I. Table des matières

I. Table des matières 1
II. Contexte 2
III. Objectifs 2
IV. Périmètre de l'étude 2
V. Analyse des données 3
A. Prétraitements 3
B. Post-traitements 4
VI. Architectures réseau de classification de série temporelle 5
A. Principales architectures de classification de série temporelle 5
B. Choix de l'architecture 6
VII. Mise en place du réseau 7
A. Formatage des données 7
B. Représentation du modèle 7
C. Entraînement du modèle 8
D. Analyse des performances 9
VIII. Réflexion sur le modèle 10
A. Résultats en fonction de la fenêtre de temps 10
B. Traitement en temps réel 11
IX. Conclusions 11

II. Contexte

Jeune ingénieur diplômé de l'école nationale supérieure des arts & métiers, j'ai toujours été intéressé par les outils informatiques et l'aide que ces derniers apportent dans le milieu industriel. J'ai eu l'occasion de me confronter à différentes problématiques liées à l'intelligence artificielle dans des projets professionnels et personnels. Afin d'approfondir mes connaissances sur ces outils, et de m'initier au monde de la programmation informatique, je me suis inscrit au C.N.A.M. dans l'ensemble des modules constituant le certificat de spécialisation d'intelligence artificielle sur l'année 2020-2021.

III. Objectifs

Le but de ce rapport est de démontrer mes capacités de conception de réseau en traitant une problématique réelle de l'entreprise. Parmi celles proposées j'ai sélectionné le sujet « Step Detection » : la détection d'une foulée basée sur des données relevées par une centrale inertielle, maintenue auprès de la cheville. Il s'agit ici d'une classification de série temporelle, pour cela le premier objectif est le choix ainsi que la construction du réseau de neurones, qui construira la matrice de features, le second objectif est d'en analyser ses capacités de prédictions sur des séries temporelles.

IV. Périmètre de l'étude

Dans ce cas d'étude, il s'agit de réaliser des travaux permettant de détecter des débuts et de fins de foulées. Ces travaux serviront d'éléments de base pour de futures interprétations plus complexes au travers d'autres algorithmes de reconstruction. Il est donc essentiel que ce premier réseau ait une capacité de prédiction qui soit la plus fiable possible ceci afin de garantir une information correcte pour tous les traitements postérieurs.

Le périmètre d'étude étant défini, voici ce qui est à ma disposition : un ensemble de séries temporelles ainsi que leurs labels correspondants. Pour mener à bien cette étude, j'ai décidé de la séparer en plusieurs temps forts :

- Analyse des données
- Recherche d'architectures réseau pour la classification de série temporelle
- Mise en place du réseau choisi, et extraction des « features » et des prédictions
- Mesure des performances de prédictions de classification

Pour cette étude je vais concevoir un réseau pour faire du traitement en différé, et j'aborderai la réflexion du traitement en temps réel dans la section « Traitement en temps réel » (cf. VII.E).

Ci-après se trouve une analyse des données, ainsi qu'une mise en forme.

V. Analyse des données

Dans cette section nous allons mettre en forme les données afin de les visualiser et d'essayer d'en tirer des premiers éléments de réflexions sur les liens entre les séries et leurs labels. À noter que la fréquence des relevés est de 130 Hz, donc l'abscisse représente l'observation pour chaque 1/130ème de seconde.

A. Prétraitements

Après la récupération des données sous un format exploitable, j'ai tiré au hasard des séries temporelles et leurs labels afin de les tracer, voici quelques exemples :

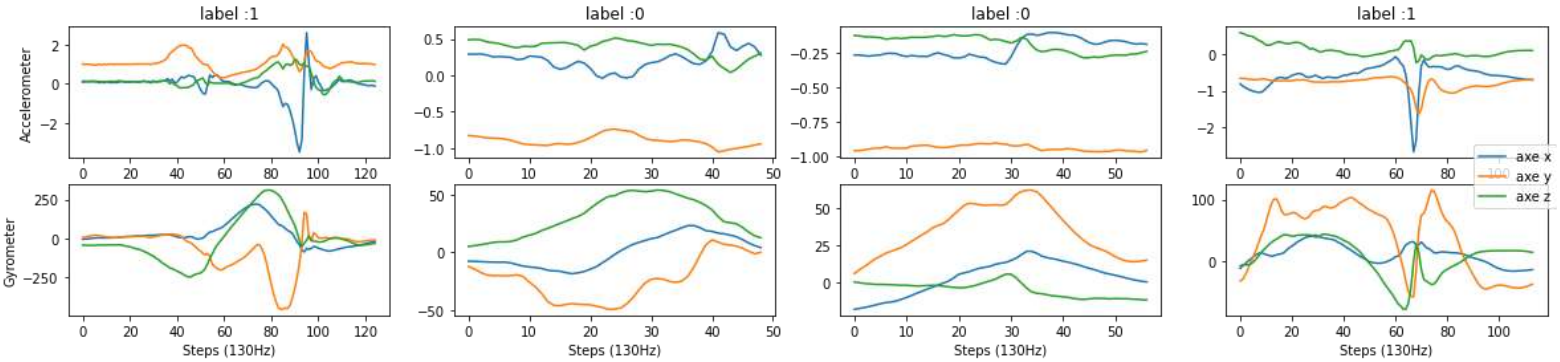

Figure 1 tracée aléatoire de 4 séries temporelles et leurs labels issus du jeu de donnée

Les images sont importées au format PNG, et sont également disponibles en annexe dans dossier du rapport.

Sur les graphiques ci-dessus, pour chaque label nous avons 2 séries temporelles associées qui représentent respectivement les accélérations et les vitesses angulaires autour des 3 axes cartésiens. Basé sur ces quelques graphiques, il est très difficile de faire des conclusions qui permettraient naturellement de classifier. Cependant, je distingue 2 tendances générales :

- Les labels « 1 » de début / fin de foulée semblent avoir des « pics » avec des valeurs numériques plus fortes
- Les labels « 0 » semblent être plus « bruités » ainsi que des variations sur l'axe des ordonnées plus faibles

Il est aisé de relever au travers de ces représentations que l'on retrouve un problème classique dans le traitement des séries temporelles : la non-uniformité des fenêtres. Pour mieux se rendre compte de cette problématique, je vais mettre en forme la donnée.

B. Post-traitements

Ci-dessous un graphique en barres regroupant par label, les séries ayant la même longueur d'enregistrement, en les triant des plus petites aux plus grandes. Pour rappel, dans cette étude, il est donné 6316 labels, dont 49% de « 0 », et 51% de « 1 ».

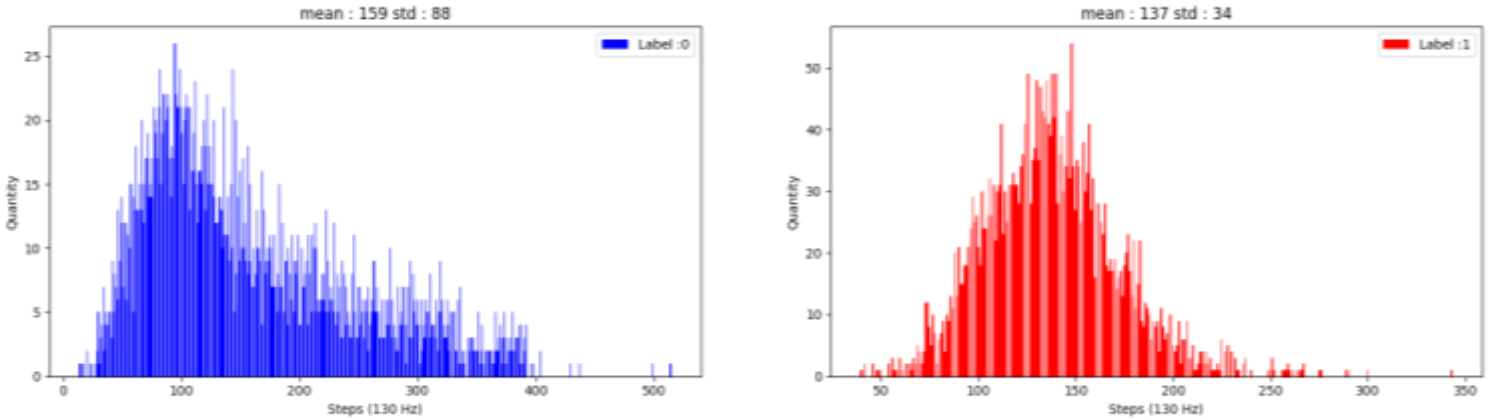

Figure 2 Graphiques en barres, représentant les quantités d'enregistrements ayant la même durée selon chaque label

Les titres contiennent la durée moyenne et l'écart type pour chaque catégorie. Voici également les 10 enregistrements les plus courts et les plus longs pour chaque label.

- Label « 0 » :

10 enregistrements les plus courts										
Longueurs	14	15	16	18	20	23	24	28	29	30
Quantités	1	1	1	1	2	1	1	1	2	5
10 enregistrements les plus courts										
Longueurs	390	391	392	396	398	404	429	437	499	514
Quantités	3	2	4	1	1	2	1	1	1	1

- Label « 1 » :

10 enregistrements les plus courts										
Longueurs	40	41	42	46	48	49	50	54	56	57
Quantités	1	1	2	2	1	1	1	2	3	1
10 enregistrements les plus courts										
Longueurs	264	265	266	267	275	276	289	290	300	343
Quantités	1	1	1	2	1	1	1	1	1	1

Malgré un équilibre entre les 2 catégories, ces graphiques permettent de mettre en avant la répartition des données fournies, ainsi les labels « 1 » sont plus concentrés et plus nombreux autour d'une moyenne d'à peu près 1 seconde d'enregistrement tandis que les labels « 0 » sont répartis de façon un peu plus uniforme.

La plage globale des fenêtres est très large : de 14 à 514 étapes soit un facteur de ~ 36 entre ces extrêmes. Ce traitement soumet l'idée que le modèle doit être en mesure d'identifier une foulée sur 14 étapes, ce qui correspond à ~ 1/10ème de seconde, mais également sur un enregistrement de plus de 4 secondes. Cela représente un vrai défi pour n'importe quels réseaux neuronaux, il serait donc pratique de diminuer cette plage.

Comme l'objectif est de détecter un mouvement, il est tout à fait envisageable de se contenter de la plage du label « 1 », allant de 40 à 343 étapes, ce qui donne un facteur de ~ 8,5 entre ces extrêmes.

Par la suite, les données sont sélectionnées dans cette optique de réduire cette variation temporelle, cependant j'utiliserai une plage légèrement plus grande, allant de 30 à 343 étapes (facteur ~11,5) afin de laisser plus de souplesse au modèle. Les séries en dehors de cette plage ne sont donc pas prises en compte, à noter que ces exclusions n'affectent pas l'équilibre entre les 2 classes.

Théoriquement, cette fenêtre de 30 étapes permet également de faire une analyse d'un mouvement sur une période minimum doublée, ~ 2/10ème de seconde, cet élargissement donne l'opportunité de capter plus d'informations et par conséquent facilite l'interprétation ainsi que la classification.

VI. Architectures réseau de classification de série temporelle

La classification de série temporelle est une tâche complexe, car elle nécessite de reconnaitre des motifs clés afin de les attribuer à un label, or ces derniers peuvent avoir une représentation spatio-temporelle différente d'une mesure à l'autre, sans compter que les durées d'enregistrement sont variables, ce qui est une difficulté supplémentaire.

A. Principales architectures de classification de série temporelle

Il existe une multitude d'algorithmes de classification des séries basés sur la spatialité des observations, en voici les principales :

- « Dynamic Time Warping » : permets de comparer des séries en utilisant la matrice d'alignement
- « Clustering » : regroupe les séries temporelles en se basant sur une distance ou un critère de similarité
- « Proximity Forest » : est un ensemble d'arbres de décision qui sont divisés sur une mesure de distance élastique
- « Shapelet-Based Classifiers » : utilise la similitude entre un « shapelet » et une série comme une caractéristique discriminatoire

Ci-dessous une autre liste d'algorithme, cependant ceux-ci qui extraient les « features » des séries temporelles, avant d'en prédire une classification :

- « Deep naive » : Série de couches complètement connectées
- « Convolutional Network » : Utilisation de couche convolutive successive, puis de couches linéaires

- « Inception Time » : Convolution “Bottleneck” suivie de convolution superposée avec des noyaux différents
- « Recurrent Neural Network » : Traitement de la série temporelle à l’aide d’un réseau récurrent

Comme il est demandé d’extraire des « features » des données, c’est la seconde liste d’algorithme qui est intéressante dans ce cas.

B. Choix de l’architecture

Je vais décrire brièvement les avantages et /ou inconvénients des différents algorithmes qui permettent l'extraction de features pour justifier mon choix final. Tout d’abord les réseaux de neurones profonds dit « naïfs » ont beaucoup de paramètres, et une convergence plus lente. De plus ils n'extraient pas des « features interprétables ». Ils ne sont donc pas retenus pour la suite de l’étude.

Les réseaux de neurones convolutifs sont une des techniques les plus populaires pour cette tâche, car ils sont capables de capturer les motifs spatiaux temporels grâce à l'utilisation de filtres entraînables, à noter toutefois qu'il faut une fenêtre d’entrée fixe. Ils sont donc une piste intéressante pour l’extraction de « features ».

Le réseau « Inception Time » est basé sur le principe des couches convolutives, cependant la technique globale est légèrement plus complexe à mettre en place, comme précédemment il est nécessaire d’avoir une fenêtre à taille fixe à l’entrée de ce modèle. Je trouve qu’il serait intéressant de le tester en comparaison de l’architecture convolutive « classique ». C'est un modèle à réserver dans le cas d'une étude plus approfondie.

Les réseaux récurrents semblent répondre à cette problématique de variabilité de la fenêtre d’entrée, cependant ces réseaux sont souvent utilisés pour faire de la prédiction et non de la classification, ce qui se comprend par son architecture, il est également moins « intuitif » d’en extraire des « features ». De plus ce type de réseaux supporte mal la complexification du modèle, et par conséquent, plus difficile à manier.

Dans le cas de cette étude, et afin de démontrer mes capacités à concevoir un réseau dans un temps bref, je vais choisir l’architecture convolutive « classique », qui correspond au besoin d’extraire des « features » et d’en déduire une classification. Le bon fonctionnement de ces modèles repose toutefois sur la taille et de la qualité des données d'apprentissage, en particulier, la longueur de la série temporelle.

Dans la section suivante, je vais concrétiser concevoir le réseau, formater les données, et faire des prédictions.

VII. Mise en place du réseau

Dans les sections suivantes, je vais formater la donnée pour que cette dernière soit traitable par le réseau convolutif, que je représenterai schématiquement. J'aborderai également l'aspect de l'entraînement et des performances obtenus.

A. Formatage des données

Comme énoncé au préalable, il faut une fenêtre fixe en entrée de notre réseau, cependant les données que nous avons sont établies sur une plage de 30 à 343 étapes (cf. V.B). Pour uniformiser ces observations, je vais utiliser le principe de la fenêtre glissante sur tous les enregistrements, avec une taille fixée par le paramètre « steps ». Par défaut ce paramètre est égal à la taille de la fenêtre le plus petite, ici « steps = 30 ».

Ceci à l'avantage d'augmenter la quantité de données à notre disposition, passant de ~6 000 à ~700 000 échantillons (facteur ~115), la répartition entre les labels reste équilibrée : 50.5% « 0 » et 49.5% « 1 ».

Ces échantillons nouvellement obtenus sont mélangés, puis séparés en 3 jeux : Entraînement, Validation, Test avec un ratio respectif de 0,7 ; 0,15 ; 0,15. La taille des jeux étant conséquente, j'ai généré des « batchs » qui représente 1/100ème du volume originale. Ce sont ces données ainsi formatées qui seront utilisées lors de l'entraînement du modèle.

B. Représentation du modèle

Voici un schéma représentatif des différentes couches de traitement appliqué au modèle :

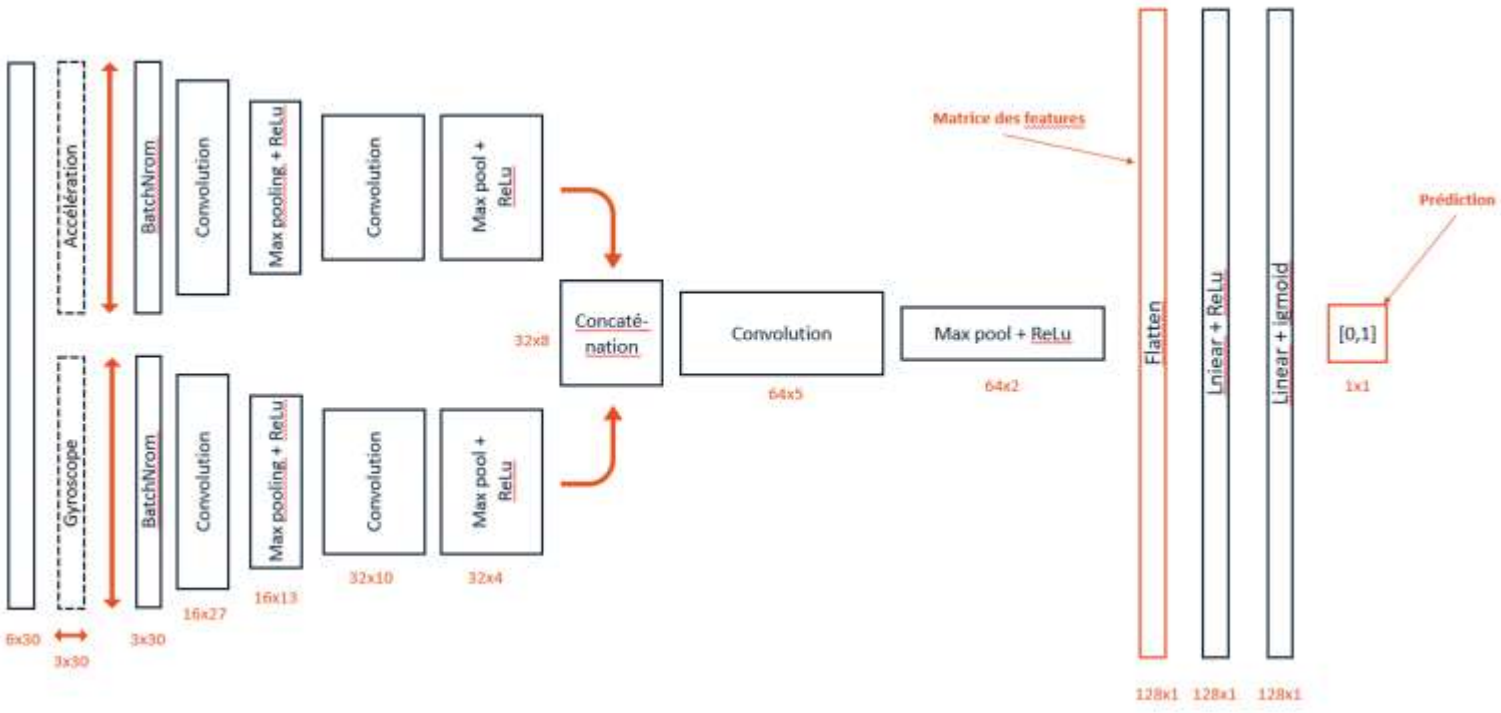

Figure 3 Schéma de l'architecture réseau

Les données d'accélérations et de vitesse angulaire n'exprimant pas la même mesure physique, elles sont d'abord traitées séparément. Chacune passe par les couches suivantes :

- Une couche de normalisation par Batch
- Une couche de convolution extrayant 16 filtres sur les 3 séries temporelles, avec un noyau de 4 étapes
- Une couche de maxpooling, avec un noyau de 4 étapes et un « stride » de 2 étapes
- Une activation non linéaire : ReLu
- Une seconde couche de convolution extrayant 32 filtres sur les 16 canaux, avec un noyau de 4 étapes
- Une couche de maxpooling, avec un noyau de 4 étapes et un « stride » de 2 étapes
- Une activation non linéaire : ReLu

Les 2 matrices ainsi obtenues sont ensuite concaténées, avant de passer dans une 3 ème couche convolutive :

- Dernière couche de convolution extrayant 64 filtres sur les 32 canaux, avec un noyau de 4 étapes
- Une couche de maxpooling, avec un noyau de 4 étapes et un « stride » de 2 étapes
- Une activation non linéaire : ReLu

La matrice issue de cette 3ème couche convolutive est ensuite aplatie et elle définit la matrice de features du modèle. Ces features extraites par ces couches de convolutions successives, vont ensuite être traité par la seconde partie du réseau :

- Une couche complètement connectée de 128 neurones
- Une activation non linéaire : ReLu
- Une seconde couche complètement connectée de 128 neurones
- Une activation non linéaire : Sigmoid

La matrice obtenue est une matrice de nombre flottant, il faut l'arrondir à l'entier le plus proche pour obtenir la prédiction booléenne du modèle.

C. Entraînement du modèle

Voici les fonctions utilisées pour l'entraînement :

- La fonction de calcul du loss : utilise le principe de l'entropie binaire croisée
- L'optimisateur : ADAM avec un pas initial de 0.01

Chaque itération, le modèle fait une passe complète sur les jeux d'entraînement et de validation. Comme le démontre le graphique ci-dessous, au bout de 90 itérations il y a une apparition de bruit de plus en plus ample pour le jeu de validation. En s'appuyant sur le principe de « early stopping » cela signifie que le modèle est en train de surapprendre et qu'il est temps d'arrêter.

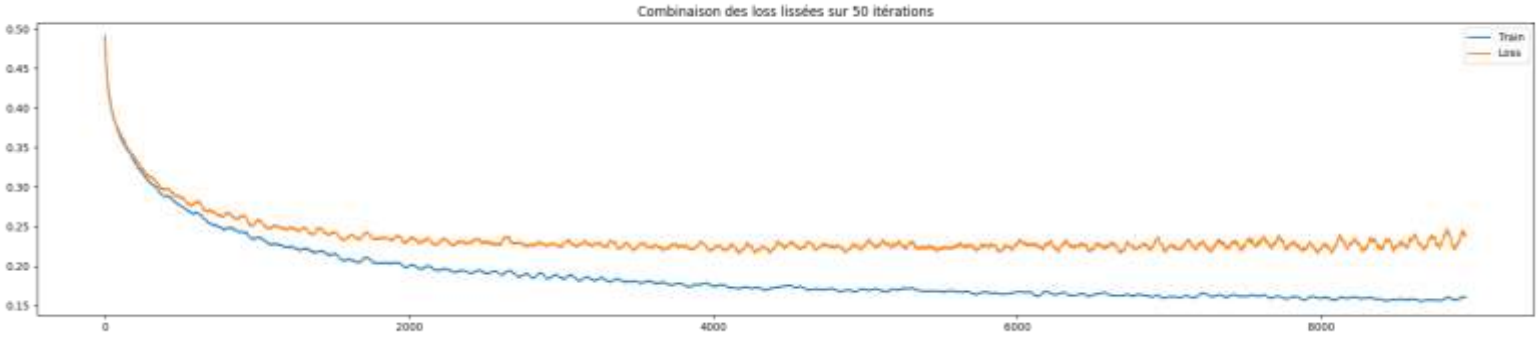

Figure 4 Combinaisons des loss lissées sur 50 itérations

D. Analyse des performances

Afin de mesurer les performances du modèle, je vais m'appuyer sur les matrices de confusion, appliquées à chacun des jeux, pour évaluer la capacité d'apprentissage ainsi que sa robustesse.

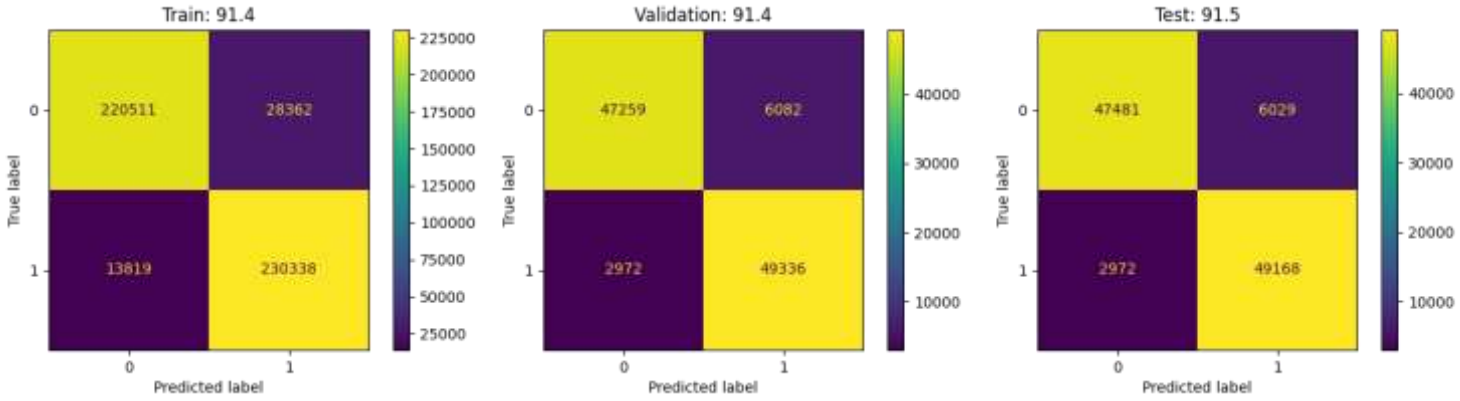

Figure 5 Matrice de confusion pour l'ensemble des jeux de données

Le modèle démontre une stabilité de prédiction pour chacun des jeux de données, avec une précision globale aux alentours de 91%. Les précisions de prédictions pour chacune des classes sont équilibrées, cependant il fait environ 2 fois plus d'erreurs dans la prédiction de label « 0 » que dans les labels « 1 ».

Il serait intéressant de prolonger l'analyse en projetant sur des séries temporelles de labels « 0 » et par fenêtre de 30 étapes successives la prédiction du modèle, peut-être que ce dernier identifiera des mouvements isolés, qui ne sont pas catégorisés comme un début ou une fin de foulée. De la même manière, observer les erreurs prédites par le modèle sur les labels 1, pour identifier des « temps morts » avant ou après une fin de foulée.

Globalement cette première conception semble être un point de départ solide pour la détection de foulée, et il serait intéressant d'ajuster les paramètres pour grappiller quelques pourcentages de précision supplémentaires. Il est également un bon élément de comparaison pour le réseau convolutif plus complexe : Time Inception.

VIII. Réflexion sur le modèle

Ci-dessous quelques éléments de réflexion sur d'autres résultats obtenus par le modèle, ainsi que sa capacité à faire du traitement en temps réel.

A. Résultats en fonction de la fenêtre de temps

Lors de la conception du modèle, j'ai réalisé des calculs avec des fenêtres fixé à 15 et 40 étapes, voici les matrices de confusion obtenues :

- 15 étapes :

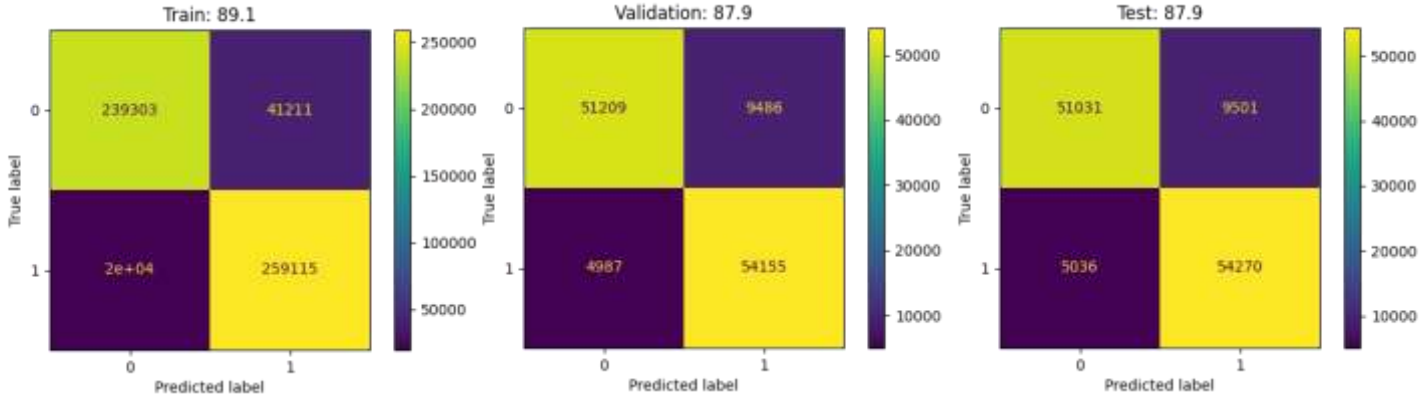

Figure 6 Matrice de confusion obtenues par le modèle pour des fenêtres de 15 étapes

- 40 étapes :

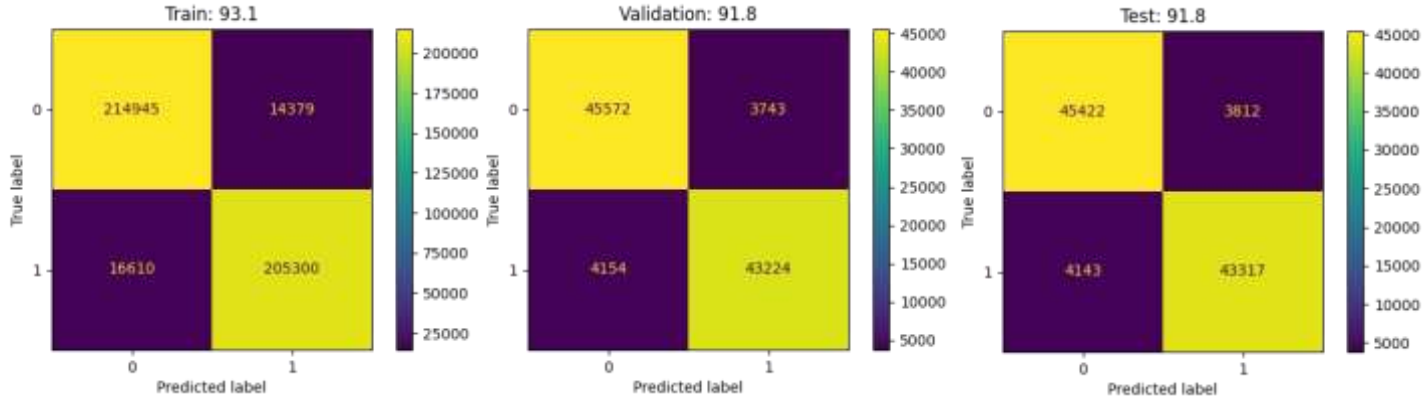

Figure 7 Matrice de confusion obtenues par le modèle pour des fenêtres de 40 étapes

Pour que le modèle fonctionne avec une fenêtre d'entrée de 15 étapes, il m'a fallu modifier le « stride » de la fonction de « max pooling » en l'abaissant de 2 à 1.

Ces résultats montrent qu'en doublant la fenêtre de 15 à 30 étapes, le modèle gagne ~3,5% de précision supplémentaire, cependant en augmentant le fenêtre de 30 à 40 étapes, il ne gagne que ~0,3%. Une modeste augmentation pour un coût de calcul plus grand. Cette stabilité autour des fenêtres de 30 étapes suggère que le modèle est apte à prédire sur cette largeur d'intervalle, et que pour obtenir des meilleures performances il faut trouver de meilleurs paramètres du réseau.

B. Traitement en temps réel

Il est envisageable d'utiliser l'architecture précédente pour s'approcher d'une technique en temps réel, en demandant continuellement au modèle de faire une prédiction sur les 30 dernières mesures. Cependant, il faudrait que la prédiction se fasse en moins de 1/130ème de seconde pour être capable de suivre la cadence des relevées d'observations, sur mon PC voici le temps moyen obtenu pour une prédiction du modèle sur le CPU :

- 602 µs ± 20.2 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

Dans ce cas il faudrait diviser le temps de calcul par 80, ce qui est envisageable en utilisant un hardware spécifique, et un langage de plus bas niveau. Ainsi le modèle précédent pourrait servir dans du traitement en temps réel.

Du côté de la conception, il est également possible de simplifier les calculs et par conséquent gagner en rapidité. Cela se fait en s'affranchissant de certaines couches et donc de centaines de paramètres, tout en essayant de préserver au mieux la qualité de prédiction. Cette idée, va à l'encontre de la conception de modèle plus complexe qui seraient probablement en mesure d'avoir une meilleure capacité modélisation des données. Il faut trouver une balance entre ces 2 concepts.

Une autre piste de réflexion autour de la conception, serait de changer totalement du modèle actuel en privilégiant un réseau neuronal récurrent, ce dernier traitera les données de façon continue, applicable dans le cas d'un traitement en temps réel.

IX. Conclusions

Le traitement de série temporelle est un vrai défi en soi, et celui de sa classification en est un à part entière ! Pour y faire face, je me suis renseigné sur les différentes techniques existantes, mesurer le pour et le contre de chaque avant d'engager des travaux.

Le fait que le modèle est des prédictions relativement correctes permet de confirmer les dires des diverses sources sur l'efficacité des architectures convolutives dans l'exécution de cette tâche.

Cet exercice m'a fait explorer un nouvel aspect du « machine learning », et ces outils m'ont encore étonné par leurs capacités de modélisation sur des données complexes. J'espère que ce travail sera être apprécié de mon évaluateur.

Comme ouverture, voici d'autres réseaux / méthodes qu'ils seraient intéressants de mettre en place et d'en comparer les performances :

- Combinaison du modèle actuel avec des cellules « Long Short Term Memory »
- Time Inception
- L'utilisation de réseaux neuronaux récurrents
- RNN « Deep Kalman Filter »