الجـــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Automatique

Spécialité : Automatique et informatique industrielle

Par : ZIZI Ilies Mohammed-Riad et OUALI CHAOUCHE Mohammed Arsalan

Sujet

Détection de l'activité humaine en utilisant les données inertielles de téléphone mobile

Mme CHOUKCHOU-BRAHAM Amal	Professeur	Université de Tlemcen	Président
Mme WAHIDA Handouzi	Docteur	Université de Tlemcen	Examinateur
M HADJ ABDELKADER Amine	Professeur	Université de Tlemcen	Encadreur

Année universitaire : 2021 / 2022

Remerciement

Tous nos remerciement vont vers tout les personnes qui nous ont aidés et qui ont contribué au succès de notre parcours scolaire et étudiant.

Nos famille, ami et proche, pour leur soutien constant et leurs encouragements.

Nous tenons à exprimer toute notre reconnaissance et un sincère et grand remerciement à notre encadreur de mémoire, Monsieur Amine HADJ ABDELKADER. nous le remercions de nous avoir encadrés, orientés, aidés et conseillés.

Nous remercions aussi tout les professeurs qui ont contribué à notre réussite durant notre parcours scolaire et universitaire.

À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Depuis l'apparition des premiers téléphones mobiles portatifs commerciaux en 1979, nous avons observé une croissance accélérée du marché de la téléphonie mobile qui a atteint en 2011 près de 80 % de la population mondiale [1].

Cela montre qu'en très peu de temps, les dispositifs mobiles sont devenus facilement accessibles à pratiquement tout le monde. Les smartphones, qui sont une nouvelle génération de téléphones mobiles, offrent désormais de nombreuses fonctionnalités en plus de la téléphonie de base telles que le multitâche et le déploiement d'une variété de de capteurs. Les travaux actuels visent à intégrer toutes ces fonctionnalités tout en conservant la durée de vie des batteries et les dimensions du dispositif.

L'intégration de ces dispositifs mobiles dans notre vie quotidienne se développe rapidement. Il est prévu que ces appareils suivent de manière transparente nos activités, en tirent des enseignements et nous aident ensuite à prendre de meilleures décisions concernant nos actions futures [2].

Dans cette étude, nous utilisons les données des capteurs du smartphone pour la reconnaissance de l'activité humaine, avec des applications potentielles dans les technologies d'assistance à la vie quotidienne. La reconnaissance d'activité vise à identifier les actions effectuées par une personne à partir d'un ensemble d'observations d'elle-même et de son environnement. La reconnaissance peut être réalisée, par exemple, en exploitant les informations récupérées à partir de capteurs inertiels tels que les accéléromètres [3]. Dans certains smartphones, ces capteurs sont embarqués par défaut et nous en profitons pour classifier un ensemble d'activités physiques (se tenir debout, s'allonger, marcher, monter les escaliers et descendre les escaliers) en traitant les signaux corporels inertiels par un algorithme d'apprentissage automatique, Nous proposons ainsi de faire une comparaison entre deux familles de méthodes : d'une part les algortithmes usuels de la classification tels que les KNN [1], les arbres de décision et les SVM [2] et, d'autre part le réseau de neurones récurrent LSTM (Long short-term memory), algorthme très utilisé dans le cadre de l'apprentissage automatique.

Le document est structuré de la manière suivante :

- Le chapitre 1 présente un état de l'art de la classification, avec l'introduction des alghorthmes KNN, arbres de décisions et SVM. Nous détaillerons les principes de chaque méthodes et les variantes connues de chacun d'entre-eux. Nous donnerons également un aperçu sur la problématique de détection de mouvement et les différents travaux l'ayant abordé depuis l'avènement des technologies mobiles intelligentes. Les prétraitements nécessaires aux données sont également présentés.

1. K-Nearest Neighbors
2. Support Vector Machine

- Le chapitre 2 donne un état de l'art du réseau récurrent à mémoire court et long terme (LSTM), aprés une bève présentation de l'apprentissage automatique et de ses applications actuelles. Nous présentons la structure du LSTM ainsi que son fonctionnement avec ses différentes cellules d'entrée, de sorties et d'oubli.
- Le chapitre 3 présente la méthodologie adoptée dans ce travail, l'application des différents algorthmes décris précédemment ainsi que les résultats obtenus. Nous avons exploités des bases de données diponibles pour l'usage académique afin d'exécuter chacune de ses méthodes et comparer leurs résultats respectifs. Deux outils différents nous ont facilité la tâche que sont Classification Learner de Matlab, regroupant toutes les variantes des méthodes de classification ; et la librairie TensorFlow, déployé sur le Cloud informatique Google Collab permettant ainsi d'obetnir de bonnes performance en apprentissage et en classifications des données.

Une conclusion générale permet de synthétiser l'ensemble de ce travail.

Chapitre 1

Généralités sur les méthodes la classification automatique

L'apprentissage machine (ML) est un vaste domaine interdisciplinaire qui s'appuie sur des concepts issus de l'informatique, des statistiques, des sciences cognitives, de l'ingénierie, de la théorie de l'optimisation et de nombreuses autres disciplines des mathématiques et des sciences. Les applications de l'apprentissage automatique sont nombreuses, mais l'exploration de données est la plus importante de toutes. L'apprentissage automatique peut être classé en deux grandes catégories : l'apprentissage automatique supervisé et l'apprentissage automatique non supervisé.

L'apprentissage automatique non supervisé est utilisé pour tirer conclusions à partir d'ensembles de données constitués de données d'entrée sans réponses étiquetées. Dans l'apprentissage non supervisé, le résultat souhaité n'est pas donné. Les techniques d'apprentissage automatique supervisé tentent de trouver la relation entre les attributs d'entrée (variables indépendantes) et un attribut cible (variable dépendante). Les techniques supervisées peuvent être classées en deux catégories principales : la classification et la régression. Dans la régression, la variable de sortie prend des valeurs continues tandis que dans la classification, la variable de sortie prend des étiquettes de classe. La classification est une approche d'exploration de données (apprentissage automatique) utilisée pour prévoir l'appartenance à un groupe pour les instances de données. Bien qu'il existe une variété de techniques disponibles pour l'apprentissage automatique, la classification est la technique la plus largement utilisée. la classification possède une grande valeur dans l'apprentissage automatique, en particulier dans la planification future et la découverte de connaissances.

La classification est considérée comme l'un des problèmes les plus étudiés par les chercheurs dans les domaines de l'apprentissage automatique et de l'exploration de données. La figure 3.4 présente un modèle général d'apprentissage supervisé (techniques de classification).

Bien que la classification soit une technique bien connue dans l'apprentissage automatique, elle souffre de problèmes tels que le traitement des données manquantes. Les valeurs manquantes dans un ensemble de données peuvent poser problème pendant les phases de formation et de classification. Certaines des raisons potentielles des données manquantes sont : la non saisie d'un enregistrement en raison d'une non-conception, données reconnues non pertinentes au moment de l'entrée, la suppression de données en raison d'un écart avec d'autres données documentées et un dysfonctionnement de l'équipement.

Le problème des données manquantes peut être surmonté par plusieurs approches tels que : les mineurs de données peuvent ignorer les données manquantes, échanger des valeurs omises entières avec une constante globale individuelle, échanger une valeur omise avec la moyenne de sa caractéristique pour la classe donnée, observer manuellement les échantillons avec valeurs omises et insérer une valeur réalisable ou probable. Dans ce travail, nous nous concentrerons uniquement sur quelques méthodes de classification sélectionnées en adéquation avec l'objectif de travail. [4].

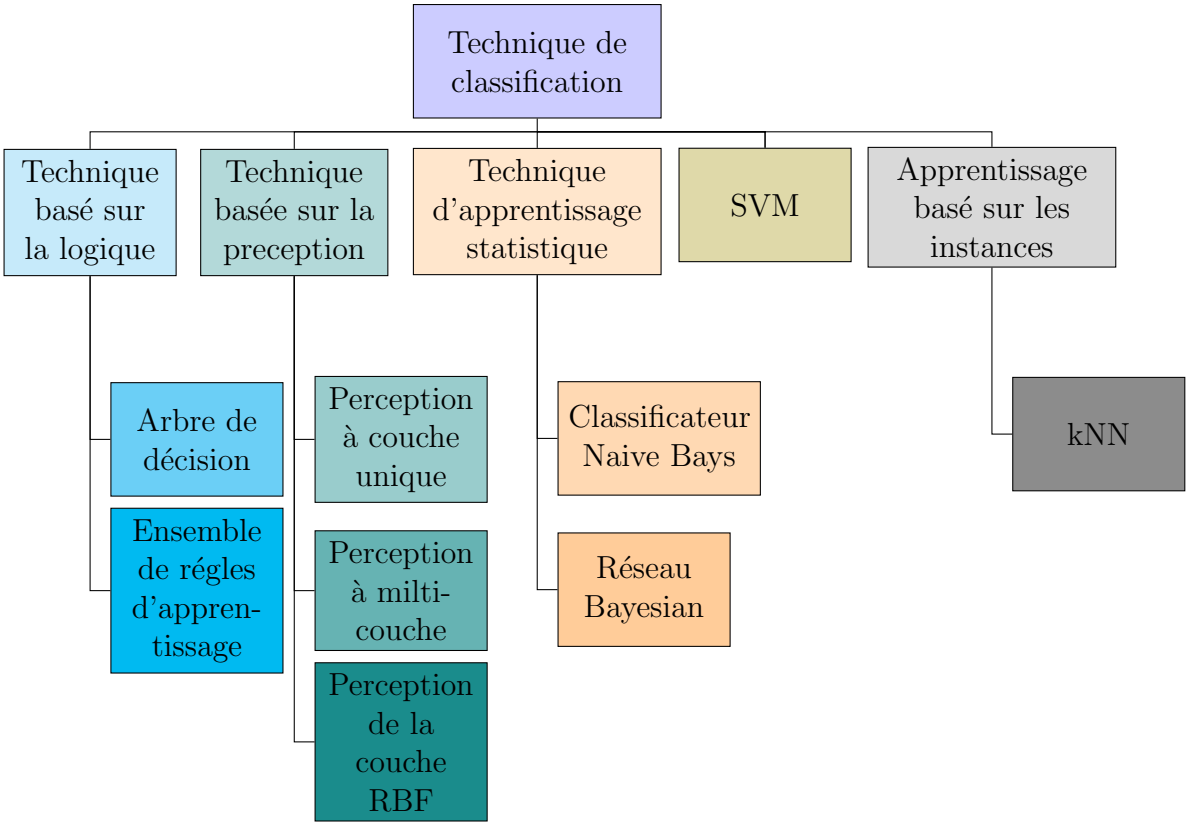


FIGURE 1.1 – Techniques de classification par apprentissage supervisé [4].

1.1 Méthodes classiques de l'apprentissage automatique

1.1.1 Méthode KNN (k plus proches voisins)

1.1.1.1 Introduction

La classification K-Nearest Neighbour (K-NN) est l'une des méthodes de classification les plus fondamentales et les plus simples et devrait être l'un des premiers choix pour une étude de classification lorsqu'il y a peu ou pas de connaissances préalables sur la distribution des données. Elle a été développée à partir de la nécessité d'effectuer une analyse discriminante lorsque des estimations paramétriques fiables des densités de probabilité sont inconnues ou difficiles à déterminer. Dans un rapport de l'US Air Force School of Aviation Medicine en 1951, Fix et Hodges ont introduit une méthode non paramétrique de classification des modèles

qui est depuis devenue la règle du k plus proche voisin [5]. Plus tard, en 1967, certaines des propriétés formelles de la règle du k-plus proche voisin ont été élaborées ; par exemple, il a été montré que pour $k = 1$ et $n \longrightarrow \infty$ l'erreur de classification du k-plus proche voisin est délimitée au-dessus par deux fois le taux d'erreur de Bayes [6]. Une fois que ces propriétés formelles de la classification par k-plus proches voisins ont été établies, une longue série de recherches s'en est suivie, y compris de nouvelles approches de rejet [7], des raffinements par rapport au taux d'erreur de Bayes [8], des approches pondérées par la distance [9], des méthodes d'informatique douce [10] et des méthodes floues [11].

L'algorithme kNN est utilisée dans de nomreux domaines :

- La reconnaissance de formes.
- La recherche de nouveaux biomarqueurs pour le diagnostic.
- Algorithmes de compression.
- Analyse d'image satellite.
- Marketing ciblé.

1.1.1.2 Principes de fonctionnement

Le KNN est l'une des techniques de classification couramment utilisées pour classer les données d'entrée dans des classes prédéfinies (k). Le mécanisme simple de l'algorithme KNN consiste à calculer la fonction de distance euclidienne entre les classes prédéfinies et chaque échantillon variable. Ensuite, l'algorithme KNN choisit le minimum de voisins les plus proches en fonction de chaque catégorie. Les échantillons sont affectés à leur catégorie en fonction des k voisins les plus proches. Il existe de nombreuses versions de la fonction de distance entre les échantillons. Dans ce document, la plus couramment utilisée est la distance euclidienne, exprimée par l'équation 1.1 [12].

$$d = \sqrt{\sum_{k=1}^{n} (X_1 k - X_2 k)^2} \tag{1.1}$$

Où k est le nombre de valeurs dans chaque vecteur échantillon,et X_1, X_2 sont les échantillons d'entrée.

Cette méthode n'a pas réellement de phase d'apprentissage, c'est-à-dire qu'il n'y a pas de construction de modèle. Tout repose sur :

- L'ensemble d'apprentissage stocké en mémoire.
- Une mesure de distance, c'est-à-dire la fonction noyau. Parmi les distances les plus souvent utilisées, on peut citer la distance euclidienne, cependant, en fonction du problème, on peut également utiliser les distances de Hamming, de Mahalanobis, etc.
- Une méthode de choix de la classe, en général, la méthode consiste à choisir la classe majoritaire parmi les k observations d'apprentissage les plus proches.

Notons que la capacité de généralisation de cette méthode dépend du paramètre k. Le réglage de k permet de lisser la modélisation. En effet, un k élevé permet d'englober plus de voisins et ainsi d'être moins sensible aux erreurs d'apprentissage, mais dans le même temps les k plus proches voisins doivent rester très proches de y pour que les approximations soient fiables. Par suite, il est nécessaire de réaliser un compromis, en choisissant une valeur de k plus petite que le nombre de d'observations

m. Cette méthode a l'avantage de pouvoir s'appliquer à des cas de discrimination faisant intervenir un nombre élevé de classes.

Les défauts de ces algorithmes simples sont d'une part le besoin de garder en mémoire les données d'apprentissage et d'autre part la nécessité de calculer les distances à tous les points d'apprentissage. Ils sont donc fort consommateur en espace mémoire et en temps de calcul. En plus, si les données d'apprentissage ne couvrent pas suffisamment les classes à étudier, les performances du classificateur se dégradent rapidement.

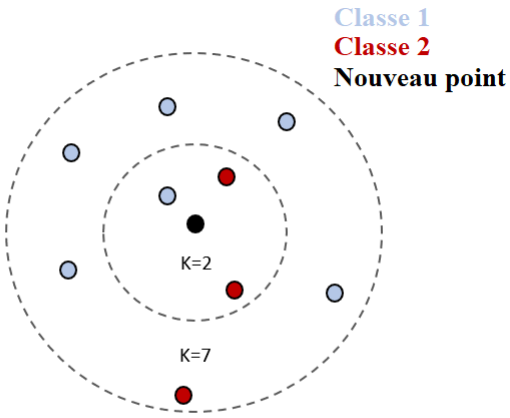


FIGURE 1.2 – Exemple de kNN avec nuage de point

Il existe 6 différents classificateurs KNN disponibles dans MATLAB qui peuvent être utilisés pour classifier nos données, qui sont :

1. **Fine KNN** : Un classificateur de plus proches voisins qui fait des distinctions finement détaillées entre les classes avec le nombre de voisins fixé à 1.
2. **Medium KNN** : Un classificateur de voisins les plus proches qui fait moins de distinctions qu'un KNN fin avec le nombre de voisins fixé à 10.
3. **Coarse KNN** : Un classificateur de plus proches voisins qui fait une distinction grossière entre les classes, avec un nombre de voisins fixé à 100.
4. **Cosine KNN** : Un classificateur des plus proches voisins qui utilise la distance en cosinus distance.
5. **Cubic KNN** : Un classificateur de plus proches voisins qui utilise la distance cubique.
6. **Weighted KNN** : Un classificateur de plus proches voisins qui utilise la pondération de la distance.

1.1.1.3 Les variantes

Il existe de nombreuses variantes de l'algorithme KNN proposées dans des études antérieures tel que :

Adaptive KNN (A-KNN) L'algorithme KNN adaptatif est une variante qui se concentre sur la sélection de la valeur k optimale pour un point de données de test. Il fonctionne en mettant en œuvre un algorithme distinct pour déterminer la valeur k optimale pour chaque point de données de l'ensemble de données d'apprentissage. L'algorithme principal trouve alors le plus proche voisin de l'ensemble de données d'apprentissage et hérite de sa valeur k pour un point de données de test donné. Cette variante de KNN fonctionne comme l'algorithme KNN classique pour prédire la sortie en utilisant cette valeur k héritée [13].

Locally adaptive KNN with Discrimination class (LA-KNN) Cette variante tient compte des informations provenant des classes de discrimination pour déterminer la valeur k optimale. Le concept de classe de discrimination prend en compte la quantité et la distribution des voisins de la classe majoritaire et des voisins de la seconde classe majoritaire dans le voisinage k d'un point de données de test donné. L'algorithme utilise plusieurs étapes pour définir les classes de discrimination. Après avoir sélectionné une de ces classes, il procède à la formation d'un tableau de classement avec différentes valeurs de k, les distances des centroïdes et leur rapport. A partir de ce tableau, il suit un processus de classement pour obtenir la valeur k optimale [13].

Fuzzy KNN (F-KNN) L'algorithme KNN flou est basé sur le principe de l'affectation des membres. Comme l'algorithme KNN classique, la variante procède à la recherche des k plus proches voisins d'un ensemble de données de test à partir de l'ensemble de données d'apprentissage. Elle procède ensuite à l'attribution de valeurs d'appartenance à chaque classe trouvée dans la liste des k plus proches voisins. Les valeurs d'appartenance sont calculées à l'aide d'un algorithme mathématique flou qui se concentre sur le poids de chaque classe. La classe ayant la valeur d'appartenance la plus élevée est ensuite sélectionnée pour le résultat de la classification [14].

K-means clustering-based KNN (KM-KNN) La variante KNN basée sur le clustering implique la combinaison de deux algorithmes populaires : k-means et 1NN. Cette variante utilise l'algorithme k-means pour regrouper l'ensemble de données d'apprentissage selon une variable prédéfinie (nombre de clusters). Il calcule ensuite les centroïdes de chaque cluster, créant ainsi un nouvel ensemble de données d'apprentissage qui contient les centroïdes de tous les clusters. L'algorithme 1NN est exécuté sur ce nouvel ensemble de données d'apprentissage, où le voisin le plus proche est pris pour la classification [15].

Weight adjusted KNN (W-KNN) Cette version de l'algorithme KNN se concentre sur l'application de la pondération des attributs. Cet algorithme attribue d'abord un poids à chacun des points de données de formation en utilisant une fonction connue sous le nom de fonction noyau. Cette affectation de poids vise à donner plus de poids aux points les plus proches et moins de poids aux points éloignés. Lorsque la distance augmente, toute fonction qui diminue la valeur peut être utilisée comme fonction noyau. La fréquence de tous les plus proches voisins est ensuite utilisée pour prédire la classe de sortie d'un point de données de test donné. Cette variante de

KNN prend en compte l'importance de la classification des différents attributs dans la définition de la fonction noyau pour un ensemble de données multi-attributs [13].

Hassanat distance KNN (H-KNN) L'algorithme KNN de Hassanat est une variante dont le point central est la formule de mesure de la distance. Cette variante suit la conception simple de l'algorithme KNN, mais elle propose une manière avancée de déterminer la distance entre deux points de données. La nouvelle formule de distance s'appelle la distance de Hassanat et s'articule autour de l'utilisation de points vectoriels maximums et minimums, similaires aux attributions de poids dans d'autres variantes. La métrique de distance de Hassanat de cette variante calcule les plus proches voisins d'une requête de test et applique la règle du vote majoritaire, comme l'algorithme classique KNN [16].

1.1.2 Méthode SVM (séparateurs à vaste marge)

1.1.2.1 Introduction

Les machines à vecteurs de support, ou support vector machine (SVM), sont des modèles de machine learning supervisés centrés sur la résolution de problèmes de discrimination et de régression mathématiques. Elles ont été conceptualisées dans les années 1990 à partir d'une théorie d'apprentissage statistique développée par les informaticiens russes Vladimir Vapnik et Alexey Chervonenkis : la théorie de Vapnik-Chervonenkis. Ce modèle a été rapidement adopté en raison de sa capacité à travailler avec des données de grandes dimensions, ses garanties théoriques et ses bons résultats réalisés en pratique. Requérant un faible nombre de paramètres, les SVM sont appréciées pour leur simplicité d'usage [17].

Les SVM ont été développés dans l'ordre inverse du développement des réseaux de neurones (NN). Les SVM ont évolué d'une théorie solide à la mise en œuvre et aux expériences, tandis que les NN ont suivi un chemin plus heuristique, des applications et des expériences approfondies à la théorie. Il est intéressant de noter que le contexte théorique très solide des SVM n'a pas permis de les apprécier largement au début. La publication des premiers articles par Vapnik, Chervonenkis et leurs collègues en 1964/65 est passée largement inaperçue jusqu'en 1992[18]. Cela était dû à une croyance répandue dans la communauté statistique et/ou d'apprentissage automatique que, malgré leur attrait théorique, les SVM ne sont ni appropriés ni pertinents pour des applications pratiques. Les SVM n'ont été pris au sérieux que lorsque d'excellents résultats ont été obtenus sur des benchmarks d'apprentissage pratique dans les domaines de la reconnaissance des chiffres, de la vision par ordinateur et de la catégorisation de textes. Aujourd'hui, les SVMs montrent de meilleurs résultats que (ou des résultats comparables à) les NNs et d'autres modèles statistiques, sur les problèmes de référence les plus populaires.

Le problème d'apprentissage des SVM est le suivant : il existe une dépendance (fonction) inconnue et non linéaire $y = f(x)$ entre un vecteur d'entrée x de haute dimension et une sortie scalaire y (ou la sortie vectorielle y dans le cas des SVM multiclasses). Il n'existe aucune information sur les fonctions de probabilité conjointes sous-jacentes. Il faut donc effectuer un apprentissage sans distribution. La seule information disponible est un ensemble de données d'apprentissage $D = (x_i, y_i) \in X \times Y, i = 1, l$, où l représente le nombre de paires de données d'apprentissage et est donc égal

à la taille de l'ensemble de données d'apprentissage D. Souvent, y_i est noté d_i, où d représente une valeur souhaitée (cible). Les SVM font donc partie des techniques d'apprentissage supervisé [18].

1.1.2.2 Principe de fonctionnement

Les SVM peuvent être utilisés pour résoudre des problèmes de discrimination, c'est-à-dire décider à quelle classe appartient un échantillon, ou de régression, c'est-à-dire prédire la valeur numérique d'une variable. La résolution de ces deux problèmes passe par la construction d'une fonction h qui à un vecteur d'entrée x fait correspondre une sortie y : $y = h(x)$

On se limite pour l'instant à un problème de discrimination à deux classes (discrimination binaire), c'est-à-dire $y \in \{-1, 1\}$, le vecteur d'entrée x étant dans un espace X muni d'un produit scalaire. On peut prendre par exemple $X = \mathbb{R}^N$ [19].

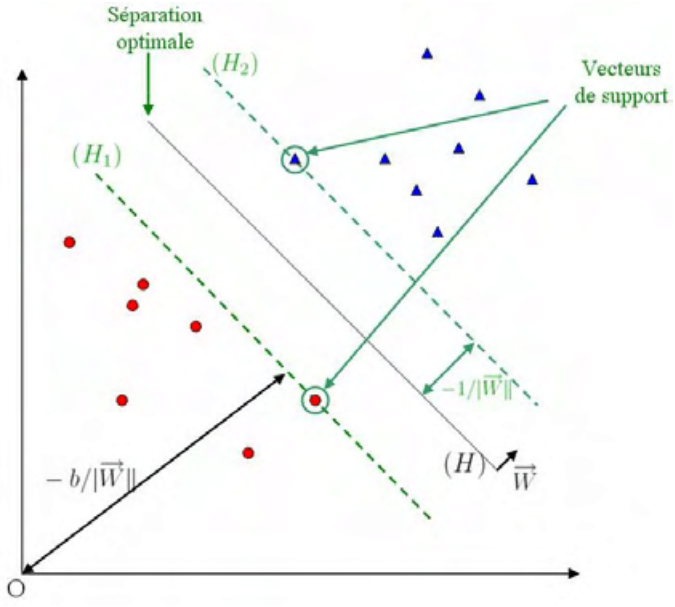


FIGURE 1.3 – Exemple graphique SVM [20]

Il existe 6 différents classificateurs SVM disponibles sous MATLAB qui peuvent être utilisés pour classifier nos données, qui sont :

1. **Linear SVM** : Effectue une simple séparation linéaire entre les classes.
2. **Quadratic SVM** : la fonction noyau du classificateur étant une fonction carrée
   $k(x_i, x_j) = (x_i^T x_j)^2$.
3. **Cubic SVM** : la fonction noyau du classificateur étant une fonction cubique $k(x_i, x_j) = (x_i^T x_j)^3$.
4. **Fine Gaussian SVM** : Fait des distinctions très détaillées entre les classes, avec une échelle de noyau fixée à $\frac{\sqrt{P}}{4}$ où P est le nombre de prédicteurs.
5. **Medium Gaussian SVM** : Distinctions moyennes, avec une échelle de noyau fixée à $\sqrt{P}$.
6. **Coarse Gaussian SVM** : Fait des distinctions grossières entre les classes, avec une échelle de noyau fixée à $\sqrt{P} \times 4$,

1.1.2.3 Les variantes des SVM

SVMs multiclasse Dans son type le plus simple, le SVM ne supporte pas nativement la classification multi-classes. Il supporte la classification binaire et la séparation des points de données en deux classes. Pour la classification multi-classes, le même principe est utilisé après avoir décomposé le problème de multi-classification en plusieurs problèmes de classification binaire.

L'idée est de faire correspondre les points de données à un espace de haute dimension pour obtenir une séparation linéaire mutuelle entre chaque deux classes. C'est ce qu'on appelle l'approche One-to-One, qui décompose le problème multi-classes en plusieurs problèmes de classification binaire. Un classificateur binaire pour chaque paire de classes [21].

Une contre le reste (OVR : One vs Rest) On calcule pour chaque classe un hyperplan la séparant des autres. Lors de la phase de sélection, on prend la classe maximisant la fonction de décision (Figure 1.4).

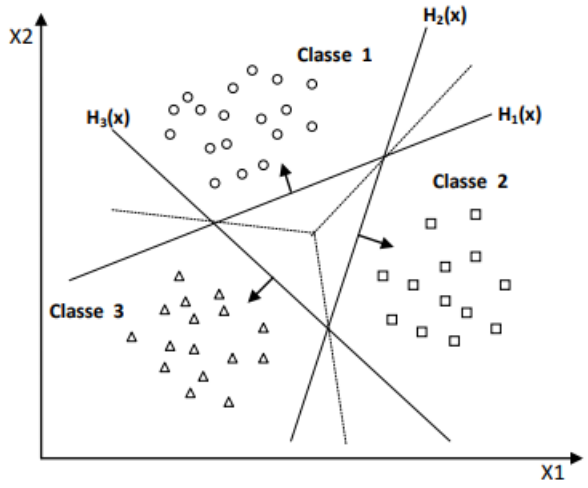


FIGURE 1.4 – Approche une contre le reste avec des zones d'indécision [22]

Une contre une (1vs1) On calcule pour chaque classe les hyperplans la séparant de chaque autre classe. Dans la phase de sélection, on prend la classe qui maximise le nombre d'appartenances par rapport aux autres classes (Figure 1.5).

SVM basées arbres de décision Dans cette méthode, on apprend pour K classes, (K - 1) hyperplans. Chaque hyperplan sépare une ou plusieurs classes du reste, selon un découpage choisi. On peut choisir, par exemple un découpage semblable à la méthode 1vsR où l'hyperplan H_i sépare la classe i des classes $i+1, i+2, ..., K$. (Figure 1.6.)

Dans la phase de classification, pour classer un nouvel exemple x, on teste les hyperplans dans l'ordre croissant et on s'arrête sur le premier hyperplan qui retourne une valeur de décision positive. L'exemple x appartient alors à la classe positive de cet hyperplan.

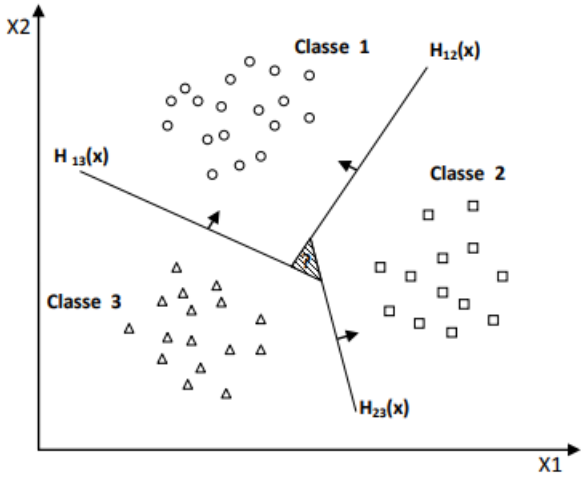


FIGURE 1.5 – Approche une-contre-une [22]

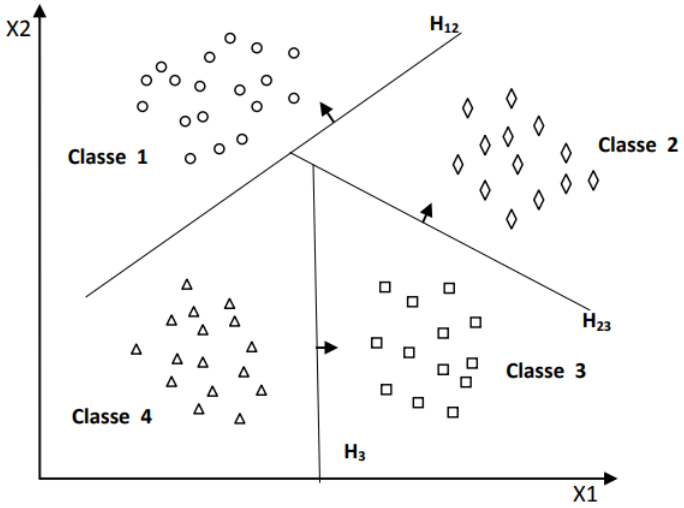


FIGURE 1.6 – SVM multiclasse par arbre de décision [22]

Les méthodes basées sur les arbres de décisions sont généralement plus rapides que la méthode 1vsR. Cela est dû au fait que la méthode 1vsR utilise, pour entrainer chaque hyperplan, tous les exemples, tandis que dans les méthodes basées sur les arbres de décision, le nombre d'exemples d'entraînement diminue en descendant dans l'arbre [23].

SVM monoclasse (Novelty detection) Dans les SVM et multiclasse précédentes, nous avons toujours des exemples positifs et d'autres négatifs c-à-d des exemples et des contre-exemples. De telles informations ne sont pas disponibles dans tous les cas d'application. Parfois, il est très coûteux, voire impossible, de trouver des contre-exemples qui représentent réellement la classe négative. Prenons l'exemple de reconnaissance d'une catégorie particulière de pièces par un robot dans une usine, il est facile d'avoir des exemples suffisants de cette pièce, mais il est difficile d'avoir des exemples de toutes les pièces différentes. Il est souhaitable, dans de tels cas, d'avoir

un modèle de décision permettant de reconnaître autant d'exemples possibles de cette catégorie et de rejeter tous les autres. Ce problème est souvent appelé "Novelty détection" ou détection des nouveautés, puisque le modèle de décision connaît un ensemble d'exemples et détecte tous ce qui est nouveau (étranger ou outlier).

Pour la classification SVM monoclasse, il est supposé que seules les données de la classe cible sont disponibles. L'objectif est de trouver une frontière qui sépare les exemples de la classe cible du reste de l'espace, autrement dit, une frontière autour de la classe cible qui accepte autant d'exemples cibles que possible. Cette frontière est représentée par une fonction de décision positive à l'intérieur de la classe et négative en dehors. La figure 1.7 représente, en deux dimensions, un cas de séparation d'une classe de toute autre classe.

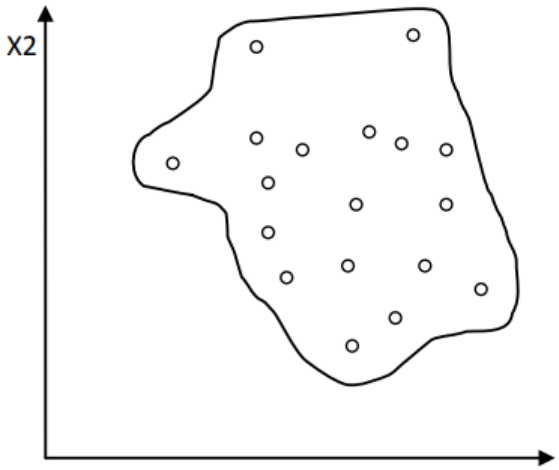


FIGURE 1.7 – Séparation des exemples d'une classe du reste de l'espace [24]

1.1.3 Méthode decision tree learning (Arbre de décision)

1.1.3.1 Introduction

Un arbre de décision est un modèle très simple. Étant donnée plusieurs caractéristiques, la décision commence par un de ces caractéristiques, si ce n'ai pas suffisant, on utilise une autre, ainsi de suite. Il est largement connu et utilisé dans de nombreuses entreprises pour faciliter le processus de prise de décision et l'analyse des risques. Il a été largement utilisé dans les années 1960-1980 pour la construction de systèmes experts. Les règles sont introduites manuellement, pour cette raison ce modèle a perdu sa popularité après les années 80. L'apparition des méthodes mathématiques pour construire les arbres de décision fait revenir ce modèle à la bataille des algorithmes de l'apprentissage automatique.

1.1.3.2 Principes de fonctionnement

Voici un exemple d'un arbre de décision le plus basique :
Vais-je jouer au tennis ?

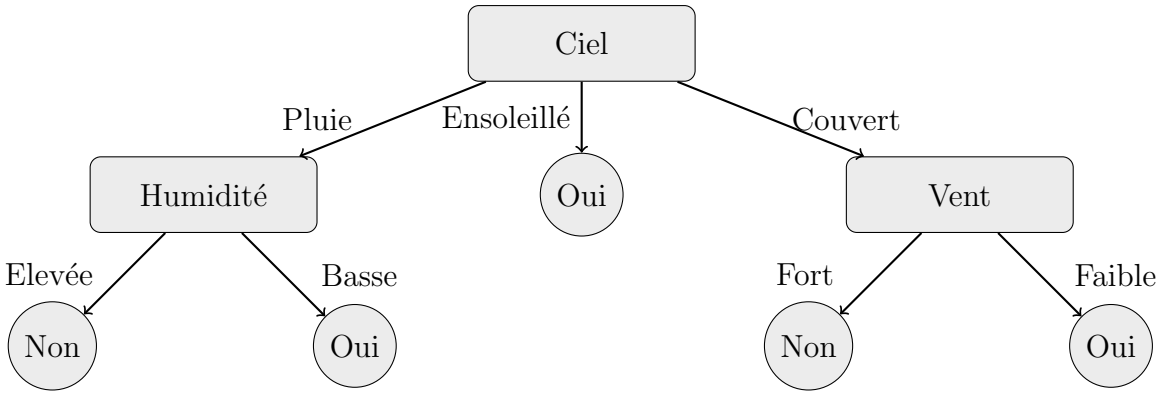


FIGURE 1.8 – Exemple d'un arbre de décision

Les points forts de l'arbre de décision L'une des nombreuses qualités des arbres de décision est qu'ils nécessitent très peu de préparation de données. En particulier, ils ne nécessitent pas du tout de mise à l'échelle ou de centrage des caractéristiques.

Comme vous pouvez le voir, les arbres de décision sont assez intuitifs et leurs décisions sont faciles à interpréter, nous appelons ce genre de modèle : des boîtes blanches. En revanche, d'autres algorithmes comme le boosting ou les réseaux de neurones sont généralement considérés comme des modèles de boîte noire.

1.1.3.3 Les variantes des arbres de décision

- **ID3 (Iterative Dichotomiser 3)** : Dans l'apprentissage par arbre de décision, ID3 (Iterative Dichotomiser 3) est un algorithme inventé par Ross Quinlan [25] utilisé pour générer un arbre de décision à partir de l'ensemble des données. ID3 est généralement utilisé dans les domaines de l'apprentissage automatique et du traitement du langage naturel. La technique de l'arbre de décision consiste à construire un arbre pour modéliser le processus de classification. Une fois l'arbre construit, il est appliqué à chaque uplet [1] de la base de données et donne lieu à une classification pour ce uplet. [26].
  L'algorithme ID3 est un algorithme de classification basé sur l'entropie d'information. Son idée de base est que tous les exemples sont affectés à différentes catégories en fonction des différentes valeurs de l'ensemble d'attributs de condition ; son cœur est de déterminer le meilleur attribut de classification à partir des ensembles d'attributs de condition. L'algorithme choisit le gain d'information comme critère de sélection des attributs ; en général, l'attribut qui présente le gain d'information le plus élevé est sélectionné comme attribut de division du nœud actuel, afin que l'entropie d'information dont les sous-ensembles divisés ont besoin soit la plus petite possible [27]. Selon les différentes valeurs de l'attribut, des branches peuvent être établies, et le processus ci-dessus est appelé récursivement sur chaque branche pour créer d'autres nœuds et branches jusqu'à ce que tous les échantillons d'une branche appartiennent à la même catégorie. Pour sélectionner les attributs de division, les concepts d'entropie et de gain d'information sont utilisés.

1. Un n-uplet (appelé tuple en anglais) est une séquence non modifiable de données ordonnées

- **C4.5** : une extension de ID3 par Ross Quinlan. Il peut être appliqué sur tous les types de caractéristiques. Il est utilisé pour le classement.
- **C5.0** : une extension commerciale de C4.5, toujours par Ross Quinlan.
- **CART (Classification and Regression Trees)** : comme C4.5 mais utilise d'autres métriques. Aussi, l'algorithme supporte la régression.

1.2 Méthodes utilisées dans l'apprentissage profond

> On dit d'un programme informatique qu'il apprend de l'expérience E en ce qui concerne une certaine classe de tâches T et une mesure de performance P, si sa performance aux tâches dans T, telle que mesurée par P, s'améliore avec l'expérience E.
>
> *Tom Mitchel*
> *Machine Learning [28]*

1.2.1 Introduction à LSTM

LSTM (pour Long Short-Term Memory) est un réseau de neurones artificiels utilisé dans les domaines de l' intelligence artificielle et de l'apprentissage profond. Contrairement aux réseaux de neurones à anticipation standard, LSTM a des connexions de rétroaction. Un tel réseau neuronal récurrent peut traiter non seulement des points de données uniques (tels que des images), mais également des séquences entières de données (telles que la parole ou la vidéo). Par exemple, LSTM est applicable à des tâches telles que la reconnaissance d'écriture manuscrite non segmentée et connectée, la reconnaissance vocale, la traduction automatique, la commande de robots, les jeux vidéo, et les soins de santé. LSTM est devenu le réseau neuronal le plus cité du 20ème siècle [29].

Une unité LSTM commune est composée d'une cellule, d'une porte d'entrée, d'une porte de sortie et d'une porte d'oubli. La cellule se souvient des valeurs sur des intervalles de temps arbitraires et les trois portes régulent le flux d'informations entrant et sortant de la cellule.

Les réseaux LSTM sont bien adaptés à la classification , au traitement et à la réalisation de prédictions basées sur des données de séries chronologiques, car il peut y avoir des décalages de durée inconnue entre des événements importants dans une série chronologique. Les LSTM ont été développés pour traiter le problème du gradient de fuite qui peut être rencontré lors de la formation des RNN traditionnels. L'insensibilité relative à la longueur de l'écart est un avantage de LSTM par rapport aux RNN, aux modèles de Markov cachés et à d'autres méthodes d'apprentissage de séquences dans de nombreuses applications.

La figure 1.9 présente le schéma d'un seul neurone LSTM

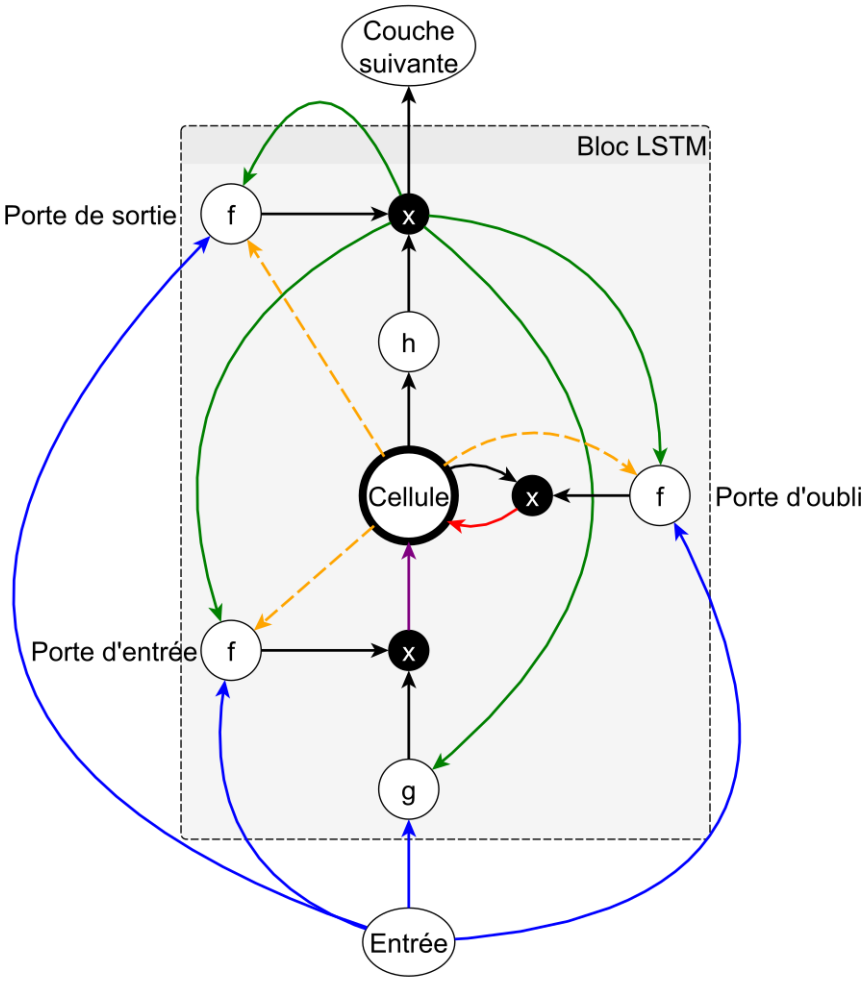


FIGURE 1.9 – Schéma d'un neurone LSTM comprenant une mémoire interne (cell) contrôlée par les trois portes d'entrée (input), d'oubli (forget) et de sortie (output) [30].

Comme on peut le constater sur la figure 1.9, la cellule mémoire peut être pilotée par trois portes de contrôle qu'on peut considérer comme des vannes :

- **la porte d'entrée** décide si l'entrée doit **modifier le contenu** de la cellule ;
- **la porte d'oubli** décide s'il faut **remettre à 0** le contenu de la cellule ;
- **la porte de sortie** décide si le contenu de la cellule doit **influer sur la sortie** du neurone.

Le mécanisme des trois portes est strictement similaire. L'ouverture/la fermeture de la vanne est modélisée par une fonction f qui est généralement une sigmoïde. Cette sigmoïde est appliquée à la somme pondérée des entrées (en bleu), des sorties (en vert) et de la cellule (en orange), par des poids w_{ab} spécifiques à chaque connexion entre les signaux a et b [31] [32] [33].

Soient x_i et z_h les entrées et les sorties de la cellule, sc la valeur de la cellule et l, ω, c les indices décrivant respectivement les signaux issus de la porte d'entrée, d'oubli et de la cellule. w_{il} sont donc les poids reliant les entrées à la cellule d'entrée, $w_{h\phi}h$ les poids reliant les sorties à la porte d'oubli, etc.

Les équations régissant les trois portes de contrôle sont donc les suivantes ; elles

sont l'application de la somme pondérée suivie de l'application d'une activation$f$, typiquement la sigmoïde :

Porte d'entrée :

$$a_l^t = \sum_{i=1}^{I} w_{il} x_i^t + \sum_{h=1}^{H} w_{hl} z_h^{t-1} + \sum_{c=1}^{C} w_{cl} x_c^{t-1} \tag{1.2}$$

$$b_l^t = f(a_l^t)$$

porte d'oubli :

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} z_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1} \tag{1.3}$$

$$b_\phi^t = f(a_\phi^t) \tag{1.4}$$

porte de sortie :

$$a_w^t = \sum_{i=1}^{I} w_{iw} x_i^t + \sum_{h=1}^{H} w_{hw} z_h^{t-1} + \sum_{c=1}^{C} w_{cw} s_c^t \tag{1.5}$$

$$b_w^t = f(a_w^t) \tag{1.6}$$

1.2.2 Généralités sur la méthode de LSTM

L'utilisation de l'expérience passée pour améliorer les performances futures est une pierre angulaire de l'apprentissage profond, et même de l'apprentissage automatique en général. Une définition de l'apprentissage automatique expose explicitement l'importance de l'amélioration par l'expérience :

Dans les réseaux neuronaux, l'amélioration des performances avec l'expérience est codée comme une mémoire à très long terme dans les paramètres du modèle. Après avoir appris à partir d'un ensemble d'exemples annotés, un réseau neuronal a plus de chances de prendre la bonne décision lorsqu'on lui montre d'autres exemples similaires mais non vus auparavant. C'est l'essence même de l'apprentissage profond supervisé sur des données présentant une correspondance claire entre elles, par exemple un ensemble d'images correspondant à une classe par image (chat, chien, hotdog, etc.).

Il existe de nombreux cas où les données forment naturellement des séquences et, dans ce cas, l'ordre et le contenu sont tout aussi importants. Parmi les autres exemples de données séquentielles, on peut citer la vidéo, la musique, les séquences d'ADN et bien d'autres encore. Lors de l'apprentissage à partir de données séquentielles, la mémoire à court terme devient utile pour traiter une série de données connexes dans un contexte ordonné. Pour cela, les chercheurs en apprentissage automatique se sont longtemps tournés vers le réseau neuronal récurrent ou RNN.

Un RNN standard est essentiellement un réseau neuronal de type feed-forward déroulé dans le temps. Cet arrangement peut être atteint simplement en introduisant

des connexions pondérées entre un ou plusieurs états cachés du réseau et les mêmes états cachés du dernier point temporel, fournissant ainsi une certaine mémoire à court terme. Le problème est que cette mémoire à court terme est fondamentalement limitée de la même manière que la formation de réseaux très profonds est difficile, ce qui rend la mémoire des RNN classiques très courte.

L'apprentissage par rétro-propagation à travers de nombreuses couches cachées est sujet au problème de la disparition du gradient. Sans entrer dans les détails, l'opération consiste généralement à multiplier de manière répétée un signal d'erreur par une série de valeurs (les gradients de la fonction d'activation) inférieures à 1,0, atténuant le signal à chaque couche. La rétro-propagation dans le temps présente le même problème, limitant fondamentalement la capacité d'apprendre à partir de dépendances à relativement long terme.

1.2.2.1 Le développement de nouvelles fonctions d'activation dans les réseaux de neurones à apprentissage profond

En ce qui concerne l'apprentissage profond avec les réseaux de neurones à action directe, le défi de la disparition des gradients a conduit à la popularité de nouvelles fonctions d'activation (comme les ReLU [2]) et de nouvelles architectures (comme ResNet et DenseNet). Pour les RNN, une des premières solutions a été de ne pas former les couches récurrentes, mais de les initialiser de manière à ce qu'elles effectuent une transformation chaotique non linéaire des données d'entrée en représentations de dimension supérieure.

La rétroaction récurrente et l'initialisation des paramètres sont choisies de manière à ce que le système soit très peu instable, et une simple couche linéaire est ajoutée à la sortie. L'apprentissage est limité à cette dernière couche linéaire, et de cette manière, il est possible d'obtenir des performances raisonnablement correctes sur de nombreuses tâches tout en évitant de traiter le problème du gradient de fuite en l'ignorant complètement. Ce sous-domaine de l'informatique s'appelle le calcul par réservoir [34], et il fonctionne même (dans une certaine mesure) en utilisant un seau d'eau comme réservoir dynamique pour effectuer des calculs complexes [35].

Insuffisance des RNN de type réservoir Avec des réservoirs dynamiques quasi-stables, l'effet d'une entrée donnée peut persister pendant une très longue période. Cependant, les RNN de type réservoir sont encore insuffisants pour plusieurs raisons :

1) le réservoir dynamique doit être très proche de l'instabilité pour que les dépendances à long terme persistent, de sorte que des stimuli continus pourraient faire exploser la sortie au fil du temps
2) il n'y a toujours pas d'apprentissage direct sur les parties inférieures ou supérieures du réseau. La thèse de diplôme de Sepp Hochreiter en 1991 [36] décrivait le problème fondamental de la disparition des gradients dans les réseaux neuronaux profonds, ouvrant la voie à l'invention des réseaux neuronaux récurrents à mémoire à long terme (LSTM) par Sepp Hochreiter et Jürgen Schmidhuber en [36].

2. Rectified Linear Unit

Les LSTM peuvent apprendre des dépendances à long terme que les RNN "normaux" ne peuvent fondamentalement pas apprendre. L'idée maîtresse de cette capacité est un module persistant appelé état cellulaire qui constitue un fil conducteur à travers le temps, perturbé uniquement par quelques opérations linéaires à chaque pas de temps. Étant donné que la connexion de l'état cellulaire aux états cellulaires précédents n'est interrompue que par les opérations linéaires de multiplication et d'addition, les LSTM et ses variantes peuvent se souvenir d'information à court terme (c'est-à-dire d'activités appartenant au même "épisode") pendant très longtemps.

Un certain nombre de modifications de l'architecture LSTM originale ont été suggérées au fil des ans, mais il peut être surprenant que la variante classique continue d'obtenir des résultats de pointe sur une variété de tâches de pointe plus de 20 ans plus tard. Ceci étant dit, quelles sont les variantes de LSTM et à quoi servent-elles ?

1.3 Discussion des différents algorithmes

Les différents algorithmes de classification seront entraînés et étudié avec une base de données dédié le tout Sous Matlab classification learner, un comparatif des différents résultats des classificateurs sera établi, puis un comparatif sera fait entre les différents classificateurs et le LSTM pour conclure quel algorithme est le plus adéquat pour la reconnaissance d'activité humaine.

Chapitre 2

Aperçu sur la technique de détection de mouvement par smartphone

Les smartphones sont de plus en plus omniprésents et puissants. La puissance de calcul et les capacités de stockage des smartphones se sont considérablement améliorées grâce aux puces multicœurs et aux mémoires plus volumineuses. Les smartphones sont naturellement équipés de différents types de capteurs comme le thermomètre, le magnétomètre, le baromètre, le microphone, la caméra, le capteur de luminosité ambiante, le GPS, le capteur de proximité, l'accéléromètre, etc.

Jusqu'à présent, parmi ces capteurs pour smartphones, l'accéléromètre a reçu le plus d'attention dans la recherche sur la reconnaissance d'activité. Cependant, ces dernières années, d'autres capteurs, comme le gyroscope et le magnétomètre, ont été combinés avec un accéléromètre dans le but d'améliorer les performances de reconnaissance d'activité :

- Associé à un gyroscope, ces capteurs permettent de stabiliser les calculs d'orientation et de déterminer l'orientation par rapport à la Terre.
- La combinaison avec un magnétomètre donne une boussole électronique.
- La combinaison des trois types de capteurs (accéléromètre + gyroscope + magnétomètre) donne un capteur tout en un de type 9-DoF.

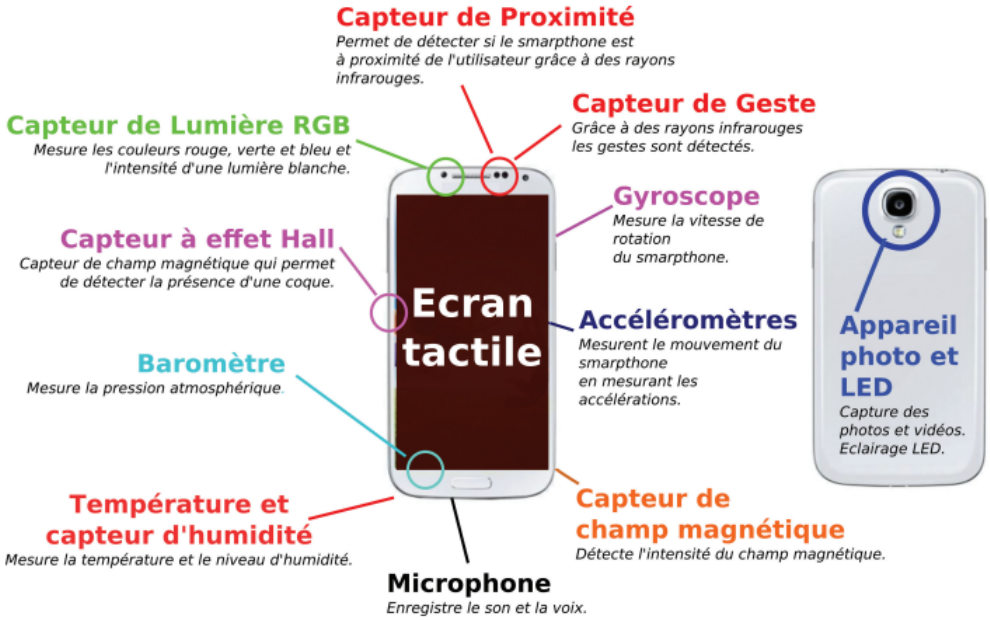


FIGURE 2.1 – Les différents capteurs d'un smartphone [37].

2.1 Les avantages par rapport aux réseaux de capteurs sans fil

Un réseau de capteurs sans fil (Wireless Sensors Network - WSN) se compose d'un ensemble de capteurs répartis géographiquement travaillant ensemble pour suivre les conditions environnementales et physiques. Aujourd'hui, le WSN est identifié comme l'une des technologies les plus importantes et les plus largement utilisées. Les déploiements de WSN couvrent de nombreux objectifs tels que : la surveillance de l'environnement, les villes intelligentes, les bâtiments intelligents, les véhicules, le trafic routier, les soins de santé, etc.

Les principaux avantages d'un Smartphone par rapport au WSN sont [38] :

- Le faible coût de l'équipement : Profiter des capteurs déjà présents sur les smartphones permet de réduire considérablement les coûts qui se limitent principalement aux coûts des logiciels et des équipements centraux.
- Les smartphones ont une grande puissance de calcul : chaque smartphone est un ordinateur avec des capacités de calcul parallèle qui peuvent aider à traiter partiellement les données capturées. Cela réduit la quantité de données échangées et traitées au niveau du serveur central.
- La programmation pour les smartphones est pratique : les outils de développement sont gratuits et puissants. Pour les smartphones Android, il est possible de programmer en Java sur Eclipse ou Android Studio Integrated Development Environment (IDE) . Les applications pour iPhone peuvent être programmées avec Objective-C ou Swiff sur l'IDE XCode.
- Les smartphones sont largement connectés à Internet : les smartphones sont capables non seulement de se connecter les uns aux autres, comme dans les réseaux ad hoc, mais également de se connecter à Internet. En plus du faible coût de l'accès Internet et de la large couverture WiFi, les données collectées

par les smartphones pourraient être envoyées directement aux serveurs centraux. En conséquence, le modèle de transfert de données devient plus simple et l'architecture réseau d'un smartphone devient également plus simple qu'un WSN traditionnel.

- Le nombre de smartphones est de plus en plus important : des milliards de smartphones sont potentiellement exploitables dans le monde. Plusieurs recherches ont été menées sur la construction de systèmes de capteurs utilisant des smartphones. Les résultats de l'analyse des données à l'échelle du système apporteront de nombreux avantages dont les participants pourront tirer parti. De tels systèmes sont appelés crowdsensing systems.

2.2 Calcul de la Moyenne et de la Variance

La moyenne La moyenne un nombre exprimant la valeur centrale ou typique dans un ensemble de données, en particulier le mode, la médiane ou (le plus souvent) la moyenne, qui est calculé en divisant la somme des valeurs de l'ensemble par leur nombre. La formule de base pour la moyenne de n nombres $x_1, x_2, \ldots\ldots x_n$ est :

$$A = (X_1 + X_2 \ldots\ldots + X_n)/n \tag{2.1}$$

La variance La variance est la somme des carrés des différences entre tous les nombres et les moyennes. La formule mathématique de la variance est la suivante,

$$\sigma^2 = \frac{\Sigma_{i=1}^{N}(x_i - \mu)^2}{N} \tag{2.2}$$

où, μ est la moyenne, N est le nombre total d'éléments ou la fréquence de distribution.

2.3 La décomposition en composante principale

L'analyse en composantes principales (ACP ou PCA en anglais pour principal component analysis), ou, selon le domaine d'application, transformation de Karhunen–Loève (KLT) [39] ou transformation de Hotelling, est une méthode de la famille de l'analyse des données et plus généralement de la statistique multivariée, qui consiste à transformer des variables liées entre elles (dites « corrélées » en statistique) en nouvelles variables décorrélées les unes des autres. Ces nouvelles variables sont nommées « composantes principales » ou axes principaux. Elle permet au statisticien de résumer l'information en réduisant le nombre de variables.

Il s'agit d'une approche à la fois géométrique [40] (les variables étant représentées dans un nouvel espace, selon des directions d'inertie maximale) et statistique (la recherche portant sur des axes indépendants expliquant au mieux la variabilité — la variance — des données). Lorsqu'on veut compresser un ensemble de $\mathbb{N}$ variables aléatoires, les n premiers axes de l'analyse en composantes principales sont un meilleur choix, du point de vue de l'inertie ou de la variance.

L'outil mathématique est appliqué dans d'autres domaines que les statistiques et est parfois appelé décomposition orthogonale aux valeurs propres ou POD (anglais : proper orthogonal decomposition).

2.4 Signaux et filtrage

Les signaux des capteurs (accéléromètre et gyroscope) ont été prétraités en appliquant des filtres de bruit, puis échantillonnés dans des fenêtres coulissantes de largeur fixe de 2,56 secondes avec un chevauchement de 50% (128 lectures par fenêtre). Le signal d'accélération du capteur, qui comporte des composantes gravitationnelles et de mouvement du corps, a été séparé à l'aide d'un filtre passe-bas de Butterworth en accélération du corps et gravité. La force gravitationnelle étant supposée n'avoir que des composantes de basse fréquence, un filtre avec une fréquence de coupure de 0,3Hz a été utilisé. À partir de chaque fenêtre, un vecteur de caractéristiques a été obtenu en calculant des variables dans le domaine temporel et fréquentiel.

Les caractéristiques sélectionnées pour cette base de données proviennent des signaux bruts 3-axiaux de l'accéléromètre et du gyroscope tAcc-XYZ et tGyro-XYZ. Ces signaux temporels (préfixe 't' pour indiquer le temps) ont été capturés à une fréquence constante de 50Hz. Ils ont ensuite été filtrés à l'aide d'un filtre médian et d'un filtre Butterworth passe-bas de 3e ordre avec une fréquence de coupure de 20Hz pour éliminer le bruit. De même, le signal d'accélération a ensuite été séparé en signaux d'accélération du corps et de la gravité (tBodyAcc-XYZ et tGravityAcc-XYZ) à l'aide d'un autre filtre Butterworth passe-bas avec une fréquence de coupure de 0,3Hz.

Ensuite, l'accélération linéaire et la vitesse angulaire du corps ont été dérivées dans le temps pour obtenir des signaux de secousses (tBodyAccJerk-XYZ et tBodyGyroJerk-XYZ). L'amplitude de ces signaux tridimensionnels a également été calculée en utilisant la norme euclidienne (tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).

Enfin, une transformée de Fourier rapide (FFT) a été appliquée à certains de ces signaux, produisant fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag. (Notez le 'f' pour indiquer les signaux du domaine fréquentiel).

2.5 Application de la détection de mouvement

FIGURE 2.2 – Google Fit LOGO

La détection d'activité humaine et de mouvement par capteur a connu un engouement récent avec l'arrivé des smartphones pour différentes applications, notamment dans le domaine de la santé avec des applications dédiées au sport qui récupèrent des données physiques de la personne (sexe, âge, taille, poids) pour estimer la quantité de calories brûlées au cours d'un effort. Ces applications utilisent aussi les données d'accéléromètre et de gyroscope pour détecter quand l'utilisateur est entrain de courir ou faire son activité sportive. Des petites startups au grand groupe de la tech, beaucoup d'entreprise ont commencé à développer leurs propres solutions tel que Google avec Google fit qui est une application de tracking d'activité sportive qui collecte les données physiques de la personne pour estimer la quantité de calories brûlées.

FIGURE 2.3 – CROSCSCALL et NEOSAFE LOGO

Dans le domaine de la sécurité des travailleurs isolés, deux entreprises se sont aussi illustrées afin d'intervenir rapidement sur site en cas de malaise ou de chute, c'est les entreprises CROSSCALL et Neosafe qui se sont associées pour créer toute une gamme de produit nommé CORE spécialement pour les travailleurs isolés, grâce aux multiples capteurs qu'embarque un smartphone qui peuvent spécialement être utilisé pour ce type d'alerte. Ainsi, les terminaux embarquent un gyroscope capable de mesurer la perte de verticalité, un accéléromètre pour détecter une chute ou encore un GPS pour la géolocalisation.

Plusieurs base de données de projet universitaire ou bien open source sont disponible avec le protocole utilisé pour l'enregistrement cependant pour les applications officiel qui offre un réel service peu d'information sont disponible que ce soit pour la collecte (filtrage, fréquence d'échantillonage, technique d'enregistrement, taille de la base de données, nombre de volontaire) ou bien les algorithme utilisé ainsi que la méthode d'entraînement, dans ce travail on propose d'enrichir une base de données avec des données nouvelle enregistrées par nos soin en suivant le protocole détaillé plus bas.

Chapitre 3

Méthodologie

3.1 Étude des données pour les différentes activités

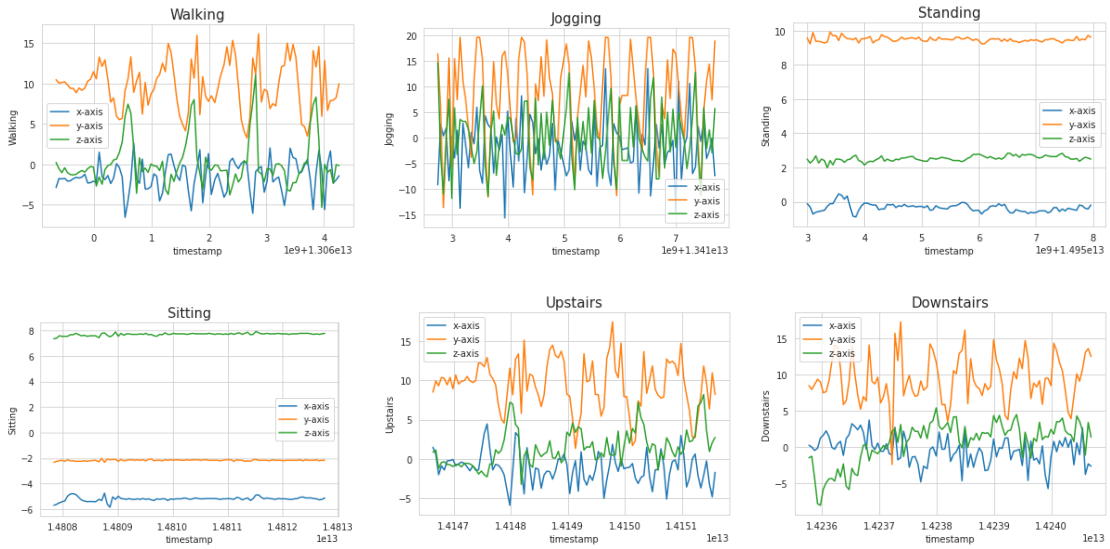


FIGURE 3.1 – Graphique des activitées avec comme variable des accélérations selon x, y et z

Comparé à la base de données précédente cette base de données possède plus d'activité mais ne possede pas l'activité laying. On suppose que la pertinence de cette dernière n'était pas très haute vu sa forte ressemblance avec l'activité sitting.

Pour la visualisation, nous avons considéré un sous-ensemble de 100 échantillons. Cela équivaut à 5 secondes d'activité (la fréquence de collecte des données étant de 20 Hz). Comme nous le remarquons, le signal montre un comportement périodique pour des activités telles que la marche, le jogging, la montée et la descente des escaliers, tandis qu'il présente un mouvement très faible pour les activités stationnaires telles que la position assise et debout. Ces signaux peuvent être modélisés comme des séries de données temporelles.

3.1.1 Chargement et Concaténation des données

Dans un premier temps on nous allons charger les différents données du DataSet dans notre projet et les concaténer pour n'avoir plus qu'une seul trame de données, dans un soucis de pertinence nous conserverons uniquement que les données du gyroscope et de l'accéléromètre, dans la base de données de cette étude deux smartphone ont été utilisés pour enregistrer l'activité des volontaires, l'un sera placé à la droite et l'autre à la gauche du volontaire ce qui donnera une base de données plus riche variée et aidera à la robustesse de notre classificateur, les deux sets de données seront donc chargés puis concaténés ensemble pour avoir notre trame de données, nous procéderons ensuite à l'étape de labélisation de notre jeu de données pour les différentes activités qui seront étudiées.

3.1.2 Pré-traitement des données

La feuille de route pour le prétraitement des données sera comme suit :

- Étape 1 : Puisque l'ensemble des données ont été collectées par 36 personnes différentes en utilisant leurs appareils respectifs, tous ces fichiers seront concaténés en un seul.
- Étape 2 : diviser le jeu de données en deux distributions : Entraînement et Validation.
- Étape 3 : Encodage des étiquettes en nombres.
- Étape 4 : Puisque les données collectées par les capteurs à chaque horodatage sont continues par nature, il est préférable de les convertir en données de séries temporelles afin qu'elles puissent s'adapter comme une séquence sur un horodatage dans les LSTM ou les GRU (modèles continus).

Note Pour la partie prétraitement des données nous avons procédé à une découpe du jeu de données en fonction des différents axe X,Y,Z et nous avons labélisé chaque données en fonction de nos différentes activité : walking, standing, jogging, sitting, biking, upstairs, downstairs, upsatirs, nous avons ensuite affecté à chaque label un numéro pour la partie encodage des labels.

3.1.2.1 Diviser les données en données d'entrainement et de validation

Notre trame de données a été divisée entre deux jeux de données un pour l'entraînement et le second pour la validation avec une répartition de 80% pour l'entrainement et 20% de validation afin d'avoir une robustesse et une précision correctes.

3.1.2.2 Convertir un ensemble de données en une séquence de séries temporelles

Les valeurs de sortie des capteurs du smartphone sont continues et dépendent de leur horodatage, comme nous pouvons le voir dans l'ensemble de données et les graphiques. Cela signifie qu'il est impossible de prédire quoi que ce soit en observant un seul point de données, car chaque valeur dépend de sa valeur précédente. Une valeur continue ne peut être prédite qu'en observant certaines valeurs dans une fenêtre fixe. Mais qu'est-ce que cela signifie ? Voyons-le à l'aide de cet exemple 3.2.

Supposons que nous avons un modèle qui peut faire des prédictions en prenant des valeurs continues en entrée.

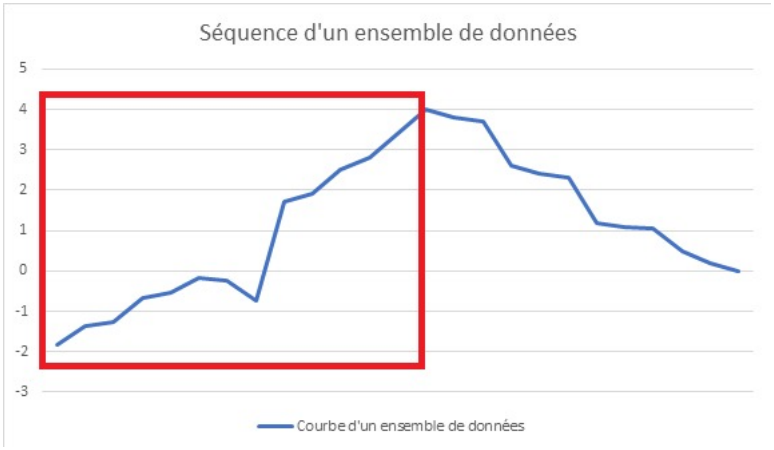


FIGURE 3.2 – Séquence d'un ensemble de données avec fenêtre de prédiction

Il essaie déjà de classifier ces deux sorties. Maintenant, nous avons un ensemble de données à prédire, donc nous ne pouvons pas introduire un seul point de l'ensemble de données dans le modèle, parce qu'il ne signifie rien et aussi nous ne pouvons pas obtenir quelque chose d'approprié en introduisant l'ensemble de données en une seule fois car ça ne correspond à aucune des classes étudiées. Donc, pour résoudre ce problème, nous pouvons créer une fenêtre de taille fixe que nous ferons glisser sur l'ensemble des données.

3.2 Modèle LSTM et compilation

L'architecture du modèle est la suivante :

- Couche 1 : La première couche est une couche LSTM pour l'apprentissage à partir d'une séquence de 100 points à chaque horodatage et qui renvoie la cartographie de la séquence.
- Couche 2 : La couche d'aplatissement est utilisée pour la sortie bidimensionnelle (nombre d'horodatages, nombre de caractéristiques) de la couche LSTM ci-dessus et la convertit en vecteur 1-d.
- Couche 3 : A partir de cette couche, la partie classificatrice commence, c'est une couche dense qui prend la sortie aplatie de la couche ci-dessus et la passe à la couche 4.
- Couche 4 : C'est une couche softmax qui prend l'entrée de la couche 3 et prédit la probabilité correspondant à chaque activité.

Le modèle utilise (Categorical Crossentropy) comme mesure de perte et Adam ou Adaptive Momemtum comme optimiseur de descente de gradient.

Il a été formé pour 50 époques avec le rappel ModelCheckpoint pour sauvegarder le meilleur modèle pendant la formation.

3.3 Entrainement

La formation du réseau neuronal se fera sous 50 époque où on notera 4 paramètres précision, Pertes et la valeur pour ces deux dernières

Epoque	Précision	Pertes	Valeur de pertes	Valeur de précision
1	0.6757	0.9302	0.5659	0.8029
10	0.9002	0.2686	0.2592	0.9011
20	0.9452	0.1619	0.1783	0.9394
30	0.9635	0.1117	0.1525	0.9507
40	0.9724	0.0847	0.1336	0.9589
50	0.9779	0.0689	0.1357	0.9585

TABLE 3.1 – Tableau de Résultat d'entraînement

On remarque avec le tableau que la précision augmente au fur et a mesure et que les pertes baissent.

3.4 Exportation et déploiement

Le modèle après formation va être exporté au format Hadoop (H5), La raison de l'exportation du fichier au format Hadoop (h5) vers le fichier Proto Buffer (.pb) est qu'il est léger et sera pris en charge par Tensorflow lite pour le déploiement dans les appareils Android.

Le modèle pourra ensuite être déployé sur Android sur des appareils qui supportent tensorflow lite pour pouvoir procédé à une prédiction de l'activité humaine en temps réel.

3.5 Base de données utilisées

3.5.1 Protocole d'enregistrement de la base de données :

La base de données sera enregistré grâce a une application avec une interface graphique comme sur la figure suivante 3.3 afin d'avoir une base de données de même nature que la base de données initial que nous allons enrichir, cette méthode d'enregistrement est similaire a celle utilisé a la base de données initial, l'application demande a l'utilisateur de sélectionné l'activité qu'il souhaite enregistrer parmi différente proposition, une activité jamais enregistrer auparavant peut être ajouté, la fréquence d'échantillonage peut être sélectionnées pour minimisé la perte de données.

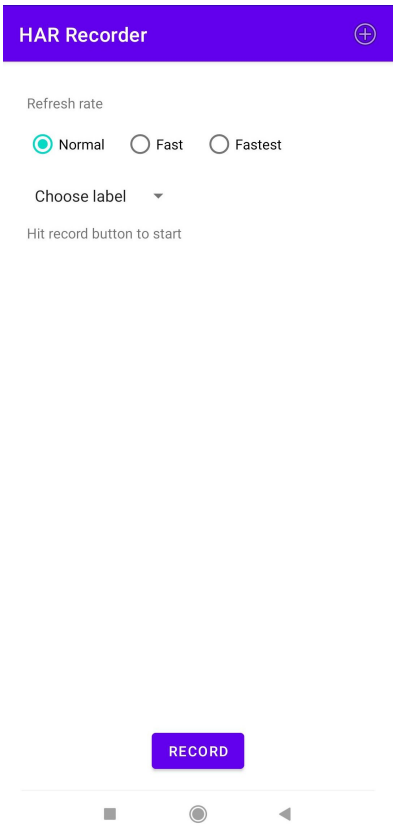


FIGURE 3.3 – Interface de l'application d'enregistrement

Un fichier CSV sera ensuite généré a la fin de l'enregistrement, qui comportera les données des trois axe (x, y ,z) des différents capteurs embarqué sur le smartphone (accélerometre, gyrscope), le fichier sera ensuite chargé avec les enregistrements des autres participants, ensuite l'ensemble de données sera converti en une séquence temporelles.

3.5.2 Base de données utilisées sous MATLAB Classification Learner

Les expériences ont été réalisées avec un groupe de 30 volontaires dans une tranche d'âge de 19 à 48 ans. Chaque personne a effectué les six activités mentionnées précédemment en portant le smartphone à la taille. Les expériences ont été enregistrées sur vidéo pour faciliter l'étiquetage des données. La base de données obtenue a été divisée aléatoirement en deux ensembles, où 70% des modèles ont été utilisés à des fins d'entraînement et 30% comme données de test : l'ensemble d'entraînement est ensuite utilisé pour entraîner des classificateur SVM, kNN, TREE qui sont décrit dans les sections 1.1.1, 1.1.2, 1.1.3. Un smartphone a été exploité pour les expériences, car il contient un accéléromètre et un gyroscope pour mesurer l'accélération linéaire triaxiale et la vitesse angulaire respectivement à une fréquence constante de 50 Hz, ce qui est suffisant pour capturer le mouvement du corps humain.

Pour la Reconnaissance d'activités, une application pour smartphone basée sur le système d'exploitation Android de Google peut être développé. Le processus de reconnaissance commence par l'acquisition des signaux du capteur, qui sont ensuite

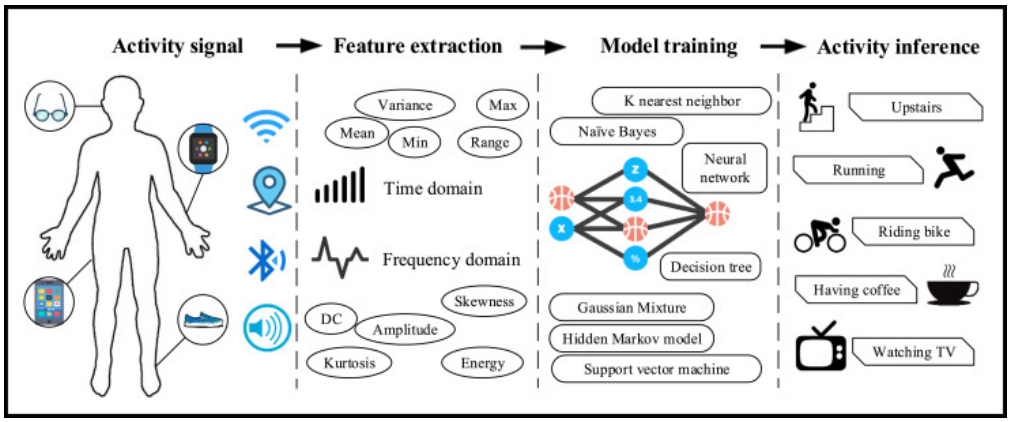


FIGURE 3.4 – Pipeline du processus de reconnaissance des activités[41].

pré traités en appliquant des filtres de bruit, puis échantillonnés dans des fenêtres glissantes à largeur fixe de 2,56 secondes et à recouvrement de 50 %. À partir de chaque fenêtre, un vecteur de 17 caractéristiques est obtenu en calculant les variables des signaux d'accéléromètre dans le domaine temporel et fréquentiel (par exemple, la moyenne, l'écart type, la zone d'amplitude du signal, l'entropie, la corrélation entre paires de signaux, etc.) La transformation de Fourier rapide est utilisée pour trouver les composantes de fréquence du signal. L'ensemble du pipeline du processus de reconnaissance d'activité est présenté dans la figure 3.4.

3.5.3 Base de données utilisées sous Google Colab et TensorFlow

La base de données utilisées est une base de données OpenSource pour un projet de recherche du département d'informatique et de sciences de l'information de l'Université de Fordham situé au États Unis et toute information relative a la base de données est tiré du papier publié de leur recherche "Activity Recognition using Cell Phone Accelerometers"[42].

Afin de collecter des données pour cette tâche d'apprentissage supervisé, il a été nécessaire de demander à un grand nombre d'utilisateurs de porter un téléphone intelligent Android tout en effectuant certaines activités quotidiennes. Les auteurs ont ensuite demandé à vingt-neuf sujets volontaires de porter un téléphone intelligent tout en effectuant une série d'activités spécifiques. Ces sujets portaient le téléphone Android dans la poche avant de leur pantalon et devaient marcher, courir, monter et descendre des escaliers, s'asseoir et rester debout pendant des périodes spécifiques.

La collecte des données était contrôlée par une application que développée par les auteurs et qui s'exécutait sur leurs téléphones. Cette application, par le biais d'une simple interface utilisateur graphique simple, a permis d'enregistrer le nom de l'utilisateur, de démarrer et d'arrêter la collecte de données, et d'étiqueter l'activité en cours. L'application a permis de contrôler les données des capteurs (par exemple, GPS, accéléromètre) qui étaient collectées et à quelle fréquence. (fréquence de collecte). Dans tous les cas, les données de l'accéléromètre ont été collectées toutes les 50 ms, soit 20 échantillons par seconde. La collecte des données a été supervisée par

l'un des membres de l'équipe WISDM pour en assurer la qualité.

3.6 Tensorflow

FIGURE 3.5 – TensforFlow LOGO

TensorFlow est une interface pour exprimer des algorithmes d'apprentissage automatique et une implémentation pour les exécuter. Un calcul exprimé à l'aide de TensorFlow peut être exécuté avec peu ou pas de changement sur une grande variété de systèmes hétérogènes, allant des appareils mobiles tels que les téléphones et les tablettes jusqu'aux systèmes distribués à grande échelle de centaines de machines et de milliers d'appareils de calcul tels que les cartes GPU[43].

Le système est flexible et peut être utilisé pour exprimer une grande variété d'algorithmes, y compris des algorithmes de formation et d'inférence pour les modèles de réseaux neuronaux profonds, et il a été utilisé pour mener des recherches et pour déployer des systèmes d'apprentissage automatique en production dans plus d'une douzaine de domaines de l'informatique et d'autres, tels que la reconnaissance vocale, la vision par ordinateur, la robotique, la recherche d'informations, le traitement du langage naturel, l'extraction d'informations géographiques et la découverte de médicaments par ordinateur. L'API TensorFlow et une implémentation de référence ont été publiées en tant que package open source sous la licence Apache 2.0 en novembre 2015 et sont disponibles sur `www.tensorflow.org`.

3.6.1 Tensorflow lite

TensorFlow Lite est une interface d'apprentissage profond multiplateforme, open-source et prêt à l'emploi, qui convertit un modèle pré-entraîné dans TensorFlow en un format spécial qui peut être optimisé pour la vitesse ou le stockage.

Le modèle au format spécial peut être déployé sur des appareils périphériques tels que des mobiles utilisant Android ou iOS ou des appareils embarqués basés sur Linux comme Raspberry Pi ou des microcontrôleurs pour effectuer l'inférence au périphérique Edge.

Les caractéristiques d'un modèle d'apprentissage profond pour l'inférence au niveau des périphériques Edge sont :

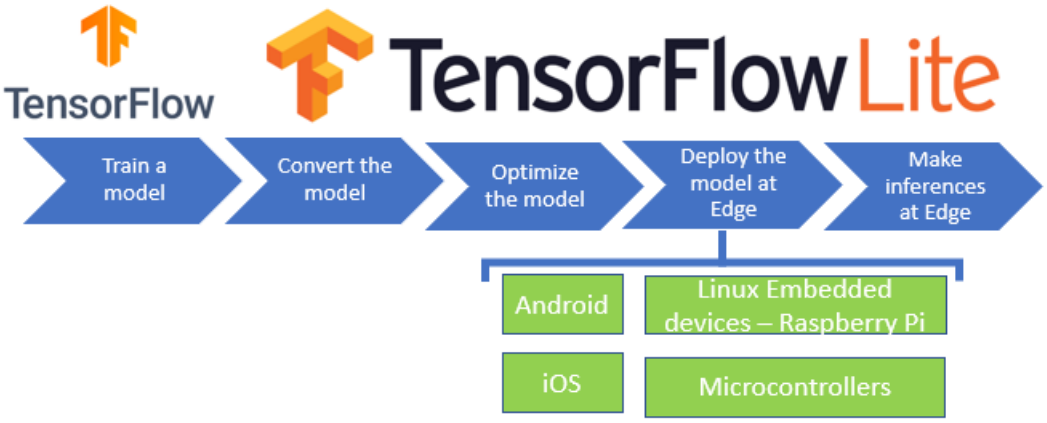


FIGURE 3.6 – Tensorflow lite feuille de route de déploiement[44]

1. **Léger** : Les dispositifs de périphérie disposent de ressources limitées en termes de stockage et de capacité de calcul. Les modèles d'apprentissage profond sont gourmands en ressources. Les modèles que nous déployons sur les périphériques doivent donc être légers et présenter des tailles binaires réduites.
2. **Faible latence** : Les modèles d'apprentissage profond à la périphérie devraient faire des inférences plus rapides, indépendamment de la connectivité du réseau. Comme les inférences sont effectuées sur le périphérique, un aller-retour entre le périphérique et le serveur sera éliminé, ce qui rendra les inférences plus rapides.
3. **Sécurisé** : Le modèle est déployé sur le dispositif Edge, les inférences sont effectuées sur le dispositif, aucune donnée ne quitte le dispositif ou n'est partagée sur le réseau, il n'y a donc aucun problème de confidentialité des données.
4. **Consommation d'énergie optimale** : Le réseau a besoin d'une grande quantité d'énergie, et les périphériques ne sont pas forcément connectés au réseau, d'où une faible consommation d'énergie.
5. **Pré-entraînement** : Les modèles peuvent être formés sur site ou dans le cloud pour différentes tâches d'apprentissage profond telles que la classification d'images, la détection d'objets, la reconnaissance vocale, etc. et peuvent être facilement déployés pour faire des déductions au niveau de la périphérie.

3.6.1.1 Comment fonctionne Tensorflow Lite (TF Lite) ?

Sélectionner et former un modèle Disons que vous voulez effectuer la tâche de classification d'images. La première chose à faire est de choisir le modèle pour cette tâche. Vous avez le choix entre :

- Créer un modèle personnalisé
- Utiliser un modèle pré-entraîné comme InceptionNet, MobileNet, NASNet-Large, etc.
- Appliquer l'apprentissage par transfert sur un modèle pré-entraîné.

Convertir le modèle à l'aide de Converter Une fois le modèle formé, vous devez le convertir en version Tensorflow Lite. Le modèle TF Lite est un modèle de format spécial, efficace en termes de précision et également une version légère qui occupera moins d'espace. Ces caractéristiques font des modèles TF Lite la bonne solution pour travailler sur des appareils mobiles et embarqués.

3.7 Matlab Classification Learner

En apprentissage automatique, le choix du bon modèle de classification est souvent une question d'essais et d'erreurs. L'application Classification Learner app dans Statistics and Machine Learning Toolbox™ dans Matlab développée par la société mathworks est un excellent outils pour la classification et une excellent porte d'entrée pour l'étude des modèle de classification.

L'application Classification Learner vous permet d'explorer l'apprentissage automatique supervisé à l'aide de divers classificateurs. Vous pouvez explorer vos données, sélectionner des caractéristiques, spécifier des schémas de validation croisée, former des modèles et évaluer les résultats. Vous pouvez effectuer une formation automatisée pour rechercher le meilleur modèle de classification, notamment les arbres de décision, l'analyse discriminante, les machines à vecteurs de support (SVM), la régression logistique, les voisins les plus proches (KNN) et la classification d'ensemble, les Bayes naïfs, l'approximation à noyau, l'ensemble et la classification par réseau neuronal, etc.

CLassification Learner permet également [45] :

- Effectuer un apprentissage automatique supervisé en fournissant un ensemble connu de données d'entrée (observations ou exemples) et des réponses connues aux données (c'est-à-dire des étiquettes ou des classes).
- Utiliser les données pour former un modèle qui génère des prédictions pour la réponse à de nouvelles données.
- Utiliser le modèle avec de nouvelles données, ou pour en savoir plus sur la classification, vous pouvez exporter le modèle vers un autre site,
- Exporter le modèle vers l'espace de travail ou générer du code MATLAB® pour recréer le modèle formé.

3.8 Google Colab

Google Colaboratory (également connu sous le nom de Colab) est un service cloud basé sur Jupyter Notebooks pour diffuser l'enseignement et la recherche sur l'apprentissage automatique. Il fournit un runtime entièrement configuré pour l'apprentissage profond et un accès gratuit à un GPU robuste [46].

L'utilisation de colab offre un environnement basé sous jupyter complètement dédié a python et à l'apprentissage automatique, cette catégorie d'applications repose généralement sur des calculs lourds sur des ensembles de données massifs. Par conséquent, le calcul parallèle est traditionnellement envisagé pour exécuter le processus de formation en un temps raisonnable.

Les unités de traitement graphique (GPU) sont des dispositifs massivement parallèles candidats pour effectuer une telle tâche. Ce type d'accélérateur est omniprésent, accessible et fournit un taux de GFlops/Dollar élevé, en d'autre terme leur coûts est très cher et leur accessibilité est réduite. Ces principaux frameworks d'apprentissage profond sont programmés pour les GPU NVIDIA (CUDA)[47] .

En utilisant Google Colaboratory, on doit aussi parler de Jupyter Notebooks, la technologie sur laquelle Colaboratory est basé. Jupyter est un outil à code source ouvert et basé sur un navigateur qui intègre des langages interprétés, des bibliothèques et des outils pour la visualisation. Un Notebook Jupyter peut fonctionner soit localement ou sur le cloud. Chaque document est composé de plusieurs cellules, où chaque cellule contient un langage de script ou un code markdown, et la sortie est intégrée dans le document. Les résultats typiques comprennent du texte, des tableaux, des diagrammes et des graphiques, l'utilisation de cette technologie facilite le partage et la reproduction des travaux scientifiques puisque les expériences et les résultats sont présentés d'une manière autonome [48].

Les carnets de notes Colaboratory sont basés sur Jupyter et fonctionnent comme un objet Google Docs : ils peuvent être partagés et les utilisateurs peuvent collaborer sur le même notebook. Colaboratory fournit des moteurs d'exécution Python 2 et 3 pré-configurés avec les bibliothèques essentielles d'apprentissage automatique et d'intelligence telles que TensorFlow, Matplotlib et Keras. La machine virtuelle sous le runtime (VM) est désactivée après un certain temps, et toutes les données et configurations de l'utilisateur sont perdues. Cependant, le carnet de notes est préservé, et il est également possible de transférer des fichiers du disque dur de la VM vers le compte Google Drive de l'utilisateur. Enfin, ce service de Google fournit un accélération par GPU, entièrement configuré avec les logiciels décrits précédemment. L'infrastructure du Google Colaboratory est hébergée sur la plateforme Google Cloud [49].

3.9 Android Studio

FIGURE 3.7 – Android Studio LOGO

Android Studio est l'environnement de développement intégré (IDE) officiel pour

le développement d'applications Android, basé sur IntelliJ IDEA . En plus de l'éditeur de code et des outils de développement puissants d'IntelliJ, Android Studio offre encore plus de fonctionnalités qui améliorent la productivité lors de la création d'applications Android [50], telles que :

- Un système de construction flexible basé sur Gradle
- Un émulateur rapide et riche en fonctionnalités
- Un environnement unifié dans lequel vous pouvez développer pour tous les appareils Android.
- L'application des modifications pour pousser les changements de code et de ressources dans l'application en cours d'exécution sans la redémarrer.
- Des modèles de code et l'intégration de GitHub pour aider à créer des fonctionnalités d'application communes et à importer des exemples de code.
- Des outils et des frameworks de test étendus
- Des outils Lint pour détecter les problèmes de performance, de convivialité, de compatibilité de version et autres.
- Une prise en charge de C++ et NDK [1]
- Une prise en charge intégrée de Google Cloud Platform, ce qui facilite l'intégration de Google Cloud Messaging et d'App Engine.

1. Native Development Kit

Chapitre 4

Résultats de l'application

4.1 Résultats de la classification sous MATLAB Classfication Learner

L'outil Classification Learner comme discuté précédemment permet l'étude de plusieurs types de classificateur classique avec différents outils d'étude que nous utiliseront pour étudier les diffèrentes variantes des classificateurs SVM, KNN, arbre de décision.

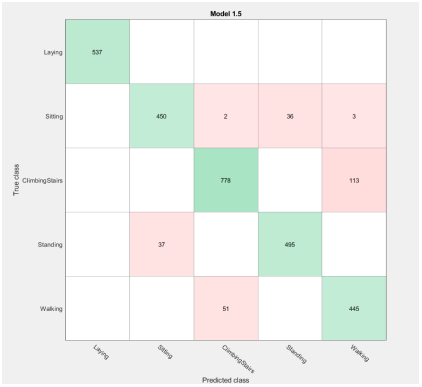
4.1.1 KNN

Type de classificateur	Interprétabilité [1]	Flexibilité du modèle
Fine KNN 	Dure	Haute
Medium KNN 	Dure	Moyenne
Cubic KNN 	Dure	Moyenne

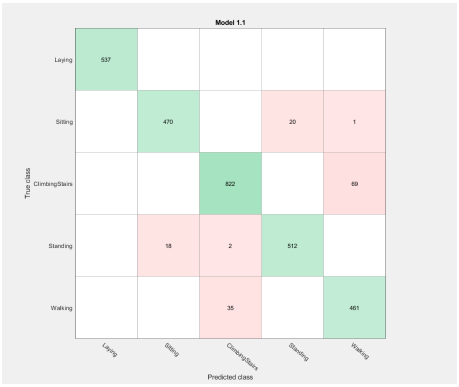
TABLE 4.1 – Tableau descriptif pour les différents KNN dans Classification Learner[52]

- Fine KNN : Distinctions finement détaillées entre les classes. Le nombre de voisins est fixé à 1.
- Medium KNN : Distinctions moyennes entre les classes. Le nombre de voisins est fixé à 10.
- Cubic KNN : Distinctions moyennes entre les classes, en utilisant une métrique de distance cubique. Le nombre de voisins est fixé à 10.

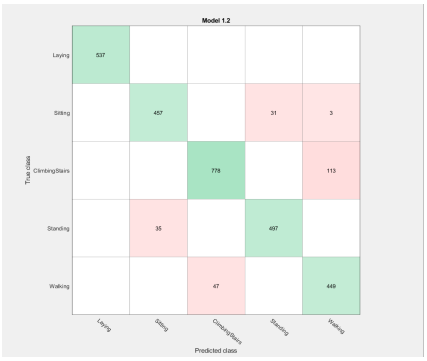
1. L'interprétabilité est le degré à quel point un humain peut expliquer de manière cohérente les prédictions du modèle [51]



(a) Cubic KNN



(b) Fine KNN



(c) Medium KNN

FIGURE 4.1 – Matrice de confusion KNN

	Temps d'apprentissage	Vitesse de prédiction	Précision
Fine kNN	9.1972 sec	5900 obs/sec	95.1%
Medium kNN	8.423 sec	5500 obs/sec	92.2%
cubic kNN	33.75 sec	370 obs/sec	91.8%

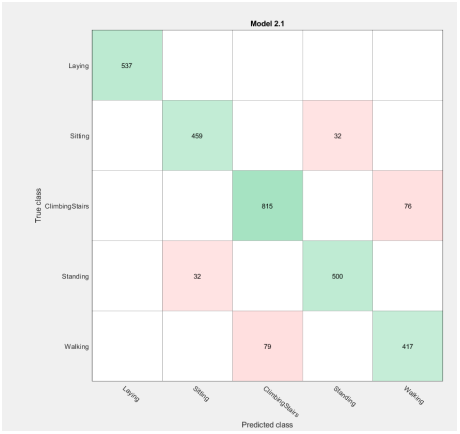
TABLE 4.2 – Tableau des résultats de kNN

Le KNN possède le second meilleur score au niveau de la précision de tous les types des classificateurs que nous avons testé avec 95.1% sur le fine KNN, ce dernier possède un temps de formation moyen entre le medium KNN et le cubic KNN et une vitesse de prédiction qui est la plus haute pour les KNN avec 5900 obs/sec.

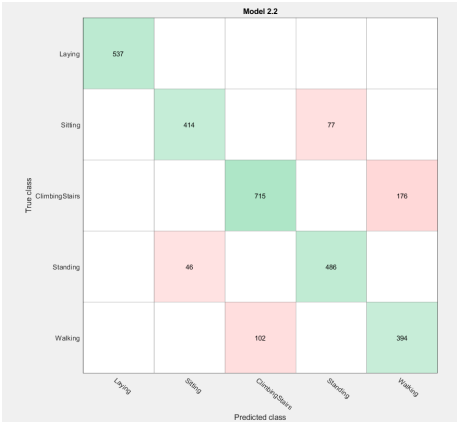
4.1.2 Arbre de décision

Type de classificateur	Interprétabilité	Flexibilité du modèle
Coarse Tree 	Facile	Basse
Medium Tree 	Facile	Moyenne
Fine Tree 	Facile	Haute

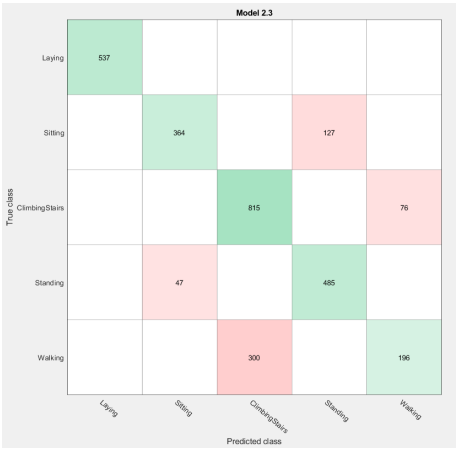
TABLE 4.3 – Tableau descriptif pour les différents arbre de décision dans Classification Learner[52]



(a) Fine TREE



(b) Medium TREE



(c) Coarse TREE

FIGURE 4.2 – Matrice de confusion Arbre de décision

	Temps d'apprentissage	Vitesse de prédiction	Précision
Fine Tree	11.112 sec	13000 obs/sec	92.6%
Medium Tree	10.503 sec	14000 obs/sec	86.4%
Coarse Tree	10.334 sec	12000 obs/sec	81.3%

TABLE 4.4 – Tableau des résultats de Tree

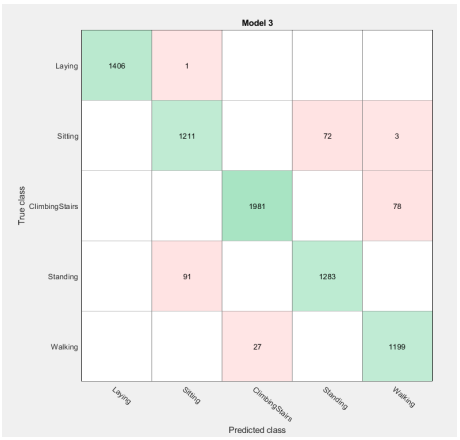
- Coarse Tree : Peu de feuilles pour faire des distinctions grossières entre les classes (le nombre maximum de divisions est de 4).
- Medium Tree : Nombre moyen de feuilles pour des distinctions plus fines entre les classes (le nombre maximum de divisions est de 20).
- Fine Tree : De nombreuses feuilles pour faire de nombreuses distinctions fines entre les classes (le nombre maximum de divisions est de 100)[52].

On remarque des résultats obtenus durant notre phase de test que le fine tree a la plus haute précision avec une vitesse qui se situe entre le medium et le coarse tree, le medium ayant le plus haut score au niveau de la vitesse de prédiction et le coarse tree. Ce dernier a le temps d'apprentissage le plus court alors que le fine a le temps d'apprentissage le plus long.

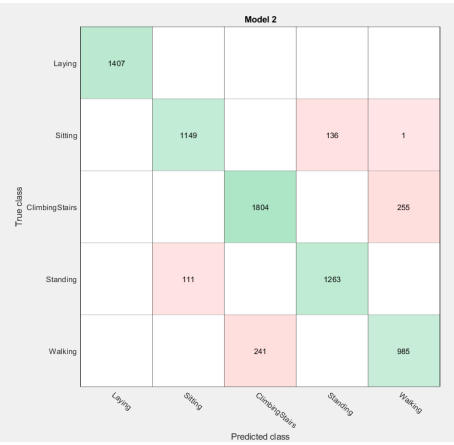
4.1.3 SVM

Type de classificateur	Interprétabilité	Flexibilité du modèle
SVM linéaire 	Facile	Basse
Cubic SVM 	Dure	Moyenne
SVM medium gaussien 	Dure	Haute

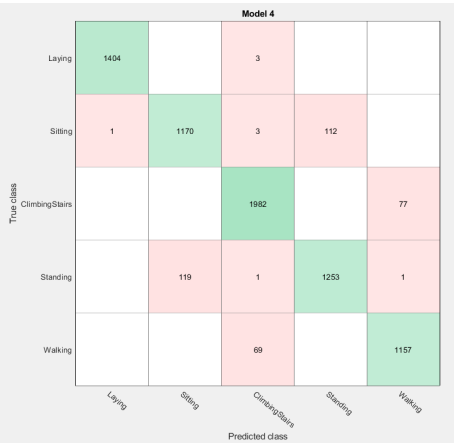
TABLE 4.5 – Tableau descriptif pour les différents SVM dans Classification Learner[52]



(a) Cubic SVM



(b) SVM Linéaire



(c) SVM medium gaussien

FIGURE 4.3 – Matrice de confusion SVM

- SVM linéaire : Effectue une simple séparation linéaire entre les classes.
- SVM cubic : Distinctions moyennes, avec un noyau cubic $k(x_i, x_j) = (x_i^T x_j)^3$.
- SVM Mediume gaussien : Distinctions moyennes, avec une échelle de noyau fixée à racine carré de P.

	Temps d'apprentissage	Vitesse de prédiction	Précision
Linéar SVM	12.411 sec	31000 obs/sec	89.9%
Medium Guassien SVM	17.994 sec	7400 obs/sec	94.7%
Cubic SVM	77.369 sec	17000 obs/sec	96.3%

TABLE 4.6 – Tableau des résultats de SVM

Le SVM obtient le meilleur score au niveau de la précision très proche du fine KNN avec 96.3% pour le cubic SVM. Le Medium gaussien SVM possède cependant le plus faible score de prédiction de tous les SVM avec 7400 obs/sec par rapport au linéaire SVM 31000 obs/sec et au cubic SVM avec 17000 obs/sec. En terme de temps d'apprentissage le medium gaussien SVM se situe au milieu entre le linéaire et le cubic, ce dernier et le plus lent de tous les classificateurs.

4.2 Résultats LSTM sous (TensorFlow/Google Colab)

Le modèle LSTM attend des séquences de longueur fixe comme données d'apprentissage. Nous allons utiliser une méthode simple pour les générer. Chaque séquence (ou fenêtre) générée contient 50 enregistrements correspondant à 2,5 secondes d'activité (rappelons que la fréquence de collecte des données mentionnée dans le site Web source est de 20 Hz)[42].

Notons que nous avons considéré ici des fenêtres de données qui se chevauchent (avec un chevauchement de 80%). Comme notre activité est continue, ce chevauchement permet de s'assurer que chaque fenêtre suivante porte certaines informations de la fenêtre précédente.

A ce stade, si nous vérifions la forme des données transformées, nous trouvons (108531, 50, 3) (nombre de séquence, nombre d'enregistrement (époque), nombre de paramètre (x, y, z)).

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 128)	67584
dropout (Dropout)	(None, 128)	0
dense (Dense)	(None, 64)	8256
dense_1 (Dense)	(None, 6)	390

Total params: 76,230
Trainable params: 76,230
Non-trainable params: 0

FIGURE 4.4 – Résultat base de données

108531 séquences de 200 rangées, chacune contenant des données x, y et z. La taille de notre ensemble de données brutes d'origine a été considérablement réduite après la transformation. Notez que l'étiquette de classe assignée à une séquence (fenêtre) est l'activité qui se produit le plus fréquemment dans cette fenêtre.

Comme vous pouvez le voir, nous avons plus de 76 000 paramètres entraînables. En raison du nombre élevé de paramètres entraînables, le modèle a tendance à s'adapter facilement. C'est pourquoi une couche d'exclusion est recommandée pour éviter l'overfitting.

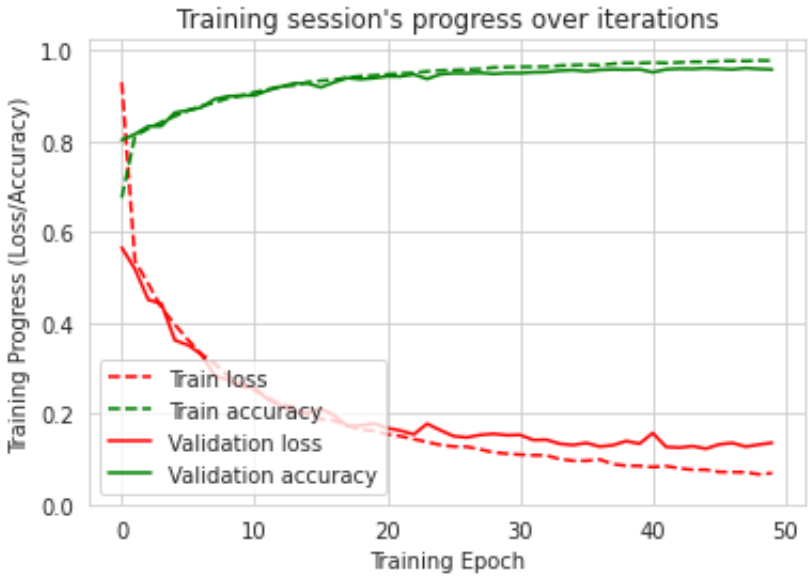


FIGURE 4.5 – Graphique de la session D'apprentissage

Notre modèle LSTM semble bien apprendre avec une précision supérieure à 97% et une perte d'entropie croisée bien inférieure à 0,2 pour les données de formation et de validation, le score de pertes quant à lui est de 0.0689 avec une valeur de 0.1357.

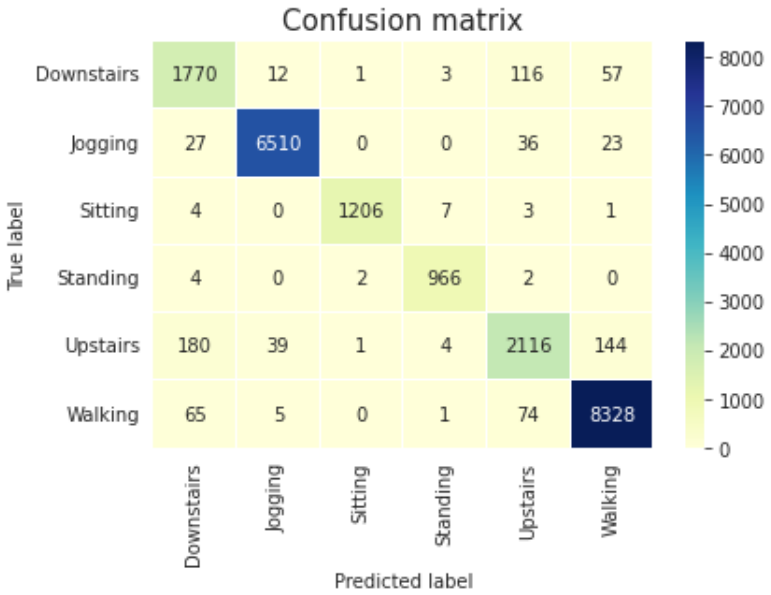


FIGURE 4.6 – Matrice de confusion

Comme on peut le remarquer dans la matrice de confusion 4.6, les deux activités les plus courantes de notre jeu de données, à savoir la marche et le jogging (Walking, Jogging), sont correctement classées avec une très grande précision. Bien que les activités Assis et Debout (Sitting, Standing) soient des classes minoritaires, notre modèle est capable de les différencier avec précision.

La précision n'est pas aussi élevée que celle des autres classes pour les activités en haut et en bas. Ceci est attendu car ces deux activités sont très similaires et les données sous-jacentes peuvent ne pas être suffisantes pour les différencier avec précision.

Concernant la vitesse d'apprentissage on a eu un temps approximatif de 18ms par époque sauf pour la première qui a pris 26 ms ce qui fait un total de 908ms pour le temps de formation.

```
22/22 [==============================] - 0s 10ms/step - loss: 0.1264 - accuracy: 0.9626
Test Accuracy : 0.962638795375824
Test Loss : 0.12637069821357727
```

FIGURE 4.7 – Score de validation

Concernant l'étape de validation, on a procédé ensuite a une validation sous 22 étapes comme montré dans les résultat suivant 4.7 avec un temps de détection par étape de 10 ms ce qui est satisfaisant pour avoir une prédiction correcte.

```
22/22 [==============================] - 0s 9ms/step - loss: 0.1162 - accuracy: 0.9659
Test Accuracy : 0.9658635258674622
Test Loss : 0.11618278920650482
```

FIGURE 4.8 – Score de validation pour uniquement la base de données enregistrée

Un second score de validation a été effectuer uniquement sur la base de données enregistrer, on remarque un léger meilleur score de précision et des pertes légèrement plus basse, on déduit donc que le protocole suivi pour l'enregistrement de la base de données a été efficace, une perspective d'enregistrement d'une base de données avec plus d'activité et aussi plus de participants peut être conclus.

4.3 Discussion des résultats

Les algorithmes de classification ont pue être étudié puis entraîné avec la base de données que l'ont a vu dans la sous section 3.5.2, a partir des résultats de la section 4.1 ont déduit que en terme de vitesse d'entraînement le medium KNN a le temps de formation le plus court, le SVM linéaire lui possède le plus d'observation par seconde et le SVM Cubic le plus haut score de précision, on déduit que le SVM Cubic est le classificateur le plus efficace avec le meilleurs compromis en terme de temps d'apprentissage (17.994 sec), vitesse de prédiction (31 000 obs/sec) et précision (96.3%).

En ce qui concerne le LSTM il a était entraîner avec la base de données enrichi, puis lors de l'étape de test il a testé une première fois avec uniquement la base de données initial puis uniquement avec les données que nous avons enregistrées, on remarque que le réseau réagi de façon très précise au deux réseaux et même que le score pour la base de données enregistré était légèrement plus élevés (0.9659%), ainsi on déduit que le protocole utilisé pour l'enregistrement de la base de données était efficace.

On déduit donc que le LSTM est l'algorithme le plus efficace pour la détection d'activité humaine sur la base d'enregistrement de capteur (accéléromètre, gyroscope), cela se traduit grâce à la bonne flexibilité de l'algorithme en ce qui concerne le traitement de séquence de données temporelles grâce sa capacité de mémoire à court et long terme qui permet de non seulement filtrer les données non pertinentes mais aussi gardé les données pertinentes en mémoire bien plus longtemps que un RNN classique, nous recommandons alors dans une perspective de suite à ce PFE de procédé à l'enregistrement d'une base de données grâce au protocole utiliser dans celui-ci et d'ensuite procéder au développement d'une application ANDROID avec l'outil ANDROID STUDIO ainsi que le déploiement d'un LSTM dans cette dernière pour permettre la reconnaissance d'activité humaine sur smartphone.

Conclusion Générale

La reconnaissance de l'activité humaine représente un domaine de recherche important, notamment pour ses applications possibles d'aide à l'assistance de vie ou encore de la sécurité basée sur la surveillance.

Dans ce travail, nous nous sommes d'abord orientés vers une étude entre différents classificateurs que sont les KNN, les SVM et les arbres de décision et d'un réseau de neurones récurrents (LSTM). Suite à une étude bibliographique en ce début de second semestre, nous avons pu constater que les méthodes d'apprentissage profond surpassent les méthodes d'apprentissage classiques.

Nous avons donc choisi de nous orienter vers un modèle d'un réseau de neurones récurrents (LSTM), qui permet d'exploiter les dépendances temporelles entre les séries chronologiques des différentes données de capteurs.

Le choix de l'architecture du modèle à utiliser représente une étape cruciale. Ce choix impactera directement les performances du réseau.

Suite a l'étude des différents classificateurs sous MATLAB Classification Learner, nous avons déduit que le SVM est le plus performant entre tous les classificateurs étudiés, cependant il reste tout de même inférieur au score du LSTM testé sous Google colab et Tensorflow avec 97 % de précision.

Dans l'ensemble après une étude bibliographique et architecturale, nous avons déduit que le LSTM est plus performant que le SVM dans tous les scénarios. Ceci est dû à sa capacité mémoire lui permettant de se souvenir ou d'oublier les données d'une manière plus efficace que les SVM n'ont aucune capacité mémoire. Le LSTM fournit une plus grande précision et garde également une trace des différentes activités prédites pendant une période plus longue, tout en effectuant des meilleures prédictions par rapport aux SVM.

Ainsi, pour une future implémentation de ce travail sur un dispositif mobile intelligent, nous préconisons une utilisation du LSTM avec la librairie dédiée TensorFlow Lite, qui est mieux adaptée aux dispositifs mobiles, limités en capacités de calcul et en espace de stockage par rapport à l'outil Collab. Les classifieurs peuvent également être considérés dans ce contexte mais leurs résultats seraient moins fiables dans la reconnaissance de l'activité humaine par rapport aux algorithmes de l'apprentissage automatique. Cependant, ils ont l'avantage d'avoir des périodes d'entraînement plus courtes, ce qui leurs permet de réaliser des prédictions en des temps plus réduits que les outils de l'apprentissage automatique.

Bibliographie

[1] J. Ekholm et S. Fabre, "Forecast : Mobile data trac and revenue, worldwide, 2010-2015," *Gartner Mobile Communications Worldwide.(July 2011)*, 2011.

[2] D. J. Cook et S. K. Das, "Pervasive computing at scale : Transforming the state of the art," *Pervasive and Mobile Computing*, t. 8, n° 1, p. 22-35, 2012.

[3] F. R. Allen, E. Ambikairajah, N. H. Lovell et B. G. Celler, "Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models," *Physiological measurement*, t. 27, n° 10, p. 935, 2006.

[4] Aized Amin Soofi et Arshad Awan, "Classification Techniques in Machine Learning : Applications and Issues," en, *Journal of Basic & Applied Sciences*, t. 13, p. 459-465, jan. 2017, issn : 1927-5129, 1814-8085. doi : 10.6000/1927-5129.2017.13.76. adresse : https://setpublisher.com/pms/index.php/jbas/article/view/1715 (visité le 21/04/2022).

[5] B. W. Silverman et M. C. Jones, "E. fix and jl hodges (1951) : An important contribution to nonparametric discriminant analysis and density estimation : Commentary on fix and hodges (1951)," *International Statistical Review/Revue Internationale de Statistique*, p. 233-238, 1989.

[6] T. Cover et P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, t. 13, n° 1, p. 21-27, 1967.

[7] M. Hellman et J. Raviv, "Probability of error, equivocation, and the Chernoff bound," *IEEE Transactions on Information Theory*, t. 16, n° 4, p. 368-372, 1970.

[8] K. Fukunaga et L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on information theory*, t. 21, n° 1, p. 32-40, 1975.

[9] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, t. 4, n° 2, p. 1883, 2009.

[10] S. Bermejo et J. Cabestany, "Adaptive soft k-nearest-neighbour classifiers," *Pattern Recognition*, t. 33, n° 12, p. 1999-2005, 2000.

[11] G. Bhattacharya, K. Ghosh et A. S. Chowdhury, "An affinity-based new local distance function and similarity measure for kNN algorithm," *Pattern Recognition Letters*, t. 33, n° 3, p. 356-363, 2012.

[12] A. Ali, M. Alrubei, S. Abdulwahed et M. M. Al-Ja'afari, "DIABETES DIAGNOSIS BASED ON KNN," t. 21, p. 175-181, jan. 2020.

[13] O. Altay, M. Ulas et K. E. Alyamac, "Prediction of the fresh performance of steel fiber reinforced self-compacting concrete using quadratic SVM and weighted KNN models," *IEEE Access*, t. 8, p. 92 647-92 658, 2020.

[14] J.-H. Zhai, N. Li et M.-Y. Zhai, "The condensed fuzzy k-nearest neighbor rule based on sample fuzzy entropy," in *2011 International Conference on Machine Learning and Cybernetics*, IEEE, t. 1, 2011, p. 282-286.

[15] S. Uddin, I. Haque, H. Lu, M. A. Moni et E. Gide, "Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction," *Scientific Reports*, t. 12, nº 1, p. 1-11, 2022.

[16] A. B. Hassanat, "Dimensionality invariant similarity measure," *arXiv preprint arXiv :1409.0923*, 2014.

[17] *Machine à vecteurs de support (SVM) : définition et cas d'usage*, fr. adresse : https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501879-machine-a-vecteurs-de-support-svm-definition-et-cas-d-usage/ (visité le 21/04/2022).

[18] V. Kecman, "Support Vector Machines – An Introduction," in *Support Vector Machines : Theory and Applications*, t. 177, Journal Abbreviation : Support Vector Machines : Theory and Applications, mai 2005, p. 605-605, isbn : 978-3-540-24388-5. doi : 10.1007/10984697_1.

[19] *Machine à vecteurs de support*, fr, Page Version ID : 192207500, mars 2022. adresse : https://fr.wikipedia.org/w/index.php?title=Machine_%C3%A0_vecteurs_de_support&oldid=192207500 (visité le 21/04/2022).

[20] H. Wannous, "CLASSIFICATION MULTI VUES DE RÉGIONS COULEUR-APPLICATION A L'ÉVALUATION 3D DES PLAIES CHRONIQUES," thèse de doct., Université d'Orléans, 2008.

[21] baeldung, *Multiclass Classification Using Support Vector Machines — Baeldung on Computer Science*, en-US, oct. 2020. adresse : https://www.baeldung.com/cs/svm-multiclass-classification (visité le 08/05/2022).

[22] A. Z. Tlib Dalila, "Optimisation Multi objectif pour la Sélection de modèle SVM ; Application l'indexation et recherche d'image," p. 30-32, 2014.

[23] *Un système de tri automatique des dattes par SVM 1vsR - PDF Free Download*. adresse : https://docplayer.fr/90329042-Un-systeme-de-tri-automatique-des-dattes-par-svm-1vsr.html (visité le 08/05/2022).

[24] A. Djeffal, "Utilisation des méthodes Support Vector Machine (SVM) dans l'analyse des bases de données," en, *Data mining*, p. 212,

[25] J. R. Quinlan, "Induction of decision trees," *Machine learning*, t. 1, nº 1, p. 81-106, 1986.

[26] M. Haghighat, H. Rastegari, N. Nourafza, N. Branch et I. Esfahan, "A review of data mining techniques for result prediction in sports," *Advances in Computer Science : an International Journal*, t. 2, nº 5, p. 7-12, 2013.

[27] H. Ming, N. Wenying et L. Xu, "An improved decision tree classification algorithm based on ID3 and the application in score analysis," in *2009 Chinese control and decision conference*, IEEE, 2009, p. 1876-1879.

[28] T. M. Mitchell, *Machine Learning*, en, sér. McGraw-Hill series in computer science. New York : McGraw-Hill, 1997, isbn : 978-0-07-042807-2.

[29] S. Hochreiter et J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, t. 9, nº 8, p. 1735-1780, nov. 1997, issn : 0899-7667. doi : 10.1162/neco.1997.9.8.1735. adresse : https://doi.org/10.1162/neco.1997.9.8.1735 (visité le 06/01/2022).

[30] B. Stuner, "Cohorte de réseaux de neurones récurrents pour la reconnaissance de l'écriture," thèse de doct., Normandie, 2018.

[31] A. Graves, "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*, Springer, 2012, p. 5-13.

[32] A. Graves et J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," *Advances in neural information processing systems*, t. 21, 2008.

[33] K. Cho, B. Van Merriënboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv :1406.1078*, 2014.

[34] G. Tanaka, T. Yamane, J. B. Héroux et al., "Recent advances in physical reservoir computing : A review," en, *Neural Networks*, t. 115, p. 100-123, juill. 2019, issn : 08936080. doi : 10.1016/j.neunet.2019.03.005. adresse : https://linkinghub.elsevier.com/retrieve/pii/S0893608019300784 (visité le 01/06/2022).

[35] C. Fernando et S. Sojakka, "Pattern Recognition in a Bucket," in *ECAL*, 2003.

[36] S. Hochreiter et J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, t. 9, nº 8, p. 1735-1780, 1997.

[37] U. Delabre, *Smartphonique : Expériences de Physique avec un smartphone*. Dunod, juin 2019. adresse : https://hal.archives-ouvertes.fr/hal-02457000.

[38] M. Krichen, "Utiliser les Smartphones pour la Détection de Différents Types d'Anomalies," thèse de doct., ReDCAD Laboratory, 2021.

[39] R. Dony et al., "Karhunen-loeve transform," *The transform and data compression handbook*, t. 1, nº 1-34, p. 29, 2001.

[40] K. Yata et M. Aoshima, "Effective PCA for high-dimension, low-sample-size data with noise reduction via geometric representations," *Journal of multivariate analysis*, t. 105, nº 1, p. 193-215, 2012.

[41] J. Wang, Y. Chen, S. Hao, X. Peng et L. Hu, "Deep learning for sensor-based activity recognition : A survey," *Pattern Recognition Letters*, t. 119, p. 3-11, 2019.

[42] S. A. M. Jennifer R. Kwapisz Gary M. Weiss, "Activity Recognition using Cell Phone Accelerometers," en, p. 9, 2010. adresse : https://www.cis.fordham.edu/wisdm/includes/files/sensorKDD-2010.pdf.

[43] M. Abadi, A. Agarwal, P. Barham et al., "TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems," en, p. 19,

[44] R. Khandelwal, *A Basic Introduction to TensorFlow Lite*, en, juin 2020. adresse : https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292 (visité le 14/05/2022).

[45] *Train models to classify data using supervised machine learning - MATLAB - MathWorks France*, 2016. adresse : https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/m/88253_92995_ML_Train_Classification_Models_App.pdf (visité le 02/06/2022).

[46] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque et P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *IEEE Access*, t. 6, p. 61 677-61 685, 2018, Conference Name : IEEE Access, issn : 2169-3536. doi : 10.1109/ACCESS.2018.2874767.

[47] N. Corporation, *TESLA V100 performance guide : Deep learning and HPC applications*, 2016. adresse : https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/v100-application-performance-guide.pdf (visité le 02/06/2022).

[48] B. M. Randles, I. V. Pasquetto, M. S. Golshan et C. L. Borgman, "Using the Jupyter Notebook as a Tool for Open Science : An Empirical Study," en, in *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, Toronto, ON, Canada : IEEE, juin 2017, p. 1-2, isbn : 978-1-5386-3861-3. doi : 10.1109/JCDL.2017.7991618. adresse : http://ieeexplore.ieee.org/document/7991618/ (visité le 02/06/2022).

[49] Google, *Google Colab*, 2018. adresse : https://research.google.com/colaboratory/faq.html (visité le 02/06/2022).

[50] *Meet Android Studio*, en. adresse : https://developer.android.com/studio/intro (visité le 14/05/2022).

[51] T. Miller, "Explanation in artificial intelligence : Insights from the social sciences," *Artificial intelligence*, t. 267, p. 1-38, 2019.

[52] *Choose Classifier Options - MATLAB & Simulink - MathWorks France*. adresse : https://fr.mathworks.com/help/stats/choose-a-classifier.html#bunt0ky (visité le 07/06/2022).

Résumé

La reconnaissance de l'activité humaine basée sur l'utilisation de différents types de capteurs (objets, ambiant ou portés) est un problème de classification de série chronologique. Le choix du type de modèle à utiliser ainsi que son architecture représente une étape cruciale.

Le travail proposé dans ce mémoire de fin d'étude repose sur l'étude entre différent type de classificateurs et d'un réseau de neurones récurent pour ainsi déduire les points forts de chaque méthode et conclure sur quelle architecture il vaudrait mieux s'orienter pour le thème de la reconnaissance d'activité humaine.

Ce travail compare donc plusieurs algorithmes de classification et un algorithme d'apprentissage automatique. La comparaison concerne les performances des algorithmes, leurs taux de prédictions et leurs durées d'apprentissage ainsi que les différentes technologies disponibles pour les implémenter.

Abstract

The recognition of human activity based on the use of different types of sensors (object, ambient or wearable) is a time series classification problem. The choice of the type of model to use as well as its architecture represents a crucial step.

The work proposed in this thesis is based on the study between different types of classifiers and a recurrent neural network to deduce the strengths of each method and conclude on which architecture it would be better to focus on the theme of human activity recognition.

This work compares several classification algorithms and a machine learning algorithm. The comparison concerns the performances of the algorithms, their prediction rates and their learning times as well as the different technologies available to implement them.