Projet de Fin d'Etudes
Mémoire Master II

Analyse de la Posture par Apprentissage Profond

Dulau Idris
Duhayon Rodin
Dubrasquet-Duval Guillaume

Lundi 28 Mars 2022

Table des matières

Introduction

La prise en charge d'un patient dans un parcours de soin peut être coûteuse en ressources humaines. L'apprentissage automatique apporte des solutions à des problématiques que rencontrent les professionnels de santé, notamment en assistant un diagnostic et en réduisant drastiquement la durée de tâches chronophages.

Présentation du Projet

L'analyse de la posture d'un patient en rééducation lors de l'exécution d'exercices est une tâche qui nécessite la présence d'au moins une personne habilitée durant toute la durée de l'exécution de ces exercices. Cette tâche nécessite également une grande attention de l'analyste durant ces exécutions.
Le temps du personnel n'est pas compressible et l'attention est un facteur variable. Le projet s'articule autour de ces deux critères. Une technologie permettant d'analyser la posture d'une personne durant un mouvement pourrait permettre la mise en place d'un soutien dans l'autogestion à long terme des exécutions des exercices de rééducation pour les patients. Les analyses sont automatisées et leurs qualités restent constantes. La détection de diverses caractéristiques posturales peut être pertinente : des mouvements réflexes, des comportements stéréotypés, une altération des performances motrices, une modification de la tension musculaire. Ainsi Kleinsmith et al.[1], Randhavane et al.[2], et Luo et al.[3] ont montré que les mouvements et les postures du corps contiennent des informations sur la personne.

Le projet, via un réseau de neurones, avec l'utilisation de données sur la posture issues de vidéos de patients en rééducation lors de l'exécution d'exercices, et par l'extraction des caractéristiques posturales telles que les positions des articulations dans l'espace ou les orientations de ces articulations dans l'espace, devra permettre de déterminer la bonne ou la mauvaise exécution d'un mouvement.

Pour réaliser ce projet, dans un premier temps, nous avons effectué une **Analyse** du contexte dans lequel il s'inscrit, une analyse de l'état de l'art ainsi qu'une analyse des besoins. Le domaine de l'apprentissage profond appliqué à la vision par ordinateur est riche d'informations en libre accès. Cette phase d'analyse nous a permis de prendre connaissance de différentes techniques utilisées à des fins proches de la nôtre. Dans un même temps, sur demande du client, nous avons analysé et récupéré une base de code d'un projet de reconnaissance d'actions humaines ainsi qu'une base de données composée de coordonnées issues de vidéos d'exercices de rééducation.
Ensuite, dans l'objectif d'utiliser la base de code précédemment récupérée, nous sommes passés par une phase de **Conception** du projet dans laquelle nous avons utilisé l'architecture du projet de reconnaissance d'actions humaines pour concevoir l'architecture de notre projet. Dans un même temps, nous avons pu constater les modifications nécessaires à son installation sur nos machines et sur celles du CREMI. De l'étude de la base de données composée de coordonnées issues de vidéos d'exercices de rééducation que nous voulons utiliser, et vis-à-vis des bases de données utilisées par le code du projet de reconnaissance d'actions humaines, nous avons pu prévoir les ajouts nécessaires à l'utilisation de notre base de données dans le code existant.
Suite à ça, dans le chapitre **Réalisation**, nous avons mis en place les modifications évoquées dans la partie Conception. Les changements apportés à la base de code déjà acquise et les ajouts de code constituent la majeure partie du travail à réaliser et sont décrits en suivant l'ordre d'exécution du programme. Dans un premier temps, nous avons mis en place les modifications nécessaires à l'installation du code récupéré sur nos machines et sur celles du CREMI. Nous avons formaté les données de notre base de données pour obtenir une entrée conforme au réseau de neurones que nous utilisons. Nous avons modifié le code en fonction des nouvelles données. Toutes les modifications nous permettent de passer d'une classification multi-classes qui reconnaît des actions en se basant sur des données en deux dimensions à une classification binaire qui reconnaît la qualité d'actions en se basant sur des données en trois dimensions. Une partie dédiée aux tests vient compléter et ajuster les choix d'implémentation. Les modifications apportées au code existant affectent l'entièreté de celui-ci, et, pour valider nos résultats, il est important de s'assurer de leurs bons fonctionnements. Dans le cadre de l'étude d'un réseau de neurones, nous évaluons ses performances par sa rapidité et la qualité de la sortie attendue et nous mettons en relation ces données avec nos capacités matérielles.
Enfin la **Conclusion** permettra de mettre en évidence ce qui a été réalisé et de discuter des potentielles améliorations pour notre projet.

Organisation du Projet

Dans le cadre d'un suivi de projet, et comme vu dans le cours qui va de pair avec le PFE, nous avons utilisé une méthode agile, la méthode SCRUM. L'entièreté du travail à réaliser pour finaliser le projet est répartie sous forme de tâches d'une durée estimée inférieure à une demi-journée. La répartition des tâches se fait entre les membres du groupe et à chaque semaine correspond un nombre de tâches à effectuer.

Dans l'objectif de définir clairement les besoins associés au projet, deux réunions préliminaires ont été effectuées avec les clients. Par la suite, au moins une réunion a été faite chaque semaine avec les clients pour présenter l'avancement des tâches effectuées. Ces réunions nous ont permis d'obtenir des retours sur nos tâches réalisées et de modifier des tâches futures.

Pour collaborer sur le projet, nous avons utilisé plusieurs outils. Le stockage du code a été réalisé sur un dépôt GitHub privé, la communication entre les membres du groupe s'est faite en message écrit et en appel vocal via Discord, le stockage des dossiers volumineux a été réalisé au CREMI sur les espaces dédiés. La mise en place d'une installation du projet au CREMI nous a également permis d'effectuer chacun des tests en utilisant une même architecture.

La mise en place du diagramme de Gantt prévisionnel, Figure 1 Page 5, découle de l'identification des *User Story* relatives au projet qui sont les suivantes :

En tant que programmeur, je souhaite analyser et comprendre le contexte dans lequel s'inscrit le projet pour déterminer au mieux les choix à faire lors des parties de conception et de réalisation.

En tant que programmeur, je souhaite intégrer les notions relatives à l'article et au code de l'auteur Plizzari afin de pouvoir me baser sur ses recherches pour les adapter à mes besoins.

En tant que programmeur, je souhaite intégrer les notions relatives à l'article de l'auteur Vakanski ainsi qu'au reste de l'état de l'art afin de pouvoir comprendre les évolutions qui nous ont amenées à utiliser un réseau de type Transformer pour ce projet.

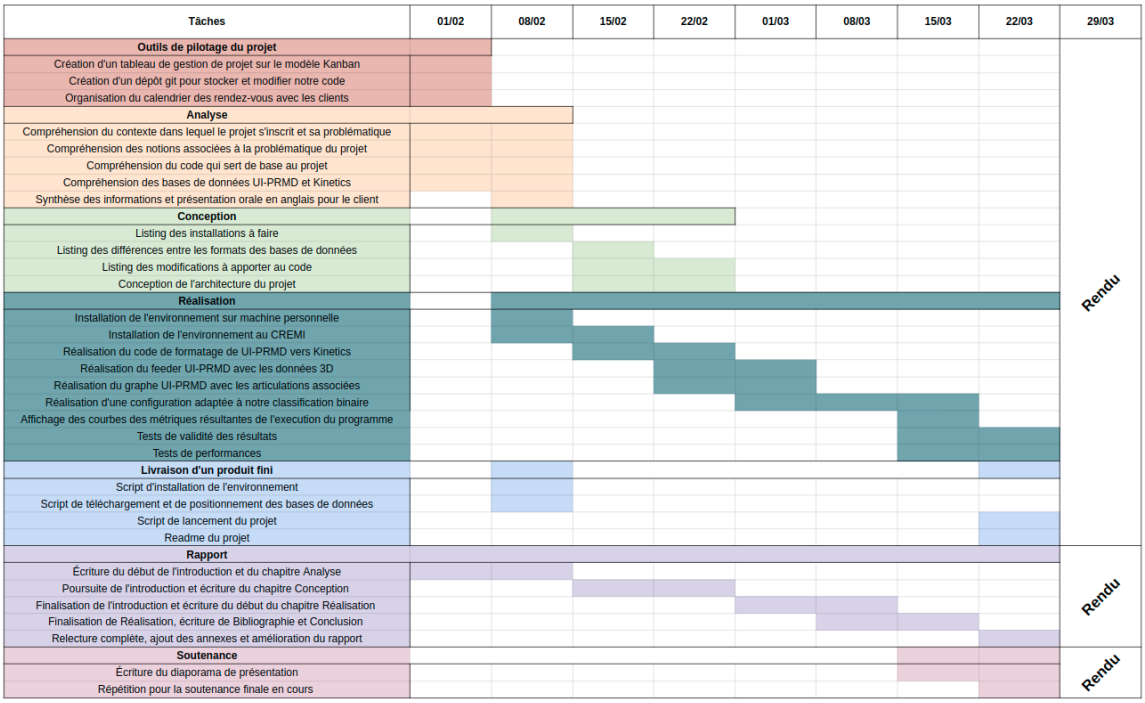

FIGURE 1 – Diagramme de Gantt prévisionnel

En tant que programmeur, je souhaite comprendre les différences entre les formats des bases de données utilisées dans l'article de l'auteur Plizzari et la base de données imposée à l'utilisation pour pouvoir formater cette dernière comme nécessaire.

En tant que programmeur, je souhaite faire fonctionner le code de l'auteur Plizzari afin d'avoir une base de code fonctionnelle sur laquelle m'appuyer pour démarrer mon implémentation.

En tant que spécialiste de la posture, je souhaite avoir confiance en la véracité des résultats du logiciel afin de céder l'analyse posturale lors des exercices des patients à ce logiciel.

En tant que client du projet, je souhaite obtenir un produit fini afin de pouvoir utiliser le produit demandé immédiatement et sans ajout.

Chapitre 1

Analyse

Dans le cadre du démarrage d'un projet nouveau, une phase d'analyse est primordiale au bon déroulé des événements. Ce chapitre est dédié à la mise en avant des notions clé du projet, par la façon dont elles s'inscrivent dans celui-ci, et par l'explication des domaines qui leur sont connexes. En conséquence, ce chapitre s'articule autour de trois axes, une partie contexte qui va définir ce qu'il faut faire lors du projet, une partie état de l'art qui va mettre en avant des éléments connexes au sujet et une partie bases de données qui va détailler les éléments importants de ces dernières.

1.1 Contexte

Le projet, via un réseau de neurones, avec l'utilisation de données sur la posture issues de vidéos de patients en rééducation lors de l'exécution d'exercices, et par l'extraction des caractéristiques posturales telles que les positions des articulations dans l'espace ou les orientations de ces articulations dans l'espace, devra permettre de déterminer la bonne ou la mauvaise exécution d'un mouvement.

Les données d'étude à utiliser comme entrée du réseau sont imposées par le client, une base de données composée de points d'ancrage représentant des positions d'articulations dans l'espace et les orientations de ces articulations dans l'espace qui se nomme UI-PRMD [4]. Cette base de données constituée uniquement des informations de positions et d'angles d'Euler a été générée par l'extraction préalable des données sur la base de données de vidéos des exercices en question.

Les informations de la base de données sur laquelle apprend le réseau doivent être introduites sous forme d'un graphe représentant la posture humaine.
Le modèle de réseau de neurones à utiliser lors du traitement des données est défini par le client, une architecture de type Transformer a été choisie, une base de code nous est assignée pour débuter l'implémentation.
La représentation des informations générée par le réseau est laissée libre par le client, mais se doit d'être pertinente et justifiée.

Précisément, on met en place un modèle de réseau de neurones selon une architecture de type Transformer qui traite des données sous forme de graphes issus de points d'ancrage extraits des vidéos de la base de données UI-PRMD. La demande explicite des clients d'utiliser une architecture de type Transformer et des données d'entrée sous forme de graphe a pour but le test des capacités de ce type de modèle et l'évaluation de sa pertinence pour ce type de tâche et de la faisabilité de sa mise en place dans ce cas de figure.

1.2 Etat de l'art

1.2.1 Les Réseaux de Neurones Récurrents

Un réseau de neurones est dit récurrent s'il présente des connexions récurrentes ; s'il existe au moins un cycle dans sa structure. Cette singularité rend le réseau adapté à l'analyse de séries chronologiques. Cependant, lors de la rétro-propagation du gradient, le réseau peut être gêné par des problèmes de faible convergence ou de forte divergence de la solution. Le problème de disparition du gradient fait référence à la diminution de la valeur des gradients au fil du temps. Les poids sont donc faiblement mis à jour par descente de gradient ralentissant de manière croissante la convergence vers la solution. Le problème d'explosion du gradient fait référence à l'augmentation de la valeur des gradients au fil du temps. Les poids sont trop fortement mis à jour par descente de gradient entraînant une trop forte divergence de l'algorithme. Ces deux problèmes gênent donc l'apprentissage de longues séquences de données.

Les réseaux de type LSTM

Un type de réseau de neurones récurrent vient pallier à ces problèmes, les LSTM[5] pour Long Short-Term Memory. Par l'ajout de portes et d'états supplémentaires aux RNN classiques, ces réseaux agissent sur les informations qu'ils gardent en mémoire. Différentes versions existent, leurs principes sont similaires. La mise à jour des poids

qui n'était pas régulée peut désormais se voir bloquer, la porte d'entrée contrôle les informations stockées dans l'état mémoire. La porte de sortie contrôle les informations envoyées au pas suivant du réseau. La porte d'oubli sélectionne les informations à garder et à supprimer de la mémoire en fonction de nouvelles informations amenées par la porte d'entrée. La gestion de la mémoire à chaque étape du processus d'apprentissage par cette dernière porte permet au réseau de converger vers la solution désirée à partir du gradient d'erreur. Les LSTM, par leur architecture, résolvent ainsi le problème de la dépendance à long terme et peuvent donc traiter des séquences de données plus grandes que les RNN classiques.

Les LSTM ont déjà été utilisés avec succès pour l'analyse de la posture humaine[6]. Cependant, des modèles plus récents permettent d'obtenir des résultats supérieurs pour le traitement de séquences de données, nous nous intéresserons donc à ceux-ci.

Les mécanismes d'attention

Les mécanismes d'attention[7] apparaissent postérieurement aux réseaux précédemment cités et améliorent le modèle d'apprentissage automatique existant qui se nomme l'encodeur-decodeur. Ce dernier est constitué de deux réseaux de neurones, un RNN appelé encodeur qui va encoder une information en fonction de son entrée, suivi d'un décodeur qui va produire une sortie en fonction de cette information. Dans sa version originale, l'encodeur produit une information de taille fixe et des pertes d'informations se produisent pour traiter une longue séquence de données. Des évolutions de ces fonctionnements permettent de lever ces limitations et ce sont les mécanismes d'attention. Le nouveau procédé conserve le principe des couches RNN où chaque pas est utilisé pour calculer la sortie du pas suivant tout en gardant les sorties récurrentes de chaque pas en mémoire. L'encodeur ainsi défini par les couches RNN couplées à l'attention va transmettre beaucoup plus d'information au décodeur que le principe des couches RNN seul. On augmente le volume du contexte transmis au décodeur et ainsi, on améliore sa sortie.

Une nouvelle architecture appelée Transformer[8] a montré que les performances des mécanismes d'attention couplés aux RNN étaient équivalentes à celles des mécanismes d'attention seuls.

1.2.2 Les réseaux de type Transformer

Basé sur les mécanismes d'attentions, le transformer se départit des couches RNN et conserve le principe de l'encodeur-décodeur. La partie qui concerne l'encodage de l'information contient une succession d'encodeurs telle que l'entrée d'un encodeur est la sortie de celui qui le précède. La partie qui concerne le décodage de l'information contient une succession de décodeurs telle que l'entrée d'un décodeur est la sortie de celui qui le précède ainsi que la sortie du dernier encodeur. Les encodeurs sont constitués d'une couche d'attention et d'un réseau à propagation avant. Il en est de même pour les décodeurs en plus d'avoir, entre ces deux, une couche d'attention liée à la sortie du dernier encodeur. L'attention appliquée au sein des Transformers est nommée auto-attention et n'est autre que le mécanisme d'attention précédemment décrit appliqué à une seule séquence. Elle détermine l'interdépendance des éléments d'une séquence pour en associer une représentation la plus pertinente possible.

À une période très proche de l'apparition des Transformers, est publié un article[9] traitant d'une approche couplant des réseaux de neurones à convolution et des données d'entrée sous forme de graphe pour analyser des actions humaines basées sur le squelette humain.

1.2.3 Le réseau ST-GCN

La généralisation des réseaux de neurones aux données avec des structures de graphes est un sujet émergent dans la recherche en apprentissage profond. Le réseau ST-GCN pour Spatial Temporal Graph Convolutional Networks a montré son efficacité sur les données de l'état de l'art. Il surpasse les autres méthodes en apprenant des données à la fois spatiales et temporelles.

Construction du graphe

Le traitement des séquences vidéos permet d'extraire les coordonnées de points d'ancrage sur le corps humain ainsi que les liens entre ces points, et cela, pour chaque image de la séquence. L'état de l'art concatène les données spatiales que sont les coordonnées des points d'ancrage et leurs liens pour obtenir un unique vecteur de données correspondant à une image, cela pour chaque image de la séquence. Dans le cas de l'article présent, la construction d'un graphe avec ces mêmes données se fait de manière différente. Dans un premier temps, comme pour l'état de l'art, chaque point d'ancrage dans le domaine spatial est un noeud relié par des arêtes que sont les relations entre ces points d'ancrage, formant pour chaque image de la séquence

un graphe dans le domaine spatial. Ensuite, la nouveauté, chaque noeud du graphe est relié par une arête à lui-même dans l'image suivante. On obtient ainsi un unique graphe pour lequel chaque noeud à un ou plusieurs voisins dans le domaine spatial et un unique voisin dans le domaine temporel.

Utilisation du graphe dans le réseau

Le réseau prend en entrée les données constitutives du graphe sous forme de vecteurs de coordonnées. Le graphe est ensuite construit comme expliqué précédemment. Des opérations de convolution sont ensuite appliquées aux noeuds du graphe dans le domaine spatial. Le principe est similaire à une convolution sur une grille en deux dimensions qui est faite sur les pixels voisins au centre de la grille ; comme les distances entre les pixels sont équivalentes, un ordre pour effectuer la convolution est aisément défini. Sur le graphe, un échantillonnage et une pondération des noeuds sont nécessaires pour définir les grilles de convolutions dans le domaine spatial. Pour étendre les convolutions au domaine temporel, comme pour la construction du graphe, le voisinage d'un noeud est étendu à ce même noeud à une ou plusieurs images consécutives au besoin. L'application de ces convolutions se fait ensuite selon des stratégies de partitionnement qui peuvent être diverses.

Suite à cela, l'article[10] sur lequel se base notre projet et duquel nous avons récupéré du code mêle les principes d'utilisation des graphes du ST-GCN en utilisant une architecture de type Transformer.

1.2.4 Le réseau ST-TR

Le réseau ST-TR est introduit dans l'objectif de lever certaines limitations existantes dans l'état de l'art en terme de reconnaissance d'action humaine via l'utilisation de données sous forme de graphe.

Le réseau ST-GCN vu précédemment fait partie de cet état de l'art et permet de traiter le problème. Bien que démontré efficace vis-à-vis de ses contemporains, il présente des limitations.
Dans un premier temps, la représentation du graphe modélisant le squelette humain est fixée et invariante pour toutes les couches du modèle et toutes les actions effectuées. La variabilité de cette représentation pourrait permettre d'extraire plus de données et donc de former des représentations plus riches pour les mouvements du

squelette au cours du temps. Le phénomène s'accroît si les liens du graphe sont dirigés et que les informations ne peuvent circuler que le long d'un chemin prédéfini.
Ensuite, comme vu précédemment dans le réseau ST-GCN, les convolutions spatiales et temporelles sont implémentées à partir d'une convolution en deux dimensions. Les opérations sont donc limitées par la taille du noyau de convolution.
Enfin, la représentation du squelette sous forme de graphe ne tient pas directement compte des corrélations entre les articulations qui ne sont pas jointes dans le squelette humain. Il n'y a par exemple pas d'arête reliant le pied droit au pied gauche sans passer par un chemin composé d'autres noeuds qui serait par exemple le genou droit puis la tête du fémur droit, la tête du fémur gauche et enfin le genou gauche. Ainsi, les informations lors d'actions engageant majoritairement des extrémités lors du mouvement ne sont pas estimées à leur juste valeur pour des cas dans lesquels ce serait extrêmement pertinent.

Le réseau ST-TR fait face à ces limitations en substituant les convolutions en deux dimensions dans les domaines spatiaux et temporels par l'opérateur d'auto-attention du Transformer. À l'origine, l'auto-attention du Transformer permet l'encodage des corrélations à court et à long terme entre les mots d'une phrase. Considérant les articulations composant un squelette comme un ensemble de mots, la même approche peut être appliquée à la reconnaissance d'actions basée sur ce squelette pour extraire les relations entre les articulations environnantes. Là où la convolution de graphe précédemment citée, où seuls les noeuds adjacents sont comparés est limitante, cette nouvelle approche se départit de toute structure de squelette prédéfinie et laisse à la place l'auto-attention du Transformer découvrir automatiquement les relations entre les articulations qui sont pertinentes pour prédire l'action en cours. L'opération résultante agit de manière similaire à une convolution de graphe, mais dans laquelle les valeurs du noyau sont dynamiquement prédites sur la base des relations entre les articulations qui ont été découvertes. De même, à la manière dont les relations entre les phrases sont construites, ce mécanisme d'auto-attention est appliqué au niveau de la séquence d'images pour analyser l'évolution de chaque articulation au cours de l'action et en construisant des relations à longue portée qui couvrent différentes images. Les corrélations entre les noeuds sont cruciales à la fois sur la dimension spatiale et sur la dimension temporelle et l'opérateur résultant est capable d'obtenir une représentation dynamique s'étendant sur ces deux dimensions.

L'extraction des caractéristiques des squelettes au cours d'une action est dans un premier temps traitée de façon disjointe sur les deux dimensions, d'abord spatiale, ensuite temporelle. Ensuite, une combinaison des résultats peut être effectuée.

Auto Attention Spatiale

L'extraction des caractéristiques des squelettes au cours d'une action est traitée dans la dimension spatiale par le module d'auto attention spatiale qui applique l'auto-attention à l'intérieur de chaque image pour extraire des caractéristiques de bas niveau intégrant les relations entre les articulations du corps humain. Ceci est réalisé en calculant indépendamment les corrélations entre chaque paire d'articulations dans chaque image, comme illustré sur la Figure 2 Page 12. Les annotations Q, K, V sur les Figures 2 et 3 correspondent aux termes *Query*, *Key*, *Value* qui permettent le calcul des coefficients d'attention.

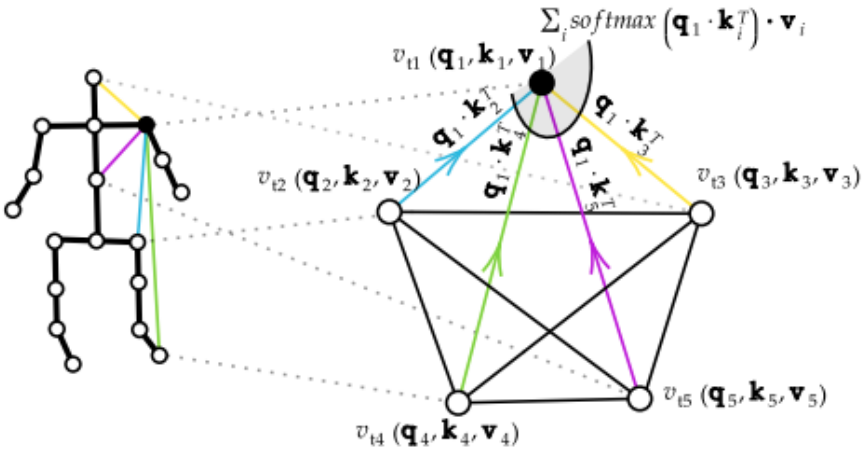

FIGURE 2 – Auto Attention Spatiale

Auto Attention Temporelle

L'extraction des caractéristiques des squelettes au cours d'une action est traitée dans la dimension temporelle par le module d'auto attention temporelle qui considère chaque articulation comme indépendante le long d'une séquence d'image et calcule les corrélations entre les images en comparant l'évolution des positionnements d'une même articulation au fil des images de la séquence et donc du temps comme illustré sur la Figure 3 Page 13.

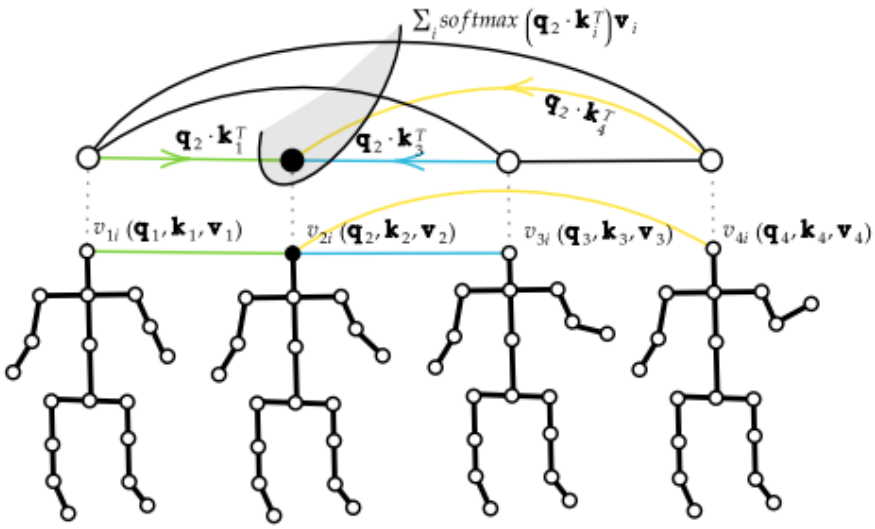

FIGURE 3 – Auto Attention Temporelle

Les performances du réseau ST-TR ont été évaluées sur trois bases de données qui constituent l'état de l'art dans le domaine de la reconnaissance d'action basée sur le squelette. Ces bases de données se nomment Kinetics 400 [11], NTU RBG D 60 [12] et NTU RBG D 120 [13].

1.3 Bases de données

Les bases de données précédemment citées contiennent des vidéos d'actions humaines, qu'il s'agisse d'actions entre humains ou d'actions qui font intervenir un humain et un ou plusieurs objets. Dans l'objectif d'étudier les points d'ancrage qui représentent des squelettes, nous avons récupéré les données disponibles sous cette forme et non les données vidéos.

1.3.1 Kinetics

La base de données Kinetics est utilisée dans le code et dans l'article. L'extraction des données sous forme de points d'ancrage a été réalisée via la technologie Openpose [14] sur des extraits vidéos représentants des actions sur une durée d'environ 10 secondes. La base de données est constituée de 18 positions d'articulations dans un

espace en deux dimensions par image d'une séquence. Les données sont représentées au format Json.

La structure des données est représentée Figure 4 page 14, la *pose* correspond aux coordonnées en deux dimensions, le *score* correspond à l'intervalle de confiance associé à chaque couple de coordonnées. Pour une même image d'une séquence, si plusieurs humains sont détectés, plusieurs couples *pose*/*score* seront présents.


```
{"data": [{"frame_index": 1, "skeleton": [{"pose": [0.514, 0.
465, 0.561, 0.707, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
782, 0.720, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
477, 0.435, 0.577, 0.435, 0.436, 0.489, 0.712, 0.462], "score":
[0.151, 0.064, 0.000, 0.000, 0.000, 0.195, 0.000, 0.000, 0.000,
0.000, 0.000, 0.000, 0.000, 0.000, 0.319, 0.536, 0.073, 0.319]}
, {"pose": [0.241, 0.446, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.213, 0.427, 0.000, 0.000, 0.068, 0.443, 0.
000, 0.000], "score": [0.099, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
430, 0.000, 0.153, 0.000]}]}, {"frame_index": 2, "skeleton": [
{"pose": [0.542, 0.451, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.
```


FIGURE 4 – Fichier descriptif des coordonnées et de leurs scores

La base de données est certes énorme, mais n'a pas été générée par des experts, de cela résulte des problèmes d'acquisition. Le second fichier, Figure 5 page 15 permet de stipuler la présence des données squelettiques ou non pour chaque fichier, en plus de récapituler les labels de ceux-ci.

FIGURE 5 – Fichier récapitulatif des labels

1.3.2 NTU RGB D

La base de données NTU RBG D 60 et son extension NTU RBG D 120 sont également utilisées dans le code et dans l'article. Les données vidéos pour constituer ces deux bases de données ont été capturées par des caméras Kinect sur des extraits représentants des actions d'une durée de quelques secondes. Les bases de données sont constituées de 25 positions d'articulations dans un espace en trois dimensions par image d'une séquence. Les données sont représentées au format Skeleton.

La structure des données est représentée Figure 6 page 16, le chiffre de la première ligne correspond au nombre d'images acquises pour l'action décrite par le fichier et n'apparaît qu'au début de celui-ci. Le chiffre de la seconde ligne correspond au nombre de personnes identifiées sur l'action et donc au nombre d'ensembles de coordonnées correspondant à une image de la séquence. La troisième ligne contient des informations relatives au référencement du fichier. La quatrième ligne correspond au nombre de points d'ancrage des données squelettiques, ici 25 qui restera inchangé. Les 25 lignes qui suivent correspondent aux coordonnées acquises via différents systèmes, seules les trois premières de chaque ligne nous intéressent. À l'exception de la première ligne, les autres précédemment décrites forment un bloc d'information correspondant à une image de la séquence et se répètent avec des valeurs différentes, autant de fois que notifié à la première ligne, pour former la séquence entière.

```
71
1
72057594037937714 0 1 1 1 1 0 0.07263379 -0.1667504 2
25
0.5804712 0.5633132 4.306256 305.7654 160.5389 1116.242 429.3723 0.01253613 0.01632788 0.9746307 -0.2228706 2
0.5901552 0.7962888 4.193919 308.0093 138.7896 1123.024 366.9543 0.006736697 0.01765371 0.974688 -0.2227695 2
0.5956972 1.026487 4.073137 310.1296 115.8209 1129.46 301.1804 -0.001308254 0.0115397 0.966922 -0.2548074 2
0.5924639 1.142216 3.994447 310.9379 103.2515 1132.028 265.2123 0 0 0 0 2
0.4377418 0.951628 4.113326 295.4399 123.512 1087.151 322.9549 0.172019 0.7294973 -0.6433109 0.1561871 2
0.3879205 0.7577295 4.231327 289.9706 142.8203 1071.038 378.2323 0.127593 0.7250928 -0.2579305 -0.6256456 2
0.3516378 0.5691128 4.281889 286.4291 159.8105 1060.651 427.0083 0.1584786 0.829938 -0.007929943 0.5348126 2
0.3564328 0.5229455 4.304696 286.6709 164.007 1061.267 439.0871 0.1663431 0.8229803 0.1611727 0.5187066 1
0.7538905 0.9539624 4.109423 323.8196 123.1181 1168.621 322.3234 0.1651172 0.7221254 0.6514535 -0.1639497 2
0.808997 0.751578 4.210184 326.9485 142.9061 1177.321 379.0929 0.1950686 0.9475425 0.1645761 0.192422 2
0.7984567 0.5632423 4.272752 324.9383 160.1111 1171.396 428.4411 0.1348238 0.7204674 0.08938837 0.6743581 2
0.7829522 0.5047072 4.292256 323.2773 165.3515 1166.577 443.4725 0.1986825 0.6964517 0.02164717 0.6892108 1
0.5074601 0.5644862 4.270454 299.8998 160.0497 1099.48 427.8799 0.01096952 -0.634976 0.7121037 -0.2993217 2
0.5204908 0.2933539 4.392238 299.734 184.0428 1098.58 496.9549 -0.164466 -0.6943063 0.1314756 0.6881889 2
0.5369848 0.1025284 4.559347 299.4533 200.2653 1097.217 543.7271 -0.2709459 -0.6556568 0.2263313 0.6674404 2
0.5721567 0.1726865 4.470084 303.2076 194.3525 1108.31 526.7107 0 0 0 0 2
0.6449077 0.553777 4.278249 311.6276 161.0275 1133.176 430.8652 -0.00554263 0.6932401 0.6578163 -0.2943893 2
0.7245586 0.2917787 4.392901 316.7986 184.1522 1147.655 497.4686 -0.003641876 0.541504 0.242323 0.8050091 2
0.6894395 0.116369 4.573092 311.5663 199.1753 1132.047 540.6924 0.3191 0.6297009 0.2296948 0.6699941 2
0.6876037 0.1853381 4.471811 312.6755 193.3129 1135.546 523.8088 0 0 0 0 2
0.5942783 0.9694726 4.105435 309.5569 121.6936 1127.721 317.9866 -0.001169776 0.01118248 0.9721881 -0.2339315 2
0.3454388 0.4680133 4.338236 285.4996 169 1057.773 453.448 0 0 0 0 2
0.3853926 0.5120633 4.279 289.3336 164.6683 1069.018 441.0239 0 0 0 0 2
0.7751886 0.4423845 4.304794 322.4018 170.8 1164.026 459.1228 0 0 0 0 2
0.8000001 0.4777714 4.282778 324.8871 167.563 1171.223 449.8534 0 0 0 0 2
1
72057594037937714 0 1 1 1 1 0 0.07521217 -0.1707789 2
25
```

FIGURE 6 – Fichier descriptif des coordonnées

1.3.3 UI-PRMD

La base de données imposée par le client pour servir d'entrée au modèle se nomme UI-PRMD. Elle compile des données de mouvements liés à des exercices couramment effectués par des patients dans le cadre de programmes de kinésithérapie et de réadaptation. Elle est constituée de données relatives à 10 mouvements de rééducation. Un échantillon de 10 personnes en bonne santé a effectué chaque mouvement 10 fois devant deux systèmes de capture de mouvement, une caméra Vicon et une caméra Kinect. Les données sont présentées sous forme de positions et d'orientations des articulations du corps humain dans l'espace au format texte pour chaque système de capture. Les données récupérées par la caméra Vicon constituent 39 points d'ancrage dans un espace en trois dimensions par image d'une séquence. Les données récupérées par la caméra Kinect constituent 22 points d'ancrage dans un espace en trois dimensions par image d'une séquence. Les données sont labellisées en fonction de l'exécution du mouvement qui peut être correct ou incorrect. Les données sont ainsi triées par label, puis par système de capture vidéo et enfin par les caractéristiques posturales qu'elles représentent. Les 10 répétitions effectuées par un sujet pour un exercice sont accessibles, soit par un seul fichier représentant les 10 répétitions, soit par 10 fichiers représentant une répétition. L'architecture de la base de données est présentée Figure 7 page 17, les encadrés rouges représentent les dossiers qui nous intéresserons par la suite.

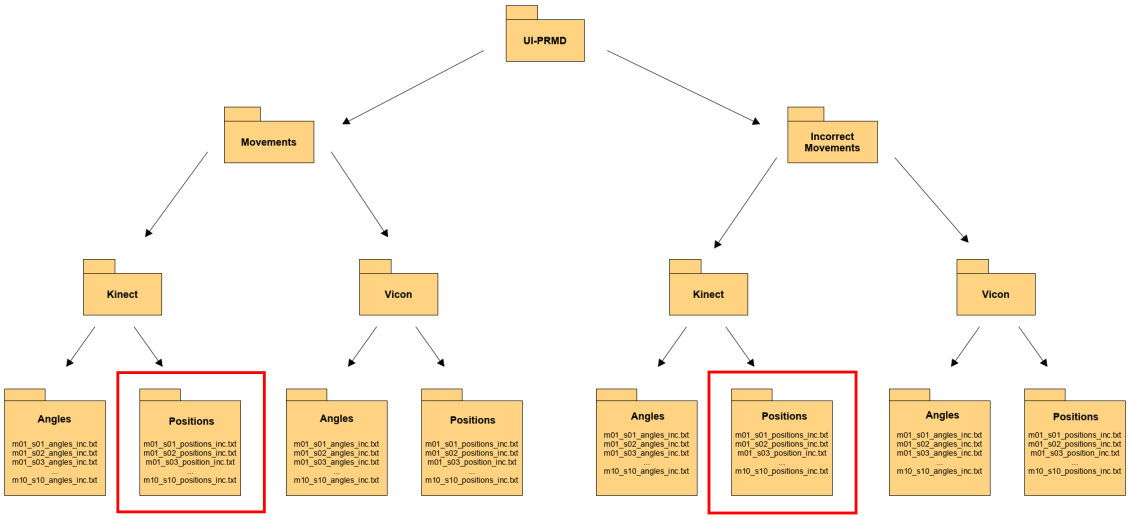

FIGURE 7 – Fichier descriptif de l'architecture de UI-PRMD

La structure des données pour un fichier acquis par Kinect est représentée Figure 8 page 18, une unique personne est représentée dans chaque mouvement et de par l'expertise des personnes ayant acquis les données tous les squelettes sont présents. Chaque ligne contient 66 valeurs correspondant à 22 triplets de coordonnées, un par point d'ancrage. Chaque ligne décrit une image de la séquence.

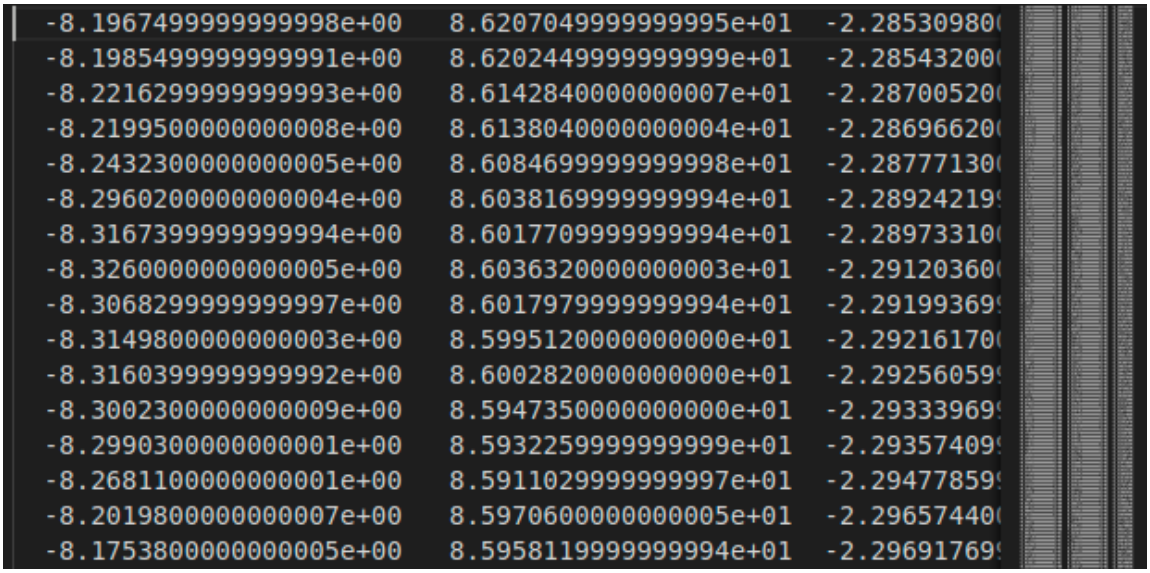

FIGURE 8 – Fichier descriptif des coordonnées

1.4 Besoins

1.4.1 Besoins Fonctionnels

La contextualisation du projet et de sa problématique nous permet d'énoncer les besoins fonctionnels qui lui sont associés.

- **Installation** des bibliothèques.
- **Formatage** des données d'entrée du réseau.
- **Exécution** du code avec les nouvelles données d'entrée.
- **Production** d'un résultat en sortie du modèle.
- **Visualisation** du résultat.

1.4.2 Besoins non-Fonctionnels

Des besoins fonctionnels précédemment énoncés, découle les besoins non-fonctionnels qui suivent.

- **Minimalité** de l'installation des bibliothèques.
  L'installation n'inclue aucune bibliothèque superflue.
- **Exhaustivité** de l'installation des bibliothèques.
  L'installation inclue l'entièreté des bibliothèques nécessaire.
- **Automaticité** de l'installation des bibliothèques.
  L'installation requiert une unique manipulation du client.
- **Rapidité** du formatage des données d'entrée du réseau.
  Le formatage des données UI-PRMD s'effectue en moins de 10 secondes.
- **Indépendance** du formatage des données d'entrée du réseau vis-à-vis du reste du programme.
  Le formatage des données UI-PRMD s'effectue indépendamment de l'entraînement du réseau de neurones.
- **Réutilisabilité** des données formatées.
  Les données formatées sont conservées et n'ont plus à être générées de nouveaux.
- **Rapidité** de l'exécution du code avec les nouvelles données d'entrée.
  L'entraînement du modèle se fait en moins de 30 secondes par époque avec des paramètres pertinents et sur du matériel disponible au CREMI.

— **Pertinence** de la production d'un résultat en sortie du modèle.
Les résultats produits, en tenant compte des limitations matérielles, correspondent à l'état de l'art.

— **Fiabilité** de la production d'un résultat en sortie du modèle.
Les résultats produits ne comportent pas de biais liés à une dispersion singulière des classes lors de l'entraînement.

— **Rapidité** de la visualisation du résultat.
La visualisation des résultats se fait dans les 10 secondes succédant au calcul des résultats.

— **Fiabilité** de la visualisation du résultat.
Les résultats visualisés correspondent aux résultats calculés.

Chapitre 2

Conception

Le chapitre d'analyse a permis la mise en avant des notions clés du projet, par la façon dont elles s'inscrivent dans celui-ci, et par l'explication des domaines qui leur sont connexes. En connaissance de ces informations, il nous est maintenant possible de concevoir la succession des étapes nécessaires à la réalisation du projet.

On peut définir quatre étapes majeures autour desquelles la conception va s'articuler. Dans un premier temps, il est primordial de mettre en place un environnement de développement adapté au démarrage du projet, nous devons être en mesure d'exécuter le code fourni par les clients. Les trois autres étapes de la conception concernent l'entrée, le traitement et la sortie du réseau de neurones. Il nous faut une entrée pour notre réseau, et donc déterminer le format de cette entrée, cela passe par la compréhension des bases de données utilisées dans le code et de la base de données qui nous est fournie par les clients. Il nous faut ensuite pouvoir traiter les données de l'entrée, cela passe par la compréhension du code fourni, de son architecture, de l'architecture du réseau de neurones fourni et de l'architecture du graphe utilisé afin d'effectuer les modifications nécessaires à l'utilisation de nos données d'entrée. Enfin, il nous faut choisir la sortie du réseau, cela passe par l'étude des métriques d'évaluation et du type de classification utilisé dans le code fourni et dans l'état de l'art.

2.1 Environnement de développement

Le projet s'articule autour du code fourni par les clients, ce code est disponible au téléchargement sur la plateforme github et à l'utilisation pour notre projet. Pour être dans un environnement propice au développement du projet, il nous faut pouvoir exécuter ce code. En ce sens, il faut premièrement mettre en place les prérequis que

sont : installer les bibliothèques et effectuer les commandes de traitement des données d'entrée puis d'exécution du code, pour ensuite effectuer les modifications nécessaires à l'exécution du code. Même si ce code est fourni par les clients, ils n'en sont pas les auteurs et les conditions d'installations, les explications d'exécution ou encore les dossiers contenant le code peuvent être incomplets.

2.1.1 Prérequis

Trois actions composent les prérequis, installer les bibliothèques, effectuer les commandes de traitement des données d'entrée, effectuer les commandes d'exécution du code.

L'installation des bibliothèques se fait dans un premier temps sur nos machines personnelles d'après la liste disponible avec le code fourni. Ensuite, les installations seront faites au CREMI. L'exécution future du code nous permettra de déterminer si des installations sont manquantes ou superflues. Ce travail, par la suite, nous amènera à développer un script, paragraphe 3.1.1, qui viendra installer ces bibliothèques de façon automatique, exhaustive et minimale. Le script pourra au besoin être exécuté seul par l'utilisateur ou par un script antérieur d'exécution globale du projet.

Les commandes de traitement des données d'entrée sont listées et disponible avec le code fourni. Nous allons effectuer ces commandes pour déterminer les modifications éventuelles à fournir au code ou constater le bon déroulement du traitement des données d'entrée.

Les commandes d'exécution du code sont listées et disponible avec le code fourni. Nous allons effectuer ces commandes pour déterminer les modifications éventuelles à fournir au code ou constater le bon déroulement de l'exécution du code.

2.1.2 Modifications

Par la mise en place des prérequis, des modifications nécessaires à l'exécution du code peuvent être mises en avant. L'application des correctifs prendra au besoin la forme de modifications du code, modifications des chemins d'accès aux données, modifications de l'architecture des éléments constitutifs du code, modifications des configurations du réseau de neurones ou bien modification des données d'entrées.
S'il est nécessaire d'effectuer une ou plusieurs des modifications précédemment citées le détail des opérations sera notifié dans le paragraphe 3.1.2.

2.2 Entrée du réseau

Suite à la mise en place d'un environnement propice au développement du projet, nous pouvons nous intéresser aux trois autres éléments nécessaires à sa conception. Le premier élément à considérer est l'entrée du réseau, nous allons, grâce à l'analyse précédemment faite des bases de données, déterminer comment mettre en place les procédures permettant l'utilisation des données UI-PRMD pour notre réseau.

2.2.1 Utilisation des bases de données

Avec l'objectif d'utiliser les informations de la base de données UI-PRMD comme entrée du réseau, et considérant l'architecture du projet précédemment acquis, Figure 22 Page 51 , deux manières de procéder se distinguent.

La première consiste à traiter les données sous leur format actuel. Les formats d'encodages des bases de données NTU RGB D et Kinetics étant très différents de celui de UI-PRMD cela induit une forte modification des fichiers du code fourni qui utilisent ces informations par la suite. Pour cette stratégie, il n'y a donc rien à mettre en place au niveau de l'entrée du réseau de neurones, le travail se déporte entièrement au niveau du traitement de l'entrée.

L'autre stratégie consiste à tirer profit des fichiers acquis par le code fourni en effectuant un formatage des données UI-PRMD se rapprochant le plus fidèlement possible du format d'une des deux bases de données utilisée par le code fourni. Dans ce cas, seul une faible modification des fichiers qui utilisent ces informations sera nécessaire par la suite. Pour cette stratégie, étant donné qu'il est possible de formater les données UI-PRMD pour se rapprocher du format de l'une ou l'autre des bases de données utilisées dans le code fourni, de plus amples explications sont nécessaires.

UI-PRMD vers NTU

Nous avons pu disposer des données squelettiques de la base UI-PRMD, et également, nous avons pu disposer des données squelettiques de la base NTU-RGB D et les analyser respectivement dans les paragraphes 1.3.3 page 17 et 1.3.2 page 15. L'énonciation des caractéristiques des formats de stockage de chacune de ces deux bases de données nous a permis de mettre en évidence leurs différences, leurs similitudes et nos capacités à agir sur celles-ci pour obtenir le meilleur formatage possible

des données UI-PRMD vers le format NTU-RGB D. Bien que présentant des données 3D comme UI-PRMD, nous avons déterminé que le format de présentation des données ainsi que l'exécution du programme avec ces données étaient moins avantageux pour nos objectifs et nous avons donc décidé de ne pas opter pour cette méthode.

UI-PRMD vers Kinetics

En plus des deux autres bases de données précédemment citées, nous avons pu disposer des données squelettiques de la base Kinetics et les analyser dans le paragraphe 1.3.1 page 13. L'énonciation des caractéristiques des formats de stockage des bases de données UI-PRMD et Kinetics nous a permis de mettre en évidence leurs différences, leurs similitudes et nos capacités à agir sur celles-ci pour obtenir le meilleur formatage possible des données UI-PRMD vers le format Kinetics. Bien que présentant des données 2D et un score associé à ces données qui résulte d'un système d'acquisition différent, nous avons déterminé que le format de présentation de ces données était beaucoup plus simple à analyser et présentait moins d'informations superflues que le format NTU-RGB D. En ce sens, il semble avantageux pour nos objectifs.

Après ces comparaisons, notre choix se porte sur le formatage des données UI-PRMD au format Kinetics au détriment du format NTU-RGB D ainsi qu'au détriment de l'absence de formatage. La mise en place de notre stratégie sera développée dans le paragraphe 3.2.2.

2.3 Traitement de l'entrée

Après avoir généré les données UI-PRMD dans un format proche des données Kinetics, il faut traiter ces données dans la suite du programme. Les données vont être utilisées par le réseau de neurones en étant présentées sous forme de graphe, sachant l'importance du coût mémoire relatif aux calculs liés à ce type de structure, l'architecture du réseau de neurones sera un facteur important à prendre en compte. Les modifications dues aux changements de données peuvent se répercuter sur l'architecture globale des éléments constitutifs du projet et cela doit également être pris en compte. Ce que nous prévoyons de faire pour traiter chacun de ces aspects est décrit dans les paragraphes suivants.

2.3.1 Architecture du graphe

L'architecture du graphe du code fourni et fonctionnant avec la base de données Kinetics est représentée sur la Figure 21 page 50, nous prévoyons de nous baser sur ce graphe et l'encodage qui est fourni dans le code, pour grâce aux informations relatives aux données squelettiques de UI-PRMD, construire notre propre graphe, comme présenté Figure 9 page 25. Le graph est construit à partir des données acquises par Kinect et non Vicon pour limiter le nombre de noeuds.

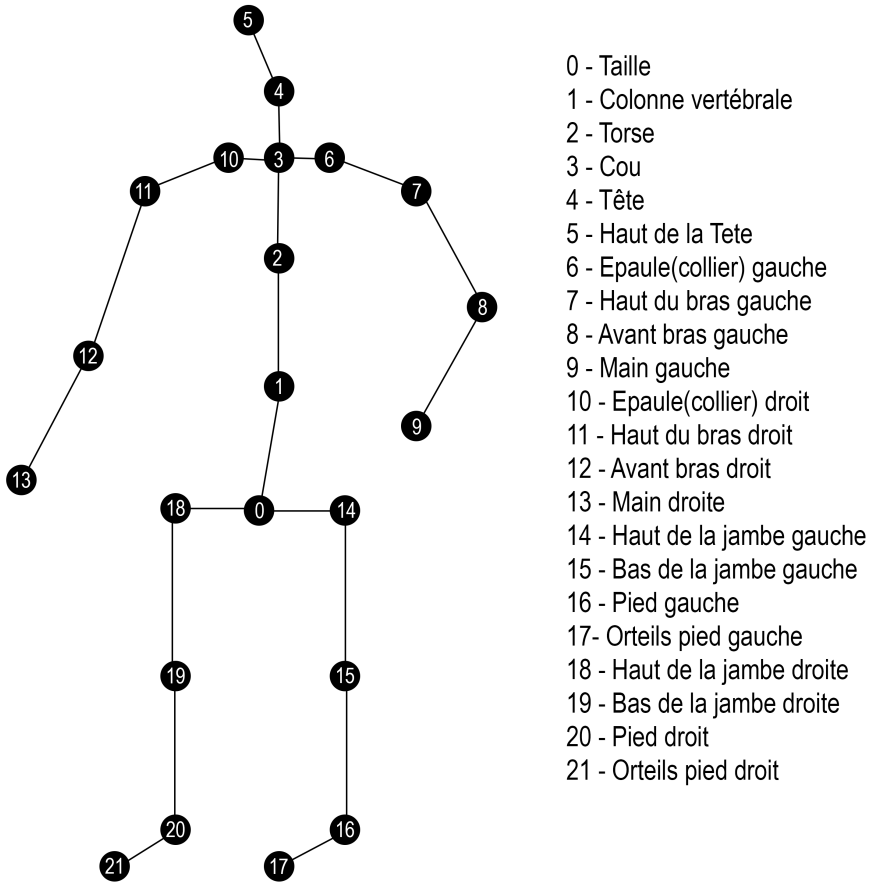

FIGURE 9 – Architecture du graphe pour UI-PRMD

2.3.2 Architecture du projet

L'architecture du projet qui nous a été fourni, 22 page 51 permet l'utilisation des données Kinetics et NTU-RGB D. Nous prévoyons de nous baser sur cette architecture pour construire notre propre architecture fonctionnant avec les données UI-PRMD, comme présentée Figure 10 page 26.

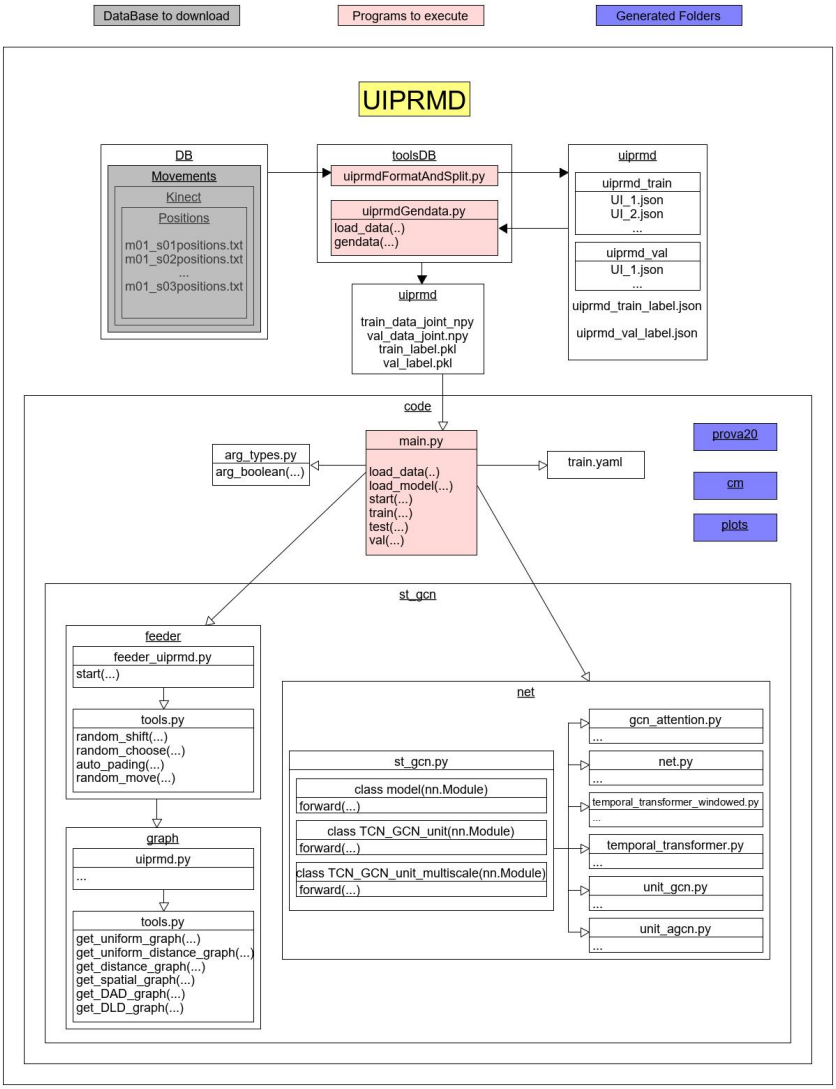

FIGURE 10 – Architecture du projet pour UI-PRMD

2.3.3 Architecture du réseau de neurones

Nous ne prévoyons pas actuellement d'effectuer des modifications sur l'architecture du réseau de neurones. Les tests du chapitre Réalisation, paragraphe 3.4.3, pourront permettre de définir les besoins spécifiques d'une nouvelle architecture en fonction des exécutions du code à venir. Nous conservons donc le réseau présenté Figure 23 page 52, ou si un besoin de modifications apparaît, nous nous baserons fortement sur celui-ci.

2.4 Sortie du réseau

Suite au traitement des données d'entrée par notre réseau de neurones, une sortie qui témoigne de son apprentissage sur ces données est attendue. Le projet a pour objectif de reconnaître la qualité d'exécution d'un exercice et en particulier la mauvaise exécution d'un exercice pour éviter des douleurs au patient qui le réalise, en ce sens, on souhaite classifier comme correcte le moins d'exécution incorrecte possible. On préférera classifier comme incorrecte une exécution correcte et faire intervenir un personnel qualifié inutilement. Le modèle, dans l'objectif de déterminer l'exécution correcte d'un mouvement, devra minimiser le nombre faux positif, l'exécution est dite correcte, mais ne l'est pas, au détriment d'une potentielle augmentation des faux négatif, l'exécution est dite incorrecte, mais ne l'est pas.
Pour satisfaire ce besoin, un choix judicieux des métriques d'évaluation des performances du modèle est primordial.

2.4.1 Métriques d'évaluation

Le choix des métriques d'évaluation résulte de la prise en compte des contraintes estimées précédemment. Lors de tests futurs, nos choix pourront être revus si l'on obtient des résultats peu pertinents.
Par la suite, les notations TN, TP, FN, FP correspondent respectivement aux termes vrai négatif, vrai positif, faux négatif et faux positif.

Accuracy

La métrique *Accuracy* est la métrique de base présente dans le code qui est un bon repère d'évaluation, non seulement pour se comparer à l'état de l'art, notamment aux résultats des auteurs du code que l'on exploite, mais également pour évaluer le modèle. Notre échantillon de données d'entraînement étant parfaitement équilibré,

il n'engage pas des biais d'évaluation résultants d'un déséquilibre de classe. Cette métrique permet de connaître la proportion de bonnes prédictions par rapport à l'ensemble des prédictions. La métrique *Accuracy* se calcule via la formule suivante :

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$$

Precision

La métrique *Precision* est une métrique qui permet de mettre en avant le taux de faux positifs, dans notre cas, l'exécution est dite correcte, mais ne l'est pas. Si l'on veut s'assurer de classifier comme incorrecte chaque exécution incorrecte, il faudra s'intéresser à cette métrique. La métrique *Precision* se calcule via la formule suivante :

$$Precision = \frac{TP}{TP + FP}$$

Sensitivity & Specificity

Les métriques *Sensitivity* et *Specificity* mesurent respectivement la proportion d'éléments positifs correctement identifiés et la proportion d'éléments négatifs correctement identifiés. Ces deux métriques sont complémentaires et permettent comme pour la métrique *Precision* de se concentrer sur un aspect spécifique de notre modèle. La métrique *Sensitivity* se calcule via la formule suivante :

$$Sensitivity = \frac{TP}{TP + FN}$$

La métrique *Specificity* se calcule via la formule suivante :

$$Specificity = \frac{TN}{TN + FP}$$

F1-Score

La métrique *F1-Score* est une métrique qui combine *Sensitivity* et *Precision*. Elle évalue le modèle dans sa globalité comme la métrique *Accuracy* mais ne prend pas en compte les *vrai négatif* dans l'évaluation, ainsi sur un ensemble de données fortement déséquilibré notre perception de la performance de l'algorithme n'est pas biaisée. Dans notre cas, pour des données d'entraînement contenant majoritairement des exécutions d'exercices incorrectes, la métrique *F1-Score* est adaptée. On peut

concevoir ce cas de figure comme intervenant suite à une extension de la base de données actuelle.
La métrique *F1-Score* se calcule via la formule suivante :

$$F1 - Score = \frac{Precision * Sensitivity}{Precision + Sensitivity}$$

2.4.2 Visualisation

Après avoir déterminé les métriques d'évaluation du modèle, il faut pouvoir visualiser les résultats. Plusieurs options sont à notre disposition et nous prévoyons d'utiliser les suivantes.

La matrice de confusion

La matrice de confusion est demandée par les clients et représente une information visuelle qui pourra montrer rapidement si notre système de classification parvient à classifier correctement.

Les courbes

L'affichage de données sous forme de courbes est un incontournable de l'état de l'art en matière de visualisation des résultats. Aussi, pour corroborer nos résultats sous forme textuelle, en plus de la visualisation d'une matrice de confusion, nous prévoyons de générer les courbes représentatives de la fonction de perte pour constater un sur-apprentissage du modèle ainsi que les courbes représentatives des métriques d'évaluations calculées pour constater que le modèle apprend.

Chapitre 3

Réalisation

Le chapitre d'analyse a permis la mise en avant des notions clé du projet, par la façon dont elles s'inscrivent dans celui-ci, et par l'explication des domaines qui leur sont connexes. Le chapitre de conception, en connaissance de ces informations, a permis d'anticiper et de définir la succession des étapes nécessaires à la réalisation du projet. En conséquence, ce dernier chapitre intitulé réalisation va venir préciser par l'implémentation chacun des points abordés dans le chapitre précédent et apporter des ajouts qui répondent à des problématiques non anticipées lorsque cela est nécessaire.

Étant complémentaire au chapitre précédent, notre chapitre réalisation s'articule autour de quatre axes, les mêmes axes que ceux du chapitre conception. Dans un premier temps, il était primordial de mettre en place un environnement de développement adapté au démarrage du projet, c'est pourquoi nous avons mis en place un script qui réalise l'installation des bibliothèques. Également, nous avons apporté des modifications au code fourni pour faire fonctionner les commandes de traitement des données d'entrée et d'exécution du code. Les 3 autres étapes de la réalisation concernent l'entrée, le traitement et la sortie du réseau de neurones. Nous avons mis en place un script qui formate les données UI-PRMD pour permettre leur utilisation en tant que données d'entrée de notre réseau. Pour traiter ces nouvelles données d'entrée, nous avons ensuite apporté des modifications à l'architecture globale du projet, et notamment aux parties dédiées à l'architecture du graphe, à l'architecture du feeder et à l'architecture du réseau de neurones. Enfin, nous avons calculé et permis la visualisation des métriques qui témoignent de l'apprentissage de notre réseau en fonction des requêtes des clients et par empirisme suite à nos différents tests.

3.1 Environnement de développement

3.1.1 Script d'installation des prérequis

En amont de l'automatisation des installations, il nous a fallu déterminer la liste des installations nécessaires. Comme indiqué dans les besoins non-fonctionnels paragraphe 1.4.2 page 19, il faut installer l'ensemble des bibliothèques nécessaires sans installer de bibliothèques superflues. Pour arriver à ce résultat, nous avons simplement procédé de façon empirique en lançant le code dans un environnement virtuel qui autorise les paquets à être installés pour une application particulière, plutôt que d'être installés au niveau du système. Nous avons ainsi installé les bibliothèques une par une, et par l'aboutissement de cette tâche d'installation qui est l'exécution du projet, nous avons obtenu l'ensemble des bibliothèques nécessaires. Pour déterminer l'ensemble des bibliothèques superflues, nous avons effectué l'opération inverse, nous avons désinstallé les bibliothèques une à une pour déterminer celles qui, par leur absence, n'empêchaient pas la bonne exécution du programme.

Via l'utilisation du module *subprocess* et de sa fonction *run()*, nous avons écrit le script, Figure 11 page 31, qui exécute l'ensemble des commandes qui a été précédemment défini comme exhaustif et minimal, rendant ainsi l'installation automatique.

```
tmp = 1
while(tmp):
    yn = input("Proceed to install the libraries ? (y/n) : ")

    if(yn == 'N' or yn == 'n'):
        print('Quit installing')
        sys.exit()

    if(yn == 'Y' or yn == 'y'):
        tmp = 0

    if tmp != 0:
        print("write either y or n")

#Proceeds installations
print("\n Installing tdqm :\n")
list_files = subprocess.run(["pip3", "install", "tdqm"])

print("\n Installing sacred :\n")
list_files = subprocess.run(["pip3", "install", "sacred"])
```

FIGURE 11 – Extrait du code du script d'installation

Nos tests et notre implémentation permettent ainsi de répondre aux exigences fixées par les besoins non-fonctionnels pour ce qui est de l'installation des bibliothèques, à savoir, minimalité, exhaustivité et automaticité de l'installation des bibliothèques.

3.1.2 Modifications pour la première exécution

Suite à la mise en place des prérequis, nous avons été en mesure d'initier le processus d'exécution du code. De manière empirique, nous avons constaté que des modifications étaient nécessaires pour effectuer cette tâche. Ces correctifs s'appliquent en tant que modifications du code, modifications des chemins d'accès aux données, modifications de l'architecture des éléments constitutifs du code, modifications des configurations du réseau de neurones ou bien modification des données d'entrées.

Après avoir téléchargé l'archive disponible sur github, en voulant exécuter le code, la première erreur qui survient est une erreur de chemin d'accès aux données, le fichier de configuration du réseau de neurones existe, mais n'est pas à l'endroit indiqué dans le code et apparaît donc comme inexistant. Le déplacement du fichier à l'endroit lu par le main ou le changement du chemin de lecture associé règle ce problème.

Ensuite, le fichier de configuration indique l'accès aux données d'entrées du réseau qui ne sont pas présentes. Il faut donc dans un premier temps récupérer les bases de données, ce qui sur nos machines personnelles peut déjà être problématique considérant le poids de ces données, puis ensuite utiliser le script de génération de ces données en données adaptées à l'entrée du réseau de neurones. Ce code présente une incompatibilité des chemins entre l'emplacement de génération des données et l'accès par le main à ces données générées. Encore une fois, le déplacement des dossiers à l'endroit lu par le main ou le changement des chemins de lecture associés règle ce problème.

Le fichier de configuration des paramètres du code associé à l'utilisation de la base de données Kinetics contient des informations sur des données relatives à la base de donnée NTU RGB D. Ainsi, la modification de ce code pour passer en argument du feeder lié à Kinetics les bonnes données d'entrée précédemment générées regle ce problème.

À partir de là, le code s'exécute, mais une erreur intervient, elle est liée à la précédente, le code attendait des données qui n'existent pas et donc il se crée un

décalage entre le nombre de canaux instanciés par le modèle et les données que l'on lui donne. La modification du fichier de configuration règle ce problème.

Le code s'exécute de nouveau, mais une erreur liée au matériel apparaît. Un manque de mémoire GPU, sur nos machines personnelles comme sur celles du CREMI, cause l'arrêt du programme. Cette limitation est gênante pour le bon déroulé du projet et la comparaison future avec l'article relatif, mais elle se résout par soit, une diminution du nombre de données propagées dans le réseau lors d'une même propagation, soit, une modification de l'architecture du réseau de neurones pour faire diminuer le nombre de paramètres entraînables. Dans un premier temps, nous avons opté pour une diminution de la taille des lots de données propagées puis par la suite nous avons modifié le réseau, comme indiqué dans le paragraphe 3.4.3 page 43.

Le code s'exécute de nouveau, mais une erreur que nous avons su identifier comme étant liée à un problème de chargement des poids survient. Après avoir effectué des modifications, notamment de renommage des poids, le problème se règle.

Le code s'exécute de nouveau, mais ne donne pas de résultats en fin d'exécution, cette fois sans erreur apparente. Un appel à une fonction était manquant et ne permettait pas l'affichage des métriques d'évaluation. Le problème a été réglé en déterminant l'appel manquant et en le rajoutant.

Le dernier problème était l'absence de création d'un dossier dans lequel devaient se générer les sauvegardes des étapes d'exécution du modèle, ainsi en le créant avant exécution du code, nous avons pu régler ce problème et par conséquent l'ensemble des problèmes qui empêchaient l'exécution complète du code.

3.2 Entrée du réseau

Suite à l'installation des prérequis et à la correction des problèmes qui empêchaient l'exécution du code fourni par les clients, nous sommes en mesure de travailler sur les étapes de la réalisation qui concernent l'entrée, le traitement et la sortie du réseau de neurones. Le premier élément à considérer est l'entrée du réseau, nous avons implémenté les conceptions faites dans la partie 2.2 page 23, à savoir, le formatage des données UI-PRMD au format Kinetics.

3.2.1 Script de formatage des données UI-PRMD

Après comparaisons des différentes possibilités de réalisation dans la partie conception précédemment citée, notre choix s'est porté sur le formatage des données UI-PRMD au format Kinetics au détriment du format NTU-RGB D ainsi qu'au détriment de l'absence de formatage. La stratégie consistant à tirer profit des fichiers acquis par le code fourni pour n'avoir à effectuer qu'une faible modification des fichiers qui utilisent ces informations s'est avérée payante puisque nous avons été en mesure de réaliser ce formatage et par la suite de faire fonctionner le modèle avec ces données d'entrée comme expliqué dans le paragraphe 3.3 page 38.

Comme indiqué dans les besoins non-fonctionnels paragraphe 1.4.2 page 19, il faut que le script de formatage réponde aux besoins que sont l'*Indépendance*, la *Réutilisabilité* et la *Rapidité*.

Le besoin d'*Indépendance* correspond à la capacité de dissocier l'entraînement du réseau de neurones et le formatage des données. Pour répondre à cela, notre script effectuant le formatage ne fait pas partie d'un bloc d'exécution commun à l'entraînement du réseau, mais s'exécute seul. Le script n'est pas appelé par le point d'entrée du programme comme présenté dans l'architecture du projet Figure 10 page 26.

Le second besoin, la *Réutilisabilité* correspond à la capacité de conserver les données générées pour les exploiter lors de plusieurs entraînements du réseau de neurones. Pour répondre à ce besoin il faut faire un choix qui tient compte d'un autre besoin non spécifié puisque subsidiaire qui est le choix de la répartition des données fournies en données d'entraînement ou données de validation. Il est possible de formater les données une seule fois, toutes ensemble, pour ensuite les répartir en tant que données d'entraînement ou données de validation. Cependant, cette répartition, de par le format des données, nécessite un formatage supplémentaire et donc des créations et déplacements de fichiers qui font défaut au besoin de *Rapidité*, comme expliqué paragraphe 3.2.1 page 35. L'autre possibilité, celle que nous avons choisie, consiste à spécifier la répartition lors de l'exécution du script pour n'effectuer ainsi qu'une seule phase de formatage. Le désavantage réside dans l'obligation d'effectuer un formatage complet pour changer cette répartition, mais nous avons jugé que ce besoin de changement était secondaire et qu'il était plus important d'avantager la rapidité du premier formatage.

Le dernier besoin, celui de *Rapidité*, correspond à la capacité de produire le résultat attendu dans un temps minimal. Ce besoin aura été le plus simple à réaliser, comme nous travaillons sur des créations et déplacements de fichier, il a fallu limiter le nombre de ces manipulations au strict nécessaire pour éviter d'entraîner des consommations de temps superflues.

L'implémentation du code s'est faîte en plusieurs étapes, la première est celle de choix et de vérification de la valeur du taux de répartition des données en données d'entraînement ou de validation. Comme présenté Figure 12 page 35, l'utilisateur peut choisir son taux de répartition ou conserver celui par défaut, il devra confirmer ce choix et ne pourra pas poursuivre l'exécution du script sans sélectionner une valeur cohérente.

```
#region 1 : Ensure split rate value consistency
#-----------------------------------------------
splitRate = 0
if len(sys.argv) == 1:
    splitRate = 90
    print("Defaultly chosen 90% split rate")
elif (type(int(sys.argv[1])) != int) or (int(sys.argv[1]) > 99) :
        print('Select an integer value under 100')
        print('Quit processing')
        sys.exit()
else:
    splitRate = int(sys.argv[1])

inputSentence = str("Proceed to split values with " +  str(splitRate) +  "% trai

tmp = 1
while(tmp):
    yn = input(inputSentence)

    if(yn == 'N' or yn == 'n'):
        print('Quit processing')
        sys.exit()

    if(yn == 'Y' or yn == 'y'):
        tmp = 0

    if tmp != 0:
        print("write either y or n")
#-----------------------------------------------
#endregion
```

FIGURE 12 – Extrait du code du script de formatage

Suite à ça, il faut créer les dossiers de stockage des données qui vont être générées, définir les chemins d'accès et définir les configurations associées aux données comme les labels ou le nombre de points d'ancrage. De là, suite à l'étude des caractéristiques des données faîtes dans la partie 1.3 page 13, nous avons lu les informations des fichiers UI-PRMD pour leur associer les configurations précédemment nommées et les organiser comme celles des fichiers Kinetics. Ensuite, nous avons sauvegardé ces données sous le même format, les données squelettiques de chaque fichier sont stockées dans un dossier regroupant les données lié à l'entraînement ou la validation. Un bloc d'informations relatives aux configurations associées à chaque fichier est ajouté à un fichier dédié, hors des dossiers, lié soit à l'entraînement soit à la validation.

La partie formatage, génération et répartition des données squelettiques est présentée Figure 13 page 36.

```
#region 4 : For each correctPositions format file, generates a new UI file in json/kinetics format
#-----------------------------------------------
splitCount = 0
for curFileIndex in range (len(corArrListFiles)):
    arrData = []
    #-----------------------------------------------
    #Open one file to read in it
    f = open(correctdbPath+corArrListFiles[curFileIndex], "r")

    #For a file, store a string per line
    lines = f.readlines()

    #Split and crop elements of a line and store in array
    frameIndex = 1
    for lineIndex in range (len(lines)):
        arrOneFrame = lines[lineIndex].split()
        arrOneFrame_5digits = []
        for i in range(len(arrOneFrame)):
            arrOneFrame_5digits.append(arrOneFrame[i][:5])

        x = { "frame_index": frameIndex, "skeleton": [{ "pose": arrOneFrame_5digits }]}
        arrData.append(x)
        frameIndex+=1
    #-----------------------------------------------

    #-----------------------------------------------
    #For a file dump the json format in a new file
    final = {"data": arrData, "label": corLabel, "label_index": corLabelIndex}
    z = json.dumps(final)
    new_filename = "UI_"+str(curFileIndex+1)
    new_file = new_filename +".json"
    subprocess.run(["touch", new_file])
    with open(new_file, 'w') as outfile:
        outfile.write(z)
    if(splitCount < splitRate):
        subprocess.run(["mv", new_file, trainFolder])
        splitCount += 1
        #Storage for next step labels file process
        trainArrLabelNames.append(new_filename)
        trainArrLabelData.append({"label": corLabel, "label_index": corLabelIndex})
    else:
        subprocess.run(["mv", new_file, valFolder])
        valArrLabelNames.append(new_filename)
        valArrLabelData.append({"label": corLabel, "label_index": corLabelIndex})

    #-----------------------------------------------
#-----------------------------------------------
#endregion
```

FIGURE 13 – Extrait du code du script de formatage

La partie formatage, génération et répartition des données de configuration est présentée Figure 14 page 37.

```
#region 6 : Generate the train label files
#-----------------------------------------------
#Generate the label file
trainLabelFile = "uiprmd_train_label.json"
#Concat dictionnary inside dictionnary
finalDict = {}
for i in range (len(trainArrLabelNames)):
    finalDict = dict(finalDict, **{trainArrLabelNames[i]: trainArrLabelData[i]})

output = json.dumps(finalDict)
with open(globalFolder+"/"+trainLabelFile, 'w') as outfile:
    outfile.write(output)
#-----------------------------------------------
#endregion
```

FIGURE 14 – Extrait du code du script de formatage

Les figures 13 et 14 représentent une partie du code et non sa globalité, les traitements qui sont effectués pour les données de positions correctes et les données d'entraînement le sont de façon similaire pour les données de positions incorrectes et les données de validation.

3.2.2 Script de génération des données d'entrée

Comme anticipé dans la partie 2.2.1 page 23, la stratégie de formatage des données précédemment développée nous permet de ne modifier que peu le fichier de génération des données d'entrée.

La partie liée à l'utilisation des données en trois dimensions en remplacement des données en deux dimensions et d'un score est présentée Figure 15 page 38.

```
sample_name = self.sample_name[index]
sample_path = os.path.join(self.data_path, sample_name)
with open(sample_path, 'r') as f:
    video_info = json.load(f)

# fill data_numpy
data_numpy = np.zeros((self.C, self.T, self.V, self.num_person_in))
for frame_info in video_info['data']:
    frame_index = frame_info['frame_index']
    for m, skeleton_info in enumerate(frame_info["skeleton"]):
        if m >= self.num_person_in:
            break
        pose = skeleton_info['pose']
        data_numpy[0, frame_index, :, m] = pose[0::3]
        data_numpy[1, frame_index, :, m] = pose[1::3]
        data_numpy[2, frame_index, :, m] = pose[2::3]

#region ⋯

# get & check label index
label = video_info['label_index']
assert (self.label[index] == label)

#region ⋯

return data_numpy, label
```

FIGURE 15 – Extrait du code du script de génération des données d'entrée

3.3 Traitement de l'entrée

Après avoir généré les données UI-PRMD dans un format le plus proche possible des données Kinetics via le script de formatage, et après avoir généré des données d'entrée utilisable par le réseau dans l'étape suivante, nous allons expliquer comment nous avons implémenté les fonctions clé du traitement de ces données dans la suite du programme. Les données vont être utilisées par le réseau de neurones en étant présentées sous forme de graphe, nous allons expliquer notre implémentation en commençant par ce point.

3.3.1 Création du graph

Comme prévu lors de la conception dans le paragraphe 2.3.1 page 25 nous nous sommes basés sur les fichiers présents dans le code fourni pour, grâce aux informations relatives aux données squelettiques de UI-PRMD, construire notre propre graphe qui s'inscrit dans notre architecture 2.3.2 page 26.

Notre encodage est celui présenté Figure 16 page 39, les couples, au nombre de 21, représentent les liaisons existantes entre les différents points d'ancrage des données UI-PRMD, la matrice d'adjacence calculée est un moyen de stockage de ces données.

```
# Edge format: (origin, neighbor)
num_node = 22
self_link = [(i, i) for i in range(num_node)]
inward = [(5, 4), (4, 3), (3, 6), (6, 7), (7, 8), (8, 9), (3, 10), (10, 11),
          (11, 12), (12, 13), (3, 2), (2, 1), (1, 0), (0, 14), (14, 15), (15, 16),
          (16, 17), (0, 18), (18, 19), (19, 20), (20, 21)]
outward = [(j, i) for (i, j) in inward]
neighbor = inward + outward

class Graph():

    def __init__(self, labeling_mode='uniform'):…

    def get_adjacency_matrix(self, labeling_mode=None):…

def main():…

if __name__ == '__main__':…
```

FIGURE 16 – Extrait du code du graphe pour UI-PRMD

3.3.2 Création du feeder

De la même façon que pour la création du graphe, nous nous sommes basés sur les fichiers présents dans le code fourni pour construire notre propre feeder qui s'inscrit dans notre architecture 2.3.2 page 26.

De par le formatage des données précédemment réalisé, la structure du code a pu être conservée en n'effectuant que de légères modifications des chemins d'accès aux données.

3.3.3 Création du réseau de neurones

Les prévisions de la phase de conception concernant l'architecture du réseau de neurones sont discutables. L'utilisation de données en trois dimensions couplée à l'augmentation du nombre de points d'ancrage du graphe entraîne une augmentation de la mémoire GPU nécessaire et aggrave le problème rencontré dans le paragraphe 3.1.2 page 33.

En conséquence, il a été nécessaire de modifier le réseau de neurones pour diminuer le nombre de paramètres entraînables. L'architecture du nouveau réseau de neurones est présentée Figure 17 page 40, la taille des couches a été divisée par huit.

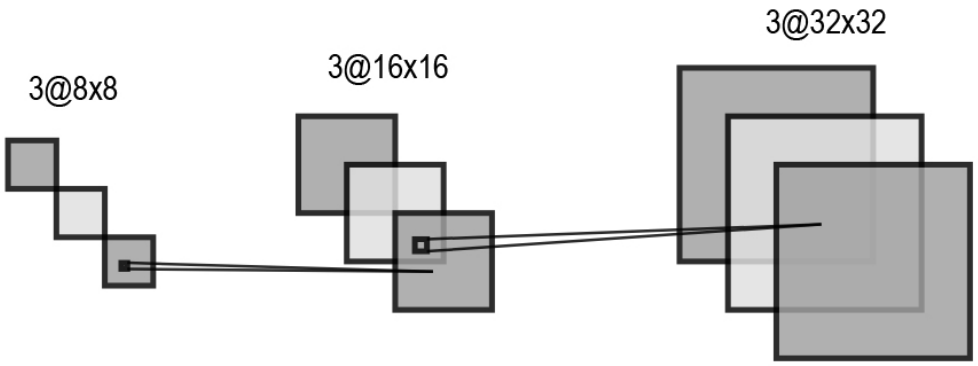

FIGURE 17 – Architecture du réseau de neurones pour UI-PRMD

Ce réseau nous permettra par la suite de répondre dans la partie dédiée aux tests 3.4.3 page 43 au besoin de *Rapidité* stipulé paragraphe 1.4.2 page 19.

3.4 Sortie du réseau

Suite au traitement des données d'entrée par notre réseau de neurones, une sortie qui témoigne de son apprentissage sur ces données est produite. Comme présenté dans la partie dédiée 2.4 page 27 du chapitre conception, nous avons défini des métriques et des outils de visualisation qu'il nous semble pertinent d'utiliser. Notre implémentation est présentée dans les paragraphes qui suivent.

3.4.1 Calcul des métriques d'évaluation

Le modèle, dans l'objectif de déterminer l'exécution correcte d'un mouvement, devra minimiser le nombre de faux positif, l'exécution est dite correcte, mais ne l'est pas, au détriment d'une potentielle augmentation des faux négatif, l'exécution est dite incorrecte, mais ne l'est pas. Pour satisfaire ce besoin, un choix judicieux des métriques d'évaluation des performances du modèle est primordial et nous avons, en concertation avec les clients, fait le choix de calculer les métriques *Accuracy* et *F1-Score*. Notre échantillon de données d'entraînement étant parfaitement équilibré, il n'engage pas des biais d'évaluation résultants d'un déséquilibre de classe. La métrique *Accuracy* permet de connaître la proportion de bonnes prédictions par rapport à l'ensemble des prédictions. La métrique *F1-Score* évalue le modèle dans sa globalité comme la métrique *Accuracy* mais ne prend pas en compte les *vrai négatif* dans l'évaluation, ainsi sur un ensemble de données fortement déséquilibré notre perception de la performance de l'algorithme n'est pas biaisée, ce cas de figure peut intervenir suite à une extension de la base de données actuelle.

Le calcul de ces deux métriques s'effectue à la fin de l'entraînement en utilisant les données stockées par la matrice de confusion que nous sauvegardons pour la visualisation des données 3.4.2 page 41. Toujours dans l'objectif de répondre au besoin de *Rapidité* énoncé dans les besoins non-fonctionnels, l'entraînement du modèle doit se faire en moins de 30 secondes par époque sur du matériel disponible au CREMI. C'est en ce sens que nous effectuons les calculs après l'entraînement, on évite d'une part la redondance des calculs, comme la matrice de confusion doit être générée, il est plus judicieux de calculer nos métriques depuis les données extraites de la matrice, d'autre part, le calcul des métriques peut être effectué à n'importe quel moment depuis les données des matrices de confusion enregistrées et n'a pas nécessairement à se faire à la fin de l'exécution du programme.

3.4.2 Visualisation des résultats

Après avoir effectué les calculs liés aux métriques d'évaluation, il faut pouvoir visualiser ces résultats. Comme prévu dans la partie 2.4.2 page 29 nous proposons la sauvegarde et la visualisation des matrices de confusions ainsi que des courbes relatives aux métriques d'évaluation. La Figure 18 page 42 montre une partie des étapes de l'implémentation de la visualisation de la matrice de confusion et des courbes représentant les métriques d'évaluation.

```
cm = np.load("./checkpoints/cm/cm_"+str((self.arg.num_epoch)-1)+".npy")
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.savefig("./checkpoints/plots/lastEpochCm_epoch"+str((self.arg.num_epoch)-1)+".png")

plt.clf()

plt.plot(epoch_plt,accuracy_plt,'g',label='Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("./checkpoints/plots/Accuracy_epoch"+str((self.arg.num_epoch)-1)+".png")

plt.clf()

plt.plot(epoch_plt,f1score_plt,'g',label='F1-Score')
plt.xlabel('Epochs')
plt.ylabel('F1-Score')
plt.legend()
plt.savefig("./checkpoints/plots/F1-Score_epoch"+str((self.arg.num_epoch)-1)+".png")
```

FIGURE 18 – Extrait des étapes de visualisation des résultats

La Figure 19 page 42 montre les courbes représentant les métriques d'évaluation *Accuracy* et *F1-Score* ainsi que la courbe de perte.

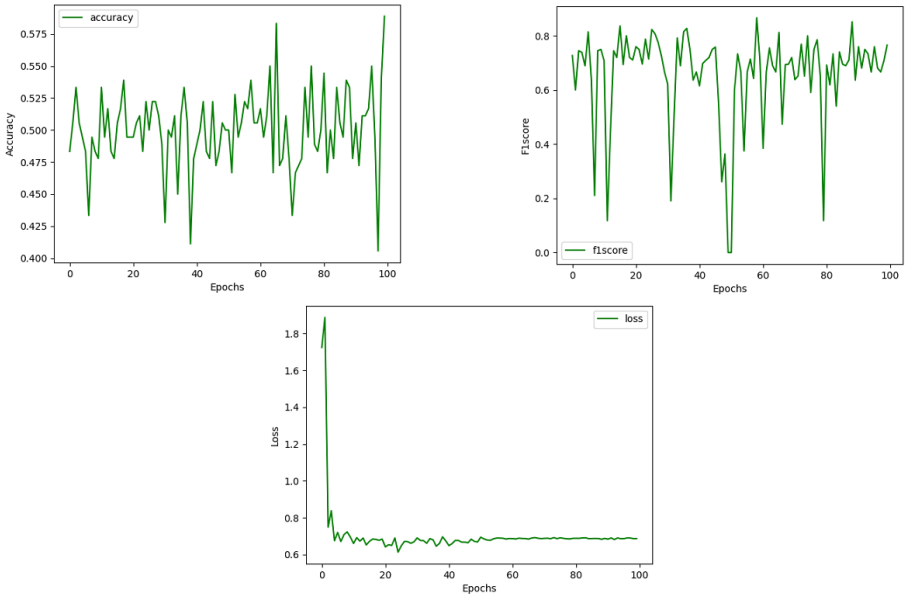

FIGURE 19 – Courbes des métriques d'évaluation

3.4.3 Tests

Lors du chapitre de conception, nous pensions pouvoir conserver le réseau de neurones proposé dans l'article ST-TR et l'entraîner sur nos données d'entrée et notre graphe. Les différents tests que nous avons réalisés, mis en relation avec le matériel informatique dont nous disposons, ont montré qu'il était nécessaire de changer la taille des couches du réseau de neurones pour exécuter notre code.

Nous avons dans un premier temps été en mesure d'entraîner le réseau de neurones de l'article ST-TR avec la base de données Kinetics en diminuant la taille des lots propagés de 128 à 8. Cet entraînement s'effectue sur une durée approximative de six heures par époque sur la base de données complète sur les machines adéquates au CREMI et ne nous permet donc pas d'entraîner le réseau sur un nombre d'époques suffisant pour retrouver les résultats de l'article.

Notre adaptation du code ST-TR amène un ajout du nombre de noeuds du graphe représentant les données d'entrée. Le réseau précédent n'est plus utilisable pour des raisons de manque de mémoire GPU et il nous a fallu diviser la taille des couches par huit pour pouvoir exécuter notre code. En ce sens, nos résultats ne peuvent pas être comparés à ceux de l'article ST-TR.

Les données UI-PRMD sont beaucoup moins volumineuses que les données Kinetics et notre entraînement peut donc se faire en un temps inférieur à 30 secondes par époque. En ayant réalisé jusqu'à 600 époques, et avec des variations sur les paramètres que sont : le pas d'apprentissage, la taille des lots propagés, le nombre d'images utilisées pour décrire une séquence ; il ne nous a pas été possible d'obtenir une courbe de la métrique *Accuracy* qui converge.

Comparaison des Gantt

Dans le cadre du suivi de projet, et comme vu dans le cours qui va de pair avec le PFE, nous avons utilisé une méthode agile, la méthode SCRUM. L'entièreté du travail à réaliser pour finaliser le projet a été répartie sous forme de tâches d'une durée estimée inférieure à une demi-journée. La répartition des tâches s'est faite entre les membres du groupe et à chaque semaine correspond un nombre de tâches à effectuer. Pour permettre une comparaison avec le diagramme de Gantt prévisionnel Figure 1 page 5, le diagramme de Gantt effectif est présenté Figure 20 page 49.

Les tâches relatives aux *Outils de pilotage du projet*, *Livraison d'un produit*, *Rapport* et *Soutenance* ont été réalisés dans les temps prévus.

Les tâches d'*Analyse* et de *Conception* ont été réalisées avec un retard maximal d'une semaine. Nous avons mal mesuré le temps nécessaire à la réalisation d'une présentation orale en anglais de l'état de l'art du projet au groupe de travail des clients.

Les tâches de *Réalisation* ont été réalisées avec un retard maximal de deux semaines. Nous avons mal mesuré le temps qui nous était nécessaire pour rendre le code de l'article ST-TR utilisable. Les tâches de modification du code étaient cependant plus simples que prévu ce qui nous a permis de rattraper ce retard et d'avoir une légère avance sur le démarrage des tests.

Parmi l'ensemble des tâches, une seule n'a pas été réalisée, nous l'avons, peu avant le rendu du projet, et en accord avec les clients, considérée comme non-pertinente. C'est la tâche de *Création d'un script de lancement du projet* qui s'inscrit dans l'objectif de *Livraison d'un produit fini* a été retirée. Les commandes, notamment de téléchargement et positionnement des bases de données ne sont pas automatisées, mais sont correctement indiquées dans le Readme du projet.

Bilan/Perspectives

Pour notre projet de fin d'études, deux objectifs étaient à réaliser.
Le premier objectif consistait à s'approprier le code de l'article ST-TR pour pouvoir par la suite le rendre utilisable, en prenant en compte ses particularités : la structure des données d'entrée du réseau de neurones est un graphe et l'architecture du réseau de neurones est une architecture de type Transformer. Nous avons été en mesure de réaliser ce premier objectif.

Le second objectif consistait à adapter le code à l'utilisation de la nouvelle base de données UI-PRMD et à adapter le fonctionnement du code pour classifier la qualité des actions, en prenant en compte la différence de format des bases de données et la différence des données elles-mêmes. Nous avons été en mesure de réaliser ce second objectif sur le plan théorique. Nous disposons d'un code qui utilise les données UI-PRMD en trois dimensions, qui traite ces données représentant des mouvements pour classifier la qualité des mouvements, mais qui ne produit pas de résultats probants sur le plan de l'apprentissage du réseau de neurones.

Suite aux tests que nous avons réalisés, nous avons pu déterminer des améliorations possibles pour notre projet, pour permettre un entraînement sur un réseau de neurones de taille conséquente avec nos données d'entrée sous forme de graphe. Nous avons mis en place notre code au Labri avant la fin du projet pour utiliser du matériel plus performant et faire des tests sur un réseau de taille supérieure. Nos tests n'ont pas pu aboutir dans le temps imparti, mais notre projet étant continué par Monsieur Kévin Réby, si le matériel n'est pas un frein, des résultats probants pourront être obtenus. Si le coût mémoire et le temps de calcul restent trop élevés, malgré du matériel très performant, une solution peut être la diminution du nombre de noeuds du graphe. Si théoriquement le projet d'analyser la qualité de plusieurs mouvements à partir d'un unique graphe semble réalisable, dans les faits, des adaptations pourraient être envisagées, notamment en se concentrant sur un unique mouvement pour pouvoir diminuer fortement le nombre de noeuds.

Bibliographie

[1] Kleinsmith et al.
"Affective Body Expression Perception and Recognition : A Survey"

[2] Randhavane et al.
"Identifying Emotions from Walking using Affective and Deep Features"

[3] Luo et al.
"ARBEE : Towards Automated Recognition of Bodily Expression of Emotion in the Wild"

[4] Vakanski et al.
"A Data Set of Human Body Movements for Physical Rehabilitation Exercises"

[5] Hochreiter et al.
"Long Short-Term Memory"

[6] Luo et al.
"LSTM Pose Machines"

[7] Bahdanau et al.
"Neural Machine Translation by Jointly Learning to Align and Translate"

[8] Vaswani et al.
"Attention Is All You Need"

[9] Yan et al.
"Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition"

[10] Plizzari et al.
"Skeleton-based Action Recognition via Spatial and Temporal Transformer Networks"

[11] Deepmind
"https ://deepmind.com/research/open-source/kinetics"

[12] & [13] Rapid-Rich Object Search Lab
"https ://rose1.ntu.edu.sg/dataset/actionRecognition/"

[14] Cao et al.
"OpenPose : Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields"

Annexe

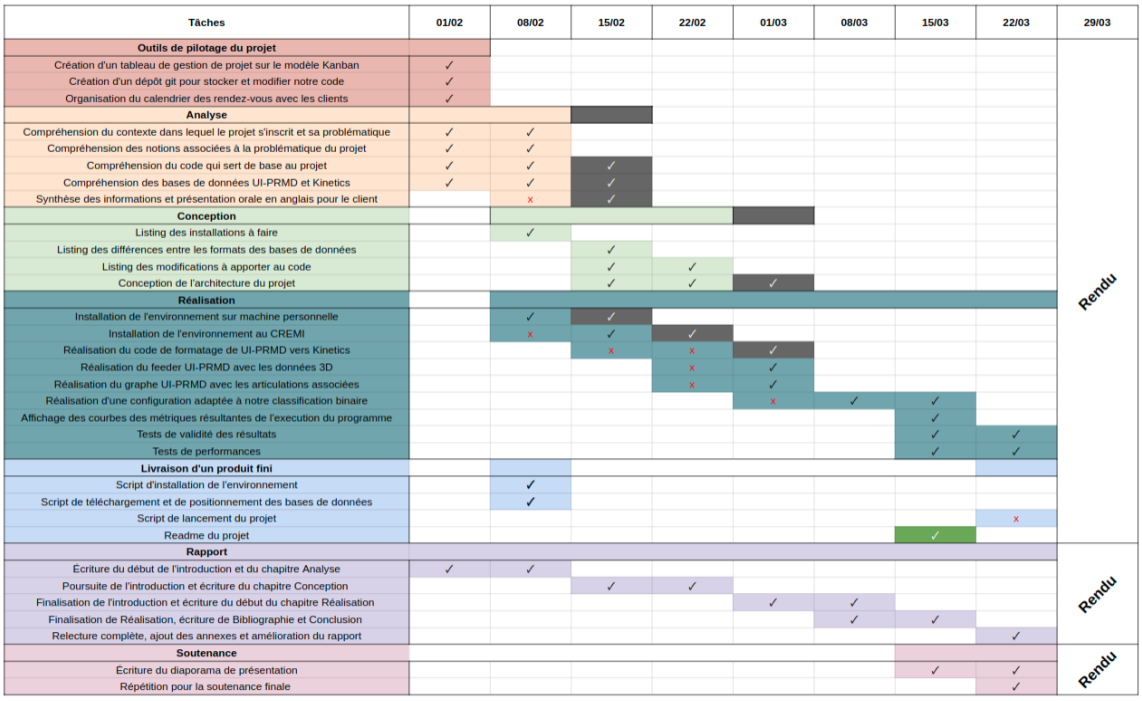

FIGURE 20 – Diagramme de Gantt effectif

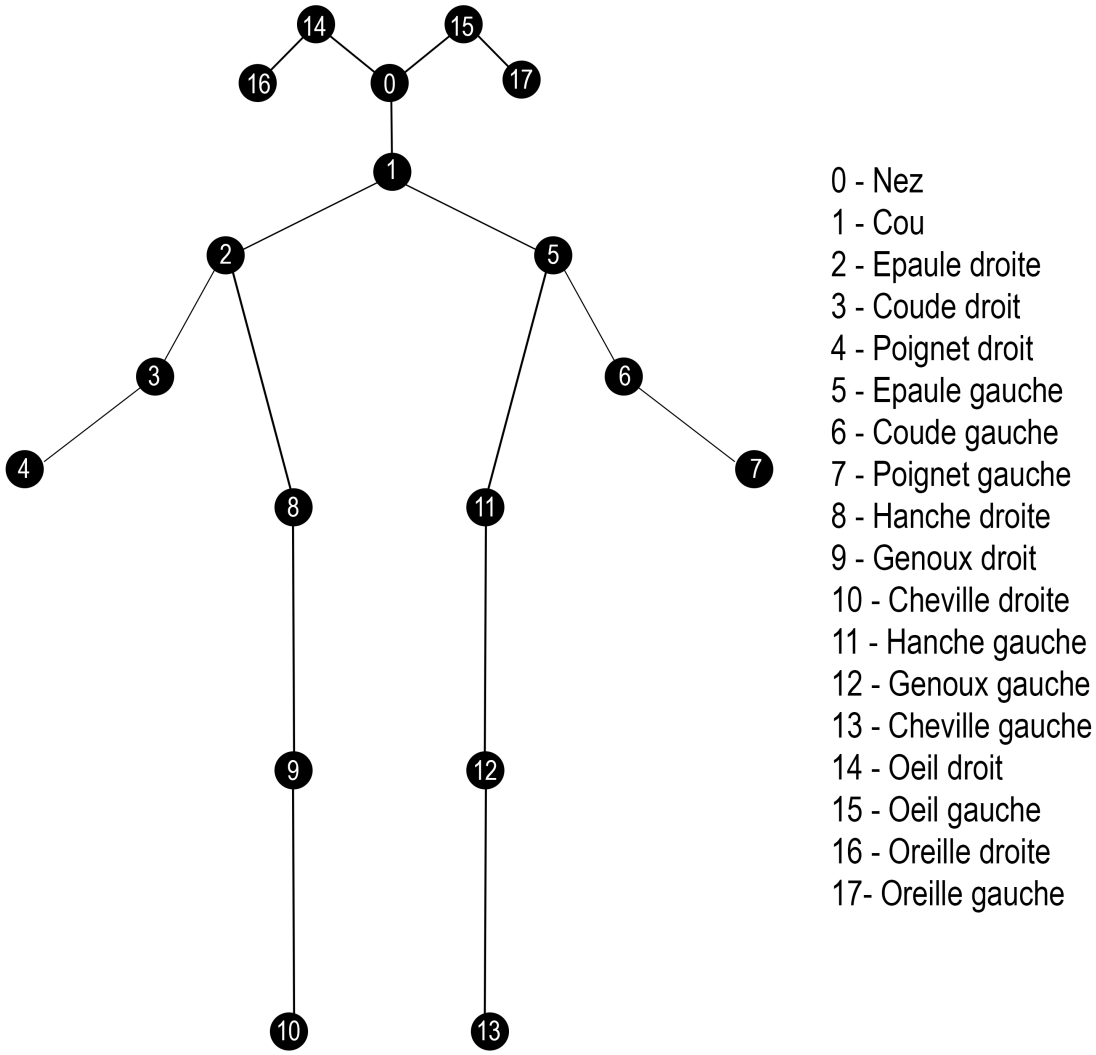

FIGURE 21 – Architecture du graphe pour Kinetics

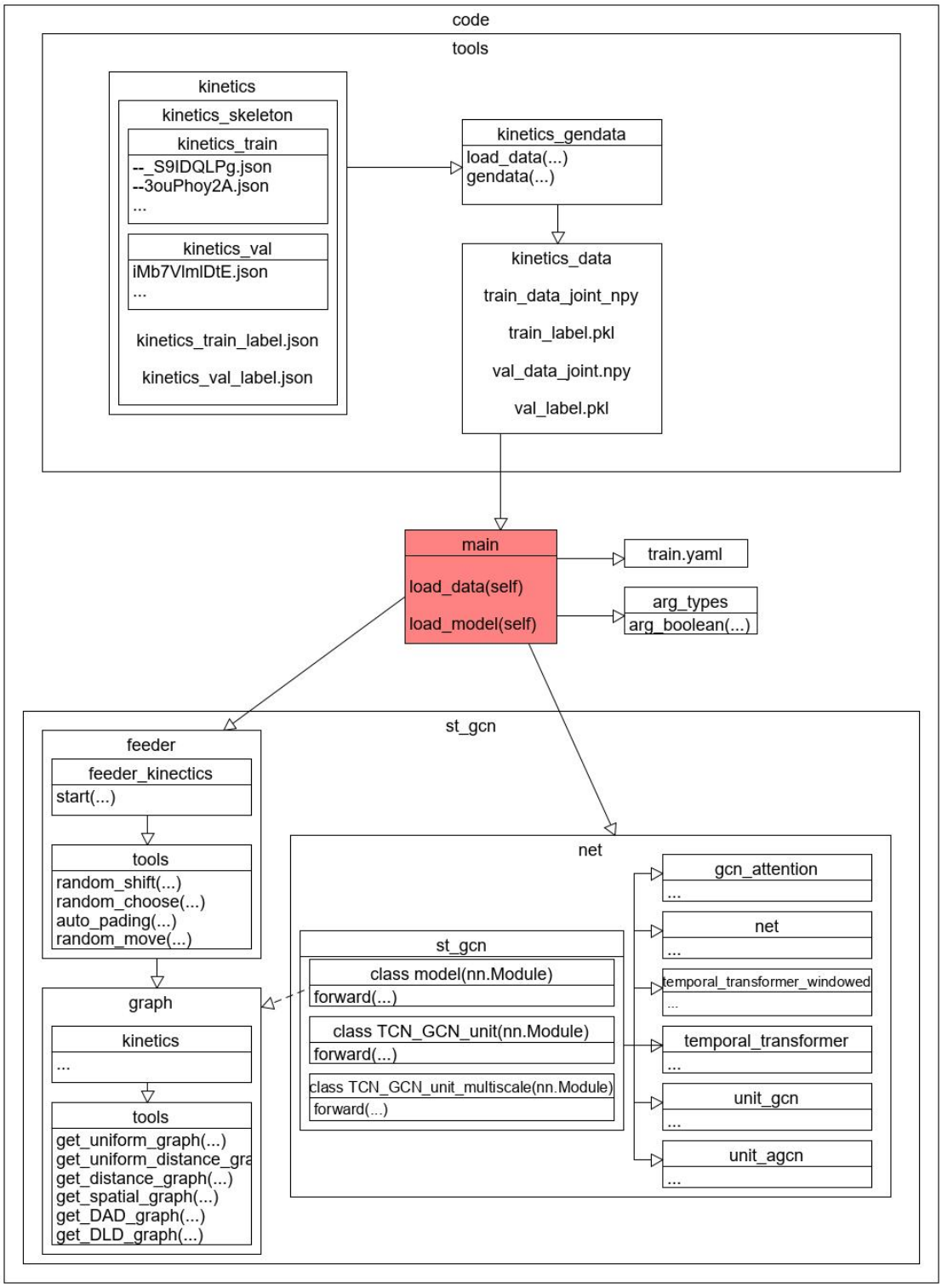

FIGURE 22 – Architecture du projet pour Kinetics

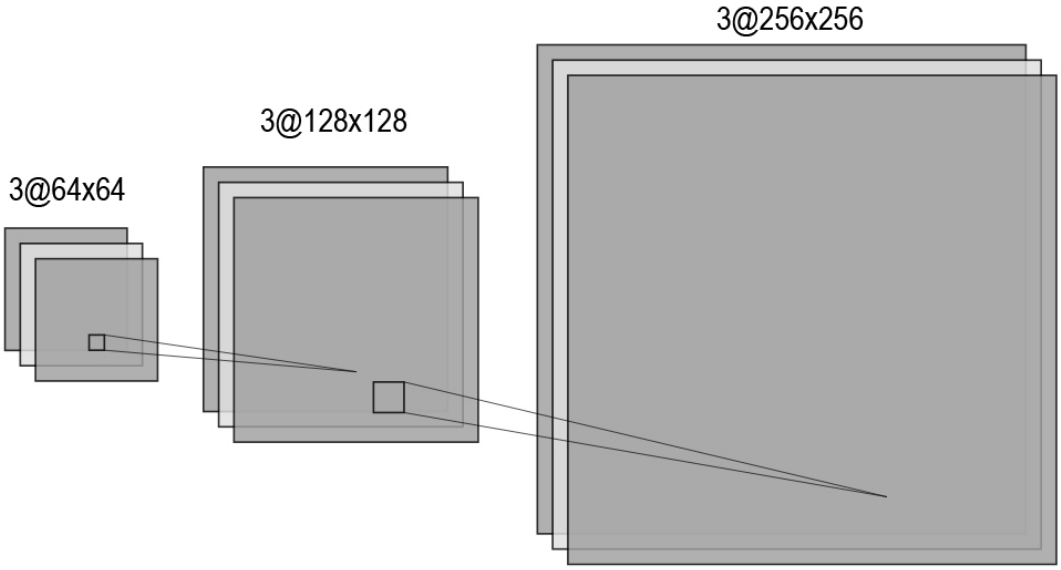

FIGURE 23 – Architecture du réseau de neurones pour Kinetics