Reconnaissance d'actions à l'aide d'une approche hiérarchique basée sur l'apprentissage machine

par

Nicolas LEMIEUX

MÉMOIRE PAR ARTICLE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

MONTRÉAL, LE 17 DÉCEMBRE 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE:

Mme Rita Noumeir, directrice de mémoire
Département de génie électrique à l'École de technologie supérieure

M. David Labbé, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Jean-Marc Lina, membre du jury
Département de génie électrique à l'École de technologie supérieure

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 1 DÉCEMBRE 2020

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Reconnaissance d'actions à l'aide d'une approche hiérarchique basée sur l'apprentissage machine

Nicolas LEMIEUX

RÉSUMÉ

Actuellement, les méthodes de reconnaissance d'actions proposées par la littérature performent assez bien sur les actions simples, mais requièrent, bien souvent, des puissances de calcul importantes et éprouvent encore de la difficulté à reconnaître les actions ayant des séquences d'exécution dont la temporalité est variable. Par conséquent, la méthode présentée dans le cadre de ce mémoire adressera ces difficultés.

Pour ce faire, celle-ci propose un cadre hiérarchique d'apprentissage. Ainsi, la reconnaissance des différentes actions s'appuie sur la signature de sous-mouvements localisables dans le temps. Plus précisément, ces signatures correspondent aux valeurs maximales et minimales d'activations de filtres de convolution appris dans le cadre d'un problème de classification dont l'entraînement procède à l'aide d'une méthode de descente de gradient.

Tout en démontrant une grande économie en termes de calculs et une plus grande robustesse par rapport à l'information superflue, le taux de succès de classification obtenu par l'approche proposée est comparable à l'état de l'art lorsque les données sur lesquelles elle s'appuie sont issues d'un capteur inertiel porté à même le corps.

Mots-clés: Séries-temporelle, apprentissage machine, méthode hiérarchique, reconnaissance d'action

A Learning Based Hierarchical Approach for Human Action Recognition

Nicolas LEMIEUX

ABSTRACT

Nowadays, action recognition's literature proposes methods performing relatively well on simple actions, but are typically computationally heavy and less suited for actions with inconsistent sequences of execution. Hence, the method proposed in the article attached to this master's thesis addresses these difficulties.

More specifically, it proposes a hierarchical learning approach that consists in recognizing the signature of some time-localized dynamics. Explicitly, these signatures correspond to the maximum and minimum activation of convolution kernels learned throughout the process.

While being computationally cheap and robust to redundant/unnecessary information, the proposed method also achieves near state-of-the-art performances in terms of classification accuracy when applied on data provided by wearable inertial sensors.

Keywords: Action recognition, time-series, hierarchical method

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

1D-CNN	Réseau à convolutions sur une dimension
HAR	Human action recognition
IMU	Inertial measurement unit
IoT	Internet of Things
k-NN	k-nearest neighbour
LSTM	Long short-term memory
MLP	Multi-layer-perceptron
MAC	Multiply–Accumulate operation
ReLU	Rectified Linear Unit
RNN	Recurrent neural network
SeLU	Scaled exponential linear unit
SVM	Support vector machine
UTD-MHAD	University of Texas at Dallas-Multimodal Human Action Dataset

LISTE DES SYMBOLES ET UNITÉS DE MESURE

M	Nombre total d'exemples d'entraînement
S_c	Nombre d'exemples de la classe c
$\hat{y}_c$	Vecteur des probabilités inférées
$\mathbf{y}$	Étiquette mise forme de vecteur
y_c	Valeur de l'étiquette pour une classe spécifique
c	Indice référant à une classe spécifique
η	Taux d'apprentissage
N	Nombre total de classes
σ	Fonction d'activation
k	Indice référant à un filtre de convolution spécifique et/ou une dimension de la série temporelle
l	Indice référant à une couche de convolutions
t	Indice référant à un moment particulier d'une série temporelle
m	Indice référant à la taille le long de l'axe temporelle d'un filtre de convolution
θ	Ensemble des paramètres d'un réseau de neurones
Θ	Ensemble des hyper-paramètres d'un réseau de neurones
$\mathcal{V}$	Vecteur représentatif
$\mathbf{W}$	Banque de filtres
T_l	Nombre de pas de la $l^{\text{ième}}$ série temporelle
F_l	Nombre de filtres de convolutions de la $l^{\text{ième}}$ couche
$\mathbf{X}$	Série temporelle
γ	Accélération
ω	Variation de la vitesse angulaire

INTRODUCTION

De nos jours, les données occupent de plus en plus de place dans nos vies. En effet, l'augmentation des puissances de calcul, des modalités d'acquisition et des variétés de flux permettent à un nombre toujours croissant d'applications de voir le jour. De par son potentiel à offrir un support personnalisé à plusieurs applications intriquées dans différentes aires de recherche telles que l'analyse sportive ou l'ergonomie des interfaces humain-machine, la reconnaissance d'actions est certainement un des domaines qui a su en tirer avantage et sera le thème abordé par ce mémoire. Plus précisément, celui-ci s'inscrit comme l'aboutissement de deux années de recherches menées en partenariat avec Aerosystem International dans le but de développer une solution automatisée de détection d'événements pouvant compromettre l'intégrité physique de la population carcérale canadienne.

Bien que les applications découlant des algorithmes de reconnaissance d'actions soient relativement récentes, on s'aperçoit déjà du caractère incisif qu'elles peuvent avoir sur la vie des gens. En effet, à certains endroits cette technologie est mise de l'avant dans l'application de systèmes de crédit social par lesquels des citoyens se voient octroyer ou révoquer des privilèges. Ainsi en accord avec *la déclaration de Montréal pour un développement responsable de l'intelligence artificielle* demandant que les systèmes d'acquisition et d'archivage garantissent l'anonymisation des profils personnels [1] il a été entendu que la méthode développée s'appuie sur des flux d'information ne permettant pas d'identifier directement les individus.

À cet escient, les flux de données considérés lors de mes recherches ont porté sur des signaux représentant le corps humain par une vingtaine de points d'intérêts, dérivés du flux de profondeur d'une caméra 3D ainsi que sur les signaux issus d'un capteur inertiel porté à même le corps. En raison des meilleurs résultats empiriques obtenus, c'est sur la base de cette dernière modalité que

[1] Cinquième point du principe de protection de l'intimité et de la vie privée. https://www.declarationmontreal-iaresponsable.com/la-declaration

l'article s'inscrivant dans ce mémoire, Lemieux & Noumeir (2020), abordera la reconnaissance d'actions.

La méthode qui y est décrite propose une solution à deux des principaux problèmes relatifs à ce domaine de recherche, en lien avec la problématique. D'une part, contrairement aux approches les plus populaires, la méthode proposée est très économe en calculs, lui permettant ainsi d'être intégrée à des systèmes aux ressources limitées. Conséquemment, l'analyse des signaux pourrait être prise en charge localement et seuls les scénarios jugés critiques seraient communiqués, respectant ainsi plus la vie privée des détenus. D'autre part, de par son aspect hiérarchique, c'est-à-dire dans lequel les instances de haut niveau, soient les actions, sont composées par des ensembles d'instances de bas niveau, soient la signature de dynamiques discriminantes, l'article fait aussi état de résultats indiquant que la méthode proposée est mieux adaptée à la reconnaissance d'actions dont la séquence d'exécution est variable que celles de la littérature. Comme les bagarres se composent généralement de quelques mouvements brusques (coup de poing ou de pied) entre coupés de périodes plus ou moins longues de lutte, cette propriété est donc d'un intérêt certain dans le cadre de l'application visée.

Finalement, la revue de la littérature présentée au chapitre 1 présentera la reconnaissance d'actions sous un spectre large en donnant une description des différentes modalités de signaux, puis un bref survol des méthodes se basant sur les points d'intérêts alors que celles basées sur les signaux inertiels seront revues au chapitre 3 dans le cadre de l'article rattaché à ce mémoire. Au chapitre 2, les assises méthodologiques et raisonnements clés ayant mené à la méthode proposée seront présentés. Enfin, un dernier chapitre présentant les conclusions et les recommandations clôturera ce mémoire.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

La reconnaissance d'action est une branche de l'informatique qui a vu le jour au cours des années 80s. Il existe deux familles d'approches permettant de faire de la reconnaissance d'action. D'une part, il y a les méthodes se basant sur des flux visuels et puis celles s'appuyant plutôt sur des mesures de nature inertielle.

1.1 Flux visuels

Tout d'abord, il existe plusieurs types de flux visuels. Comme l'illustre la figure 1.1 ces flux peuvent être soit un flux d'images (vidéo), un flux d'images de profondeur (acquis à l'aide d'une caméra 3D) ou encore une représentation exosquelettique qui consiste en une poignée de points d'intérêts (correspondant aux articulations principales du corps humain) obtenus suite à un traitement de l'information relative au flux de profondeur.

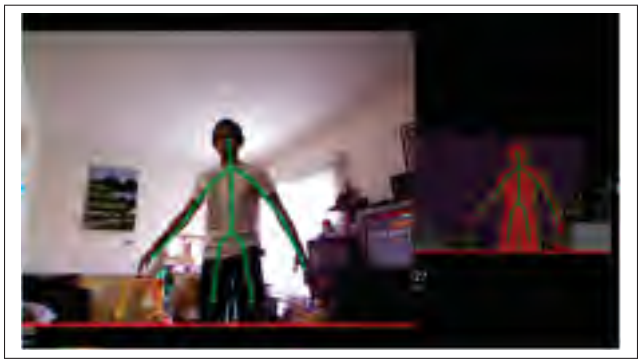

Figure 1.1 Modalité des flux visuels

Ces méthodes ont l'avantage d'être non-invasives et jouissent d'une information généralement plus riche que celle fournie par les capteurs inertiels, cependant leur utilisation est contrainte à des espaces délimités et les puissances de calculs requises à leur mise en place sont importantes.

Au fil du temps, plusieurs bases de données ont été proposées afin de mettre au défi les différentes méthodes (De la Torre, Hodgins, Bargteil, Martin, Macey, Collado & Beltran, 2009; Müller,

Röder, Clausen, Eberhardt, Krüger & Weber, 2007; Ni, Wang & Moulin, 2011; Fothergill, Mentis, Kohli & Nowozin, 2012; Wang, Liu, Wu & Yuan, 2012; Xia, Chen & Aggarwal, 2012b; Bloom, Makris & Argyriou, 2012; Yun, Honorio, Chattopadhyay, Berg & Samaras, 2012; Ofli, Chaudhry, Kurillo, Vidal & Bajcsy, 2013; Wan, Zhao, Zhou, Guyon, Escalera & Li, 2016; Shahroudy, Liu, Ng & Wang; Liu, Hu, Li, Song & Liu, 2017a), or, ces dernières années, celle proposée par l'université de Nanyang à Singapour, NTU-RGB-D Shahroudy *et al.* est devenue un véritable jalon dans le domaine de la reconnaissance d'actions étant donnée la diversité qu'elle propose au niveau de ses actions, de ses points de vue, de ses sujets (acteurs), ainsi que pour son grand nombre d'exemples.

Étant donné que pour les flux visuels seul le flux exosquelettique fut jugé suffisamment anonyme pour l'application visée, seules les méthodes y ayant recours seront revues ci-dessous.

1.1.1 Approches traditionnelles

Principalement basées sur des méthodes de classification telles que les machines à vecteurs de support, les arbres de décision ou la méthode des plus proches voisins, les premières approches utilisées dans le domaine de la reconnaissance d'actions faisaient appel à différentes méthodes statistiques afin de distinguer différentes actions sur la base de signaux générés lors de leur exécution. En plus de devoir permettre la reconnaissance d'une action par rapport à une autre, celles-ci visaient aussi à minimiser l'impact des variations corporelles et/ou de points de vue.

Dans cette optique, Gavrila, Davis et al. (1995) suggérèrent d'étudier les angles formés par certains ensembles de points, alors que Xia, Chen & Aggarwal (2012a) proposèrent un système de coordonnées dont l'origine se situait au niveau du bassin. En plus d'étudier les relations spatiales, Yang & Tian (2012) proposèrent de normaliser les séquences en soustrayant la valeur des signaux au premier instant de ceux qui suivent et dérivèrent aussi des informations de nature cinétique.

En calculant la matrice de covariance sur l'entièreté de la séquence d'exécution d'une action et en créant un espace de représentation basé sur ses éléments, Hussein, Torki, Gowayyed & El-

Saban (2013) réussirent à faire avancer l'état de l'art avant d'être surpassé par Vemulapalli, Arrate & Chellappa (2014) qui proposèrent de représenter les différentes actions par une courbe dans un hyperespace constitué de points appartenant à un groupe de Lie obtenus en représentant les différents segments de l'exo-squelette par une combinaison de rotations et de translations.

Finalement, soulevant le fait que deux actions peuvent être exécutées simultanément, Wei, Zheng, Zhao & Zhu (2013) introduisirent une approche à noyaux multiples basée sur les coefficients issus d'une transformée par ondelettes appliquée à la trajectoire des différents points d'intérêts. En profitant de la propriété de localisation temps/fréquence, ils construisirent aussi une représentation prenant en compte la logique temporelle d'exécution. Au final, les performances de leur approche se sont avérées comparables à l'état de l'art du moment.

1.1.2 Approches basées sur les réseaux de neurones artificiels

Ces dernières années, les progrès réalisés dans le domaine de l'apprentissage machine permirent d'augmenter considérablement la performance des algorithmes en termes de classification. En effet, comme le note Herath, Harandi & Porikli (2017), depuis que les méthodes d'apprentissage profond sont utilisées dans le cadre de la reconnaissance d'actions, l'état de l'art est surpassé très fréquemment. Bien que ces méthodes aient l'avantage d'apprendre automatiquement représentations et critères discriminants, il n'en reste pas moins que beaucoup d'efforts sont mis en oeuvre afin de proposer de meilleures architectures. En pratique, une représentation des signaux d'entrée est apprise à l'aide de réseaux à convolutions et/ou de réseaux récurrents, tandis que les critères discriminants sont appris avec un perceptron multicouche.

1.1.2.1 Réseaux récurrents

Bien que les réseaux récurrents soient efficaces pour capter des informations contextuelles, ils le sont toute fois moins pour les relations spatiales. Ainsi dans le but d'apprendre certaines relations inhérentes entre certains joints, en plus de l'information temporelle, Zhu, Lan, Xing, Zeng, Li, Shen & Xie (2016) alternèrent des couches récurrentes avec des couches entièrement

connectées. Poursuivant le même objectif, Liu, Shahroudy, Xu, Kot & Wang (2018) proposèrent plutôt une séquence de traitement des points d'intérêts au lieu de le faire en parallèle. Finalement, un mécanisme d'attention apprenant à accorder plus d'importance à certains points d'intérêts en fonction de l'information contextuelle a été proposé par Liu, Wang, Hu, Duan & Kot (2017b), améliorant, une fois de plus, les performances des méthodes basées sur des réseaux récurrents.

1.1.2.2 Réseaux à convolutions

Pour ce qui est des réseaux de neurones à convolutions, l'approche la plus populaire consiste à représenter les séquences de points d'intérêts sous la forme d'une image. Pour ce faire, Du, Fu & Wang (2015) proposèrent l'analogie suivante : largeur, hauteur et canaux d'une image correspondent respectivement aux différentes mesures temporelles, points d'intérêts et coordonnées tridimensionnelles de ceux-ci. En ayant recours à cette représentation, Ke, Bennamoun, An, Sohel & Boussaid (2017) observèrent qu'il était possible de tirer avantage de réseaux préentraînés en comparant deux architectures VGG19 dont les poids avaient été initialisés ou non avec la base de données ImageNet. Puis, peu de temps après Li, Zhong, Xie & Pu (2018) proposèrent une architecture très légère, capable de capter des relations sur l'ensemble des joints contrairement aux architectures issues de la reconnaissance d'images ne couvrant que des sous-ensembles de points d'intérêts en raison de la petite taille de leurs filtres.

Adressant aussi ce problème, Yan, Xiong & Lin (2018) proposèrent de représenter les liens physiques unissant les différents points d'intérêts par une matrice de contiguïté qu'ils multiplièrent aux résultats de convolutions appliquées sur chaque joint le long de l'axe temporel. Argumentant que certaines relations importantes ne sont pas nécessairement liées physiquement, comme celles unissant les mains par exemple, Ye, Tang, Wang & Liang (2019) réussirent à améliorer ces résultats en proposant un mécanisme apprenant une matrice de contiguïté reflétant des relations plus significatives.

1.1.2.3 Architecture hybrides

Des méthodes tirant avantage de ces deux types de réseau font aussi partie de la littérature. Par exemple, Chuankun Li, Pichao Wang, Shuang Wang, Yonghong Hou & Wanqing Li (2017) proposèrent de fusionner les vecteurs représentatifs issus de ces deux architectures et observèrent un gain en performances par rapport à chacune des approches prises individuellement. Finalement en y ajoutant aussi plusieurs représentations issues de méthodes statistiques, Zhang, Lan, Xing, Zeng, Xue & Zheng (2019) arrivèrent à une méthode aux performances impressionnantes, mais nécessitant énormément de calculs.

1.1.3 Flux Inertiel

Pour ce qui est de la reconnaissance d'actions basée sur des données inertielles, les signaux utilisés sont issus de centrales inertielles comme celle illustrée à la figure 1.2. Généralement, ces modules de petite taille sont intégrés à l'intérieur d'objets portés à même le corps, comme c'est le cas pour les montres intelligentes ou encore au sein d'objets tels que les téléphones intelligents.

Figure 1.2 Station inertielle avec 6 degrés de liberté

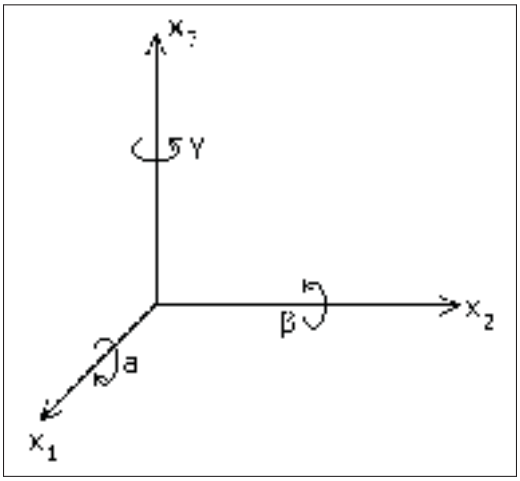

Figure 1.3 Repère cartésien illustrant les six degrés de liberté d'un capteur inertiel

L'information que ces capteurs permettent de recueillir consiste à la variation de vitesse (accélération) et de la vitesse angulaire selon les différents axes d'un système de coordonnées orthonormal tel qu'illustré à la figure 1.3.

Pour le reste de ce mémoire ainsi que dans l'article qui s'y rattache, les signaux inertiels seront identifiés par le vecteur de droite de l'égalité 1.1 alors que celui de gauche explicite la nature des signaux en fonction de la figure 1.3.

$$\begin{bmatrix} \frac{dx_1}{dt^2} \\ \frac{dx_2}{dt^2} \\ \frac{dx_3}{dt^2} \\ \frac{d\theta_\alpha}{dt^2} \\ \frac{d\theta_\beta}{dt^2} \\ \frac{d\theta_\gamma}{dt^2} \end{bmatrix} = \begin{bmatrix} \gamma_t^x \\ \gamma_t^y \\ \gamma_t^z \\ \omega_t^x \\ \omega_t^y \\ \omega_t^z \end{bmatrix} \tag{1.1}$$

Comme la stratégie de placement des capteurs inertiels varie en fonction de l'application visée, une grande partie de la littérature utilisant cette modalité le fait en proposant leurs propres données. Ainsi, il est plus difficile de comparer les méthodes qui la composent. Néanmoins, certains travaux récents, adressant la reconnaissance d'actions sur l'angle des données inertielles (Imran & Raman, 2020; Wei, Jafari & Kehtarnavaz, 2019) l'ont fait en s'appuyant sur la base de données UTD-MHAD (Chen, Jafari & Kehtarnavaz, 2015), qui propose un peu plus de 500 exemples répartis en 27 actions et 8 sujets. Comme nous le verrons au chapitre 3, c'est sur cette dernière que les performances de la méthode proposée seront évaluées. De plus, ce chapitre proposera aussi une revue de la littérature relative aux méthodes utilisant les signaux de nature inertielle.

CHAPITRE 2

DÉMARCHE DE TRAVAIL ET MÉTHODOLOGIE

Dans ce chapitre, les fondements méthodologiques et techniques liés à l'apprentissage machine dans le cadre d'un problème de classification seront d'abord présentés. Par la suite, nous survolerons les différents hyper-paramètres liés au processus d'apprentissage avant d'aborder les différentes procédures et outils d'analyse mis en place afin d'optimiser la méthode. Finalement, les raisonnements clés ayant guidé le processus de recherche vers la solution proposée seront présentés.

2.1 Fondements méthodologiques de l'apprentissage machine

L'objectif poursuivi par les méthodes d'apprentissage machine dans le cadre d'une tâche de classification consiste à apprendre une fonction discriminante capable d'associer un signal donné, $\mathbf{X}$, à une étiquette, $\mathbf{y}$.

Typiquement, les exemples recueillis au sein d'une base de données sont répartis en trois ensembles mutuellement exclusifs :

1. Ensemble d'entraînement,
2. Ensemble de validation,
3. Ensemble de test.

D'une part, l'ensemble d'entraînement sert à apprendre les différents paramètres, θ alors que l'ensemble de validation sert plutôt à l'optimisation des hyper-paramètres, Θ, que l'on définit comme étant les paramètres qui ne sont pas modifiés par le processus d'apprentissage. Finalement, l'ensemble de test permet d'établir dans quelle mesure un réseau arrive à inférer sur des exemples n'ayant pas servi à l'optimisation des paramètres ni des hyper-paramètres.

2.1.1 Fondement d'apprentissage des paramètres des réseaux classifieurs

En étudiant les différentes contraintes régissant le flux des données, il est possible d'exprimer les réseaux de neurones utilisés dans le cadre de ce mémoire comme une fonction vectorielle $\vec{F}_{\{\theta\cup\Theta\}} : \mathbb{R}^{dim(\mathbf{X})} \rightarrow \left]0,1\right[^N$ dont le co-domaine représente un espace de probabilités dont le nombre de dimensions correspond au nombre d'actions différentes, soit N. Pour ce faire, $\vec{F}$ construit d'abord un espace probabilisable, $\mathcal{A} = \{f^c_{\{\theta\cup\Theta\}}\}^N_{c=1} \mid f^c_{\{\theta\cup\Theta\}} : \mathbb{R}^{dim(\mathbf{X})} \rightarrow \mathbb{R}$, dont la topologie dépend de Θ et où les variables sont θ. En lui appliquant $\mathbf{X}$, $\vec{F}$ définit ensuite un point $\vec{o} \in \mathcal{A}$, que la fonction $Softmax \subset \mathrm{C}^\infty : \mathcal{A} \rightarrow \left]0,1\right[^N$, transpose dans l'espace de probabilités en appliquant l'équation 2.1 à chacune de ses dimensions, $c = \{1, ..., N\}$.

$$\hat{y}_c = \frac{e^{o_c}}{\sum_{j=1}^N e^{o_j}} = \frac{e^{f_c(\mathbf{X}|\theta\cup\Theta)}}{\sum_{j=1}^N e^{f_j(\mathbf{X}|\theta\cup\Theta)}} \implies \hat{y}_c > 0, \quad \text{et} \quad \sum_{c=1}^N \hat{y}_c = 1 \tag{2.1}$$

En comparant le vecteur de probabilités résultant du procédé décrit ci-dessus, $\hat{y}$, avec celui de l'étiquette, $\mathbf{y}$, à l'aide de la méthode de l'entropie-croisée, C-E$(\hat{y}, \mathbf{y})$, donnée par l'équation 2.1.1, il est possible d'établir une mesure (fonction de coût) différentiable permettant d'inscrire la recherche des paramètres dans le cadre d'un problème d'optimisation.

$$\text{C-E}(\hat{y}, \mathbf{y}) = -\sum_{c=1}^N y_c \cdot \log(\hat{y}_c) \quad \Big| \quad y_c \subset \mathbf{y} \in \{0,1\}^N$$

(2.2)

Ainsi, si le gradient calculé par rapport à cette mesure en fonction des paramètres du réseau est non nul, en modifiant la valeur de ceux-ci dans la direction opposée, la probabilité inférée sur la classe définie par l'étiquette est bonifiée comme l'explique la relation 2.3.

$$\exists\, \epsilon > 0 \;\; t.q. \;\; \theta' \leftarrow \theta - \epsilon \cdot \nabla_\theta[\text{C-E}(\hat{y}_\theta, \mathbf{y})] \Rightarrow \text{C-E}(\hat{y}_{\theta'}, \mathbf{y}) < \text{C-E}(\hat{y}_\theta, \mathbf{y}) \equiv \hat{y}^{\theta}_{Etiq.} < \hat{y}^{\theta'}_{Etiq.} \quad (2.3)$$

Comme la topologie du réseau n'est calculée que localement, il est impossible de déterminer la valeur optimale de ϵ, c'est pourquoi l'apprentissage des paramètres procède plutôt à l'aide d'une méthode de descente du gradient pour laquelle le pas, η, communément connu sous le nom de taux d'apprentissage, est préétabli.

Procédant de manière itérative, les méthodes de descente du gradient dépendent donc d'un choix initial de paramètres. Par conséquent, différentes approches pour lesquelles un processus pseudo-aléatoire génère un ensemble de paramètres en respectant différentes distributions ont été proposées et intégrées aux librairies d'apprentissage pouvant parfois ce traduire par un gain en performances appréciable.

2.2 Hyper-paramètres liés au processus d'apprentissage

Étant donné que les hyper-paramètres affectant la topologie du réseau seront vus en détail lors de la description de la méthode au chapitre 3, ceux que nous survolerons au cours de cette section réfèrent au processus d'apprentissage.

2.2.1 Taux d'apprentissage

Tel que discuté à la section précédent, le taux d'apprentissage η définit le pas multipliant le gradient calculé par rapport à la fonction de coût afin de mettre à jour les paramètres. Typiquement un taux d'apprentissage trop grand peut faire en sorte que les performances calculées par rapport à la fonction de coût divergent ou oscillent. Inversement, un taux d'apprentissage trop faible peut faire en sorte que l'entraînement soit plus long.

2.2.2 Nombre d'époques

En apprentissage machine, on dit qu'une époque a été complétée lorsque l'entièreté des exemples de l'ensemble d'entraînement ont été utilisés afin de modifier les paramètres du réseau. En pratique, savoir à quel moment arrêter le processus d'apprentissage peut avoir un impact important sur les performances de l'algorithme. En effet, arrêter le processus trop tôt mène à une solution sous-optimale, tout comme l'arrêter trop tard et ainsi tomber dans une situation de sur-apprentissage [1].

De manière à établir le nombre d'époques optimal, les scientifiques des données font appel à l'ensemble de validation. Pour ce faire, il suffit d'identifier l'époque à partir de laquelle la valeur de la fonction de coût commence à remonter lorsqu'évaluée sur celui-ci.

2.2.3 Méthode de descente du gradient

La méthode de descente du gradient permet d'apprendre les paramètres de manière à optimiser la fonction objectif évaluée sur le sous-ensemble d'entraînement. Dans le cadre de mes recherches, deux méthodes ont été testées, soient la méthode d'optimisation connue sous l'acronyme ADAM (pour *Adaptative Moment Estimation*) ainsi que l'algorithme de la descente du gradient stochastique.

D'une part, la méthode ADAM permet généralement de trouver des paramètres minimisant grandement la fonction de coût en un nombre restreint d'époques malgré un taux d'apprentissage généralement initialisé à une valeur plus faible que pour méthode de la descente de gradient stochastique. La raison expliquant cette convergence accélérée est que le pas pris afin d'optimiser la fonction de coût s'accentue à chaque itération si le gradient reste de même signe. Or, dans les faits, la méthode de la descente du gradient stochastique a permis d'obtenir de meilleurs résultats sur l'ensemble de test, démontrant ainsi une meilleure capacité de généralisation.

[1] Situation où les gains en performances sur l'ensemble d'entraînement ne se répercutent pas sur les ensembles de test ou de validation.

2.2.4 Taille des mini-lots

Pendant le processus d'apprentissage, il est possible d'accumuler le gradient sur plusieurs exemples avant de procéder à la mise à jour des paramètres. Ainsi, on désigne par mini-lot l'ensemble des exemples sur lequel le gradient est accumulé avant de mettre à jour les paramètres.

La taille des mini-lots peut jouer un rôle crucial sur le processus d'apprentissage. D'un point de vue théorique, plus la taille des mini-lots est grande, plus les performances atteintes sur l'ensemble d'entraînement auront tendance à être généralisables. Inversement, plus la taille des mini-lots est petite, plus le risque de tomber en situation de sur-apprentissage est grand. D'un point de vue pratique, il existe aussi une limite à la taille maximale des mini-lots qui est imposée par les ressources mémoires de la machine sur laquelle l'algorithme est entraîné.

2.3 Optimisation des hyper-paramètres

Afin d'optimiser le choix d'un hyper-paramètre, il faut faire varier sa valeur lors de différents entraînements en gardant toutes choses égales par ailleurs. Malheureusement, bien souvent sa valeur optimisée en fonction d'un ensemble d'hyper-paramètres ne l'est plus lorsqu'un de ceux-ci est à son tour modifié. Comme il n'existe pas de procédé clair permettant de à trouver la solution optimale, la présente section proposera, dans un premier temps, un aperçu des procédures mises en place de manière à maximiser le nombre d'ensembles d'hyper-paramètres testés tout en réduisant l'effort associé aux différentes hypothèses, puis proposera ensuite un survol des outils d'analyse des performances.

2.3.1 Procédures

Concrètement, l'optimisation des procédures regroupe deux aspects clés. Le premier, de nature méthodologique, consiste à construire une architecture logicielle permettant de tester différentes options à l'aide d'un fichier de commande. Le second, quant à lui, consiste à optimiser les ressources matérielles de manière à pouvoir tester plusieurs hypothèses en parallèle. Ainsi la présente sous-section présentera un survol des méthodes mises en place.

2.3.1.1 Fichier de commande

L'importance méthodologique d'un fichier de commande est capitale. En effet, grâce à lui, les différents hyper-paramètres de la méthode peuvent être testés avec un minimum d'effort et surtout sans avoir à aller modifier directement le reste du code, réduisant ainsi le risque d'erreurs au fil des entraînements. Comme mes efforts de recherche ont porté sur deux types de données, un fichier de commande distinct a été proposé pour chacune des options. À titre d'exemple il est possible de les consulter. (Voir ANNEXE I, p.45-46 figure I-1 et figure I-2)

2.3.1.2 Utilisation des ressources nuagiques

L'accès aux ressources offertes par *Calcul Canada* a aussi joué un rôle important dans le succès de mes recherches en me permettant de tester plusieurs options en parallèle. Pour exécuter des entraînements sur les serveurs de *Calcul Canada*, il faut faire une demande d'allocation de ressources sous forme d'un script (Voir ANNEXE II, p.47 figure II-1) spécifiant les ressources voulues ainsi que le temps pendant lequel on désire en faire usage. Le script spécifie aussi le code à exécuter ainsi que le registre de sortie où les informations produites en cours d'entraînement sont sauvegardées. Finalement, étant donné que l'allocation des ressources se fait selon un algorithme de gestion assurant un partage équitable entre les chercheurs, il est souvent nécessaire de patienter un certain temps avant que débute l'exécution d'une tâche. Ainsi, afin d'avoir une certaine rétroaction sur le moment où elle débute et/ou se termine, il est possible de mettre en place une alerte sous forme de courriel.

2.3.2 Analyse et performances

Afin d'adopter un cadre d'analyse rigoureux, je me suis appuyé sur la plate-forme *Tensorboard* qui permet de comparer côte à côte et en temps-réel différents indicateurs de performances que nous aborderons plus en détail dans la présente sous-section. Pour ce faire, il suffit de sélectionner différents entraînements à partir d'un menu défilant tel qu'illustré à la figure 2.1.

Figure 2.1 Liste permettant de sélectionner les apprentissages dont on souhaite visualiser les métriques d'apprentissage

2.3.2.1 Courbes d'apprentissage

Les courbes d'apprentissage illustrent les performances face à une métrique en fonction l'évolution de l'entraînement. Comme nous l'avons vue, la valeur moyenne de la fonction de coût sur une époque, telle que présentée sur l'ensemble de validation à la figure **??**, en est un bon exemple. Dans le cadre d'un problème de classification, il n'est pas rare non plus de s'intéresser à l'évolution du taux de succès de classification comme en témoigne la figure 3.6 de l'article rattaché à ce mémoire.

2.3.2.2 Matrice de confusion

La matrice de confusion, telle qu'illustrée à la figure 3.7, est un outil permettant de visualiser les prédictions faites sur différents exemples en fonction de leurs étiquettes. Ainsi chaque exemple vient incrémenter la case de la matrice correspondant à la classe prédite, en abscisse, et la classe de son étiquette en ordonnée. Grâce à cet outil, il est possible d'identifier facilement les classes pour lesquelles l'algorithme est susceptible d'émettre une prédiction exacte ou non. Elle nous informe aussi sur la nature des erreurs commises, permettant ainsi d'identifier des sous-ensembles de classes sur lesquelles l'algorithme a plus de difficultés.

2.3.2.3 Histogramme

Comme nous le verrons au prochain chapitre, la méthode proposée dans le cadre de l'article scientifique rattaché à ce mémoire est construite autour de plusieurs classificateurs binaires, soit un pour chaque action. Afin d'avoir une idée de la performance individuelle de chacun des classificateurs, j'ai eu recours à un histogramme tel qu'illustré à la figure 3.9. Dans les faits, celui-ci illustre les valeurs de précision [2] et de rappel [3] afin de donner une idée des performances individuelles associées à chaque classificateur.

2.3.3 Étude des signaux

Dans le domaine de la reconnaissance d'actions, les exemples étudiés ont la forme de séries temporelles. Ainsi, qu'ils soient issus de flux visuels et/ou inertiels le format des données permettant de décrire une action a la forme d'une suite de valeurs numériques représentant l'évolution de signaux au cours du temps. Comme l'exprime la relation 2.4, une série temporelle, $\mathbf{X}$, se compose d'un nombre fini d'échantillons, T, représentés sous forme de tableaux dont la structure, $\mathcal{S}$, et le nombre de dimension, d, varient en fonction de la modalité des signaux utilisés.

---

[2] $\frac{\text{vrais positifs}}{\text{vrais positifs + faux positifs}}$

[3] $\frac{\text{vrais positifs}}{\text{vrais positifs + faux négatifs}}$

$$\mathbf{X} \in \mathbb{R}^{\mathcal{S} \times T} \quad | \quad \mathcal{S}_d \in \mathbb{N}^d \tag{2.4}$$

Tel que représenté par la relation 2.5, lorsque celle-ci est de nature inertielle, chaque pas de la série temporelle, **X**, se compose alors d'un tableau unidimensionnel, pour lequel un indice particulier est associé à chaque degré de liberté du capteur inertiel.

$$\mathbf{X}_{inertiel} \Rightarrow \mathcal{S}_1 = [DL_{\{\gamma_{\{x,y,z\}},\omega_{\{x,y,z\}}\}}] \tag{2.5}$$

Pour les données de type exosquelettique, le tableau en est un en deux dimensions et nécessite donc deux indices, soit un pour définir un joint particulier et un autre pour définir l'axe spatial de ses coordonnées.

$$\mathbf{X}_{exosquelette} \Rightarrow \mathcal{S}_2 = [J_{oints} \times C_{\{x,y,z\}}] \tag{2.6}$$

Finalement, lorsque la modalité utilisée consiste en un flux vidéo, ce sont alors trois indices qui sont nécessaires afin de parcourir le tableau relatif à chaque échantillon temporel. Il y a donc un axe pour définir la position vertical d'un pixel, H, un second pour sa position horizontal, L, et finalement un dernier pour l'encodage de la couleur, RGB.

$$\mathbf{X}_{video} \Rightarrow \mathcal{S}_3 = [H \times L \times RGB] \tag{2.7}$$

2.3.4 Réseaux convolutifs

En utilisant une méthode d'apprentissage paramétrique comme celle décrite à la section 2.1.1, les réseaux à convolutions ont récemment démontré qu'ils sont de puissants outils dans le cadre d'un problème de classification. Plus précisément, cette architecture neuronale est en mesure

d'extraire des traits discriminants des structures de données, lesquels permettent ensuite de distinguer les signaux issus de différentes catégories. Pour ce faire, les réseaux convolutifs procèdent en appliquant successivement des couches de convolution et de mise en commun à la structure des signaux composant les différents exemples.

2.3.4.1 Couche de convolution

Une couche de convolution consiste à appliquer une banque de filtres à une certaine structure de données. Comme l'exprime la relation 2.8, une banque de filtres est composée d'un certain nombre de filtres, f, couvrant chacun un sous ensemble de valeurs du tableau et de la plage temporelle, m, formant la structure de données des exemples.

$$W \in \mathbb{R}^{\bar{\mathcal{S}} \times m \times f} \cap \theta \quad | \quad \bar{\mathcal{S}} \in \Theta \Rightarrow \mathbb{R}^{\bar{\mathcal{S}}} \subseteq \mathbb{R}^{\mathcal{S}} \quad \text{et} \quad m \leq T \tag{2.8}$$

De manière générale, l'opérateur de convolution, $*$, génèrent un résultat sous forme de combinaisons linéaires. Par exemple, l'équation 2.9, explicite les opérations effectuées lors de l'application d'un filtre sur une structure en deux dimensions (par exemple temps et signaux inertiels) lors d'une marche sur les données et introduit aussi l'indice l qui spécifie une couche en particulier à laquelle les autres éléments sont liés.

$$\mathbf{Z}^{l+1}_{s,t} = \begin{bmatrix} \mathbf{w}^{l+1}_{1,1} & \mathbf{w}^{l+1}_{1,2} \\ \mathbf{w}^{l+1}_{2,1} & \mathbf{w}^{l+1}_{2,2} \end{bmatrix} * \begin{bmatrix} & \vdots & \vdots & \\ \cdots & \mathbf{x}^{l}_{s,t} & \mathbf{x}^{l}_{s,t+1} & \cdots \\ \cdots & \mathbf{x}^{l}_{s+1,t} & \mathbf{x}^{l}_{s+1,t+1} & \cdots \\ & \vdots & \vdots & \end{bmatrix} = \mathbf{w}^{l+1}_{1,1} \cdot \mathbf{x}^{l}_{s,t} + \mathbf{w}^{l+1}_{1,2} \cdot \mathbf{x}^{l}_{s,t+1} + \mathbf{w}^{l+1}_{2,1} \cdot \mathbf{x}^{l}_{s+1,t} + \mathbf{w}^{l+1}_{2,2} \cdot \mathbf{x}^{l}_{s+1,t+1} \tag{2.9}$$

Au résultat issu des combinaisons linaires générées par un filtre, il est aussi possible d'appliquer une fonction d'activation, ϕ. Bien que la raison pour laquelle ces fonctions apportent parfois un gain en performances soit obscure, les résultats empiriques démontrent, quant à eux, leur intérêt.

En effet, comme l'illustre la figure 2.2, qui réfère à un test effectué dans le cadre de l'optimisation de la méthode qui sera présentée au prochain chapitre, la fonction d'activation SeLU, démontre clairement sa supériorité, permettant ainsi à l'algorithme d'effectuer des prédictions exactes dans une plus grande proportion que sans (Linear) ou qu'avec d'autres fonctions d'activation.

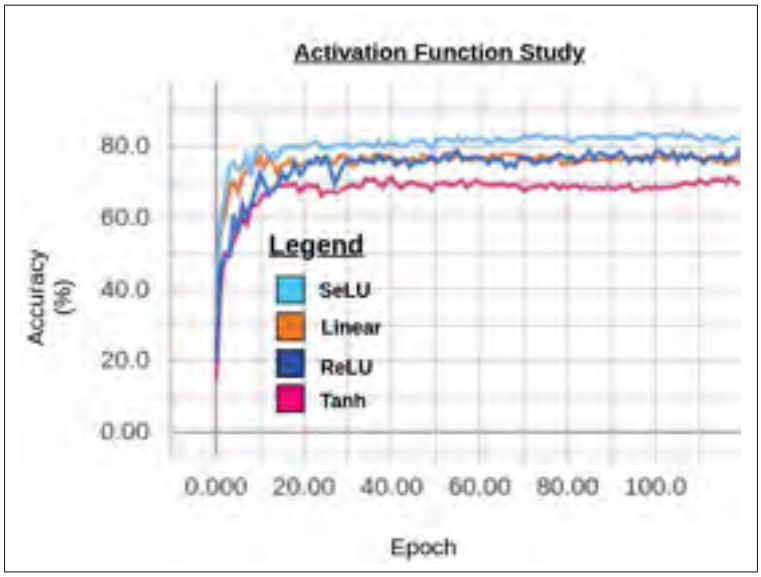

Figure 2.2 Avantage de la fonction d'activation SeLU

2.3.4.2 Couche de mise en commun (*Pooling*)

Suite à l'application d'une couche de convolution, que l'on peut résumer à l'aide de l'équation 2.10, il est habituel d'appliquer une couche de mise en commun (*Pooling*).

$$\phi(\mathbf{W}^{l+1} * \mathbf{X}^{l}) \Rightarrow \mathbf{Z}^{l+1} \tag{2.10}$$

Grâce à cette méthode, il est possible de réduire la complexité de la structure de données, $\mathbf{Z}^{l+1}$, encodée par les couches de convolution. Pour ce faire, un critère de mise en commun tel que la sélection de la valeur maximale, $MaxPool$, ainsi qu'une certaine plage de mise en commun, $\mathbf{P}$, sont définis. Tel que décrit par l'équation 2.11, de ce mécanisme résulte alors une nouvelle

structure, $\mathbf{X}^{l+1}$, sur laquelle il est possible d'appliquer une nouvelle couche de traitement sur les données.

$$MaxPool(\mathbf{Z}^{l+1}, \mathbf{P}) \Rightarrow \mathbf{X}^{l+1} \in \mathbb{R}^{\mathbf{S}_{l+1} \times T_{l+1}} \quad | \quad \mathbf{P} \in \Theta \quad \text{et} \quad T_{l+1} \leq T_l \tag{2.11}$$

2.3.5 Critique

Comme il l'a été mentionné au chapitre 1, dans le cadre de la revue de la littérature, une approche populaire consiste à créer une image à partir des signaux acquis durant l'exécution d'une action pour ensuite traiter la reconnaissance d'actions comme un problème de reconnaissance d'images. Or, comme le souligne Li *et al.* (2018), en raison de la petite taille des filtres utilisés par les architectures éprouvées dans le domaine de la reconnaissance d'images (généralement 2X2 ou 3x3), celles-ci ne sont pas en mesure de capter les relations unissant des éléments qui seraient éloignés au sein de la structure des signaux. Par exemple, un filtre de taille 2x2 qui serait appliqué à la structure de données, $\mathbf{X}^{l=0}_{inertiel} \in \mathbb{R}^{\mathcal{S} \times T_{l=0}}$ ne pourrait encoder que des relations unissant des signaux voisins au sein de la structure. Or, l'ordre des indices référant aux différents signaux de la structure ne reflète pas une information intrinsèque quant à celle-ci, chaque signal ou combinaison de signaux étant a priori aussi important que les autres.

2.3.6 1D-CNN

À la lumière de cette critique, la solution proposée au prochain chapitre fait donc appel aux réseaux convolutifs en une dimension (1D-CNN). Contrairement à celles utilisées dans les problèmes de reconnaissance d'images, cette architecture consiste à appliquer des filtres de convolution couvrant l'entièreté des éléments des tableaux relatifs à chaque échantillon temporel. Comme le démontre l'exemple décrit par l'équation 2.12, lorsque la modalité d'acquisition est de type inertielle, un filtre de cette nature (et ayant une composante temporelle, $m_{l=1} = 3$), $W^k \subset \mathbf{W} \in \mathbb{R}^{6 \times 3 \times f}$, fait donc intervenir les signaux des six degrés de liberté du capteur inertiel au sein des combinaisons linaires dont sont issus les éléments de la couche suivante. En utilisant

cette approche, on remarque aussi que dorénavant, la marche sur les données ne se fait plus que le long l'axe temporel et que la nature chronologique du signal résultant est préservée.

$$z_t^{l=1,k} = W^{l=1,k} * \mathbf{X}^0_{[t,t+m-1]} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \\ w_{4,1} & w_{4,2} & w_{4,3} \\ w_{5,1} & w_{5,2} & w_{5,3} \\ w_{6,1} & w_{6,2} & w_{6,3} \end{bmatrix}^{1,k} * \begin{bmatrix} \gamma_t^x & \gamma_{t+1}^x & \gamma_{t+2}^x \\ \gamma_t^y & \gamma_{t+1}^y & \gamma_{t+2}^y \\ \gamma_t^z & \gamma_{t+1}^z & \gamma_{t+2}^z \\ \omega_t^x & \omega_{t+1}^x & \omega_{t+2}^x \\ \omega_t^y & \omega_{t+1}^y & \omega_{t+2}^y \\ \omega_t^z & \omega_{t+1}^z & \omega_{t+2}^z \end{bmatrix} \tag{2.12}$$

Afin que la marche sur les données puisse continuer à être faite uniquement selon l'axe temporel, dans les couches profondes du réseau, les filtres d'une certaine couche doivent couvrir l'ensemble des valeurs issues de la couche précédente tel qu'exprimé par l'équation 2.13.

$$\mathbf{z}_t^l = \left\{ \begin{bmatrix} w_{1,1} & \dots & w_{1,m_l} \\ \vdots & \cdot & \vdots \\ w_{f_{l-1},1} & \dots & w_{f_{l-1},m_l} \end{bmatrix}^{l,k} * \begin{bmatrix} x_{1,t} & \dots & x_{1,t+m_l-1} \\ \vdots & \ddots & \vdots \\ x_{f_{l-1},t} & \dots & x_{f_{l-1},t+m_l-1} \end{bmatrix}^{l-1} \right\}_{k=1}^{f_l} \tag{2.13}$$

2.3.7 Échantillonage hiérarchique

La principale contribution de l'article présenté au prochain chapitre réside dans la manière qu'est construit le vecteur de représentatif utilisé par la partie classificateur du réseau. En effet, dans la littérature, ce dernier est issu de l'isomorphisme, ζ, décrit par l'équation 2.14, que l'on applique à la structure encodée par la dernière couche de convolution. Ainsi la composante temporelle des signaux d'origine y est uniformément représentée.

$$\zeta : [D_1 \times D_2 \times ...] \rightarrow [D_1 \cdot D_2 \cdot ...] \quad | \quad D_i \in \mathcal{S}_{\mathbf{X}^L} \cup T_L \tag{2.14}$$

Or, dans le cas de l'approche proposée, telle qu'illustrée par la figure 2.3, c'est plutôt un échantillonnage de la valeur maximale et minimale issus de l'application de chaque filtre au travers des différentes couches du réseau qui compose le vecteur représentatif. Ainsi, celui-ci est construit de manière hiérarchique étant donné que ses éléments ne sont pas intrinsèquement liés à la structure temporelle des signaux initiaux.

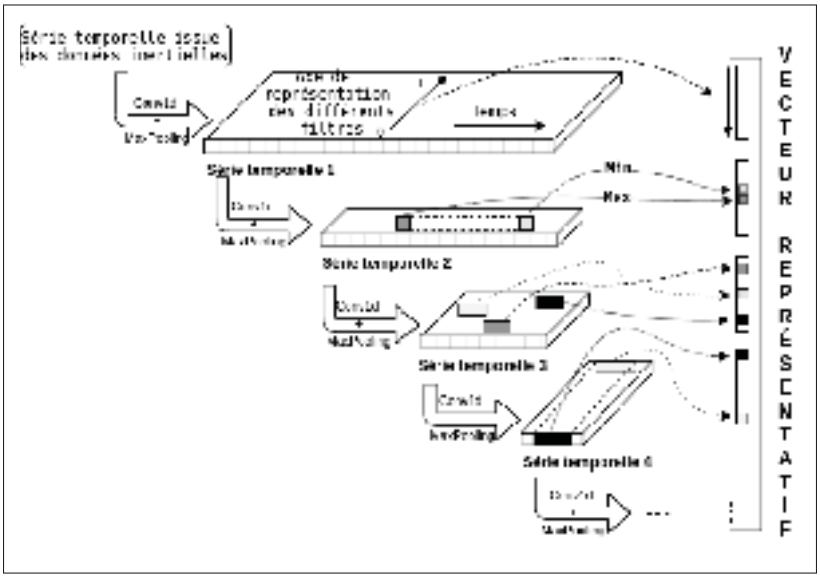

Figure 2.3 Illustration de la méthode d'échantillonage hiérarchique

Comme nous le verrons au prochain chapitre, en utilisant cette approche, la méthode proposée démontre une plus grande robustesse quant à l'ajout d'information superflue et/ou redondante, tout en rivalisant l'état de l'art actuel en termes de succès de classification.

CHAPITRE 3

A LEARNING BASED HIERARCHICAL APPROACH FOR HUMAN ACTION RECOGNITION

Nicolas Lemieux[1] , Rita Noumeir[2]

[1] Département de Génie électrique, École de Technologie Supérieure,
[2] Département de Génie électrique, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article publié par la revue « MDPI Sensors » en septembre 2020.

3.1 Abstract

Recognizing actions based on sensor signals is a challenging yet fascinating problem due the vast quantity of potential practical applications it could enable. As a matter of fact, previous deep-learning based literature addressed it by learning a latent representation of the whole action sequence in order to classify it. In this paper we propose to harvest specific information in a hierarchical way that consist in sampling the maximum and minimum values of all the feature maps throughout the different layers of a 1D-CNN architecture in order to learn an efficient representation. We validated the proposed method on the UTD-MHAD dataset using the inertial sensors signals and achieved very competitive results, but also observed some interesting proprieties. For instance, the method proved itself to be robust against zero-padding. It was also observed that training and inference could be done on inputs signals extended up to different length (zero-padded) without affecting the classification performances. Finally, we also demonstrate that each element of the latent representation can be localized in time, which provides some insight on what the algorithm learns and also opens the door to new applications.

3.2 Introduction

As human beings, part of our happiness or suffering depends on our interactions with our direct environment, which is governed by an implicit or explicit set of rules (Nishida, 2007).

Meanwhile, a growing proportion of these interactions are now targeted towards connected devices such as smart phones, smart watches, intelligent home assistants and other IoT objects since their number and range of applications keep growing every year (Vermesan, Friess et al., 2014). As a consequence of this phenomenon, efforts are made in order to improve the users' experience through the elaboration of some ever-more intuitive and comfortable human-machine interfaces (Rautaray & Agrawal, 2015). With the relatively recent successes of artificial intelligence based algorithms and the emergence of a wide range of new sensors, human action recognition (HAR) has become one of the most popular problems in the field of pattern recognition currently and plays a key role in the development of such interfaces. In that context, RGB, depth, skeleton and infrared stream based approaches attracted most of the scientific community's attention as their non-invasive nature is well suited for applications such as surveillance and led to the creation of commonly accepted benchmarks such as NTU-RGB-D (Shahroudy *et al.*), which was recently enhanced as NTU-RGB-D 120 (Liu, Shahroudy, Perez, Wang, Duan & Kot Chichung, 2019). On the other hand, action recognition based on visual streams can only be used within limited setups, is prone to issues such as occlusion and often needs substantial computational resources.

Admittedly less popular, HAR based on inertial sensor signals is still a relevant subject of research. Despite the fact that with this approach, the users are required to wear some kind of device, thanks to the miniaturization of microelectromechanical systems (MEMS), inertial measurement units (IMU) are now seamlessly integrated with different day-to-day objects such as remote controls or smart-phones and wearable technologies such as smart watches or smart cloths. Furthermore, for applications such as sports analysis, fall detection or remote physiotherapy, it provides an interesting alternative, as it can be used anywhere, provides more privacy and is considerably less expensive.

Although many different datasets exist in that branch of HAR, none have really established themselves as a clear benchmark, as placement strategies of the IMUs, or the set of activities are mostly targeted towards specific applications. In this work, we opted for the University of Dallas, Texas, multimodal human action dataset (UTD-MHAD) in order to validate our approach, as we

believed that it offered the most variability in terms of activities (27) and actors (eight). Not only did the proposed method achieve competitive results on the classification task using IMU signals, it was also designed in order to address two of the biggest challenges currently faced by HAR algorithms (Wang, Chen, Hao, Peng & Hu, 2019). For that purpose, we propose a hierarchical representation of the different actions by sampling the maximum and minimum activation of each convolution kernel throughout the network, which could potentially help with (1) the recognition of high-level activities characterized by the large time variations in the sub-movements composing them. Moreover, as the proposed solution is built around a 1D-CNN architecture, it was also observed that it necessitated dramatically less operations or memory than the most popular architectures, thus greatly contributing to (2) the portability to limited resource electronics for HAR algorithms. Finally, in this paper, we also investigated the tractability of the features throughout the proposed framework, both in time and duration, as we believe it could play an important role in future works in order to make the solution more intelligible, hardware-friendly, robust and accurate. For the rest of the present article, the following structure will observed :

- Section 3.3 quickly reviews the literature.
- Section 3.4 provides a detailed explanation of the method.
- Section 3.5 details the temporal analysis, showing how feature elements can be localized in time and that they report the dynamics of different durations.
- Section 3.6 provides the results in order to validate both the performances and properties of the proposed method.
- Section 3.7 summarizes the important results and their implications and provides ideas for future work.

3.3 Related Works

3.3.1 Neural Network Based HAR

Originally based on conventional machine learning methods such as support vector machines (SVM), decision trees or k-nearest neighbour (k-NN), early methods for HAR relied on various time and/or frequency domain features extracted from the raw data. However, as for many other fields, inertial sensor based HAR benefited from gradient descent based methods. Besides improving the classification capabilities, these methods also alleviated the requirement of feature engineering as they learn the features and classifier simultaneously from the annotated data. As a matter of fact, Kwapisz, Weiss & Moore (2011) observed that multi-layer-perceptrons (MLPs) were superior to decision trees and logistic regression for HAR. Similarly, Weiss & Lockhart (2012) observed that MLP outperformed decision trees, k-NN, naive Bayes and logistic regression when the training and evaluation was done on data gathered from the same user and also noted that these user-specific models performed dramatically better than impersonal models where training and evaluation users are different. Since then, models based on convolution neural networks (CNN) have been proposed for inertial sensor based HAR (Imran & Raman, 2020; Zeng, Nguyen, Yu, Mengshoel, Zhu, Wu & Zhang, 2014; Yang, Nguyen, San, Li & Krishnaswamy, 2015; Ronao & Cho, 2016) and have also been proven superior to k-NN and SVM (Yang *et al.*, 2015). With these methods, the input signals usually go through multiple convolution and pooling layers before being fed to a classifier. Although 1D-CNN architectures are well suited for the time series type of data such as IMU signals, CNNs are more famous in their 2D form, considering the tremendous amount of success they have had in the field of image recognition. Consequently, Jiang & Yin (2015) proposed to construct an activity image, based on IMU signals, in order to proceed to HAR as an image recognition problem. Back to 1D-CNN, Ronao & Cho (2016) showed that wider kernels and a low pooling size resulted in better performances. More recently, Murad & Pyun (2017) implemented a recurrent neural network (RNN) and demonstrated the prevalence of their method over SVM and k-NN. Moreover, employing the long short-term memory (LSTM) variation of RNN along with CNNs, Hammerla, Halloran & Ploetz (2016) and Ordóñez & Roggen (2016) achieved better results than with some strictly CNN based

methods. However, the CNN architectures they used in their comparative study had respectively a maximum of three and four convolution blocks. Meanwhile, Imran & Raman (2020) conducted an analysis on the topology of the 1D-CNN, from which they concluded that five consecutive blocks of convolution and pooling layers were better than four. Furthermore, with their architecture and a data augmentation strategy, they also achieved state-of-the-art results on the UTD-MHAD dataset for both sensor based and multi-modal HAR. Yet, leaving data augmentation aside, the proposed method in this article slightly outperformed theirs for the sensor based HAR.

3.3.2 Temporal Normalization

As the popular machine learning frameworks require the inputs to be of a constant shape during the training process and as different actions usually have different durations, temporal normalization is mandatory. On that account, different strategies have been proposed. One way to do it is by randomly sampling a fixed number of frames from a longer sequence while keeping the chronological ordering (Imran & Raman, 2020; Hammerla *et al.*, 2016). It can also serve as a data augmentation technique, since the sampled frames can vary from one epoch to another. Another way to do it is to sample a fixed amount of consecutive frames from the action sequence (Weiss & Lockhart, 2012). In this case, the algorithms often have to learn from incomplete representations. Finally, a more conservative way to achieve temporal normalization is to extend, up to a fixed point, the signals with zeros. This technique is also known as zero-padding. It has the benefit of enabling the algorithms to learn features on complete and undistorted sequences. Therefore, it will be the preferred strategy in this work. Moreover, contrary to Imran & Raman (2020), who reported that the performances of their CNN based method decayed with the augmentation of the amount of zero-padding, our method proved itself to be robust against it.

3.4 Proposed Method

Like the majority of HAR algorithms, the main objective of our method is to provide a way to successfully classify specific actions based on a multivariate time series input. In the sensor

based variation of HAR addressed here, the input time series, $\mathbf{X}^0$, consist of a fixed number of time steps, T_0, each reporting a dynamic occurring at a certain moment t, by providing, in a vector form, $\mathbf{x}_t$, the speed and angular rate variations, γ and ω, with respect to a 3-dimensional (x,y,z) orthogonal coordinate system whose origin corresponds to the IMU sensor. This can be expressed mathematically as :

$$\mathbf{X}^0 = [\mathbf{x}_1^0, \dots \mathbf{x}_t^0, \dots \mathbf{x}_{T_0}^0], \quad \text{where} \quad \mathbf{x}_t^0 = \begin{bmatrix} \gamma_t^x \\ \gamma_t^y \\ \gamma_t^z \\ \omega_t^x \\ \omega_t^y \\ \omega_t^z \end{bmatrix} \tag{3.1}$$

In order to train our algorithm, the original time series, $\mathbf{X}^0$, is conveyed to a 1D-CNN constituted of multiple convolution blocks in cascade (i.e., the output of a specific block serves as the input for the next), each applying a certain set of convolution filters, batch normalisation and max pooling. From this process, L other time series are created, $\{\mathbf{X}^l \in \mathbb{R}^{F_l \times T_l}\}_{l=1}^{L}$. Here, L is the number of convolution blocks ; F_l is the number of filters in the lth convolution block's filter bank, $\mathbf{W}^l \in \mathbb{R}^{F_l \times m_l \times F_{l-1}}$ (whose filters are of size m_l along the temporal axis) ; T_l is the number of corresponding time steps of the lth generated time series, $\mathbf{X}^l$. Hence, the lth time series is described as :

$$\mathbf{X}^l = [\mathbf{x}_1^l, \dots \mathbf{x}_t^l, \dots \mathbf{x}_{T_l}^l], \quad \text{where} \quad \mathbf{x}_t^l = [x_t^{l,1}, \dots x_t^{l,k}, \dots x_t^{l,F_l}]^\intercal \tag{3.2}$$

In these blocks, convolutions are carried without bias ; thus, they can be described by the following equation :

$$\mathbf{z}_t^l = \phi\left(\mathbf{W}^l * \mathbf{X}_{[t,t+m_l-1]}^{l-1}\right) \tag{3.3}$$

where $\mathbf{z}_t^l$ is the vector resulting from the application of all the convolution filters of $\mathbf{W}^l$ to the portion of the previous time series whose vectors are between the tth and $(t + m_l - 1)$th time steps inclusively, $\mathbf{X}^{l-1}_{[t,t+m_l-1]}$; ϕ is the SeLU activation function (Klambauer, Unterthiner, Mayr & Hochreiter, 2017), which demonstrated better experimental results ; and $*$ is the convolution operator, which can be equivalently expressed as Equation (3.4) where $\mathbf{W}_i^{l,k}$ is the ith column of the kth filter of $\mathbf{W}^l$.

$$\mathbf{W}^{l,k} * \mathbf{X}^{l-1}_{[t,t+m_l-1]} \equiv \sum_{i=1}^{m_l} \langle (\mathbf{W}_i^{l,k})^\top, \mathbf{x}^{l-1}_{t+i-1} \rangle \tag{3.4}$$

Subsequently applying batch normalization and max pooling to the convolution output, the elements of the time step vector, $x_t^{l,k}$, of the different time series, $\{\mathbf{X}^l\}_{l=1}^L$, are calculated as :

$$x_t^{l,k} = \max \left\{ BN_{\Gamma^k,\beta^k}(z_{2t}^{l,k}), BN_{\Gamma^k,\beta^k}(z_{2t+1}^{l,k}) \right\} \tag{3.5}$$

where $BN_{\Gamma^k,\beta^k}(z_{2t}^{l,k})$ is the batch normalization operation, which is defined as :

$$BN_{\Gamma^k,\beta^k}(z_t^{l,k}) = \frac{z_t^{l,k} - \mu_B^k}{\sqrt{(\sigma_B^k)^2 + \epsilon}} \times \Gamma^k + \beta^k \tag{3.6}$$

where μ_B^k and $(\sigma_B^k)^2$ refer to the mean and variance of the elements of the kth dimension computed on the examples of the mini-batch, B ; ϵ is an arbitrarily small constant used for numerical stability ; and finally, Γ and β are parameters learned in the optimization process in order to restore the representation power of the network (Ioffe & Szegedy, 2015). It can be deduced from Equation (3.5) that the width of the max pooling operator is 2 ; hence, the length of the time series is halved after each convolution block (i.e., $T_l = \frac{1}{2}T_{l-1}$).

Rather then connecting the output of the last convolution block to a classifier, as was done in previous 1D-CNN based works (Imran & Raman, 2020; Zeng *et al.*, 2014; Yang *et al.*, 2015; Ronao & Cho, 2016), the proposed method instead achieves high-level reasoning (inference and learning) by connecting to a classifier the maximum and minimum values of each dimension of

each time series generated throughout the network. Equivalently, this concept can be regarded as generating a feature vector, $\mathcal{V}$, by sampling elements of the different time series as such :

$$\mathcal{V} = \bigcup_{l} \left\{ \max_{t}\{x_t^{l,k}\}, \quad \min_{t}\{x_t^{l,k}\} \right\}_{k=1}^{F_l} \tag{3.7}$$

and when doing so, it is important to keep a constant ordering of the elements of the feature vector. This way, the sampled values from a specific time series associated with a specific convolution filter always exploit the same connections to the classifier. For that purpose, Algorithm 1 was used in order to create the feature vectors :

As CNNs learn convolution filters that react to specific features, these maximum and minimum activation values are correlated with specific motions. To ensure that the convolution filters learn and capture discriminating dynamics for every action class, different sets of filter banks were independently learned through multiple binary classification problems and grouped convolutions. From this process, illustrated in Figure 3.1, N different feature vectors, $\{\mathcal{V}\}_{c=1}^{N}$, were created based on the discriminating filters learned for each of the N different actions. Additionally, this

process also resulted in N different binary classifiers.

Algorithme 3.1 Feature vector harvesting method

Input : *The set of time series generated by the L convolution blocks* $\mathbf{X} = [\mathbf{X}^1, \ldots \mathbf{X}^l, \ldots \mathbf{X}^L]$

Output : *Feature vector :* $\mathcal{V} = [\min_t\{x_t^{1,1}\}, ..., \max_t\{x_t^{L,F_L}\}]$

$\mathcal{V}$ <- new hash table;

for *each time series* $\mathbf{X}^l \in \mathbf{X}$ **do**

    min <- new hash table;

    Max <- new hash table;

    for *each dimension, k, of the lth time series vector* $\mathbf{x}_t^l \in \mathbf{X}^l$ **do**

        insert $\min_t\{x_t^{l,k}\}$ into min;

        insert $\max_t\{x_t^{l,k}\}$ into Max;

    end for

    insert min into $\mathcal{V}$;

    insert Max into $\mathcal{V}$;

end for

As each classifier can be taken individually with their specific set of filters in order to recognize a specific action, our method is modular. Thus, it is possible to reduce the computation and memory requirements dramatically during inference if a single or a reduced set of actions is targeted.

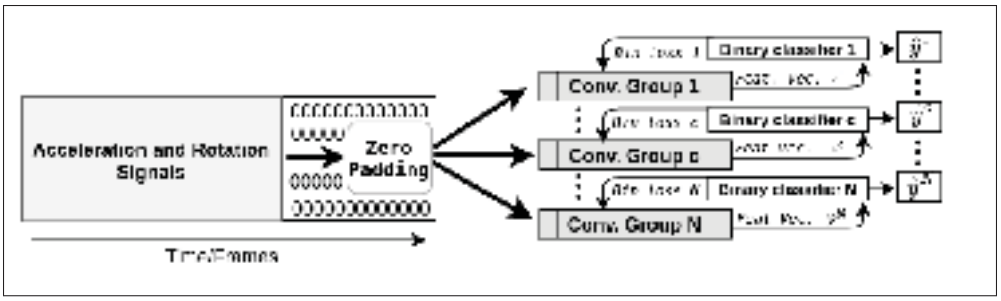

Figure 3.1 An high-level representation of the network architecture and learning mechanisms. Each class has its own convolution group that does the feature extraction and its own binary classifier

Although it is probable that performances could be improved by tailoring a specific architecture for each class, this avenue was not explored in the present work. Instead, each convolution group has the same architecture that is provided by Figure 3.2.

As illustrated, it consists of five consecutive blocks that are defined by the following operations :

- A multi-channel 1D convolution layer,
- Batch normalization,
- Max pooling,
- Feature sampling.

According to the proposed architecture, it is also important to observe that in the first block, the convolution kernels' width (m_1) is set to 1 ; hence, instantaneous dynamics are sampled. As the information moves to subsequent blocks, longer discriminating dynamics are harvested by the coaction of both larger kernels and pooling layers compressing the temporal information while providing a certain degree of invariance towards translations and elastic distortions (Graham, 2014). It is also possible to observe, in Figure 3.2, that each convolution layer has a total of 32 filters (f^l) from which it ensures that each feature vector (depicted at the bottom) has a total of 320 elements (32 min and 32 max for each of the 5 generated times series).

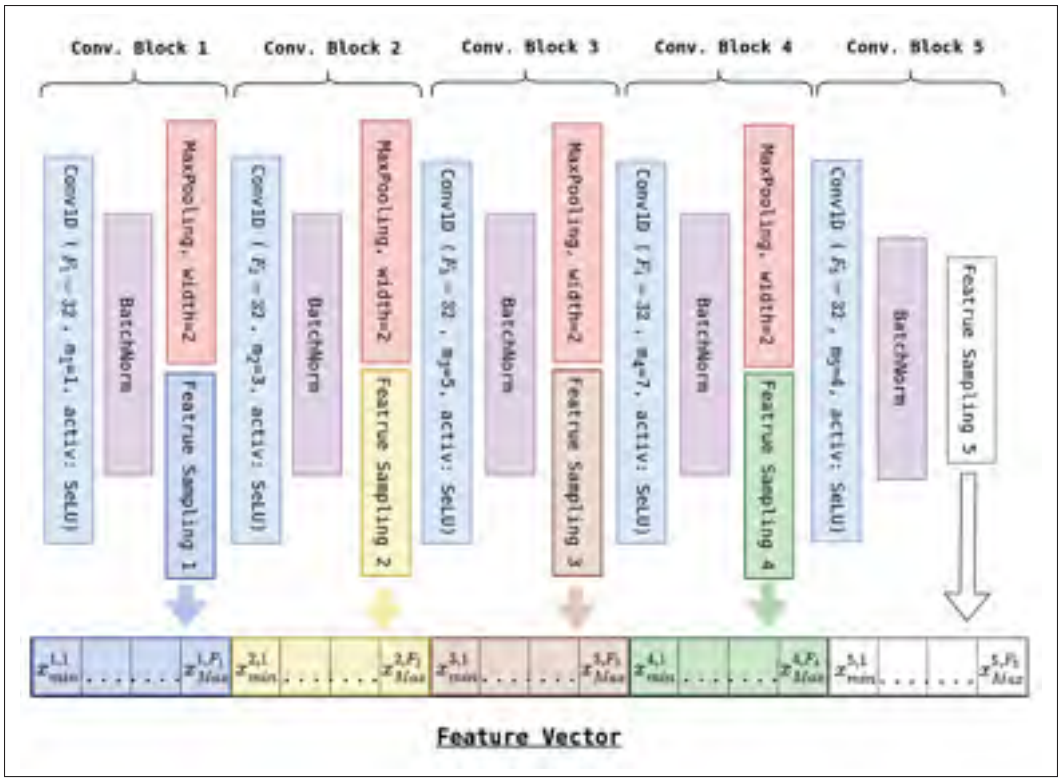

Figure 3.2 Architecture of a convolution group. For the convolution operation (in blue), F_l and m_l refer to the number of convolution filters (F_l) and their width along the temporal dimension (m_l) according to the conv. block to which they belong (l), and activ. refers to the activation function. The purple rectangle is there to illustrate the batch normalization operation applied to the convolutions' output. From this normalized output, max pooling is applied, resulting in a new time series $\mathbf{X}^l$ from which elements of the feature vector are sampled and serving as the next block's input

In order to train binary classifiers (see Figure 3.1), some relabelling had to be done. Therefore, all the examples were relabelled with respect to the different groups as 1 if the group index, c, matched the original multi-class label and 0 otherwise.

$$y^c = \begin{cases} 1, & \text{if } y_c = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

This process is thus described by Equation (8), where y^c refers to the label attributed to the examples going through the cth binary classifier and y_c is the cth element of the original multi-class label expressed as the vector $\mathbf{y} \in \{0, 1\}^N$ such that the true class element is set to 1 and all others to 0 (one-hot-encoding).

$$\mathcal{P}_{rediction} = \arg\max_c \{\hat{y}^c\}_{c=1}^N, \tag{3.9}$$

In the proposed method, predictions were made based on a one-vs.-all approach, which is depicted by Equation (9), where $\hat{y}^c$ corresponds to the probability output for the positive class of the cth binary classifier.

As for the convolution groups, the architectures of the binary classifiers are identical. As shown on Figure 3.3, the feature vector (whose elements are in red) starts by going through a batch-normalization layer (in purple) before going through two fully connected layers (in blue). Unlike the convolution modules, bias was allowed, and the chosen activation function was ReLU. Additionally, dropout layers (in yellow) randomly dropping half of the connections after both fully connected layers were inserted in order to reduce overfitting. Lastly, as part of each learning process, the classifiers' outputs ($\hat{y}^c$ and $1 - \hat{y}^c$) were made "probabilistic" (positive and summing to one) using the softmax function (in green).

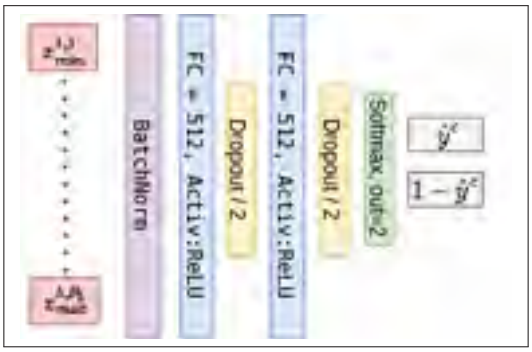

Figure 3.3 Binary classifier

The cost function used during the training phase was the weighted cross-entropy, which is expressed by Equation (3.10). As it has been shown to be an effective way to deal with unbalanced data (Aurelio, de Almeida, de Castro & Braga, 2019), the weights were set to be inversely proportional to the number of training examples of a specific class relative to the total number of training examples. Thus, by assigning a weight of 1 to the positive class' examples, the weight of the negative class' examples (label = 0) is $\frac{S_c}{M-S_c}$, where S_c is the number of training examples of the specific class c and M is the total number of training examples.

$$- y^c \log\left(\hat{y^c}\right) - \frac{S_c}{M - S_c}\left(1 - y^c\right) \log\left(1 - \hat{y^c}\right) \tag{3.10}$$

3.5 Temporal Analysis

With the proposed method, it is possible to localize in time each element of the feature vector used in order to make predictions and/or train the network. As a matter of fact, the functions torch.max and torch.min of the PyTorch library, used in order to create our feature vectors, return the index location of the maximum or minimum value of each row of the input tensor in a given dimension, k. Hence, this section will explain how the sampled elements of the feature vector can be associated with a confidence interval in the original time series (IMU signals) and that elements sampled from different convolution blocks report dynamics of different durations.

For the first convolution block output, this analysis is trivial. As the convolution kernels of the first block are of width one, if the returned index of the min and/or max function is t, Equation (3.5) indicates that this element is related to a specific dynamics that occurred either during the $(2t)$th or $(2t+1)$th time frame of the original time series (IMU signals). Unfortunately, the exact occurrence cannot be determined as pooling is not a bijective operation. Furthermore, as information gets sampled from the elements generated by deeper blocks, this uncertainty grows, but remains bounded, making it possible to determine a confidence interval for the sampled features.

In order to illustrate this process, let us take a look at Figure 3.4, which takes the previous example one convolution block deeper. For both convolution blocks, the first line refers to the block's input; the following is the result of the convolution layer; and the third is the result of the pooling layer. In order to simplify the analysis, the time step vectors only have one dimension, and batch normalization is omitted as it has no influence on the temporal traceability. As the second block's convolution filter has a width of three (i.e., $m_2 = 3$), if a feature is sampled from the second time series, $\mathbf{X}^2$, with a relative position index of t, the corresponding dynamic consequently occurred between the $(4t)$th and the $(4t + 7)$th time steps of the original signal.

Generally speaking, knowing that the stride of the convolutions are set to one and the max pooling width is two, it is possible to determine the length of a certain feature based on the convolution block from which it was sampled.

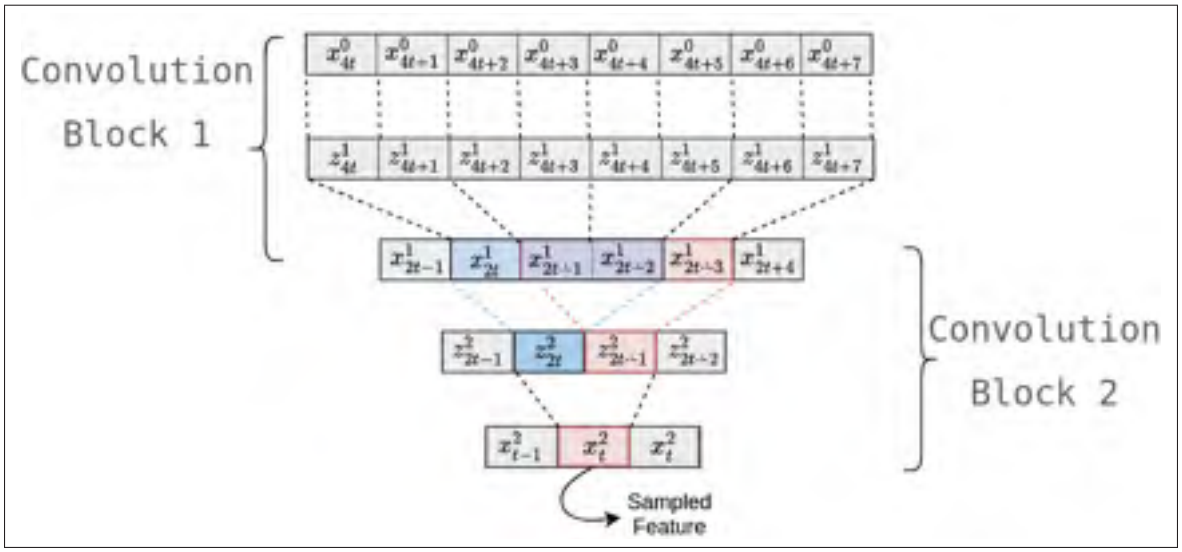

Figure 3.4 Temporal tracing of a feature of the second convolution block. The indexes correspond to those of Equation 3.10. The blue and red rectangles express the two different positions of the same convolution kernel of the second convolution block

Defining D^l, m_l and E^l as the duration of the features, the width of the convolution filters and the temporal uncertainty caused by max pooling of the lth convolution block, it is possible to inductively calculate the duration of each convolution block feature using the following set of

equations :

$$\begin{aligned} E^l &= 2^{l-1} \\ D^1 &= 1 \\ D^l &= D^{l-1} + E^{l-1} + (m_l - 1) \times 2^{l-1} \end{aligned} \tag{3.11}$$

Consequently, the confidence interval, I, during which the dynamic of a certain feature occurred, can be expressed as :

$$I_t^l \in [t \times 2^l, t \times 2^l + D^l + E^l[\tag{3.12}$$

Table 3.1 gives the relative duration, D, the relative uncertainty, E (both with respect to the original time step), and the real duration (knowing that the original signal was acquired at 30 Hz) based on the layer from which they were sampled and the width of the convolution filters specified by the proposed architecture.

Tableau 3.1 Characterisation of the uncertainty (E) and relative duration (D) of the sampled features based on the networks hyper-parameters (l) and (m)

Conv. Block (l)	**Kernel width (m)**	**Relative Duration (D)**	**Relative Uncertainty (E)**	**Feature Duration**
1	1	1	1	0.033 s
2	3	6	2	0.2 s
3	5	22	4	0.73 s
4	7	70	8	2.33 s
5	4	118	16	3.93 s

3.6 Experiments

Using the publicly available UTD-MHAD dataset (Chen *et al.*, 2015), which proposes a total of 27 different actions performed 4 times by eight different subjects (4 males and 4 females), this section will first demonstrate how our hierarchical framework makes our method robust towards zero-padding and that it is possible to train and infer on sequences of different lengths without

impacting the performances. Then, we will assess the performances and compare them with state-of-the-art methods. Finally, we will conduct an analysis on the complexity of the method and see how it compares with some well-known CNN architectures.

Although it provides RGB, depth and skeleton streams, our experiments were conducted only on the inertial sensor signals, which were acquired from a single IMU worn on the right wrist for Actions 1 through 21 and on the right tight for Actions 22 through 27, as displayed in Figure 3.5. For all our experiments, the evaluation protocol suggested in the original UTD-MHAD paper (Chen *et al.*, 2015) was followed, where odd subjects are for training and evens for evaluation, except in Section 3.6.2, which will additionally provide an assessment of the performances through a leave-one-out k-fold cross-validation protocol. In all cases, the experiments were performed using the PyTorch deep learning framework with stochastic gradient descent as the optimizer and a batch size of 16. The learning rate was initialized at 0.01 and decayed 1% per epoch. Weights were initialized using the Xavier method, and the bias of the classifier was initialized using a Gaussian distribution of mean 0 and variance 1.

Figure 3.5 Wearable inertial sensor placements
Taken from Chen *et al.* (2015)

3.6.1 Robustness against Zero-Padding and Flexibility Towards the Input's Length

The idea behind our feature sampling method was to create a feature vector based on information sampled from multiple local signals instead of encoding the whole sequence into a feature vector.

Hence, meaningless and/or redundant information such as zero-padding does not affect the feature vector, nor the classification performances. In order to validate this hypothesis, signals were extended up to various lengths with zero-padding. More precisely, training using sequences normalized up to 326, 700 and 1000 frames was conducted, for which the results are reported by Figure 3.6.

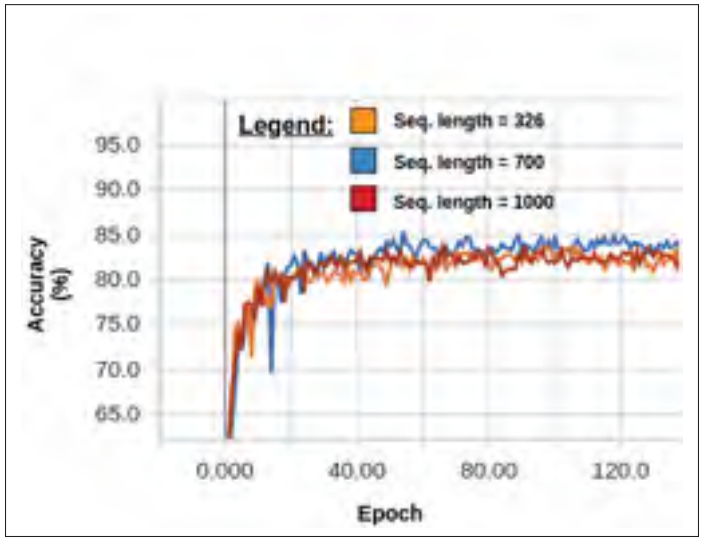

Figure 3.6 Learning curves assessing the accuracy obtained on the test set (Y axis) relative to the number of training epochs (X axis) for sequences (Seq.) zero-padded up to different extent

As we can see, the performances are rather steady around 83% regardless of the amount of zero-padding added to the original sequence. This is an interesting result as Imran & Raman (2020) previously conducted a 4-fold cross-validation analysis using only the subjects of the training set and observed that using the conventional CNN approach, where only the information encoded in the last layer is provided to the classifier, zero-padding the IMU signals of UTD-MHAD dataset caused the accuracy to decay by nearly 7%. More precisely, they transitioned from randomly sampled sequences of 107 time frames (which corresponded to the duration of the shortest action) to sequence zero-padded up to 326 time frames (which corresponded to the longest action). Intermediate lengths were also tested, where shorter sequences were zero-padded, while

longer ones were randomly sampled. As observed on Table 3.2, their accuracy kept decreasing as the amount of zero-padding increased.

Tableau 3.2 Mean accuracy of the k-fold cross-validation analysis produced by Imran & Raman (2020) with respect to zero-padding on their 5 layer 1D-CNN

Normalized Sequence Length	107	180	250	326
Max. Accuracy	83.99%	80.05%	78.42%	77.04%

Moreover, it was also observed that with our hierarchical method, it was possible to train the model using sequences of a certain length and proceed to classify sequences of a different length, which is impossible with the other CNN approaches. As a matter of fact, the parameters learned at the 100th epoch of the model trained with the sequence extended to 700 time frames (see Figure 3.6) resulted in an accuracy of 84.88%, which was also reproduced regardless of the testing examples being zero-padded up to 326 or 1000 time frames.

3.6.2 Performances

With our method, we obtained a maximal accuracy score of 85.35%, which resulted in the confusion matrix provided by Figure 3.7. As we can see, the performances were better on actions for which the inertial sensor was placed on the thigh rather than on the wrist.

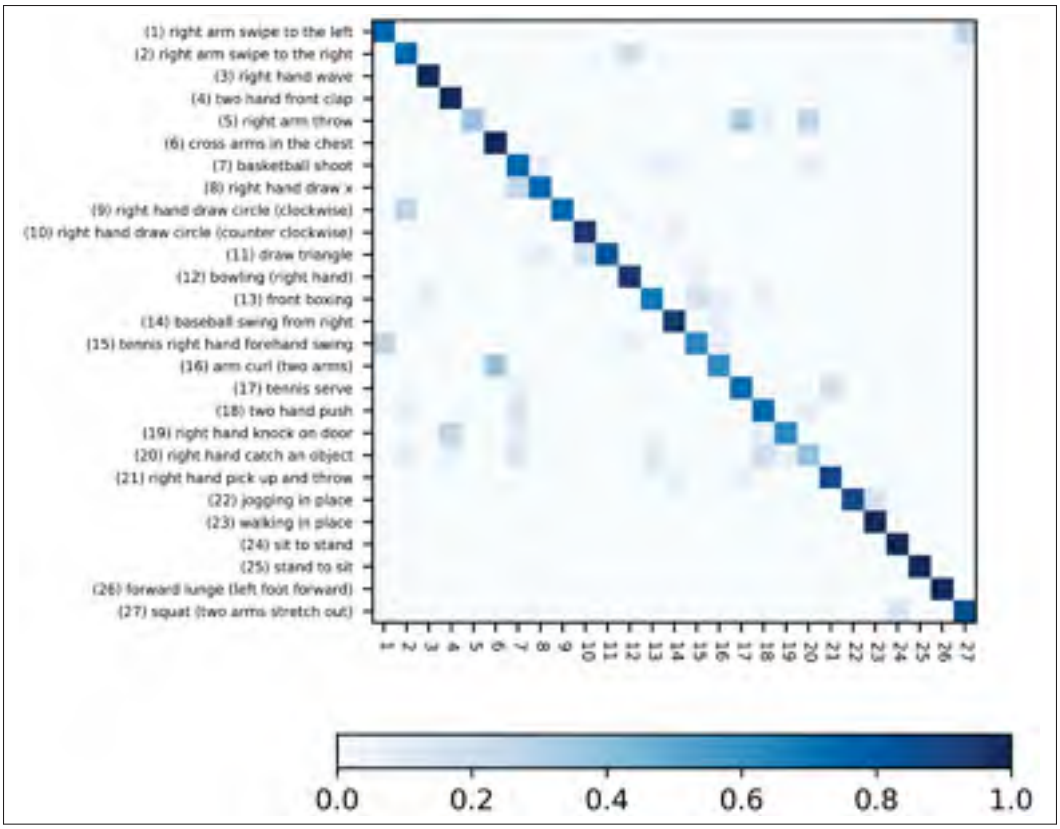

Figure 3.7 Confusion matrix obtained using the one-vs.-all discriminating approach

In order to compare our result, we built Table 3.3, which indicates the accuracy of previous methods and the modality that they used in order to achieve it. Furthermore, we also report the performances on the NTU-RGB-D dataset for the methods that validated their approach on it using the cross-subject/cross-view evaluation protocol (CS/CV), as it is a well-established benchmark for HAR. As we can see, our method is very competitive as it outperformed the current state-of-the-art using inertial data before it benefited from the data augmentation, which, in our case, was not implemented. Another important observation that can be made by taking a closer look at Table 3.3 is that the best overall performance was achieved by Imran & Raman (2020) through a multi-stream fusion approach using inertial signals. Moreover, in the conclusion of their work, Imran and Raman argued that it was not only the complementarity between the different modalities that helped them achieve these results, but also the complementarity

between the architecture treating the different data streams, thus further stressing the importance of 1D-CNN architectures.

Tableau 3.3 Comparison with the literature. CS/CV, cross-subject/cross-view

Mehtod	**Modality**	**UTD-MHAD**	**NTU-RGB-D (CS/CV)**
Chen *et al.* (2015)	Inertial	67.20%	
Imran & Raman (2020) (w/o data augmentation)	Inertial	83.02%	
Imran & Raman (2020) (with data augmentation)	Inertial	86.51%	
Hussein *et al.* (2013)	Skeleton	85.58%	
Wang, Li, Li & Hou (2016)	Skeleton	87.90%	76.32%/81.08%
Hou, Li, Wang & Li (2018)	Skeleton	86.97%	
Imran & Raman (2020)	Skeleton	93.48%	
Li, Hou, Wang & Li (2019)	Skeleton	95.58%	82.96%/90.12%
Chen *et al.* (2015)	Depth + inertial	79.10%	
Wang, Wang, Gao, Hou & Li (2017)	Depth + skeleton	89.04%	
El Din El Madany, He & Guan (2016)	Depth + Inertial + Skeleton	93.26%	
Imran & Raman (2020)	RGB + inertial + skeleton	97.91%	
Proposed method	Inertial	85.35%	

In order to compare the performances of our method against the one proposed by Wei *et al.* (2019) that uses a 2D-CNN architecture applied to images generated from the IMU's signals, we also proceeded to the evaluation of the performances using a leave-one-out 8-fold cross-validation, for which the results are summarized by Table 3.4.

Tableau 3.4 k-fold validation on UTD-MHAD

Test Subject	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Accuracy	96.3%	89.8%	88.0%	90.7%	94.4%	91.6%	90.7%	85.1%

Computing the mean value of all these runs, we get a value of 90.8%, which is slightly better than the 90.3% obtained by Wei *et al.* (2019).

3.6.3 Computational Complexity Analysis

As stated earlier, one of the greatest challenges currently faced by HAR algorithms is their portability to limited resource electronics; hence, we proceeded to do an analysis of the computational complexity and the number of parameters, which ended up demonstrating that our approach could be very easily integrated in limited resource electronics.

As a matter of fact, the number of operations needed by our model is directly correlated to the normalized length of the input sequence. As we can see in Figure 3.8, the relation is linear and is equal to 86.37 million multiply and accumulate (MAC) operations when the normalized length is equal to 326.

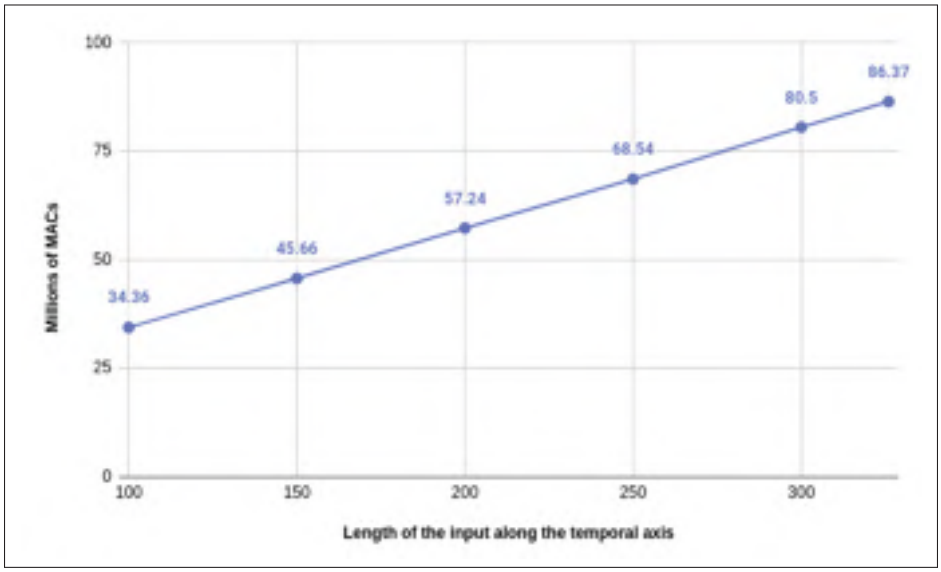

Figure 3.8 Learning curves assessing the accuracy obtained on the test set (Y axis) relative to the number of training epochs (X axis) for different amounts of zero-padding. MAC, multiply and accumulate

Moreover, as our method is modular, the performance of each binary classifier can be studied independently. For that purpose, the precision and recall scores for each class are given by Figure 3.9.

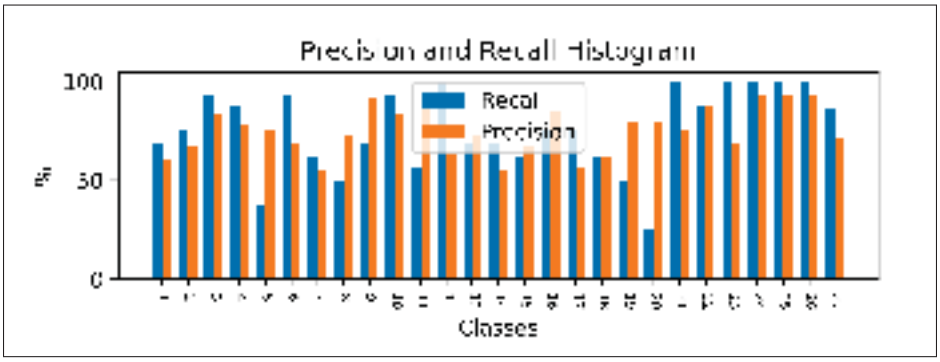

Figure 3.9 Histogram reporting the precision and recall score for each class

Therefore, the number of operations and parameters can be divided by 27 in order to be integrated into a simple device only recognizing one action. In that case, the number of parameters would only be 343,830, which only takes about 1.4 MB of memory if the parameters are represented with single precision floating-point numbers, as was the case in our experiments. In the case of the number of operations, it comes down to about 3.2 million MACs. As MACs account for two floating point operations (one multiplication and one addition), if a processing unit has a throughput of 1 operation per clock cycle, a processing speed of only 3.2 MHz would be necessary in order to run our model under 1 s. As many inexpensive microcontrollers offer sufficient performances, this proves that our method could easily be embedded into smart devices. Finally, comparing our method with some of the most popular CNN architectures regarding the number of computations, on Figure 3.10 and the number of parameters on Figure 3.11, we can appreciate once more how hardware-friendly our method really is.

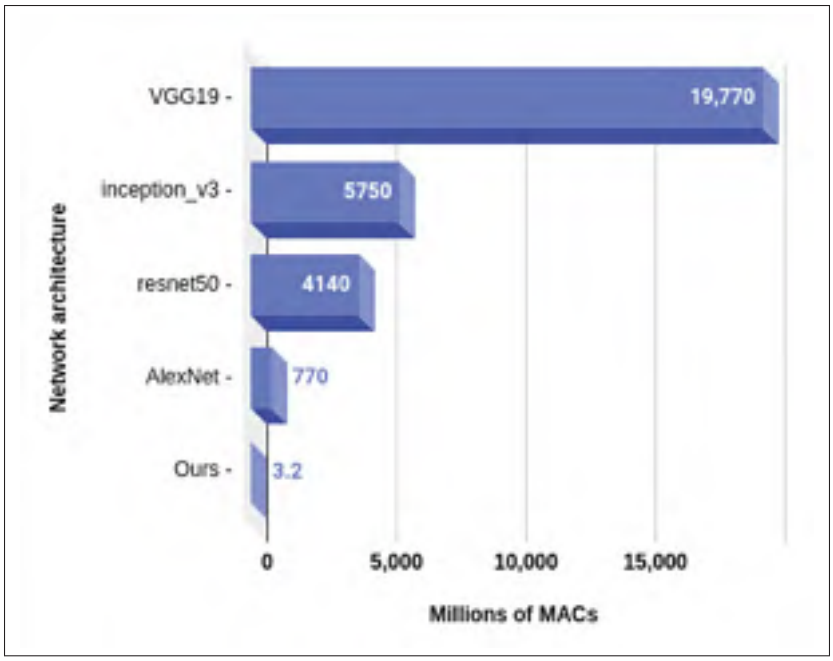

Figure 3.10 Number of computations needed by our model compared to popular CNN architectures

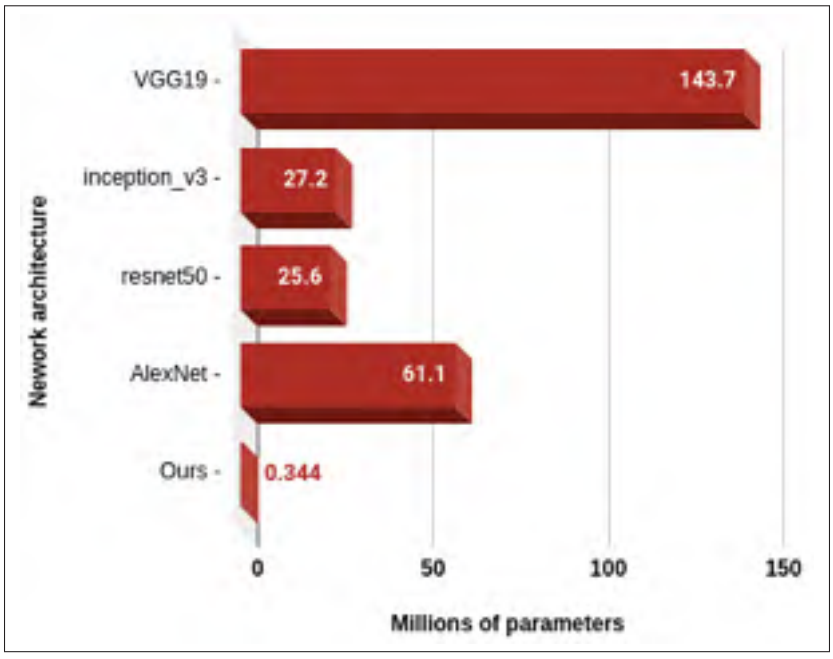

Figure 3.11 Number of parameters needed by our model compared to popular CNN architectures

3.7 Discussion

In this paper, we presented a new framework based on a 1D-CNN architecture in order to perform action recognition on signals generated by IMUs. Although less accurate than methods using visual streams, this approach still has its advantages as it provides more privacy and can be used anywhere, as long as wearing the acquisition device does not become too inconvenient. Moreover, we demonstrated that our method is computationally inexpensive enough to be integrated into embedded devices. On another note, as the latent representation can be viewed as an organized set of elements assessing the presence of specific sub-events, the proposed method works in a hierarchical way. Yet, the temporal relationships between these sub-events are not captured implicitly or explicitly ; thus, we believe that the proposed approach could help address the unsolved problem (Song, Kim & Park, 2018) of high-level action recognition with machine learning based methods as the difficulty that previous algorithms faced with this type of actions lies in the fact that they have important temporal variations between the sub-events composing them. This hierarchical approach also allowed us to infer on times series extended up to various lengths without compromising the accuracy.

Finally, we also demonstrated that the element of the latent representation can be traced in time, both in location and duration. In future work, we would like to investigate this ability further as we believe that it offers more transparency to the inner workings of the model by providing a way to interpret them. For example, this could be done by assessing if specific kernels are in fact reacting or not to discriminating movements based on the movements occurring in their confidence interval. On top of opening a window into the black-box of deep-learning algorithms, we believe that this approach could help reduce over-fitting and propose better architectures, tailored to specific actions, thus improving the overall performances.

CONCLUSION ET RECOMMANDATIONS

En apprenant à reconnaître la signature de dynamiques particulières sans tirer avantage de la logique temporelle d'exécution, la méthode proposée est capable d'inférer sur des séquences de longueurs variables. Afin de reconnaître des actions de haut-niveau (e.g. une bagarre) ces propriétés sont particulièrement intéressantes étant donné que la durée ou la séquence d'exécution ne sont pas déterminées pour ce type d'actions. De plus, grâce à sa grande économie en termes de calculs, la solution proposée s'intègre dans un cadre d'utilisation plus éthique en permettant de développer un système dans lequel l'analyse des signaux serait prise en charge localement et qui ne produirait d'alertes que lorsqu'une intervention serait nécessaire.

De manière plus générale, grâce à son approche proposant un classificateur binaire pour chacune des classes, la méthode présentée offre aussi la possibilité détecter plusieurs actions simultanément et témoigne de performances comparables à l'état de l'art en ce qui a trait à la reconnaissance d'actions basée sur des signaux inertiels.

Au final, si mes travaux devaient être poursuivis, plusieurs avenues seraient envisageables. Premièrement, tel que proposé dans l'article, il pourrait être intéressant d'étudier les dynamiques associées aux éléments du vecteur représentatif de manière à incorporer plus de connaissances humaines. Par exemple, une action ayant des mouvements brusques pourrait privilégier une architecture dont les filtres attesteraient de dynamiques plus courtes. Enfin, pour les actions n'étant pas de haut-niveau, l'implémentation d'une méthode capable de tirer avantage de la logique temporelle d'exécution des différentes actions semble aussi une avenue intéressante.

ANNEXE I

FICHIERS DE COMMANDE

```yaml
# General
work_dir: ./work_dir/HCN_CS
num_epoch: 250
save_interval: 1000
eval_interval: 1

# Dataloader
train_batch_size: 16
test_batch_size: 16
take_n_examples: 752
train_subjects: [1,3,4,5,6,7,8]

# Model
input_window_size: [700, 6]
model: model.hierarchical_CNN_SeLU.Model
model_args:
  f_1: 32
  f_2: 32
  f_3: 32
  f_4: 32
  f_5: 32

  nb_class: 27
  bin_layer: 256
  multi_layer: 512


# Optimizer
optimizer: torch.optim.SGD
loss: torch.nn.CrossEntropyLoss
binary_weights: [0.0385, 1]
optimizer_args:
  lr: 0.01

# Schedule
num_worker: 4
scheduler: torch.optim.lr_scheduler.ExponentialLR
scheduler_args:
  gamma: 0.99
```

Figure-A I-1 Fichier de configuration pour de la solution basée sur l'information inertielle

```yaml
work_dir: ./work_dir/HCN_CS
num_epoch: 150
save_interval: 25
eval_interval: 1

# Dataloader
train_batch_size: 1
test_batch_size: 32
take_n_examples: -1
NTU120: False
eval_protocol: CV
validation_view: 3

## model
model: model.init_model.Model
model_args:
  in_channels: 3
  num_class: 60


# ncls model
model: model.binary_learned_multiclass_ncls.Model
model_args:
  pose_out: 2
  h_rn: 2
  nb_poses: 2
learning_type: multiclass
binary_weights: [0.0166, 1]
interest_class: 7
input_window_size: [150, 25]
pre_treatment_type: raw_buffed

# Optimizer
optimizer: torch.optim.Adam
optimizer_args:
  lr: 0.001
  weight_decay: 0.0001
loss: torch.nn.CrossEntropyLoss
focal_loss: False
regulize_orthogonality: False

# Initialization
weights: '/home/ncls/Documents/HCN_custom/model/saved_models/CSV60_epoch250_model.pt'
ignore_weights: [semantic_learner_pose.pose_convolution, semantic_learner_motion.pose_convolution] #classifier
ignore_weights: [classifier]
freeze_layers: [conv_position, conv_motion] #pose_convolution, time_convolution, action_card_convolution]

# Scheduler
num_worker: 1
scheduler: torch.optim.lr_scheduler.ExponentialLR
scheduler_args:
  gamma: 0.99
sampler:  # weighted
weight_function: #ratio  #balanced ratio
interest_ratio: 1

# Evaluation and visualization
specific_action_list: [24,31,50,51,52,54,93,98,102,106,107,108,110,118]
dangerous_act_list: [50]  #[50, 51, 52, 102, 106, 107, 108, 118]
suspicious_act_list: [] #[24, 31, 54, 93, 98, 118]
```

Figure-A I-2 Fichier de configuration pour de la solution utilisant la représentation exo-squelettique

ANNEXE II

SCRIPT D'ALLOCATION DES RESSOURCES

```
#!/bin/bash
#SBATCH --time=0-01:15:00  # jours-heures:minutes:secondes
#SBATCH --account=def-rnoumeir

#SBATCH --gres=gpu:1       # Request GPU "generic resources"
#SBATCH --cpus-per-task=8  # Cores proportional to GPUs: 6 on Cedar, 16 on Graham.
#SBATCH --mem=20000M       # Memory proportional to GPUs: 32000 Cedar, 64000 Graham.


#SBATCH --mail-user=nicolas.lemieux.1@etsmtl.net
#SBATCH --mail-type=ALL



#SBATCH --output=job-%j.out


cd ../Inertial_action_recognition/
python utd_activation_study.py



sleep 30
```

Figure-A II-1 Script de demande d'allocation des ressources sur *Calcul Canada*

ANNEXE III

GUIDE D'UTILISATION DES OUTILS LOGICIELS

Reconnaissance d'actions basée sur des données inertielles utilisant une approche hiérarchique basée sur l'apprentissage machine

A Hierarchical learning approach for humain action recognotion est le titre de l'article publié dans le journal MDPI à la lumière des résultats générés par le code de ce programme. [Publication]

Contributions

1) Méthode d'échantillonnage multi-résolution et hiérarchique des éléments du vecteur représentatif d'un réseau convolutif en une dimension (1D-CNN)

Architecture logicielle inspirée de : * https://github.com/yysijie/st-gcn

Base de données

Les signaux inertiels de la base de données UTD-MHAD ont été utilisés pour exécuter les expériences. Ceux-ci doivent être téléchargés et placés dans longlet `./data/Inertial` afin de reproduire les expériences.

Prérequis

Ce code est basé sur **Python 3.6**. Les dépendances sont spécifiées dans le fichier environment.yml. Afin de les installer, il est suggéré de créer un environnement conda puis de lancer la commande suivante:

```
conda env create -f environment.yml
```

Utilisation

Exécution du programme

Il est suggéré de lancer les entrainements à l'aide d'un fichier de commande (dont les paramètres spéciables sont décrits ci-dessous). Dans le cas où le fichier de commande se nomerait exemple.yaml et serait placé sous l'onglet (`./config`), la commande d'exécution du programme serait la suivante :

```
$ python run.py -c config/exemple.yaml
```

Fichier de commande

General

use_gpu *(type:bool)*

Définit si les calculs doivent / peuvent exécuté sur une carte graphique.

work_dir *(type:str)*

Permet de définir le fichier où les sorties (Graphes d'analyse / modèle sauvegardé / fichier de configuration utilisé) du programme sont enregistrées.

num_epoch *(type:int)*

Définit le nombre d'époques d'entraînement.

save_interval *(type:int)*

Définit l'intervalle (en nombre d'époques d'entraînement) auquel le model est sauvegardé.

eval_interval *(type:int)*

Définit l'intervalle (en nombre d'époques d'entraînement) pour lequel les performances sont évalué sur l'ensemble de test.

Dataloader

train_batch_size *(type:int)*

Définit la taille des mini-lots lors de l'entraînement.

test_batch_size *(type:int)*

Définit la taille des mini-lots lors de la phase de test.

take_n_examples *(type:int)*

Définit le nombre d'exemples maximum dans chaque ensemble (entraînement et validation) afin d'aider au débuggage. Lorsque la valeur est mise à -1 tous les exemples disponibles sont alors utilisés.

Model

input_window_size *(type:list)*

Définit la longueur des séquences jusqu'à concurence de laquelle on ajoute des 0s (Zero-Padding) ainsi que le nombre de dimension du signal d'entrée

model

Définit le model à utiliser.

weights

Permet de définir le fichier où les poids appris lors d'un entraînement antérieur auraient été sauvegardés afin d'initialiser le model.

freeze_layers *(type:str)*

Permet de définir des couches pour lesquelles on voudrait empêcher l'apprentissage de nouveaux poids.

model_args

Permet de définir la topologie du réseau.

f_1 *(type:int)*

Nombre de filtres de convolution dans la première couche.

f_2 *(type:int)*

Nombre de filtres de convolution dans la deuxième couche.

f_3 *(type:int)*

Nombre de filtres de convolution dans la troisième couche.

f_4 *(type:int)*

Nombre de filtres de convolution dans la quatrième couche.

f_5 *(type:int)*

Nombre de filtres de convolution dans la cinquième couche.

nb_class *(type:int)*

Nombre de classes du problème.

bin_layer *(type:int)*

Nombre de neurones dans les couches cachées des classifieurs binaires.

phase *(type:str)*

Permet de définir si le modèle est en mode entraînement (train) ou d'inférence (test).

Optimizer

Permet de définir les hyper-paramètres liés à la méthode d'optimisation des poids.

optimizer

Definit la méthode d'optimisation à utiliser.

loss

Définit la fonction de coût que la méthode d'optimisation tente de minimiser.

binary_weights *(type:list)*

Permet de pondérer la fonction de coût en fonction de la classe de l'étiquette d'un problème de classification binaire.

optimizer_args:

lr

Définit le taux d'apprentissage associé à la méthode d'optimisation.

Scheduler

Permet de définir une méthode adaptative pour le processus d'apprentissage. Par exemple réduire le taux d'apprentissage en fonction de nombre d'époque.

Visualisation des résultats

La visualisation des résultats et graphes d'analyses générés lors des entraînements et/ou phase de test (générés dans le dossier work_dir) est possible à l'adresse suivante : http://localhost:6006/ (à lancer dans un navigateur internet) après avoir lancé la commande suivante :

```
commandline $ tensorboard --logdir ./work_dir
```

Lire la documentation

Pour lire la documentation relative au projet il suffit de cliquer sur le fichier index.html sous longlet docs/_build ou de lancer la commande suivante depuis un terminal dont le dossier courant est celui du projet

```
commandline $ google-chrome _build/index.html
```

LISTE DE RÉFÉRENCES

Aurelio, Y. S., de Almeida, G. M., de Castro, C. L. & Braga, A. P. (2019). Learning from Imbalanced Data Sets with Weighted Cross-Entropy Function. *Neural Process. Letters*, 50(2), 1937–1949.

Bloom, V., Makris, D. & Argyriou, V. (2012). G3D : A gaming action dataset and real time action recognition evaluation framework. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 7–12.

Chen, C., Jafari, R. & Kehtarnavaz, N. (2015). UTD-MHAD : A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 168–172.

Chuankun Li, Pichao Wang, Shuang Wang, Yonghong Hou & Wanqing Li. (2017). Skeleton-based action recognition using LSTM and CNN. *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 585–590.

De la Torre, F., Hodgins, J., Bargteil, A., Martin, X., Macey, J., Collado, A. & Beltran, P. (2009). Guide to the carnegie mellon university multimodal activity (cmu-mmac) database.

Du, Y., Fu, Y. & Wang, L. (2015). Skeleton based action recognition with convolutional neural network. *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 579–583.

El Din El Madany, N., He, Y. & Guan, L. (2016). Human action recognition via multiview discriminative analysis of canonical correlations. *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 4170–4174.

Fothergill, S., Mentis, H., Kohli, P. & Nowozin, S. (2012). Instructing people for training gestural interactive systems. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1737–1746.

Gavrila, D. M., Davis, L. S. et al. (1995). Towards 3-d model-based tracking and recognition of human movement : a multi-view approach. *International workshop on automatic face-and gesture-recognition*, pp. 272–277.

Graham, B. (2014). Fractional Max-Pooling. *arXiv :cs.CV/1412.6071*.

Hammerla, N. Y., Halloran, S. & Ploetz, T. (2016). Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables.

Herath, S., Harandi, M. & Porikli, F. (2017). Going deeper into action recognition : A survey. *Image Vis. Comput.*, 60, 4–21.

Hou, Y., Li, Z., Wang, P. & Li, W. (2018). Skeleton Optical Spectra-Based Action Recognition Using Convolutional Neural Networks. *IEEE Trans. Circuits Syst. Video Technol.*, 28(3), 807–811.

Hussein, M. E., Torki, M., Gowayyed, M. A. & El-Saban, M. (2013). Human Action Recognition Using a Temporal Hierarchy of Covariance Descriptors on 3D Joint Locations. *IJCAI*.

Imran, J. & Raman, B. (2020). Evaluating fusion of RGB-D and inertial sensors for multimodal human action recognition. *J. Ambient Intell. Humaniz. Comput.*, 11(1), 189–208.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Jiang, W. & Yin, Z. (2015). Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks. *Proceedings of the 23rd ACM international conference on Multimedia*, (MM '15), 1307–1310.

Ke, Q., Bennamoun, M., An, S., Sohel, F. & Boussaid, F. (2017, July). A New Representation of Skeleton Sequences for 3D Action Recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. (2017). Self-Normalizing Neural Networks.

Kwapisz, J. R., Weiss, G. M. & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2), 74–82.

Lemieux, N. & Noumeir, R. (2020). A Hierarchical Learning Approach for Human Action Recognition. *Sensors*, 20(17).

Li, C., Hou, Y., Wang, P. & Li, W. (2019). Multiview-Based 3-D Action Recognition Using Deep Networks. *IEEE Transactions on Human-Machine Systems*, 49(1), 95–104.

Li, C., Zhong, Q., Xie, D. & Pu, S. (2018). Co-occurrence Feature Learning from Skeleton Data for Action Recognition and Detection with Hierarchical Aggregation.

Liu, C., Hu, Y., Li, Y., Song, S. & Liu, J. (2017a). Pku-mmd : A large scale benchmark for continuous multi-modal human action understanding. *arXiv preprint arXiv :1703.07475*.

Liu, J., Wang, G., Hu, P., Duan, L.-Y. & Kot, A. C. (2017b). Global context-aware attention LSTM networks for 3D action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1647–1656.

Liu, J., Shahroudy, A., Xu, D., Kot, A. C. & Wang, G. (2018). Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12), 3007–3021.

Liu, J., Shahroudy, A., Perez, M. L., Wang, G., Duan, L.-Y. & Kot Chichung, A. (2019). NTU RGB+D 120 : A Large-Scale Benchmark for 3D Human Activity Understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*

Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B. & Weber, A. (2007). Documentation mocap database hdm05.

Murad, A. & Pyun, J.-Y. (2017). Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11), 2556.

Ni, B., Wang, G. & Moulin, P. (2011). Rgbd-hudaact : A color-depth video database for human daily activity recognition. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1147–1153.

Nishida, T. (2007). Social intelligence design and human computing. Dans *Artifical Intelligence for Human Computing* (pp. 190–214). Springer.

Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R. & Bajcsy, R. (2013). Berkeley mhad : A comprehensive multimodal human action database. *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 53–60.

Ordóñez, F. J. & Roggen, D. (2016). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, 16(1).

Rautaray, S. S. & Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction : a survey. *Artificial Intelligence Review*, 43(1), 1–54.

Ronao, C. A. & Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Syst. Appl.*, 59, 235–244.

Shahroudy, A., Liu, J., Ng, T.-T. & Wang, G. NTU RGB+D : A Large Scale Dataset for 3D Human Activity Analysis.

Song, D., Kim, C. & Park, S.-K. (2018). A multi-temporal framework for high-level activity analysis : Violent event detection in visual surveillance. *Inf. Sci.*, 447, 83–103.

Vemulapalli, R., Arrate, F. & Chellappa, R. (2014). Human action recognition by representing 3d skeletons as points in a lie group. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 588–595.

Vermesan, O., Friess, P. et al. (2014). *Internet of things-from research and innovation to market deployment*. River publishers Aalborg.

Wan, J., Zhao, Y., Zhou, S., Guyon, I., Escalera, S. & Li, S. Z. (2016). Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 56–64.

Wang, J., Liu, Z., Wu, Y. & Yuan, J. (2012). Mining actionlet ensemble for action recognition with depth cameras. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1290–1297.

Wang, J., Chen, Y., Hao, S., Peng, X. & Hu, L. (2019). Deep learning for sensor-based activity recognition : A survey. *Pattern Recognit. Lett.*, 119, 3–11.

Wang, P., Li, W., Li, C. & Hou, Y. (2016). Action Recognition Based on Joint Trajectory Maps with Convolutional Neural Networks.

Wang, P., Wang, S., Gao, Z., Hou, Y. & Li, W. (2017). Structured images for RGB-D action recognition. *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1005–1014.

Wei, H., Jafari, R. & Kehtarnavaz, N. (2019). Fusion of Video and Inertial Sensing for Deep Learning-Based Human Action Recognition. *Sensors*, 19(17).

Wei, P., Zheng, N., Zhao, Y. & Zhu, S. (2013). Concurrent Action Detection with Structural Prediction. *2013 IEEE International Conference on Computer Vision*, pp. 3136–3143.

Weiss, G. M. & Lockhart, J. (2012). The Impact of Personalization on Smartphone-Based Activity Recognition. *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Xia, L., Chen, C. & Aggarwal, J. K. (2012a). View invariant human action recognition using histograms of 3D joints. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27.

Xia, L., Chen, C.-C. & Aggarwal, J. K. (2012b). View invariant human action recognition using histograms of 3d joints. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 20–27.

Yan, S., Xiong, Y. & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *Thirty-second AAAI conference on artificial intelligence*.

Yang, J., Nguyen, M. N., San, P. P., Li, X. L. & Krishnaswamy, S. (2015). Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. *Twenty-Fourth*

International Joint Conference on Artificial Intelligence.

Yang, X. & Tian, Y. L. (2012). Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. *2012 IEEE computer society conference on computer vision and pattern recognition workshops*, pp. 14–19.

Ye, F., Tang, H., Wang, X. & Liang, X. (2019). Joints Relation Inference Network for Skeleton-Based Action Recognition. *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 16–20.

Yun, K., Honorio, J., Chattopadhyay, D., Berg, T. L. & Samaras, D. (2012). Two-person interaction detection using body-pose features and multiple instance learning. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 28–35.

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P. & Zhang, J. (2014). Convolutional Neural Networks for human activity recognition using mobile sensors. *6th International Conference on Mobile Computing, Applications and Services*, pp. 197–205.

Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J. & Zheng, N. (2019). View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*

Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L. & Xie, X. (2016). Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks. *Thirtieth AAAI Conference on Artificial Intelligence*.