Unihockey détection de tir et passe

Quantified Self – Mini-project

Cédric Campos Carvalho, Henri Jaton et Thomas Frantzen

1 CONTEXTE ET OBJECTIFS

Dans le cadre du cours de Quantified Self, les étudiants sont demandés de chercher un projet et d'y amener une approche scientifique pour en découler des résultats concordant aux objectifs du travail. Le but du projet est de pouvoir détecter différentes actions liées à l'Unihockey. L'objectif est de porter différents capteurs au niveau de la canne. Ces capteurs récoltent en données temporelles plusieurs types physiologiques comme l'accélération ou les battements de cœurs. Pour faire suite, l'intérêt est de détecter sur ces données différentes actions grâce à un travail analytique (Data Science).

2 DESCRIPTION DES DONNÉES

Les données temporelles sont récupérées via l'appareil Polar Sensor accroché sur la canne. Il est accroché au-dessus de la palette côté backend comme le montre l'image ci-dessous.

Figure 1 Canne unihockey avec en rouge l'emplacement du capteur.

Cet emplacement permet d'encombrer le moins possible le joueur tout en détectant au maximum les mouvements. Le choix de l'emplacement au niveau du poignet a été mis de côté, car on rencontre moins de mouvements. C'est pour cette raison que le seul choix viable de variables sont les données d'accélération. Ces variables sont présentées sous trois axes ; x(mG), y(mG) et z(mG). L'unité représente des milli-G, G étant plus communément appelé « G-Force ».

Lors de l'enregistrement des données, le joueur effectue plusieurs passes et tirs dans un certain lapse de temps. Les données sont également indexées par timestamp pour les représenter au fil du temps comme le montre l'image ci-dessous.

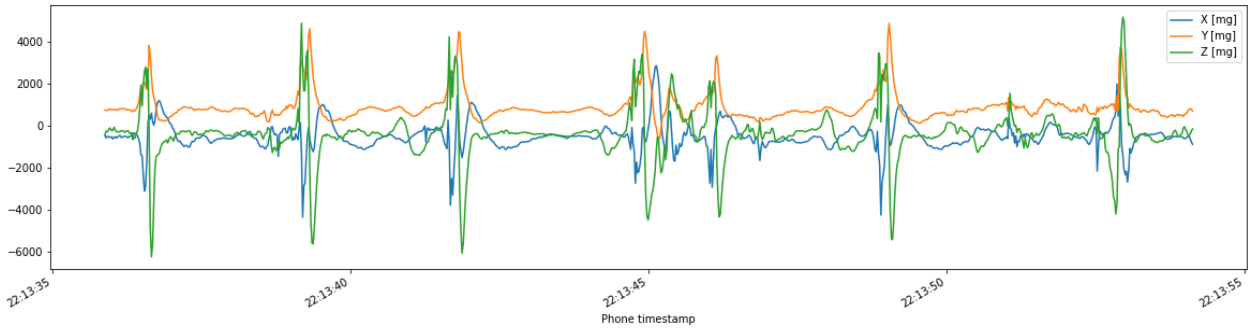

Figure 2 Exemple données temporelles

3 PRE-PROCESSING

La capture des données s'effectue séparément pour différencier les passes des tirs. Pour chaque action, elles possèdent deux sous-catégories ; *backend* et *forward*. La création des marqueurs permet de labéliser correctement les différentes catégories pour faciliter l'apprentissage des modèles.

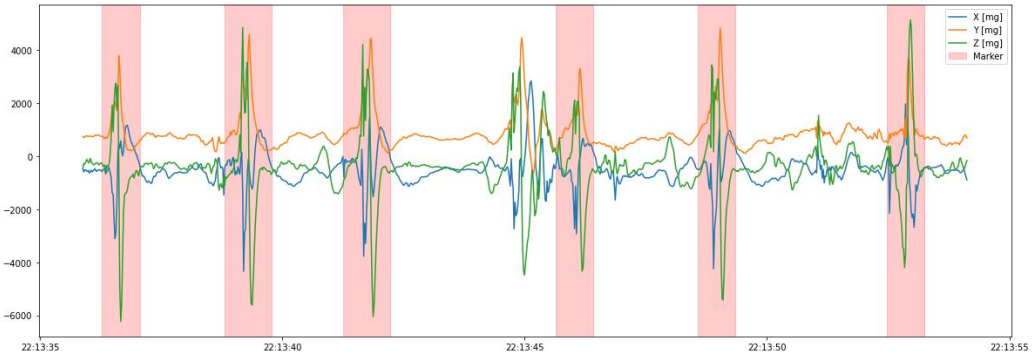

Figure 3 Données temporelles avec les marqueurs délimitant les zones en rouge.

Les données étant limitées, la technique est de concaténer plusieurs données temporelles en une seule pour obtenir un plus grand nombre d'actions à prédire.

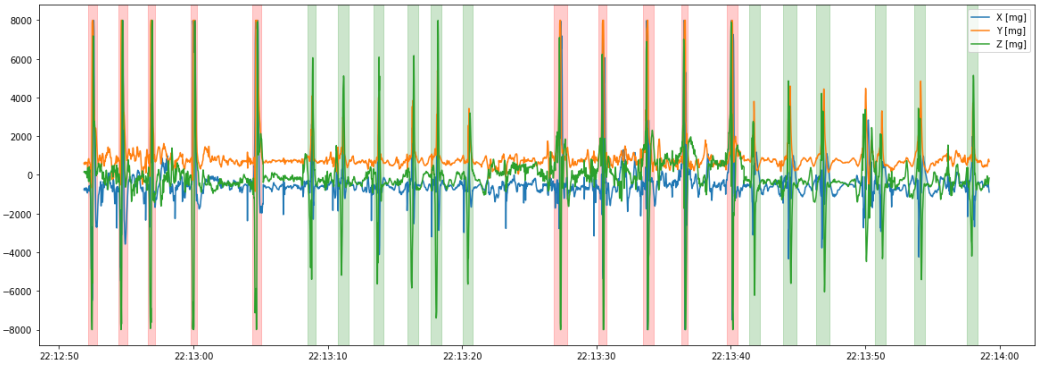

Figure 4 Données temporelles concaténées avec en vert les passes et rouge les tirs.

L'utilisation de la norme des trois accélérations permet de simplifier la complexification des modèles et ainsi obtenir de meilleures performances.

4 TECHNIQUES ET PARAMÈTRES UTILISÉS

Le principe global de ce travail consiste à séparer en blocs de même taille les données temporelles pour les utiliser sous différentes façons via les différents modèles. La séparation se fait en détectant la plus grande durée d'une action. Cette durée est multipliée par la fréquence d'échantillonnage de l'appareil (50Hz) pour obtenir la taille des blocs. Par suite de multiples tests, les résultats sont meilleurs en réduisant la taille max par 55%.

Un problème majeur du projet vient du manque des données car compliquées à enregistrer et délimiter avec des marqueurs principalement pour les actions. Pour cette raison, une fonction permet d'ajouter des passes et tirs avec un bruit gaussien.

La technique de Window Slide permet d'extraire chaque bloc et y extraire des features via la norme si nécessaire. Pour les modèles simples (sans réseau de neurones), l'utilisation de deux trois variables sont choisies en raison des résultats des boxs and whiskers :

- Range : |max-min|
- Max
- 1er quartile

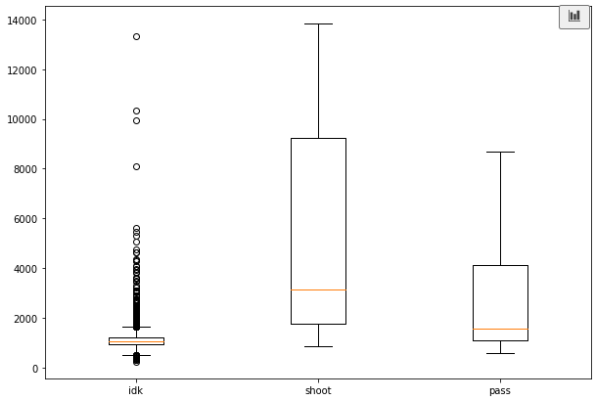

Figure 5 Box and Whisker par type d'actions

Le graphe suivant montre la séparation par actions via les deux premières variables décrites au-dessus. Comme il le montre, il existe une claire séparation avec quelques valeurs aberrantes.

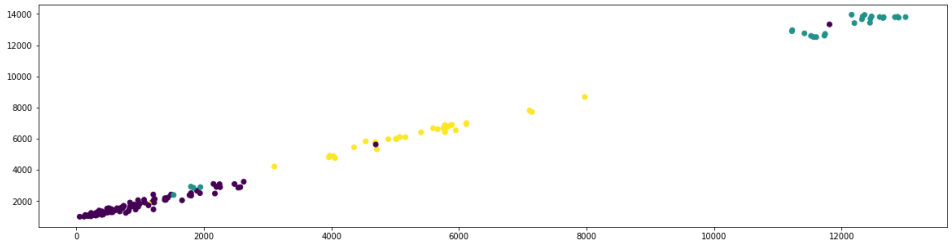

Figure 6 Représentation des variables range et max par action (en couleurs).

Une dernière étape communément utilisée consiste à normaliser les données pour faciliter la descente de gradient dans les réseaux de neurones. La « Z-Norm » permet d'obtenir des valeurs entre [-1,1] et non plus des données pouvant aller jusqu'à 14000.

5 EXPÉRIENCES ET RÉSULTATS

Les expériences effectuées sont séparées sous deux catégories ; modèles simples/statistiques et modèles avec réseau de neurones. L'idée est d'entraîner des modèles avec des données statiques (où le joueur n'avance pas) et marquées (pour délimiter les actions) puis d'ensuite utiliser ce modèle sur des données plus longues et non labélisées.

L'entrainement des modèles simples s'effectue avec deux tiers des données puis utilise l'extraction des trois variables décrites précédemment. Les résultats sont les suivants :

	Logistic Regression	*K Neighbors Classifier*
% précision entraînement	94.29	95.24
% précision test	94.34	94.34

Ces résultats sont obtenus avec deux actions possibles, le label « idk » désigne les blocs où il n'y a aucune des deux actions.

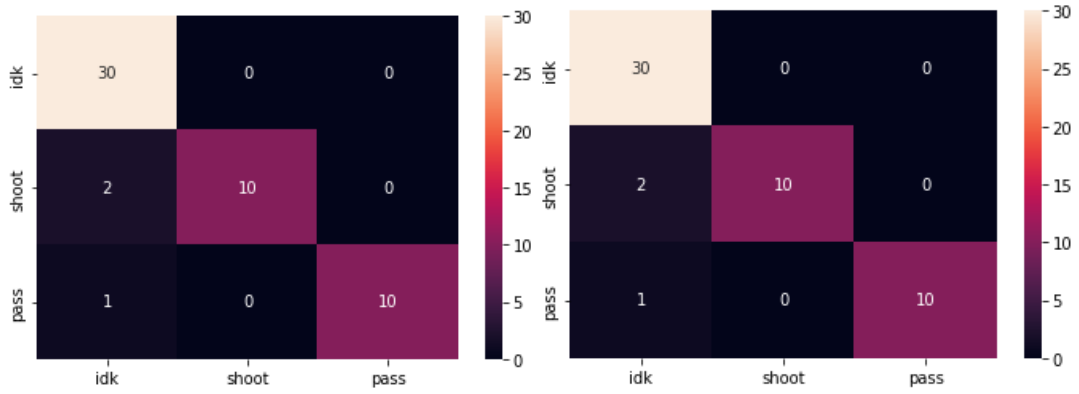

Figure 7 Matrices de confusion pour les données de test pour le modèle Logistic Regression (à gauche) et K Neighbors Classifier (à droite).

Les dernières expériences constituent les modèles avec réseau de neurone. Le principe est de garder le bloc normalisé sans extraire des variables et les passer à un réseau de neurone convolutif. Un tel réseau permet de ne pas over-fit les données avec un trop grand nombre de paramètres comme dans le cas avec des couches LSTM. Une première modélisation d'un modèle simple sur les couches suivantes :

Type	*Paramètres*
Conv1D	Filtres : 32 Taille kernel : 5
MaxPooling1D	Taille pool : 4
Conv1D	Filtres : 16 Taille kernel : 5
MaxPooling1D	Taille pool : 4
Conv1D	Filtres : 8 Taille kernel : 5
MaxPooling1D	Taille pool : 2
Flatten	-
Dense	Unités : 32
Dropout	Ratio : 0.5

Un entraînement s'effectue sur 50 *epochs* et des *batchs* de taille 8 avec 20% des données d'entraînement en tant que validation. Son entraînement dure 0.2 secondes et arrive à converge mais sans être stable comme le montre le graphe ci-dessous.

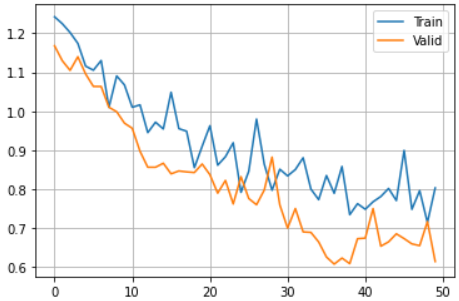

Figure 8 Fonction loss pour la validation et entraînement du modèle.

Pour donner suite à ces résultats peu convaincants, l'équipe a souhaité poursuivre ses recherches grâce à un outil appelé *Keras Tuner*. Il permet l'automatisation de création de modèles pour faciliter la recherche du meilleur. En vue de la rapidité de l'entraînement le projet se base sur une recherche aléatoire sur plus de 150 modèles différents basés sur la structure suivante :

- Conv1D + MaxPooling1D (1-2x) :
  - Filtres : 8,16,32
  - Taille kernel : 3,5,7
  - Taille pool : 2,3,4
- Flatten
- Dense (2-5x) :
  - Unités : 8,16,32
- Dropout :
  - Ratio : 0.3,0.4,0.5,0.6,0.7

Le meilleur modèle obtenu contient les couches suivantes :

Type	*Paramètres*
Conv1D	Filtres : 16 Taille kernel : 7
MaxPooling1D	Taille pool : 4
Conv1D	Filtres : 16 Taille kernel : 7
MaxPooling1D	Taille pool : 4
Flatten	-
Dense	Unités : 32
Dense	Unités : 32
Dropout	Ratio : 0.4

Comme le montre les résultats de précision ci-dessous, le modèle obtenu via le tuner résulte à de meilleures précisions dans les deux situations.

	Modèle à la main	*Meilleur modèle (tuner)*
% précision entraînement	79.69	*87.50*
% précision test	69.70	72.73

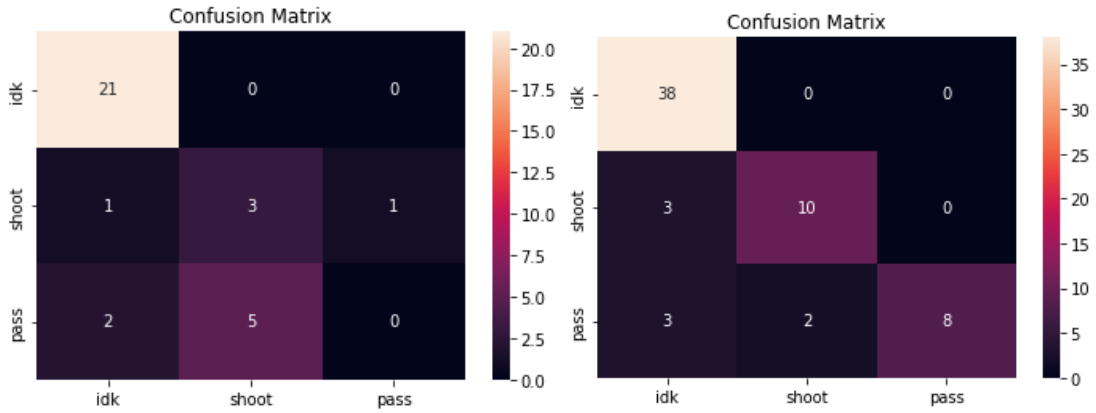

Figure 9 Matrices de confusion avec le meilleur modèle pour les données de test (à gauche) et d'entraînement (à droite).

6 ANALYSE ET CONCLUSION

En conclusion, ce projet a amené une approche scientifique et tente de résoudre un objectif commun au sein des différents sports possibles. Ici en particulier l'unihockey où des équipes de statistiques ou entraîneurs veulent posséder des données concernant les joueurs. Ainsi, ce travail calcule le nombre de passes ou de tirs pour une certaine série temporelle extraite d'un capteur positionné au niveau de la canne. En donnant suite à une analyse plus profonde, l'idée de détecter la différence entre backend et forward peut s'avérer difficile voir impossible avec la quantité de données. Pour ces raisons, ce projet se focalise sur la détection de passes et tirs seulement.

Ce choix permet d'obtenir de très bons résultats avec des modèles simples et trois simples features (range, max et le 1er quartile). En effet, via un classifieur utilisant la propriété des voisins les plus proches, la précision s'élève à ≈94% pour les données de test. Ces résultats même meilleurs en comparaison avec un réseau convolutif dont il obtient ≈72% via le meilleur modèle de la recherche.

Ces résultats offrent une première idée des statistiques d'un joueur puis peut être visualisé via l'application web de ce projet.