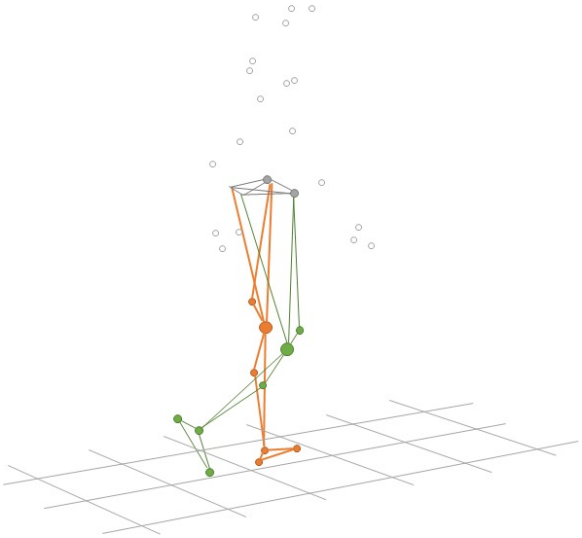
h e g

Haute école de gestion
Genève



SimGait - Classification de démarche pour patients ayant une pathologie de la marche

Barreiro Lindo Flávio
Gülen Lucas
Perritaz Alex

Rapport pour un projet de recherche dans le cadre de

Master en Sciences de l'information - Data Science

Superviseur : Professeur Alexandros Kalousis

15 janvier 2020

Remerciements

Le travail de recherche présenté dans ce mémoire a été proposé dans le cadre du Master en Sciences de l'information de la Haute école de gestion - Genève (HEG), sous la responsabilité de Monsieur Alexandros KALOUSIS.

Tout d'abord, nous tenons à remercier vivement Monsieur Alexandros KALOUSIS, responsable du Data Mining and Machine Learning group, de nous avoir chaleureusement accueilli dans son équipe, procuré tous les moyens nécessaires au bon déroulement du travail et de l'attention particulière qu'il y a porté.

Nous sommes particulièrement reconnaissants envers Pablo STRASSER et João Candido RAMOS de la Haute école de gestion - Genève ainsi que le DMML group dans son ensemble pour leurs précieux conseils qui nous ont permis de mener à bien notre travail sur la classification de démarche de patients ayant une pathologie de la marche.

Nous tenons également à remercier Monsieur Douglas TEODORO, professeur de la Haute école de gestion - Genève pour son encouragement et pour avoir donné un avis extérieur sur ce travail.

Finalement, nous tenons à remercier particulièrement Madame la professeure Magda Gregorová pour ses conseils précieux et ses enseignements tout au long de cette année de master très enrichissante.

Abstract

Ce travail est orienté vers le développement d'une solution capable de soutenir les médecins dans la prise de décision lors d'une intervention visant à corriger et à améliorer la condition des patients souffrant de pathologies de la marche.

Cette étude a pour but de démontrer l'efficacité des trois méthodes d'apprentissage automatique à savoir, les forêts d'arbres décisionnels (RF), les réseaux de neurones convolutifs (CNN) ainsi que les réseaux de neurones récurrents (RNN) dans la classification de pathologies de la marche.

Les résultats de notre base de référence à savoir les forêts d'arbres décisionnels, parviennent à classifier correctement les pathologies dans 46% des cas. La seconde approche, utilisant les réseaux de neurones récurrents spécialisés dans les séquences temporelles atteint une précision de 51%. La troisième approche quant à elle, utilise les réseaux de neurones convolutifs contribue à une classification à hauteur de 83% de précision pour l'identification des pathologies.

Keywords : Big Data, Gait Analysis, Analyse de la marche, Machine Learning, Apprentissage automatique, RNN, CNN, Random Forest

Décharge de responsabilité

Barreiro Lindo Flávio, Gülen Lucas et Perritaz Alex assument toutes les responsabilités concernant les déclarations et autorisations reportées dans ce projet de recherche. Monsieur Alexandros Kalousis, en sa qualité de superviseur de projet ne pourra en aucun cas être retenu responsable en cas de controverses résultant des cas décrits dans ce travail.

Table des matières

Liste des tableaux

Table des figures

Nomenclature

Symboles grecs

α	Taux de signification
χ	Khi carré
γ	Différence d'erreur entre un modèle A et un modèle B
∂	Dérivée partielle
σ	Sigmoïde - fonction d'activation
$\sum$	Somme des éléments
$\tanh$	Tangente hyperbolique - fonction d'activation
θ	Paramètre de l'équation

Symboles romains

A	Couche cachée du réseau de neurones récurrent
c	Contient l'historique des états cachés depuis un instant t
E	Correspond à l'erreur
F	Taille du filtre w de poids
f	Fonction
H	Hypothèse
h	Représentation de l'état caché d'un réseau de neurones récurrent
k	La prédiction d'un modèle
L	Perte
n	Nombre d'éléments dans l'échantillon
o	Prédiction faite par le LSTM
p	Niveau de signification observé
$soft$	Softmax

t	Instant dans le temps
u	Matrice de poids de la couche cachée
v	Matrice de poids entre chaque instant t
w	Matrice de poids de la couche de sortie
x	Instance ou sous partie d'une matrice d'instance
y	Représentation de la prédiction
z	Sortie d'une opération de convolution

1 Introduction

Instinctif au premier abord, se déplacer en alternance sur ses deux jambes cache derrière cette apparente simplicité, un processus complexe perfectionné par l'apprentissage. La difficulté résulte principalement de la complexité du mouvement qui implique coordination des muscles, des os et du système nerveux dans un processus cyclique.

1.1 Informations générales

Le projet SimGait est orienté vers le développement d'une solution capable de soutenir les médecins dans la prise de décision lors d'une intervention visant à corriger et à améliorer la condition des patients souffrant de pathologies de la marche.

L'objectif du projet est de développer, au travers de la simulation neuromécanique et du Machine Learning, des modèles précis de locomotion humaine. Ces modèles permettront aux chirurgiens de visualiser au préalable l'impact de l'intervention sur un simulateur et de faire les ajustements nécessaires en conséquence.

Pour établir un diagnostic sur la condition d'un patient, les médecins ont recours à deux sources d'information :

- Le premier provient directement des mesures prises sur les patients (données cliniques)
- Le second provient des capteurs et enregistrements vidéos relevés en laboratoire sur les patients

1.2 Etat de l'art

1.2.1 Démarche des patients

1.2.1.1 Contexte

Traduit de l'anglais *gait analysis*, l'analyse de la marche est l'étude méthodique de la locomotion, plus spécifiquement, l'étude du mouvement humain.

Considéré comme acquis pour la plupart, la marche est un moyen de locomotion naturel et consiste en un déplacement en appui alternatif sur les jambes.

Chez le nourrisson, l'apprentissage de la marche se fait après ses 10 premiers mois et marque une étape importante dans le développement de celui-ci. Sitôt l'apprentissage terminé, l'humain marche de manière instinctive et il ne prête attention au système locomoteur que lorsque survient une gêne ou une douleur[1].

Malgré cette apparente simplicité et les importantes avancées réalisées dans les domaines de la science et de la technologie, il reste des zones d'ombre dans la compréhension du fonctionnement de la démarche. La difficulté résulte principalement de la complexité du mouvement qui implique coordination des muscles, des os et du système nerveux dans un processus cyclique. Ces mécanismes réunis, permettent notamment de se tenir debout et de garder l'équilibre en conditions statiques et dynamiques [2, 3, 4]

1.2.1.2 Actuel

Dans cette section, nous allons aborder les points essentiels à la compréhension des travaux effectués dans le domaine de la marche afin de mieux en comprendre les mécanismes.

1.2.1.2.1 Cycle de marche

Le cycle de marche (Stride ou Gait Cycle, voir dans Figure 1.1) représente l'intervalle qui sépare le contact initial du talon jusqu'à l'occurrence suivante du même talon. Les deux principales étapes de ce cycle sont la phase d'appui (Stance) et la phase oscillante (Swing). Cependant un nombre supplémentaire d'étapes est à considérer si l'on veut pouvoir s'attaquer à des problèmes plus complexes[5, 6, 3].

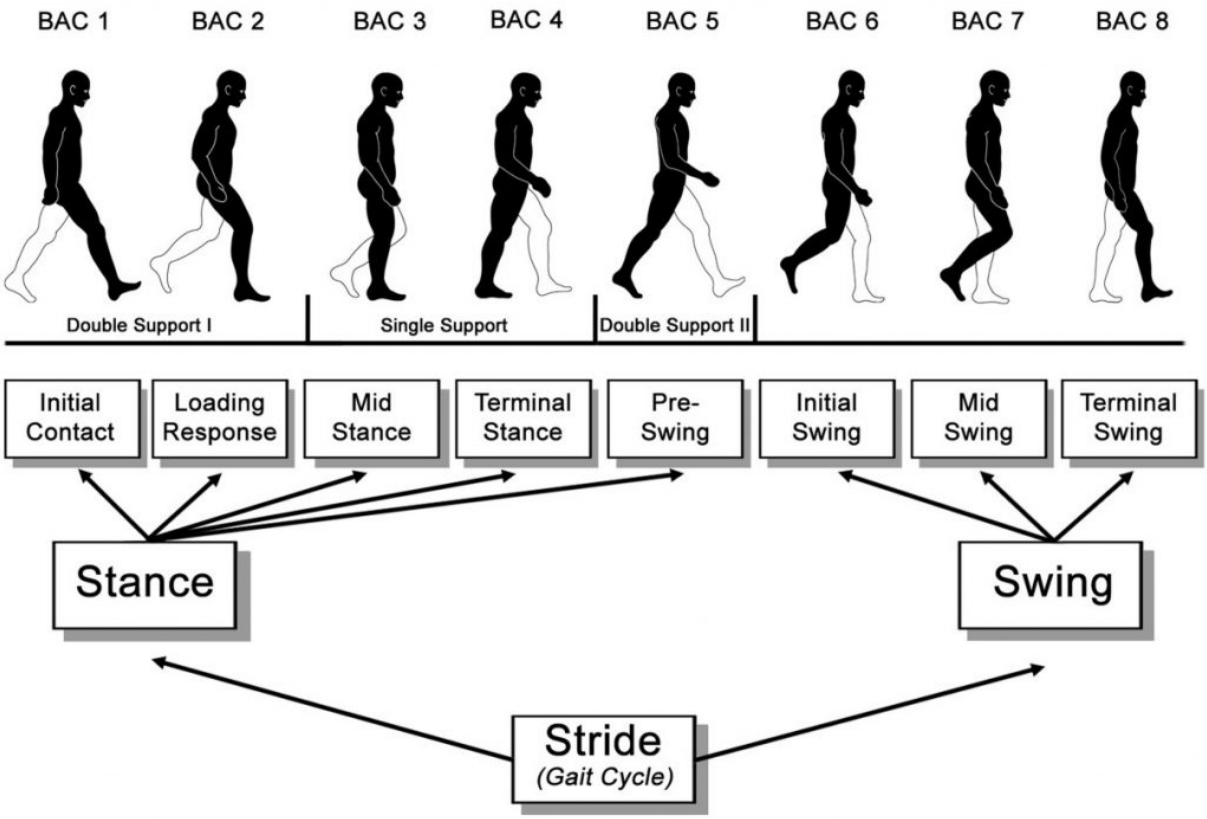


FIGURE 1.1 – Cycle de marche

Figure reprise de protokinetics [7]

L'étude de la démarche chez l'Homme est un défi à relever pour plusieurs domaines tels que la biomécanique, l'orthopédie, médecine physique et de réadaptation, kinésiologie, physiothérapie, et la médecine du sport[3].

1.2.1.2.2 Les différents systèmes de mesure et d'enregistrement des informations de la marche

Différents types de systèmes de détection existent pour enregistrer et mesurer les multiples informations en lien avec les différents paramètres de la démarche. Les différents systèmes sont divisibles en trois grandes catégories :

- Systèmes portables
- Traitement d'images
- Capteurs au sol

Ces différentes méthodologies sont abordées en-dessous détaillant les différentes technologies actuellement utilisées.

1.2.1.3 Systèmes portables

Un système portable, comme son nom l'indique, est composé de capteurs qui seront intégrés dans des équipements ou directement placés sur le patient dans le but d'apporter des données cliniquement pertinentes.

1.2.1.3.1 Senseurs de force et de pression

Les capteurs de force permettent de mesurer la force de réaction au sol (GRF) sous le pied et de retourner un courant ou un voltage proportionnel à la pression mesurée.

Les capteurs de pression quant à eux, permettent de mesurer la force appliquée au capteur sans prendre en compte les différentes composantes de ces forces sur chacune des axes.

Cette méthode se retrouve souvent dans les systèmes portables d'analyse de la démarche. Ils sont généralement incorporés dans les semelles des chaussures comme illustré dans la figure 1.2 [8, 9].

Des études ont démontré que les mesures effectuées grâce à ces semelles équipées de capteurs de forces produisaient des résultats similaires à ceux obtenus en laboratoire spécialisé dans le domaine de la démarche [10].

FIGURE 1.2 – Semelles ayant des capteurs de force

Figure reprise de Muro-de-la-Herran et al. [8]

1.2.1.3.2 Senseurs d'inertie

Les senseurs d'inertie sont des composants électroniques mesurant la vélocité, l'accélération, l'orientation, et la force gravitationnelle d'un objet grâce à une synergie entre accéléromètres et gyroscopes[8, 9].

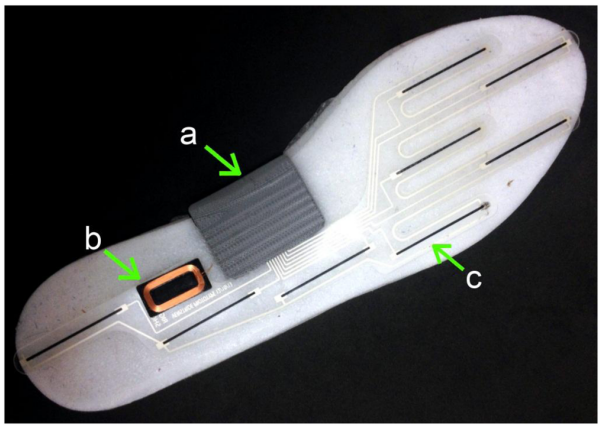


FIGURE 1.3 – Senseur d'inertie

Figure reprise de Muro-de-la-Herran et al. [8]

1.2.1.3.3 Goniomètres

Ce type de capteurs permettent d'étudier les angles des chevilles, des genoux, des hanches et des métatarses.

Les goniomètres fonctionnent grâce à la résistance engendrée lors de la torsion de celui-ci. Lorsque le matériau se plie, il s'étire, ce qui signifie que le courant qui le parcourt doit parcourir une plus longue distance. Par conséquent, lorsque le capteur se plie, sa résistance augmente proportionnellement à l'angle de torsion [8, 6, 9].

FIGURE 1.4 – Goniomètres

Figure reprise de biometricsltd [11]

1.2.1.3.4 Senseurs ultrasoniques

Les senseurs ultrasoniques sont des transducteurs permettant de convertir un signal physique en un autre. Ils sont utilisés pour mesurer des variables cinématiques comme la longueur d'un pas, la longueur d'une foulée, la distance séparant les deux pieds et la distance séparant le pied du sol.

Ces senseurs se basent sur la vitesse de déplacement du son dans l'air pour continuellement calculer les variations des distances entre deux points grâce à la mesure du temps entre l'émission et de la réception de l'onde [8, 9].

1.2.1.3.5 Électromyographie

L'électromyographie permet de mesurer la contraction volontaire ou non des muscles à l'aide d'électrodes (invasive ou non-invasive). Les signaux mesurés sont ensuite amplifiés et ajustés pour un usage clinique ou scientifique plus adapté [8, 9].

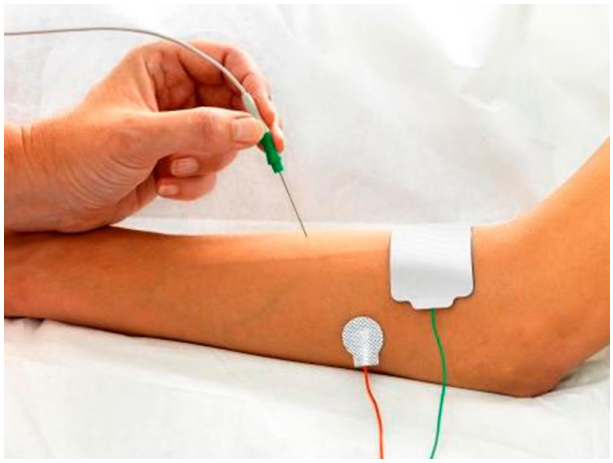


FIGURE 1.5 – Électromyographie

Figure reprise de Technomed [12]

1.2.1.4 Traitement d'images

La configuration typique d'un système de traitement par images est constituée d'une ou de plusieurs caméras (analogues ou digitales).

Parmi toutes les fonctions du traitement d'images, une technique en particulier se démarque particulièrement : la mesure de la profondeur. Dans cette section nous allons discuter plusieurs technologies permettant de cartographier la profondeur des points [8].

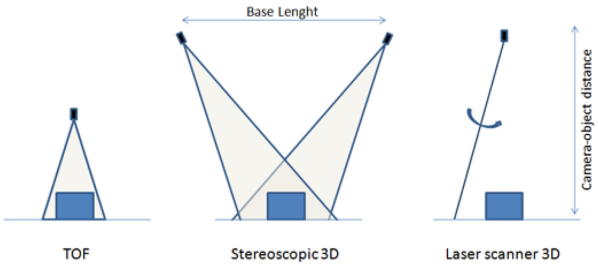


FIGURE 1.6 – Traitement d'images

Figure reprise de Muro-de-la-Herran et al. [8]

1.2.1.4.1 Stéréoscopie

Pour obtenir un effet de relief ou profondeur des éléments, la stéréoscopie se base principalement sur un système de caméra en stéréo. Les enregistrements sont ensuite analysés pour trouver une correspondance de points sur les différentes images.

Ce modèle se base sur le calcul de triangles similaires entre les capteurs optiques, la source de lumière et l'objet en question. Afin d'avoir un modèle représentatif, il est nécessaire d'avoir recours à plusieurs images [8].

1.2.1.4.2 Caméra temps de vol

Le fonctionnement de ce principe est similaire au principe utilisé par les senseurs ultrasoniques, mais au lieu d'employer des ultrasons, ici on emploi un flash de lumière. Les caméras TOF illuminent la scène et les objets et calculent le temps que ce flash lumineux prend pour effectuer le trajet entre l'objet et la caméra. La mesure du temps de vol est effectuée de manière indépendante pour chaque pixel de la caméra, permettant la modélisation de l'objet en 3-Dimensions [13, 14, 8].

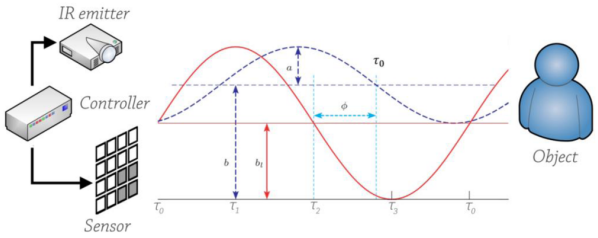


FIGURE 1.7 – Caméra temps de vol

Figure reprise de Muro-de-la-Herran et al. [8]

1.2.1.4.3 Lumière structurée

La technique de lumière structurée a fait partie des premières méthodes de numérisation 3D développées dans les années 80. Cette technique de détection de profondeur est aussi employée par des appareils destinés au grand public tels que la Kinect de Microsoft.

Le projecteur du motif structuré permet de projeter un motif dans la pièce ou sur la surface désirée, et la caméra permet d'analyser les déformations du motif (points, rayures, etc.) et de modéliser l'objet causant l'altération.

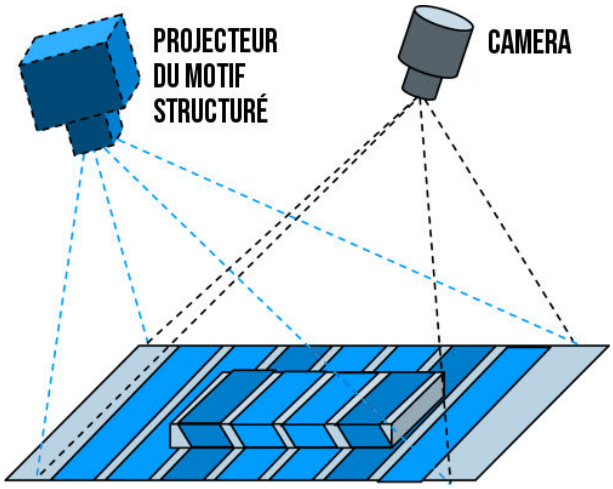


FIGURE 1.8 – Lumière structurée

Figure reprise de Valente [15]

1.2.1.4.4 Thermographie infrarouge

Ce procédé permet de créer des images en fonction des températures des surfaces. Notamment, la capacité à mesurer la thermographie du corps humain.

FIGURE 1.9 – Thermographie infrarouge

Figure reprise de Muro-de-la-Herran et al. [8]

1.2.1.5 Capteurs au sol

Ici comme son nom l'indique, il s'agit de capteurs placés au sol. Tout comme les senseurs de force et de pression dans la section des systèmes portables, on peut y retrouver deux types de capteurs au sol : les capteurs de forces et capteurs de pression [8].

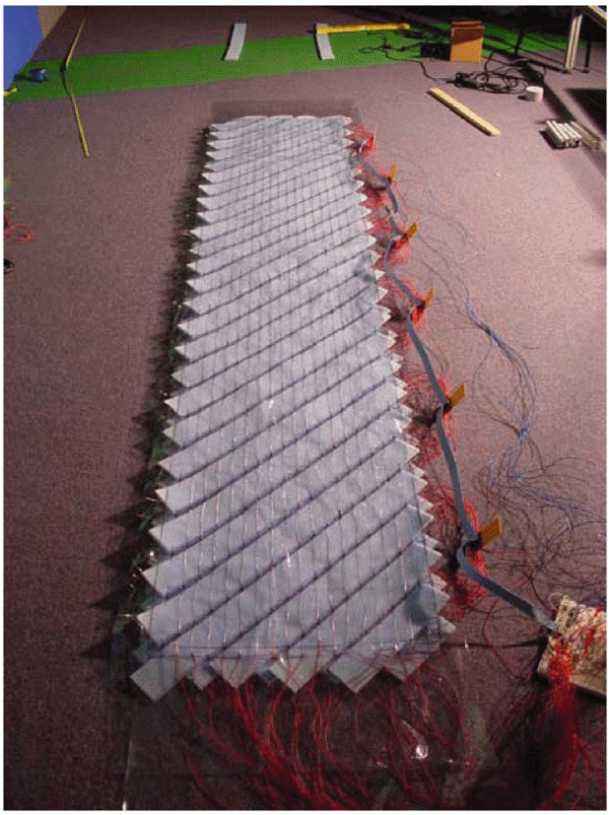


FIGURE 1.10 – Capteurs au sol

Figure reprise de Middleton et al. [16]

Non seulement le fonctionnement de ces deux méthodes sont semblables, mais les résultats obtenus aussi en témoignent[10].

1.2.2 Modèles statistiques appliqués

Dans cette section, nous allons tout d'abord expliquer les méthodes utilisées pour extraire / transformer des caractéristiques dans un projet d'analyse de la démarche et ensuite, détailler plusieurs types de cas qui peuvent être abordés dans un tel projet :

— Détection d'événements
— Identification de personnes
— Classification de différentes pathologies

1.2.2.1 Extraction / Transformation de caractéristiques

Certains chercheurs se sont penchés sur la possibilité de représenter les données d'un patient avec seulement certains attributs, par exemple, tout en préservant la variance totale. Appelé "Principal Component Analysis (PCA)", cet algorithme tente de trouver les composants orthogonaux qui capturent suffisamment la variation totale des données originelles. Ces composants sont simplement le résultat de combinaisons linéaires des attributs du patient. D'autre part, l'application du PCA peut être graphiquement interprété comme une projection des données dans un espace qui aura moins de dimensionalité que celui d'origine [17]. R. Shiavi et P. Griffin ont utilisé cet algorithme pour réduire la dimensionalité des données électromyographiques de plusieurs muscles sur 25 patients. Ceci dans le but de pouvoir appliquer une analyse par « Cluster » et de séparer les patients, selon les « patterns » identifiés, en plusieurs groupes hiérarchiques (deux groupes à chaque niveau) [18].

Par ailleurs, l'analyse de correspondance multiple (MCA) d'un patient est une méthode descriptive et graphique pour mesurer l'interdépendance entre les attributs de celui-ci. Le but de cet algorithme est similaire au PCA qui est de réduire la dimensionalité de la représentation des attributs. L'application du MCA sur une série temporelle de signaux électromagnétique de la démarche peut se résumer en 6 étapes [17] :

1. Rendre les signaux de démarche qui sont continus (nombre non défini de valeurs possibles) en variables discrètes (nombre fini de valeurs possibles).
2. Compiler les observations de ces catégories distinctes dans un tableau de contingence.
3. Calculer les totaux et de colonne de la table.
4. Trouver la meilleure projection en basse dimension des points de profil de manière à préserver au maximum les distances entre points.
5. Tracer et étiqueter les points de profil projetés dans le plan du facteur de petite dimension.
6. Identifier et interpréter qualitativement les associations

Loslever et al. ont utilisé cet algorithme pour faire une analyse de signaux multidimensionnels correspondant aux signaux électromagnétiques enregistrés lors de la démarche d'un patient. Cette première étape avait pour objectif de pouvoir repérer les tendances générales au changement dans les signaux multidimensionnels et les composantes les plus importantes des signaux. Dans un second temps, ils

ont utilisé ces informations pour sélectionner une composante du signal qui semblait être discriminante pour y appliquer une analyse hiérarchique par « Cluster » [19]. Voici un exemple graphique de la manière dont fonctionne une telle analyse :

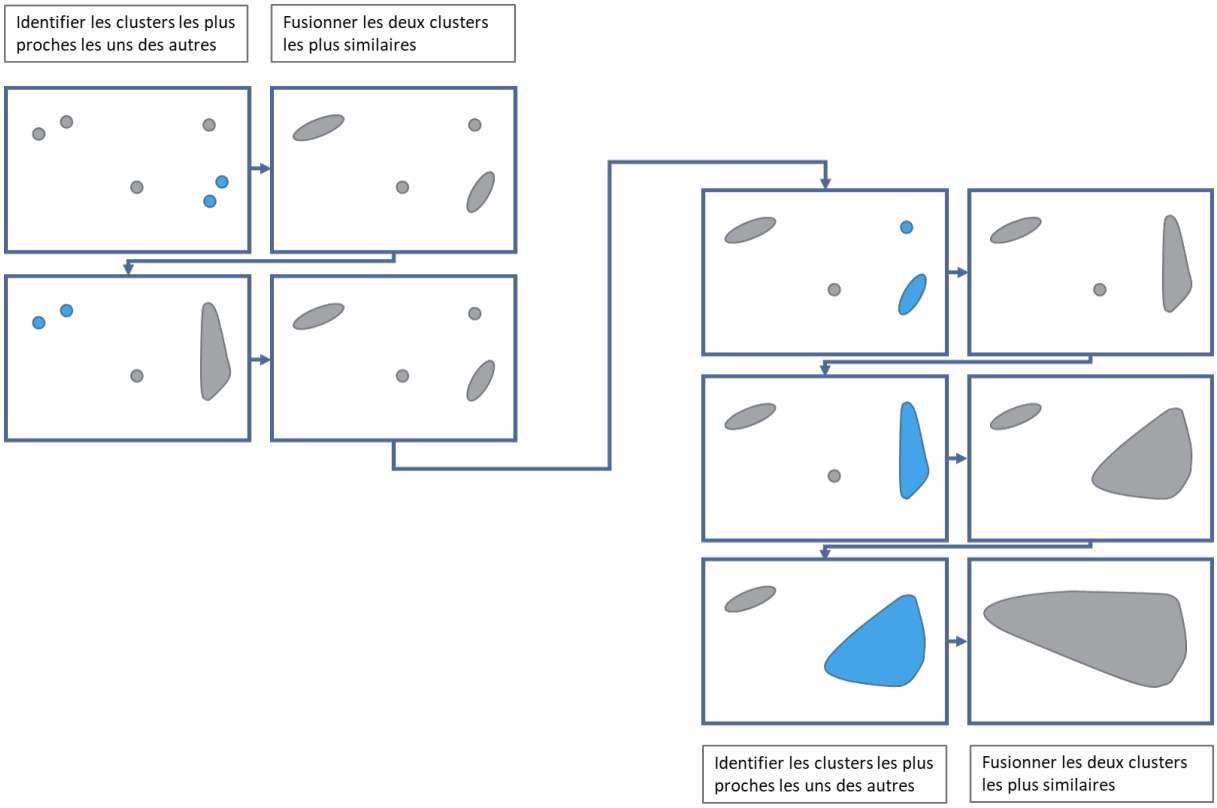


FIGURE 1.11 – Méthode de clustering utilisée

Figure adaptée de Bock [20]

Après avoir discuté des différentes méthodes employées pour collecter les données sur la démarche des patients, nous allons aborder les différents modèles et techniques utilisés pour les analyser. Le traitement de telles données qui sont représentées essentiellement sous forme

- cinématique (étude des mouvements indépendamment des causes qui les produisent),
- cinétique (relatif à la vitesse des mouvements),
- électromyographique (relatif au fonctionnement des muscles et des nerfs)
- et anthropométrique (mesure des particularités dimensionnelles d'un homme)

nécessite souvent l'utilisation de modèles statistiques afin de les exploiter de manière efficiente [17].

1.2.2.2 Détection d'événements

L'un des types importants de classification dans le domaine de la démarche est la détection d'événements. Les événements décrivent des parties d'un cycle de démarche et nous pouvons en identifier quatre dans l'ordre suivant :

1. Contact du talon droit

2. Lever du talon gauche

3. Contact du talon gauche

4. Lever du talon droit

L'objectif de la recherche dans ce domaine est de pouvoir déterminer à quel instant un événement précis a lieu, en se basant sur les données de capteurs.

Quelques chercheurs ont pu obtenir des résultats satisfaisants en utilisant des dispositifs permettant de détecter les événements de la démarche. Ces méthodes passent par des tapis roulants munis de détecteurs de force [21], par des plaques de pression placées de façon stratégique [22], des semelles connectées [23] et même des chaussettes connectées [24].

Cependant, selon [25], ces méthodes, bien que fiables, ne permettent pas de résoudre certains problèmes de classification, tels que le contact des talons ou le lever des orteils. De plus, Salim Ghoussayni et al. [25] nous disent que la précision de la prédiction de ces événements est essentielle dans la classification de CP (Cerebral Palsy) où il est possible de comparer une marche d'un d'individu atteint d'une maladie d'une marche normale.

En 2019, Kidzinski et al. ont utilisé des réseaux de neurones artificiels pour détecter les évènements décrits ci-dessus. Pour cela, ils ont utlisé un « Long-short-Term Memory » (LSTM) qui est un type de réseau de neurones artificiels utilisé pour sa flexibilité quant à la taille des vecteurs de données en entrée et de sa capacité à traiter des séries temporelles. Leur modèle a été appliqué sur un jeu de données comprenant 18'184 observations sur la démarche de patients entre 4 et 19 ans entre 1994 et 2017 [26].

Ci-dessous, une représentation de leur LSTM avec, sur la première partie de cette figure, trois séries temporelles d'un même patient lors de sa marche et qui constituent les données enregistrées par les capteurs sensoriels. Ces données sont injectées dans le LSTM représenté par la partie en vert. D'autre part, Kidzinski et son équipe ont réussi à atteindre des résultats d'environ 95-99 % de précision dans les prédictions d'évènements selon la pathologie du patient [26].

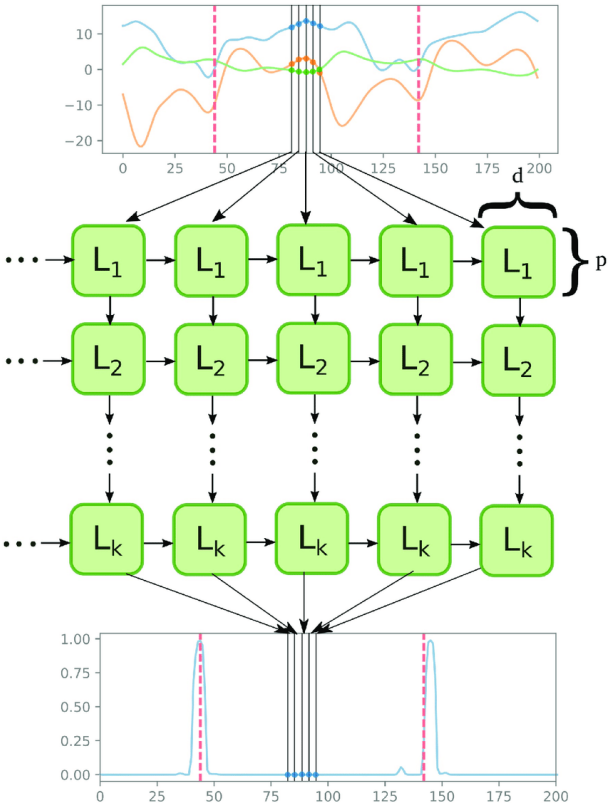


FIGURE 1.12 – LSTM utilisé par Lukz Kidzinski

Figure reprise de Kidzinski et al. [26]

1.2.2.3 Identification de personnes

Une autre utilisation des caractéristiques de la marche est la classification de personnes. C'est-à-dire, savoir quel individu est présent en se basant sur sa démarche. Pour atteindre cet objectif, il y a deux approches possibles : Analyse basée sur des modèles ou des analyses orientées apparence.

Les analyses basées sur des modèles utilisent des caractéristiques comme la taille des pas, la vitesse de la marche et les angles des membres. Ces méthodes ont été utilisées à des fins de surveillance automatisée [27, 28, 29, 30] et qui ont réussi à obtenir des résultats satisfaisants mais sous des conditions idéales et utilisant un bon angle de vue. Les modèles ne peuvent pas être généralisés à d'autres angles de vue car leur problème majeur est le fait que les bras et les jambes se croisent lors d'un cycle de marche pendant la phase d'extraction de caractéristiques des vidéos de surveillance (Figure 1.13). BenAbdelkader et al. ont créé une méthode permettant d'identifier à partir d'images de surveillance. Cette méthode est robuste aux changements de lumière, d'habits, et de point de vue et ayant une précision d'environ 40% [31].

D'autres techniques ont été utilisées, notamment, la classification de personnes basées sur l'acoustique de leur marche, c'est-à-dire, les bruits qu'une personne fait en marchant [32]. Les auteurs de ce papier se sont basés sur des modèles de Markov pour classifier les personnes et ont pu atteindre une précision de 65%.

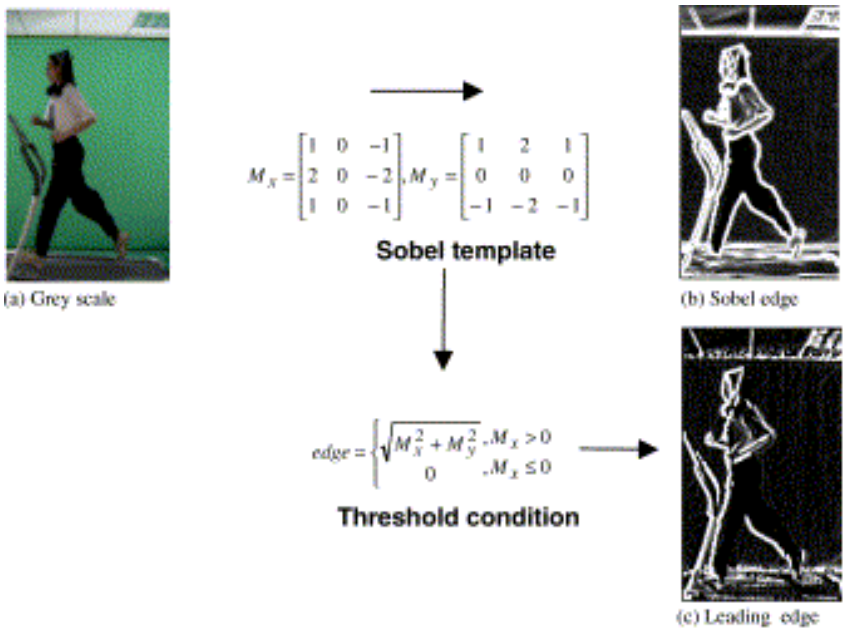


FIGURE 1.13 – Cycle de marche

Figure reprise de Geiger et al. [32]

D'autre part, en 2016 Yang Feng et al. ont publié un article dans lequel ils ont utilisé un LSTM pour effectuer de la classification de personnes et ont réussi à obtenir une précision variant de 83% à 40% selon l'angle des prises de vues [33]. Un autre algorithme, kNN, a été utilisé dans l'article scientifique de Marcin Derkatka et Mariusz Bogdan en 2015. Les résultats de leur modèle a atteint plus de 97% de précision basés sur des mesures de plus de 200 personnes (3500 cycles de démarche) [34].

Les avancées dans ce domaine sont très importantes, car selon Vijay Bhaskar Semwal et al., il est possible de comparer la démarche d'une personne à une empreinte digitale. Dans le sens où chaque individu a aussi bien une empreinte digitale unique qu'une démarche propre [35]. Ceci nous permet de nous rendre compte du degré de variabilité d'une démarche d'un patient à un autre et de la complexité dans la reconnaissance de patterns entre les différentes démarches. Leur travail a consisté à comparer un KNN avec un ANN dans l'identification de personne selon leur démarche. KNN obtient des résultats assez proches de l'ANN et variant de 100% à 50% de précision selon la complexité.

1.2.2.4 Classification de différentes pathologies

En 2016, Wei Zeng et al. ont utilisé un modèle de classification de 93 patients étant atteints de la maladie de Parkinson et 73 autres étant en bonne santé. Ce jeu de données contient des mesures de sévérité de la maladie en plus des données de 8 capteurs déposés sur différentes parties du corps des patients. Parmi ces données, ils ont extrait et utilisé un vecteur de 4 dimensions contenant la différence entre deux capteurs placés sur la jambe gauche et droite et la somme de deux autres placés également sur chacune des jambes [36].

Ils ont ensuite entraîné un « Radial Basis Function Neural network » (RBF), un modèle utilisé pour les

données temporelles comme c'est le cas dans un cycle de démarche d'un patient[37], avec les données sélectionnées dans l'étape précédente. Après avoir entraîné leur modèle, ils ont utilisé le principe de testé leur modèle avec de la « Cross-validation »[38] et ils ont réussi à atteindre une précision dans leurs prédictions d'environ 92-94

Holzreiter et Köhle ont, par exemple, tenté de catégoriser les patients selon leur pathologie à l'aide d'une plaque de pression posée au sol et en analysant ces données avec un ANN. Ils ont analysé deux pressions successives lors de la démarche normale de 131 patients atteints de fracture du calcanéus et d'insuffisance des membres. Ils ont utilisé l'algorithme appelé « Fast Fourrier transform », qui sert à transformer des données discrètes du domaine temporel dans le domaine fréquentiel [39], comme données d'entrée de leur ANN. Avec leur réseau de neurones artificiels, Holzreiter et Köhle ont réussi à obtenir une précision de 95% dans la classification des personnes atteintes d'une pathologie et celles étant en bonne santé [40].

Quelques temps après, c'est Barton et Lees qui tentent de construire un ANN avec un objectif similaire à Holzreiter et Köhle, mais cette fois en tentant de classifier les patients en trois catégories : « healthy foot », « pes cavus », « hallux valgus ». Leur jeu de données était constitué de données sur la marche de 18 patients enregistrées à l'aide de plaques de pression. La précision des prédictions communiquée par Barton et Lees variait de 77 à 100%, dépendant des patients utilisé pour l'entraînement et le test et basé sur le petit volume de patients qu'ils avaient à leur disposition [41].

Prenons l'exemple de certains travaux effectués entre les années 1990 et 2000 pour contextualiser la problématique avec, par exemple, O'Malley et al. qui ont appliqué la méthode appelée “Fuzzy c-means clustering" dans les années en 1997 [42]. Cet algorithme de clustering tente de regrouper des patients selon des variables concernant leur démarche pendant sa phase d'entraînement. "Fuzzy c-means clustering" peut être vu comme un algorithme de classification car chaque instance se voit attribuée une probabilité d'appartenir à chacune des classes. Dans le travail d'O'Malley et al., ce modèle avait pour but de pouvoir catégoriser un patient comme ayant une démarche neurologiquement intacte ou ayant une infirmité motrice cérébrale “cerebral palsy”. Ainsi, cela leur permettait d'avoir une méthode de mesure objective à utiliser sur des données de test avant et après opération d'un enfant atteint d'une “cerebral palsy”[43].

1.2.3 Interpretabilité

L'interprétabilité est un élément crucial dans notre projet de recherche. En effet, dans le domaine médical, un médecin doit pouvoir expliquer pourquoi un certain diagnostic a eu lieu [44, 45, 46]. Or, les modèles statistiques complexes possèdent trop de couches d'abstraction et rendent leur compréhension très difficile.

Aussi, un humain doit pouvoir comprendre pourquoi une machine prend une décision. Il doit également pouvoir lui faire confiance [47]. Par exemple, un accident (Three Mile Island accident) est survenu dans une centrale nucléaire où un système a conseillé à un opérateur d'éteindre une des machines. Puisqu'il ne l'a pas fait, les conséquences ont été tragiques [48]. Même la DARPA a publié un article dans

lequel est relevé l'importance de pouvoir expliquer comment une Intelligence Artificielle marche : « XAI (Explainable AI) a pour but de de produire des modèles statistiques explicables, tout en gardant un haut niveau de performance [...] permettant aux humains de comprendre, faire confiance et gérer la génération émergent de partenaires dotés d'intelligence artificielle » [49].

Aussi, les data-scientists peuvent compter sur la confiance des médecins après qu'un modèle statistique soit en production depuis un temps qui n'est pas clairement défini, mais qui permet de faire assez de prédictions correctes [50]. Ceci est problématique car un modèle statistique peut ne pas être correct et faire beaucoup de mauvaises prédictions avant que les utilisateurs s'en rendent compte. Certains aspects psychologiques ont été utilisés dans la construction de modèles statistiques. Notamment, la notion de contrôle. Un papier scientifique relate d'un modèle qui a été créé et qui devait être paramétré en fonction de chaque patient, et ce, par le médecin [51]. D'autres essais ont également eu lieu comme la simple utilisation de conditions IF-THEN-ELSE [52, 53]. Ceci permet la compréhension facile par la part des professionnels de la santé. En revanche, leur précision est impactée aux dépens de l'interprétabilité.

if hemiplegia **and** age > 60 **then** *stroke risk* 58.9% (53.8%–63.8%)
else if cerebrovascular disorder **then** *stroke risk* 47.8% (44.8%–50.7%)
else if transient ischaemic attack **then** *stroke risk* 23.8% (19.5%–28.4%)
else if occlusion and stenosis of carotid artery without infarction **then** *stroke risk* 15.8% (12.2%–19.6%)
else if altered state of consciousness **and** age > 60 **then** *stroke risk* 16.0% (12.2%–20.2%)
else if age ≤ 70 **then** *stroke risk* 4.6% (3.9%–5.4%)
else *stroke risk* 8.7% (7.9%–9.6%)

FIGURE 1.14 – Example de modèle utilisant des conditions simples

Figure reprise de Letham et al. [53]

Nous pouvons constater grâce à la Figure 1 que quelques conditions booléennes permettent de déterminer la probabilité qu'une personne survive à la catastrophe du Titanic. Malgré la facilité de compréhension et que ce modèle ait été construit grâce à des données, certains modèles sont construits grâce à l'expertise des médecins qui doivent donner les règles à utiliser [52].

Ces difficultés ont donné origine à plusieurs écoles de pensée. La première consiste à affirmer que l'interprétabilité prime sur la précision d'un algorithme. La deuxième essaie de créer un modèle statistique performant et essaie de le rendre compréhensible. Et finalement, le plus utilisé actuellement, s'appelle model-agnostic. Aujourd'hui, l'objectif est d'essayer de créer un algorithme capable d'expliquer n'importe quel modèle, indépendamment de sa structure interne.

Selon Freitas Alex, le modèle Decision Tree est un des meilleurs modèles car il permet de créer un modèle précis et facile à expliquer [54]. En effet, même un professionnel n'ayant pas de bases solides en informatique ou algorithmique, peut comprendre les règles d'un nœud. Il est tellement important, qu'en 1996 une équipe de chercheurs a créé une méthode capable d'avoir des précisions inférieures, d'environ 2%, en se basant sur les prédictions d'un perceptron multicouche. L'idée était d'entraîner un

MLP sans se soucier de sa complexité et d'ensuite générer un modèle plus simple mais compréhensible [55]. Les chercheurs y sont parvenus en requêtant le réseau de neurones artificiels, en analysant la prédiction et en adaptant cette réponse à un Decision Tree.

Plusieurs études ont été menées pour ce modèle statistique. Freitas a démontré que les médecins préfèrent avoir des modèles plus complexes que des modèles simples[54]. L'être-humain n'aime pas les solutions trop simplistes pour des problèmes difficiles. Ainsi, il est conseillé d'avoir un arbre assez complexe pour résoudre un problème donné, mais en pouvant être analysé dans un temps convenable. D'autres modèles sont également facilement compréhensibles : kNN et régression linéaire. Dans le cas de k-Nearest Neighbors, l'algorithme peut être expliqué facilement avec des schémas et les diagrammes. Les difficultés se présentent lorsque le nombre de dimensions devient trop élevé pour être compris conceptuellement par un humain. Les mêmes avantages et inconvénients se présentent pour une régression linéaire. Ce dernier possède encore le désavantage d'imposer une compréhension mathématique par son utilisateur[54].

De plus, des méthodes d'apprentissage automatique appelés « Ensembles » permettent souvent d'avoir une meilleure prédiction que des algorithmes simples [56]. Par exemple, un Random Forest combine les multiples prédictions de plusieurs Decision Trees, ce qui rend la compréhension de ceci plus difficile[54].

Concernant la dernière méthode d'amélioration d'interprétabilité, l'idée est d'imaginer une black-box qui contient le modèle statistique. Ensuite, selon ses prédictions, il est possible d'expliquer le pourquoi en utilisant des schémas graphiques ou textuels.

Cette méthode possède l'avantage de permettre au créateur d'un modèle statistique de se concentrer uniquement sur la précision et la performance de celui-ci. Cela offre plus de flexibilité au modèle qui peut s'adapter à des cas plus complexes.

Des techniques ont été proposées pour combler à ce problème mais qui doivent être adaptées en fonction du modèle statistique de classification utilisé[57]. En revanche, seuls des modèles, certes complexes, mais moins qu'un réseau de neurones artificiels profond, ont été utilisés dans ce papier scientifique.

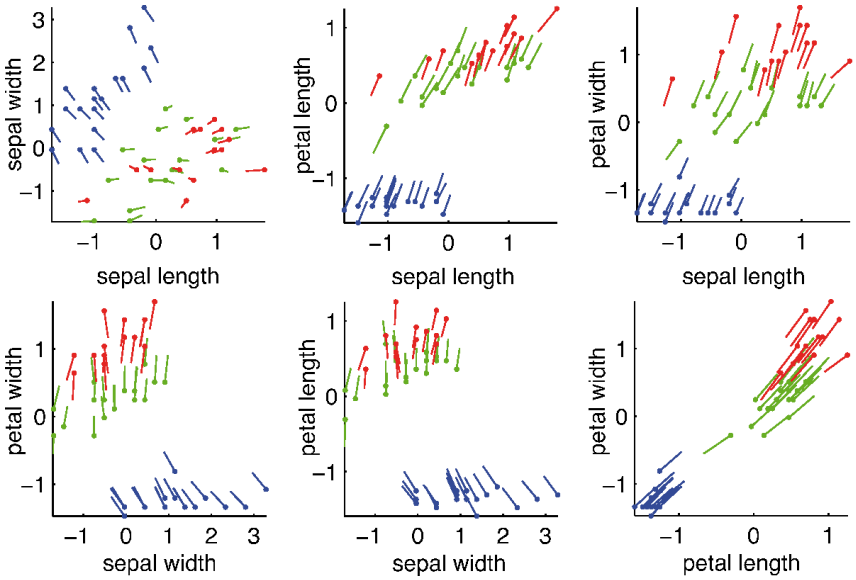


FIGURE 1.15 – Explication de pourquoi certaines classifications du jeu de données Iris se font avec kNNs

Figure reprise de Baehrens et al. [57]

Nous pouvons constater grâce à la Figure 1.15 que la largeur (width) et la longueur (length) des pétales jouent un rôle déterminant pour distinguer les fleurs entre Iris setosa (bleu) et, Iris virginica et versicolor (rouge et vert, respectivement). Cette façon de faire devient impraticable lorsqu'il s'agit de dizaines de dimensions.

Pour contrer cela, une solution a été proposée par Ribeiro et al. LIME (Local Interpretable Model-agnostic Explanations) est cette solution qui peut aider l'utilisateur à comprendre pourquoi une certaine prédiction a eu lieu.

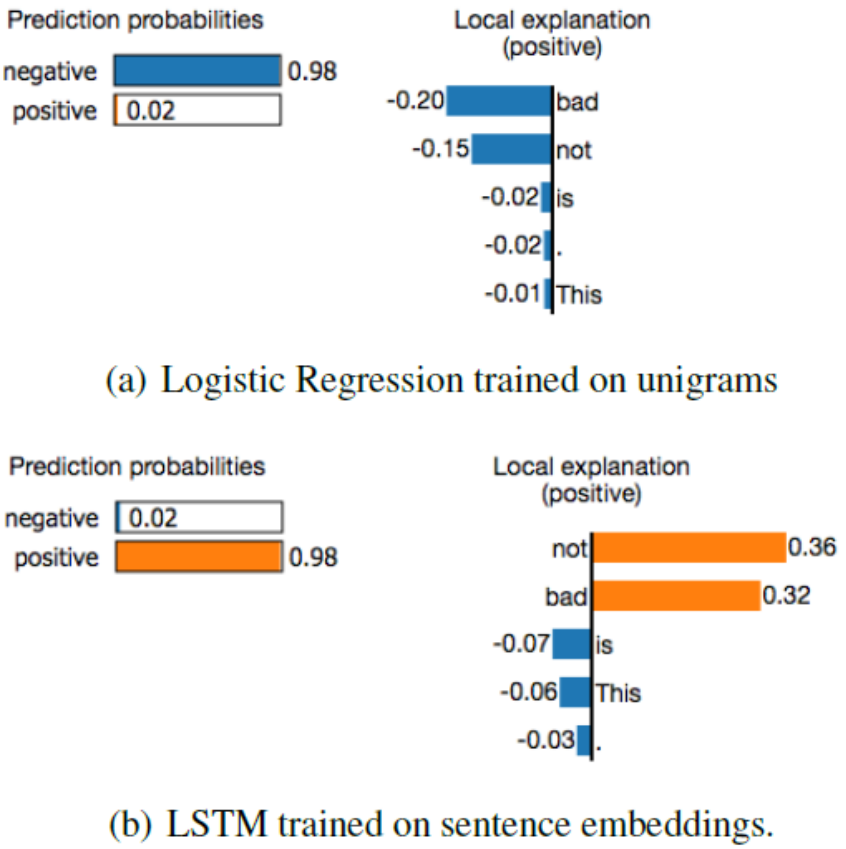


FIGURE 1.16 – Comparaison d'explication d'une prédiction entre une régression linéaire et un LSTM

Figure reprise de Ribeiro et al. [58]

La Figure 1.16 nous montre comment cette explication peut aider un data-scientist à améliorer son modèle statistique. Ici, en utilisant de la régression linéaire, la prédiction nous indique que bad et not ont un impact dans la prédiction qui nous indique que la phrase possède une connotation négative. Cependant, en utilisant un LSTM, le modèle est capable de comprendre que la combination des deux mots rend la phrase positive. De plus, This, is et . ont un impact négligeable par rapport au poids donnés aux autres mots.

Cette solution possède deux qualités : Elle peut être facilement explicable à un professionnel de la santé et peut aider un data-scientist à créer un modèle statistique plus robuste. Par exemple, Kaufman et al ont démontré à quel point un identifiant lié à un patient peut impacter la prédiction finale [59]. Ceci peut être dû à un oubli de suppression de cette « feature ». En utilisant LIME, le créateur de l'algorithme pourrait facilement détecter que l'identifiant ressortirait en tant que l'élément ayant le plus d'influence.

Dans un autre papier scientifique, Ribeiro et al ont démontré comment cet algorithme pourrait également être appliqué à un CNN.

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

FIGURE 1.17 – Comment utiliser LIME pour expliquer la prédiction d'un CNN

Figure reprise de Ribeiro et al. [47]

Nous pouvons voir ci-dessus (Figure 1.17) qu'un humain peut facilement analyser ce qui a impacté les prédictions. Malgré la mauvaise prédiction dans la Figure 1.17, il est possible de comprendre que le réseau de neurones artificiels ait commis une erreur.

En 2016, une autre méthode à été proposée par Krause et al. Celle-ci est basée sur un principe simple : modifier les entrées dans le modèle statistique et analyser les changements en sortie.

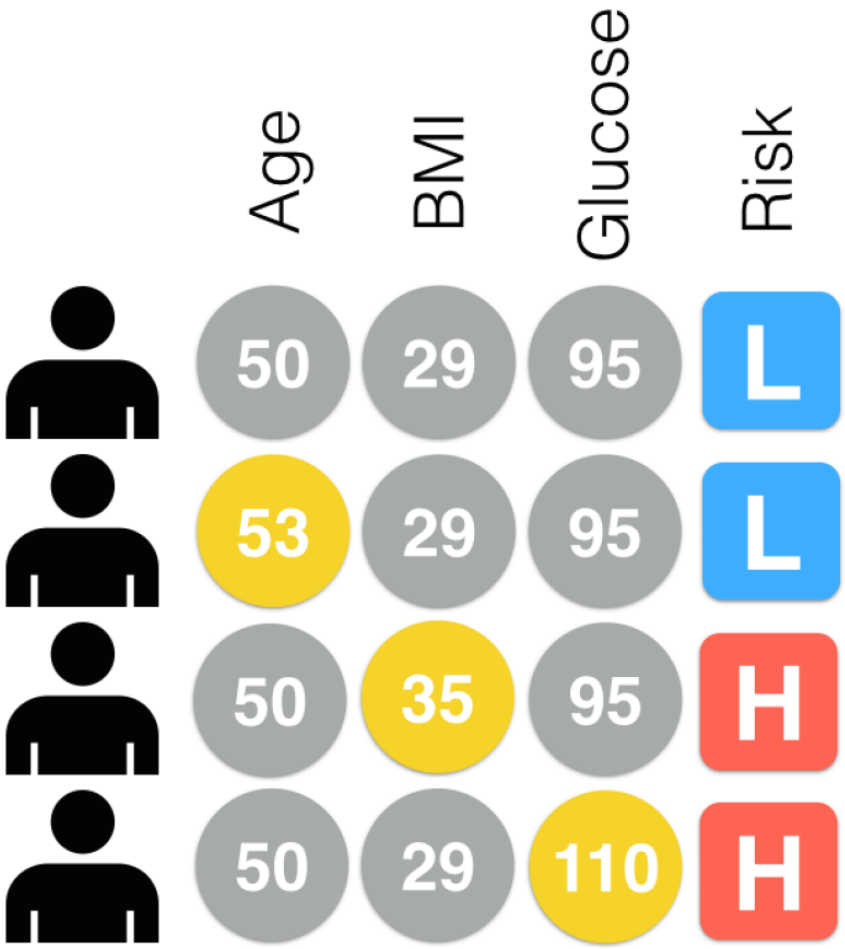


FIGURE 1.18 – Comment une modification des entrées affectent la sortie d'un modèle statistique

Figure reprise de Krause et al. [56]

Dans ce papier, les auteurs cherchent à pouvoir expliquer quelles caractéristiques permettent de déterminer si un patient possède un risque élevé de développer le diabète. Par exemple, nous pouvons voir ci-dessus que lorsque l'âge d'un patient est modifié, son risque n'est pas modifié. Nous pouvons ainsi partir du principe que cet attribut possède peu d'influence sur la prédiction. En revanche, le BMI et les glucoses sont déjà plus importants.

Ce papier scientifique propose également une interface graphique permettant à l'utilisateur de changer manuellement les valeurs d'une instance.

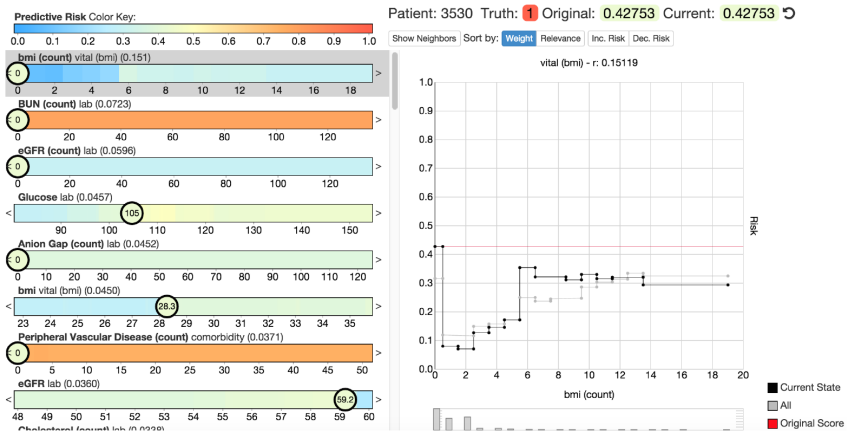


FIGURE 1.19 – UI Prospector permettant d'interagir avec une seule instance d'un patient

Figure reprise de Krause et al. [56]

Malgré l'interface graphique qui se présente de manière "user-friendly", nous pouvons imaginer que cela devient ingérable lorsqu'une instance possède plusieurs centaines de dimensions. De plus, ce ne serait pas une solution applicable dans le cadre d'une analyse d'image.

Une dernière technique sera ici décrite qui correspond de manière directe à ce que notre groupe veut réaliser : déterminer quelle partie d'un cycle de démarche permet de déterminer la prédiction d'un modèle statistique.

Horst et al. proposent une méthode d'analyse sans se préoccuper du type de réseau de neurones artificiels utilisé.

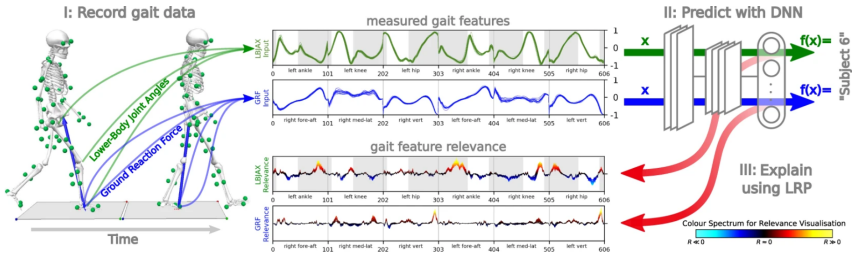


FIGURE 1.20 – Analyse de la marche d'un patient

Figure reprise de Horst et al. [60]

Le but démontré dans ce papier diffère par rapport à ce que nous voulons de faire lors de ce projet de recherche. En effet, notre objectif est d'aider les médecins dans la prise de décision pour déterminer si un patient possède une pathologie et de pouvoir lui expliquer pourquoi. Or, dans ce papier l'objectif est de déterminer quel patient a été donné en entrée. De plus, seules des personnes ne possédant pas de pathologies ont participé à la population du jeu de données. Malgré cette différence d'objectif, nous pensons que cette technique pourrait être appliquée à notre cas précis.

1.3 Analyse descriptive

Les données avec lesquelles nous avons travaillé nous ont été fournies par les Hôpitaux Universitaires de Genève. Elles comprennent des informations relatives au cycle de marche de 198 patients grâce à un ensemble de 33 capteurs soigneusement placés sur leur corps.

Chacun des capteurs permettent de recueillir des informations sur leur situation dans l'espace sur 3 axes et par la suite, de calculer les angles articulaires à partir de ces positions. Ces informations nous permettent de détecter les évènements majeures du cycle de marche, définis entre deux appuis du talon d'une même jambe sur le sol comme illustré dans l'image ci-dessous.

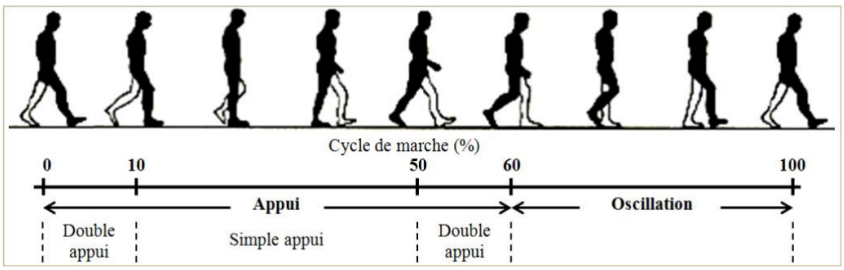


FIGURE 1.21 – Cycle de marche

Figure reprise de Viel [61]

Il est important de noter la dimension manuelle dans le protocole de placement des capteurs. Elle introduit de la variabilité dans les mesures de ceux-ci, en plus de la variabilité inhérente à la démarche de chaque individu, ajoutant du bruit dans les données.

2 Les données à disposition

2.1 La distribution de celles-ci

L'analyse des données est une étape importante dans bien des domaines, permettant une meilleure compréhension et utilisation des données. En comprendre la distribution permet notamment d'avoir un meilleure connaissance de l'environnement ainsi que des relations entre les différentes attributs.

Parmi les 1782 fichiers à notre disposition, seuls 1262 sont retenus. Comme les 520 autres fichiers contiennent plus de 15% de valeurs vides, nous avons pris la décision de les retirer du jeu de données. Ces 1262 fichiers comportent des informations relatives à 169 patients, dont 160 avec une pathologie et uniquement 9 en bonne santé.

Les six différentes pathologies présentes dans le jeu de données sont les suivantes :

CP_Ataxic
Paralysie cérébrale ataxique, causant trouble de la coordination des mouvements volontaires

CP_Spastic_Bi_Diplegia
Paralysie cérébrale spastique, causant contraction musculaire involontaire intense et passagère sur les deux jambes

CP_Spastic_Uni_Hemiplegia
Paralysie cérébrale spastique, causant contraction musculaire involontaire intense et passagère sur une des deux jambes

Healthy
Sain, la démarche du patient est considérée normale

Idiopathic_Toe_Walker
Marche sur la pointe des pieds idiopathique, chez les enfants ayant trois ans révolus persistant avec une marche pointe, au lieu d'avoir adopté une marche talon-pointe

Polytraumatisme

Présent chez une personne ayant subi plusieurs traumatismes (plaie, fracture, brûlure...) dont au moins un met en danger les fonctions vitales

Dans ce jeu de données, une instance représente la visite d'un patient et celles-ci sont regroupées par pathologie dans la figure 2.1 ci-dessous.

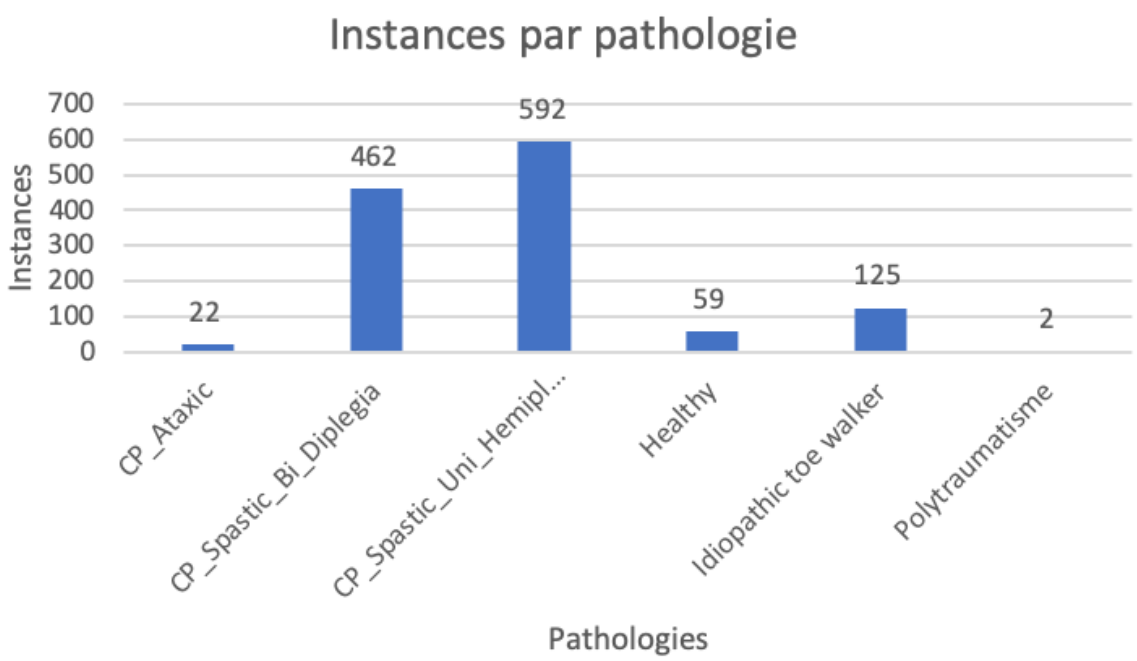


FIGURE 2.1 – Patients par pathologie

Cependant, le nombre de visites par patient n'étant pas limité, le graphique ci-dessus (Figure 2.1) n'est pas représentatif de la distribution de pathologies par patients. Effectivement, lorsque nous visualisons le nombre de visites par patient dans la Figure 2.2 , nous pouvons constater que la participation de la plupart des patients est de deux ou plus.

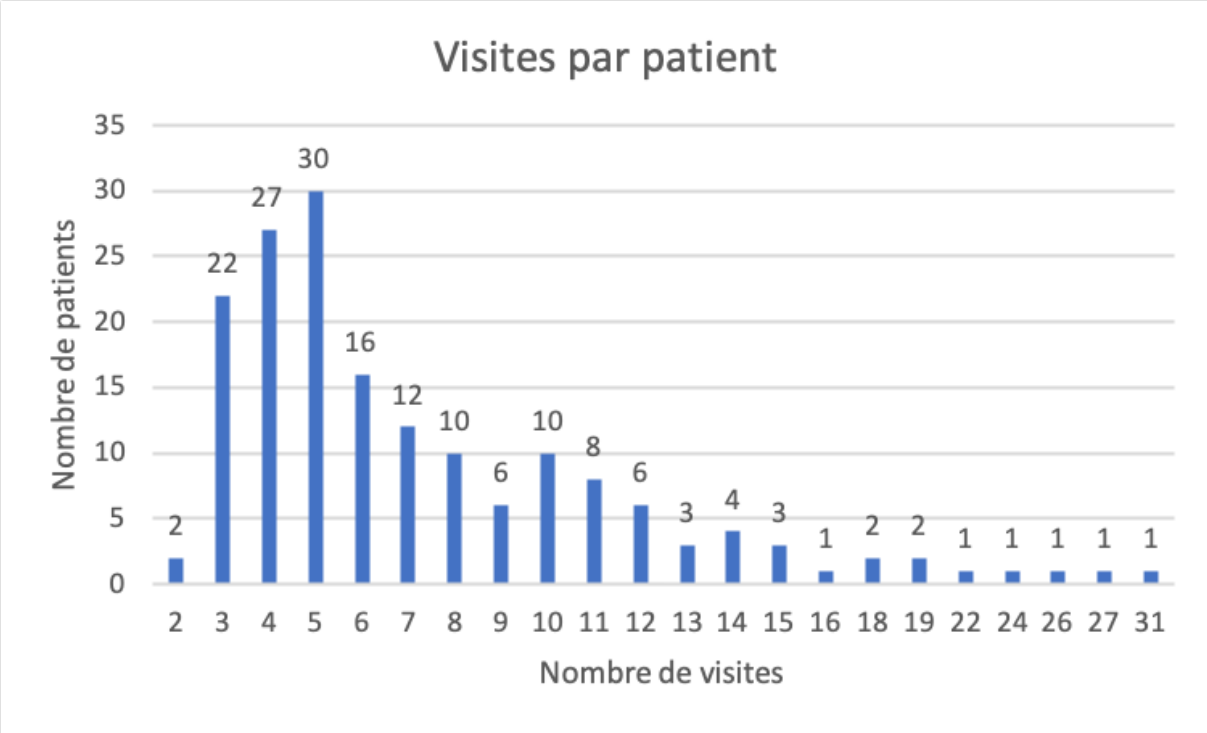


FIGURE 2.2 – Visites par patient

Nous représentons donc le nombre de patients par pathologie dans le diagramme en bâtons ci-dessous (Figure 2.3) et pouvons constater une légère différence dans la répartition des pathologies de notre jeu de données.

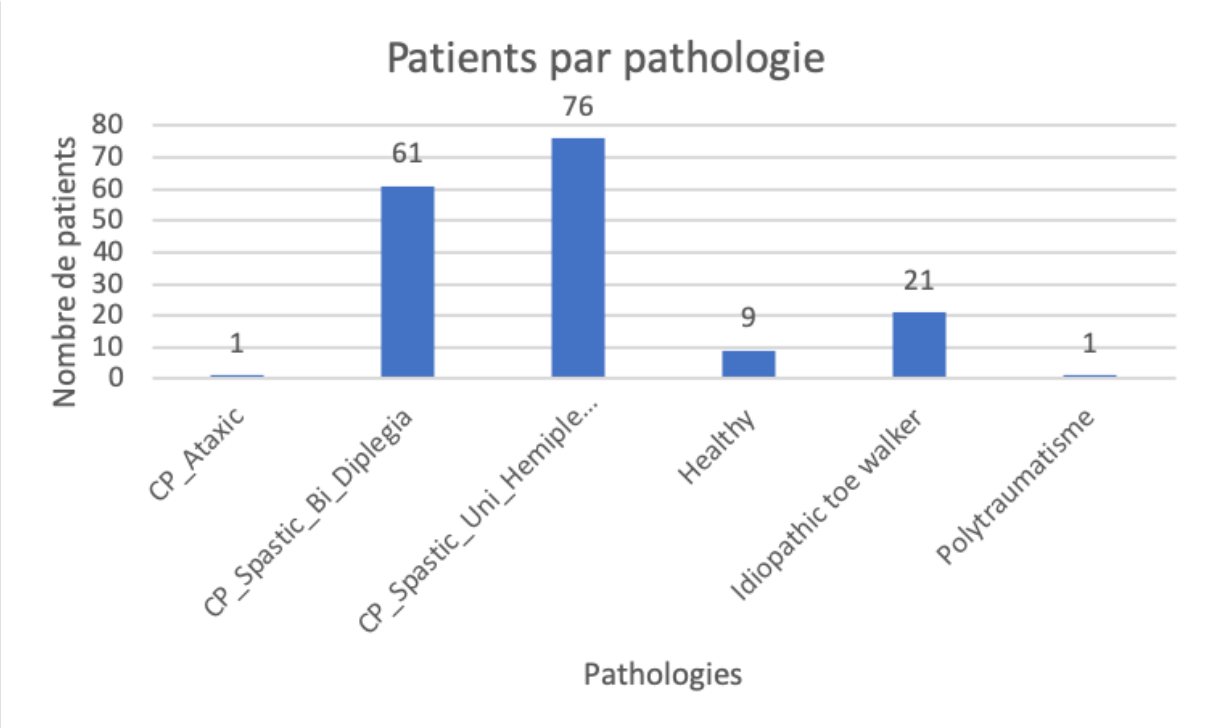


FIGURE 2.3 – Patients par pathologie

Lorsque les patients se rendent dans les laboratoires pour effectuer les tests, ils génèrent des données à partir des capteurs placés sur eux. Chacun des capteurs relève des informations par rapport à leur situation dans l'espace sur trois axes, ainsi que l'angle calculée par rapport à la position de ceux-ci. Les données brutes sont récupérées sous la forme suivante :

LTOE			RTOE			LKneeAngles			LAnkleAngles		
mm	mm	mm	mm	mm	mm	deg	deg	deg	deg	deg	deg
X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
2295.4	174.424	30.6894	1999.65	288.851	43.75	2.42928	0.828603	-5.67456	1.48095	-0.762629	6.76903
2294.96	174.785	30.4031	1964.49	284.646	47.3754	2.01541	0.62376	-5.44807	1.66162	-0.709147	6.30821
2294.57	175.1	30.173	1931.76	280.811	51.0691	1.77076	0.430467	-5.18286	1.91012	-0.661115	5.89363
2294.23	175.37	29.9989	1901.52	277.332	54.753	1.69697	0.247913	-4.87648	2.22525	-0.618062	5.52151
2293.93	175.591	29.8791	1873.89	274.172	58.2305	1.79819	0.0745651	-4.52212	2.60415	-0.579165	5.18489
2293.67	175.76	29.8081	1849.06	271.262	61.1672	2.08099	-0.091648	-4.10988	3.03983	-0.543551	4.87638
2293.42	175.875	29.7745	1827.28	268.502	63.1295	2.55376	-0.253812	-3.63402	3.51662	-0.511118	4.59517
2293.16	175.938	29.7591	1808.76	265.764	63.6901	3.22382	-0.416151	-3.10156	4.0059	-0.483285	4.35367
2292.89	175.953	29.7348	1793.58	262.896	62.5725	4.0949	-0.584128	-2.54001	4.4618	-0.463631	4.18303
2292.59	175.934	29.6686	1781.69	259.745	59.7761	5.16552	-0.764351	-2.00067	4.81847	-0.457873	4.13303
2292.27	175.899	29.525	1772.84	256.203	55.6222	6.4291	-0.964121	-1.5515	4.99115	-0.473064	4.26494

TABLE 2.1 – Exemple de données de capteurs sous format .c3d

Une autre manière de visualiser ces données est de les représenter sous forme de graphique linéaire. Nous pouvons, en tant que personne, plus aisément visualiser ces données et les comparer plus facilement entre elles. Sur la Figure 2.4 ci-dessous, nous pouvons comparer les données des capteurs de chaque pied et en déduire les évènements du cycle de marche en fonction de la manière de l'évolution de celles-ci. Chacune des courbes évoluent en alternance et se croisent lorsque les deux pieds sont en contact avec le sol avant que l'un des deux ne se soulève à nouveau, continuant ainsi le cycle.

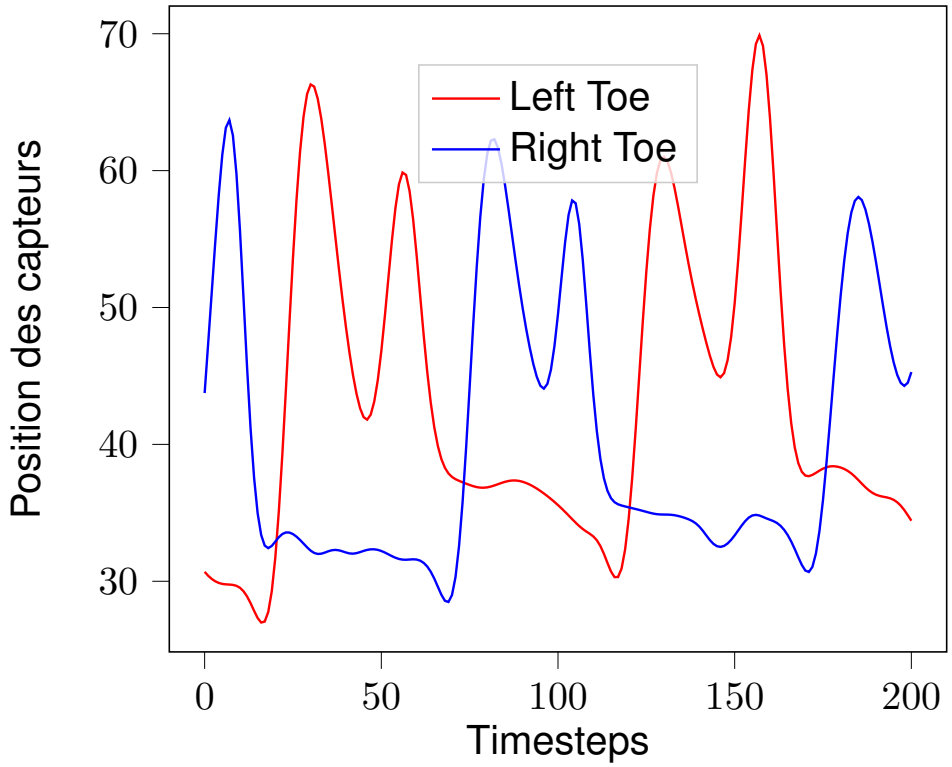


FIGURE 2.4 – Deux capteurs d'un patient

Nous pouvons à travers ce graphique, les différentes données enregistrées par les capteurs. Cependant, la différence de valeurs entre chacune des données rend la tâche d'apprentissage par le réseau de neurones difficile. Afin de simplifier la tâche au réseau de neurones, il est important de mettre toutes ces données à la même échelle, simplifiant ainsi la comparaison et l'apprentissage.

2.2 Preprocessing et normalization

Les données des patients sont stockées dans des fichiers au format ".c3d". Celui-ci est utilisé dans des laboratoires biomécaniques, d'animation et d'analyse de la démarche et permet notamment de conserver des données analogiques synchronisées dans un format standardisé [62].

Pour faciliter le traitement de ces données, nous les avons extraites au format ".csv" et ".npy" et nous n'avons gardé que les fichiers ne contenant pas d'erreurs. Les fichiers étant initialement séparés par pathologie ou par personne en bonne santé, nous les avons regroupés dans un seul et même jeu de données.

Une fois les données regroupées, il est impératif de les normaliser, les mettre à une échelle commune afin de les rendre plus facilement comparables. Cela permet ainsi de limiter l'influence des variables de très grandes valeurs sur la prise de décision finale.

Dans le domaine du Machine Learning, afin de s'assurer que les résultats soient indépendants de la distribution des données qui lui ont été fournies lors de l'entraînement, il est nécessaire de diviser le jeu de données en deux. Une partie "entraînement" qui sera utilisée par chacun des modèles pour apprendre les meilleurs paramètres à estimer pour faire les meilleures prédictions possibles et une partie de test qui sera utilisée pour valider que le modèle est capable de donner des prédictions aussi satisfaisantes que sur le jeu d'entraînement.

3 Modèles

3.1 Informations générales

Dans ce chapitre, nous allons décrire les différents modèles auxquelles nous nous sommes intéréssés. Ceci dans le but d'approcher une solution à notre problématique. Un modèle est une forme d'équation mathématique qui effectue des prédictions en fonction de ses paramètres. Dans le cadre de notre projet, ces paramètres ne lui sont pas fournis et c'est au modèle de les apprendre et de les estimer en fonction des données que nous lui fournissons.

3.2 Algorithmes d'apprentissage

Pour chacun de ses modèles, une extraction et une préparation des données a dûe être effectuée en amont.

Afin d'assurer une division plus objective, nous avons utilisé la technique de la validation croisée. Cette méthode consiste à créer k lots de jeu "d'entraînement" et de "test" dans le but d'utiliser l'intégralité de notre jeu de données pour l'entraînement et pour le test. Pour ce faire, on divise le jeu de données en k parties égales et tour à tour chacune des k parties est utilisée comme jeu de test alors que le reste est utilisé comme jeu d'entraînement. Voici une figure qui illustre le découpage d'un jeu de données en cinq parties avec à chaque fois le jeu de test en blanc et le jeu d'entraînement en orange :

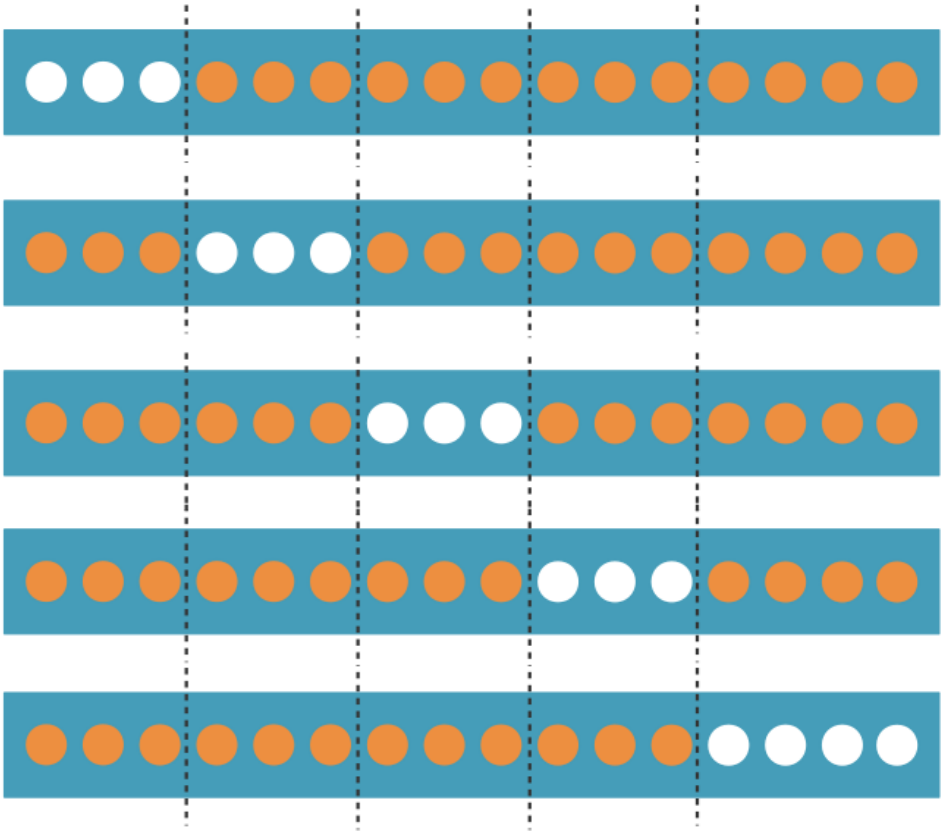


FIGURE 3.1 – Validation croisée - 5 parties

En plus d'avoir un jeu de "test" et un jeu "d'entraînement", ceci permet de valider que les résultats obtenus sont indépendants des données choisies lors de la division du jeu de données initial. Du fait que nous n'avons pas beaucoup de données à notre disposition dans notre projet, nous avons décidé de séparer le jeu initial en quatre jeux de validation croisée avec pour chacun un jeu de "test" et un jeu "d'entraînement", alors que habituellement la séparation se fait plutôt en dix [38]. Il est important de noter que lorsque nous avons utilisé des modèles de type de réseau neurones, chacun des jeux de test a été divisé en deux pour constitué un jeu de validation et un jeu de test.

Nous avons dans un premier temps pris la décision d'utiliser Random Forest comme basline du fait que son fonctionnement est plus simple à comprendre. Nous avons ensuite décider d'utiliser plusieurs architectures de "Recurrent Neural Network"(RNN) et plusieurs architectures de "Convolutional Neural Network"(CNN).

3.2.1 Random Forest

Les arbres décisionnels font partie des méthodes d'apprentissage supervisé des plus simples. Sa classification se fait à travers la construction d'un arbre dont chaque nœud correspond à une décision quant à la valeur à prédire. Chacun des nœuds est connecté à deux ou plusieurs branches. Chaque feuille correspond à une classification ou une décision prise par le modèle.

Cette structure arborescente rend la lecture et la compréhension du modèle possible par un être humain,

contrairement à certaines méthodes de classification. De par ce fait, nous nous en servirons comme modèle de référence.

Les forêts d'arbres décisionnels, comme son nom l'implique, constitue un ensemble d'arbres décisionnels qui se différencient les uns des autres par leur sous-échantillonnage aléatoire sur les données initiales.

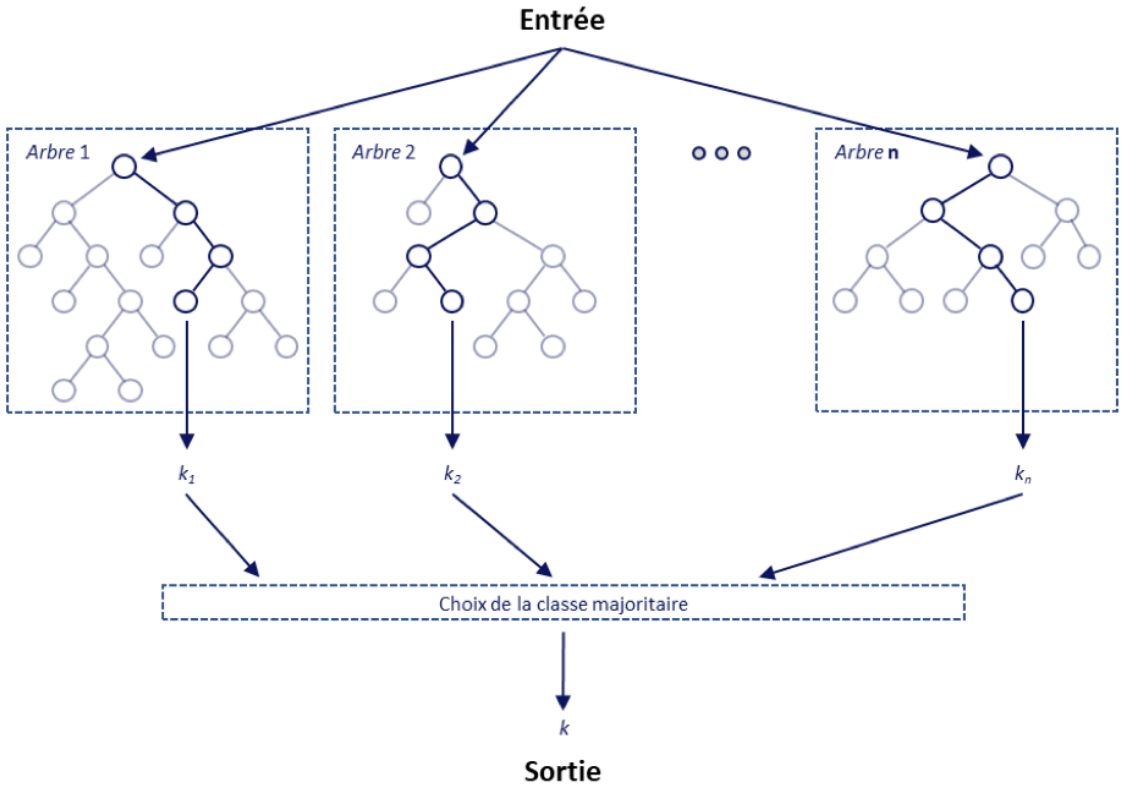


FIGURE 3.2 – Forêts d'arbres décisionnels

Figure adaptée de Koehrsen [63]

La décision prise par chaque arbre est ensuite représentée par un « vote » et la classe ayant accumulé le plus de votes servira de prédiction pour notre modèle.

L'utilisation de plusieurs arbres décisionels permet ainsi, l'obtention d'une prédiction plus précise et minimize l'effet de surentraînement [64].

3.2.1.1 Données

Le pré-traitement à effectuer sur les données pour les forêts d'arbres décisionnels en plus du traitement en amont, consiste en premier lieu à retirer le résultat aux données, puis de convertir les données discretes en données numériques. Par la suite, pour les valeurs nulles des instances, remplacer celles-ci par la moyenne du descripteur. Dans le cas où les instances contiennent plus de 15% de valeurs nulles, les retirer du jeu de données.

3.2.2 RNN

Un "Recurrent Neural Network" ou RNN est un type de réseau de neurones artificiels. Il se différencie des Perceptron multi-couches ou des réseaux de neuronnes à convolution du fait qu'il peut traiter des données à taille variable et qu'il est particulièrement destiné à analyser des séries temporelles. La Figure 3.3 montre comment les données sont traitées par un RNN classique où x est une instance de notre jeu de donnée, x_t est une caractéristique de cette instance, A est la couche cachée de notre RNN et h_t est un état caché donné par notre RNN en fonction de x_t. h_t peut être interprété comme étant l'information que le réseau choisi de garder.

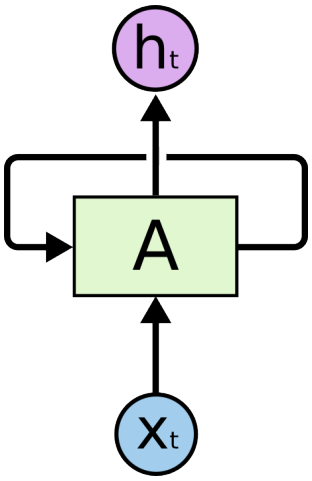


FIGURE 3.3 – RNN

Figure reprise de Rolah [65]

Les flèches de la Figure 3.3 indiquent dans quelles parties du RNN les données ou résultats transitent. De ce fait, l'entrée x_t est injectée dans la couche cachée A qui produit une sortie h_t et qui est retransmise à la couche cachée A pour être utilisée avec la prochaine entrée. Ce fonctionnement est plus facile à visualiser en imaginant un RNN comme un enchaînement de la même couche cachée qui à chaque fois reçoit une entrée différente et émet une prédiction en fonction de celle-ci :

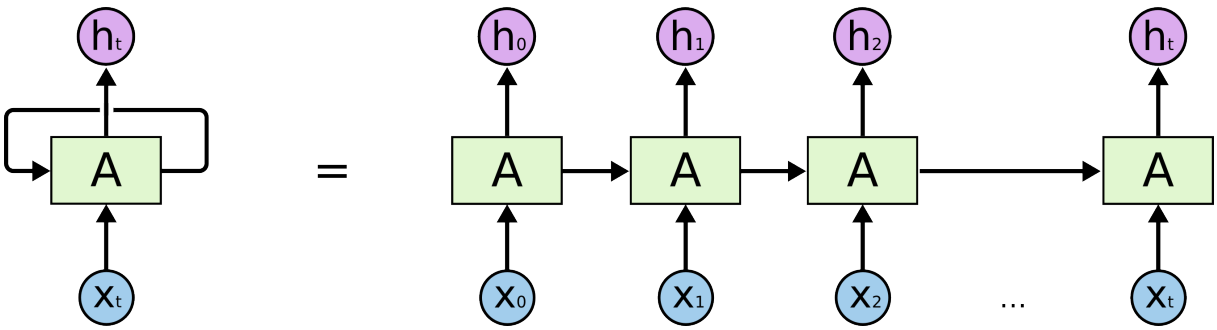


FIGURE 3.4 – RNN - Déroulé

Figure reprise de Rolah [65]

Il est important de noter que sur la Figure 3.4, il s'agit d'une instance x qui contient t caractéristiques qui est passée à travers une seule et même couche cachée mais de manière successive. C'est-à-dire que x_0 est la première caractéristique à être injectée dans la couche cachée A et que la prédiction résultante, ou l'état caché, est h_0. Cet état caché est ensuite utilisé avec x_1 qui est la prochaine caractéristique de notre instance. Ceci est effectué jusqu'à arriver à la dernière caractéristique de notre instance. Dans notre projet, l'état caché de la dernière caractéristique sera utilisé comme prédiction de l'instance. En effet, celui-ci aura pris en compte tous les états cachés de chacune des caractéristiques prédécentes et répondra à notre problème de classification (problème nécessitant qu'une seule valeur à prédire par instance). Par conséquent, le calcul de l'état caché h d'un RNN est défini par la formule suivante où l'état caché h^t est la fonction f de l'état caché h^{t-1} et de l'entrée courante x^t avec θ comme paramètre :

$$h_{(t)} = f(h_{(t-1)}, x_{(t)}; \theta) \tag{3.1}$$

Le paramètre θ de l'équation 3.1 représente des matrices de poids et sont apprises par le réseau lors de la phase d'entraînement. Sur la Figure 3.5 est représenté le "forward pass" d'un RNN qui a de multiples entrées/caractéristiques et qui fournit une sortie/prédiction à chaque état pour une seule instance :

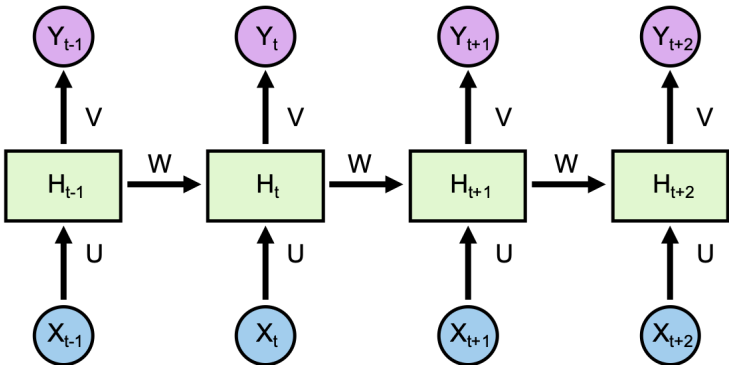


FIGURE 3.5 – RNN - Forward

Figure reprise de Mady [66]

La Figure 3.5, x_{t-1} représente une caractéristique de l'instance x et précède la caractéristique x_t et ainsi de suite. La même logique est appliquée à y qui représente la prédiction et h qui représente l'état caché. Les paramètres u, w et v sont des matrices de poids et représentent donc le θ de l'équation 3.1. u est la matrice de poids pour la couche cachée, w est la même matrice de poids entre les différents "Timesteps" et v est la matrice de poids pour la couche de sortie. Il est important de noter que ce schéma est à interpréter comme une ligne du temps et à lire de gauche à droite. Ainsi, le RNN ici représenté va d'abord recevoir x_{t-1} et effectuer des calculs avec avant de recevoir x_t. De plus, sur chaque prédiction et sur chaque état caché, un RNN utilise une fonction de non-linéarité ou fonction d'activation que nous allons noter σ pour Sigmoid et $soft$ pour Softmax.

Pour rester simple dans l'explication des détails des calculs mathématiques effectués par le RNN, pre-

nons comme exemple l'instant t, ou "Timestep", défini dans la Figure 3.5. Ainsi, l'état caché h_t est défini par la multiplication de x_{t-1} avec la matrice de poids u. Ce résultat est ensuite additionné avec la multiplication de l'état h_{t-1} avec la matrice de poids w. Pour que le h_t désormais obtenu soit le résultat d'une opération non-linéaire, nous devons encore appliquer la fonction de non-linéarité qui, ici, est σ. On peut donc noter h_t ainsi :

$$h_t = \sigma(ux_t + wh_{t-1}) \tag{3.2}$$

Ensuite, afin de définir y_t, il nous faut multiplier h_t avec la matrice de poids v. Puis, nous allons appliquer la fonction de non-linéaritée $soft$ sur ce résultat pour obtenir la prédiction à l'instant t de notre RNN. y_t est ainsi définit :

$$y_t = soft(vh_t) \tag{3.3}$$

Les opérations des équations 3.2 et 3.3 sont effectuées à chaque "Timestep" du RNN avec à chaque fois une caractéristique différente de l'instance actuelle et ce jusqu'à avoir vu toutes les caractéristiques de cette instance. Si chaque instance a 5 caractéristiques, il va donc effectuer 5 "Timestep" et donner 5 prédictions avant de passer à l'instance suivante et ainsi de suite. Ce type de fonctionnement pour un RNN est couramment appelé "many-to-many", faisant référence au fait qu'il a plusieurs caractéristiques par instance en entrée et fournit plusieurs prédictions à chacune de celles-ci.

Le RNN utilisé dans notre problème de classification fait partie de la catégorie "many-to-one". C'est-à-dire que le réseau a plusieurs entrées par instance et que celui-ci ne produit qu'une seule sortie pour l'ensemble de ces entrées. De ce fait, le RNN a toujours trois matrices de poids u, w et v en paramètre à apprendre mais v n'est utilisé qu'une fois par instance plutôt qu'à chaque "Timestep". La figure 3.5 présentée plus haut se simplifie donc légèrement dans notre cas d'utilisation et se présente ainsi :

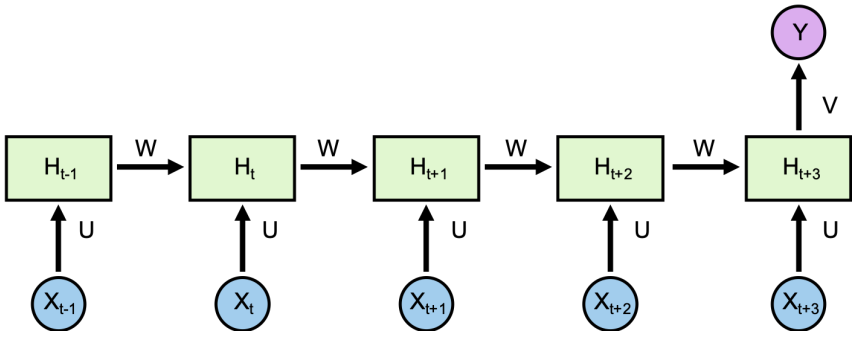


FIGURE 3.6 – RNN - "Forward" avec une seule sortie par instance

Figure adaptée de Mady [66]

La Figure 3.6 ci-dessus présente comment le RNN est capable de nous donner une prédiction en fonction des données qu'il reçoit. L'étape suivante qui permettra au réseau de s'améliorer et d'apprendre de

ses erreurs est la rétropropagation de l'erreur ou "Backpropagation". Celle-ci consiste à calculer l'erreur globale effectuée par le réseau sur une partie ou sur l'ensemble du jeu de données sur lequel il s'est entraîné. Cette étape est réalisée à l'aide de dérivées, puisqu'il s'agit pour le réseau de minimiser son erreur. En effet, le résultat d'une dérivée nous donne la direction à suivre pour maximiser une fonction. Il suffit dans notre cas de prendre la direction opposée pour minimiser ici la fonction d'erreur, appelée aussi fonction de coût. Ainsi, le réseau va mettre à jour ses matrices de poids et tenter de s'améliorer au fur et à mesure des entraînements. Ces mise à jour peuvent être écrites par l'équation 3.4 suivante :

$$\frac{\partial L}{\partial W_t} = \frac{\partial L}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_t} \tag{3.4a}$$

$$\frac{\partial L}{\partial V_t} = \frac{\partial L}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial V_t} \tag{3.4b}$$

$$\frac{\partial L}{\partial U_t} = \frac{\partial L}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial U_t} \tag{3.4c}$$

Comme nous avons été amenés à utiliser la librairie de Machine Learning Pytorch qui s'occupe de calculer les dérivées à notre place, nous n'allons pas entrer plus dans les détails de la rétropopagation de l'erreur.

3.2.2.1 LSTM

Un LSTM, ou "Long short-term memory", est un type de réseau de neurones récurrent. Celui-ci a été pensé par Hochreiter & Schmidhuber dans les années 1997-1999 après que la communauté scientifique aie attribué deux problèmes majeures aux RNNs simples. Ceux-ci étaient impactés par le principe de "Vanishing gradient" et d'"Exploding gradient".

En effet, pour apprendre et s'améliorer, un réseau a besoin de minimiser sa fonction de coût en calculant le gradient de l'erreur, appelé aussi ensemble de dérivées partielles. Or, le "Vanishing" et l'"exploding" gradient sont deux phénomènes qui font tendre le gradient vers 0 ou vers l'infini. Par conséquent, un gradient à 0 revient à ne faire aucune mise à jour sur les poids et correspond à aucune amélioration de la part du RNN dans ses prises de décisions. D'autre part, un gradient qui tend vers l'infini revient à faire des modifications tellement importantes sur les poids du réseau que cela fait "exploser" la fonction d'erreur et, à partir de là, le RNN n'est plus en mesure de minimiser celle-ci dans un temps raisonnable.

Ainsi, un LSTM est construit, de manière à limiter le plus possible le "Vanishing" et "exploding" gradient en ayant une gestion plus complexe de l'information qu'il garde en mémoire au fur et à mesure des "Timesteps". Pour ce faire, un fonctionnement un peu plus complexe s'impose notamment avec la notion de "porte" qui va contrôler si on oublie de l'information provenant des états cachés h précédents et si on en ajoute. Ci-dessous la Figure 3.7 schématisant un LSTM où h est toujours l'état caché, x une instance, o représente ici une prédiction faite par le LSTM, c contient toute l'information des états cachés h gardée depuis le premier "Timestep". F, I et O sont communément appelés des "portes" et σ et $tanh$ sont deux fonctions d'activation.

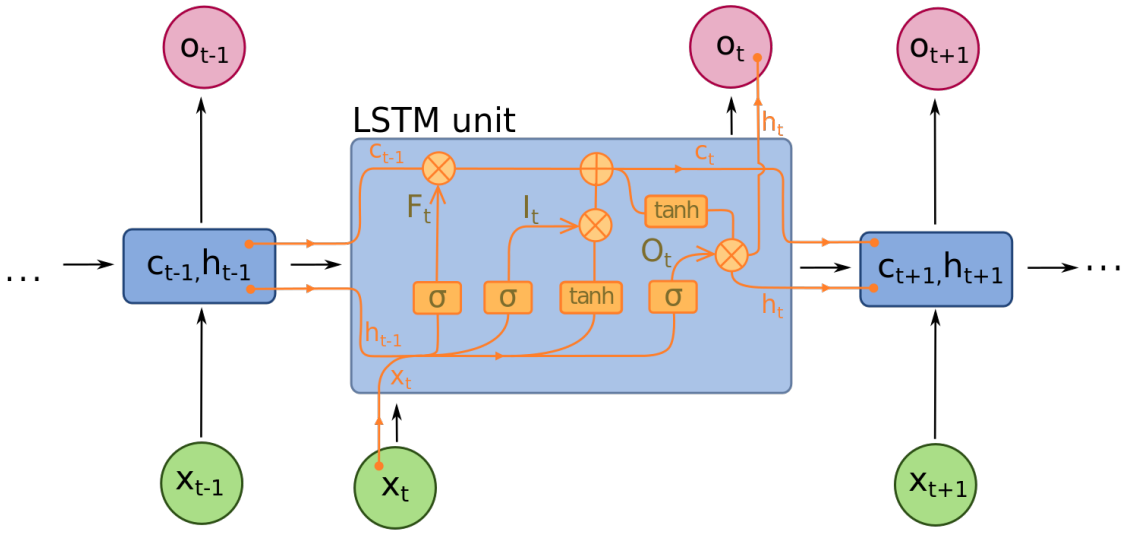


FIGURE 3.7 – LSTM - "Forward" avec plusieurs sorties par instance

Figure adaptée de Marinescu [67]

Tout comme dans un RNN simple, le même principe de temporalité dans l'injection des caractéristiques d'une instance est appliqué dans un LSTM. La matrice c ici contient un historique de toute l'information que le réseau a décidé de garder tout au long des "Timesteps" d'une instance. Par conséquent, un "Timestep" reçoit en entrée 3 paramètres : une caractéristique de l'instance courante x_t, l'état caché précédent h_{t-1} et la cellule d'état précédente c_{t-1} (couramment appelée "cell-state"). La première étape à l'instant t pour un LSTM est de passer par la "Forget gate" qui prend en paramètre la caractéristique courante x_t et l'état caché précédent h_{t-1} et y applique la fonction d'activation σ. Ce résultat est ensuite multiplier avec le c_{t-1}. Ainsi, comme Sigmoïd est une fonction qui donne des valeurs entre 0 et 1, ceci aura pour effet d'oublier ("forget") ou de garder des valeurs de c_{t-1}. Cette étape peut être formulée de la manière suivante :

$$F_t = \sigma(W_f \cdot [h_{t-1}] + b_f) \tag{3.5}$$

L'étape suivante (I_t) est le passage dans la "porte" intitulée "Input gate" qui prend en paramètre x_t et h_{t-1} et sur laquelle est appliquée σ dans un premier temps. L'étape suivante, que nous allons appelée $\tilde{C}_t$, consiste à utiliser les même paramètres sur lesquels nous appliquons cette fois-ci la fonction d'activation $tanh$. I_t et $\tilde{C}_t$ sont multipliés pour ensuite être ajoutés au C_t et constituent donc un moyen d'ajouter de l'information à la cellule d'état. Ainsi, nous pouvons formuler comme ceci :

$$\begin{aligned} I_t &= \sigma(W_i \cdot [h_{t-1}] + b_i) \\ \tilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{3.6}$$

Ainsi, la cellule d'état qui est reçue en paramètre de chaque "Timestep" et qui est ensuite envoyé au

suivant, est défini par l'équation 3.5 et 3.6 et peut s'écrire ainsi :

$$C_t = F_t \cdot C_{t-1} + I_t \cdot \tilde{C}_t \tag{3.7}$$

Finalement, la dernière étape est représentée par O_t qui s'intitule "output gate", ou porte de sortie. Celle-ci va décider quelle partie de C_t sera transmise en sortie. Pour ce faire, la fonction Sigmoïd σ est appliquée sur la caractéristique x_t de l'instance courante et de l'état caché précédent h_{t-1}. La fonction d'activation est ensuite appliquée sur C_t et, enfin, les deux résultats obtenus sont multipliés. Voici l'équation exprimant ce raisonnement :

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.8a}$$

$$h_t = O_t \cdot tanh(C_t) \tag{3.8b}$$

Il est important de bien faire la différence entre les deux sorties qu'un "Timestep" transmet au suivant, si celui-ci n'est pas le dernier de la chaîne. En effet, C_t est utilisé pour filtrer ce qui doit être gardé ou non au fil des "Timesteps", alors que h_t représente de l'information supplémentaire à garder ou non. C'est là que réside la principale différence entre un LSTM et un RNN.

L'apprentissage du LSTM se fait tout comme un RNN simple, en mettant à jour ses matrice de poids, qui sont au nombre de 4. Cette mise à jour reprend le même principe que celui expliqué dans la section 3.2.2 et dans l'équation 3.4.

Comme dans la Figure 3.5, la Figure 3.7 schématise le fonctionnement d'un réseau "Many-to-Many". La version du LSTM utilisée dans ce projet est très similaire, mais diffère légèrement et ne produit qu'une seule prédiction par instance lors du dernier "Timestep" :

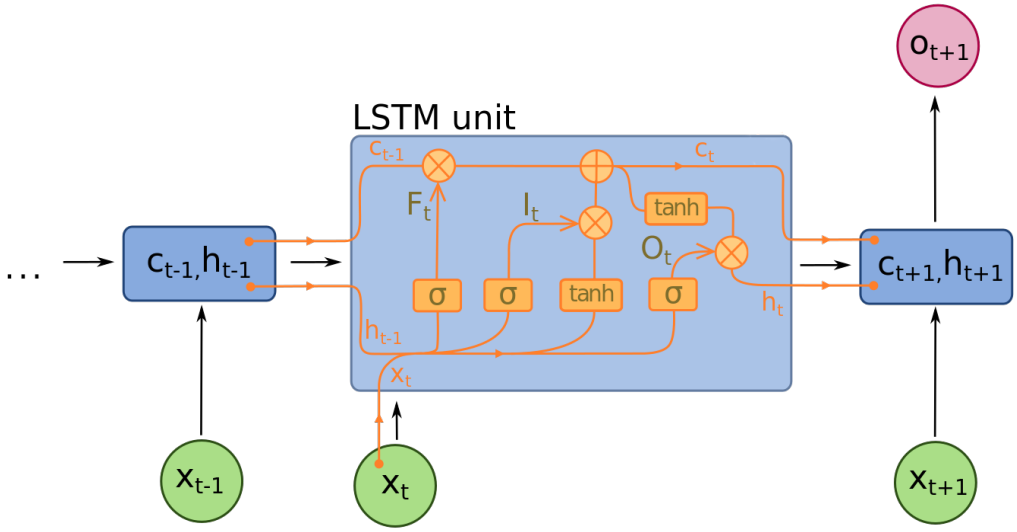


FIGURE 3.8 – LSTM - "Forward" avec une seule sortie par instance

Figure adaptée de Marinescu [67]

À chaque "Timestep", il n'y a donc pas de copie de l'état caché qui est effectuée et sur laquelle une fonction d'activation telle que Softmax est appliquée. Ce type de fonction d'activition, dans notre cas, ne sera appliquée qu'au dernier h_t.

3.2.2.2 GRU

Nous allons discuté brièvement d'une autre achitecture de RNN que nous avons utilisée dans le cadre de ce projet de recherche : les "Gated Recurrent Units" (GRU). Cettre structure est particulièrement proche des LSTM discutés dans la section 3.2.2.1. Voici une image qui met en évidence les différences d'architecture entre GRU et LSTM :

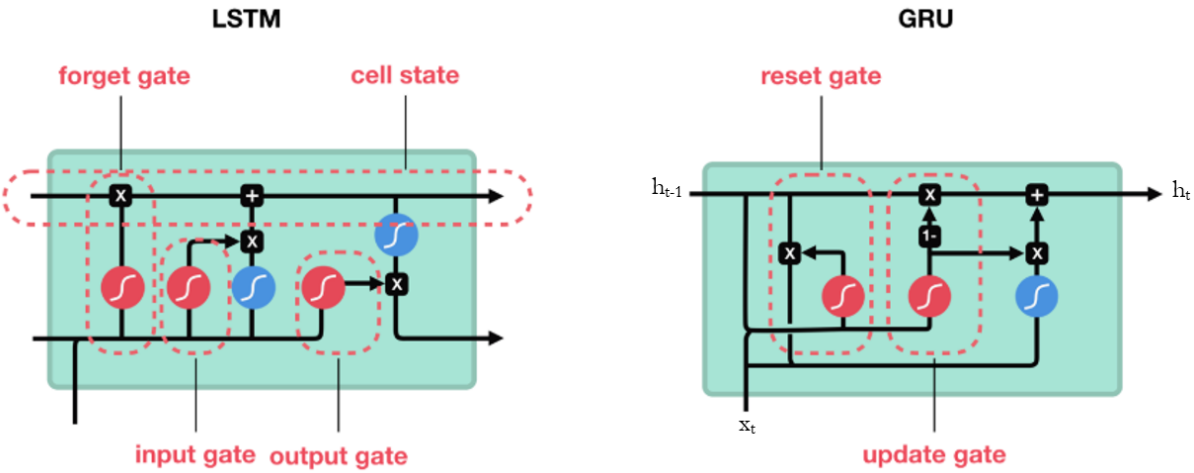


FIGURE 3.9 – GRU et LSTM - Comparaison

Figure adaptée de Nguyen [68]

Le principe fondamental du LSTM qui est de sélectionner l'information à garder au travers des "Timesteps" est préservé. Cependant, nous pouvons observer sur la Figure 3.9 qu'une seule sortie émane du GRU. En effet, la gestion de l'information se fait uniquement sur h par le biais de deux "portes", la porte de réinitialisation ou "reset gate" et la porte de mise à jour ou "update gate". De manière conceptuelle, la première porte permet de décider quelles informations il est utile de garder parmi h_{t-1}. La porte de mise à jour a le même fonctionnement que la "forget gate" et l'"input gate" que l'architecture d'un LSTM montrée sur la Figure 3.8.

Les détails techniques des opérations mathématiques effectuées lors de l'utilisation d'un GRU ne vont pas être détaillées dans ce projet de recherche. En effet, cette structure fonctionne de manière très similaire à un LSTM et les principales différences résultent dans l'ordre et le type d'opération effectuées à chaque "Timestep". De plus, les résultats de ces deux types différents de modèles sont souvent comparables.

3.2.2.3 Données

Nous avons déjà parlé des données et de leur distribution dans la section 1.3 et du pré-traitement global effectué dans la section 2.2. Nous allons ici parler de pré-traitement des données propre aux RNNs.

En effet, les RNNs sont connus pour pouvoir traiter des séquences à longueur variable. C'est exactement le cas de nos données qui contiennent un nombre de "frame" variant d'une visite médicale à une autre. Ceci est dû à la vitesse de marche qui fluctue selon les patients et leur état de santé au moment de l'enregistrement de leur démarche. Cependant, l'utilisation de la librairie Pytorch nous contraint à structurer les données en "batch" avant de pouvoir les donner en entrée de nos réseaux. Un "batch" est un regroupement de plusieurs instances.

Ainsi, nous avons été obligé de stocker les données à taille variable sous forme de "batch" de tableaux à l'aide de la librairie Numpy. Celle-ci ne nous autorisant pas à stocker des vecteurs de taille variable dans une seule et même matrice, nous avons été forcés d'utiliser du "padding". Ceci consiste à sélectionner la taille du plus grand vecteur du jeu de données comme référence et d'ajouter des 0 aux vecteurs plus petits pour que ceux-ci obtiennent la même taille que le plus grand. Par conséquent, nous utilisons des données à taille variable mais stockées dans des vecteurs de taille similaires compléter par des 0.

Il est important de noter que le "padding" n'a aucun impact sur les résultats fournis par un RNN. En effet, des valeurs à 0 sont utilisées dans notre cas. Par conséquent, l'utilisation des opérations de multiplication et d'addition des formules mathématiques présentées dans la section 3.2.2 n'affectera en rien les résultats. D'autre part, [69] a démontré qu'une importante différence dans les résultats d'un LSTM existait lorsque l'on choisissait de faire du "post-padding" ou du "pre-padding". La différence entre les deux étant qu'avec le "pre-padding", les 0 sont ajoutés au début et dans le "post-padding", les 0 sont ajoutés à la fin du vecteur.

Nous avons également préparé les données pour qu'elles aient une taille fixée d'avance et en évitant de faire du padding. Nous avons pour celà utiliser la librairie opencv-python et plus particulièrement la méthode resize() fournie avec. Celle-ci permet de redimensionner une image d'origine tout en préservant le plus possible son aspect. Chaque instance ayant une dimension pour le nombre de "frame" et une dimension pour le nombre de données de capteurs, nous pouvons interpréter ces deux dimensions comme étant la hauteur et la largeur d'une image. Cette préparation des données a été effectuée afin de pouvoir comparer les résultats obtenu avec une longueur fixe et une longueur dynamique.

3.2.3 CNN

Un CNN (Convolution Neural Network, réseau de neurones à convolution ou réseau de neurones convolutif) est un autre type d'algorithme que nous avons utilisé lors de la réalisation de ce projet de recherche.

3.2.3.1 Fonctionnement

Les réseaux de neurones à convolution existent depuis les années 1980 et ont été créés pour améliorer les prédictions dans le domaine du traitement d'image. Ceux-ci possèdent la capacité de pouvoir trouver des éléments intéressants dans l'image indépendamment de leur position et rotation.
Leur archicture peut être décomposée en 4 parties :

- *convolution*
- fonction d'activation
- *pooling*
- classification

3.2.3.1.1 Convolution

La convolution, expliquée de manière simplifiée, est l'application d'un filtre sur une image donnée. D'une manière mathématique, c'est la multiplication élément par élément d'une matrice w avec une sous partie d'une matrice x et en gardant la somme du résultat de la multiplication.
Où w est le filtre et x est l'image d'entrée.

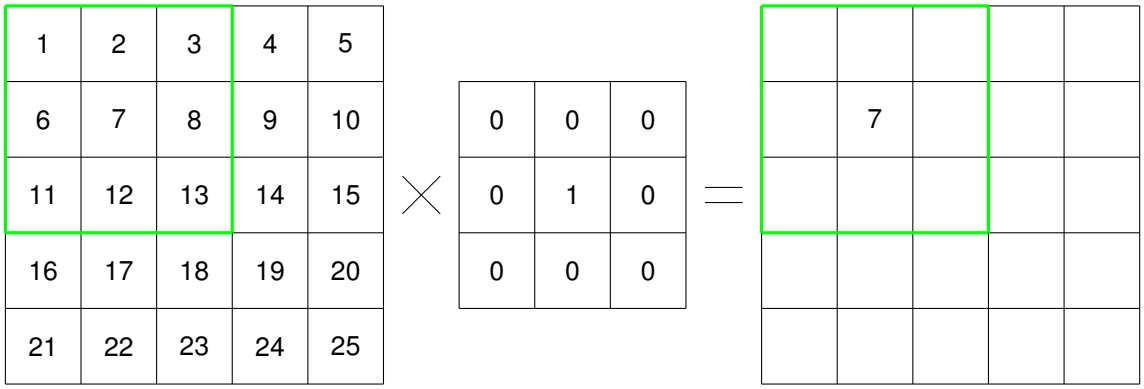


FIGURE 3.10 – Opération de convolution

Figure adaptée de Pauly et al. [70]

La première opération de convolution serait :

$$1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 6 \cdot 0 + 7 \cdot 1 + 8 \cdot 0 + 11 \cdot 0 + 12 \cdot 0 + 13 \cdot 0 = 7 \tag{3.9}$$

Toutes ces opérations peuvent être généralisées par l'équation suivante :

$$z_{ij} = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} x_{i+k,j+l} w_{k+1,l+1} \tag{3.10}$$

Où z_{ij} correspond à un élément de l'output d'une opération de convolution et F correspond à la taille du filtre w. Dans cette équation, les deux sommes vont jusqu'à F car nous partons du principe que le filtre utilisé est carré.

Lors d'une opération de convolution, à l'inverse du perceptron multicouches, tous les neurones ne sont pas connectés à tous les neurones. Cette caractéristique permet de réduire la quantité de poids à apprendre tout en ayant une meilleure précision, au dépens d'un temps de calcul supérieur - amorti par la puissance de calcul des processeurs optimisés pour le calcul matriciel. Toutefois, l'objectif premier est de pouvoir mettre en évidence les parties de l'input x qui s'approchent le plus du filtre choisi w. Pour illustrer cela, prenons la Figure 3.11 :

FIGURE 3.11 – HEG 20 ans

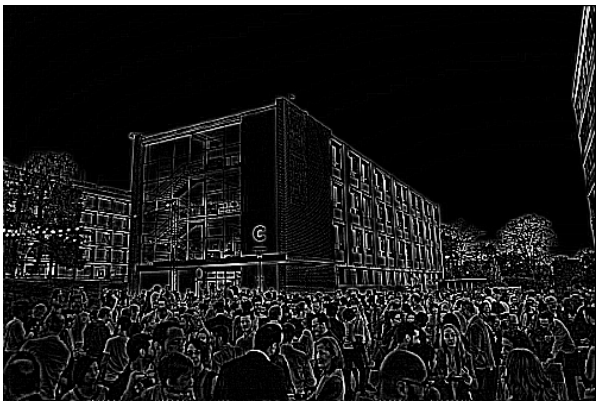
Figure reprise de Haute école de gestion de Genève [71]

Voici le rendu lorsqu'un filtre, défini manuellement, permettant de mettre en évidence les contours d'une image et appliqué sur celle-ci :

Et un filtre ayant la particularité de mettre en évidence les lignes horizontales :

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

(a) Filtre pour contours



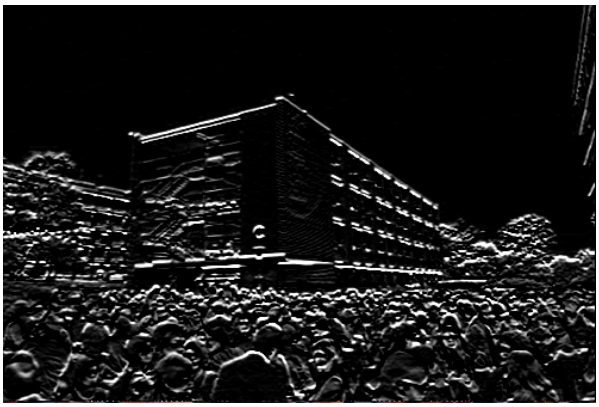
(b) Rendu de l'image

FIGURE 3.12 – Un exemple d'une opération de convolution

Figure adaptée de Haute école de gestion de Genève [71]

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

(a) Filtre pour contours



(b) Rendu de l'image

FIGURE 3.13 – Un exemple d'une opération de convolution

Figure adaptée de Haute école de gestion de Genève [71]

Ces filtres, bien que définis dans les exemples de façon manuelle dans les Figures 3.12 et 3.13, sont des poids appris lors de la phase d'entraînement du réseau de neurones à convolution. Tel que dans un perceptron multicouche utilisant SGD, les poids sont appris de la façon suivante :

$$w := w - \alpha \cdot \frac{\partial L}{\partial w} \tag{3.11}$$

Puisque nous avons utilisé la librarie de Machine Learning PyTorch, les dérivées sont calculées automatiquement grâce à un graph computationel dynamique qui est créé lors du "forward pass" et qui se sert d'une forme optimisée de la "chain rule".

Ensuite, une image est souvent composée de trois channels : (RGB - Red, Green, Blue) - Rouge, Vert et Bleu.

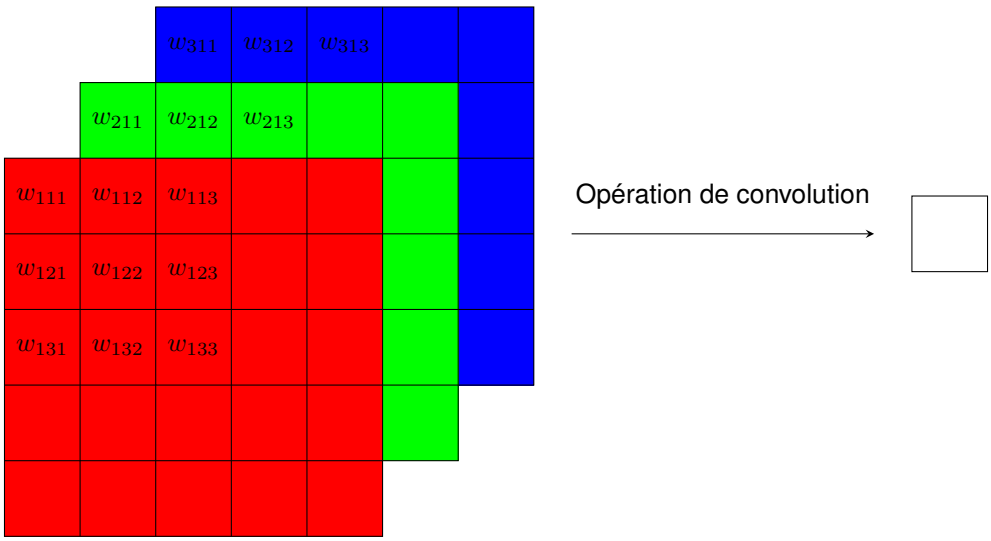


FIGURE 3.14 – Image possédant trois channels

Figure adaptée de Kalousis [72]

Comme nous pouvons le voir sur la Figure 3.14, chaque channel possède son propre w. Un filtre de convolution est appliqué sur tous les channels en même temps et cette opération créé un seul channel dans une nouvelle image générée. Nous pouvons utiliser autant de filtres que nous le voulons et ceux-ci font partie des hyperparamètres à choisir.

Dans les couches de convolution, les hyperparamètres qui nous restent à définir sont la largeur des filtres, et le nombre de pas pour chaque opération - souvent appelé *strides*. Souvent, les valeurs utilisées sont 3 ou 5 pour la taille des filtres et 1 pour les strides.

Pour finir, chaque couche de convolution doit être suivie par une fonction de non-linéarité (e.x. LReLU, ReLU).

3.2.3.1.2 Pooling

La couche de pooling est une couche dont le but est de réduire la taille d'une image tout en gardant les éléments importants à la prédiction du réseau de neurones.

Prenons l'exemple d'une image à un seul channel - Figure 3.15. Dans cette image, nous allons définir des fenêtres dans lesquels la valeur la plus élevée sera gardée dans la nouvelle image générée - cette opération s'apelle MaxPooling.

Comme nous l'avons vu dans la section 3.2.3.1.1, nous essayons de trouver les parties de l'image qui s'approchent le plus du filtre appliqué. Ces parties auront des valeurs de pixels qui sont plus élevées.

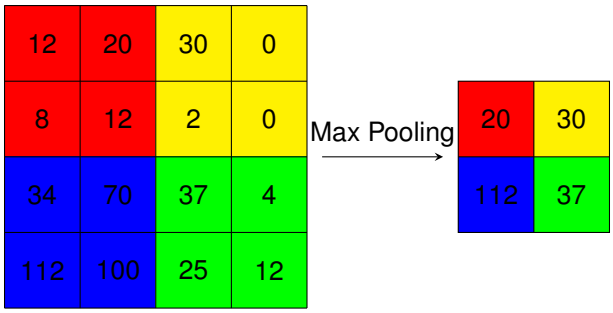


FIGURE 3.15 – Opération de MaxPooling

Figure adaptée de Stanford CS231n [73]

Puisque le max-pooling garde seulement les parties les plus élevées d'une image, seules les parties intéressantes sont gardées pour les couches suivantes.

Il existe également d'autres types de pooling différents (average pooling - utilisant la moyenne, min-pooling - utilisant la valeur la plus petite) mais celui qui présente les meilleurs résultats est le max-pooling. Pour finir, tout comme une couche de convolution, nous devons définir les hyperparamètres qui sont toujours la taille de la fenêtre (typiquement 2) et le nombre de pas (typiquement 2).

3.2.3.2 Données

Les réseaux de neurones à convolution ont besoin d'utiliser des images en entrée. En effet, comme expliqué dans la section 3.2.2.3, les données que nous avons utilisées sont structurées comme une image. Toutefois, elle doivent être adaptées à notre cas spécifique.

En effet, l'opération de convolution expliquée dans la section 3.2.3.1.1 marche pour des images en deux dimensions (trois si nous comptons les channels) et est adaptée pour trouver des éléments qui ont un lien spatial. C'est-à-dire, le rapport des pixels à la verticale et à l'horizontale est important. Or, dans notre cas, chaque capteur utilisé dans l'image n'a aucun rapport avec les capteurs se trouvant à côté. Pour cela, nous avons décidé d'utiliser un réseau de neurones à convolution à une dimension.

Dans ce type de réseau de neurones, seule une dimension sera importante : l'horizontale, qui correspond au changement dans le temps. La dimension verticale d'une image n'a pas d'importance. Ainsi, chaque ligne est traitée comme étant un channel différent et par conséquent, chacune a ses propres poids (w).

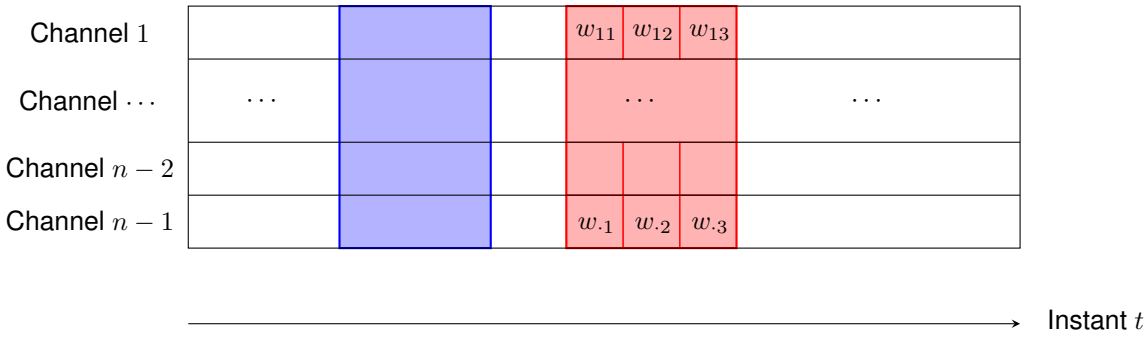


FIGURE 3.16 – Couche de convolution à une dimension

4 Résultats

4.1 McNemar

En Machine Learning, la performance d'un modèle supervisé est évaluée avec différents métriques tels que l'erreur que nous noterons E. Celle-ci est le ratio de prédictions erronées faites par un modèle A par rapport au prédictions attendues sur un jeu de données. Pour nous permettre de savoir si la différence d'erreur, notée ici γ, entre un modèle A et un modèle B est significative, nous avons utilisé le test de McNemar. [74] Celui-ci nécessite de poser deux hypothèses :

$$H_0 : \gamma = E_A - E_B = 0 \tag{4.1a}$$

$$H_1 : \gamma = E_A - E_B \neq 0 \tag{4.1b}$$

Après avoir posé ces hypothèses, il est nécessaire de définir le taux de signification α souhaité (5% par exemple). Puis, nous calculons le niveau de signification observé appelé ici p et nous vérifions si celui-ci est plus grand ou plus petit que α. Ainsi, si $p < \alpha$, alors H_0 est rejetée et signifie que la différence entre l'erreur du modèle A et B est significative. Autrement dit, l'hypothèse H_0 assume que l'erreur du modèle A et du modèle B sont équivalentes :

$$H_0 : E_A = E_B \tag{4.2a}$$

$$H_1 : E_A \neq E_B \tag{4.2b}$$

Le calcul du taux de signification observé p se fait à l'aide d'un tableau de contingence entre les prédictions que deux modèles ont faites et les prédictions attendues sur un jeu de données. Ci-après une figure exemplifiant cette table de contingence pour deux modèles différents sur le même jeu de données :

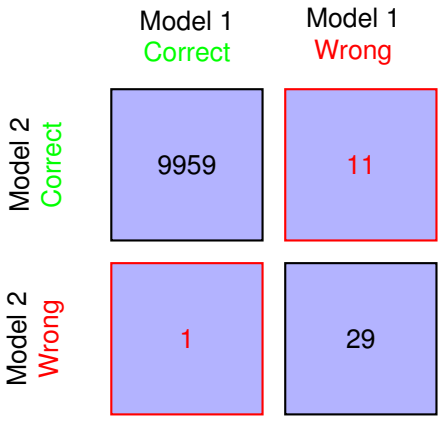


FIGURE 4.1 – McNemar - Tableau de contingence

Figure adaptée de Raschka [75]

Les chiffres de la première ligne sur la Figure 4.1 représentent le nombre de prédictions correctes du 2ème modèle et la deuxième ligne représente les prédictions erronées faite par celui-ci. La même logique est appliquée au niveau des colonnes, mais concernant le 1er modèle. Ce sont les chiffres en rouge qui nous intéressent dans ce tableau de contingence, puisqu'il s'agit des différences de performance entre les deux modèles. Si on se réfère à l'hypothèse H_0 de l'équation 4.2, l'erreur étant équivalente, les fréquences du tableau de contingence devrait l'être aussi. En réutilisant les valeurs de la Figure 4.1, nous pouvons ainsi définir la table de contingence théorique suivante :

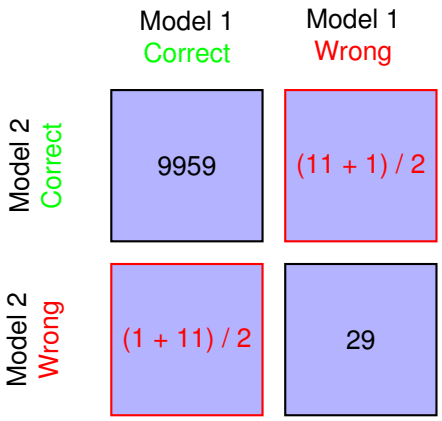


FIGURE 4.2 – McNemar - Tableau de contingence théorique selon H_0

Figure adaptée de Raschka [75]

Ces deux tables de contingence désormais établies, nous allons utiliser le test du Chi-carré noté χ^2 et défini par :

$$\chi^2 = \sum_{j=1}^{J} \frac{(f_{o_j} - f_{t_j})^2}{f_{t_j}} \tag{4.3}$$

où les fréquences observées d'une cellule de la Figure 4.1 sont définies par f_{o_j} et les fréquences

théoriques d'une cellule de la figure 4.2 sont définies par f_{t_j}. Enfin, il ne reste plus qu'à appliquer l'équation 4.3 avec les valeurs des Figures 4.1 et 4.2 :

$$\chi^2 = \frac{(11 - (11+1)/2)^2}{(11+1)/2} + \frac{(1 - (11+1)/2)^2}{(11+1)/2} \tag{4.4a}$$

$$\chi^2 = \frac{25}{6} + \frac{25}{6} \tag{4.4b}$$

$$\chi^2 = \frac{50}{6} \tilde{=} 8.33 \tag{4.4c}$$

Le niveau de signification p observé ici est donc de 8.33%. Étant donné que celui-ci est plus élevé que notre niveau de signification $\alpha = 5\%$ souhaité, l'hypothèse H_0 est acceptée et les résultats des modèles A et B sont considérés comme équivalents.

4.2 Différentes architectures

4.2.1 Random Forest

Cette section traite des forêts d'arbres aléatoires qui ont servi de base de référence pour la suite du projet. Nous avons utilisé les hyperparamètres par défaut pour notre baseline et avons poursuivi depuis là. Les résultats obtenus avec ces pour 1, 10, 100, 1000 et 10000 arbres sont respectivement 32.74%, 39,69%, 42,86%, 43,48%, 44,39%. Malgré le nombre important d'arbres utilisés, le problème reste trop complexe pour cet algorithme et la précision ne s'améliore que très légèrement pour un coût qui lui ne cesse de croître.

4.2.2 RNN

Cette rubrique a pour but de présenter les architectures de RNN simples, de LSTM et de GRU qui ont été utilisées dans ce projet de recherches. En effet, ces trois architectures faisant partie du même type de modèle, des hyper-paramètres parfois similaires ont été employés. En dehors des hyper-paramètres, nous avons utilisé des données à taille variable et des données avec une taille fixée à 100. Lorsque la taille était fixée et afin d'éviter de devoir choisir les caractéristiques à garder pour chaque instance, nous avons utilisé une librairie Python qui s'intitule "opencv-python" et qui permet de redimensionner une image tout en gardant un maximum ses proportions d'origine.

Pour simplifier la présentation de ces architectures, voici un tableau récapitulatif des hyper-paramètres utilisés :

	RNN	LSTM	GRU
Learning rate	1e-04	1e-04	1e-04
Taille de batch	128	32	128
Couches cachées	5	3	6
Nombre d'unités cachées	200	500	300
Dropout	0.3	0.3	0.3

TABLE 4.1 – Architectures des types de RNN testés

Les unités cachées et le nombre de couches cachées sont des paramètres qui contrôlent la complexité du modèle. Plus ils sont élevés et plus le modèle devient complexe, mais plus il est enclin à résoudre des problèmes complexes. Ces deux paramètres ont été augmentés progressivement en commençant par 50 unités cachées et 1 couche cachée. Etant donné que nous avons peu de données, les RNN testés avec peu de couches et peu d'unités cachées avait tendance à se surentraîner. Nous avons donc, en plus d'augmenter leur complexité pour ralentir la vitesse sur-apprentissage, diminué le "learning rate" à 1e-04 et augmenté le dropout à 0.3, dans l'espoir de les rendre plus généralisables. Le paramètre "dropout" contrôle le pourcentage de neurones de la couche cachée à désactiver lors de la phase d'entraînement de celui-ci. Plus il est élevé, plus cela va permettre au réseau de se généraliser et d'être capable de fournir des aussi bonnes prédictions sur le jeu d'entraînement que sur un jeu de données qu'il n'a jamais vu. Le "learning rate" correspond à la vitesse à laquelle le réseau va mettre à jour ses poids et, par conséquent, la vitesse à laquelle il va apprendre. Il est très important de ne pas fixer ce paramètre à une valeur trop élevée pour éviter au réseau de faire des mises à jour trop importantes de ses poids et de ne jamais réussir à apprendre correctement comment faire de meilleurs prédictions. Le dernier paramètre qu'il nous restait à modifier était la taille des "batchs" qui indique au réseau le nombre d'instances à voir avant de mettre à jour ses poids. Dans notre cas, les meilleurs résultats ont été obtenus avec une taille de 128.

Pour l'ensemble des modèles cités ci-dessus, des entraînements de 200 "epochs" ont été effectués.

4.2.3 CNN

Dans cette section nous allons détailler les architectures testées ainsi que la procédure de sélection de nos meilleurs CNN. Tout d'abord, nous avons testé des modèles avec plusieurs types de structurations de nos données. Puisque les CNN doivent avoir des tailles limitées, nous avons décidé de compresser nos données en une taille fixe de 100 *timesteps* ou 200. Nous avons également testé en utilisation tous les capteurs uniquement et les capteurs avec les angles des membres des patients.

Voici les architectures utilisées dans les CNN à une couche de convolution : (Nombre de filters, taille des filtres)

Tous les modèles utilisent learning rate de $1e-4$ et une taille de batch = 32. Aussi la taille des *strides* est toujours définie à 1 pour la convolution et elle est égale à la taille de la fenêtre pour le MaxPooling.

où

Modèle	Sans régularisation	Avec régularisation(1)	Avec régularisation(2)
Conv1d	(256, 5)	(256, 5)	(256, 5)
Regularization	-	BatchNorm	BatchNorm
Activation	LeakyReLU	LeakyReLU	LeakyReLU
MaxPooling	2	2	2
Applatissement	Oui	Oui	Oui
Regularization	-	Dropout(0.5)	BatchNorm + Dropout(0.5)
FC	128	128	128
Activation	LeakyReLU	LeakyReLU	LeakyReLU
FC	6	6	6

TABLE 4.2 – Architectures utilisées avec des modèles à une couche de convolution

Modèle	Sans régularisation	Avec régularisation
Conv1d	(256, 5)	(256, 5)
Regularization	-	BatchNorm
Activation	LeakyReLU	LeakyReLU
MaxPooling	3	3
Conv1d	(64, 3)	(64, 3)
Regularization	-	-
Activation	LeakyReLU	LeakyReLU
MaxPooling	2	2
Flatten	Oui	Oui
Regularization	-	BatchNorm + Dropout(0.5)
FC	128	128
Activation	LeakyReLU	LeakyReLU
FC	6	6

TABLE 4.3 – Architectures utilisées avec des modèles à deux couches de convolution

Types de données	100M	200M	200MA
1 Couche convolution sans régularisation	77%	-	-
1 Couche convolution avec régularisation (1)	83%	-	83%
1 Couche convolution avec régularisation (2)	78%	-	-
2 Couches convolution sans régularisation	80%	78%	82%
2 Couches convolution avec régularisation	80%	78%	82%

TABLE 4.4 – Résultats des architectures CNN sur les plusieurs types de données

— 100M correspond à une taille de 100 *timesteps* avec les capteurs

— 200M correspond à une taille de 200 *timesteps* avec les capteurs

— 200MA correspond à une taille de 200 *timesteps* avec les capteurs et les angles

4.3 Tableau comparatif des meilleurs modèles

Dans cette section, nous allons vous présenter les résultats obtenus avec les différentes architectures testées. Nous allons commencer par vous montrer un tableau comparatif des résultats obtenus sur les 3 meilleures architectures de RNN utilisées :

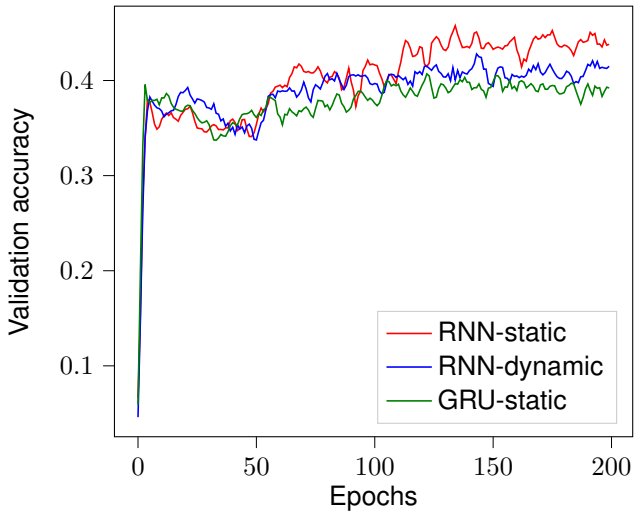


FIGURE 4.3 – Précision sur le jeu de validation pour les 3 meilleurs RNN

Sur la Figure 4.3 est affichée l'évolution de la précision sur le jeu de validation au travers des "epochs". La courbe rouge correspond au RNN avec des données en entrée d'une taille fixée à 100, la courbe bleue à la même architecture mais avec des données à taille variable et en vert au GRU avec des données d'une taille fixée à 100 également. On peut observer une précision légèrement plus élevée pour le RNN-static, même si la différence entre les trois architectures reste faible : "RNN-static" 51%, "RNN-dynamic" 48% et "GRU-static" 48%.

Pour nous assurer que cet écart en précision est véritablement significatif, nous avons utilisé le test de McNemar dont les résultats vous sont présentés dans les tableaux ci-dessous :

		RNN-static Correctes	RNN-static Fausses
RNN-dynamic	Correctes	161	94
RNN-dynamic	Fausses	106	158

p-value = 0.72

TABLE 4.5 – Test de McNemar entre RNN-static et RNN-dynamic

		RNN-static Correctes	RNN-static Fausses
RNN-dynamic	Correctes	174	74
RNN-dynamic	Fausses	93	178
p-value = 2.16			

TABLE 4.6 – Test de McNemar entre RNN-static et GRU

		RNN-static Correctes	RNN-static Fausses
RNN-dynamic	Correctes	154	94
RNN-dynamic	Fausses	101	170
p-value = 0.25			

TABLE 4.7 – Test de McNemar entre RNN-dynamic et GRU

Les tableaux 4.5, 4.6, 4.7 présentent les résultats du test de McNemar. Comme nous pouvons le constaté sur chacunes des figures, les test de significativités sont tous plus grands que 5%. Ainsi, comme expliqué dans la section 4.1, l'hypothèse H_0 considérée comme vraie et tous les modèles se valent. Nous décidons donc de garder le RNN-static comme meilleur réseau de neuronnes récurrents.

Voici désormais un graphique montrant l'évolution de la précision du RNN-static sur le jeu d'entraînement et sur le jeu de validation au travers des epochs :

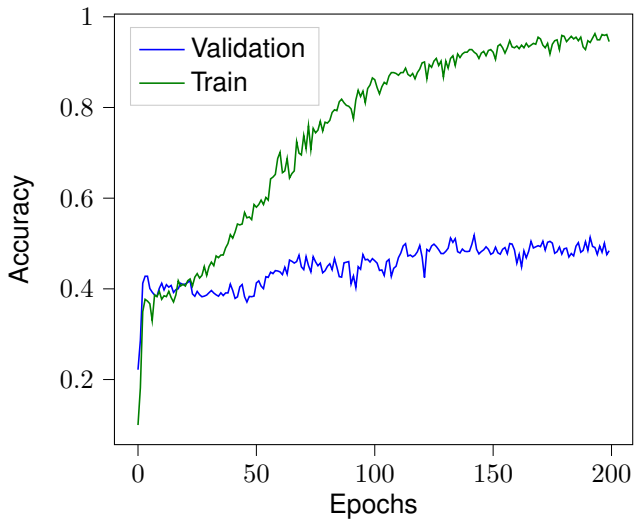


FIGURE 4.4 – Précision sur le jeu de validation pour le meilleur RNN

Sur le graphique 4.4, les "epochs" sont représentés sur l'axe des abscisses et sur l'axe des ordonnées est représentée la précision. On peut observer qu'à partir de l'"epoch" 50, environ, le réseau commence à surentraîner sur le jeu d'entraînement. En effet, la précision sur celui-ci continue d'augmenter alors que la précision sur le jeu de validation stagne. Ainsi, si l'entraînement est continué, le réseau risque

de spécialisé sur le jeu de données passer à l'entraînement et ne de ne pas réussir à donner des prédictions satisfaisantes sur un jeu de données qu'il n'a jamais vu.

Voici maintenant les résultats concernant les réseaux de neurones convolutifs :

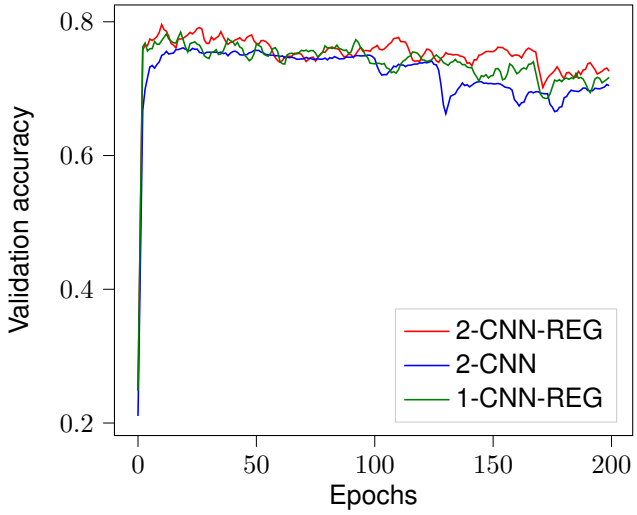


FIGURE 4.5 – Précision sur le jeu de validation pour les 3 meilleurs CNN

Dans la Figure 4.5, 2-CNN-REG correspond à un réseau de neurones à convolution à deux couches utilisant de la régularisation ; 2-CNN correspond au même modèle sans régularisation. 1-CNN-REG correspond au modèle vu dans 4.4 qui s'appelle 1 Couche convolution avec régularisation (1). Comme nous pouvons voir ici, les trois modèles sont presque équivalents au niveau de leur précision. En effet, leur précision sur le jeu de test est de 82%, sauf pour le CNN à une couche de convolution qui a pu atteindre 83% de précision.

		2-CNN-REG Correctes	2-CNN-REG Fausses
2-CNN	Correctes	365	50
2-CNN	Fausses	44	145

p-value = 0.38

TABLE 4.8 – Test de McNemar entre 2-CNN-REG et 2-CNN

		2-CNN-REG Correctes	2-CNN-REG Fausses
1-CNN-REG	Correctes	373	33
1-CNN-REG	Fausses	36	62

p-value = 0.13

TABLE 4.9 – Test de McNemar entre 2-CNN-REG et 1-CNN-REG

		2-CNN Correctes	2-CNN Fausses
1-CNN-REG	Correctes	363	43
1-CNN-REG	Fausses	52	46

p-value = 0.85

TABLE 4.10 – Test de McNemar entre 2-CNN-REG et 1-CNN-REG

Comme nous pouvons constater grâce aux tables 4.8, 4.9 et 4.10, les trois modèles sont équivalents. Pour cette raison, nous avons décidé de garder le modèle ayant la plus haute précision.

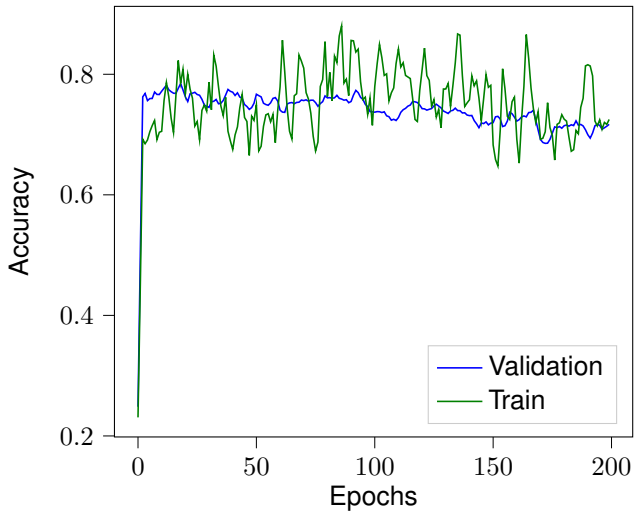


FIGURE 4.6 – Précision sur le jeu de validation pour meilleur CNN

		1-CNN-REG Correctes	1-CNN-REG Fausses
RNN-static	Correctes	210	53
RNN-static	Fausses	196	45

p-value = 82.12

TABLE 4.11 – 1-CNN-REG vs RNN-static

Après ce dernier test de McNemar entre le meilleur RNN et le meilleur CNN, les deux modèles se valent. Le résultat de la p-value nous semble surprenant mais logique après analyse des valeurs du tableau de contingence - Table 4.11. Malgré cela, nous aurions choisi le meilleur CNN comme modèle final.

5 Améliorations

Dans cette section nous allons énumérer certains éléments que nous aurions pu améliorer. En effet, puisque c'était notre premier projet de *data science*, nous avons commis certaines erreurs, malgré les conseil de notre professeur responsable.

Ces erreurs ou difficultés sont :

- Trop de focalisation sur les architectures des modèles
- Données complexes
- Aligner les markers
- Génération de données avec un simulateur
- Complexification des modèles
- Meilleur partage pour la cross validation

5.1 Trop de focalisation sur les architectures des modèles

Ayant une formation et étant intéréssés par l'informatique de gestion, nous nous sommes lancés trop vite dans le code. En effet, le processus de réalisation d'un projet informatique est souvent définit en trois étapes :

1. Compréhension des besoins du mandant
2. Réalisation d'un modèle UML représentant les besoins
3. Implémentation des modèles dans un language informatique adapté

Ayant cette optique en tête, nous avons cru comprendre les besoins du mandant mais en réalité nous avons sauté une étape essentielle : la compréhension des données.

Sans comprendre les données correctement, les modèles et les inputs que nous avons utilisés ne pouvaient qu'être bancales. Aussi, nous nous sommes souvent retrouvés à devoir refaire une partie du code pour améliorer nos performances alors qu'une meilleure compréhension des nos inputs nous aurait mis sur la bonne voie plus rapidement.

5.2 Données complexes

Les données utilisées durant ce projet avaient un format spécifique : c3d. Cette spécification nous a posé quelques difficultés dans le sens où en plus de devoir comprendre les données, nous avons dû apprendre à utiliser ce type de fichier. Heureusement, les assistants du laboratoire de *data science* travaillant sur ce projet depuis des années, ont pu nous aider en nous fournissant du code qui pouvait extraire les données facilement. Malgré cela, il était toujours un défi pour nous de manipuler les données correctement.

5.3 Aligner les markers

Comme mentionné dans la section 1.3, les données sont récoltées dans un laboratoire de kinésiologie aux HUG. Puisque de vraies personnes sont à la source de la génération des données et la démarche est un élément caractéristique à chaque individu, de la variabilité est introduite au début des cycles de démarche. Par exemple, une personne peut commencer à marcher avec le pied gauche ou le pied droit. Pour améliorer les prédictions, nous aurions pu aligner tous les markers pour que tout le monde commence en même temps. Dans la Figure 5.1, nous pouvons voir deux patients qui possèdent la même pathologie mais qui commencent à marcher de manière très différente. Ayant des courbes alignées comme dans la Figure 5.2, pourrait permettre aux modèles entraînés de mieux les approximer, et par conséquent, être plus précis.

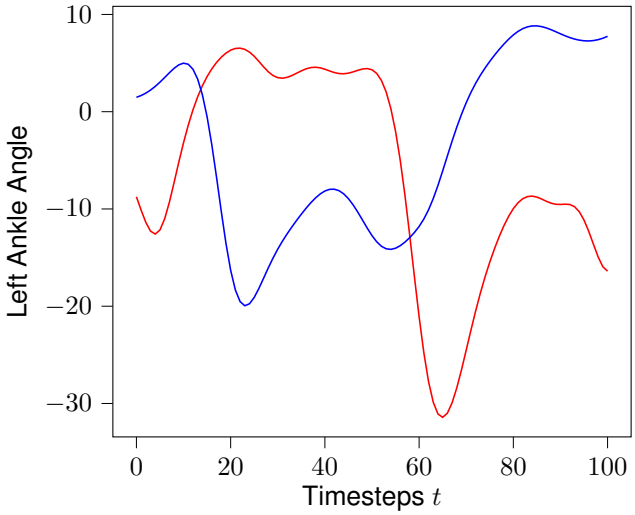


FIGURE 5.1 – Exemple de markers non alignés

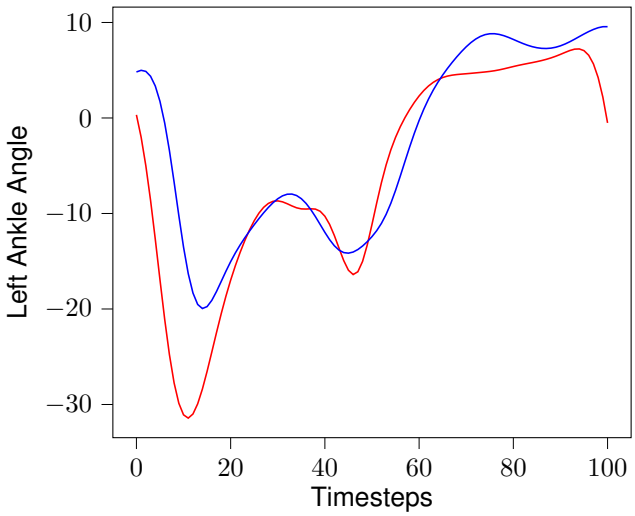


FIGURE 5.2 – Exemple de markers alignés

5.4 Génération de données avec un simulateur

L'une des difficultés que nous avons rencontrées durant ce projet est la disproportion des classes présentes dans le jeu de données. En effet, comme vu dans section 2.1, la classe CP est beaucoup plus grande que le reste des classes.

Pour contrer à cela, nous aurions pu générer des données à l'aide d'un simulateur. En effet, Charalambous et al. a créé un papier permettant d'atteindre ce même objectif. Le problème est qu'il faut déjà avoir correctement approximé la distribution de la démarche des patients.

5.5 Complexification des modèles

Lors de la phase de préprocessing, le groupe s'est aperçu du manque de données. En effet, pour entraîner des réseaux de neurones il faut une énorme quantité de données alors que nous n'en avions que quelques centaines d'instances à disposition.

Pour cette raison, nous avons essayé d'ajouter le maximum de données possibles à nos modèles : ajouter tous les markers possibles, angles et ce, dans les trois axes (x, y, z).

Puisque le modèle possède déjà peu de données, nous aurions dû essayer de choisir seulement les meilleurs capteurs pour que nos modèles statistiques soient moins complexes.

5.6 Meilleur partage pour la cross validation

Comme nous l'avons vu dans la section 2.1, certains patients reviennent plusieurs fois dans le même centre de kinésiologie. Lors de la création de nos jeux d'entraînement, validation et de test, nous aurions dû s'assurer qu'un même patient n'est pas présent dans plus qu'un ensemble de données.

De plus, nous aurions dû vérifier que les classes sont distribuées correctement. Par exemple, une classe "Healthy" n'est peut-être disponible que dans l'ensemble d'entraînement, réduisant ainsi la précision du modèle utilisé.

6 Conclusion

Au cours de cette étude, nous avons pu comparer l'efficacité de différents modèles d'apprentissage automatique dans la tâche complexe de classification de patients souffrant de troubles de la motricité dans le but d'aider les médecins à établir des diagnostics, pour en améliorer la condition physique de ces personnes.

Il a fallu dans un premier temps prendre connaissance des différentes données qui ont été mise à notre disposition et d'en comprendre les relations, pour en faire le tri et de ne sélectionner que les données qui contribueraient aux bonnes performances de nos modèles. Pour chacun de ces modèles, il nous aura fallu en comprendre les différentes architectures et de les tester avec différents hyperparamètres pour atteindre un degré de précision maximum.

Nous avons évalué la performance des trois architectures suivantes : notre base de référence, les forêts d'arbres aléatoires, les réseaux de neurones récurrents ainsi que les réseaux de neurones convolutifs. La précision obtenu avec notre base de référence est de 44,39%, un résultat qui démontre que malgré le nombre d'arbres utilisés, le problème de classification demeure trop complexe pour ce type d'architecture. Les résultats obtenus avec les réseaux de neurones récurrents sont décevants en vue de nos attentes, un résultat de 51% pour un réseau qui se spécialise dans les séries temporelles. Finalement, les réseaux convolutifs, utilisé dans la reconnaissance d'images est un algorithme qui a fait ses preuves en ayant obtenu une précision de 83%, considérablement plus que les deux méthodes précédentes.

Le domaine de la simulation neuromécanique et du Machine Learning sont des domaines complexes qui recèlent encore de l'exploration, mais la recherche dans ce domaine ne fait que de s'intensifier et avec cela vient le progrès et nous contents d'avoir pu y contribuer.

Bibliographie

[1] E. A. Walle. Infant Social Development across the Transition from Crawling to Walking. *Frontiers in Psychology*, 7, June 2016. ISSN 1664-1078. doi : 10.3389/fpsyg.2016.00960.

[2] S. R. Simon. Quantification of human motion : Gait analysis—benefits and limitations to its application to clinical problems. *Journal of Biomechanics*, 37(12) :1869–1880, Dec. 2004. ISSN 0021-9290. doi : 10.1016/j.jbiomech.2004.02.047.

[3] J. Taborri, E. Palermo, S. Rossi, and P. Cappa. Gait Partitioning Methods : A Systematic Review. *Sensors (Basel, Switzerland)*, 16(1), Jan. 2016. ISSN 1424-8220. doi : 10.3390/s16010066.

[4] S. Hong and E. Kim. A New Automatic Gait Cycle Partitioning Method and Its Application to Human Identification. *International Journal of Fuzzy Logic and Intelligent Systems*, 17(2) :51–57, June 2017. ISSN 1598-2645, 2093-744X. doi : 10.5391/IJFIS.2017.17.2.51.

[5] M. W. Whittle. *Gait Analysis : An Introduction*. Butterworth-Heinemann, May 2014. ISBN 978-1-4831-8373-2.

[6] Z. O. Abu-Faraj, G. F. Harris, P. A. Smith, and S. Hassani. Human gait and Clinical Movement Analysis. In *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–34. John Wiley & Sons, Inc., Hoboken, NJ, USA, Dec. 2015. ISBN 978-0-471-34608-1. doi : 10.1002/047134608X.W6606.pub2.

[7] protokinetics. Phases of the Gait Cycle : Gait Analysis » ProtoKinetics. https ://www.protokinetics.com/2018/11/28/understanding-phases-of-the-gait-cycle/, 2019.

[8] A. Muro-de-la-Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla. Gait Analysis Methods : An Overview of Wearable and Non-Wearable Systems, Highlighting Clinical Applications. *Sensors*, 14 (2) :3362–3394, Feb. 2014. doi : 10.3390/s140203362.

[9] D. Tarni. Wearable sensors used for human gait analysis. page 10, 2016.

[10] A. M. Howell, T. Kobayashi, H. A. Hayes, K. B. Foreman, and S. J. M. Bamberg. Kinetic Gait Analysis Using a Low-Cost Insole. *IEEE Transactions on Biomedical Engineering*, 60(12) :3284–3290, Dec. 2013. doi : 10.1109/TBME.2013.2250972.

[11] biometricsltd. Twin-Axis Electrogoniometers (Goniometer) for Joint Movement Analysis. http ://www.biometricsltd.com/goniometer.htm, 2019.

[12] Technomed. EMG | Technomed. https ://technomed.nl/application-area/emg, 2019.

[13] R. R. Jensen, R. R. Paulsen, and R. Larsen. Analyzing Gait Using a Time-of-Flight Camera. In A.-B. Salberg, J. Y. Hardeberg, and R. Jenssen, editors, *Image Analysis*, Lecture Notes in Computer Science, pages 21–30. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02230-2.

[14] M. O. Derawi, H. Ali, and F. A. Cheikh. *Gait Recognition Using Time-of-Flight Sensor*. Gesellschaft für Informatik e.V., 2011. ISBN 978-3-88579-285-7.

[15] C. Valente. Scanner 3D à lumière structurée, Feb. 2019.

[16] L. Middleton, A. Buss, A. Bazin, and M. Nixon. A Floor Sensor System for Gait Recognition. In *Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'05)*, pages 171–176, Buffalo, NY, USA, 2005. IEEE. ISBN 978-0-7695-2475-7. doi : 10.1109/AUTOID.2005.2.

[17] T. Chau. A review of analytical techniques for gait data. Part 2 : Neural network and wavelet methods - ScienceDirect. https ://reader.elsevier.com/reader/sd/pii/S0966636200000953 ?token=A4C700A9F754E2C661A2366E2F476ECF440351E3FAC1E9487EEF70E0993F7B4C90CD35DC00B610096 2001.

[18] R. Shiavi and P. Griffin. Representing and clustering electromyographic gait patterns with multivariate techniques. *Medical &amp ; Biological Engineering &amp ; Computing*, 19(5) :605–611, 1981. ISSN 0140-0118.

[19] P. Loslever, F. X. Lepoutre, A. Kebab, and H. Sayarh. Descriptive multidimensional statistical methods for analysing signals in a multifactorial biomedical database. *Medical and Biological Engineering and Computing*, 34(1) :13–20, Jan. 1996. ISSN 1741-0444. doi : 10.1007/BF02637017.

[20] T. Bock. What is Hierarchical Clustering ? | Displayr.com, 2019.

[21] C. M. O'Connor, S. K. Thorpe, M. J. O'Malley, and C. L. Vaughan. Automatic detection of gait events using kinematic data. *Gait & Posture*, 25(3) :469–474, Mar. 2007. ISSN 0966-6362. doi : 10.1016/j.gaitpost.2006.05.016.

[22] J. A. Zeni, J. G. Richards, and J. S. Higginson. Two simple methods for determining gait events during treadmill and overground walking using kinematic data. *Gait & Posture*, 27(4) :710–714, May 2008. ISSN 0966-6362. doi : 10.1016/j.gaitpost.2007.07.007.

[23] M. Skelly and H. Chizeck. Real-time gait event detection for paraplegic FES walking. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(1) :59–68, Mar. 2001. ISSN 1558-0210. doi : 10.1109/7333.918277.

[24] S. J. Preece, L. P. Kenney, M. J. Major, T. Dias, E. Lay, and B. T. Fernandes. Automatic identification of gait events using an instrumented sock. *Journal of NeuroEngineering and Rehabilitation*, 8(1) : 32, May 2011. ISSN 1743-0003. doi : 10.1186/1743-0003-8-32.

[25] S. Ghoussayni, C. Stevens, S. Durham, and D. Ewins. Assessment and validation of a simple automated method for the detection of gait events and intervals. *Gait & Posture*, 20(3) :266–272, Dec. 2004. ISSN 0966-6362. doi : 10.1016/j.gaitpost.2003.10.001.

[26] L. Kidzinski, S. Delp, and M. Schwartz. Automatic real-time gait event detection in children using deep neural networks. *PLOS ONE*, 14(1) :e0211466, Jan. 2019. ISSN 1932-6203. doi : 10.1371/journal.pone.0211466.

[27] F. Tafazzoli and R. Safabakhsh. Model-based human gait recognition using leg and arm movements. *Engineering Applications of Artificial Intelligence*, 23(8) :1237–1246, Dec. 2010. ISSN 0952-1976. doi : 10.1016/j.engappai.2010.07.004.

[28] I. Bouchrika and M. S. Nixon. People detection and recognition using gait for automated visual surveillance. pages 576–581, Jan. 2006. doi : 10.1049/ic:20060364.

[29] C. Yam, M. S. Nixon, and J. N. Carter. Automated person recognition by walking and running via model-based approaches. *Pattern Recognition*, 37(5) :1057–1072, May 2004. ISSN 0031-3203. doi : 10.1016/j.patcog.2003.09.012.

[30] D. Cunado, M. S. Nixon, and J. N. Carter. Automatic extraction and description of human gait models for recognition purposes. *Computer Vision and Image Understanding*, 90(1) :1–41, Apr. 2003. ISSN 1077-3142. doi : 10.1016/S1077-3142(03)00008-0.

[31] C. BenAbdelkader, R. Cutler, and L. Davis. Stride and cadence as a biometric in automatic person identification and verification. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 372–377, May 2002. doi : 10.1109/AFGR.2002.1004182.

[32] J. T. Geiger, M. Kneißl, B. W. Schuller, and G. Rigoll. Acoustic Gait-based Person Identification using Hidden Markov Models. In *Proceedings of the 2014 Workshop on Mapping Personality Traits Challenge and Workshop*, MAPTRAITS '14, pages 25–30, Istanbul, Turkey, Nov. 2014. Association for Computing Machinery. ISBN 978-1-4503-3956-8. doi : 10.1145/2668024.2668027.

[33] F. Yang, L. Yuncheng, and L. Jiebo. Learning effective Gait features using LSTM. https ://www.researchgate.net/publication/316444033_Learning_effective_Gait_features_using_LSTM, 2016.

[34] M. Derlatka and M. Bogdan. Ensemble kNN classifiers for human gait recognition based on ground reaction forces. In *2015 8th International Conference on Human System Interaction (HSI)*, pages 88–93, June 2015. doi : 10.1109/HSI.2015.7170648.

[35] B. S. Vijay, R. Manish, and N. G.C. Biometric gait identification based on a multilayer perceptron | Elsevier Enhanced Reader. https ://reader.elsevier.com/reader/sd/pii/S0921889014002632, 2014.

[36] Z. Wei, L. Fenglin, W. Qinghui, W. Ying, M. Limin, and Z. Yu. Parkinson's disease classification using gait analysis via deterministic learning | Elsevier Enhanced Reader. https ://reader.elsevier.com/reader/sd/pii/S0304394016307261 ?token=8D85C539BB80B1764CAB8C33B17473D1790BECCD573CCB67B9D0DB8ED7A03EE7BA42A675B5023D2 2016.

[37] D. Sanjeev Kumar, B. Ajit Kumar, D. Satchidananda, and C. Sung-Bae. (1) (PDF) Radial basis function neural networks : A topical state-of-the-art survey. https ://www.researchgate.net/publication/303424749_Radial_basis_function_neural_networks_A_topical_state-of-the-art_survey, 2016.

[38] G. Seif. Why and How to do Cross Validation for Machine Learning. https ://towardsdatascience.com/why-and-how-to-do-cross-validation-for-machine-learning-d5bd7e60c189, May 2019.

[39] E. O. Brigham and R. E. Morrow. The fast Fourier transform. *IEEE Spectrum*, 4(12) :63–70, Dec. 1967. doi : 10.1109/MSPEC.1967.5217220.

[40] S. H. Holzreiter and M. E. Köhle. Assessment of gait patterns using neural networks. *Journal of Biomechanics*, 26(6) :645–651, June 1993. ISSN 0021-9290. doi : 10.1016/0021-9290(93) 90028-D.

[41] J. Barton and A. Lees. Development of a connectionist expert system to identify foot problems based on under-foot pressure patterns. *Clinical Biomechanics*, 10(7) :385–391, Oct. 1995. ISSN 0268-0033. doi : 10.1016/0268-0033(95)00015-D.

[42] M. J. O'Malley, M. F. Abel, D. L. Damiano, and C. L. Vaughan. Fuzzy clustering of children with cerebral palsy based on temporal-distance gait parameters. *IEEE Transactions on Rehabilitation Engineering*, 5(4) :300–309, Dec. 1997. doi : 10.1109/86.650282.

[43] T. Chau. A review of analytical techniques for gait data. Part 1 : Fuzzy, statistical and fractal methods - ScienceDirect. https ://www.sciencedirect.com/science/article/pii/S0966636200000941, 2001.

[44] R. Bellazzi and B. Zupan. Predictive data mining in clinical medicine : Current issues and guidelines. *International journal of medical informatics*, 77 :81–97, Mar. 2008. doi : 10.1016/j.ijmedinf.2006.11. 006.

[45] N. Lavrač. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16 (1) :3–23, May 1999. ISSN 0933-3657. doi : 10.1016/S0933-3657(98)00062-1.

[46] M. J. Pazzani, S. Mani, and W. R. Shankle. Acceptance of Rules Generated by Machine Learning among Medical Experts. *Methods of Information in Medicine*, 40(05) :380–385, 2001. ISSN 0026-1270, 2511-705X. doi : 10.1055/s-0038-1634196.

[47] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You ?" : Explaining the Predictions of Any Classifier. *arXiv :1602.04938 [cs, stat]*, Feb. 2016.

[48] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. Machine Learning, Neural and Statistical Classification. page 298, Feb. 1994.

[49] M. Turek. Explainable Artificial Intelligence. https ://www.darpa.mil/program/explainable-artificial-intelligence, 2017.

[50] T. Jiang and A. E. Keating. AVID : An integrative framework for discovering functional relationship among proteins. https ://www.researchgate.net/publication/7811920_AVID_An_integrative_framework_for_discovering_functional_re June 2005.

[51] B. Ustun and C. Rudin. Supersparse Linear Integer Models for Optimized Medical Scoring Systems. *Machine Learning*, 102(3) :349–391, Mar. 2016. ISSN 0885-6125, 1573-0565. doi : 10.1007/s10994-015-5528-6.

[52] F. Wang and C. Rudin. Falling Rule Lists. *arXiv :1411.5899 [cs]*, Nov. 2014.

[53] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and Bayesian analysis : Building a better stroke prediction model. *The Annals of Applied Statistics*, 9 (3) :1350–1371, Sept. 2015. ISSN 1932-6157. doi : 10.1214/15-AOAS848.

[54] A. A. Freitas and A. A. Freitas. Comprehensible Classification Models – a position paper. 15(1) : 10, Mar. 2014.

[55] M. Craven and J. W. Shavlik. Extracting Tree-Structured Representations of Trained Networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 24–30. MIT Press, 1996.

[56] J. Krause, A. Perer, and K. Ng. Interacting with Predictions : Visual Inspection of Black-box Machine Learning Models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 5686–5697, Santa Clara, California, USA, 2016. ACM Press. ISBN 978-1-4503-3362-7. doi : 10.1145/2858036.2858529.

[57] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, and K. Hansen. How to Explain Individual Classification Decisions. page 29, Dec. 2009.

[58] M. T. Ribeiro, S. Singh, and C. Guestrin. Model-Agnostic Interpretability of Machine Learning. *arXiv :1606.05386 [cs, stat]*, June 2016.

[59] S. Kaufman, S. Rosset, and C. Perlich. Leakage in Data Mining : Formulation, Detection, and Avoidance. page 8, Aug. 2011.

[60] F. Horst, S. Lapuschkin, W. Samek, K.-R. Müller, and W. I. Schöllhorn. Explaining the unique nature of individual gait patterns with deep learning. *Scientific Reports*, 9(1) :1–13, Feb. 2019. ISSN 2045-2322. doi : 10.1038/s41598-019-38748-8.

[61] E. Viel. *La marche humaine, la course et le saut - Biomécanique, eplorations, normes et dysfonctionnement*. ELSEVIER-MASSON, Mar. 2000. ISBN 978-2-225-83640-4.

[62] C3D. C3D.ORG - The biomechanics standard file format. https ://www.c3d.org/, 2020.

[63] W. Koehrsen. Random Forest Simple Explanation. https ://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d, Dec. 2017.

[64] K. Matsuki, V. Kuperman, and J. A. Van Dyke. The Random Forests statistical technique : An examination of its value for the study of reading. *Scientific studies of reading : the official journal of the Society for the Scientific Study of Reading*, 20(1) :20–33, 2016. ISSN 1088-8438. doi : 10.1080/10888438.2015.1107073.

[65] C. Rolah. Understanding LSTM Networks. https ://colah.github.io/posts/2015-08-Understanding-LSTMs/, Aug. 2015.

[66] M. S. Mady. Chapter 10 : DeepNLP - Recurrent Neural Networks with Math. https ://medium.com/deep-math-machine-learning-ai/chapter-10-deepnlp-recurrent-neural-networks-with-math-c4a6846a50a2, Jan. 2018.

[67] A.-I. Marinescu. Bach 2.0 - generating classical music using recurrent neural networks. *Procedia Computer Science*, 159 :117–124, Jan. 2019. ISSN 1877-0509. doi : 10.1016/j.procs.2019.09.166.

[68] M. Nguyen. Illustrated Guide to LSTM's and GRU's : A step by step explanation. https ://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, July 2019.

[69] M. Dwarampudi and N. V. S. Reddy. Effects of padding on LSTMs and CNNs. *arXiv :1903.07288 [cs, stat]*, Mar. 2019.

[70] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes. Deeper Networks for Pavement Crack Detection. July 2017. doi : 10.22260/ISARC2017/0066.

[71] Haute école de gestion de Genève. La HEG-Genève célèbre ses 20 ans d'existence ! https ://www.hesge.ch/heg/actualites/2018/heg-geneve-celebre-ses-20-ans-dexistence, Sept. 2018.

[72] A. Kalousis. CNNs Building Blocks, Oct. 2019.

[73] Stanford CS231n. CS231n Convolutional Neural Networks for Visual Recognition. http ://cs231n.github.io/convolutional-networks/, 2020.

[74] J. Brownlee. How to Calculate McNemar's Test to Compare Two Machine Learning Classifiers, July 2018.

[75] S. Raschka. Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. *Psychometrika*, 13(3) :185–187, 2019. ISSN 0033-3123, 1860-0980. doi : 10.1007/BF02289261.