelle avec les humains.

I.4. 5 Systèmes experts

Ces systèmes utilisent des bases de connaissances et des règles pour imiter le raisonnement humain dans des domaines spécifiques. Ils sont utilisés dans des applications telles que le diagnostic médical, la gestion des risques financiers et le support technique.

I.4. 6 Prise de décision automatisée

L'IA est utilisée pour développer des systèmes capables de prendre des décisions complexes en se basant sur des données et des modèles. Cela peut inclure des applications telles que la planification logistique, l'optimisation des ressources et la gestion des stocks.

Ces domaines représentent une fraction des applications de l'intelligence artificielle, qui continue de se développer et de s'étendre à de nouveaux domaines d'application.

I.5 Structure d'un système d'IA

Un système d'intelligence artificielle (IA) peut être structuré de différentes manières en fonction de son application spécifique, mais voici une structure générale qui est couramment utilisée pour de nombreux systèmes d'IA :

I.5.1 Collecte de données

Tout système d'IA nécessite des données pour fonctionner. Ces données peuvent être de différents types, tels que des images, des textes, des vidéos, des signaux, des données structurées ou non structurées. La collecte de données peut se faire à partir de sources variées, y compris des capteurs, des bases de données, des réseaux sociaux, des sites web, etc.

I.5.2 Prétraitement des données

Avant que les données ne puissent être utilisées par un système d'IA, elles doivent souvent être nettoyées, transformées et préparées. Cela peut inclure des étapes telles que la normalisation, la réduction de dimension, le filtrage, la tokenisation (dans le cas du traitement du langage naturel), etc.

I.5.3 Modélisation et apprentissage

Une fois que les données ont été préparées, elles sont utilisées pour former des modèles d'IA. Ces modèles peuvent être basés sur des algorithmes d'apprentissage automatique, des réseaux de neurones, des systèmes experts, etc. L'entraînement des modèles implique de les exposer aux données pour qu'ils puissent apprendre à effectuer des tâches spécifiques.

I.5.4 Évaluation et validation

Les modèles d'IA doivent être évalués pour s'assurer qu'ils fonctionnent correctement. Cela peut impliquer la division des données en ensembles d'entraînement et de test, la validation croisée, la mesure de différentes métriques de performance (précision, rappel, F1-score, etc.) et la comparaison avec des modèles existants.

I.5.5 Déploiement

Une fois qu'un modèle d'IA a été entraîné et validé, il peut être déployé dans un environnement de production pour effectuer des tâches réelles. Cela peut impliquer l'intégration du modèle dans une application logicielle, un système embarqué, un robot, etc.

I.5.6 Surveillance et maintenance

Les systèmes d'IA déployés nécessitent une surveillance continue pour s'assurer qu'ils fonctionnent correctement. Cela peut inclure la détection des défaillances, la mise à jour des modèles, l'optimisation des performances, etc**.**

Il est important de noter que cette structure générale peut varier en fonction de la complexité et de la nature spécifique du système d'IA, mais elle fournit une base pour comprendre les étapes clés impliquées dans la construction et le déploiement de systèmes d'IA.

I.6 L'Intelligence Artificielle Embarquée (IAE)

I.6. 1 Définition

L'IA embarquée est une technologie qui consiste à intégrer des capacités d'intelligence artificielle (les capacités d'évaluer ses propres opérations) directement dans des dispositifs électroniques tels que des ordinateurs, des capteurs, des robots ou encore des véhicules.

Contrairement à l'IA traditionnelle, qui fonctionne principalement sur des serveurs distants, l'IA embarquée permet aux dispositifs électroniques de prendre des décisions en temps réel, sans avoir besoin d'une connexion Internet constante. Cette capacité de traitement en temps réel permet non seulement une prise de décision plus rapide et efficace, mais également une anonymisation des données des utilisateurs sans avoir à les partager sur des serveurs externes. Ces capacités uniques permettent des gains de productivité importants dans de nombreux domaine [8].

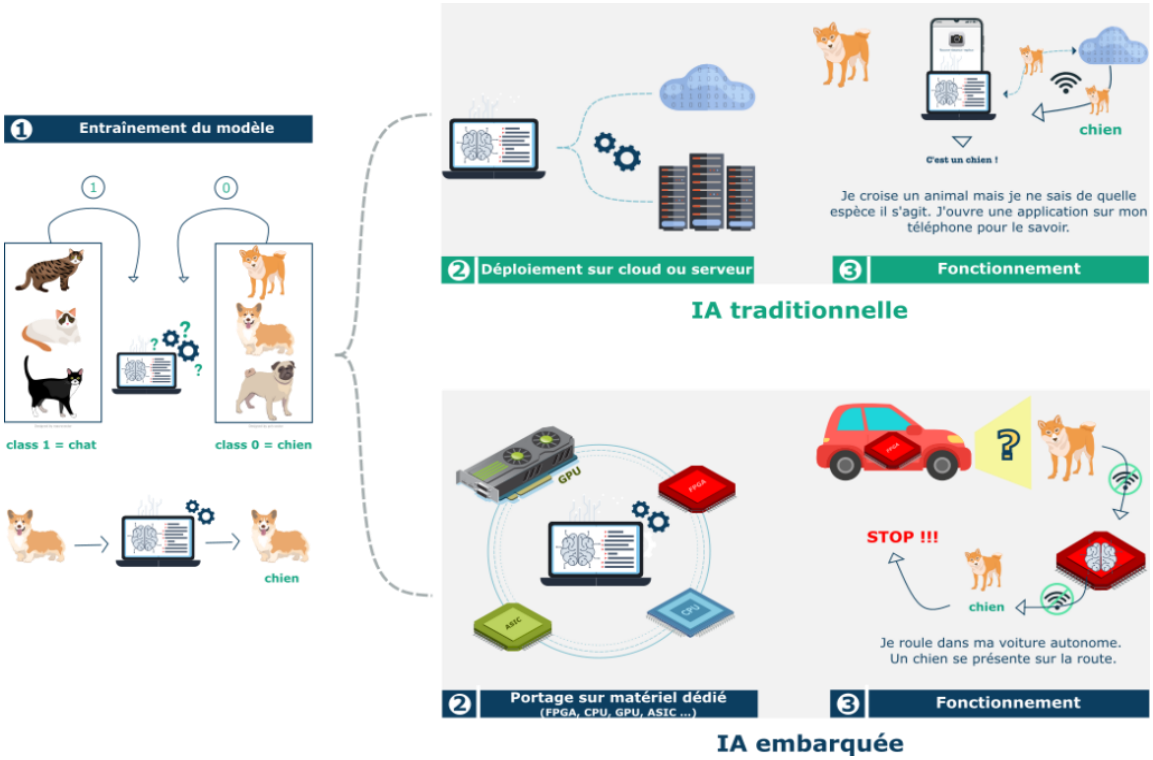


Figure I.3 : l'IA traditionnelle et l'IA Embarquée

I.6. 2 Fondement de l'IA Embarquée

L'IA embarquée repose sur plusieurs composants clés, notamment :

I.6.2.1 Les Capteurs

Les capteurs sont les "yeux" et les "oreilles" des systèmes d'IA embarqués. Ils collectent des données à partir de l'environnement, telles que des images, du son, des températures, et bien d'autres, fournissant ainsi des informations essentielles pour la prise de décision.

I.6.2.2 Les Microprocesseurs

Les microprocesseurs sont le cerveau de l'IA embarquée. Ils traitent les données collectées par les capteurs et exécutent des algorithmes d'IA pour prendre des décisions en temps réel.

I.6.2.3 Le Logiciel Embarqué

Le logiciel embarqué comprend les algorithmes d'IA, les bibliothèques de machine learning, et les codes qui permettent à l'IA embarquée de fonctionner. Il est spécialement conçu pour être exécuté sur des systèmes embarqués avec des ressources limitées.

I.6.2.4 Les Interfaces Utilisateurs

Pour les systèmes IA qui nécessitent une interaction avec les utilisateurs, des interfaces utilisateur adaptées sont conçues pour permettre une communication efficace.

I.6. 3 Applications de l'IA Embarquée

L'IA embarquée a des applications diverses et passionnantes dans de nombreux domaines. Parmi les exemples notables, citons :

I.6.3.1 La Robotique

Les robots équipés d'IA embarquée peuvent naviguer dans des environnements complexes, prendre des décisions en temps réel et accomplir des tâches variées tels que :

- **Les Robots Intelligents**

Les robots intelligents embarqués sont équipés de capteurs avancés et de systèmes d'IAE qui leur permettent de percevoir leur environnement, de prendre des décisions en temps réel et d'interagir de manière adaptative avec les objets et les humains. Ces robots trouvent des applications dans de nombreux domaines, de la fabrication à la médecine en passant par l'exploration spatiale.

- **Robotique Autonome**

L'IA embarquée a permis le développement de robots autonomes capables de fonctionner sans intervention humaine dans des environnements variés. Par exemple, les robots autonomes peuvent être utilisés pour l'inspection de pipelines, la collecte de données environnementales ou même la livraison de marchandises.

Figure I.4 : Image d'un robot autonome

- **Robotique Médicale**

Dans le domaine de la médecine, l'IAE est utilisée pour créer des robots chirurgicaux de haute précision. Ces robots assistent les chirurgiens lors d'interventions complexes, améliorant la précision et la sécurité des procédures médicales.

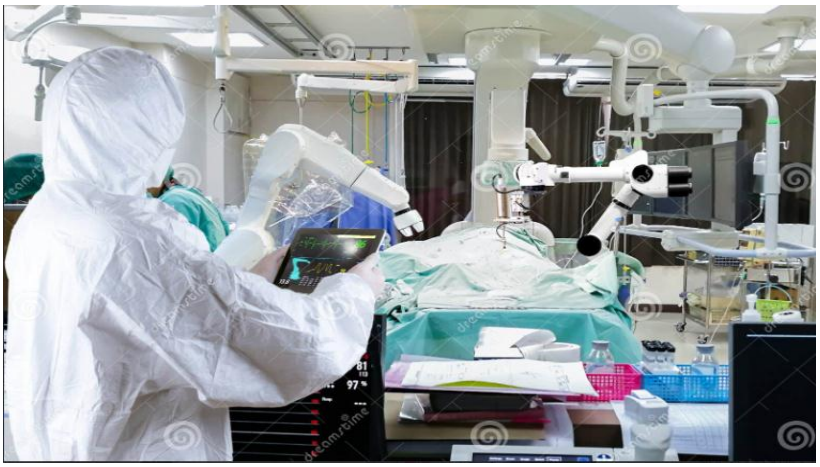


Figure I.5 : Image d'un robot médical

I.6.3.2 L'Aéronautique

Les aéronefs modernes intègrent des systèmes d'IA embarquée pour améliorer la sécurité, la navigation et la gestion des données en vol.

- **L'IAE dans la Conception Aéronautique**

L'IAE a profondément transformé le processus de conception des aéronefs. Les systèmes d'IAE avancés sont utilisés pour optimiser la conception des avions, en prenant en compte des facteurs tels que la résistance aérodynamique, la consommation de carburant et la sécurité. Les simulations informatiques assistées par l'IAE permettent aux ingénieurs de tester virtuellement de multiples configurations et de sélectionner la plus efficace.

Figure I.6 : Image de l'IAE dans l'aéronautique

I.6.3.3 Automobile

Les véhicules autonomes reposent sur des systèmes électroniques intelligents pour la perception, la navigation et la prise de décision. L'intelligence artificielle embarquée contribue à rendre la conduite plus sûre et efficiente.

- **Véhicules Autonomes**

Les véhicules autonomes sont l'un des domaines les plus visibles de l'IA. Ces voitures, camions et drones sont équipés de capteurs avancés, de systèmes de perception et de décision, et ils peuvent naviguer sans conducteur humain. L'IA est essentielle pour la cartographie, la planification des itinéraires, la reconnaissance d'obstacles et la prise de décision en temps réel.

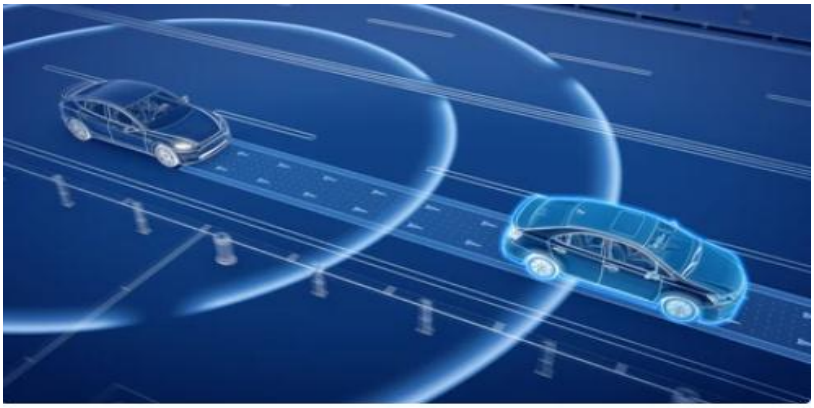


Figure I.7 : Image de véhicules autonomes

I.6.3.4 Drones Autonomes

Les drones autonomes sont largement utilisés dans des domaines tels que la surveillance, la cartographie, l'agriculture de précision et la livraison de colis. L'IA permet aux drones de voler de manière autonome, d'effectuer des missions complexes et d'interagir avec leur environnement de manière intelligente.

Figure I.8: Image d'un drone autonome

I.6.3.5 L'Internet des Objets (IOT)

Les objets connectés, tels que les thermostats intelligents et les caméras de sécurité, utilisent l'IA embarquée pour automatiser les processus et améliorer l'efficacité énergétique.

I.6. 4 Compétences des développeurs de l'IAE

Les développeurs de logiciels embarqués dotés de compétences en intelligence artificielle sont également en demande croissante. Leur rôle consiste à créer les logiciels qui alimentent les systèmes embarqués intelligents.

Voici quelques compétences clés pour les développeurs dans ce domaine :

I.6.4.1 Programmation orientée embarquée

Les développeurs doivent être capables de programmer pour des environnements embarqués, en tenant compte des contraintes matérielles.

I.6.4.2 Machine Learning

Comprendre les concepts de machine learning, y compris les réseaux de neurones, est essentiel pour développer des systèmes d'IA.

I.6.4.3 Analyse de données

Les développeurs doivent être capables de traiter et d'analyser de grandes quantités de données pour former des modèles d'IA.

I.7 Intelligence artificielle vs Machine Learninig vs Deep Learninig

I.7. 1 Le Machine Learning

Le machine Learning, ou apprentissage automatique, est une branche de l'intelligence artificielle qui se concentre sur la conception, le développement et l'utilisation de systèmes capables d'apprendre et de s'améliorer automatiquement à partir de données. Le but est de permettre aux ordinateurs d'identifier des modèles dans les données et de prendre des décisions en se basant sur ces modèles sans avoir été explicitement programmés pour chaque tâche. Le machine Learning est utilisé dans de nombreuses applications, telles que la reconnaissance d'images, la recommandation de produits, la prédiction de la demande et la détection de fraudes. L'apprentissage automatique peut être appliqué à différents types de données, tels des graphes, des arbres, des courbes, ou plus simplement des vecteurs de caractéristiques, qui peuvent être des variables qualitatives ou quantitatives continues ou discrètes [9].

La figure suivante illustre l'IA et l'apprentissage automatique :

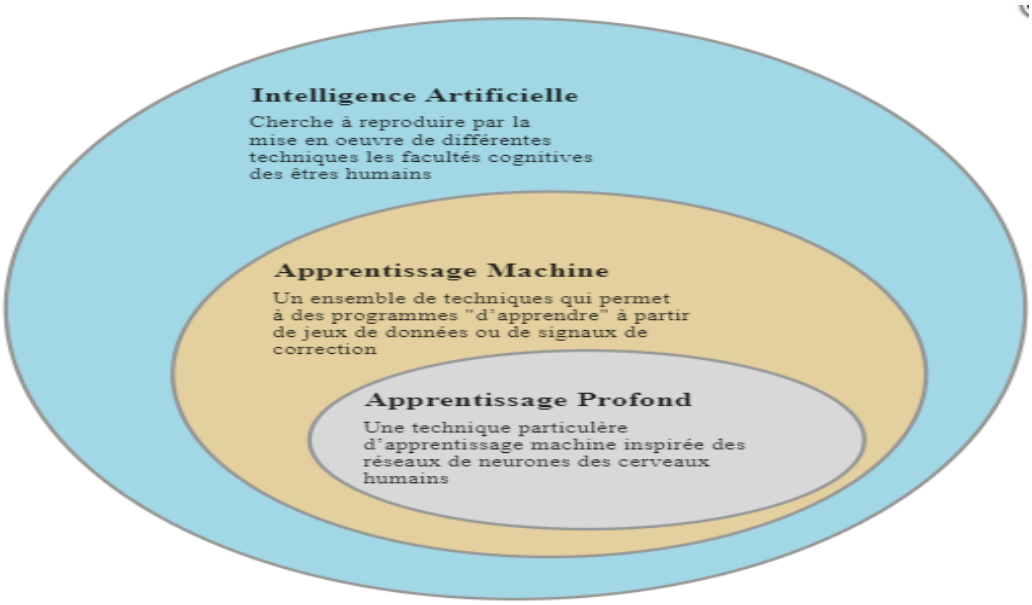


Figure I.9 : l'Intelligence Artificielle et Apprentissage Machine

I.7. 2 Le Deep Learning

Le Deep Learning, ou apprentissage profond, est une branche du machine Learning qui se concentre sur l'utilisation de réseaux de neurones artificiels profonds pour apprendre à partir de données complexes. Les réseaux de neurones profonds sont constitués de couches de neurones qui peuvent apprendre à détecter des caractéristiques de plus en plus abstraites à mesure que les données passent à travers les différentes couches. Les réseaux de neurones profonds sont utilisés dans de nombreuses applications, telles que la reconnaissance de la parole, la traduction automatique, la reconnaissance d'images et la classification de données complexes. Le Deep Learning est considéré comme une avancée majeure dans l'intelligence artificielle et est en train de révolutionner de nombreux domaines, notamment la santé, la finance et les sciences [9].

I.8 Principales techniques liées à l'intelligence Artificielle Embarqué

L'intelligence artificielle embarquée (IA embarquée) intègre des capacités d'apprentissage automatique et de traitement intelligent directement dans les systèmes embarqués. Cette technologie révolutionne de nombreux domaines en offrant une prise de décision autonome et en temps réel, une meilleure efficacité et une réduction de la consommation énergétique. Voici quelques-unes des principales techniques liées à l'IA embarquée :

I.8. 1 Apprentissage Automatique Embarqué

L'apprentissage automatique embarqué (ou Embedded Machine Learning) désigne l'implémentation de techniques d'apprentissage automatique directement sur des dispositifs matériels embarqués, tels que des microcontrôleurs, des capteurs intelligents, des appareils IoT (Internet des objets), et d'autres systèmes électroniques autonomes. Contrairement aux systèmes traditionnels où les données sont envoyées à un serveur central pour traitement, l'apprentissage automatique embarqué permet d'effectuer les calculs et les prédictions directement sur le dispositif [3].

I.8.2 Réseaux de neurones profonds (DNN)

Utilisés pour la reconnaissance d'images, la classification d'objets, la reconnaissance vocale et le traitement du langage naturel.

- **Réseaux convolutifs profonds (CNN)**

Particulièrement adaptés à la vision artificielle et à l'analyse d'images.

- **Réseaux neuronaux récurrents (RNN)**

Efficaces pour traiter les séquences temporelles, comme la parole ou les données de capteurs.

I.8. 4 Matériel embarqué

- **Processeurs d'apprentissage automatique (ML)**

Conçus pour optimiser les performances et l'efficacité énergétique des applications d'IA embarquée.

- **Unités de traitement de bord (Edge TPU)**

Modules spécialisés pour l'IA embarquée, offrant une accélération matérielle pour les modèles d'apprentissage automatique.

I.8.5 Systèmes d'exploitation embarqués

Doivent être adaptés pour prendre en charge les exigences spécifiques de l'IA embarquée, comme la gestion de la mémoire et la sécurité.

- **Linux embarqué**

Système d'exploitation largement utilisé pour les systèmes embarqués, offrant une grande flexibilité et une large communauté de développeurs.

I.8.6 Bibliothèques et frameworks d'IA

- **TensorFlow Lite**

Bibliothèque open source pour l'apprentissage automatique embarquée, optimisée pour les appareils mobiles et IoT.

- **ONNX Runtime**

Plateforme open source pour l'exécution de modèles d'apprentissage automatique sur différents types de matériel.

I.8.7 Outils de développement

- **Kits de développement logiciel (SDK)**

Fournissent des outils et des bibliothèques pour simplifier le développement d'applications d'IA embarquée.

- **Environnements de développement intégrés (IDE)**

Offrent un environnement de développement complet pour les applications embarquées, avec des fonctionnalités spécifiques pour l'IA.

I.8.8 Sécurité et confidentialité

Des mesures de sécurité et de confidentialité doivent être intégrées pour protéger les données et les systèmes contre les attaques et les intrusions.

- **Authentification et cryptage**

Des technologies essentielles pour garantir la sécurité des données et des communications.

I.8.9 Mise à jour et maintenance des modèles

Les modèles d'IA doivent être régulièrement mis à jour pour maintenir leur précision et leur efficacité.

- **Mises à jour logicielles en direct (OTA)**

Permettent de mettre à jour les modèles et le firmware des appareils embarqués à distance.

L'IA embarquée est un domaine en pleine expansion qui offre de nombreuses opportunités pour l'innovation et l'amélioration de la performance dans une grande variété d'applications.

I.9 Perspectives de l'Intelligence Artificielle

L'IA embarquée est un domaine en croissance rapide qui présente de nombreuses opportunités et défis passionnants. Une perspective est le potentiel de l'IA embarquée pour permettre de nouvelles applications et services qui étaient auparavant impossibles ou peu pratiques en raison des limitations de la puissance de calcul, de la durée de vie de la batterie ou de la connectivité réseau. Par exemple, l'IA intégrée peut permettre aux appareils intelligents d'effectuer des tâches complexes localement, réduisant ainsi le besoin de cloud computing et améliorant les temps de réponse.

Une autre perspective est l'importance de développer des algorithmes d'IA légers et économes en énergie pour les systèmes embarqués. Les algorithmes d'IA traditionnels sont souvent trop gourmands en ressources pour les appareils embarqués, dont la puissance de calcul et la durée de vie de la batterie sont limitées. Par conséquent, il existe un besoin pour

de nouveaux algorithmes optimisés pour les systèmes embarqués, tels que l'apprentissage automatique minuscule (TinyML) et d'autres techniques d'IA de pointe.

La sécurité est également une préoccupation majeure dans l'IA embarquée, car ces systèmes sont souvent utilisés dans des applications critiques telles que les soins de santé, les transports et l'automatisation industrielle. Il est donc essentiel de développer des systèmes d'IA embarqués robustes et sécurisés, capables de résister aux cyberattaques et de garantir la confidentialité et la sécurité des utilisateurs.

Enfin, une collaboration interdisciplinaire entre les chercheurs en IA, les ingénieurs en matériel informatique et les experts du domaine est nécessaire pour développer des systèmes d'IA embarqués efficaces. Cette collaboration est essentielle pour garantir que les systèmes d'IA embarqués sont adaptés aux besoins et contraintes spécifiques du domaine d'application, et qu'ils s'intègrent de manière transparente aux autres composants du système.

Dans l'ensemble, l'IA embarquée est un domaine prometteur avec de nombreuses opportunités et défis passionnants. En développant des algorithmes économes en énergie et sécurisés et en favorisant la collaboration interdisciplinaire, nous pouvons libérer tout le potentiel de l'IA embarquée et permettre de nouvelles applications et services qui améliorent nos vies d'innombrables façons.

I.10 Conclusion

Ce chapitre nous donne un aperçu sur l'IA, son développement qui repose sur diverses techniques et approches, telles que l'apprentissage automatique (machine learning), l'apprentissage profond (deep learning), les réseaux de neurones artificiels, et les algorithmes évolutionnaires…

Dans le chapitre suivant, nous allons aborder les fondements des réseaux de neurones, les SVMs ainsi mettre en évidence leurs principes et leurs performances.

CHAPITRE II :
LES ALGORITHMES DE L'INTELLIGENCE ARTIFICIELLE

II.1 Introduction

L'IA est une technologie en constante évolution qui a le potentiel de transformer notre monde. De la reconnaissance vocale à la conduite autonome, l'IA est de plus en plus présente dans notre vie quotidienne. Mais comment l'IA apprend-elle ? Quelles sont les différentes caractéristiques d'apprentissage possibles pour l'IA ? Dans ce chapitre, nous allons explorer ces questions et plus encore, une étude détaillée sur les algorithmes de l'IA.

II.2 Méthodes d'apprentissage de l'IA

L'IA est une branche de l'informatique qui vise à créer des systèmes différents capables de réaliser des tâches qui nécessitent normalement la réflexion humaine. Ces tâches incluent l'apprentissage, la compréhension du langage naturel, la perception visuelle (images), l'identification de la parole, la résolution variable de problèmes et la prise de décision…

Pour accomplir ces tâches, l'IA utilise différentes techniques, dont trois types d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Ces techniques sont essentielles pour comprendre comment l'IA fonctionne et comment elle peut être utilisée pour résoudre des problèmes complexes et détecter chaque erreur. La figure suivante illustre les différentes classes.

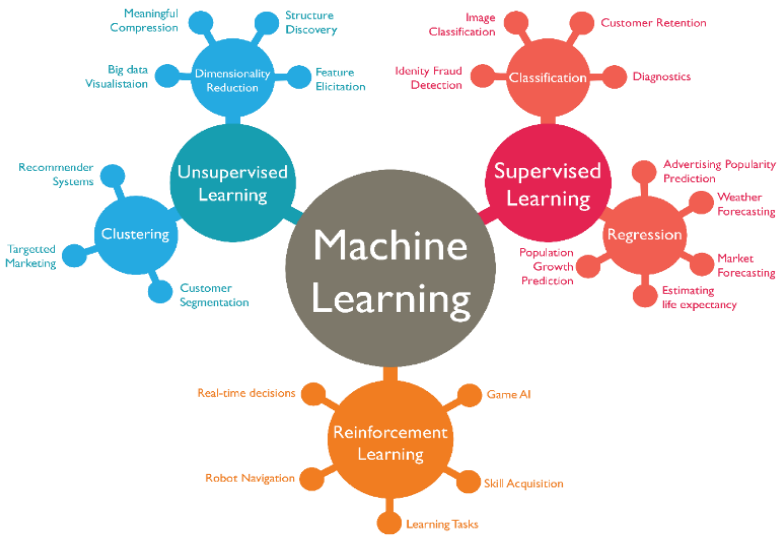


Figure.II.1 : Différents types d'apprentissage.

II.2.1 Apprentissage supervisé

L'apprentissage supervisé est la méthode la plus courante en IA. Dans cette approche, les algorithmes sont entraînés sur un ensemble de données étiqueté, où chaque exemple comprend des entrées et la sortie attendue, appelée "étiquette". Le but de l'apprentissage supervisé est de créer un modèle capable de prédire la sortie correcte pour de nouvelles entrées, en se basant sur les exemples d'entraînement.

Les exemples fournis sont sous la forme de couples entrée sortie (x_i, y_i). L'objectif est d'inférer la sortie y pour une nouvelle entrée x. On parle de classification si y_i ε {-1,1} plus généralement y_i ε N. Et on parle de régression si y_i ε R.[10]

Les tâches courantes de l'apprentissage supervisé comprennent la classification (attribuer une catégorie à une entrée) et la régression (prédire une valeur continue). Les algorithmes populaires incluent la régression linéaire, les arbres de décision, les réseaux de neurones et les machines à vecteurs de support (SVM).

La figure suivante montre un exemple d'apprentissage supervisé :

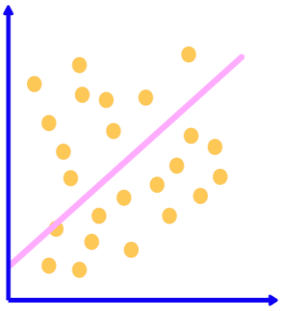


Figure II.2 : Apprentissage supervisé.

Dans l'apprentissage supervisé, un expert (ou oracle) doit préalablement correctement étiqueter des exemples. L'apprenant peut alors trouver ou approximer la fonction qui permet d'affecter la bonne « étiquette » à ces exemples.

II.2.1.1 Le modèle général de l'apprentissage supervisé

Il se décompose de trois parties :

- **Un environnement :** il engendre des entrées x_i tirées indépendamment et identiquement distribuées.
- **Un superviseur (oracle) :** retourne pour chaque entré une étiquette $U_i = f(x_i)$.
- **Un apprenant :** capable de réaliser une fonction h à partir d'un espace d'hypothèses H qui prédit au mieux la réponse du superviseur $y_i = h(x_i)$.

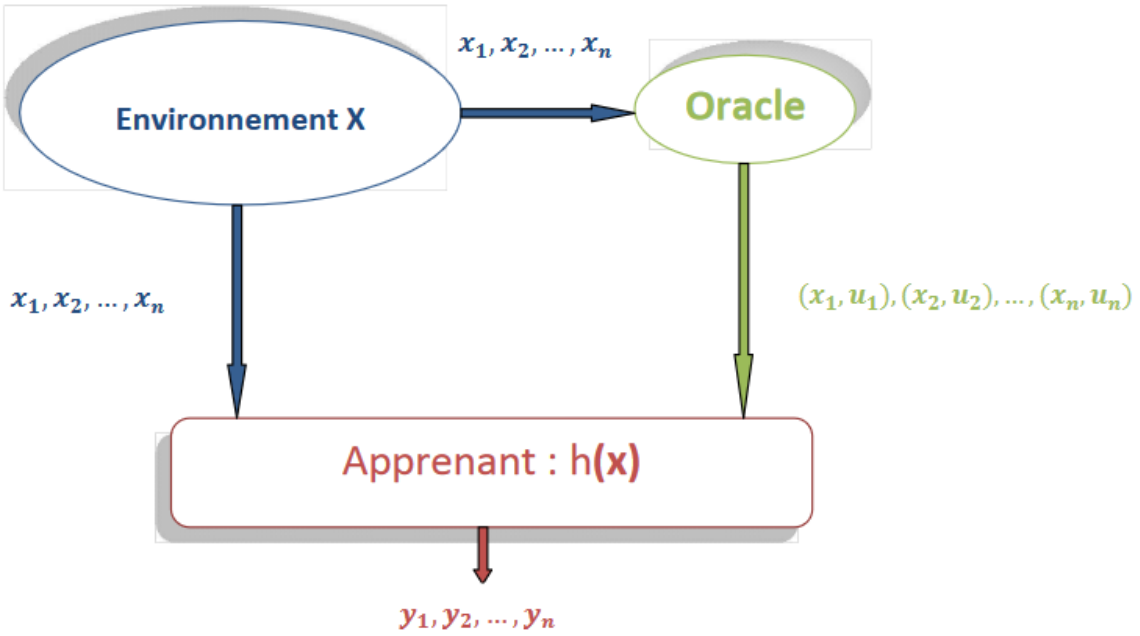


Figure II.3 : Modèle d'un apprentissage supervisé.

II.2.1.2 Avantages de l'apprentissage supervisé

- L'enseignement guidé peut être entrainement efficace.
- Il peut produire des systèmes précis et fiables si suffisamment de données de haute qualité sont disponibles
- De plus, les systèmes formés par enseignement guidé sont souvent faciles à comprendre, ce qui signifie que nous pouvons savoir comment l'IA prend ses décisions.

II.2.3 Limitations de l'apprentissage supervisé

L'enseignement guidé a aussi ses limites :

- Il nécessite une grande quantité de données d'entraînement étiquetées, ce qui peut être coûteux et chronophage à produire.
- De plus, il peut être moins efficace si les données ne représentent pas bien l'ensemble des données que le système rencontrera dans le monde réel.

II.2.2 Apprentissage non supervisé

L'apprentissage non supervisé implique l'entraînement d'algorithmes sur un ensemble de données non étiqueté, où les sorties correctes ne sont pas fournies. Le but de cette méthode est de découvrir la structure sous-jacente des données, en identifiant des relations et des modèles qui ne sont pas immédiatement évidents [11].

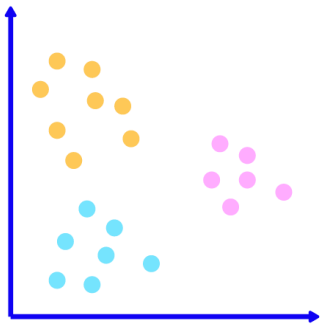


Figure II.4 : Apprentissage Non Supervisé.

Les tâches courantes de l'apprentissage non supervisé comprennent le regroupement (grouper des entrées similaires) et la réduction de la dimensionnalité (simplifier la représentation des données). Les algorithmes populaires incluent K-means, l'analyse en composantes principales (ACP).

II.2.2.1 Avantages de l'apprentissage non supervisé

- L'exploration autonome peut être utilisée lorsque nous avons beaucoup de data, mais pas beaucoup d'informations sur ce que ces données signifient.
- Il est utile pour découvrir des structures cachées dans les données
- Il peut être utilisé pour réduire le volume des données, ce qui peut aider à visualiser la data et à améliorer les performances des systèmes d'enseignement guidé.
- Il permet d'identifier des variables non observées dans les données, ce qui peut aider à comprendre les relations sous-jacentes.
- Il peut être utilisé pour la détection d'anomalies, en identifiant les valeurs qui s'écartent de la norme.
- Il peut être utilisé pour la régression, en identifiant les relations entre différentes variables dans les données.

II.2.2.2 Limitations de l'apprentissage non supervisé

- Il peut être plus difficile à comprendre et à interpréter que celui qui est supervisé.
- Il peut être difficile de savoir si le système a appris quelque chose d'utile.

- Il peut être plus difficile de valider ses résultats, car nous n'avons pas de vérité de base à comparer.
- Il peut être difficile de déterminer l'erreur, car sans données étiquetées, il n'y a pas de mesure objective de la performance.
- Il peut nécessiter un réseau plus complexe pour identifier les structures dans les données.
- Il peut être difficile de déterminer la ligne de meilleure adaptation, car il n'y a pas de sortie spécifique à prédire.

II.2.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une méthode d'apprentissage qui combine les caractéristiques de l'apprentissage supervisé et non supervisé. Dans cette approche, les algorithmes sont entraînés sur un ensemble de données partiellement étiqueté, où certaines entrées ont des étiquettes tandis que d'autres n'en ont pas. L'objectif de l'apprentissage semi-supervisé est d'utiliser les informations disponibles dans les données étiquetées et non étiquetées pour créer un modèle performant.

Les tâches courantes de l'apprentissage semi-supervisé incluent la classification et la régression. Les algorithmes populaires incluent les graphes de propagation de l'étiquette (label propagation), les modèles génératifs et les auto encodeurs variationnels [11].

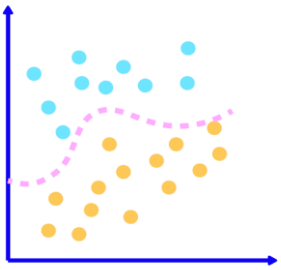


Figure II.5 : Apprentissage Semi-Supervisé.

II.2.4 L'apprentissage par renforcement

L'apprentissage par renforcement est une méthode d'apprentissage où un agent apprend à prendre des décisions en interagissant avec un environnement. L'agent reçoit des récompenses ou des punitions en fonction de ses actions, et il apprend à maximiser ses récompenses cumulatives au fil du temps. Cette méthode est particulièrement adaptée aux problèmes de décision séquentielle et de contrôle, où les actions ont des conséquences à long terme.

Les tâches courantes de l'apprentissage par renforcement comprennent la navigation, la manipulation d'objets et les jeux. Les algorithmes populaires incluent Q-learning, l'apprentissage par politique (policy gradient) et l'apprentissage par acteur-critique (actor-critic).

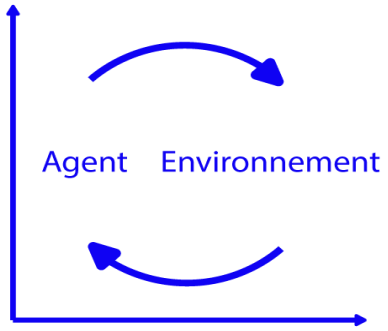


Figure II.6 : Apprentissage par Renforcement.

II.3 Algorithme d'intelligence artificielle

Un algorithme d'intelligence artificielle (IA) est un ensemble d'instructions et de processus mathématiques qui permet à un ordinateur d'imiter l'intelligence humaine et d'accomplir des tâches complexes. Ces algorithmes peuvent être classés en deux catégories principales : les algorithmes d'apprentissage automatique (Machine Learning) et les algorithmes de traitement du langage naturel (Natural Language Processing).

Les algorithmes d'apprentissage automatique (Machine Learning) sont conçus pour apprendre et s'améliorer automatiquement en analysant des données. Ils peuvent être entraînés sur des ensembles de données étiquetées ou non étiquetées pour effectuer des tâches telles que la classification, la régression et le clustering.

Les exemples d'algorithmes d'apprentissage automatique comprennent la régression linéaire, la régression logistique, les forêts aléatoires et les réseaux de neurones.

Les algorithmes de traitement du langage naturel sont conçus pour comprendre, générer et analyser le langage humain. Ils peuvent être utilisés pour la traduction automatique, la reconnaissance vocale, la synthèse vocale, la détection de sentiments et la génération de texte. Les exemples d'algorithmes de traitement du langage naturel comprennent les modèles de langage, les réseaux de neurones récurrents, les transformateurs et les modèles de langage pré-entraînés[12].

II.3.1 À quoi ressemble un algorithme ?

D'une manière générale, un algorithme peut être exprimé de différentes manières : langage naturel, pseudo-code, organigrammes, langages de programmation, drakon-chartes et tableaux de contrôle.

Les algorithmes d'IA permettent aux machines de :

- **Apprendre :** Analyser des données pour identifier des comportements inhabituels.
- **Raisonner :** Déduire des conclusions et faire des prédictions en utilisant des algorithmes de traitement du langage naturel
- **Résoudre des problèmes :** Trouver des solutions optimales à des problèmes complexes en utilisant des algorithmes
- **Détecter des anomalies et des fraudes :** Analyser des données pour identifier des comportements inhabituels. Aussi, prévenir les risques et les menaces et assurer la sécurité des systèmes et des données.

Les algorithmes d'IA sont utilisés dans de nombreux domaines, tels que la médecine, la finance, la logistique, la robotique et l'automatisation industrielle. Ils sont également utilisés dans les applications grand public telles que les assistants virtuels, les chatbots, les moteurs de recherche et les systèmes de recommandation…

Figure II.7 : Algorithme d'intelligence artificielle.

II.3.2 Importance des algorithmes en IA

Les algorithmes sont l'élément central et essentiel qui donne vie et puissance à l'intelligence artificielle. Voici pourquoi ils sont si importants :

II.3.2.1 Apprentissage et adaptation

Les algorithmes permettent aux machines d'apprendre et de s'améliorer en analysant des données. Ils identifient des modèles, font des prédictions et ajustent leur comportement en

fonction des nouvelles informations. C'est ce qui permet à l'IA d'être plus qu'une simple programmation statique.

II.3.2.2 Accomplir des tâches complexes

Les algorithmes spécialisés permettent aux machines de réaliser des tâches complexes qui nécessiteraient normalement une intelligence humaine. Cela va de la reconnaissance d'images et de la compréhension du langage à la prise de décision et au contrôle de robots.

II.3.2.3 Automatisation et efficacité

Grâce aux algorithmes, on peut automatiser des processus répétitifs et fastidieux, ce qui libère les humains pour des tâches plus créatives et stratégiques. Cela améliore également la productivité et l'efficacité globale.

II.3.2.4 Analyse et détection

Certains algorithmes peuvent analyser des données massives pour détecter des anomalies, des fraudes ou des problèmes potentiels. Cela est utile dans des domaines comme la cybersécurité, la finance et la maintenance prédictive.

II.3.2.5 Génération créative

Des algorithmes innovants peuvent même générer du contenu créatif, comme de la musique, des articles ou des images. Ils le font en analysant des styles et des patterns existants pour créer quelque chose de nouveau.

Les algorithmes sont la base de l'IA. Ils permettent aux machines d'apprendre, de s'adapter, d'accomplir des tâches complexes, d'automatiser des processus, d'analyser des données et même de générer du contenu créatif. Sans algorithmes sophistiqués, l'IA ne serait pas capable des prouesses que l'on voit aujourd'hui.

II.3.3 Réseaux de neurones

Lorsqu'on parle de réseau de neurones en IA, nombreuses personnes, spécialement profanes en la matière, pensent directement à des grandes fonctions mathématiques et au fonctionnement complexe du cerveau humain. Du coup, ils ne veulent même pas entrer à fond de ce domaine récent qui révolutionne l'informatique de nos jours afin d'en comprendre les merveilles qu'il cache et même la simplicité de son fonctionnement dans la résolution presque de la quasi-totalité des problèmes complexes modernes. Bien sûr, la mise au point d'un réseau de neurones ne s'écarte pas de l'implémentation de quelques fonctions mathématiques et dont chacune réalise une tâche bien précise.

Commençons par explorer le fonctionnement d'un neurone biologique, nous définirons ensuite le neurone artificiel, nous plongerons dans l'architecture des réseaux de neurones.

Nous aborderons également les principes fondamentaux qui sous-tendent leur fonctionnement et leur capacité à apprendre et à s'adapter.

II.3.3.1 Neurone biologique

Un neurone biologique est une cellule nerveuse qui est l'unité de base du système nerveux. Sa fonction principale est de transmettre des informations électrochimiques à travers le corps. Les neurones sont constitués d'un corps cellulaire, de dendrites qui reçoivent des signaux d'autres neurones, d'un axone qui transmet les signaux à d'autres neurones et d'une terminaison axonale qui relâche les neurotransmetteurs pour communiquer avec d'autres neurones. Les neurones sont essentiels pour la communication rapide et efficace entre les différentes parties du corps et pour la coordination des fonctions corporelles [13].

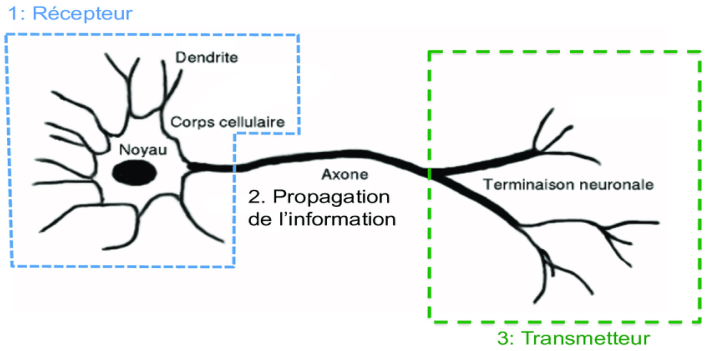


Figure II.8 : Image descriptive d'un neurone biologique.

II.3.3.2 Neurone Artificiel

Un neurone artificiel est une unité de traitement de l'information inspirée des neurones biologiques. Le premier neurone formel est proposé en 1943 par McCulloch et Pitts, dont le schéma présenté sur la figure 2.11 est une évolution. Chaque neurone est composé de N entrées connectées à l'information extérieure (équivalent aux boutons synaptiques des dendrites) et reliées à une fonction chargée de « combiner » les signaux incidents. Il s'agit en général d'une somme pondérée, dont le résultat contrôle l'intensité du signal d'activation produit à la sortie du neurone.

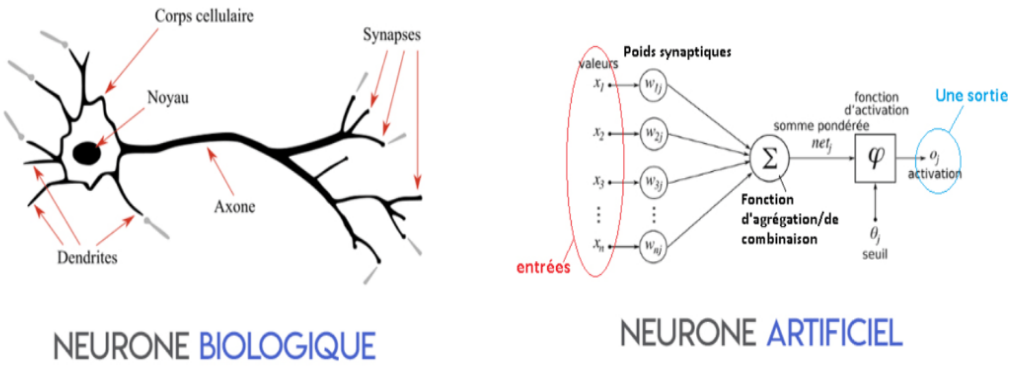


Figure II.9 : Comparaison entre un neurone biologique et un Neurone Artificiel.

II.3.3. 3 Caractéristiques des Réseaux de Neurones Artificiels

II.3.3.3.1 Présentation

Les réseaux de neurones artificiels sont des modèles qui s'inspirent du fonctionnement du cerveau humain. Leur objectif est de concevoir des machines capables de reproduire, non pas d'imiter, mais de se rapprocher autant que possible du comportement intelligent des êtres humains. C'est ainsi que naît la notion d'Intelligence Artificielle.

II.3.3.3.2 Les Entrées

Le neurone calcule Chaque neurone du réseau est doté de plusieurs entrées, notées $Ei\ (1 \leq i \leq n)$, où n représente le nombre de neurones d'entrée. Chaque entrée est associée à un poids ωi, auquel s'ajoute un coefficient de biais $w0$ qui est attribué à une entrée fictive $Ei = 1$. Ainsi, l'information qui parvient au neurone est le résultat du produit $\omega i\ x\ Ei$ pour chaque entrée i. la somme *:* $\omega i\ x\ Ei\ ni = 1,$ celle-ci sera le potentiel du neurone, la sortie sera générée par une fonction d'activation, qui sera très importante car elle déterminera par la suite le fonctionnement du réseau.[14]

II.3.3.3.3 Les fonctions d'activation

La fonction d'activation, également appelée fonction de transfert, est une fonction qui doit produire un nombre proche de 1 lorsque les entrées sont pertinentes et proche de 0 lorsque les entrées sont non pertinentes. En général, on utilise des fonctions qui renvoient des valeurs dans l'intervalle réel [0 ;1]. Lorsque le nombre est proche de 1, l'unité (le neurone) est considérée comme active, tandis que lorsqu'il est proche de 0, l'unité est considérée comme inactive. Les fonctions d'activation peuvent prendre différentes formes telles que binaire, linéaire, sigmoïde, et bien d'autres encore.

Le tableau ci-dessous présente les fonctions d'activation les plus couramment utilisées.

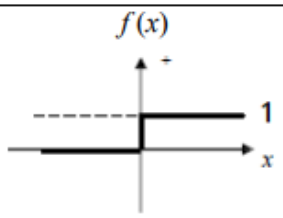
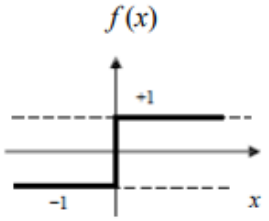
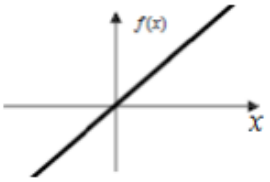
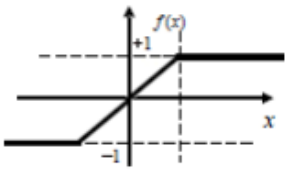
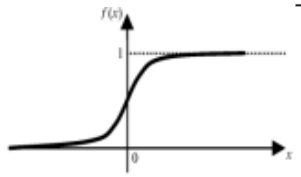
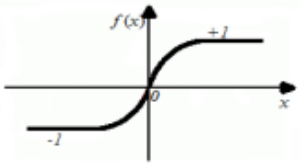
Catégories	Types	Equation	Allure
Seuil	Heaviside	$f(x) = \begin{cases} 0 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases}$	
	Signe	$f(x) = \begin{cases} -1 \; si \; x \leq 0 \\ 1 \; si \; x > 0 \end{cases}$	
Linéaire	Identité	$f(x) = x$	
	Saturé symétrique	$f(x) = \begin{cases} -1 \; si \; x \leq -1 \\ 1 \; si \; x \geq 1 \\ x \quad sinon \end{cases}$	
Non linéaire	Sigmoïde	$f(x) = \frac{1}{1 + e^{-x}}$	
	Tangente hyperbolique	$f(x) = \frac{2}{1 + e^{-x}} - 1$	

Figure II.10 : Les fonctions d'activations.

II.3.3.3.4 La sortie

Peut-être utiliser en tant que résultat final, ou pour alimenter d'autres neurones.

II.3.3.4 Architecture des réseaux de neurones

Après avoir découvert les neurones biologiques et artificiels, on pouvait classifier les Réseaux de neurones selon le mode d'apprentissage, on verra maintenant la classification par architecture qui elle aussi se subdivise en deux grandes familles : Les réseaux Feed-forward et les réseaux Feed-back.

II.3.3.4.1. Les réseaux Feed-forward

Un réseau de neurones à action directe, ou à propagation avant, en anglais *feedforward neural Network*. Ils ont des couches de neurones connectées de manière unidirectionnelle sans boucle de rétroaction entre les couches, c'est à dire, les neurones d'une couche sont connectés à tous les neurones de la couche suivante, mais pas à ceux de la couche précédente. On y trouve :

II.3.3.4.1.A Le perceptron monocouche

Également appelé perceptron à couche unique, le perceptron monocouche désigne un réseau de neurones artificiels qui contient N neurones en entrée et M neurones en sortie. A priori, le perceptron monocouche n'a pas de connaissance. En conséquence, les poids initiaux sont donc attribués de façon aléatoire. D'autre part, le perceptron à couche unique est un outil d'apprentissage supervisé qui ne peut apprendre que des fonctions linéaires séparables.

Lorsque les fonctions ne sont pas linéairement séparables, il n'y a en effet aucune chance que le processus d'apprentissage atteigne un point où tous les points sont correctement classés[15]. Il s'agit donc d'un outil mathématique d'analyse de données donc les fonctionnalités sont limitées.

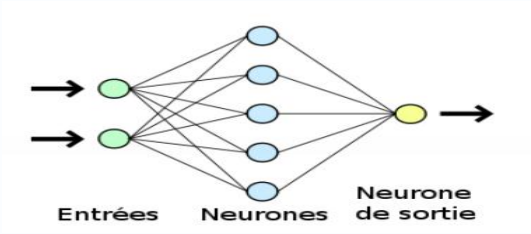


Figure II.11 : Exemple d'une architecture du perceptron monocouche.

II.3.3.4.1.B Perceptron Multi-Couche

Le PMC est un modèle de réseau qui contient, à la différence de neurones formels qui n'ont que la couche d'entrée et celle de sortie, des couches cachées ayant pour fonction de faire des traitements intermédiaires pour une meilleure prédiction. Ces architectures font usage des fonctions qui sont non linéaires telles que la fonction tangente hyperbolique ou fonction logistique pour activer les sorties des neurones. On peut instruire avec ces modèles un réseau avec plusieurs couches cachées et de neurones. Ainsi, plus il y a de couches, plus le réseau est profond et le modèle devient riche. C'est du Deep Learning.

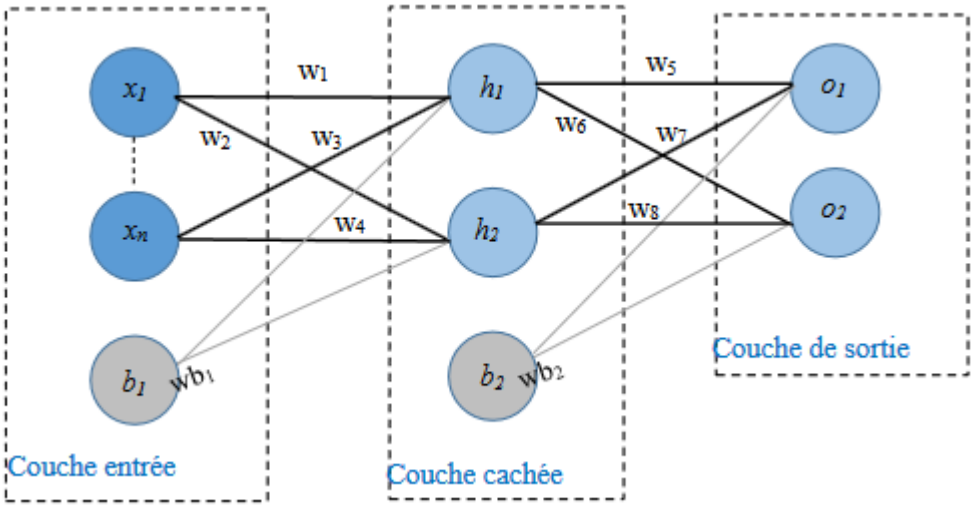


Figure II.12 : Exemple d'une architecture du perceptron multicouche.

La figure II.12 représente un réseau composé de neurones interconnectés en trois couches successives. La première couche, qui est celle d'entrée est composée de neurones dits transparents qui n'effectuent aucun calcul mais qui distribuent leurs entrées à tous les neurones de la couche suivante dite couche cachée. Les neurones de la couche cachée reçoivent ainsi les entrées $\{x_1, ..., x_n\}$ de la couche d'entrée avec les poids associés $\{w_1, ..., w_n\}$. Chaque neurone de la couche cachée comprend deux phases. La première que l'on peut appeler *netinput* contient le résultat du calcul de la somme pondérée des entrées de la couche d'entrée et du biais: $Netinputj = (x_1*w_1) + (x_2*w_2) + ... + (x_n*w_n) + (b_j*w_{bj})$ et la seconde *netoutput* comprend le résultat de la transformation de *netinput* par l'intermédiaire de la fonction d'activation. En général, les couches cachées utilisent la fonction ReLU (Rectified Linear Unit), en français, Unité Linéaire Rectifiée pour le cas de prédiction des valeurs positives non dérivable et la couche de sortie utilise soit la fonction sigmoïde pour le cas d'une sortie binaire soit la fonction softmax pour des classifications multi-classe comme fonctions d'activation [14].

II.3.3.4.2 Les réseaux Feed-back

Appeler aussi les réseaux récurrents, sont des types de réseaux de neurones artificiels où les connexions entre les neurones forment des boucles, permettant ainsi de conserver une mémoire à court terme des informations précédemment traitées. Contrairement aux réseaux de neurones classiques où les informations circulent uniquement dans un sens (feed-forward), les réseaux récurrents ont des connexions récurrentes qui leur permettent de prendre en compte les séquences et les dépendances temporelles dans les données en entrée[15].

Les réseaux récurrents sont particulièrement adaptés pour traiter des données séquentielles telles que des séquences de mots dans un texte, des séquences temporelles dans des données

temporelles, des séquences d'actions dans des vidéos, etc. Grâce à leur capacité à conserver une mémoire des états précédents, les réseaux récurrents sont capables de capturer des motifs complexes et d'effectuer des prédictions sur des données séquentielles.On y trouve:

II.3.3.4.2.A Les réseaux de Hopfield

Sont des réseaux entièrement connectés où chaque neurone est lié à tous les autres sans distinction entre les neurones d'entrée et de sortie. Ces réseaux peuvent retrouver un objet stocké même avec des représentations partielles ou bruitées. Leur utilisation principale est le stockage de connaissances et la résolution de problèmes d'optimisation. L'apprentissage se fait de manière non supervisée[15]**.**

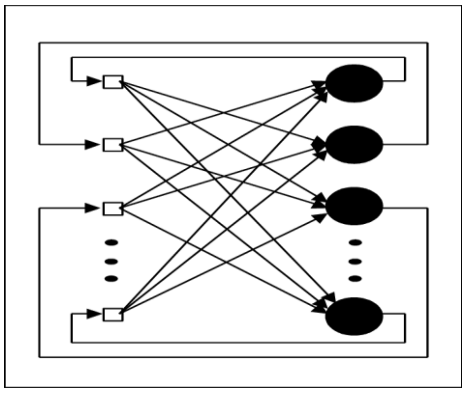


Figure II.13 : Architecture d'un Réseau de Hopfield

II.3.3.5 Techniques d'implémentation dans différentes architectures matérielles

Programmer une application sur des réseaux de neurones implique d'utiliser des bibliothèques logicielles dont les plus populaires sont TensorFlow , Caffe , Torch et PyTorch , etc. Ces bibliothèques sont utilisables sur ordinateurs avec CPU multicœurs sans ou avec GPU.

Différents supports matériels

Les opérations effectuées par les réseaux de neurones pouvant être parallélisées, une forte accélération des calculs peut être obtenue grâce aux GPU ou aux FPGA.

Depuis le milieu des années 2010, des accélérateurs compatibles avec les besoins de l'embarqué sont apparus, regroupant des centaines de GPU dans un stick au format proche d'une clé USB. On peut citer :

Intel Movidius NCS stick ;

Google Coral USB stick ;

Nvidia Jetson Nano.

La charge de calcul est énormément plus intense lors de l'apprentissage que lors de l'inférence. Les réseaux sont donc généralement entraînés sur des serveurs ou gros GPU avant

d'être implémentés sur des supports matériels embarqués. L'inférence utilise des données de la vie courante ce qui légitimiste le besoin de l'exécuter sur du matériel embarqué, souvent proche des capteurs qui fournissent ces données.

II.3.3.5.1 Implémentation sur CPU

A. Définition

De nombreux termes ont été proposés pour décrire l'implémentation, dont les plus précis et adéquats sont : l'adaptation, l'exécution, la fabrication, l'intégration et la concrétisation. En résumé, l'implémentation consiste à concrétiser un outil informatique en se basant sur des documents existants. Dans le contexte des réseaux de neurones artificiels (ANN), l'implémentation implique la conception, le développement et l'intégration d'un algorithme ANN au sein d'un outil informatique.

La schéma ci-dessous montre un exemple d'une implémentation d'un réseau de neurones :

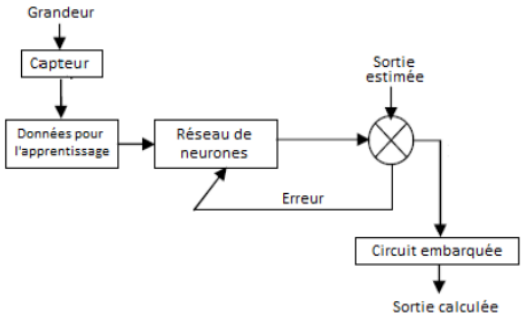


Figure II.14 : Schéma général d'une implémentation d'un ANN.

B. Le calcul séquentiel

Ce type de traitement se caractérise par l'exécution des tâches une à la suite de l'autre, de manière ordonnée. Sa simplicité d'implémentation et sa faible consommation de ressources en font un choix attractif, notamment dans le domaine des processeurs et dérivés (microcontrôleurs, DSP, etc.).

Cependant, son principal inconvénient réside dans la lenteur du traitement des données. En effet, chaque tâche doit attendre que la précédente soit terminée avant de pouvoir s'exécuter, ce qui peut s'avérer problématique pour les applications nécessitant un traitement rapide.

Au cœur des réseaux de neurones, les accumulateurs jouent un rôle crucial. Leur principe de fonctionnement repose sur la multiplication de chaque entrée par son poids correspondant (w_i x E_i), suivie de l'addition du résultat à une valeur cumulée précédemment. Cette opération se répète jusqu'à l'obtention d'une somme finale, qui est ensuite transmise à la fonction de transfert sélectionnée[1].

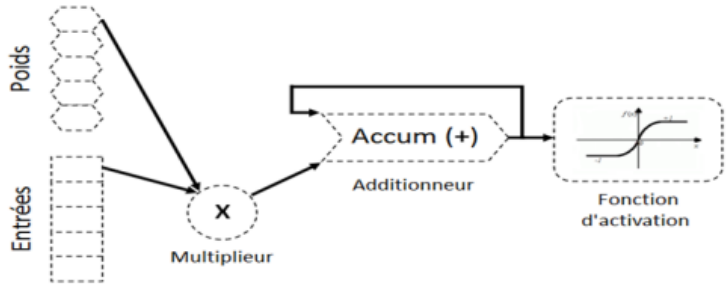


Figure II.15: Modèle d'un calcul série.

C. Le calcul parallèle

Le calcul parallèle se caractérise par l'exécution simultanée de plusieurs parties d'une même tâche, préalablement partitionnées et adaptées pour être réparties entre plusieurs processeurs. Cette approche permet d'accélérer considérablement le traitement de problèmes complexes et de grande envergure.

Domaines d'application du calcul parallèle

Le calcul parallèle trouve son application privilégiée dans le domaine des composants programmables tels que les PAL (Programmable Array Logic), les GAL (Gate Array Logic) et les FPGA (Field Programmable Gate Arrays). Ces composants offrent une flexibilité et une reconfigurabilité accrues, les rendant parfaitement adaptés à l'exécution de tâches parallèles.

En pratique, on peut avoir deux types de calcul parallèle :

D. Le calcul parallèle partiel

Ou Partial Parallel Processing (PPP) en anglais, cette technique utilise autant de multiplieurs que de nombre pair du produit $Wi\ x\ Ei$ suivis par des additionneurs a deux entrées comme le montre la figure suivante :

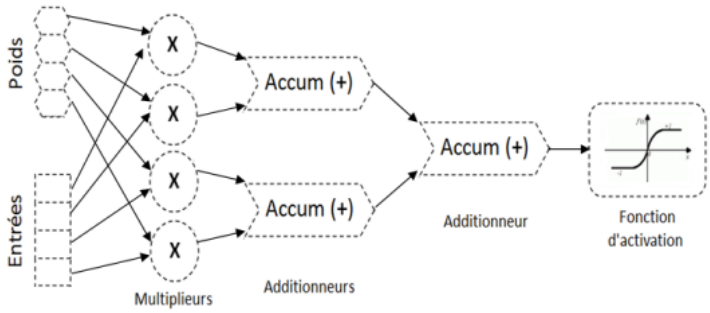


Figure II.16: Modèle d'un calcul PPP.

E. Le calcul parallèle complet

Ou Full Parallel Processing (FPP) en anglais, cette technique utilise le même nombre de multiplieurs, comme dans le modèle PPP, la différence est l'utilisation d'un seul et unique additionneur qui a autant d'entrées que le nombre de produits W_i x E_i. La sortie de cet additionneur sera connectée à l'entrée de la fonction de transfert comme le montre la figure suivante :

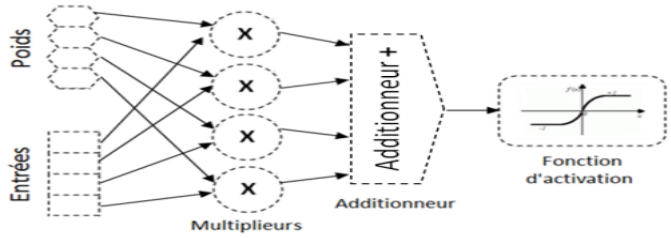


Figure II.17 : Modèle d'un calcul FPP.

Chaque technique présente des avantages et des inconvénients. Ce qui fait que le choix de la technique à utiliser dépend des ressources disponibles, de l'utilisation de notre réseau et le besoin de parallélisme[1].

II.3.3.5.2 L'implémentation sur (GPU)

L'implémentation des réseaux de neurones sur GPU (Unité de Traitement Graphique) est devenue courante en raison des avantages considérables qu'elle offre en termes de performances et d'efficacité pour l'entraînement et l'inférence des modèles de deep learning. Voici les étapes générales pour implémenter des réseaux de neurones sur GPU :

- **Choisir les Outils et Bibliothèques Appropriés**

Les frameworks de deep learning les plus populaires qui supportent l'exécution sur GPU incluent :

- TensorFlow : Utilise tf.device pour spécifier l'utilisation du GPU.
- PyTorch : Utilise .to(device) pour déplacer les tensors et les modèles vers le GPU.
- Keras : Intégré dans TensorFlow, utilise également tf.device.
- MXNet : Utilise .as_in_context(mx.gpu()) pour spécifier le GPU.

- **Installer les Outils Nécessaires**

Installer les Outils Nécessaires :

Installer les versions des bibliothèques compatibles avec le GPU (e.g., « tensorflow-gpu », PyTorch avec CUDA).

- **Vérifier la Disponibilité du GPU**

Utiliser des fonctions spécifiques des bibliothèques pour vérifier si un GPU est disponible)" (tf.config.list_physical_devices('GPU')" pour TensorFlow et "torch.cuda.is_available()" pour PyTorch.

- **Définir et Entraîner un Modèle sur GPU**

Déplacer les données et le modèle sur le GPU (".to(device) " en PyTorch).

- **Inférence sur le GPU**

Effectuer l'inférence en déplaçant les données d'entrée sur le GPU de la même manière que lors de l'entraînement, en s'assurant que le modèle est en mode évaluation (" model.eval()").

L'utilisation de GPU pour les réseaux de neurones permet d'accélérer considérablement le processus d'entraînement et d'inférence, rendant possible l'utilisation de modèles plus grands et plus complexes. Les bibliothèques modernes de deep learning comme TensorFlow et PyTorch rendent cette utilisation relativement simple avec des fonctions intégrées pour déplacer les données et les modèles sur le GPU.

II.3.3.5.3 FPGA (Field Programmable Gate Arrays)

- Implémentation de réseaux de neurones sur des FPGA pour une exécution hautement parallèle et basse consommation d'énergie.
- Utilisation de langages de description matérielle comme VHDL ou Verilog pour définir l'architecture des circuits.
- Utilisation de bibliothèques et d'outils de haut niveau pour la synthèse et la mise en œuvre des réseaux de neurones sur FPGA.

II.3.3.5.4 ASIC (Application-Specific Integrated Circuits)

- Conception de circuits intégrés spécifiquement optimisés pour les réseaux de neurones, offrant des performances et une efficacité énergétique maximales.
- Utilisation de techniques de conception matérielles spécialisées telles que le pipelining, le parallélisme et la mémoire intégrée pour accélérer les opérations de calcul.
- Intégration de blocs matériels spécialisés pour des opérations courantes dans les réseaux de neurones, comme les couches de convolution ou les opérations de multiplication matricielle.

Chaque architecture matérielle a ses propres avantages et inconvénients en termes de performances, de puissance, de coût et de flexibilité. Le choix de l'architecture dépendra des besoins spécifiques de l'application et des contraintes de conception.

II.3.3.6 Applications en intelligence artificielle

Les réseaux de neurones artificiels sont largement utilisés dans de nombreuses applications clés de l'intelligence artificielle, en voici quelques-unes parmi les plus importantes :

- **Reconnaissance et classification d'images**

Les réseaux de neurones convolutifs (CNN) sont excellents pour identifier, classer et analyser des images avec une grande précision.

- **Traitement du langage naturel**

Les réseaux de neurones récurrents (RNN) et les Transformers permettent une compréhension approfondie du langage humain, facilitant des applications comme la traduction automatique, la génération de texte, la réponse aux questions, etc.

- **Prédiction et prise de décision**

Les réseaux de neurones denses peuvent apprendre à partir de données complexes pour faire des prédictions précises et prendre des décisions éclairées, dans des domaines comme les finances, la santé, la gestion des risques, etc.

- **Robotique et contrôle**

Les réseaux de neurones aident à la perception, à la planification des mouvements et à la prise de décision pour guider efficacement des robots et d'autres systèmes autonomes.

- **Génération créative**

Les réseaux de neurones génératifs comme les GANs peuvent créer du contenu original comme de la musique, des images, des vidéos, en imitant les styles humains.

En résumé, les réseaux de neurones sont devenus un outil central pour donner aux systèmes d'IA des capacités de perception, de compréhension, de prédiction et de création à la pointe de la technologie.

II.3.4 Machines à vecteurs de support (SVM)

Les "Support Vector Machines" (SVM), ou Séparateurs à Vaste Marge, sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification et de régression. Cette méthode, initiée par Vladimir Vapnik, vise à trouver le meilleur hyperplan séparant linéairement les exemples positifs des exemples négatifs tout en maximisant la marge entre les points les plus proches des deux classes. Les SVM ont ensuite été améliorés avec l'introduction des fonctions noyau (kernels) par Boser, ce qui permet de traiter les problèmes de non-linéarité. Pour les données non linéairement séparables, Cortes et

ses collaborateurs ont proposé une version régularisée des SVM, qui tolère certaines erreurs d'apprentissage tout en les pénalisant[16].

Les SVM ont été appliqués à de très nombreux domaines (bio-informatique, recherche d'information, vision par ordinateur, finance...). Selon les données, la performance des machines à vecteurs de support est de même ordre, ou même supérieure, à celle d'un réseau de neurones ou d'un modèle de mixture gaussienne.

II.3.4.1 Notion de base

Cette technique est une méthode de classification à deux classes. Son objectif est non seulement de séparer les exemples positifs des exemples négatifs mais aussi de repousser au maximum les uns des autres. La méthode cherche alors l'hyperplan qui sépare les deux classes d'exemples, en garantissant que la marge entre les positifs et les négatifs les plus proches de l'hyperplan soit maximale. Cela garantit une meilleure généralisation du modèle car de nouveaux exemples pourraient ne pas être trop similaires à ceux utilisés pour l'apprentissage. L'intérêt de cette méthode est la sélection de vecteurs supports qui représentent les vecteurs discriminant grâce auxquels est déterminé l'hyperplan. Seuls ses vecteurs interviennent dans la solution, ce qui rend le problème moins complexe.

II.3.4.1.A. Hyper plan

Quand on est dans un espace de représentation euclidien, on peut librement faire des hypothèses sur la géométrie des classes ou sur celles de leurs surfaces séparatrices ; ceci permet de mettre au point des techniques d'apprentissage non statistiquement fondées a priori, mais peut être plus faciles à appliquer. La plus simple d'entre elles est de supposer que deux classes peuvent être séparées par une certaine surface (voir figure II.18). Les paramètres qui régissent son équation sont alors les variables à apprendre. Le nombre de paramètres à calculer est minimal si l'on suppose cette surface linéaire. Dans ; une surface linéaire est un hyperplan H; défini par l'équation : $wTx + b = 0$; Si deux classes $C1$ et $C2$ sont séparables par H, alors tous les points de la première classe sont par exemple tels que : $x \in C1 \Rightarrow wTx + b \geq 0$ et de la seconde vérifient alors : $x \in C2 \Rightarrow wTx + b < 0$.

On parle d'hyperplan optimal lorsque celui-ci sépare les deux classes en garantissant un maximum d'espace entre les vecteurs de support[16].

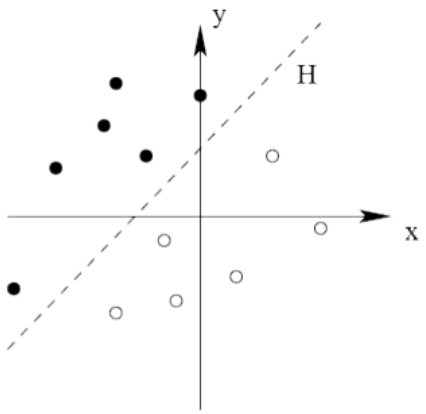


Figure II.18 : Hyperplan séparant deux classes.

II.3.4.1.B Support Vectors (vecteurs de support)

Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support.

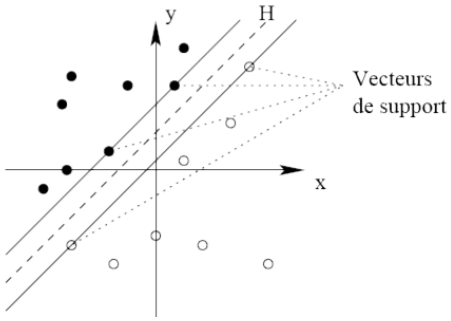


Figure II.19 : Les vecteurs de support.

ll.3.4.1.C Marge

La marge est la distance entre l'hyperplan et les échantillons les plus proches. Ces derniers sont appelés vecteurs supports. L'hyperplan qui maximise la marge est donné par :

$$\arg\max_{w,w0}\min_{k}\{||x-x_k||:x\in\mathbb{R}^N, w^Tx+w_0=0\} \quad \text{..... (1)}$$

Il s'agit donc de trouver *w* et w_0 remplissant ces conditions, afin de déterminer l'équation de l'hyperplan séparateur :

$$h(x) = w^Tx + w_0 = 0 \text{.....(2)}$$

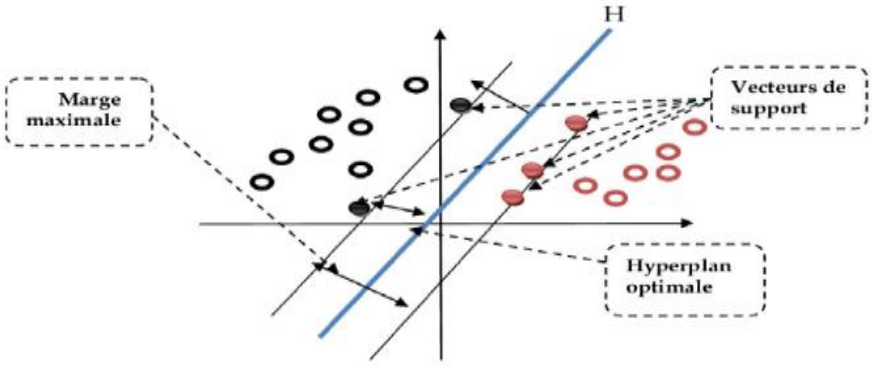


Figure II.20 : Représentation de la marge.

II.3.4.2 Principes fondamentaux d'un SVM

II.3.4.2.A Maximisation de la marge

Intuitivement, le fait d'avoir une marge plus large procure plus de sécurité lorsque l'on classe un nouvel exemple. De plus, si l'on trouve le classificateur qui se comporte le mieux vis-à-vis des données d'apprentissage, il est clair qu'il sera aussi celui qui permettra au mieux de classer les nouveaux exemples. Dans le schéma qui suit, la partie droite nous montre qu'avec un hyperplan optimal, un nouvel exemple reste bien classé alors qu'il tombe dans la marge. On constate sur la partie gauche qu'avec une plus petite marge, l'exemple se voit mal classé[16].

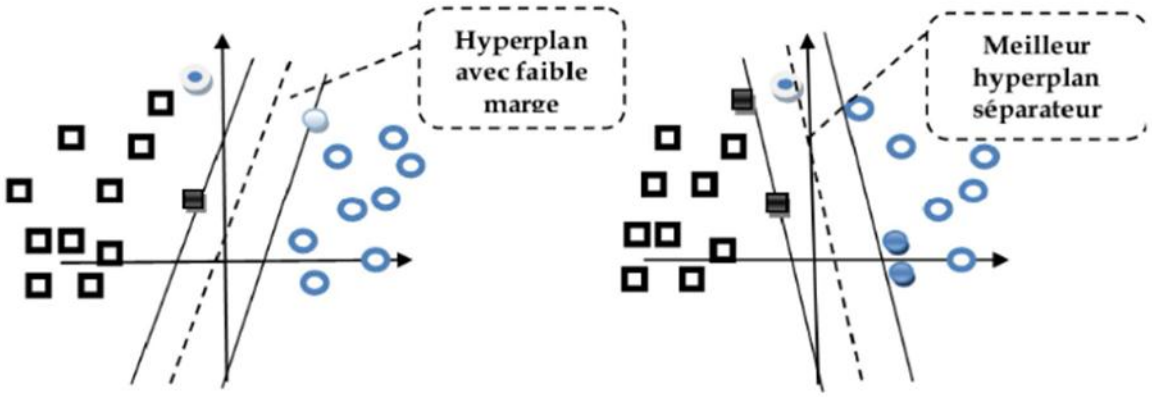


Figure II.21 : Meilleur hyperplan séparateur

En général, la classification d'un nouvel exemple inconnu est donnée par sa position par rapport à l'hyperplan optimal. Dans le schéma suivant, le nouvel élément sera classé dans la catégorie des « + ».

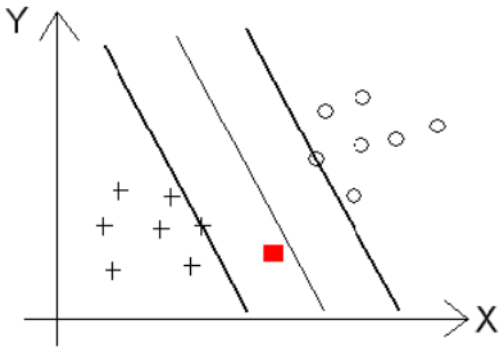


Figure II.22 : L'hyperplan optimal.

II.3.4.3 Méthodes de classification

II.3.4.3.A Cas linéairement séparable

Les cas linéairement séparables sont les plus simples des SVM car ils permettent de trouver facilement le classificateur linéaire.

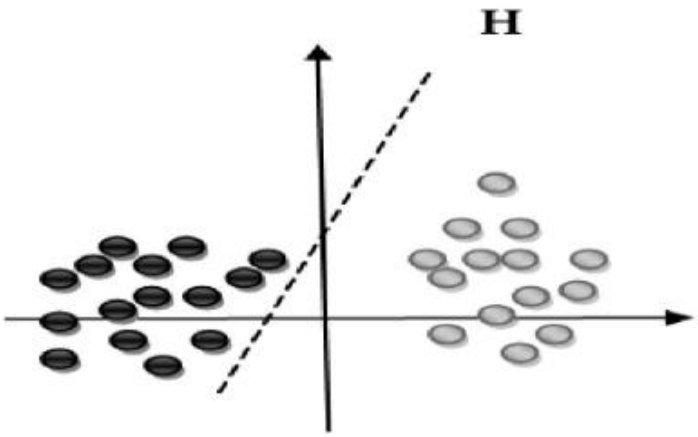


Figure II.23 : Exemple d'un cas linéairement séparable.

Dans la plupart des problèmes réels il n'y a pas de séparation linéaire possible entre les données, le classificateur de marge maximale ne peut pas être utilisé car il fonctionne seulement si les classes de données d'apprentissage sont linéairement séparables[17].

II.3.4.3.2 Cas non linéaire

Pour surmonter les inconvénients des cas non linéairement séparables, l'idée des SVM est de changer l'espace des données. La transformation non linéaire des données peut permettre une séparation linéaire des exemples dans un nouvel espace. On va donc avoir un changement de dimension. Cette nouvelle dimension est appelée « espace de re-description ». En effet, intuitivement, plus la dimension de l'espace de re-description est grande, plus la probabilité

de pouvoir trouver un hyperplan séparateur entre les exemples est élevée. Ceci est illustré par le schéma suivant :

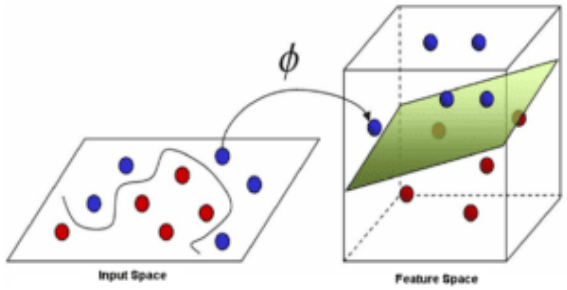


Figure II.24 : Exemple de projection dans un espace de redescription.

On a donc une transformation d'un problème de séparation non linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de re-description de plus grande dimension. Cette transformation non linéaire est réalisée via une fonction noyau. En pratique, quelques familles de fonctions noyau paramétrables sont connues et il revient à l'utilisateur de SVM d'effectuer des tests pour déterminer celle qui convient le mieux pour son application. On peut citer les exemples de noyaux suivants : polynomiale, gaussien, sigmoïde et l'Alsacien[17].

II.3.4.3.3 SVM Multi-Classes

La plupart des problèmes ne se contentent pas de deux classes de données. Il existe plusieurs méthodes pour faire la classification multi-classes. La première méthode est appelée Un-Contre-Tous. C'est une approche étendant la notion de marge aux cas multi-classes. Cette formulation intéressante permet de poser un problème d'optimisation unique. Le problème fait intervenir N fonctions de décision. La deuxième méthode est une méthode dite Un-contre-Un. Au lieu d'apprendre N fonctions de décisions, ici chaque classe est discriminée d'une autre.

II.3.4.3.3 A Un contre tous (One-vs-Rest)

L'approche la plus naturelle est d'utiliser cette m´méthode de discrimination binaire et d'apprendre N fonctions de d´décision *{fm} m = 1 . . . N* permettant de faire la discrimination entre chaque classe de toutes les autres (chaque classe est opposée à toutes les autres), il faut donc poser N problèmes binaires. Le k i 'me classificateur s´épare les données de la classe k de tout le reste des données d'apprentissage. Chaque classificateur renvoie 1 si la forme à reconnaitre appartient à la classe, -1 sinon. Alors pour reconnaitre une forme il faut la soumettre à tous les autres classificateurs, pour prendre une d´décision de classification, on garde la classe qui a eu la valeur maximale de toutes les fonctions de d´décision. Donc l'affectation d'un nouveau point x à une classe Ci se fait par la relation [16] :

i = arg max fm(x)m = 1 . . . N……..(3)

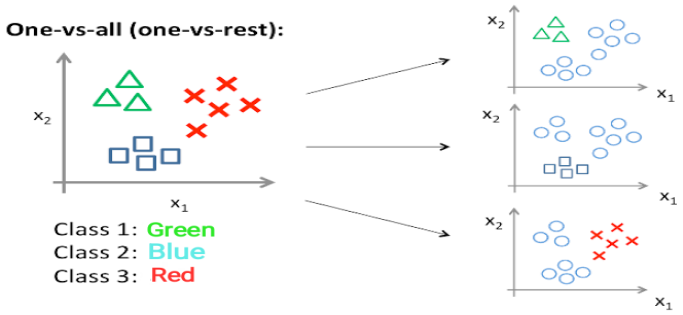


Figure II.25 : Exemple d'un SVM One vs All.

Cette méthode est critique à cause de son asymétrie, puisque chaque hyperplan est entrainé sur un nombre d'exemples négatifs beaucoup plus important que le nombre d'exemples positifs. La méthode un contre un est une méthode symétrique qui corrige ce problème.

II.3.4.3.4.B Un contre un (One-vs-One)

La deuxième m´méthode est une méthode dite d'un contre un. Au lieu d'apprendre N fonctions de d'décisions, ici chaque classe est discriminée d'une autre. Ainsi, N (N − 1) /2 fonctions de décisions (classificateurs) sont apprises et chacune d'entre elles effectue un vote pour l'affectation d'un nouveau point x. La classe de ce point x à le plus grand nombre de votes devient ensuite la classe majoritaire [9].

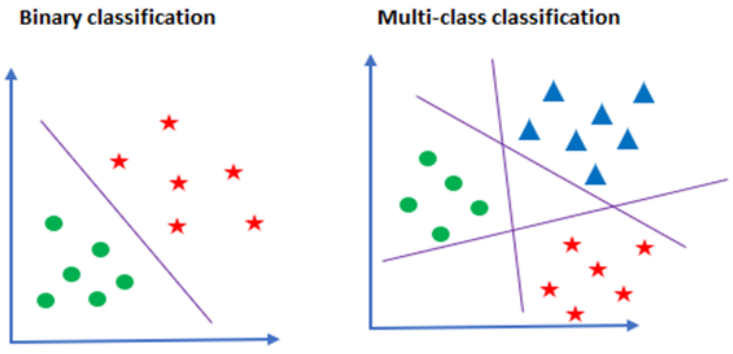


Figure II.26: Exemple d'un SVM One vs One.

II.3.5 Techniques d'implémentation dans différentes architectures matérielles

Les machines à vecteurs de support (SVM) sont des algorithmes d'apprentissage supervisé puissants largement utilisés pour les tâches de classification et de régression.

Cependant, leur implémentation efficace sur différentes architectures matérielles peut s'avérer complexe en raison de leur nature quadratique et de leur calcul intensif.

Voici quelques techniques d'implémentation courantes des SVM sur différentes architectures matérielles :

II.3.5.1 Implémentation sur CPU

- Les CPU offrent une bonne flexibilité et une programmabilité, mais leur performance est limitée par le nombre de cœurs et la vitesse d'horloge.
- Des optimisations logicielles comme le parallélisme et la vectorisation peuvent améliorer les performances, mais elles restent généralement inférieures aux implémentations dédiées.

II.3.5.2 Implémentation sur GPU

- Les GPU (Graphics Processing Units) sont des processeurs parallèles hautement optimisés pour les calculs matriciels, ce qui les rend idéaux pour les SVM.
- Leur architecture massivement parallèle permet d'accélérer considérablement les calculs quadratiques des SVM.
- Des bibliothèques logicielles comme CUDA et OpenCL facilitent l'implémentation des SVM sur GPU.

II.3.5.3 Implémentation sur FPGA

- Les FPGA (Field-Programmable Gate Arrays) sont des circuits intégrés reconfigurables qui peuvent être programmés pour implémenter des algorithmes spécifiques.
- Offrant une flexibilité et une efficacité énergétique accrues par rapport aux CPU et GPU, les FPGA sont adaptés aux applications embarquées et aux systèmes à contraintes de temps réel.
- Des outils de développement comme Vivado et Quartus facilitent la programmation des FPGA pour les SVM.

II.3.5.4 Implémentation sur ASIC

- Les ASIC (Application-Specific Integrated Circuits) sont des circuits intégrés personnalisés conçus pour une application spécifique.
- Offrant les meilleures performances et l'efficacité énergétique, les ASIC sont idéaux pour les applications à grand volume et aux exigences de performance critiques.

Cependant, leur conception et leur fabrication sont coûteuses et moins flexibles que les autres solutions.

II.3.5.5 Implémentations hybrides

- Les implémentations hybrides combinent différentes architectures matérielles pour tirer parti de leurs avantages respectifs.
- Par exemple, un CPU peut être utilisé pour prétraiter les données et gérer les tâches de communication, tandis qu'un GPU peut accélérer les calculs SVM.

Le choix de l'architecture matérielle la plus adaptée dépend de divers facteurs tels que les performances requises, les contraintes de budget, la disponibilité des ressources et la complexité de l'implémentation.

En général, les GPU offrent un bon compromis entre performances et coût, tandis que les FPGA et les ASIC sont mieux adaptés aux applications critiques en termes de performance ou d'efficacité énergétique.

Il est important de noter que le domaine de l'apprentissage automatique et de l'implémentation matérielle des SVM évolue rapidement, et de nouvelles techniques et architectures émergent continuellement.

II.4 Le choix des paramètres optimaux

La réalisation d'un programme d'apprentissage par SVM implique de résoudre un problème d'optimisation dans un espace de grande dimension. L'utilisation efficace des SVM repose principalement sur la sélection d'une bonne fonction noyau et sur le réglage des paramètres de cette fonction. Le but est de trouver les paramètres qui maximisent la performance. Bien que l'implémentation d'un algorithme SVM soit généralement rapide, la recherche des meilleurs paramètres peut nécessiter des phases de test prolongées[17].

II.4.1 Le rôle du paramètre de régularisation lambda

Dans l'algorithme de (SVM), le paramètre de régularisation lambda λ (souvent noté *C* dans la littérature) joue un grand rôle dans l'équilibre entre la maximisation de la marge entre les classes et la minimisation des erreurs de classification.

II.4.1.1 Rôle de la régularisation dans SVM

- **Maximisation de la marge**

L'objectif principal d'un SVM est de trouver un hyperplan qui sépare les données de deux classes avec la marge la plus large possible. La marge est définie comme la distance entre les points de données les plus proches de l'hyperplan (appelés vecteurs de support) et l'hyperplan lui-même. Une plus grande marge correspond généralement à une meilleure généralisation du modèle.

- **Minimisation des erreurs de classification**

Dans des situations réelles, les données ne sont souvent pas linéairement séparables sans erreur. Il est donc nécessaire de permettre quelques erreurs de classification pour obtenir un modèle plus généralisable. C'est là qu'intervient le paramètre de régularisation.

II.4.1.2 Paramètre de régularisation lambda λ ou C

- **Grand *C* (ou petit lambda λ)**
  - Lorsque *C* est grand, le SVM va chercher à minimiser les erreurs de classification, même au détriment de la marge. Cela signifie que le modèle sera strict et tentera de classer correctement le plus grand nombre de points possible, même si cela réduit la marge entre les classes.
  - Cela peut conduire à un sur apprentissage (overfitting), où le modèle s'ajuste trop aux données d'entraînement et généralise mal sur de nouvelles données.
- **Petit *C* (ou grand lambda λ)**
  - Lorsque *C* est petit, le SVM va permettre plus d'erreurs de classification, ce qui permet d'obtenir une marge plus large. Le modèle sera alors plus souple et tolérant envers les erreurs de classification dans les données d'entraînement.
  - Cela peut conduire à un sous-apprentissage (underfitting), où le modèle ne capte pas suffisamment les particularités des données d'entraînement, mais peut mieux généraliser sur de nouvelles données.

Le paramètre de régularisation lambda λ ou *C* dans l'algorithme SVM contrôle donc le compromis entre la maximisation de la marge et la minimisation des erreurs de classification.

En ajustant ce paramètre, on peut influencer la complexité et la performance de généralisation du modèle SVM.

II.4.1.3 Taux d'apprentissage

Le taux d'apprentissage (η) dans les algorithmes de descente de gradient, y compris les variantes utilisées pour entraîner des (SVM) comme la méthode du sous-gradient stochastique (SGD), est un hyperparamètre crucial qui contrôle la taille des pas effectués dans la direction du gradient lors de la mise à jour des poids du modèle.

- **Valeur typique**
  - Généralement, le taux d'apprentissage est un petit nombre positif, souvent compris entre 0 et 1. Des valeurs typiques peuvent être 0.001 ; 0.01 ; 0.1, etc.
- **Taux d'apprentissage supérieur à 1**
  - Un taux d'apprentissage supérieur à 1 est rare, mais il n'est pas théoriquement interdit. Toutefois, dans la pratique, un taux d'apprentissage trop élevé peut causer des problèmes de convergence.
  - Un taux d'apprentissage supérieur à 1 signifie que les mises à jour des poids peuvent être très grandes, ce qui peut entraîner des oscillations importantes ou même une divergence de l'algorithme (c'est-à-dire que l'algorithme ne converge pas vers une solution optimale, mais s'éloigne de celle-ci).

II.4.1.4 Problèmes potentiels avec un taux d'apprentissage élevé

- **Oscillations**

Si le taux d'apprentissage est trop élevé, les mises à jour des poids peuvent dépasser l'optimum et provoquer des oscillations autour de l'optimum plutôt que de converger vers lui.

- **Divergence**

Dans les cas extrêmes, un taux d'apprentissage trop élevé peut provoquer une divergence, où les valeurs des poids deviennent de plus en plus grandes au lieu de se stabiliser vers une valeur optimale.

Bien que le taux d'apprentissage puisse théoriquement dépasser 1, cela est rarement recommandé dans la pratique en raison des risques d'oscillations et de divergence. Un taux d'apprentissage bien choisi est très important pour assurer la stabilité et l'efficacité de la convergence de l'algorithme de descente de gradient. En général, il est préférable de tester des valeurs inférieures à 1 et d'ajuster en fonction des résultats observés lors de la validation croisée.

II.5 Conclusion

Dans ce chapitre, nous avons présenté de manière simple et complète les différentes méthodes d'apprentissage en IA, notamment les réseaux de neurones et les (SVM).

Concernant les réseaux de neurones, nous avons décrit les diverses architectures : Les réseaux feedforward, où l'information circule de manière unidirectionnelle des couches d'entrée aux couches de sortie, sans boucle de rétroaction. C'est l'architecture la plus simple et la plus couramment utilisée. Les réseaux récurrents, où les neurones peuvent avoir des connexions en boucle, permettant de prendre en compte des informations passées dans le traitement. Cela les rend particulièrement adaptés aux tâches impliquant du séquentiel, comme le traitement du langage naturel.

Pour les SVM, nous avons expliqué en détail cette méthode de classification basée sur la recherche d'un hyperplan optimal séparant les données. Nous avons traité les cas linéairement séparables ainsi que les cas non linéairement séparables, nécessitant alors l'utilisation de fonctions noyaux (kernel) pour changer d'espace de représentation.

Dans le prochain chapitre, nous nous pencherons sur l'implémentation pratique de ces algorithmes sur la plateforme Jetson.

CHAPITRE III :

IMPLÉMENTATION D'ALGORITHME D'APPRENTISSAGE DANS UNE JETSON NANO

III1 Introduction

Dans ce chapitre, nous allons présenter les outils utilisés dans notre projet, ainsi l'implémentation des algorithmes (Réseaux de neurones et SVM) sur la Nvidia Jetson Nano. Nous effectuerons des tests afin d'évaluer les performances de chaque modèle et de déterminer lequel est le plus adapté à nos besoins.

III.2 Conception matérielle

Pour notre projet, on a utilisé la carte Nvidia Jetson Nano, Au cours du développement de notre système, nous avons utilisé un ensemble de techniques et outils
afin d'améliorer et d'accroitre les performances de notre système tout en faisant fonctionner les différents dispositifs de façon efficace.

Dans les sections qui suivent, nous allons expliquer le processus de réalisation de notre système :

- Implémentation des algorithmes de l'IA sur la carte Jetson Nvidia Nano

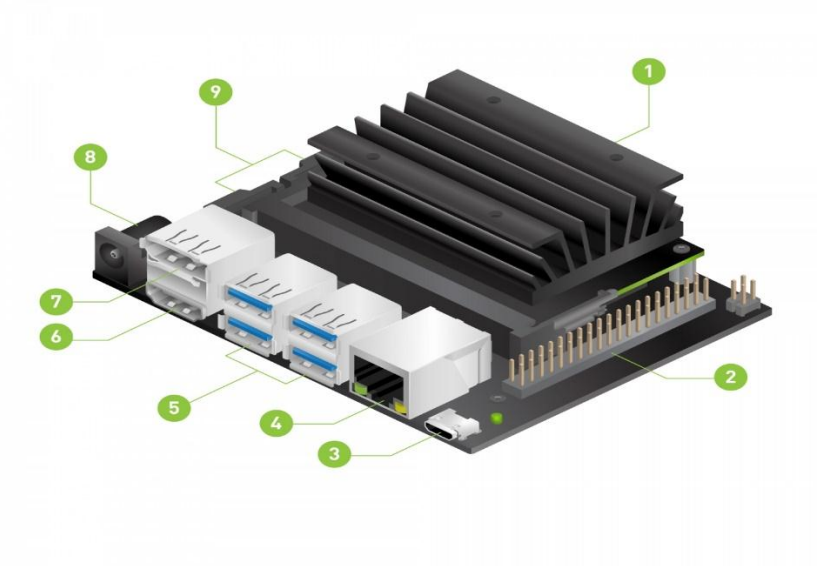
III.2.1 Carte Nvidia Jetson Nano

La carte NVIDIA Jetson Nano est une monocarte d'informatique embarquée développé par NVIDIA, principalement conçu pour les applications d'intelligence artificielle (IA) embarquées et de traitement d'images. Voici une définition détaillée de ses caractéristiques et de ses fonctionnalités :

La Jetson Nano de Nvidia est dotée de :

- Processeur ARM Cortex-A57 quad-core cadencé à 1,43 GHz. Ce processeur est capable de gérer des charges de travail intensives en calcul.
- GPU Maxwell avec 128 cœurs CUDA. Ce GPU offre des performances significatives pour les tâches de traitement d'images et d'apprentissage en profondeur.
- Mémoire de 2 Go de mémoire LPDDR4, ce qui permet de manipuler des ensembles de données volumineux et d'exécuter des applications gourmandes en mémoire.
- **Stockage :** La Jetson Nano intègre un emplacement pour une carte microSD, permettant d'étendre facilement la capacité de stockage.
- **Connectivité :** Elle est équipée de plusieurs ports de connectivité, y compris quatre ports USB 3.0, un port Gigabit Ethernet, un port HDMI et un connecteur pour caméra CSI (Camera Serial Interface). Ces options de connectivité facilitent l'intégration avec divers périphériques et accessoires.

- **Puissance :** Malgré sa petite taille, la Jetson Nano est capable de fournir des performances de calcul remarquables tout en consommant peu d'énergie, ce qui en fait un choix populaire pour les applications embarquées et mobiles.
- **Support logiciel** : NVIDIA fournit un ensemble complet d'outils logiciels pour le développement sur la Jetson Nano, y compris le kit de développement logiciel (SDK) JetPack, qui comprend des bibliothèques d'apprentissage profond telles que TensorFlow, PyTorch et Caffe, ainsi que des bibliothèques de vision par ordinateur comme OpenCV[18].



1 Emplacement pour carte microSD pour le stockage principal
2 Connecteur d'extension à 40 broches
3 Port micro-USB pour une entrée d'alimentation 5 V ou pour le mode appareil
4 Port Ethernet Gigabit
5 Ports USB 3.0 (x4)
6 Port de sortie HDMI
7 Connecteur DisplayPort
8 Prise DC Barrel pour entrée d'alimentation 5V
9 Connecteurs de caméra MIPI CSI-2

Figure III.1 : Schéma descriptif des différents composants de Jetson Nano de Nvidia.

III.2.1.1 Caractéristiques de la Nvidia Jetson

Le tableau suivant illustre les caractéristiques de la jetson :

Caractéristique	Détail
Alimentation	- 5 Vcc/3 A via USB Type-C - 5 Vcc via port GPIO
Consommation	- Via alimentation externe sur connecteur d'alim : 4.4 A maxi - Via connecteur micro-USB : 2 A maxi - Via connecteur GPIO - Jetson Nano uniquement : ajustable sur 5 ou 10 W
Microprocesseur	ARM Cortex-A57 4 cœurs à 1,43 GHz
GPU	NVIDIA Maxwell avec 128 cœurs CUDA®
Mémoire RAM	2 Go LPDDR4
Stockage	microSD (carte non incluse)
Encodage vidéo (H.264 et H.265)	- 4K à 30 IPS - 4 x 1080p à 30 IPS - 9 x 720p à 30 IPS
Décodage vidéo (H.264 et H.265)	- 4K à 60 IPS - 2 x 4K à 30 IPS - 8 x 1080p à 30 IPS - 18 x 720p à 30 IPS
Interface caméra	MIPI CSI-2
Port RJ45	Ethernet Gigabit
Ports vidéo	HDMI et DisplayPort
Module WiFi	Intégré
Ports USB	- 1 x USB 3.0 - 2 x USB 2.0 - 1 x micro-USB 2.0
Interfaces	- E/S digitales 3,3 Vcc - Bus I2C, I2S, SPI et UART
Indicateurs	LED d'alimentation
Connecteurs supplémentaires	- Connecteurs pour boutons-poussoirs (non inclus, arrêt, reset, etc.) - Connecteur pour ventilateur (non inclus)
Dimensions	- Carte : 100 x 80 x 29 mm - Module Jetson Nano : 70 x 45 mm
Version	2 GB

Tableau III-1 : Caractéristiques de la Jetson Nvidia utilisée.

III.3 Conception logicielle

III.3.1 Configuration de la carte

Voici un résumé des principales étapes pour configurer la Jetson NVIDIA Nano :

- Préparation du matériel nécessaire, incluant une carte microSD, des périphériques USB (comme un clavier et une souris), et une alimentation adaptée.
- Télécharger l'image du système d'exploitation JetPack depuis le site officiel de NVIDIA.

- Utiliser un outil comme Balena Etcher pour flasher l'image sur la carte microSD.
- Insérer la carte microSD dans la Jetson Nano, connecter les périphériques et le moniteur, puis brancher l'alimentation.
- Lors du premier démarrage, suivre les instructions à l'écran pour configurer les paramètres de base : Langue, Fuseau horaire, Compte utilisateur et Connexion réseau.
- Mettre à jour le système avec les commandes :sudo apt update sudo apt upgrade.
- Installer les outils et bibliothèques nécessaires pour le développement d'IA.
- Configurer les paramètres de performance avec nvpmodel et vérifier l'état actuel avec jetson_clocks.
- Tester la configuration en exécutant des exemples d'IA pour s'assurer que tout fonctionne correctement.

III.3.2 Implémenter un réseau de neurones sur la Nvidia Jetson

Mettre en œuvre un réseau de neurones sur une Nvidia Jetson peut paraître complexe au premier abord, mais en suivant des étapes structurées, cela devient beaucoup plus agréable.

Voici les étapes détaillées pour réaliser cette implémentation :

III.3.2.1 Préparation de l'environnement

- **Installer JetPack**

Téléchargez et installez le JetPack SDK depuis le site officiel de NVIDIA, car il inclut toutes les bibliothèques et outils nécessaires pour le développement sur Jetson. Ensuite, suivez les instructions fournies pour flasher la Jetson avec JetPack afin de préparer l'environnement de développement.

- **Configurer le Jetson**

Après avoir connecté le Jetson Nano à un moniteur, un clavier, une souris et à Internet, on a effectué les instructions de configuration initiale, notamment la création de notre compte utilisateur et la configuration réseau.

- **Installation des outils et bibliothèques nécessaires :**
  - Installer les packages Python de base (numpy, scipy, etc.)
  - Installer un framework d'apprentissage profond comme TensorFlow, PyTorch ou Keras.
- **Configuration de l'environnement de développement :**
  - Créer un environnement virtuel Python pour isoler les dépendances.
  - Installer les bibliothèques spécifiques au développement IA (ex : CUDA, cuDNN).
- **Conception du modèle de réseau de neurones :**

- Définir l'architecture du réseau (couches, hyperparamètres, etc.)
- Entraîner le modèle sur les données d'apprentissage.

➢ **Optimisation et évaluation du modèle :**

- Affiner les hyperparamètres pour améliorer les performances.
- Évaluer les performances du modèle sur les données de test.

➢ **Déploiement et intégration :**

- Convertir le modèle en un format compatible avec la Jetson Nano.
- Intégrer le modèle dans une application ou un service en temps réel.

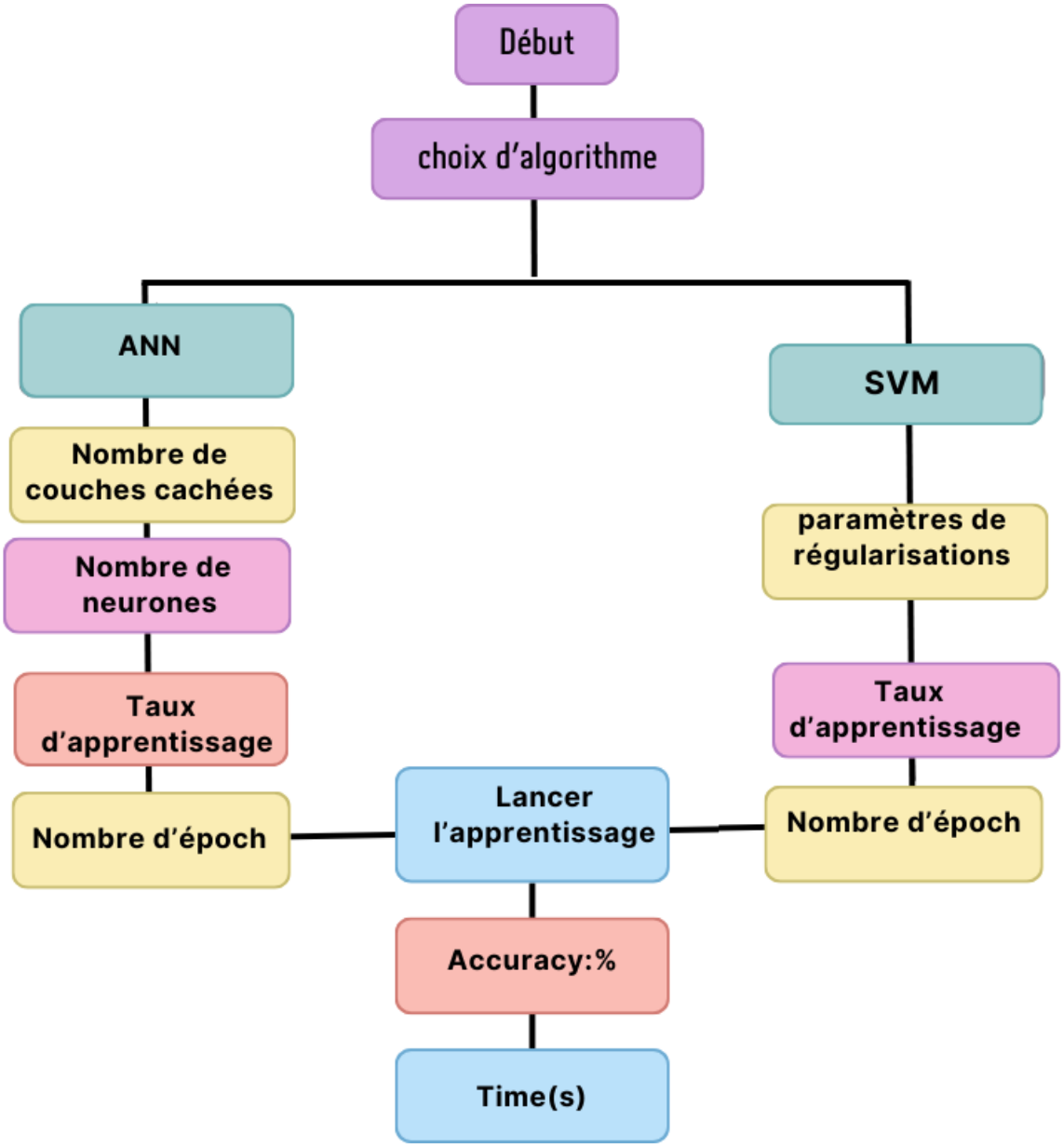
➢ **Tests et débogage :**

- Vérifier le bon fonctionnement du modèle déployé sur la Jetson Nano.
- Résoudre les éventuels problèmes de performances ou de stabilité.

III.3 L'implémentation de l'algorithme SVM sur la Jetson Nvidia Nano

Pour implémenter le classifieur SVM linéaire, on a utilisé la descente de gradient pour l'entraînement. On commence par initialiser les paramètres, à chaque itération d'entraînement, met à jour les poids *w* et le biais *b* en fonction des exemples de données, en appliquant une régularisation pour améliorer la généralisation du modèle. Tous les 100 epochs, il calcule et affiche la perte hinge pour surveiller la performance. Dans le programme on a inclut aussi une fonction de prédiction qui utilise les poids et le biais appris pour prédire les étiquettes des données, et évalue la précision du modèle en comparant les prédictions aux étiquettes réelles des données d'entraînement.

L'organigramme suivant résume l'implémentation des deux algorithmes (ANN, SVM) :



L'organigramme l'implémentation des deux algorithmes (ANN, SVM)

La figure III.3 montre une session de terminal Linux, où le script affiche un message de bienvenue et propose deux options : appuyer sur la lettre 'o' pour sélectionner une base de données, ou sur 'f' pour fermer l'application.

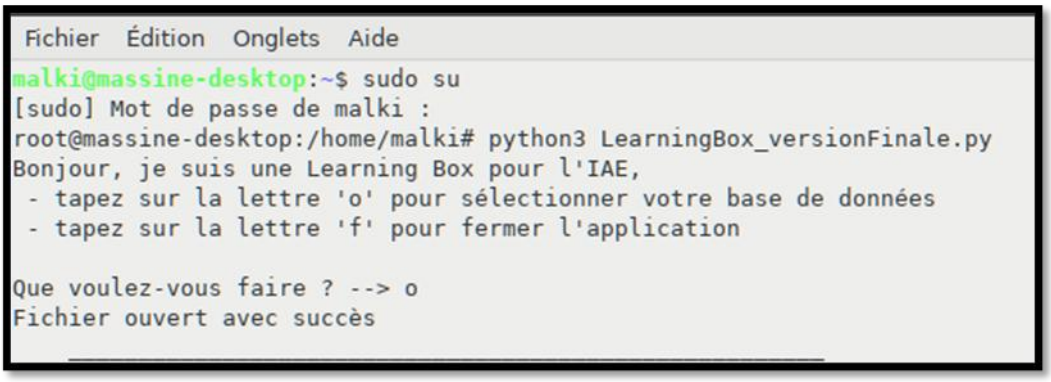


Figure III.3 : Lancement de l'application.

Après que l'utilisateur a ouvert un fichier avec succès dans le script LearningBox_versionFinale.py. Le script affiche les noms des colonnes du fichier de base de données sélectionné, qui incluent les entrées : 'Temp', 'PRN', 'DOL', etc., et des sorties : 'Inflammation de la vessie (Cystite)' et 'Néphrite d'origine pelvienne'. L'utilisateur est invité à entrer le nombre de variables d'entrée et de sortie pour la base de données. Ensuite, il spécifie les noms des colonnes pour les entrées et les sorties respectivement.

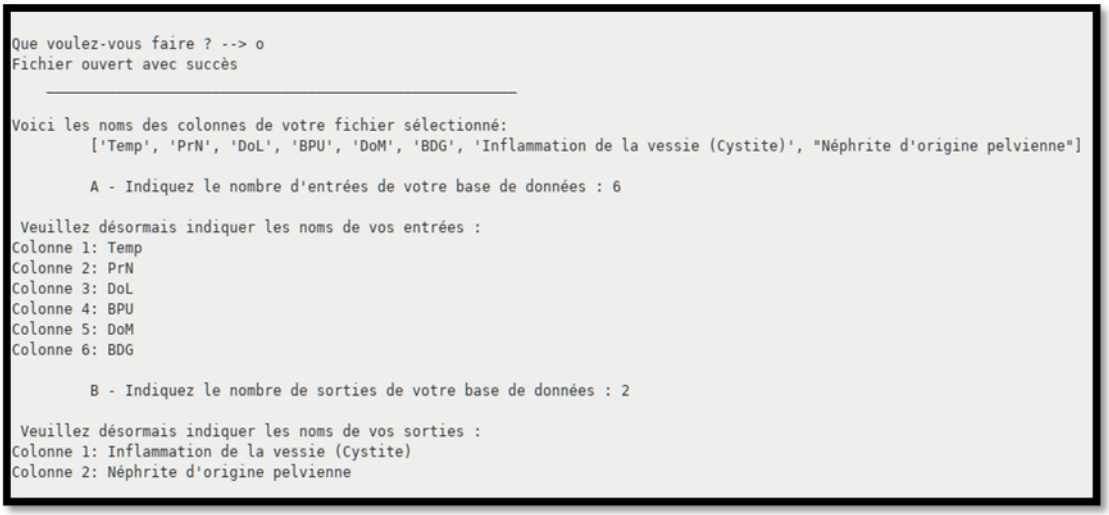


Figure III.4 : Extraction des données du fichier Excel

Après que l'utilisateur a spécifié les noms des sorties : 'Inflammation de la vessie (Cystite)' et 'Néphrite d'origine pelvienne', a choisi d'utiliser un réseau de neurones artificiels (ANN) comme classifieur, et a ensuite entré les paramètres du réseau incluant 3 neurones dans la couche cachée, un taux d'apprentissage de 0,01 et 10 000 époques pour l'entraînement.

```
 Veuillez désormais indiquer les noms de vos sorties :
Colonne 1: Inflammation de la vessie (Cystite)
Colonne 2: Néphrite d'origine pelvienne

        C - Sélectionner le type de classifieur à utiliser :
                (1) - ANN -- Réseau de neurones artificiels
                (2) - SVM -- Support Vector Machine

                Indiquez votre Choix : 1
        D - Introduisez les paramètres de votre réseau :
        Nombre de neurones dans la couche cachée: 3
        Taux d'apprentissage: 0.01
        Nombre d'époques: 10000
```

Figure III.5 : Sélection de l'algorithme et paramétrage

Enfin, l'utilisateur doit choisir une fonction d'activation pour les neurones de la couche cachée (sigmoid) et de la couche de sortie (également sigmoid).

```
        C - Sélectionner le type de classifieur à utiliser :
                (1) - ANN -- Réseau de neurones artificiels
                (2) - SVM -- Support Vector Machine

                Indiquez votre Choix : 1
        D - Introduisez les paramètres de votre réseau :
        Nombre de neurones dans la couche cachée: 3
        Taux d'apprentissage: 0.01
        Nombre d'époques: 10000

 Veuillez choisir la fonction d'activation (couche cachée) :
relu / sigmoid / tanh  -->  sigmoid

 Veuillez choisir la fonction d'activation (couche sortie) :
relu / sigmoid / tanh  -->  sigmoid
Epoch 0, Loss: 0.3619542251616075
Epoch 100, Loss: 0.08137855133646278
Epoch 200, Loss: 0.019584232850519134
Epoch 300, Loss: 0.007991972735483846
```

Figure III.6 : choix des fonctions d'activations

Ce qui est présenté dans la figure III.6 sont les résultats du modèle de réseau de neurones artificiels (ANN) qui a été entraîné sur un ensemble de données. Les informations affichées comprennent les pertes ("Loss") à différentes époques ("Epoch") de l'entraînement, ainsi que les poids et le biais final du modèle.

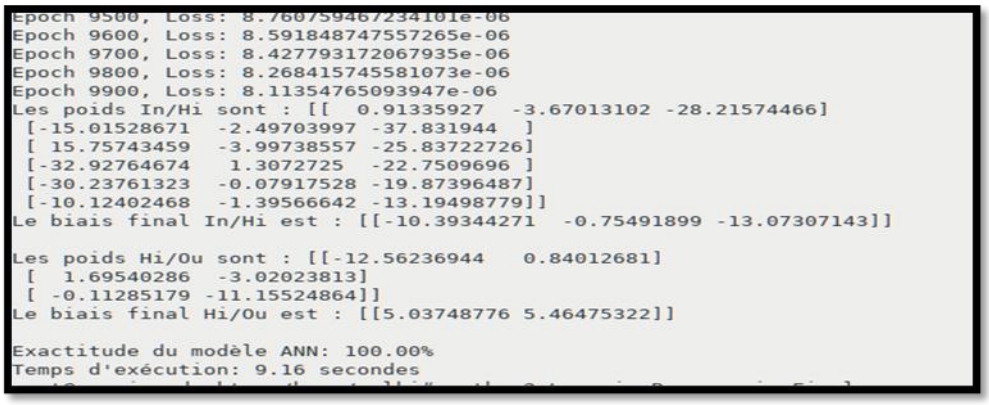


Figure III.7 : Résultats de l'apprentissage et enregistrements des poids

Lorsque l'option du SVM est choisie, le menu demande à l'utilisateur de renseigner les paramètres du modèle SVM, notamment le taux d'apprentissage (0.001) et le paramètre de régularisation (0.1). Le nombre d'époques à effectuer pour l'entraînement est également spécifié, ici 100.

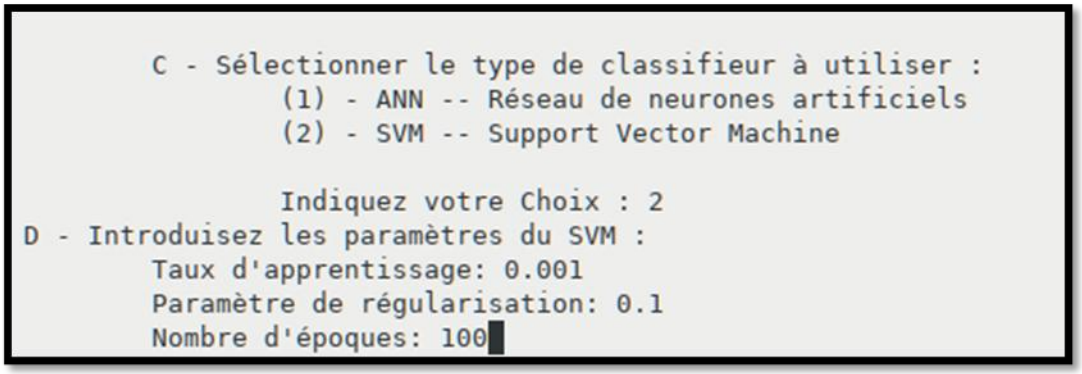


Figure III.8 : Choix du SVM comme algorithme d'apprentissage

La figure III.8 présente les résultats du modèle SVM) qui a été entraîné sur un ensemble de données. Les informations affichées incluent les pertes ("Marge Loss") à différentes époques ("Epoch") de l'entraînement, ainsi que les poids et le biais final du modèle.

Les valeurs numériques indiquent les performances du modèle au fil des itérations d'entraînement. L'exactitude finale du modèle SVM est également affichée, avec une précision de 85,71 %.

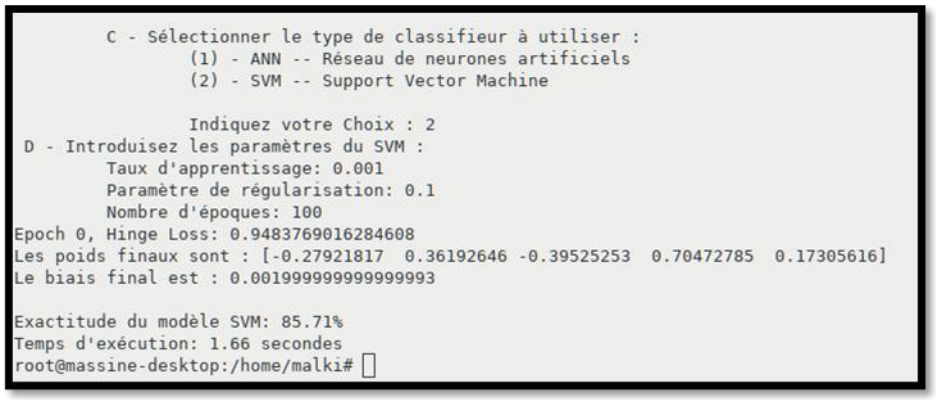


Figure III.9 : Résultats de l'apprentissage du SVM

Le tableau suivant illustre les variations du nombre de neurones dans la couche cachée

Nombre d'époques	L_Rate	Nombre de Neurone dans la couche cachée	Accuracy	Time (s)
10000	0,01	1	50,00%	6,36
10000	0,01	3	78,57%	8,31
10000	0,01	5	29%	8,06
10000	0,01	7	42,86%	7,59
10000	0,01	9	100,00%	9,12
10000	0,01	11	100,00%	14,54

Tableau III.2 : Les différents résultats obtenus avec le nombre de neurones dans la couche cachée.

Nombre de Neurones et la précision (Accuracy)

- Avec un seul neurone dans la couche cachée, la précision n'était que de 50%, indiquant une performance médiocre. En augmentant le nombre de neurones à 3, la précision s'est améliorée de manière significative, atteignant 78.57%. Cependant, en passant à 5 neurones, la précision a chuté à 29%, ce qui suggère une instabilité ou un surapprentissage du modèle avec cette configuration. Avec 7 neurones, la précision s'est améliorée à 42.86%, bien que toujours faible. Enfin, les configurations à 9 et 11 neurones ont atteint une précision parfaite de 100%, montrant que le modèle a parfaitement appris à classer les données d'entraînement avec ces architectures plus complexes

Nombre de Neurones et Temps d'Entraînement

- Le temps d'entraînement augmente globalement avec le nombre de neurones dans la couche cachée, bien que les variations ne soient pas strictement monotones. Plus de neurones nécessitent plus de calculs, ce qui augmente le temps d'entraînement

Observation

L'ajout de plus de neurones dans la couche cachée améliore généralement la précision du modèle jusqu'à un certain point.9 neurones semblent être suffisants pour atteindre une précision parfaite (100%) dans ce cas particulier.

Une augmentation au-delà de 9 neurones (jusqu'à 11) maintient la précision à 100%, mais pourrait ne pas être nécessaire si 9 neurones suffisent.

Le temps d'entraînement tend à augmenter avec le nombre de neurones, ce qui est attendu car plus de neurones impliquent plus de calculs.

Le temps est le plus long pour 11 neurones, à 14.54 secondes, contre 9.12 secondes pour 9 neurones.

Recommandation

Pour optimiser le modèle en termes de précision et de temps d'entraînement, il est recommandé d'utiliser 9 neurones dans la couche cachée. Cette configuration atteint une précision de 100% avec un temps d'entraînement raisonnable. Ajouter plus de neurones n'améliore pas la précision mais augmente le temps d'entraînement.

Nobre d'époques	Learning_Rate	Nombre de Neurone dans la couche cachée	Accuracy	Time (s)
10000	0,0001	9	100,00%	8,74
10000	0,001	9	100,00%	7,84
10000	0,01	9	100,00%	12,72
10000	0,1	9	100,00%	12,36
10000	0,5	9	100,00%	12
10000	0,9	9	97,38	10,33

Tableau III.3 : Les différents résultats obtenus avec le changement du taux d'apprentissage (l_Rate)

Remarque

On peut observer que lorsque le taux d'apprentissage (learning rate) est fixé à 0,0001, le modèle atteint une précision (accuracy) de 100% avec 9 neurones dans la couche cachée, et ce en un temps d'exécution de 8,74 secondes. En augmentant le taux d'apprentissage à 0,01, 0,1 et 0,5, on constate que la précision reste élevée, autour de 100%, mais le temps d'exécution augmente significativement, allant jusqu'à 12,36 secondes pour le taux le plus élevé de 0,5. Enfin, avec un taux d'apprentissage de 0,9, la précision diminue légèrement à 97,38%, mais le temps d'exécution est réduit à 10,33 secondes. Dans l'ensemble, cette analyse montre l'importance du choix judicieux du taux d'apprentissage et du nombre de neurones pour obtenir des performances optimales tout en limitant le temps de calcul.

L'algorithme SVM

Nombre d'époch	Learning_Rate	Paramètre de régularisation lambda	Accuracy	Time (s)
10000	0,00001	0,0001	100,00%	114,64
10000	0,001	0,0001	100,00%	90,89
10000	0,1	0,0001	100,00%	91,94
10000	0,5	0,0001	100,00%	91
10000	0,9	0,0001	100,00%	91,56
10000	1	0,0001	100,00%	92,12

Tableau III.4 : variation du taux d'apprentissage.

Analyse des résultats

Précision (Accuracy)

- Le modèle atteint une précision de 100% pour toutes les valeurs du Taux d'apprentissage (learning_rate) testées. Cela suggère que, pour cet ensemble de données et ces paramètres, le modèle est robuste au choix du learning rate, avec une régularisation lambda fixée à 0.0001.

Temps d'Entraînement

- Le temps d'entraînement est le plus élevé pour le Learning_rate le plus faible (0.00001). Cela pourrait être dû au fait qu'un Learning_rate trop faible conduit à des ajustements très petits des poids, nécessitant plus de temps pour converger.
- Pour les autres valeurs de Learning_rate (0.001 à 1), le temps d'entraînement reste relativement stable, suggérant une convergence plus rapide comparée au Learning_rate très faible.

Pour optimiser le temps d'entraînement sans sacrifier la précision, il est recommandé d'utiliser un learning rate plus élevé que 0.00001, tout en restant dans une plage où le modèle converge rapidement, comme 0.001 à 1. Une analyse plus fine pourrait affiner davantage cette plage pour des performances maximales avec un temps d'entraînement raisonnable

Nombre D'epoch	Learning_Rate	Paramètre de régularisation lambda	Accuracy	Time (s)
10000	0,001	0,0001	100,00%	14.7882
10000	0,001	0,001	100,00%	15.8537
10000	0,001	0,1	100,00%	21.4666
10000	0,001	0,5	100,00%	27.0344
10000	0,001	1	100,00%	32.1007
10000	0,001	5	33,33%	36.8367

Tableau III.5 : variation du lambda.

Analyse des résultats

Constantes

- Le nombre d'époques et le Taux d'apprentissage (learning_rate) sont constants pour toutes les expérimentations (10,000 et 0.001 respectivement).

Impact de lambda sur la précision (l'Accuracy)

- Pour les valeurs de lambda allant de 0.0001 à 1, l'accuracy est de 100%, ce qui signifie que le modèle a parfaitement appris à classer les données d'entraînement pour ces valeurs de lambda.
- Pour lambda = 5, la précision (l'accuracy) tombe à 33.33%, indiquant une mauvaise performance du modèle. Cela pourrait être dû à une régularisation excessive, pénalisant trop fortement les poids du modèle et conduisant à un sous-apprentissage.

Influence de lambda sur le Temps

- Le temps nécessaire pour l'entraînement augmente avec l'augmentation de lambda. Cela peut être dû à des calculs plus intensifs requis pour des valeurs plus élevées de régularisation.

Discussion

- Le modèle atteint une haute précision (100%) pour des valeurs de lambda de 0.0001 à 1, mais la précision chute fortement pour lambda = 5, suggérant une régularisation excessive.
- Le temps d'entraînement augmente avec l'augmentation de lambda, probablement à cause de la complexité accrue des calculs de régularisation.
- Pour une balance optimale entre précision et temps d'entraînement, il est recommandé d'utiliser une valeur de lambda entre 0.0001 et 1. Une analyse plus fine pourrait affiner davantage cette plage pour des performances maximales avec un temps d'entraînement raisonnable.
- Cela suggère que pour éviter le surapprentissage tout en maintenant une haute précision et un temps d'entraînement raisonnable, il est crucial de bien ajuster le paramètre de régularisation lambda.

III.4 Conclusion

Dans ce chapitre, nous avons présentés les outils utilisés dans notre projet ainsi que l'implémentation des algorithmes (réseaux de neurones et SVM) sur la Nvidia Jetson Nano. On a effectué des tests pour évaluer les performances de chaque modèle et déterminer lequel est le plus adapté à nos besoins. Les résultats se sont avérés très satisfaisants sur les deux

différents algorithmes. On a démontré qu'avec cette application, on peut effectuer n'importe quel type d'apprentissage en ligne sur n'importe quelle base de données.

CONCLUSION GENERALE

Ce mémoire a été consacré à l'implémentation et au développement des réseaux de neurones et SVM dans une Nvidia Jetson Nano, un domaine clé de l'intelligence artificielle. Avant d'établir leurs principes de base, leurs utilités et leurs domaines d'application.

La première partie intègre l'Intelligence artificielle (IA) qui est un domaine vaste et dynamique, incluant des disciplines telles que le traitement d'images, le traitement du langage naturel et la gestion des bases de données. Au cœur de l'IA se trouve l'apprentissage automatique, une méthode qui permet aux machines d'apprendre à partir de données sans avoir besoin d'une programmation explicite.

Le deuxième volet du travail a consisté aux différentes techniques d'apprentissage, dont trois types d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Ces dernières sont essentielles pour comprendre comment l'IA fonctionne et comment elle peut être utilisée pour résoudre des problèmes complexes et détecter chaque erreur. On a fait une étude détaillée sur les réseaux de neurones et les machines à vecteurs de support (SVM).

On a fini avec la partie pratique, où nous avons présenté les outils utilisés dans notre projet ainsi que l'implémentation des algorithmes de réseaux de neurones et de SVM sur la Nvidia Jetson Nano. Nous avons effectué des tests pour évaluer les performances de chaque modèle et déterminer lequel est le plus adapté à nos besoins. Les résultats obtenus avec les deux algorithmes se sont avérés très satisfaisants. Nous avons démontré que cette application permet de réaliser n'importe quel type d'apprentissage en ligne sur n'importe quelle base de données.

Comme perspectives de cette application, plusieurs axes d'amélioration et d'optimisation sont inclus :

- Optimisation avec TensorRT, pour optimiser les réseaux de neurones permettrait d'améliorer la vitesse d'inférence et l'efficacité énergétique, exploitant pleinement les capacités de la Jetson Nano
- Parallelisation et Pipeline d'Inference : Optimiser le pipeline d'inférence en parallélisant les calculs et en exploitant au maximum les ressources GPU et CPU de la Jetson Nano, réduisant ainsi le temps de traitement.
- Intégration de Nouvelles Bibliothèques : Tester et intégrer de nouvelles bibliothèques et frameworks dédiés à l'IA embarquée, comme NVIDIA DeepStream, pour améliorer l'efficacité du traitement des flux de données en temps réel.
- Applications en Temps Réel : Développer et tester des applications en temps réel, telles que la reconnaissance d'objets, la détection d'anomalies, et la classification d'images, pour démontrer les capacités pratiques des algorithmes sur la Jetson Nano.

Ces perspectives ouvrent la voie à des applications plus performantes, efficaces et adaptées aux contraintes des systèmes embarqués, exploitant pleinement le potentiel de la Nvidia Jetson Nano.

REFERENCES BIBLIOGRAPHIQUES

[1] G. Massine, « Implémentation d'un réseau de neurones dans un microcontrôleur »,

PhD Thesis, Université Mouloud Mammeri, 2016. Consulté le: 4 février 2024. [En ligne]. Disponible sur: https://dspace.ummto.dz/items/8c6e0b2f-fe23-43ce-85f1-a2e4e044335e

[2] « Jetson Nano apporte toute la puissance de l'IA moderne aux systèmes Edge | NVIDIA ». Consulté le: 25 juin 2024. [En ligne]. Disponible sur: https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-nano/product-development/

[3] G. Briganti, « Intelligence artificielle : une introduction pour les cliniciens », Rev. Mal. Respir., vol. 40, no 4, p. 308‑313, avr. 2023, doi: 10.1016/j.rmr.2023.02.005.

[4] « John McCarthy | Biography & Facts | Britannica ». Consulté le: 21 juin 2024. [En ligne]. Disponible sur: https://www.britannica.com/biography/John-McCarthy

[5] A. française, « Dictionnaire de l'Académie française ». Consulté le: 21 juin 2024. [En ligne]. Disponible sur: http://www.dictionnaire-academie.fr/

[6] « Dictionnaire français - Dictionnaires Larousse français monolingue et bilingues en ligne ». Consulté le: 21 juin 2024. [En ligne]. Disponible sur: https://www.larousse.fr/dictionnaires/francais

[7] R. Amellal, « Application des réseaux de neurones artificielles dans l'optimisation des performances d'antennes reconfigurables », Rev. Int. Cherch., vol. 4, no 3, 2023, Consulté le: 4 février 2024. [En ligne]. Disponible sur: https://revuechercheur.com/index.php/home/article/view/674

[8] J. Lucas, « IA embarquée : de quoi parle-t-on ? », Neovision. Consulté le: 13 février 2024. [En ligne]. Disponible sur: https://neovision.fr/quest-ce-que-lia-embarquee/

[9] J. Robert, « Machine Learning vs Deep Learning : Quelles différences ? », Formation Data Science | DataScientest.com. Consulté le: 15 février 2024. [En ligne]. Disponible sur: https://datascientest.com/quelle-difference-entre-le-machine-learning-et-deep-learning

[10] B. L, « Algorithme : mais en fait, qu'est-ce que c'est et à quoi ça sert ? », LEBIGDATA.FR. Consulté le: 18 mars 2024. [En ligne]. Disponible sur: https://www.lebigdata.fr/algorithme-definition-tout-savoir

[11] « Les principales méthodes d'apprentissage en Intelligence Artificielle (IA) », PandIA. Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://pandia.pro/guide/les-principales-methodes-dapprentissage-en-ia/

[12] « Les trois méthodes d'apprentissage clés en Intelligence Artificielle », KodKodKod Studio. Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://kodkodkod.studio

[13] M. Abadi, « Réalisation d'un réseau de neurones "« SOM »" sur une architecture matérielle adaptable et extensible à base de réseaux sur puce " « NoC »" ».

[14] « (99+) Architecture et programmation des Microcontrôleurs | houssin ait alè - Academia.edu ». Consulté le: 1 avril 2024. [En ligne]. Disponible sur: https://www.academia.edu/41389753/Architecture_et_programmation_des_Microcontr%C3%B4leurs?auto=download&email_work_card=download-paper

[15] G. B. Hacene, « Implementation d'un Réseau de Neurones sur FPGA », 2016.

[16] N. Verzelen, J. Salmon, et P. Pudlo, « Support Vector Machines (SVM) et méthodes à Noyaux ».

[17] Z. Zidelmal, « Reconnaissance d'arythmies cardiaques par Support Vector Machines (SVMs) ».

[18] G. TRONIC, « Jetson Nano Developer Kit », GO TRONIC. Consulté le: 25 juin 2024. [En ligne]. Disponible sur: https://www.gotronic.fr/art-jetson-nano-developer-kit-31717.htm