h e g

Haute école de gestion
Genève

Automatic labelling for motion capture

Travail de master réalisé par :

François DUVERNAY

Sous la direction de :

Alexandros KALOUSIS, Professeur HES

Genève, le 15 août 2022

Master en Sciences de l'information

Haute École de Gestion de Genève (HEG-GE)

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Remerciements

Le travail présenté dans ce mémoire a été proposé dans le cadre du Master en Sciences de l'information de la Haute école de gestion de Genève. Le professeur responsable de ce travail est Monsieur Alexandros Kalousis qui dirige les cours de Machine Learning de ce cursus de Master.

Je tiens tout d'abord à remercier Monsieur Alexandros Kalousis pour avoir suivi ce travail tout du long et pour toute l'aide qu'il m'a apporté quant à la direction du projet.

Je tiens également à remercier le mandant du projet, le Dr Stephane Armand et toute son équipe qui ont également participé à suivre le développement et qui sont restés présents pour répondre à toutes mes interrogations quant au cadre du projet et les détails techniques. Merci à eux de m'avoir fourni tous les outils et les données nécessaires ainsi que leur réactivité.

Enfin, je remercie Joao Candido Ramos et Pablo Strasser qui ont participé à toutes les réunions du projet et qui ont été disponibles pour répondre à toutes mes questions dans de brefs délais.

Résumé

Dans le cadre d'un travail de Master en Science de l'information, ce projet consiste à créer un algorithme capable d'étiqueter automatiquement des marqueurs après des captures vidéo en trois dimensions de patients atteints de troubles de la marche. Le but du projet est de remplacer un travail manuel par un réseau de neurones récurrents (plus précisément un LSTM) qui permet de retrouver quel est le label pour chaque marqueur présent dans la capture.

Dans cette étude, une analyse de l'existant fait parvenir plusieurs modèles de Deep Learning déjà utilisé avec leurs architectures complètes. La première partie consiste à approfondir et comprendre ces architectures afin de choisir un modèle convenable.

Le projet qui en découle, s'inspire de son environnement et de l'existant pour produire un modèle capable d'effectuer des prédictions. Le modèle doit s'adapté aux données fournies et produire un résultat aussi proche que possible des solutions existantes. Les hypothèses qui découlent de cette expérience permettent d'avoir un angle d'attaque pour améliorer les résultats.

Mots-clefs : Apprentissage automatique, Deep learning, LSTM, Motion Capture, Troubles de la marche

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

L'apprentissage automatique devient depuis quelques années une partie intégrante de notre vie commune. Cette technologie a pu imploser ces dernières années grâce aux puissances de calculs des ordinateurs récents qui montent exponentiellement. Ces calculs permettent de trouver des solutions pour de nombreux problèmes et le Deep learning a maintenant fait ses preuves notamment dans le domaine de la médecine. Parmi les nombreuses utilisations de ces algorithmes, revient souvent celle de remplacer des tâches manuelles répétitives que doit faire une personne et qui prend beaucoup de son temps de travail.

1.1 Cadre du projet

Ce projet a été proposé par un laboratoire des Hôpitaux Universitaires de Genève (HUG). Ce laboratoire s'occupe d'analyser la marche de patients atteints de certaines pathologies afin de faire des analyses pour trouver un traitement qui améliorerait la qualité de vie de personnes atteintes de troubles du mouvement.

Une partie de ce laboratoire consiste à capturer la marche d'une personne pour en extraire des données utilisables pour des analyses. La méthode de capture utilisée est la capture des mouvements plus connue sous son nom anglais « Motion Capture ». Cette technologie consiste à placer des marqueurs (capteurs) à des endroits clés sur le corps d'un sujet, ces marqueurs sont généralement de petites boules blanches qui permettent de refléter la lumière. Les capteurs sont ensuite détectés par plusieurs caméras infrarouges qui permettent de reproduire leurs mouvements sur un espace en trois dimensions. Cette méthode est notamment connue pour être utilisée dans le monde du cinéma ou du jeu vidéo lorsqu'il faut reproduire des mouvements d'un être humain dans un monde fictif.

Ce travail est réalisé dans le cadre du Master en Science de l'information de la Haute École de Gestion de Genève en tant que travail de master pour l'obtention du titre de Data Scientist. C'est un travail individuel réalisé sur la durée d'un semestre académique.

1.2 Problématique

Avant la capture, le laboratoire doit placer les marqueurs aux mêmes endroits pour chaque patient. Ils ont une liste de marqueurs avec leur étiquette (label) qu'ils doivent respecter et placer au bon endroit afin de pouvoir analyser correctement les données équitablement entre tous les différents patients. Les marqueurs sont des objets qui reflètent la lumière et non des appareils électroniques, il est donc nécessaire d'effectuer un traitement après la session de capture pour identifier quel label appartient à quel marqueur pendant la capture. À ce jour, le laboratoire ne possède pas de solution stable pour effectuer cette identification complètement automatiquement et cela requiert un traitement manuel de la part des collaborateurs qui peut se montrer long et fastidieux.

1.3 Objectif

Ce travail a pour but d'améliorer la qualité de vie des chercheurs en leur produisant un algorithme capable de déterminer quel label correspond à quel marqueur après une capture. En utilisant les données du laboratoire déjà enregistrées et selon une liste de labels à attribuer, l'algorithme doit être capable pairer les bons labels avec les bons marqueurs en ayant la précision la plus élevée possible.

Dans un premier temps, il faut considérer les solutions existantes et analyser leur méthode ainsi que leurs résultats. S'il existe déjà des algorithmes performants dans ce domaine, alors il est nettement possible que la même méthode fonctionne en appliquant nos données.

Une fois la méthode choisie, il faut comprendre et traiter les données de sorte qu'elles soient utilisables et cohérentes pour passer dans l'algorithme. Il est nécessaire de prendre le temps de les comprendre et de définir quels traitements doivent être appliqués pour améliorer la performance de notre algorithme.

Vient ensuite la création de l'algorithme, il faut trouver le plus adapté à nos besoins et à nos données pour ensuite le construire et l'appliquer. Il a été convenu avec la direction du projet, de commencer sur une base avec des données où les trajectoires sont similaires pour avoir une base solide qui pourra être appliquée et adaptée sur les données de patients ayant des trajectoires bien différentes.

2. État de l'art

Ce sujet a déjà été mentionné et traité plusieurs fois. En effet l'utilisation d'un tel algorithme peut faire un gain de temps considérable dans certains corps de métier, notamment l'animation où il peut y avoir un nombre de marqueurs conséquent et des trajectoires très variées.

L'idée d'utiliser les algorithmes d'apprentissage automatique a déjà été abordée en 2005 dans un article dédié (Holzreiter 2005). Malheureusement, cet article ne présente pas de résultats ni de code. Ils expliquent eux-mêmes que leur réseau de neurones utilisé reste très simple et qu'il pourrait être amélioré. En effet, dans le début des années 2000, l'utilisation des algorithmes d'apprentissage automatique n'était pas aussi repandue et développée qu'aujourd'hui notamment à cause des performances du matériel informatique de l'époque.

Bien entendu dans les entreprises expertes du domaine de la capture du mouvement, ils possèdent des solutions pour cette problématique. De grosses entreprises telles que Vicon sont spécialisées dans le cinéma ou le jeu vidéo et ont des attentes élevées en termes de technologies pour se démarquer. Malheureusement ces solutions ne sont pas disponibles publiquement.

Ce qui va plutôt retenir notre attention sont des solutions récentes et open sources qui découlent de cette idée d'automatiser ce processus. Il en existe plusieurs qui montrent des résultats très intéressants.

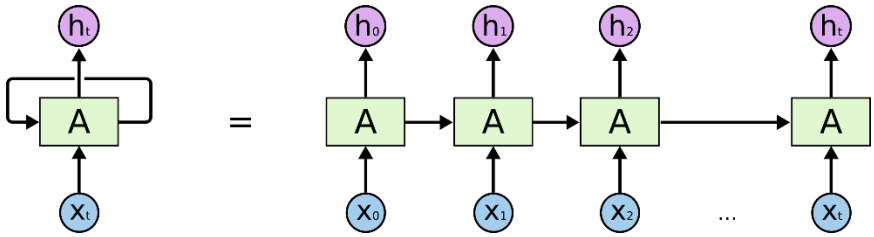
2.1 LSTM

Une approche sensée du problème est de le voir comme une séquence de points que l'on essaye d'interpréter pour en trouver la bonne valeur de sortie. Il est donc tout à fait légitime de faire appel aux RNN pour obtenir un résultat cohérent.

2.1.1 RNN

Les RNN (Réseau de neurones récurrents) ont été popularisés pour leurs performances dans des problématiques de traitement d'informations séquentielles et temporelles. Le principe consiste à répéter un traitement au long de la séquence tout en gardant en mémoire tout ce qu'il a fait à chaque traitement.

Figure 1 : Architecture des RNN



(Understanding LSTM Networks -- colah's blog 2015)

Le réseau présenté dans la Figure 1, prend comme entrée X où X_t représente un élément de la séquence X. Les données d'entrées passent ensuite dans une hidden layer A et va donner comme sortie le hidden state h_t qui peut être utilisé à n'importe quel moment de la séquence comme une prédiction. Les cellules du RNN se transmettent au fur et à mesure le hidden state des cellules précédentes, ce qui agit comme une mémoire du réseau et communique des informations pour la suite de la séquence.

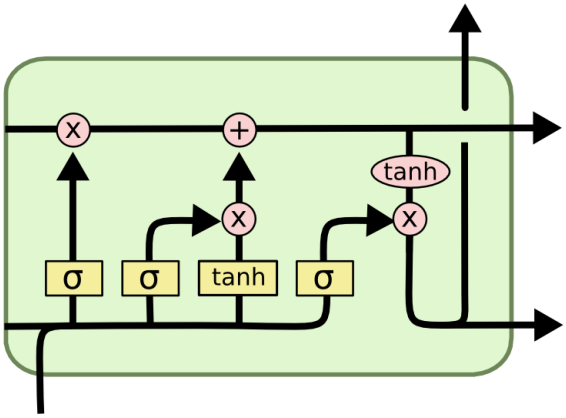
La structure du RNN classique présente cependant un problème important. En effet, lorsque la séquence devient vraiment longue, certaines valeurs peuvent n'avoir plus aucun poids et nous faire tomber dans le problème de disparition du gradient. Ce problème peut arriver rapidement dans le cadre d'un RNN classique et va impacter négativement nos résultats, rendant l'entraînement de celui-ci particulièrement difficile et instable.

2.1.2 Principe LSTM

Le réseau LSTM (Long Short-Term Memory) a été conçu pour pallier au problème principal des RNN. On obtient un moyen de retenir uniquement les éléments précédents de la séquence qui ont un réel impact sur la décision. Grâce à ce type de réseau, la longueur de la séquence a beaucoup moins de conséquences et il devient bien plus facile d'effectuer un entraînement. Aujourd'hui, on parle rarement des réseaux RNN sans mentionner les LSTM qui sont devenus une des nouvelles bases pour utiliser des réseaux de neurones récurrents.

2.1.2.1 Architecture

Figure 2 : Architecture d'une cellule LSTM



(Understanding LSTM Networks -- colah's blog 2015)

Une cellule d'un LSTM (Figure 2) a pour propriété de contenir à la fois l'état de la cellule et de différentes portes. L'état de la cellule va contenir toutes les informations précédentes retenues par le réseau et va être modifié par le résultat des informations qu'auront fait passer les différentes portes de la cellule. La cellule à 3 entrées, une pour les données d'entrée que l'on donne à notre réseau comme pour le RNN, une autre pour le hidden state qui contient les précédentes informations calculées par le réseau et enfin une nouvelle entrée par rapport au RNN classique qui sert de mémoire de la cellule. Cette mémoire peut être altérée à volonté par la cellule pendant son traitement pour ainsi sélectionner ce qui importe ou non et définir l'état de la cellule.

L'ajout des portes permet à la cellule d'altérer la mémoire générale du réseau en décidant de ce qui doit être ajouté ou oublié. Cette étape permet d'accorder plus d'importance aux éléments qui ont de la valeur pour la prédiction et permet également de ne pas surcharger le réseau avec des informations qui ne lui sont pas utiles. Trois de ces portes appliquent la même opération (un sigmoïde), bien que ce soit la même opération, elles n'ont pas la même utilité et n'appliquent pas les mêmes poids. Ces portes ont chacune un nom (dans l'ordre de traitement) : Forget Gate, Input Gate et Output Gate.

La Forget Gate dicte à la mémoire de la cellule quelles informations elle doit garder ou rejeter dépendant du résultat de sa fonction. Cette opération va permettre de faire remonter les valeurs proches de 1 (qui ont donc une grande importance) et d'oublier celles qui au contraire sont proches de 0.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Ensuite vient l'Input Gate qui va décider quelle quantité de nouvelles informations devrait être ajoutée à la mémoire de la cellule. On a donc le vecteur C qui contient les propositions de nouvelles informations et, comme pour la porte précédente, une opération sigmoïde qui décide quelles sont les informations importantes parmi ces propositions.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

C'est après ces deux portes que l'on va écrire dans la mémoire de la cellule C_t. On oublie dans un premier temps les informations de l'état de la cellule précédente qui ont été établies comme étant non pertinentes par la cellule actuelle. On ajoute à cela le résultat de l'Input Gate qui nous donne toutes les nouvelles informations jugées utiles.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t}$$

Pour terminer, nous avons l'Output Gate qui va définir le hidden state de la cellule à l'instant t. On passe dans une fonction de tangente hyperbolique la mémoire de notre cellule puis de la même manière que dans les portes précédentes, la cellule va décider quelle quantité d'informations doit sortir de la cellule pour la suite du modèle ou pour la prédiction selon l'instant.

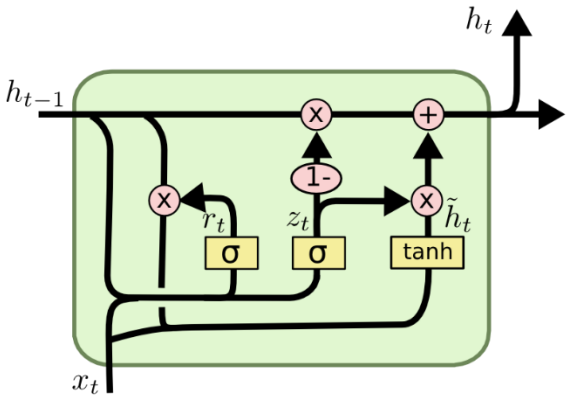
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

2.1.3 GRU

Une autre architecture de RNN populaire est le GRU (Gaited Recurrent Unit). Le mécanisme utilisé est également celui des portes, ce qui rend cette architecture similaire à celle des LSTM. De la même manière que pour les LSTM, il y a une gestion de la mémoire au niveau des cellules (Figure 3). Ces deux structures de cellules ont cependant 2 principales différences. Les GRU n'utilisent pas de stockage de mémoire de la cellule et vont directement appliquer leurs changements sur le hidden state. La seconde différence est qu'il n'y a que deux portes : La Reset Gate, qui décide quelles informations doivent être gardées ou oubliées. Puis, l'Update Gate vient combiner les opérations de l'Input Gate et la Forget Gate que l'on a vue dans les LSTM.

Figure 3 : Architecture d'une cellule GRU



(Understanding LSTM Networks -- colah's blog 2015)

Les deux architectures de cellules qui ont été présentées sont les plus communes et référencées dans les utilisations de RNN. Le choix de la structure dépend entièrement du cas dans lequel elle doit être appliquée. Il est possible qu'une architecture GRU montre de meilleures performances qu'une architecture LSTM.

2.1.4 Article

Dans l'article « Development and Validation of a Deep Learning Algorithm and Open-Source Platform for the Automatic Labelling of Motion Capture Markers » (Clouthier et al. 2021) est présentée une solution open source qui a un but commun à notre objectif. L'algorithme de Deep Learning utilisé dans ce projet est celui des LSTM. Ils utilisent également le Transfer Learning pour affiner leurs résultats au fil du temps, ce qui leur a permis d'obtenir des résultats équivalents à ceux des solutions business.

En plus de proposer une solution, le projet de l'article met à disposition une interface graphique pour faciliter son utilisation et améliorer l'expérience utilisateur. Cette fonctionnalité est non négligeable pour des personnes qui en auraient besoin sans pour autant avoir les connaissances de codage nécessaire.

2.1.4.1 Données

Les données utilisées par ce projet se résument en deux jeux de données. Leur premier jeu de données « A » consiste en 184 participants avec 45 marqueurs placés sur leur corps aux mêmes endroits (même positions anatomique). Ce sont des données réelles qu'ils ont obtenues d'un fournisseur tiers, les données étaient déjà nettoyées et n'avait donc pas de trous ou de marqueurs fantômes dans les trajectoires. Les mouvements que les participants ont reproduits sont divers et variés, on y retrouve de la marche comme de la course ou des sauts. Le second jeu de données « B » compte 18 participants avec 62 marqueurs. Les mouvements sont les mêmes qu'au jeu de données précédent avec quelques ajouts. La différence principale entre ces deux jeux de données est que dans le second il y a de nombreux marqueurs supplémentaires pour ajouter du bruit et rendre la tâche d'apprentissage plus difficile. Le second jeu ayant été séparé en deux versions, l'une qui contient toutes les données brutes et l'autre à qui l'on a retiré manuellement les marqueurs supplémentaires.

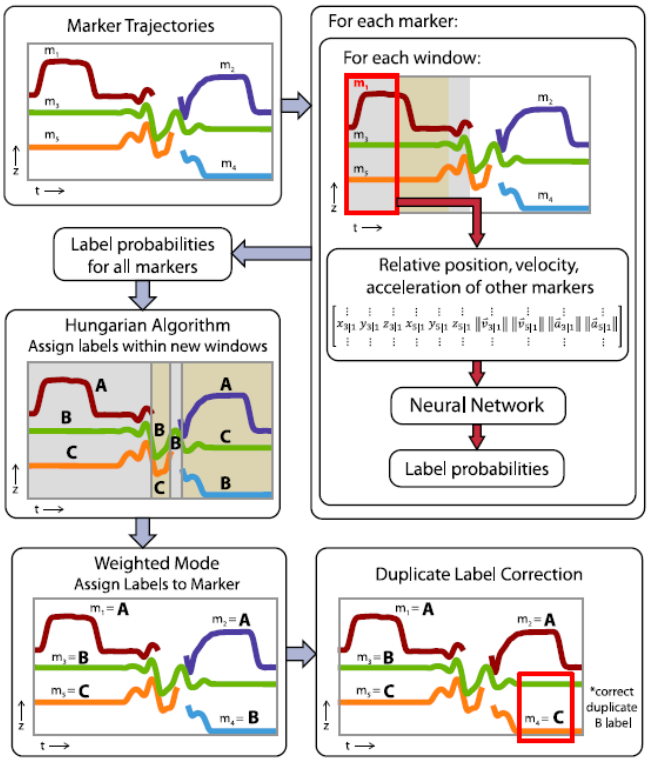
Toutes les données ont été traitées pour que les participants soient face au X positif. Il y a en effet une possibilité que les participants soient tournés dans des sens différents et il est donc nécessaire pour normaliser que tout le monde soit de base dirigée dans la même direction.

Pour ce faire, ils doivent prendre deux marqueurs (dans leur cas ceux placés sur les acromions) pour avoir un repère fixe qui permet de calculer une rotation du corps. Les trous dans les trajectoires de plus de 12 images de long ont été remplis par de l'interpolation linéaire.

Le réseau reçoit en données d'entrée une matrice avec les positions relatives des marqueurs, leur accélération et leur vélocité par fenêtre de 120 images. Les positions relatives sont la distance du marqueur par rapport aux autres marqueurs présents dans la fenêtre.

2.1.4.2 Architecture

Figure 4 : Schéma de l'architecture du projet utilisant un réseau LSTM



(Clouthier et al. 2021)

Dans le schéma d'architecture présenté dans la Figure 4. Plusieurs étapes sont décrites dans des blocs avec dans l'ordre : Les données d'entrées, Le réseau de neurone appliqué (LSTM) pour donner une première prédiction, Une application de l'algorithme Hongrois pour affiner l'assignation de labels, Une application de poids pour appliquer le même label sur une même trajectoire et enfin le dernier bloc qui corrige la duplication de labels.

Pour la partie du processus utilisant un réseau LSTM, plusieurs architectures de modèle ont été testées. Parmi toutes les architectures, celles qui ont montrés le meilleur résultat sont les LSTM et les GRU. Étant donné que ces architectures sont très similaires, il n'est pas étonnant de voir que leur résultat est autant rapproché cependant celle avec les LSTM à obtenu un résultat légèrement supérieur. Cette partie va prendre les données par fenêtre de 120 images que l'on a vues dans la section précédente avec les positions relatives, la vélocité et l'accélération. La sortie de ce réseau de neurones donne par fenêtre, un vecteur qui contient toutes les probabilités pour chaque label.

Ensuite sont générée de nouvelles fenêtres qui sont segmentées cette fois non pas par un nombre fixe d'images, mais par apparition ou disparition d'un nouveau marqueur. Cette segmentation permet d'avoir des trajectoires continues pour chaque fenêtre qui seront

ensuite traitées par l'algorithme hongrois pour assigner le marqueur optimal à chaque trajectoire pour chaque nouvelle fenêtre.

Enfin, une fois les fenêtres traitées, un traitement s'assure que chaque marqueur n'a qu'un seul label. Cette opération permet de relier toutes les fenêtres ensemble en sélectionnant le marqueur le plus probable sur la trajectoire entière de toute la séquence. Finalement, il peut y arriver que deux marqueurs se retrouvent avec le même label attribué, car ils sont plusieurs à avoir le même label le plus probable. Dans ce cas, une dernière vérification a lieu pour corriger en ne gardant que celui qui a la plus haute probabilité d'avoir ce label.

Pour affiner le réseau, une utilisation du Transfer Learning a été utilisée. Cette technique consiste à alimenter l'entraînement du réseau avec de nouvelles données qui ont été prédites correctement. Ainsi, le jeu d'entraînement devient plus fourni et les poids du réseau se mettent à jour pour ajuster davantage la précision. Grâce à cela, ils ont réussi à obtenir une précision finale bien meilleure.

2.1.4.3 Résultats

Tableau 1 : Résultats du projet utilisant un réseau LSTM

Jeu de données de test	Précisions
A	99.6
B (sans marqueurs supplémentaires)	98.8
B (avec marqueurs supplémentaires)	95.6

(Clouthier et al. 2021)

La plus haute précision atteinte est avec le jeu de données A. Comme nous l'avons vu précédemment c'est le jeu de données le plus propre avec des données de qualité qui ne donnent pas de marqueurs supplémentaires. Lorsque l'on utilise le jeu de données B avec des données plus proches du réel, on remarque une baisse minime de la précision sans marqueurs supplémentaires et une baisse plus significative lorsqu'on laisse les marqueurs supplémentaires. Il est normal d'observer ce genre de réaction lorsque les données contiennent plus de bruit, cependant l'altération du résultat reste très correcte et permet de donner des prédictions de bonne qualité.

Dans la globalité des résultats, on observe que ce projet a obtenu de très bonnes précisions. Il est donc capable de labéliser de manière efficace des données propres comme des données contenant du bruit. La structure complexe de leur architecture a permis d'affiner la prédiction et d'en faire un outil qui reproduit des performances d'une solution business. C'est un très bon exemple open source de ce que l'on est capable de faire aujourd'hui.

2.2 Transformers

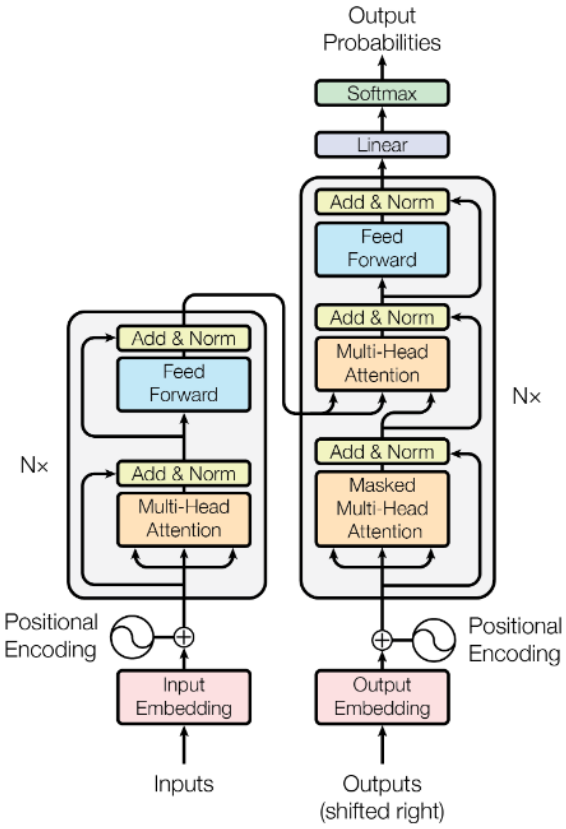
Les Transformers sont une nouvelle architecture qui a fait surface en 2017 grâce à l'article « Attention Is All You Need » (Vaswani et al. 2017) publié par des chercheurs de chez Google. Cette architecture a beaucoup fait parler d'elle, car elle a réussi à surpasser les performances des LSTM et GRU dans le domaine de la traduction alors que ces modèles étaient depuis longtemps établis comme état de l'art de ce genre de problématique séquentielle. Les Transformers ont depuis évolué et servent sur d'autres types de problèmes par exemple les Vision Transformers qui permettent de traiter des images.

Le principe de cette architecture repose sur un mécanisme d'attention. Ce mécanisme permet de ne pas souffrir de mémoire à court terme comme pour les RNN classiques et aussi les LSTM et GRU malgré leur grande marge de manœuvre. Ici on ne parcourt pas la séquence, on la prend en entier directement et on ajoute du contexte ce qui permet une meilleure compréhension du modèle.

2.2.1 Architecture

Les Transformers ont une approche de séquences à séquences, cette démarche permet de prendre une phrase en français et de la traduire en anglais sans pour autant avoir la même longueur de séquence. L'architecture se compose de deux blocs principaux : l'encodeur et le décodeur.

Figure 5 : Architecture des Transformers



(Vaswani et al. 2017)

Dans le schéma de l'architecture ci-dessus, on peut distinguer l'encodeur dans la partie de gauche et le décodeur dans la partie de droite. On remarque également qu'il y a beaucoup de traitements similaires, la particularité la plus importante de cette architecture étant les mécanismes d'attention. Les explications qui vont suivre vont prendre l'exemple qui a popularisé les Transformers, à savoir la traduction d'une langue à une autre.

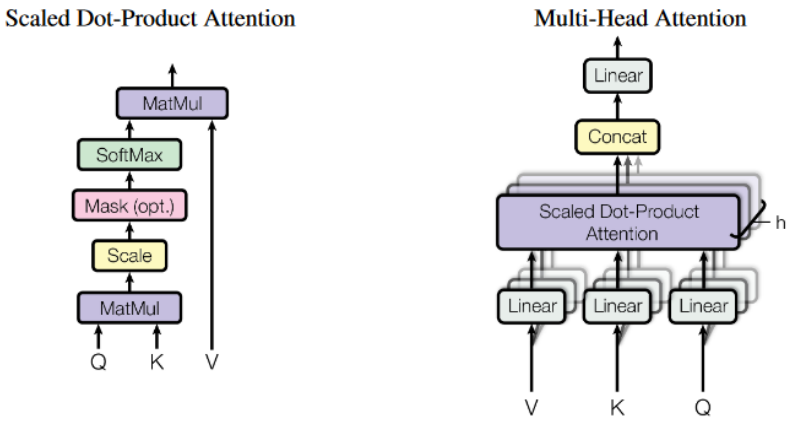
2.2.1.1 Encodeur

La première partie notable est le traitement des données d'entrée avant d'être traitées par l'encodeur le Positional Encoding. Les données sont transposées dans un espace où chaque mot a maintenant une représentation vectorielle selon le sens de la phrase. Ce traitement est

nécessaire, car il faut transformer nos mots en données numériques interprétables par le modèle et qu'il ne contient aucune convolution ou récurrence.

Les données d'entrées vont ensuite passer par une couche de « Multi-Head Attention » puis d'une couche Fully Connected. Ces deux blocs sont suivis d'une normalisation ou l'on va aussi ajouter la représentation numérique originale des mots.

Figure 6 : Fonctionnement du Multi-Head Attention



(Vaswani et al. 2017)

Dans la couche de Multi-Head Attention de la Figure 6, on remarque trois entrées distinctes. Il y a la query Q qui est une représentation vectorielle d'un mot dans la phrase, les clés K et les valeurs V qui ensemble représentent tous les mots de la séquence. La première opération consiste à faire le produit scalaire entre la query Q et les clés K pour obtenir une matrice de scores. Cette matrice nous permet de savoir le niveau d'attention d'un mot par rapport aux autres dans la phrase avec un score plus élevé pour ceux qui nécessitent plus d'attention. Elle est ensuite remise à une échelle en faisant la racine carrée des dimensions de Q et K, ce qui permet d'avoir des gradients plus stables et d'éviter l'explosion de gradients. Un SoftMax est ensuite appliqué pour transformer ces valeurs en probabilités pour que le modèle puisse encore mieux interpréter. Finalement, on multiplie ces probabilités avec tous les mots du vocabulaire pour ne prendre en compte que ceux qui ont du sens. La formule complète de l'attention est donc :

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Viens ensuite le mécanisme de « Multi-Head » ou l'on va ajouter une dimension de h fois les vecteurs Q, K et V qui seront séparés en plusieurs parties avant de calculer leur niveau d'attention pour avoir en parallèle plusieurs représentations de la phrase ce qui permet d'augmenter la robustesse de la représentation des données d'entrées. Une fois toutes les représentations calculées, elles sont concaténées et passées dans une couche linéaire pour ne former qu'une représentation des données d'entrées complètes.

Une fois que les données sont passées dans le processus complet de l'encodeur, on obtient des valeurs concaténées des représentations des données d'entrées et du résultat de notre Multi-Head Attention. Ces valeurs vont être transmises dans le processus du décodeur et vont servir de query dans une couche d'attention de celui-ci.

2.2.1.2 Décodeur

Les blocs du processus du décodeur sont très similaires à ceux de l'encodeur à l'exception d'une couche de Multi-Head Attention supplémentaire. Les données d'entrées sont différentes que celle de l'encodeur, le principe est de décaler d'une position vers la droite tous les mots. Cette simple modification va permettre au modèle d'essayer de prédire le mot suivant plutôt que le mot actuel et ne va pas juste copier les entrées. Les données sont, comme pour le décodeur, représentées numériquement par le Positional Encoding.

La première couche du processus est une Masked Multi-Head Attention. Cette couche particulière permet d'ajouter un masque optionnel pour éviter que l'algorithme connaisse les mots qu'il n'a pas encore traduits. Le résultat en sortie de cette couche est un vecteur masqué contenant des informations sur comment traiter les données d'entrée depuis le décodeur.

La seconde couche de « Multi-Head Attention » a la particularité de faire le lien entre l'encodeur et le décodeur. Les clés K et les queries Q viennent de la sortie du décodeur et les valeurs d'entrées V viennent du vecteur masqué en sortie de la couche précédente.

La sortie du décodeur passe par une couche linéaire puis un SoftMax. Ces deux dernières étapes permettent de classifier et donner les probabilités pour chaque mot dans la langue ciblée. Ces valeurs finales peuvent ensuite être utilisées pour la prédiction ou être réutilisées en tant que données d'entrée du décodeur.

2.2.2 Article

Dans l'article « SOMA : Solving Optical Marker-Based MoCap Automatically » (Ghorbani, Black 2021), l'objectif est à nouveau très similaire à notre projet. Ils appuient même qu'ils arrivent à résoudre des problèmes que certaines solutions de business n'arrivent pas à régler. Cette fois les Transformers sont mis à l'œuvre dans ce projet qui a également de résultats très concluants. L'application du projet est prévue à des fins d'animation et contient par conséquent de nombreux mouvements. La principale motivation reste de minimiser le travail manuel d'un utilisateur pour utiliser ses données de captures.

2.2.2.1 Données

Le projet prévoit de régler des problématiques pour des mouvements complexes, il y a donc des animations de mouvements divers et variés pour le corps humain. Ils incluent également les principales difficultés de la capture de mouvements à savoir les marqueurs qui disparaissent à cause de l'occlusion de certains marqueurs sur des images ou qui sont en trop à cause d'objets réfléchissants la lumière qui ajoute des marqueurs là où ils n'y en avaient pas.

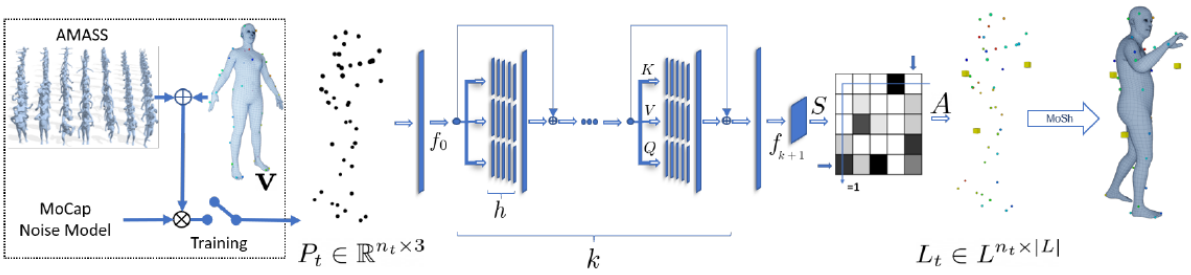
Les chercheurs du projet n'ont pas utilisé de données brutes directement capturées par eux-mêmes. Ils sont allés chercher de nombreux jeu de données disponibles pour la recherche et les ont manipulées pour correspondre à leurs besoins. Ils ont tout d'abord utilisé un corps de genre neutre tiré du modèle de corps SMPL-X. Pour les mouvements de ce corps, ils ont utilisé le jeu de données AMASS qui fournit des animations du corps nombreuses et variées qu'ils ont ensuite appliqué sur leur corps neutre. Ils se sont par la suite servis du jeu de données CAESAR qui leur donne cette fois de nombreuses morphologies différentes pour imiter une capture de mouvements sur des sujets différents.

L'étape suivante pour se rapprocher de données réelles est d'ajouter du bruit aux données. Un léger déplacement aléatoire des marqueurs a été effectué pour avoir un placement légèrement différent pour chaque sujet tout en étant sur les mêmes endroits du corps. Cette opération rend le placement des marqueurs plus organique, car un chercheur ne placera jamais tous les marqueurs aux mêmes endroits au millimètre près. La rotation du corps est aussi aléatoire pour empêcher de n'avoir que des corps qui fixent la même direction. Ensuite ont été dessinés des marqueurs fantômes à l'aide d'une distribution gaussienne 3D pour simuler des captures d'objets réfléchissants autres que les marqueurs. À l'aide d'une distribution uniforme, des points dans des trajectoires de marqueurs ont été mis aléatoirement à 0 pour cette fois reproduire le phénomène de l'occlusion de marqueurs. Le nombre de marqueurs ajoutés et d'occlusions est également aléatoire pour éviter d'en avoir le même nombre pour chaque corps et ainsi ajouter du réalisme. Finalement, ils ont également cassé volontairement des trajectoires pour que l'algorithme soit également capable de traiter des trajectoires non continues.

Le traitement des données à bien été apporté avec soins. Bien que leurs données ne viennent pas directement de captures brutes de leur part, ils ont pris en compte toutes les problématiques liées à la capture et les ont transposées sur des données construites.

2.2.2.2 Architecture

Figure 7 : Schéma de l'architecture du projet SOMA



(Ghorbani, Black 2021)

La première partie du schéma de l'architecture de la Figure 7 nous montre l'étape de préparation des données qui a été présentée dans la section précédente. Le modèle reçoit un nuage de point comme entrée. Ce nuage contient les positions de chaque marqueur dans le temps, les marqueurs sont non labélisés et sont dans le désordre. Ces données sont ensuite traitées par des couches de Multi-Head Attention, elles passent par ce processus k fois pour augmenter les capacités du modèle et permettre au réseau d'avoir une vue locale et globale du nuage.

Une fois sorti du réseau d'attention viens l'étape finale de la prédiction. Tout d'abord, une matrice de score S est établie avec une normalisation de Sinkhorn pour appliquer une couche de transport optimal. Cette normalisation nous permet d'avoir la somme des lignes et des colonnes à 1. Une ligne et une colonne sont ajoutées à cette matrice servant comme « poubelle » pour y assigner tous les marqueurs qui n'ont pas de label et permettre ainsi la gestion des marqueurs fantômes. Enfin, on obtient la matrice d'assignement A qui ne contient pas les valeurs « poubelle » et nous permet de faire l'association entre les points et les labels.

Une dernière fonctionnalité sert à adapter le résultat à un corps 3d. Cette dernière étape permet de passer du nuage de points à un corps humain déjà modélisé en trois dimensions.

Étant donné que ce projet est destiné plutôt à des fins d'animation, cette partie permet de réduire davantage le travail manuel et apporte une visualisation claire du résultat de manière instantanée.

2.2.2.3 Résultats

Tableau 2 : Résultats du projet SOMA

Train \ Test	Précisions			
	B	**B+C**	**B+G**	**B+C+G**
B	97.93	83.55	86.54	73.95
B+C	97.25	97.21	95.37	93.79
B+G	98.06	96.14	97.87	95.27
B+C+G	95.74	95.56	95.73	95.50

(Ghorbani, Black 2021)

Dans le tableau ci-dessus, on observe les résultats des précisions pour les différentes expériences du projet sur le jeu de données HDM05. Les lettres des lignes et des colonnes représentent le type d'expérience. Le « B » est pour les données de base des marqueurs sans ajout de bruits, le « C » est lorsque l'on ajoute l'occlusion qui disperse des trous dans les trajectoires. Enfin, le « G » apporte les marqueurs fantômes qui polluent la capture de mouvements en ajoutant des marqueurs qui ne devraient pas être labélisés.

Parmi tous ces résultats, celui qui est le plus intéressant à regarder est bien entendu celui qui contient toutes les variations possibles, car ce sont les données les plus proches de ce que l'on aurait réellement après une capture sans traitements humains. Ils précisent tout de même après avoir testé leur modèle sur de vraies données que la réussite dépend aussi de leur qualité. Le jeu de données qu'ils ont construit est selon eux de très bonne qualité et implique plusieurs caméras de capture.

La solution a été comparée à la solution commerciale la plus populaire étant le Shogun de Vicon. La précision de SOMA se rapproche énormément de cette solution commerciale, ce qui en fait l'une des meilleures alternatives open source pour le domaine.

Le projet réussi donc à fournir une solution robuste pour le domaine de la capture de mouvement. La précision est très élevée pour des mouvements si variés et prouve que l'utilisation des Transformers a tout son sens pour cette problématique.

3. Méthodologie

3.1 Données

Pour ce projet j'ai eu la chance d'avoir accès à suffisamment de données venant de captures réelles. En effet, de nombreux projets ont dû simuler des trajectoires et faire en sorte qu'elles se rapprochent de la réalité ou utiliser et adapter des données disponibles au public. Toutes les données utilisées pour ce projet proviennent donc du même fournisseur de données qui est également l'unique bénéficiaire prévu.

Toutes les sessions de captures proviennent du même lieu d'enregistrement et toutes sont filmées par plusieurs caméras infrarouges. Généralement, une capture comporte le trajet d'un sujet dans un sens ou dans l'autre. Bien que la capture 3d permet de positionner un sujet partout dans une pièce, ici Ils ne se déplacent que sur une ligne droite. On se retrouve avec des enregistrements de personnes qui marchent sur un axe ou seul le sens peut varier (un enregistrement pour l'aller puis un autre enregistrement pour le retour). L'ensemble de ces captures ont été réalisées sur de nombreuses années, fournissant un large flux de données pour permettre l'entraînement d'un algorithme.

Les données fournies sont dans des fichiers en format C3D (Coordinate 3D Format). Ce type de fichier contient toutes les informations qui permettent de recréer l'espace 3D et tous ses points dans le temps. Il y a de nombreuses métadonnées contenues également dans ces fichiers, il a de nombreuse information sur la personne enregistrée ainsi que toutes les informations sur la capture en elle-même. Il y a aussi des plateformes de force disposées sur la trajectoire qui enregistre des données lorsque le sujet marche dessus. Les informations qui vont nous intéresser pour ce projet sont celles des positions dans l'espace et le temps sur les trois axes. Un seul fichier rassemble une session de marche d'une personne, il y a toujours plusieurs sessions par personne et donc un nombre variable de fichiers par personne. À l'aide d'une nomenclature des fichiers, on peut déjà connaître quel type de capture est effectué et dans quelle condition. Les conditions qui ont été retenues pour les jeux de données sont uniquement les sessions de marche normales sans aide à la marche. L'aide à la marche peut représenter une personne qui assiste le patient durant la capture ou d'objets tels que des béquilles ou une canne.

Un premier algorithme du laboratoire passe les données en revue directement après la capture. Cet algorithme leur permet de reconnaître la trajectoire d'un marqueur sur la longueur de la session. C'est-à-dire qu'il identifie que le marqueur numéro 7 est positionné à tel endroit à l'image 0 et à tel endroit à l'image n. Ce premier traitement permet d'extraire directement les positions des marqueurs non labélisés sans devoir trouver leur trajectoire et ainsi identifier tous les différents marqueurs sur la longueur de la session.

Les jeux de données utilisés vont se diviser en deux parties : un jeu de participants et un jeu de patients. Bien que ces jeux utilisent des données différentes. L'environnement de capture, la structure des données et les labels à prédire sont les mêmes pour les deux.

Bien entendu, ce sont des données personnelles de patients et de participants qui ont été enregistrées dans un cadre médical uniquement. Il est donc impossible de partager les jeux de données qui suivent. Les données qui montrent les performances de l'algorithme à suivre sont utilisables uniquement par l'hôpital.

3.1.1 Jeu de participants

Le premier jeu de données contient des captures de participants. Ce ne sont donc pas des patients atteints de pathologies, mais des personnes volontaires qui ont acceptées de partager leurs données avec le laboratoire à des fins de recherches scientifiques. La particularité de ces données de capture est qu'elles se ressemblent toutes sensiblement étant donné que les sujets n'ont pas de troubles de la marche au préalable.

L'intérêt d'utiliser de telles données est pour construire notre algorithme. Le but étant de trouver comment avoir un algorithme qui peut faire des prédictions correctes sur des données avec des trajectoires similaires comme celle-ci. Une fois que l'on obtient un résultat satisfaisant, on adapte le réseau à des données avec des trajectoires plus variées.

L'ensemble de ce jeu de données rassemble 576 fichiers. Parmi ces 576 captures, il y a 20 participants distincts qui n'ont pas tous le même nombre de captures. Au total, nous avons 18'725 trajectoires de marqueurs différentes à notre disposition. Parmi elles, 15'000 sont utilisées pour le jeu d'entraînement tandis que les 3'725 restantes servent pour le jeu de test.

3.1.2 Jeu de patients

Ce lot de données présente quant à lui des données de vrais patients qui ont été capturées dans des fins médicales. Ces patients présentent des troubles de la marche et sont les principaux sujets de capture du laboratoire. Les trajectoires des marqueurs dans ce jeu de données peuvent beaucoup varier entre elles. En effet, les troubles de la marche ont différentes échelles et n'atteignent pas de manière similaire tous les patients. L'altération de la marche peut donc être tout à fait différente d'une personne à une autre, ce qui en fait la difficulté du projet.

Pour le jeu de patients, j'ai décidé d'utiliser le même nombre de fichiers que les participants à savoir 576. Il y a beaucoup plus de fichiers de patients que de participants, cependant à des fins de comparaisons entre les deux jeux de données, cela fait plus de sens que les jeux de données aient des tailles similaires. De ne pas utiliser toutes les données permet aussi de gagner du temps d'exécution de l'entraînement. Cependant, l'utilisation de toutes les données fera sens lorsque l'algorithme sera performant avec ces deux jeux de données pour l'appliquer en situation réelles.

Le nombre de trajectoires est supérieur à celui du jeu de participant. Bien que l'on utilise le même nombre de fichiers, il manque parfois des labels que l'on aimerait utiliser dans certaines captures ce qui réduit notre nombre de trajectoires extraites. Nous avons donc 19'705 trajectoires pour ce jeu de données appartenant à 71 patients différents. Une fois encore ce nombre est très différent entre les deux jeux de données, car il y a eu un mélange avec certains patients qui ont très peu de sessions de marche comparées à d'autres. Le jeu d'entraînement contient également 15'000 trajectoires tandis que le jeu de test en contient cette fois 4'705 (Tableau 3).

Tableau 3 : Résumé des jeux de données

Jeu de données	Fichiers C3D	Sujets différents	Trajectoires totales	Trajectoires entraînement	Trajectoires validation
Participants	576	20	18’725	15’000	3’725
Patients	576	71	19’705	15’000	4’705

3.2 Traitement des données

Une grande partie de l’architecture est celle du traitement des données et des opérations qu’on doit effectuer pour les rendre calculables et compréhensibles pour l’algorithme. Un premier tri a déjà été effectué au moment de la construction des jeux de données en ne prenant qu’un certain code de session pour s’assurer qu’il s’agit bien de la même expérience. Une liste de label est écrite manuellement dans le code pour n’extraire que les trajectoires qui sont étiquetées avec un label que l’on veut prédire. Cette opération est nécessaire, car les fichiers contiennent parfois de nombreux marqueurs supplémentaires qui ne sont pas ou plus présent lors de la capture.

Pour l’extraction des données, on lit tous les fichiers C3D et on prend tous les marqueurs avec les labels. Notre objectif est de faire une seule matrice qui contient toutes les trajectoires de tous les fichiers au même endroit. La première problématique qui vient à la création de cette matrice est que les captures n’ont pas le même nombre d’images. Le nombre d’images, qui correspond à la durée de l’enregistrement, peut donc grandement varier entre les fichiers. Pour y remédier, une taille fixe d’images a été choisie. Pour tous les enregistrements n’atteignant pas ce nombre d’images, on ajoute des 0 pour que la dimension de la matrice soit respectée et que le modèle ne prenne idéalement en compte ces valeurs.

Parmi les données concernant les patients, l’occlusion des marqueurs ajoute des valeurs non définies pour les images ou le marqueur disparaît. Ces valeurs posent un problème et ne peuvent pas être interprétées par l’algorithme, il faut donc les traiter au préalable. Deux méthodes différentes ont été testées pour combler les positions vides. La première consiste à simplement les remplacer par des zéros pour que l’algorithme ne les prenne pas en compte et passe outre. La seconde méthode plus élaborée est l’interpolation linéaire. Cette pratique consiste à estimer la valeur entre deux points déterminés. On va donc assumer une position pour ce point selon le point précédent et le suivant sur la trajectoire.

Une fois la matrice remplie, il est nécessaire de mélanger toutes ses entrées. Cette pratique est nécessaire, car tous les labels risquent d’être dans le même ordre et l’algorithme pourrait y comprendre un schéma lié à cela, ce que nous voulons éviter. Les labels sont quant à eux séparés et transformés en identifiant numérique afin de pouvoir les assigner avec le modèle.

Il y a eu des essais de normalisation sur les différents axes dans l’objectif d’améliorer les performances de l’algorithme en réduisant les grands changements entre les personnes capturées pour éviter de nuancer le résultat. Dans toutes les captures, les sujets marchent sur le long de l’axe X. La différence principale est le sens dans lequel ils se dirigent, soit vers le X positif, soit vers le X négatif. Des méthodes existent pour faire pivoter le corps de la personne

dans la même direction cependant pour le cadre du projet et avec le temps imparti, j'ai décidé de faire uniquement un essai en ne prenant que les fichiers ou les sujets se déplacent dans une seule des deux directions. Malheureusement les résultats en étaient que très mauvais et la méthode n'a pas été retenue, il n'y a donc aucune normalisation pour cet axe.

Quant à l'axe Y qui représente la hauteur, l'objectif est que la taille d'un sujet affecte le moins possible les résultats de l'algorithme étant donné que ce n'est pas un facteur sensé pour la prédiction. La normalisation effectuée prend la position en Y du marqueur le plus haut de toutes les trajectoires pour s'en servir comme maximum. Toutes les autres positions Y sont ensuite réduites entre zéros et un, un étant la valeur maximale. Pour l'axe Z, le principe fondamental reste le même ou nous ne voulons pas prendre en compte la morphologie. Pour ce faire, on utilise le centre d'inertie des points sur l'axe Z. En calculant la moyenne des points, on obtient ce point central, on remplace ensuite les positions Z par la distance entre les points et le centre d'inertie.

Un calcul de l'accélération et de la vélocité des trajectoires a également été effectué. Inspirée de l'article utilisant un modèle LSTM présenté précédemment, cette pratique a pour objectif d'ajouter des informations à nos données pour que l'algorithme ait plus d'informations à sa disposition. Ces deux informations sont calculées directement à partir des données de position et donnent les valeurs d'accélérations et de vélocités pour chaque image de la trajectoire pour tous les axes en même temps.

3.3 Modèle

3.3.1 Choix du modèle

Il y a plusieurs choses à prendre en compte afin de sélectionner quel modèle de deep learning est le plus adapté au projet. Dans un premier temps, avant même de commencer ce travail, j'envisageais déjà l'utilisation des LSTM ou Transformers afin de traiter cette problématique de manière séquentielle. Dans l'analyse de l'existant, on observe que des architectures avec ces deux modèles ont très bien fonctionnées et il ne fait pas de doutes qu'elles sont tout à fait adaptées au problème.

Si l'on compare les deux articles présentés, les résultats de précisions sont globalement très similaires. Il est toutefois difficile de savoir si l'un est réellement mieux que l'autre en se basant sur les résultats des articles, car ils ont des jeux de données et des expériences différentes. Cependant même s'ils ne présenteraient pas les mêmes performances avec les mêmes conditions, il n'y a pas de doutes qu'elles réussiraient la tâche de manière très correcte l'une comme l'autre.

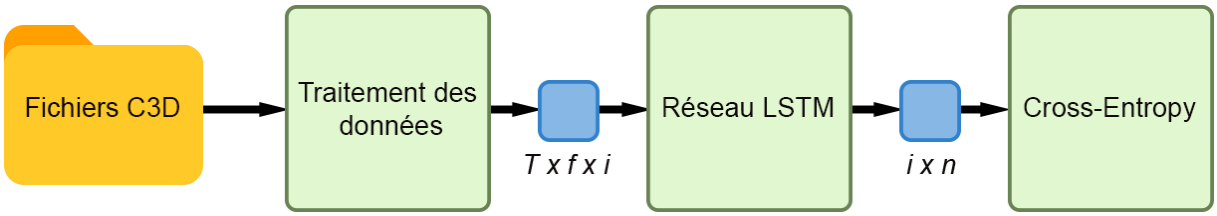
Le choix de la méthode s'est porté plutôt sur le modèle des LSTM et cela pour plusieurs raisons. Pour commencer, les LSTM sont plus adaptés à mon niveau étant donné que le projet est limité dans le temps, il est préférable quand le choix est possible de partir sur une base plus saine. De plus, le projet utilisant les Transformers prévoit des mouvements plus complexes et est prévu pour faire un rendu 3D sur un squelette normalisé, ce qui n'est pas nécessairement notre objectif et bien qu'ils remplissent les conditions de notre problématique il y a tout de même des étapes non nécessaires pour nous qui augmentent la complexité.

Le but n'est pas de copier l'architecture complète de l'article, mais plutôt d'expérimenter une architecture pour s'approcher de leurs résultats et comprendre comment arriver à leur

raisonnement. Notre modèle ne va pas traiter les trajectoires d'un seul fichier ensemble, mais va rassembler toutes les trajectoires de tous les fichiers en une seule matrice.

3.3.2 Architecture

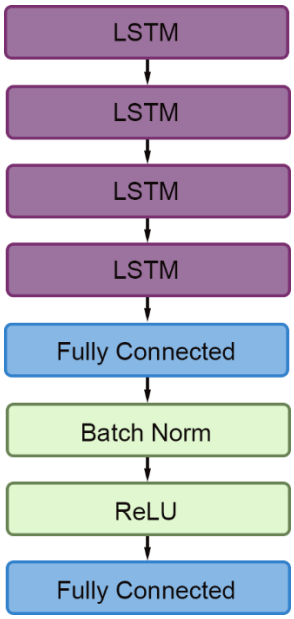
Figure 8 : Architecture globale du projet



L'architecture de notre projet (Figure 8) se déroule en plusieurs étapes. Premièrement, comme abordé plus tôt dans ce mémoire, il y a la phase de traitements des données. Les fichiers C3D sont donc transmis à cette étape et une fois les données prêtes, nous avons une matrice de dimensions $T \times f \times i$. T Représente notre nombre de trajectoires, c'est toutes les trajectoires des marqueurs de tous les fichiers C3D mis ensemble. La dimension f quant à elle représente le nombre de caractéristiques de nos trajectoires de marqueurs. Dans le cadre du projet il y a principalement 3 caractéristiques qui sont les positions sur les trois axes, mais lorsque l'on ajoute l'accélération et la vélocité alors nous en avons 5. Cette dimension n'est pas fixe, car les résultats qui découlent de ces changements sont bien différents. Enfin, la dimension i indique le nombre d'images pris en compte pour la trajectoire.

Cette matrice de données à traiter va ensuite passer le long du réseau avec quatre couches de LSTM, une couche Fully Connected, une Batch Normalization, une ReLU puis une dernière couche Fully Connected comme représenté par le schéma ci-dessous:

Figure 9 : Réseau LSTM du projet



Les quatre premières couches représentent le réseau LSTM multicouches. Le même algorithme est appliqué plusieurs fois pour améliorer l'interprétation des données et aller plus en profondeur. La sortie de ces couches est ensuite transformée pour compacter les résultats et les envoyer à la suite du réseau.

Dans la partie suivante, on effectue des opérations pour stabiliser les données et améliorer les performances du modèle. Dans un premier temps, on effectue une Batch Normalization pour normaliser et recentrer les données qui sont une bonne pratique pour stabiliser les gradients du modèle. Une fonction d'activation ReLU est ensuite appliquée, cette dernière permet de réduire davantage les problèmes de gradient en ne prenant en compte que les valeurs positives. Ces deux pratiques sont très rependues lors de la création de réseau de neurones profonds bien qu'il existe de différents types d'opérations selon les besoins pour améliorer la stabilité du modèle. Une nouvelle couche Fully Connected nous transmet ensuite une matrice de score de dimension $i \times n$ ou n représente le nombre de labels que l'on veut prédire. La matrice contient donc tous les scores par image pour chaque label à prédire.

L'architecture se termine par la fonction de coût de la Cross Entropy. Cette fonction de coût est réputée dans le domaine des algorithmes de classification et est devenue un standard. L'objectif de cette fonction est d'indiquer si notre prédiction fait partie de l'ensemble des données réelles. En minimisant cette fonction, on rapproche la distribution de la prédiction de la distribution réelle pour avoir des résultats réalistes. La fonction est définie comme :

$$H(p,q) = -\int_x p(x) \log q(x) dx$$

3.3.3 Configuration

Dans cette section, nous allons discuter du choix des paramètres qui ont été choisis au travers de l'architecture du projet. Choisir des paramètres adaptés est indispensable afin d'améliorer les performances de notre modèle. Pour le cadre de ce projet, je n'ai pas fait appel à des méthodes pour tester de nombreux paramètres automatiquement. Quelques tests manuels ont été effectués pour certains paramètres afin de trouver des améliorations dans les résultats. Étant donné le nombre de paramètres et n'ayant pas pris d'accès pour un cluster de machines, tester de nombreuses combinaisons de paramètres aurait pris trop de temps sur ma machine personnelle.

Le premier paramètre qui est appliqué est au moment du traitement des données. Comme mentionné dans la partie concernée, on prend un certain segment des trajectoires et ont rempli celles qui sont trop courtes pour que toutes les séquences aient la même taille. La taille sélectionnée est de **500** images pour chaque trajectoire. En effet, le nombre d'images au travers de l'ensemble des données est très différent et peut varier entre 200 et 600 images. De nombreuses séquences se retrouvent par conséquent allongées par des zéros afin d'atteindre la taille nécessaire, cependant des tests ont été effectués avec une taille qui ne nécessite pas de remplissage et les résultats n'ont pas subi de changements flagrants.

Les hyper paramètres d'exécution du modèle nous permettent de contrôler la cadence d'apprentissage de notre algorithme et doivent être choisis en voyant évoluer notre modèle pour atteindre une stabilité de la précision. Le batch size qui sépare les données en plusieurs lots a été fixé à **500** trajectoires. Le Learning Rate est quant à lui à **0,0001** pour contrôler la vitesse d'optimisation de l'algorithme. Enfin le nombre d'epoch a été fixé à **50** pour donner la durée de l'entraînement. On remarque dans les graphiques des résultats que ce nombre d'epoch est amplement suffisant étant donné que la courbe se stabilise rapidement. Le choix de ce nombre a été fait pour montrer que le modèle se stabilise et que le temps d'entraînement n'est pas excessivement long.

Le modèle contient également ses hyperparamètres. La majorité de ces valeurs ont été directement inspirées du projet de l'article utilisant une architecture LSTM mentionné plus tôt dans ce mémoire. Quelques essais manuels pour les changer ont été effectués, ce qui a pu améliorer légèrement les performances. Tout d'abord, comme le schéma de l'architecture le montre précédemment, il y a **4** couches de LSTM. Avoir plus de couches produit des problèmes de mémoire c'est donc une limite raisonnable. Le nombre de cellules LSTM que doit traverser la séquence est de **256** pour chacune des couches. Le Dropout, fixé à **0.17**, permet de passer outre certains nœuds du modèle, le but de cette pratique est d'éviter que le modèle soit surentrainé sur les données d'entraînement, ce que l'on veut éviter à tout prix dans un modèle d'apprentissage automatique. Enfin, le nombre de nœuds des couches Fully Connected est à **128** pour définir la taille de la transformation (Tableau 4).

Tableau 4 : Résumé des hyperparamètres

Hyperparamètre	**Valeur**
Taille de séquence (fenêtre)	500
Batch Size	500
Learning Rate	0.0001
Epoch	50
Couches de LSTM	4
Cellules LSTM	256
Dropout	0.17
Nœuds FC	128

Toutes les sessions d'entraînements ont été effectuées sur le même environnement. Un ordinateur personnel avec une carte graphique NVIDIA GeForce RTX 3080 avec 10 Gigas de mémoire et un processeur AMD Ryzen 9 5900X 12-Core. Tout le code a été réalisé sur PyCharm 2021 dans un environnement Python 3.9. Les librairies utilisées sont Pytorch, BTK, Numpy et Sklearn.

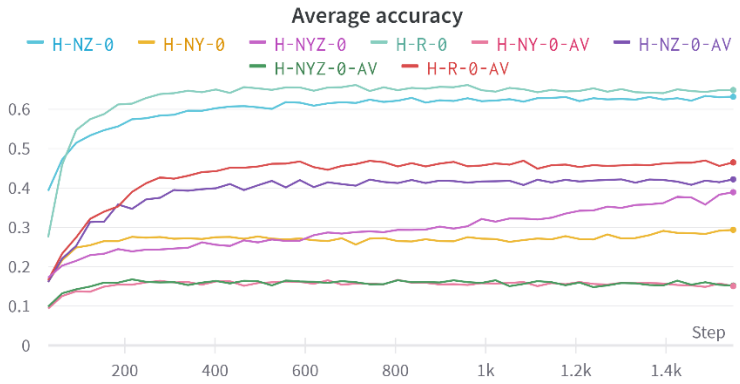
4. Résultats

Le modèle a été appliqué sur les deux jeux de données sous forme de plusieurs expériences par jeu. Les différentes expériences représentent les différents paramètres appliqués aux données ou au modèle pour analyser leur impact sur la précision obtenue. Pour chaque expérience, le jeu de données utilisé à l'entraînement sera toujours le même pour la validation, car celui-ci a été partitionné comme mentionné précédemment.

Une nomenclature des expériences a été mise en place pour distinguer et comparer plus facilement les résultats. Les noms apparaissent tels que « H-NYZ-0-AV », les termes séparés par des tirets ont leur signification :

- Le premier terme indique le jeu de données, la lettre **H** représente le jeu de données des participants tandis que la lettre **P** représente celui des patients.
- Le second nous indique quel type de normalisation a été appliqué sur nos données. La lettre **R** indique que les données sont brutes et par conséquent n'ont subi aucune normalisation. La lettre **N** est suivie des axes qui ont été normalisés à savoir **Y**, **Z** ou **les deux**. Les séparés nous permettent de constater si une normalisation à plus d'effets négative ou positive que l'autre.
- Le terme suivant désigne quelle méthode est mise en place pour remplacer les valeurs non définies des marqueurs victime d'occlusions pendant la trajectoire. Le chiffre **0** représente le remplacement de ces valeurs non définies par des zéros tandis que le terme **LIN** indique l'utilisation de l'interpolation linéaire.
- Enfin, si la mention **AV** est présente, alors l'accélération et la vélocité sont ajoutées aux caractéristiques en plus des positions dans l'espace. Lorsque ce n'est pas le cas, le terme est absent.

Figure 10 : Précisions moyennes du modèle sur le jeu de données des participants



Les courbes présentées ci-dessus représentent la moyenne de précision pour chaque pas de l'optimiseur. Ce sont les valeurs de précision qui sont calculées pendant la validation avec le jeu de test. Le premier constat que l'on observe sur ces courbes et que l'expérience qui obtient le meilleur résultat est celle où l'on n'utilise les données brutes sans traitement. En effet il semblerait que tous les traitements accordés aux données à fait baisser la précision du modèle de manière plus ou moins flagrante.

La normalisation des données pour l'axe Y a clairement fait chuter la précision tandis que celle de l'axe Z donne le résultat le plus proche de la courbe des données brutes. Le fait d'ajouter les informations de l'accélération et de la vélocité a drastiquement fait chuter la précision également. Étant donné que ce sont les mêmes expériences, on observe que les courbes avec et sans les caractéristiques en plus ont un schéma similaire seulement celles qui ajoutent les caractéristiques ont un résultat bien inférieur.

Il y a cependant une courbe qui visiblement continue de monter à la fin du graphique et n'a pas atteint son seuil de stabilité. Cette courbe représente l'expérience ou l'on normalise les données sur les axes Y et Z sans ajouter les caractéristiques de vitesse et de vélocité. Une expérience avec un nombre d'epoch supplémentaire a été effectuée et la valeur retenue dans le tableau ci-dessous en est tirée. Cependant, pour des raisons visuelles, la courbe a été raccourcie dans le graphique au niveau des autres courbes qui se sont stabilisées.

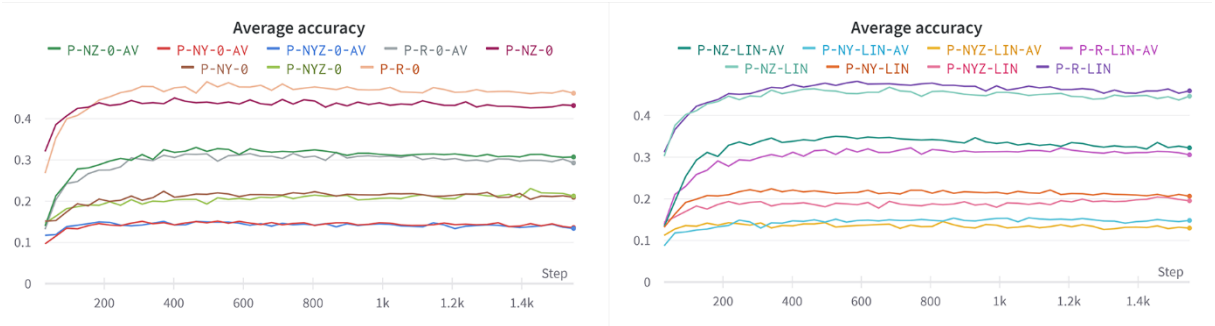
Tableau 5 : Précisions de chaque expérience sur le jeu de données des participants

Précisions obtenues sur le jeu de données des participants (H)				
	R	**NY**	**NZ**	**NYZ**
Sans AV	66.17%	29.36%	63.38%	64.36%
Avec AV	46.98%	16.67%	42.2%	16.80%

La meilleure précision atteinte est par conséquent bien celle des données brutes qui atteint au maximum 66.17% de précision sur la courbe. Le résultat le plus proche étant celui qui rajoute la normalisation sur les deux axes suivis de près par la normalisation sur l'axe Z uniquement. Ces résultats sont relativement proches, bien qu'aucun ne dépasse l'expérience des données non traitées.

La normalisation sur l'axe Y uniquement est visiblement l'expérience rapportant la précision la plus faible. Notamment lorsque l'on rajoute les caractéristiques supplémentaires ou tous les résultats perdent significativement en précision. Un constat intéressant est que les deux résultats pour la normalisation sur les deux axes soient si différents. Bien que l'un a eu recours à un entraînement plus long, l'autre a vu sa courbe se stabiliser très rapidement à un faible résultat.

Figure 11 : Précisions moyennes du modèle sur le jeu de données des patients



La principale différence avec les données des participants et qu'il y a en plus le traitement des valeurs non définies qui double le nombre d'expériences. Dans la Figure 11, le schéma de

gauche montre les expériences qui remplacent par des zéros les valeurs non définies et le schéma de droite avec de l'interpolation linéaire. Globalement, on remarque que les courbes restent similaires sur tous les graphiques, bien que la précision ait baissé par rapport aux données de participants. Le modèle obtient toujours une meilleure précision avec les données brutes et les différents changements affectent la précision de la même manière que pour l'autre jeu de données. Cette fois en revanche, la courbe de l'expérience avec la normalisation sur les deux axes semble se stabiliser rapidement en comparaison de celle utilisant le jeu des participants.

Les deux graphiques utilisant le jeu de patients (Figure 11) se montrent également très similaires. On remarque de légère différence entre les courbes, mais rien de très significatif. Ce constat se fait également dans le tableau des résultats ou l'on observe que les données utilisant de l'interpolation linéaire obtiennent généralement des précisions légèrement supérieures aux autres.

Tableau 6 : Précisions de chaque expérience sur le jeu de données des patients

Précisions obtenues sur le jeu de données des Patients (P)					
		R	NY	NZ	NYZ
0	Sans AV	48.95%	22.40%	45.03%	23.08%
	Avec AV	31.84%	15.16%	33.03%	15.09%
LIN	Sans AV	48.22%	22.40%	46.79%	20.44%
	Avec AV	32.22%	15.44%	35.01%	14.54%

Avec ou sans interpolation linéaire, le meilleur résultat reste vers les 48%. Ce résultat est toujours le meilleur avec les données qui n'ont subi aucun traitement. Les précisions obtenues sont visiblement inférieures que celles du jeu de données des participants.

4.1 Constat des résultats

Ce qui ressort le plus en inspectant les résultats, c'est l'écart entre les résultats des deux jeux de données. En effet, le jeu de participants a obtenu de meilleures performances avec le modèle que le jeu de patients. Cet effet n'est pas surprenant étant donné que les trajectoires des patients peuvent beaucoup plus varier que celles de participants. Le problème en devient plus simple à prévoir pour des sujets qui ont une démarche similaire les uns avec les autres. Cependant il y a fort à parier que si l'on applique le modèle appris avec le jeu de participants sur des données de patients, que le risque d'erreur sera nettement supérieur.

Le fait que les normalisations utilisées n'améliorent pas les performances fait penser que ce n'est peut-être pas la bonne manière de normaliser ou d'utiliser les positions pour la prédiction. Notamment la tentative de normalisation de la taille sur l'axe Y qui a eu beaucoup d'impact sur la précision. La normalisation de l'axe Z quant à elle a eu peu d'impact. Ce phénomène est peut-être dû au fait que malgré les différentes personnes et leur morphologie, les valeurs diffèrent peu. Étant donné que les capteurs sont placés aux mêmes endroits anatomiques sur

une personne et qu'ils ne se déplacent que sur l'axe X, les valeurs sont potentiellement très similaires d'une personne à une autre.

La tentative d'ajouter des informations telles que l'accélération et la vélocité n'a clairement pas eu l'effet escompté. En effet, pour toutes les expériences qui les concernent, pas une seule ne s'est approchée du meilleur résultat. Bien que la pratique ait été reprise d'un article, les données ne sont pas tout à fait les mêmes et cela empêche peut-être le modèle d'identifier une trajectoire au travers de ces informations. Il est aussi possible que la structure des données ou que le calcul doit être fait différemment pour que l'algorithme le comprenne.

Parmi les expériences du jeu de données sur les patients, on note que le changement de traitement pour les valeurs non définies a eu un impact bien que minime. Même si la valeur de précision maximum atteinte est celle des expériences ou l'on remplace par des zéros, l'expérience équivalente avec de l'interpolation linéaire arrive quasiment au même résultat. L'interpolation linéaire à même montré généralement des meilleurs résultats dans les autres expériences, son utilisation a donc tout son sens pour permettre à l'algorithme de s'améliorer.

4.2 Difficultés rencontrées

L'élaboration de ce projet ne fut pas sans difficulté, en effet ce projet est l'un des premiers que j'effectue dans le domaine du deep learning et chaque étape me prend du temps. La lecture et la compréhension des différents articles prennent une bonne partie du temps, n'étant pas habitué, il est plus difficile de trouver les informations clés et de se les approprier. Les compétences pratiques prennent également un temps considérable à acquérir. Bien que les outils soient déjà pris en main depuis un certain temps, c'est le premier réseau de ce type que je construis. Par conséquent, une bonne partie consiste à effectuer des recherches et de la lecture de code des autres projets. De nombreux essais non fructueux auraient également pu être évités avec plus d'expérience dans le domaine.

Il y a eu notamment une erreur dans le code qui a fait perdre un temps considérable. Cette erreur m'a faussé mes résultats pendant un certain temps et n'a pas été détectée dans l'immédiat. La cause à une faute d'inattention pendant l'écriture du code qui a perturbé la réflexion de l'architecture à un stade avancé du projet.

Trouver les bons changements à faire pour améliorer le résultat est également une des grandes difficultés de la problématique et dans le domaine de l'apprentissage automatique en général. À commencer par les hyperparamètres qui auraient sûrement mérité une méthode plus appropriée pour leur choix afin de maximiser leur efficacité. Les différents essais dans l'architecture et le modèle peuvent selon quoi rendre l'entrainement très long et doivent être recommencé au moindre changement, ce qui consomme une grande quantité de temps.

5. Discussion

5.1 Interprétation des résultats

En observant les courbes et les résultats, on remarque que le modèle réussit à apprendre. Les résultats n'approchent malheureusement pas aussi près qu'espérer les précisions des projets des articles présentés, bien que l'ambition n'ait jamais été de les dépasser. Malgré le temps imparti, nous avons tout de même une solution capable d'établir des prédictions avec un certain taux de réussite.

Parmi les nombreuses expériences effectuées sur les jeux de données, nous avons constaté que les effets sont similaires et pour le moment affectent négativement la précision de la prédiction. En trouvant certains ajouts au modèle ou à la transformation des données, il est possible de trouver une caractéristique qui aura l'effet inverse et va aider le modèle à améliorer ses prédictions. Puisque les effets étaient similaires sur les deux jeux de données, alors il est fort probable qu'une caractéristique qui augmente la précision du modèle sur le jeu de données des participants fonctionne alors également pour le jeu de données des patients.

Bien qu'une amélioration puisse améliorer les deux performances, il ne faut pas oublier un constat important. Les performances restent plus faibles pour le jeu de données des patients. Comme mentionné précédemment, ce constat n'a rien d'anormal étant donné que les trajectoires sont bien plus différentes les unes des autres que dans le jeu des participants. Cependant, il y a bien un moment où il faut trouver un moyen de rapprocher les résultats au mieux possible. C'est l'aspect de cette problématique qui a causé la chute des précisions dans les articles mentionnés et qui a également impacté notre projet. Le premier objectif étant de maximiser la précision sur le jeu des participants, la problématique n'a pas eu le temps d'être abordée.

L'architecture LSTM à bien porter ses fruits quant à ses capacités de calcul pour une telle problématique. La difficulté maintenant est de savoir si le modèle peut être amélioré lui seul pour atteindre des précisions plus avancées ou s'il faut ajouter un élément dans l'architecture pour complexifier le traitement.

5.2 Hypothèses

En se basant sur l'existant et les résultats observés, plusieurs hypothèses sont venues durant le projet. Par manque de temps ou simplement parce qu'elles sont arrivées trop tard pour le projet, ces hypothèses n'ont malheureusement pas pu être mises en œuvre, mais restent importantes, car c'est le fruit de tous nos constats.

Premièrement, une expérience intéressante aurait été d'utiliser toutes les données de patients à disposition pour entrainer notre modèle. La partie de données utilisées dans le jeu correspondant est une infime partie de ce que représente l'ensemble des données mises à disposition par l'hôpital. Le choix de ne prendre que cette partie était causé par les limitations du jeu de participants qui n'avait en revanche pas plus de données utilisables, mais également les limitations techniques de mon environnement. Bien qu'il aurait été possible de demander l'accès à un cluster de l'école pour effectuer l'entraînement à la place de mon environnement personnel, les performances des participants n'auraient pas été changées. À noter que l'utilisation d'un puissant cluster à distance aurait également permis de faire des tests de

performance avec plusieurs hyperparamètres pour trouver les plus adaptés. Cependant cela aurait pris un temps considérable surtout avec la quantité totale des données.

Une expérience qui aurait été intéressante serait d'appliquer notre modèle sur les données des articles de référence pour voir si les changements dans la structure des données et éventuellement les mouvements différents auraient obtenu des précisions similaires. Cependant le traitement des données est très différent et la plupart des jeux de données demandent une licence payante pour être utilisés, ce qui rend la tâche complexe à effectuer. Les comparaisons auraient été cependant un constat intéressant à relever.

Une alternative à l'utilisation d'un modèle LSTM est d'entraîner un modèle de Transformers. Comme nous l'avons vu dans l'existant, ce modèle a également fait ses preuves dans le domaine. Il est possible qu'un modèle de Transformers surpasse notre modèle sans pour autant avoir besoin d'une architecture plus complexe. Mais il est également possible que les résultats soient les mêmes ou moins bon que notre modèle étant donné que les deux types arrivent sensiblement au même résultat.

Comme nous l'avons constaté, l'utilisation de la normalisation des positions ne s'est pas passée comme prévu. N'ayant normalisé que les axes Y et Z, la normalisation de l'axe X aurait peut-être pu améliorer les performances. Une manière de le normaliser aurait été de faire pivoter les corps des sujets pour qu'ils fassent tous face à la même direction. Cette méthode est celle utilisée par l'article utilisant les LSTM en indiquant deux marqueurs comme base à identifier pour faire pivoter l'ensemble des marqueurs. La méthode est légitime, mais complexe à mettre en place, il faut visualiser le résultat pour vérifier que tout soit aligné correctement et que les trajectoires restent intactes.

Une autre normalisation expérimentale qui pourrait faire une expérience intéressante serait de rallonger les trajectoires courtes pour que l'ensemble des trajectoires dure le même temps (en l'occurrence le même nombre d'images). Cette méthode pourrait éviter l'utilisation de la fonction de remplissage par des 0. Une autre méthode de remplissage qui n'a pas été testée serait de remplir le reste de la matrice avec les dernières valeurs de position enregistrée. Avec cette méthode le squelette resterait figé à la fin de la séquence plutôt que d'avoir des valeurs nulles.

Pour remplacer les valeurs non définies, l'interpolation linéaire a eu une petite différence face aux faits de les remplacer par des zéros. D'autres méthodes d'interpolation existent et pourraient tout à fait être utilisées pour gérer l'occlusion des marqueurs. Même si la performance n'en sera pas forcément beaucoup altérée. Il serait intéressant d'avoir un traitement qui parvient à gérer l'occlusion et retrouver l'emplacement du marqueur au plus proche de la réalité.

L'ajout des deux caractéristiques tel que l'accélération et la vélocité ont montré des résultats peu probants. Cependant je reste persuadé qu'en ajoutant des caractéristiques à nos données il y a des chances de grandement aider le modèle pour identifier les trajectoires et ainsi améliorer la prédiction. Comme mentionné lors de la description des données, les fichiers C3D contenants les trajectoires collectent également de nombreuses métadonnées qui pourrait potentiellement aider l'algorithme à mieux comprendre les variations de trajectoires. Bien sûr, il faudrait réaliser de nombreuses expériences pour relever quelles métadonnées apportent un réel intérêt quant à leur impact sur la décision. Les caractéristiques peuvent venir

également de calcul à partir de nos données comme pour la tentative de l'accélération et de la vélocité qui peut-être sous une autre forme pourrait donner quelque chose de positif.

Changer l'architecture du projet est aussi envisageable. En prenant en compte le projet existant utilisant un modèle LSTM, leur architecture prend en compte les sessions de marche individuellement les unes des autres. Cette pratique à un aspect fort intéressant pour le modèle. En effet, on peut grâce à cela indiquer au modèle qu'un label ne peut être attribué qu'une seule fois par session. Cette gestion des doublons n'est pas prise en compte dans notre projet et prend tout son sens dans les trajectoires de marqueurs différents qui sont très proches entre elles. Un autre avantage obtenu par cette méthode est de pouvoir calculer la position relative des points plutôt que la position dans l'espace. Au lieu d'indiquer à quel endroit dans l'espace se trouve un marqueur, on donne sa distance sur les trois axes par rapport aux autres marqueurs de la session. En plus de communiquer une information de position, on obtient une indication de distance. C'est également peut-être la raison pour laquelle l'accélération et la vélocité fonctionnent dans leur projet en tant que caractéristique supplémentaire.

Toutes ces hypothèses pourraient aussi permettre de rapprocher les résultats du jeu des patients de ceux avec le jeu des participants. En effet, une normalisation appropriée pourrait rapprocher les ressemblances des données entre elles. Il fait sens aussi qu'un algorithme qui arrive à prédire sans faute sur le jeu des patients doive prédire également sans faute sur le jeu des participants. Un autre constat intéressant aurait été donc d'essayer de faire des prédictions sur le jeu de données des participants en utilisant le modèle entraîné sur celui des patients.

Une fois l'algorithme fonctionnel, il ne faut pas oublier qu'il y a également plusieurs types de sessions de marche. Typiquement les sessions avec de l'aide à la marche ont été omises pour le moment. En théorie, les données de ce type devraient augmenter la difficulté de l'apprentissage. Si l'aide à la marche est une personne à côté, il y a des chances que l'occlusion de marqueurs soit plus présente et quand bien même les marqueurs seraient toujours autant visibles, les trajectoires seront assez différentes des autres surtout au niveau des bras. Une solution à ce problème serait d'entraîner un modèle uniquement sur ce type de données et choisir le bon modèle au moment du labelling afin d'avoir une approche différente sur les types de données différentes.

6. Conclusion

6.1 Résumé

Dans ce mémoire, nous avons analysé plusieurs méthodes open source existantes pour la problématique. Il en existe de plusieurs types qui proposent depuis peu des résultats très probants pour des utilisations plus variées que notre cas. Les architectures utilisées ont servi à grandement inspirer la structure de l'architecture de notre projet.

Notre projet a pu expérimenter des méthodes différentes pour traiter les données de capture. Tout en utilisant les mêmes architectures de modèles que celles des projets existants qui ont déjà fait leurs preuves, notre modèle prend toutes les trajectoires en une seule grande matrice plutôt que de séparer les sessions comme le font les autres projets analysés. L'architecture de notre projet est bien plus simplifiée et peut fonctionner avec tout type de labels pour autant qu'on les indique dans une liste et qu'ils soient déjà présents dans les fichiers d'entraînement.

Les résultats ont montré que notre architecture simplifiée peut apprendre à partir de nos données. Cependant, le modèle n'est pas assez complexifié pour atteindre les mêmes performances que les projets existants. Les résultats sont tout de même en bonnes voies et avec les bons changements dans le modèle, l'amélioration est clairement possible.

6.2 Avenir du projet

Comme mentionné précédemment, il faudrait ensuite améliorer les performances du projet en complexifiant l'architecture et en adaptant les données avec d'autres méthodes pour rendre la tâche d'apprentissage plus compréhensible par l'algorithme. Une restructuration complète peut être nécessaire s'il faut reproduire la méthode d'apprentissage par session de marche.

Dans l'idéal, l'algorithme devrait être utilisable par des personnes sans avoir à installer ou exécuter du code en python. Une interface graphique pour gérer ce processus aurait alors tout son sens. Selon le logiciel utilisé par les chercheurs, l'élaboration d'une intégration directe au logiciel existant serait une alternative tout à fait envisageable. Dans l'idée les chercheurs n'auraient qu'un seul programme ou ils peuvent exécuter tous les traitements nécessaires sur les données brutes.

Une fois qu'une architecture est complète et totalement fonctionnelle. Le projet peut aller plus loin et essayer d'autres modèle qui pourraient surpasser les capacités ou même simplement pour augmenter la vitesse d'entraînement pour des questions de confort.Bien sûr, les solutions existantes, même si elles demandent d'adapter la solution aux données et labels souhaités, sont déjà bien établies et performent de manière tout à fait correcte. Une utilisation directe de ces solutions open source ferait également tout à fait sens si l'on ne veut pas continuer dans la direction de ce projet. Bien que l'idéal serait que la solution soit directement applicable aux données, les limitations dans le temps et dans mes compétences font qu'il est difficile d'arriver à des résultats de projet de grosses envergures.

Bibliographie

CLOUTHIER, Allison L., ROSS, Gwyneth B., MAVOR, Matthew P., COLL, Isabel, BOYLE, Alistair et GRAHAM, Ryan B., 2021. Development and Validation of a Deep Learning Algorithm and Open-Source Platform for the Automatic Labelling of Motion Capture Markers. *IEEE Access*. 2021. Vol. 9, pp. 36444-36454. DOI 10.1109/ACCESS.2021.3062748.

CRUZ, Rodrigo Santa, FERNANDO, Basura, CHERIAN, Anoop et GOULD, Stephen, 2017. *DeepPermNet: Visual Permutation Learning*. [en ligne]. 10 avril 2017. arXiv. arXiv:1704.02729. [Consulté le 28 juillet 2022]. Disponible à l'adresse: http://arxiv.org/abs/1704.02729arXiv:1704.02729 [cs]

CUENAT, Stéphane et COUTURIER, Raphaël, 2022. *Convolutional Neural Network (CNN) vs Vision Transformer (ViT) for Digital Holography*. [en ligne]. 27 janvier 2022. arXiv. arXiv:2108.09147. [Consulté le 2 août 2022]. Disponible à l'adresse: http://arxiv.org/abs/2108.09147arXiv:2108.09147 [cs, eess]

DOSOVITSKIY, Alexey, BEYER, Lucas, KOLESNIKOV, Alexander, WEISSENBORN, Dirk, ZHAI, Xiaohua, UNTERTHINER, Thomas, DEHGHANI, Mostafa, MINDERER, Matthias, HEIGOLD, Georg, GELLY, Sylvain, USZKOREIT, Jakob et HOULSBY, Neil, 2021. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. [en ligne]. 3 juin 2021. arXiv. arXiv:2010.11929. [Consulté le 3 août 2022]. Disponible à l'adresse: http://arxiv.org/abs/2010.11929arXiv:2010.11929 [cs]

GHORBANI, Nima et BLACK, Michael J, 2021. SOMA: Solving Optical Marker-Based MoCap Automatically. . 2021. pp. 10.

GHORBANI, Saeed, ETEMAD, Ali et TROJE, Nikolaus F., 2019. Auto-labelling of Markers in Optical Motion Capture by Permutation Learning. In: GAVRILOVA, Marina, CHANG, Jian, THALMANN, Nadia Magnenat, HITZER, Eckhard et ISHIKAWA, Hiroshi (éd.), *Advances in Computer Graphics*. [en ligne]. Cham: Springer International Publishing. pp. 167-178. Lecture Notes in Computer Science. [Consulté le 3 mars 2022]. ISBN 978-3-030-22513-1.

HOCHREITER, Sepp et SCHMIDHUBER, Jürgen, 1997. Long Short-term Memory. *Neural computation*. 1 décembre 1997. Vol. 9, pp. 1735-80. DOI 10.1162/neco.1997.9.8.1735.

HOLZREITER, Stefan, 2005. Autolabeling 3D tracks using neural networks. *Clinical Biomechanics*. 1 janvier 2005. Vol. 20, no. 1, pp. 1-8. DOI 10.1016/j.clinbiomech.2004.04.006.

SHERSTINSKY, Alex, 2020. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Physica D: Nonlinear Phenomena*. mars 2020. Vol. 404, pp. 132306. DOI 10.1016/j.physd.2019.132306. arXiv:1808.03314 [cs, stat]

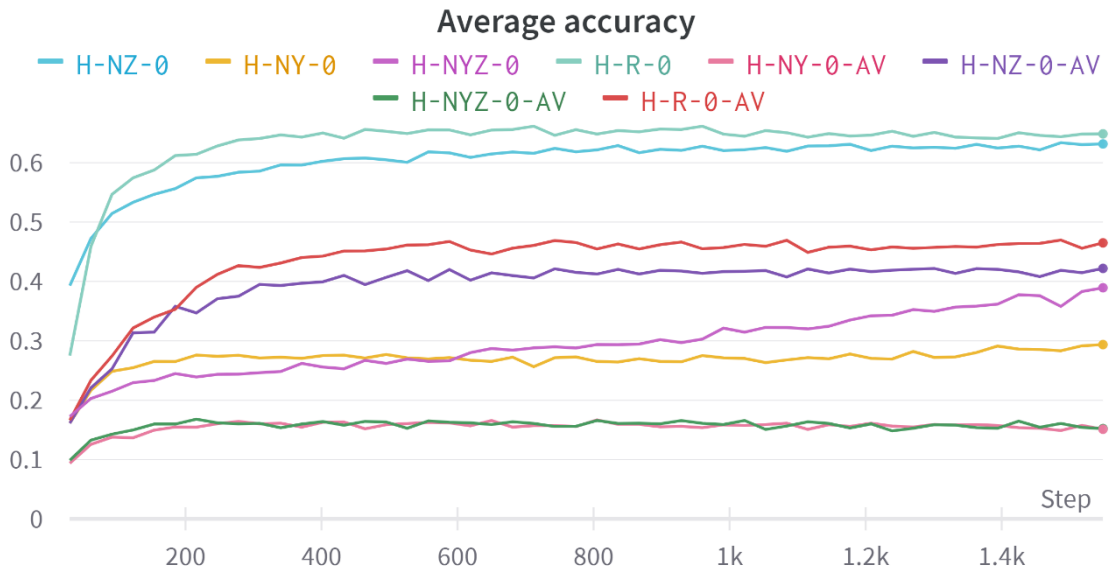
Understanding LSTM Networks -- colah's blog, 2015. [en ligne]. [Consulté le 2 août 2022]. Disponible à l'adresse: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

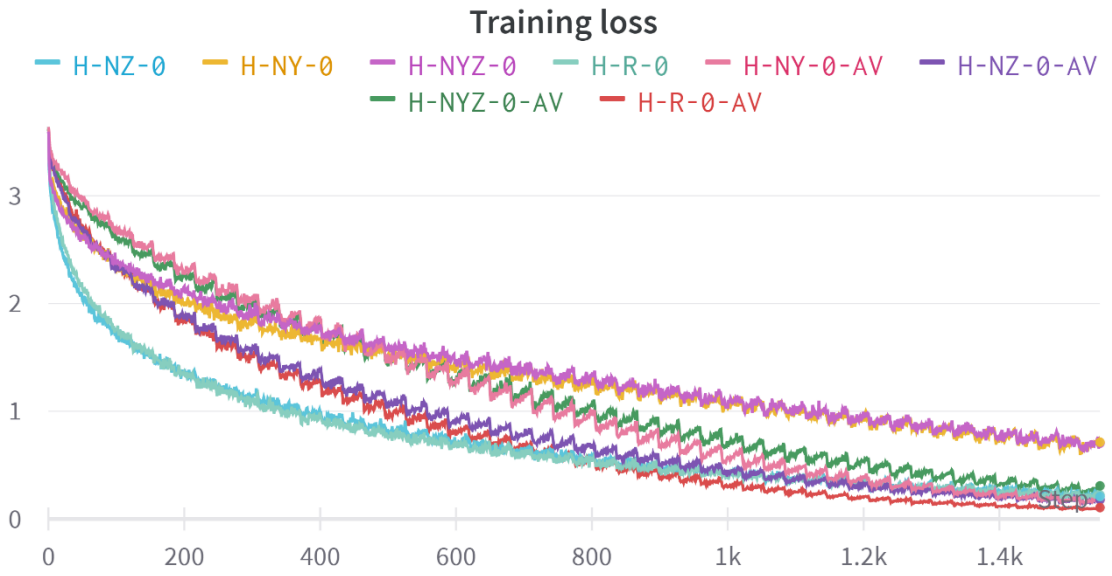
VASWANI, Ashish, SHAZEER, Noam, PARMAR, Niki, USZKOREIT, Jakob, JONES, Llion, GOMEZ, Aidan N, KAISER, Łukasz et POLOSUKHIN, Illia, 2017. Attention is All you Need. In: *Advances in Neural Information Processing Systems*. [en ligne]. Curran Associates, Inc. 2017. [Consulté le 14 juillet 2022]. Disponible à l'adresse: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

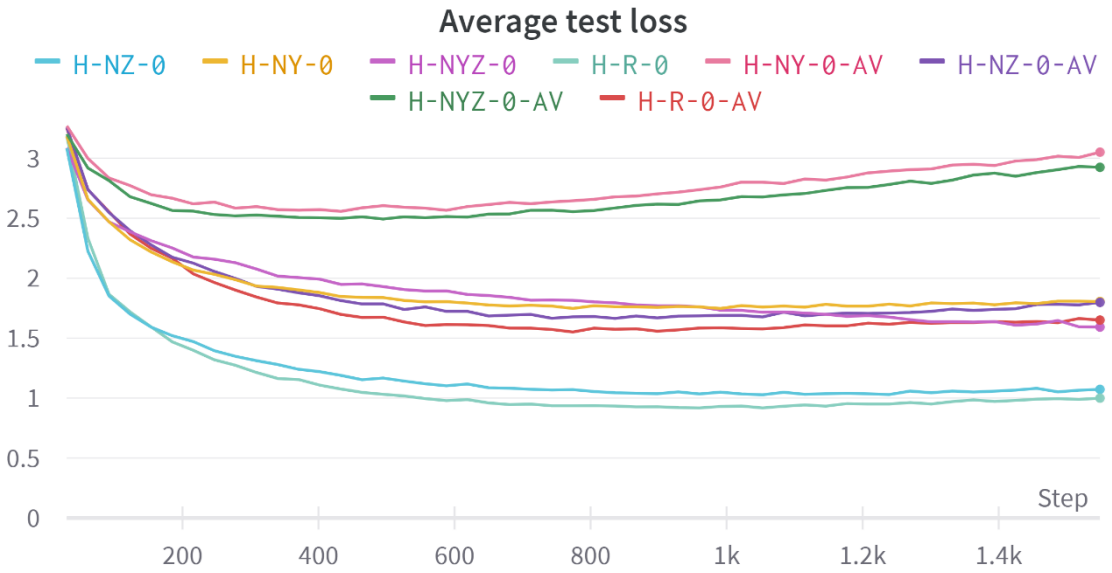
Annexe 1 : Graphiques du modèle sur le jeu de participants

Rappel de la nomenclature :

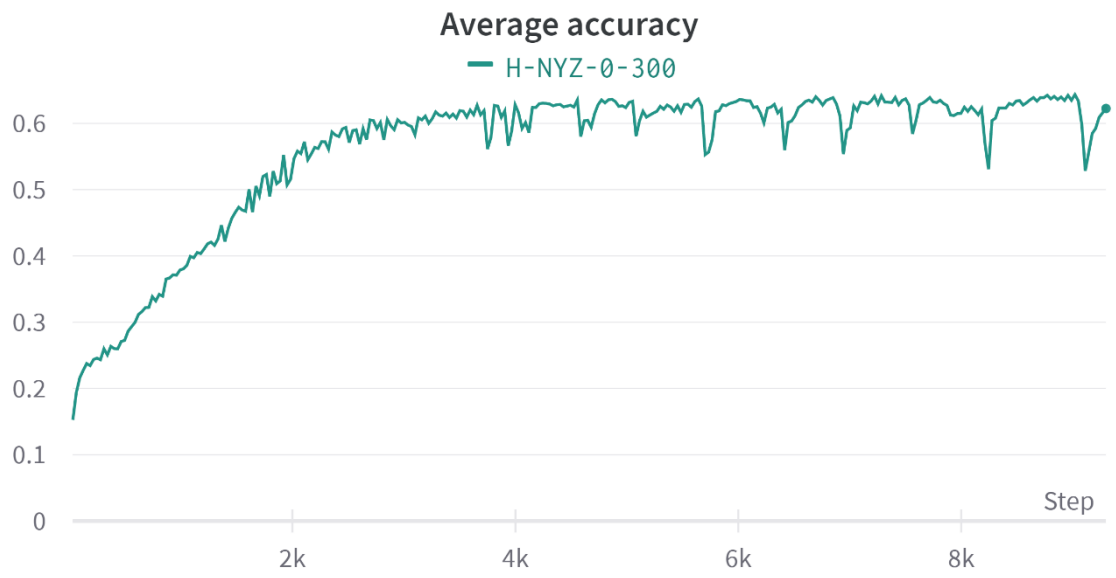
Terme	Représentation
H/P	H = Jeu de données des participants P = Jeu de données des patients
R/NY/NZ/NYZ	R = Données sans normalisation N(Y/Z) = Données normalisées sur l'axe (Y/Z)
0/LIN	0 = Valeurs non définies remplacées par des 0 LIN = Application de l'interpolation linéaire
AV	Si mentionnée, comprends l'accélération et la vélocité




Training loss
H-NZ-0
H-NY-0
H-NYZ-0
H-R-0
H-NY-0-AV
H-NZ-0-AV
H-NYZ-0-AV
H-R-0-AV
3
2
1
0
0
200
400
600
800
1k
1.2k
1.4k
Step


Average test loss
H-NZ-0
H-NY-0
H-NYZ-0
H-R-0
H-NY-0-AV
H-NZ-0-AV
H-NYZ-0-AV
H-R-0-AV
3
2.5
2
1.5
1
0.5
0
200
400
600
800
1k
1.2k
1.4k
Step

Entraînement plus long pour l'expérience qui requiert plus d'epoch pour atteindre sa précision maximum :

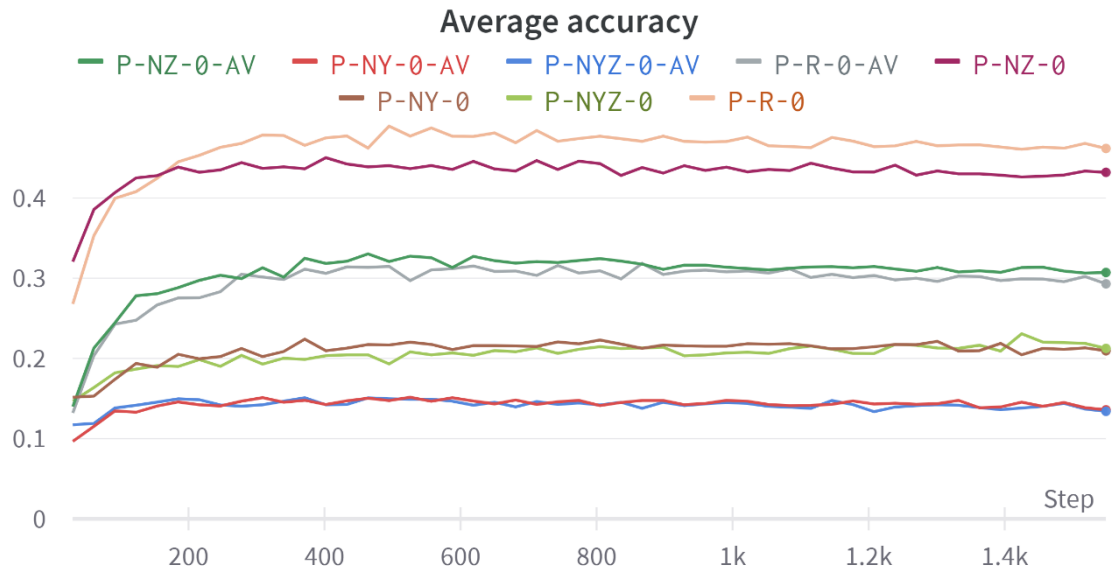


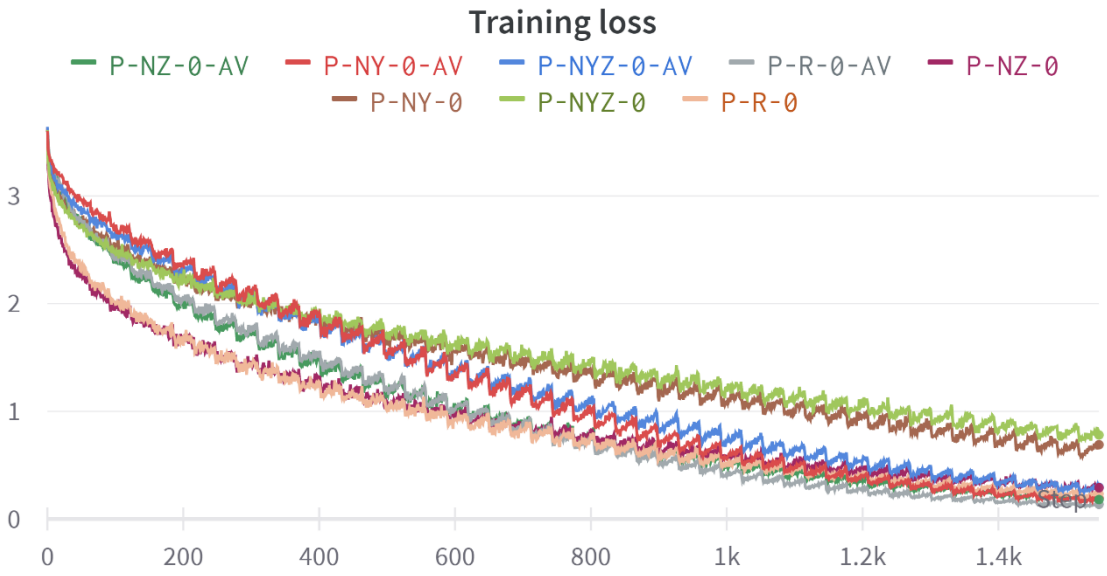
Annexe 2 : Graphiques du modèle sur le jeu de patients

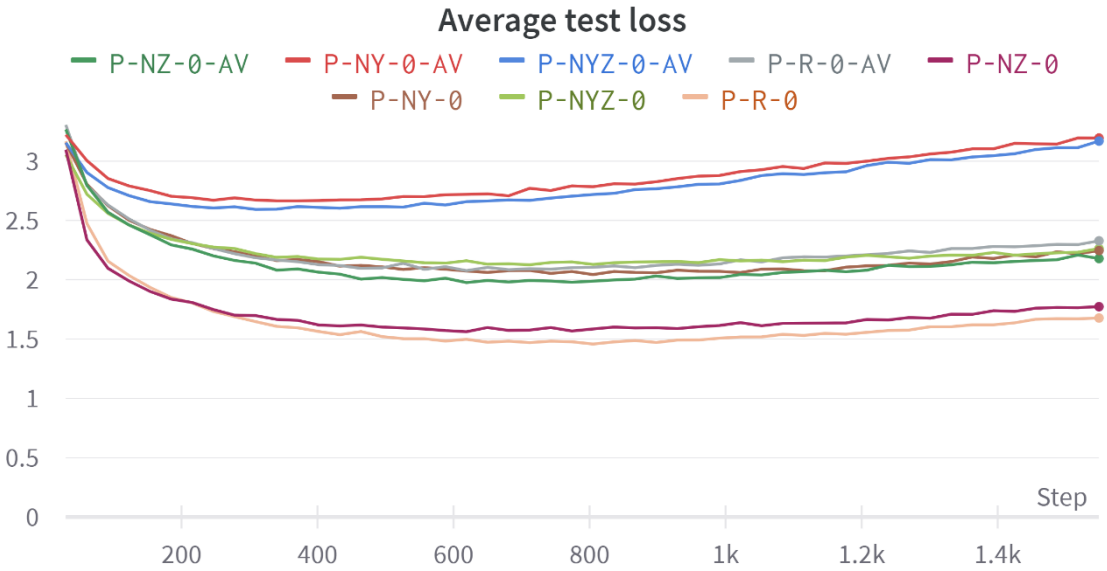
Rappel de la nomenclature :

Terme	Représentation
H/P	H = Jeu de données des participants P = Jeu de données des patients
R/NY/NZ/NYZ	R = Données sans normalisation N(Y/Z) = Données normalisées sur l'axe (Y/Z)
0/LIN	0 = Valeurs non définies remplacées par des 0 LIN = Application de l'interpolation linéaire
AV	Si mentionnée, comprends l'accélération et la vélocité

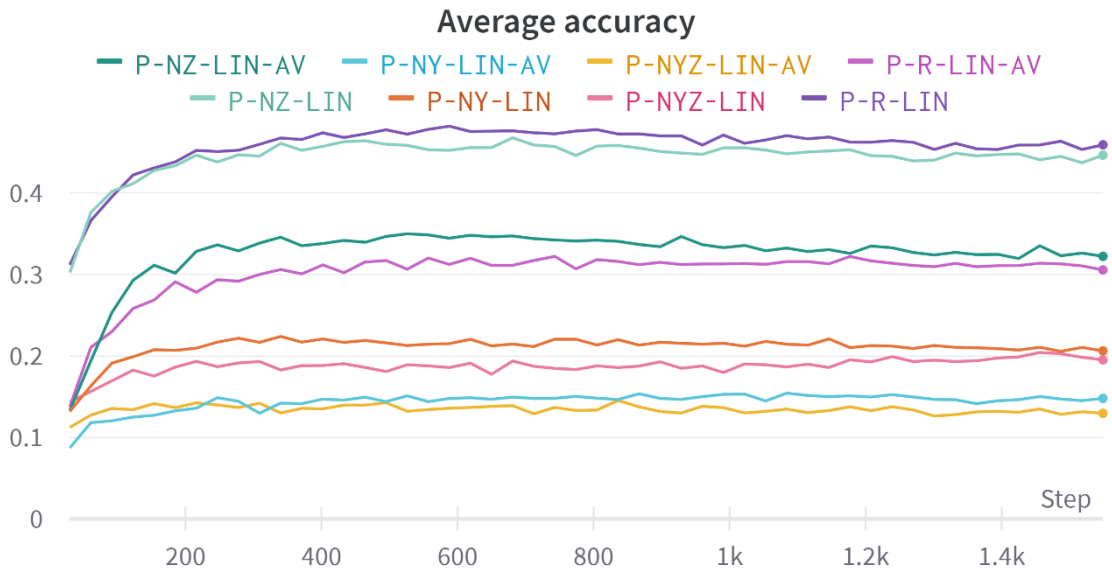
En remplaçant les valeurs non définies par des zéros :

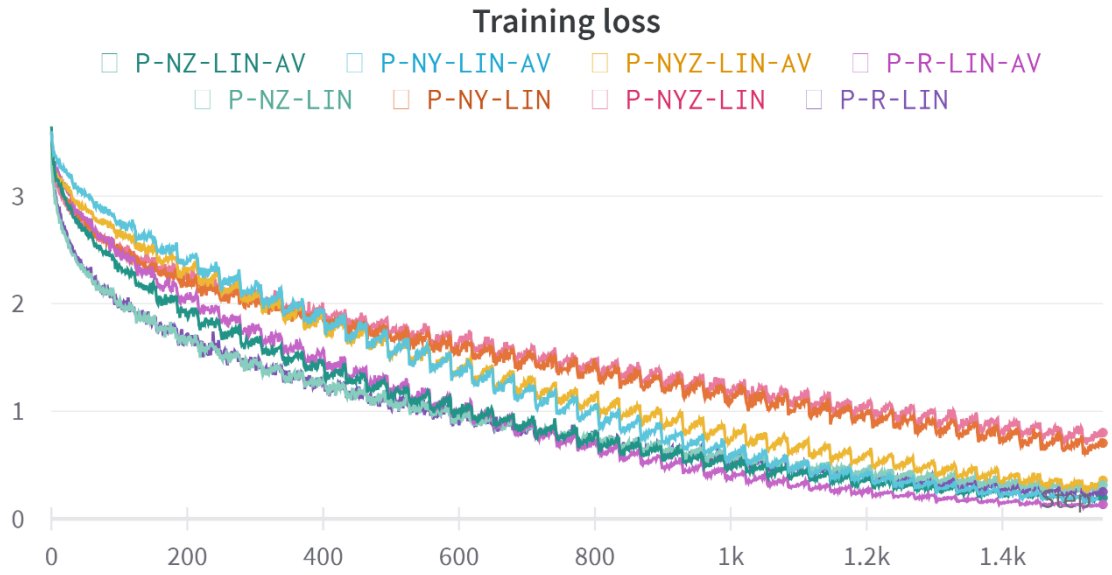


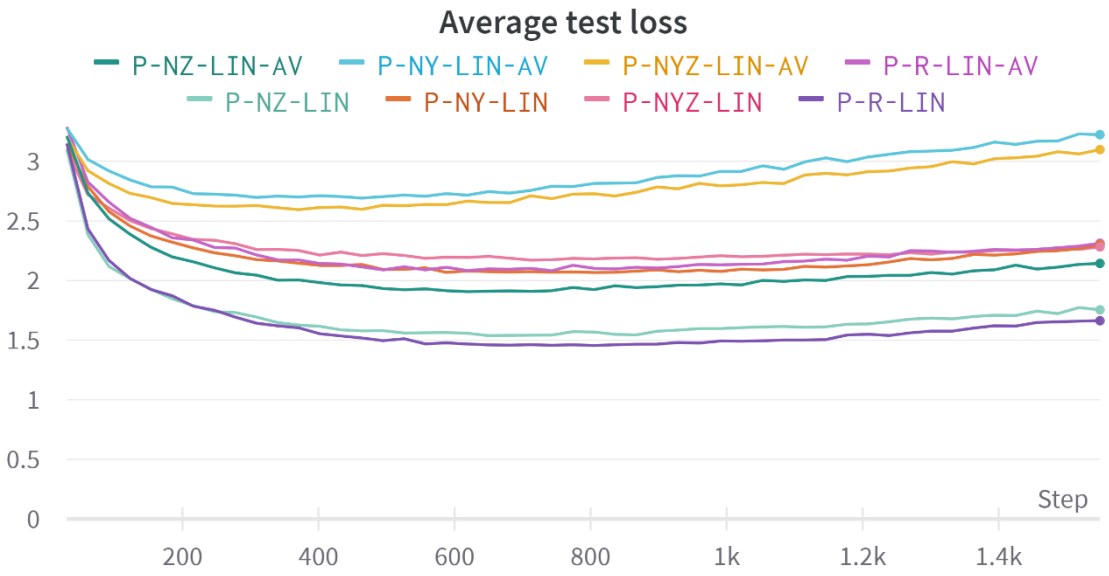

Training loss
P-NZ-0-AV
P-NY-0-AV
P-NYZ-0-AV
P-R-0-AV
P-NZ-0
P-NY-0
P-NYZ-0
P-R-0
3
2
1
0
0
200
400
600
800
1k
1.2k
1.4k


Average test loss
P-NZ-0-AV
P-NY-0-AV
P-NYZ-0-AV
P-R-0-AV
P-NZ-0
P-NY-0
P-NYZ-0
P-R-0
3
2.5
2
1.5
1
0.5
0
Step
200
400
600
800
1k
1.2k
1.4k

En remplaçant les valeurs non définies par de l'interpolation linéaire :






Average test loss
P-NZ-LIN-AV
P-NY-LIN-AV
P-NYZ-LIN-AV
P-R-LIN-AV
P-NZ-LIN
P-NY-LIN
P-NYZ-LIN
P-R-LIN
3
2.5
2
1.5
1
0.5
0
Step
200
400
600
800
1k
1.2k
1.4k