Reconnaissance d'activités quotidiennes à partir de données capteurs pour l'assistance de vie dans un environnement connecté

Stagiaire : Nassim Mokhtari, UBO
Encadrant : Alexis Nédélec, ENIB/CERV

Juin 2020

Résumé

La reconnaissance de l'activité humaine basée sur l'utilisation de différents types de capteurs (objets, ambiant ou portés) est un problème de classification de série chronologique. Le choix du type de modèle à utiliser ainsi que son architecture représente une étape cruciale.

Le travail proposé dans ce stage repose sur la configuration d'un modèle d'apprentissage profond hybride (CNN-LSTM) où nous proposons d'exploiter les métaheuristiques dans le but de choisir une bonne architecture du modèle. Nous présentons aussi une nouvelle approche pour transformer les signaux capteurs sous la forme d'images qui permettrait de configurer un simple réseau neuronal convolutif pour la reconnaissance d'activités humaines.

Mots-clés : Reconnaissance d'activités humaines, Apprentissage profond, Réseau de neurones convolutifs, Réseau de neurones récurrents, Méthodes hybrides, Métaheuristiques, Algorithme Génétique, Algorithme FireFly

1 Introduction

L'objectif de ce travail est de mettre en place un système de reconnaissance de l'activité humaine à partir d'algorithmes d'apprentissage machine dans le but de faciliter l'assistance de vie pour des personnes souffrant de troubles mentaux. Ce stage est dans la prolongation de l'étude bibliographique menée durant le semestre précédent. Dans ce contexte d'assistance à la personne, les capteurs doivent être non-invasifs, d'utilisation quotidienne, ambiants ou portés à l'exemple des smartphones et montres connectées.

L'extraction de connaissance à partir de données capteurs est devenu un domaine de recherche très actif en partie grâce à l'accessibilité des données engendrées par les avancées technologiques dans le domaine de l'internet des objets et de leur omniprésence dans la vie quotidienne [29] cité dans [6]. Les recherches sur la reconnaissance de l'activité humaine (HAR en anglais) ont pris une plus grande ampleur ces dernières années du fait de son utilisation dans plusieurs domaines tels que la sécurité basée sur la surveillance ainsi que l'assistance de vie [27] cité dans [6]. Ces recherches exploitent différentes

techniques d'apprentissage automatique dans le but de reconnaître des activités simples ou complexes comme se lever, marcher, courir, cuisiner etc... [7]. On distingue deux types de reconnaissance de l'activité humaine, celle basée sur la vidéo et celle basée sur les capteurs [25] cité dans [8]. La première catégorie s'intéresse à l'analyse de vidéos ou d'images contenant des mouvements humains, tandis que la deuxième s'intéresse aux données émises par des capteurs tels que l'accéléromètre, le gyroscope, le bluetooth, les capteurs de sons, etc... .

Nous nous intéressons ici à la reconnaissance de l'activité humaine basée sur cette dernière catégorie car elle représente le double avantage d'être non-invasive et d'utilisation quotidienne pour une majorité de personnes [8].

Les activités humaines ont des structures hiérarchiques indissociables, ce qui rend l'utilisation des capteurs dans le domaine du HAR sensibles à de simples petites variations en entrée. Des études ont été faites dans le domaine afin de déterminer les caractéristiques d'une activité décomposable en mouvements plus simples, ainsi que les différences de formes et styles d'une même activité réalisée par plusieurs personnes [30, 26] cités dans [6]. Ces caractéristiques ainsi que les séries chronologiques constituent les informations de base, si elles sont bien utilisées, à l'aide d'un classifieur et d'une extraction des caractéristiques les plus importantes au préalable, pour reconnaître des activités [6].

Les avancées récentes dans le domaine de la reconnaissance d'image et de la parole liées aux travaux de recherche sur l'apprentissage profond, particulièrement les réseaux de neurones convolutifs «*convnets* », ont démontré leurs intérêt pour l'extraction des caractéristiques et de classification et semblent être les mieux adaptés à notre problématique sur la reconnaissance d'activités humaines [6].

Le choix de l'architecture d'un réseau de neurones peut être vu comme un problème combinatoire, le but étant de choisir la combinaison d'hyperparamètres (Nombre de couches, taille des couches,etc...) qui offre les meilleurs performances. Dans notre travail, nous proposons d'exploiter la puissance des métaheuristiques, des méthodes qui ont déjà fait leurs preuves dans la résolution de ce type de problème.

Le reste du document est organisé comme suit : la Section 2 introduit une synthèse des différents travaux menés dans le domaine de la reconnaissance d'activité humaine, basée sur l'utilisation des capteurs. Les Section 3 et Section 4 introduisent respectivement l'approche choisie pour la mise en place de l'architecture de notre modèle d'apprentissage profond, ainsi que son implémentation. Dans la Section 5 nous proposerons une nouvelle représentation des signaux de capteurs. Dans la Section 6, nous présenterons différents *data sets* qui seront utilisés pour la partie expérimentale du travail, présentée dans la Section 7. Le résultat obtenu sera exploité dans un cadre applicatif Section 8 pour permettre la reconnaissance de l'activité humaine en temps réel à partir de données smartphone. Enfin, nous présenterons le bilan de ce travail ainsi que les évolutions possibles pour améliorer les taux de reconnaissance d'activités humaines obetnus actuellement Section 9.

2 État de l'art

La reconnaissance de l'activité humaine basée sur l'utilisation des capteurs, peut être traitée comme un problème de de classification de séries chronologiques (ou temporelle), multi-variables représentant les données de capteurs [8], ce pro-

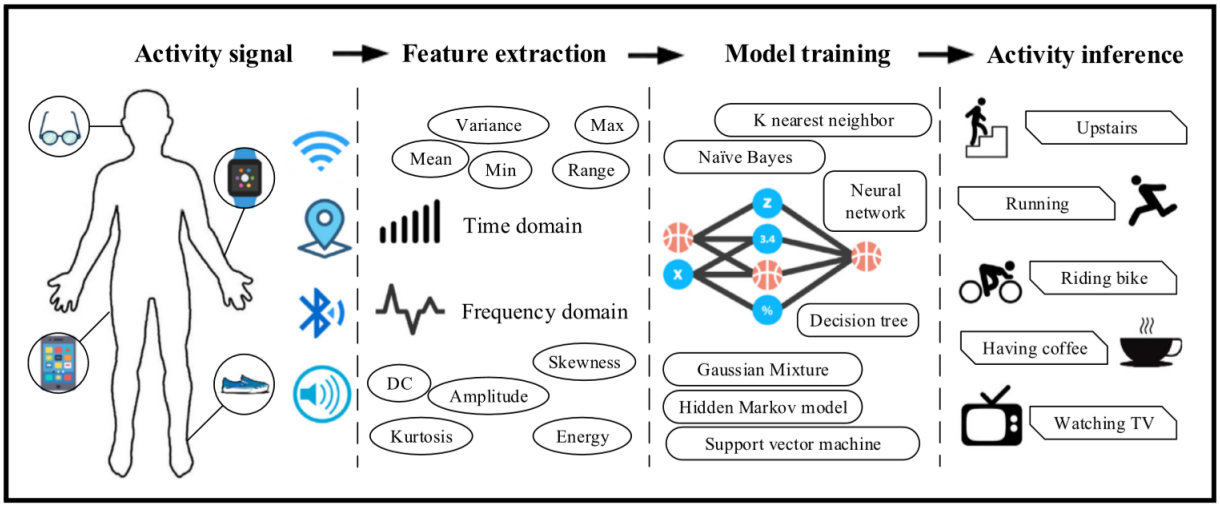

FIGURE 1 – Illustration du processus de la reconnaissance de l'activité humaine basée capteurs [8]

cessus est illustré dans la figure 1. Les différentes approches de reconnaissances de l'activité humaine sont spécifiques à un certain type de capteurs, bien qu'il soit possible de les généraliser à n'importe quel type. Les capteurs sont principalement répartis en 3 types : portés (body-worn), objets et ambiants. Les tableaux 1 illustre le type d'évènement reconnu par chaque type de capteurs, ainsi que les avantages et les inconvénients de chaque type. Cependant, selon [8], il existerait un quatrième type, représentant les capteurs hybrides, où des capteurs de types différentes (portés et ambiants par exemple), sont utilisés conjointement pour la reconnaissance de l'activité humaine, étant donné l'apport possible en terme d'informations de cette hybridation.

D'énormes progrès ont été faits dans le domaine du HAR, en adoptant les méthodes d'apprentissage automatique tel que les arbres de décision, SVM, classifieur bayésien ainsi que les HMM [29] cité dans [8] ou encore les réseaux de neurones [32] cité dans [6]. Généralement, ces méthodes reposent sur une heuristique dans le but d'extraire les caractéristiques à exploiter lors de la phase d'apprentissage qui, de ce fait est limitée par la connaissance humaine du domaine [24] cité dans [8] et que seules les caractéristiques les moins complexes peuvent être apprises, ce qui limite les performances de ces méthodes [34] cité dans [8].

L'apprentissage profond est défini par Bengion [23] cité dans [7] comme étant une technique d'apprentissage automatique qui permet de découvrir automatiquement la représentation des caractéristiques à partir de données brutes. Ils ont connu plusieurs utilisation dans plusieurs domaine à l'exemple de la reconnaissance de la parole ainsi que la reconnaissance d'image [28, 33] cités dans [7].

Les méthodes d'apprentissage profond peuvent être répartie dans cinq catégories qui sont : Machine de Boltzmann restreinte (RBM), *Sparse Coding*,*Deep Auto-encoder*, Réseau de neurones convolutif (CNN) et Réseau de neurones récur-

Type de capteur	Types d'évènement	Avantages	Inconvénients
Portés	Mouvement du corps	Déploiement facile	
Objets	Mouvement des objets	Informations détaillés	Déploiement difficile
Ambiants	Changements dans l'environnement		Déploiement difficile Facilement affecté par l'environnement Seules certaines activités peuvent être reconnues avec certitude

TABLE 1 – Récapitulatif sur les différents types de capteurs

rent (RNN) [7]. Ces catégories sont elles-mêmes répartie selon trois classes différentes :

1. Générative : Exploite la probabilité jointe entre une entrée X et une sortie (classe) Y pour effectuer la classification.
2. Discriminative : Exploite la probabilité conditionnelle P(Y|X), où X représente l'entrée et Y la sortie, pour effectuer la classification.
3. Hybride : Combinaison entre plusieurs méthodes (génératives et discriminatives).

L'apprentissage profond connaît plusieurs utilisation dans le domaine de la reconnaissance de l'activité humaine, sa première utilisation pour l'extraction automatique des caractéristiques remonte à 2011 [31] cité dans [6], dont le quel une machine de Boltzmann restreinte fût utilisée. La figure 2 résume la classification des différentes méthodes d'apprentissage profond utilisées pour la reconnaissance de l'activité humaine. Chacune d'entre elles présente des atouts ainsi que des lacunes, comme l'illustre la tableau 2.

L'hybridation entre les méthodes d'apprentissage profond connaît plusieurs utilisations dans la littérature. D'après de nombreuses études, les réseaux de neurones convolutifs, qui sont généralement utilisés pour l'extraction des caractéristiques, semblent être la meilleure méthode à combiner avec une autre méthode, qu'elle soit générative ou discriminative [7].

Il existe plusieurs hybridations appliquées à la reconnaissance de l'activité humaine, à l'exemple du réseau de neurones convolutif et Sparse Coding, réseau de neurones convolutif et Denoising Autoencoder ainsi qu'un réseau de neurones convolutif et une machine de Boltzmann restreinte [7]. Une autre hybridation, entre un réseau de neurones convolutif et un LSTM (DeepConvLSTM) a été proposé par Ordóñez et al.[12], dans le but de tirer parti des avantages qu'offre chacune des méthodes, à savoir une extraction automatique des caractéristiques et la modélisation des dépendances temporelles qui existent entre les données de capteurs.

Suite à l'étude bibliographique menée précédemment, une hybridation entre des réseaux de neurones convolutifs, dont les mécanismes d'extraction automatique de caractéristiques résistent aux petites variations dans les données des capteurs, et des réseaux de neurones récurrents, à l'instar du *Long short term memory*

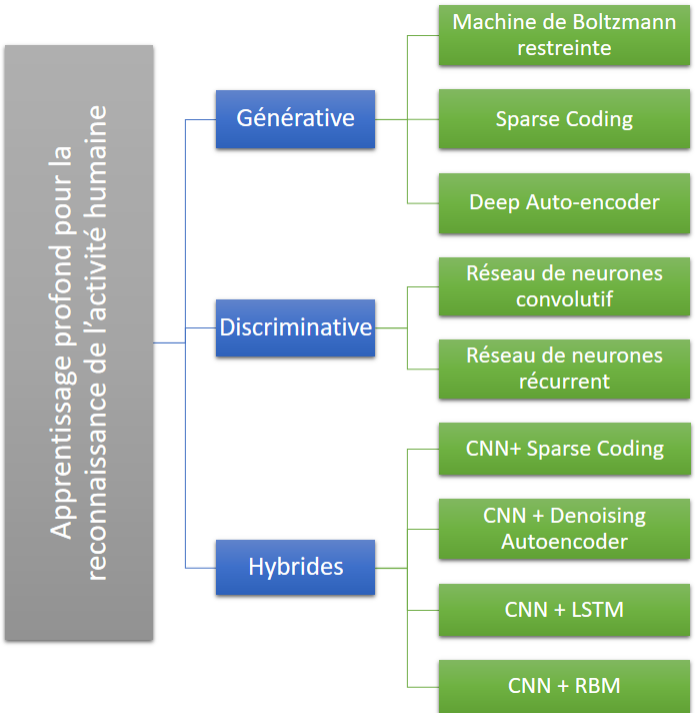

FIGURE 2 – Classification des méthodes d'apprentissages automatique appliquées à la reconnaissance de l'activité humaine selon [7]

(LSTM) qui permet d'exploiter les dépendances temporelles entre les séries chronologiques des différentes données de capteurs, nous paraît être la meilleure solution à mettre en œvre pour pour améliorer les performances de la reconnaissance d'activité humaine à partir de données issues de capteurs portés et ambiants qui sont les capteurs les plus courants pour l'analyse d'activités quotidiennes dans le cadre de l'assistance de vie dans un environnement connecté.

3 Métaheuristiques

La reconnaissance de l'activité humaine à partir de ce type de capteurs est un problème de classification, où le choix du classifieur à utiliser ainsi que son architecture représente une étape cruciale pour la reconnaissance de l'activité humaine [31] cité dans [6]. Ce choix peut être formulé comme étant un problème combinatoire, où la solution à ce problème, serait la combinaison des hyperparamètres donnant la meilleure classification (reconnaissance).

Il existe plusieurs travaux qui exploitent ces méthodes, dans le but de choisir les hyperpara-

Catégorie	Avantages	Inconvénient
Machine de Boltzmann restreinte	Apprentissage non-supervisé sur des flux de capteurs	Nécessite une puissance de calcul supérieur à celles des smartphones, donc difficile à embarquer sur ces derniers.
Auto-encoder	Denoising : Robuste dans le cas de données corrompues Sparse : produit des caractéristiques plus distinctes les unes des autres Contractive : Rend les caractéristiques invariantes aux changements en réduisant les dimensions de leurs espaces.	Denosing et sparse : Temps de calculs élevé Contractive : difficile à optimiser
Spare Coding	Minimise la complexité de calcul	La manipulation et calcul de vecteurs de caractéristiques ne sont pas évidents. Difficile de développer des architectures profondes
Réseau de neurones convolutif	Extraction automatique des caractéristiques Invariant à l'orientation des données des capteurs et les changements dans les détails d'une activité	Nécessite un grand ensemble de données et le paramétrage des hyperparamètres pour atteindre l'optimum. Peut être difficile à embarquer.
Réseau de neurones récurrent	Utiliser pour modéliser les dépendances temporelles qui existent entre les données	Apprentissage difficile liés à la descente du gradient (explosion et disparition) ainsi que la mise à jours des différents paramètres (cas du LSTM).

TABLE 2 – Récapitulatif sur les avantages et les inconvénients de chaque catégorie d'apprentissage profond

mètres qui serait les plus intéressant pour un réseau de neurones, à l'exemple de [18, 2, 19, 4, 13]

Ces méthodes auront pour objectif de déterminer l'architecture la plus intéressante après un certain nombre d'itérations lors de l'apprentissage, avant ré-effectuer l'apprentissage de cette dernière sur un nombre plus d'itérations plus important.

Les métaheuristiques sont des méthodes de recherche inspirées des systèmes naturels tels que les colonies de fourmis, évolution Génétique... etc [16]. Elles se basent sur deux principes qui sont :

- L'intensification qui consiste à chercher des solutions de meilleure qualité en se basant sur les solutions déjà trouvées.
- La diversification qui est une stratégie qui permet l'exploration de nouvelles zones de recherche dans l'espace des solutions, et donc, échapper aux éventuels blocages dans des optimums locaux

Un mauvais équilibre entre ces deux principes peut conduire à une convergence trop rapide vers des minima locaux (manque de diversification), ou à un temps d'exécution trop long dû à l'exploration trop vaste de l'espace de recherche (manque d'intensification).

Dans le but de choisir l'architecture de notre réseau hybride, composé d'une partie réseau de neurones convolutifs (CNN) et une partie réseau de neurones récurrents (LSTM), nous avons fait le choix d'utiliser trois métaheuristiques :

1. Algorithme Génétique
2. Algorithme FireFly
3. Hybridation Génétique/FireFly

Notre choix d'utiliser l'algorithme génétique, revient au fait qu'il soit une métaheuristique très populaire dans la littérature, ayant déjà fait ses preuves dans la résolution de nombreux problèmes d'optimisation. Quant aux choix de l'algorithme Firefly, ce dernier revient au fait que cette méthode regroupe plusieurs métaheuristiques. En effet, en plus de sa convergence rapide vers un optimum, l'algorithme FireFly englobe plusieurs autres métaheuristiques à l'exemple du recuit simulé (SA), ou encore de l'optimisation par essaims particulaires (PSO) [34]. Quant à l'hybridation entre l'algorithme génétique et l'algorithme FireFly, nous avons voulu tester cette méthode, proposée par Wahid et al. [20] pour la configuration d'un réseau de neurones, sachant que cette dernière a donné d'excellents résultats appliquée à un problème multi-objectifs, portant sur la minimisation du coût énergétique tout en maximisant le confort des occupants d'un bâtiment résidentiel.

3.1 Algorithme Génétique (GA)

Introduits par J.Holland en 1975 [16], les algorithmes génétiques s'appuient sur trois principes qui sont :

1. **La sélection :** Consiste à sélectionner deux parents dans le but d'effectuer un croisement entre eux.
2. **Le croisement :** Consiste à combiner deux solutions parents afin de former une ou deux solutions enfants.
3. **La mutation :** Permet d'introduire une diversification dans la recherche, en mutant certaines caractéristiques d'une solution.

Il existe plusieurs versions de l'algorithme génétique, dans notre cas, on s'intéresse à la version à remplacement de populations dont l'algorithme est donné dans l'algorithme 1 .

Le fonctionnement de l'algorithme Génétique est illustré par l'algorithme 1

L'algorithme génétique a été exploité par Sun et al. dans [19], dans le but de choisir l'architecture d'un réseau de neurones convolutif appliqué à la classification d'images.

3.2 Algorithme FireFly (FA)

Basé sur le comportement des lucioles (« firefly » en anglais), cet algorithme a été développé par Xin-She Yang en 2018 à l'université de Cambridge [34] . Il repose sur trois règles qui sont :

Algorithm 1 Algorithme Génétique

Générer aléatoirement une population P de n solutions
repeat
  P' ← ∅
  repeat
    Sélection de 2 solutions x et x' de P
    Croisement entre les deux parents x et x' pour former une solution enfants y
    Faire muter y sous certaines conditions
    Ajouter y dans P'
  until |P'|=n
  P= P'
until satisfaction des critères d'arrêt

1. Les lucioles sont unisexes, de ce fait, une luciole est attirée par une autre sans prendre en considération son genre.
2. L'attractivité est proportionnelle à la brillance, et toutes les deux sont inversement proportionnelles à la distance (diminuent lorsque la distance augmente). Pour n'importe quel couple de lucioles, la moins brillante se verra attirer par la plus brillante, et donc se déplacera vers elle, dans le cas où il n'existe pas de luciole plus brillante qu'elle, cette dernière se déplacera aléatoirement.
3. La brillance d'une luciole (solution) est donnée par la fonction objectif.

Comme l'attractivité d'une luciole est proportionnelle à l'intensité de la lumière perçue par les lucioles adjacentes, il est possible de définir la variation de l'attractivité β avec la distance r comme suit :

$$\beta = \beta_0 e^{-\gamma r^2} \quad (1)$$

Avec

- β_0 : L'attractivité à r = 0.
- γ : coefficient d'absorption, (paramètre donné en entrée).
- r : distance cartésienne entre deux lucioles

Le mouvement d'une luciole i, vers une luciole plus brillante j, est défini comme suit :

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \tau_i^t \quad (2)$$

Avec

- α : facteur aléatoire
- x_i^t : Position de la luciole i à l'instant t
- τ_i^t : Vecteur de nombre aléatoires donnés par une distribution Gaussienne à l'instant t

Pour ce qui est de la formule précédente, le second terme représente l'attraction entre la luciole i et j, quant au troisième, il sert à introduire un facteur aléatoire au déplacement, donc, si β_0=0 , alors le déplacement se fera de manière totalement aléatoire, et inversement, si γ=0 alors l'algorithme se comportera comme une variante de PSO.

Le paramètre α est responsable de l'importance donnée au facteur aléatoire, et de ce fait, sert à gérer la diversification durant la recherche.

Le fonctionnement de l'algorithme FireFly est illustré par l'algorithme 2

Algorithm 2 Algorithme Firefly

Générer aléatoirement une population de n lucioles x_i=(i=1,2,...,n).
L'intensité de la lumière I_i de chaque luciole x_i est donné par f(x_i) où f est la fonction objectif
Définir les coefficients α, β et γ.
while (tMaxGénération) **do**
  for i=1 à n **do**
    for j=1 à n **do**
      if $I_i < I_j$ **then**
        Déplacer la luciole i vers la luciole j
      end if
      Faire varier l'attractivité selon la distance r.
      Evaluer la nouvelle solution et mettre à jour l'intensité de la lumière
    end for
  end for
  Déterminer la meilleure solution ;
end while

L'algorithme firefly a été exploité par Strumberger et al. dans [18], dans le but de choisir l'architecture d'un CNN appliqué à la classification des images du *data-set* MNIST.

3.3 Hybridation Génétique/FireFly

Il existe une hybridation entre l'algorithme FireFly et l'algorithme génétique, introduit par Wahid et al. [20]. Cette approche propose d'utiliser les opérateurs de l'algorithme génétique, à savoir, la sélection, le croisement et la mutation, dans l'exécution de l'algorithme FireFly, dans le but de potentiellement améliorer la qualité des solutions.

Le fonctionnement de l'algorithme FireFly consiste à faire une convergence (rapprochement), entre les différentes solutions composants la population, grâce à son mécanisme d'attraction [9].Exploiter les opérateurs de l'algorithme génétique pourrait être perçu comme un autre mécanisme de diversification, qui permettrait de mieux explorer l'espace de recherche, en combinant les composantes des solutions déjà explorées entre elles.

Wahid et al. proposent d'exécuter une itération de l'algorithme génétique à chaque fois que l'algorithme FireFly ne parvient pas à améliorer son résultat [20]. Nous proposons d'opérer un petit changement à leur approche, en rajoutant des chances aux exécutions de l'algorithme FireFly, car ce dernier est déjà doté d'un mécanisme de diversification (déplacement aléatoire). Le doter d'un certain nombre de chances lui permettre d'opérer une diversification "mineure", ce qui permettrait de mieux explorer une région de l'espace de recherche, avant d'en explorer une autre éloignée (obtenue après l'algorithme génétique). L'algorithme 3 illustre l'approche proposée.

Algorithm 3 Algorithme Génétique/FireFly

Générer aléatoirement la population.
Définir MaxChances
chances = MaxChances
while (tMaxGénération) **do**
  Exécuter une itération de FireFly
  if $Best_t > Best_{t-1}$ **then**
    chances=MaxChances
  else
    chances–
  end if
  if chances=0 **then**
    Exécuter une itération du génétique
    chances = MaxChances
  end if
  Déterminer la meilleure solution ;
end while

4 Implémentation

Dans cette section, nous allons décrivons la représentation des solutions manipulées par les métaheuristiques ainsi que l'évaluation de leurs performances.

4.1 Représentation

Dans ce travail, une solution représente une combinaison d'hyperparamètres d'un modèle d'apprentissage profond, hybride, composé d'un ensemble de couches de convolutions (Partie CNN), suivi d'un ensemble de couches de réseau de neurones récurrents (LSTM) et d'une couche de sortie de type Dense utilisant *softmax* comme fonction d'activation dans le but de réaliser la classification. Nous avons fait le choix de représenter une solution par un tableau, en prenant en compte les couches cachées du réseau uniquement, car la couches d'entrée et de sortie seront identique pour toutes les architectures, cette représentation est illustrée dans le figure 3. Chaque case du tableau contient des informations relatives à la couche lui correspondant (type de couche (Convolution,Pooling, LSTM, Dense), taille de la couche, taille du filtre dans le cas des couches de convolutions ou de pooling).

Dans le cas des couches de pooling, nous avons fait le choix d'utiliser le max pooling, en se basant sur des études, qui démontrent que ce dernier dépasse les autres types en termes de performances[7]. La fonction d'activation des couches de convolutions a été fixé à une fonction linéaire($f(x) = x$) ,tandis que dans le cas des couches LSTM, c'est la fonction *tanh* qui a été utilisée.

Le tableau 3 regroupe les hyperparamètres qui sont considérés durant la recherche, ainsi que les différentes valeurs qu'ils peuvent prendre.

Paramètres	Valeurs possibles
Nombre de couches CNN	[1 ;10]
Nombre de couches récurrentes	[1 ;5]
Taille des filtres	[3 ;13]
Taille des couches	[10 ;100]

TABLE 3 – Hyperparamètres considérés durant la recherche

4.2 Évaluation

Afin d'exploiter les métaheuristiques, il est nécessaire d'établir un critère d'évaluation des solutions. Étant donné que les solutions sont effectuent leurs apprentissages sur un nombre d'itérations relativement petit, nous avons besoin de mettre en place une fonction d'évaluation qui permet de déterminer les architectures qui donneront les meilleurs résultats une fois réentrainées sur un nombre d'itérations plus important.

Nous avons un ensemble de 6 fonctions d'évaluation, qui sont :

1. Précision sur l'ensemble de validation à la dernière itération :

$$f_1(x) = valAccuracy_n \tag{3}$$

2. Somme des différences des précisions sur l'ensemble de validation entre deux itérations successives :

$$f_2(x) = \sum_{i=2}^{n} (valAccuracy_i - valAccuracy_{i-1}) \tag{4}$$

3. Semblable à f_2, mais en rajoutant une pondération :

$$f_3(x) = \sum_{i=2}^{n} i * (valAccuracy_i - valAccuracy_{i-1}) \tag{5}$$

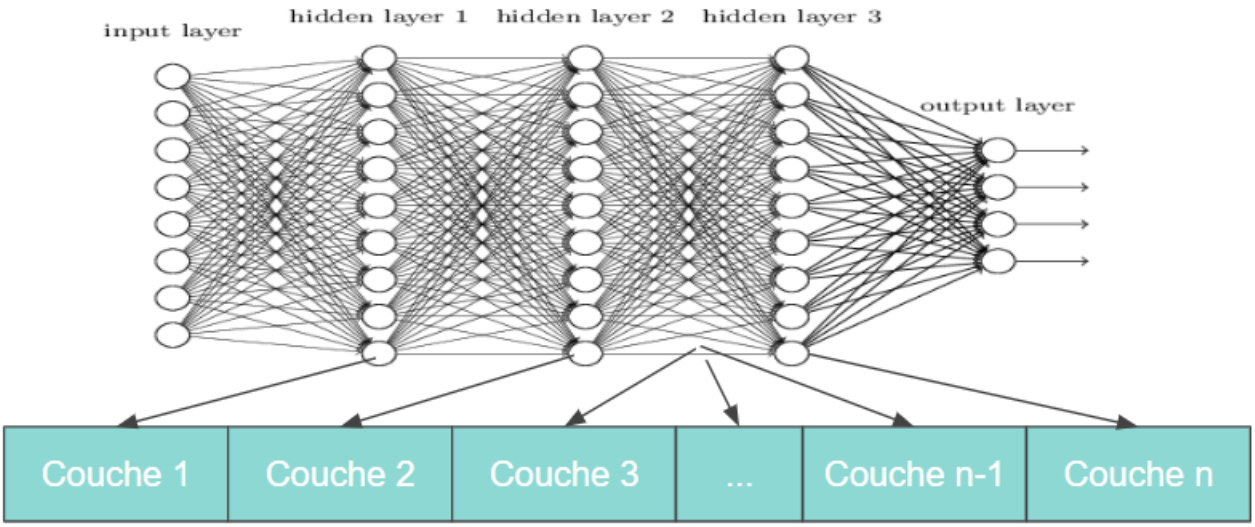

FIGURE 3 – Représentation d'une solution

4. La meilleure valeur d'erreur (minimale) obtenue :

$$f_4(x) = Min(valLoss) \quad (6)$$

5. La meilleure précision sur l'ensemble de validation, pénalisée par sa différence avec la précision sur l'ensemble d'apprentissage à la même itération :

$$f_5(x) = valAccuracy.(1-\alpha.penality) \quad (7)$$

avec :

$$penality = abs(val_{Accuracy} - train_{Accuracy})$$

6. Semblable àf_2, mais en utilisant l'erreur :

$$f_6(x) = valLoss.(1+\alpha.penality) \quad (8)$$

avec :

$$penality = abs(valLoss - trainLoss)$$

les fonctions f_4 et f_6 sont à minimiser,tandis que les autres fonctions sont à maximiser.

4.3 Opérateurs Métaheuristiques

Dans ce qui suit, nous allons détailler l'implémentation des différents opérateurs des métaheuristiques choisies.

4.3.1 Algorithme génétique

l'algorithme génétique repose sur trois opérateurs cité précédemment : Sélection, croisement et mutation.

Pour l'opérateur de sélection, nous avons opté pour une sélection aléatoire des solutions parents à partir de la population, ces solutions sélectionnées serviront à l'opérateur de croisement.

Nous avons choisir d'effectuer un croisement en un seul point, au milieu du tableau, étant

donné que nos solutions se composent d'une partie CNN et d'une partie LSTM, le croisement entre deux solutions se décompose en un croisement entre les parties CNNs et un autre entre les parties LSTMs, comme l'illustre la figure 4. Nous avons choisi d'implémenter ce croisement dans le but d'engendrer toujours des solutions composées d'une partie CNN et d'une autre LSTM.

En ce qui concerne la mutation, nous avons choisi de re-générer aléatoirement les hyperparamètres d'une couches choisie aléatoirement.

4.3.2 Algorithme FireFly

L'algorithme FireFly repose sur un seul opérateur, le déplacement d'une solution. Pour assurer ce déplacement, il est nécessaire d'introduire une notion de distance entre les solutions, pour cela, nous proposons d'utiliser la distance euclidienne pour calculer la distance entre une solution i et une autre j :

$$distance(i,j) = \sqrt{\sum_{k}(d_k(i,j)^2)} \tag{9}$$

avec $d_k(i,j)$: Distance entre la solution i et j à la k-ième couche.

$$d_k(i,j) = \sqrt{d_{size}(i,j)^2 + d_{type}(i,j)^2 + d_{kernel}(i,j)^2} \tag{10}$$

Dans l'équation 10, $d_{size}(i,j)$ représente la distance entre les tailles des couches à la k-ième position, des solution i et j, quand à $d_{type}(i,j)$ et $d_{kernel}(i,j)$, ils représentent respectivement la distance entre les types et les tailles des filtres des couches à la k-ième position, des solutions i et j.

Lorsque deux solution i et j n'ont pas le même nombre de couche, les composants des couches "manquantes" sont tous supposé à 0, ce qui implique que la distance entre une couche k appartenant a une solution i et une couche inexistante soit égale aux paramètres de la couche k.

Les distances entre les types et les tailles des filtres sont propres à la partie CNNs des solutions, et sont donc supposé à 0 pour les couches LSTM, ces distances sont calculées en utilisant la distance euclidienne.

La notion de type est propre aux couches CNNs, à la génération d'une solution, la couche se voit attribué un entier, représentant sont type, 0 pour les couches de convolution, et 1 pour les couches de pooling, lors du déplacmenet d'une solution vers une autre, cet entier est mis à jour, permettant le passage entre les deux types (passage d'une couche de convolution vers une couche de pooling et inversement), le choix du type de la couche est fait en comparant l'entier à 0.5, s'il est inférieur ou égale, le type de la couche est convolutif, pooling sinon.

5 Transformation des signaux capteurs

Dans cette partie du travail, nous proposons de représenter les séries chronologiques propres à une activité humaine, sous la forme d'images, dans le but de transformer la dépendances temporelle en dépendance spatiale. De ce fait, la reconnaissance de l'activité humaine passe d'un problème de classification de séries chronologique, en un problème de classification d'image, et donc, il serait possible de n'utiliser qu'un réseau de neurones convolutif, un type de réseau qui a fait ces preuves à plusieurs reprises pour ce genres de problèmes.

La représentation d'une série chronologique sous la forme d'image, n'est pas une innovation en son genre, cela fût déjà proposé par Hur et al. dans [10], pour la reconnaissance de l'acti-

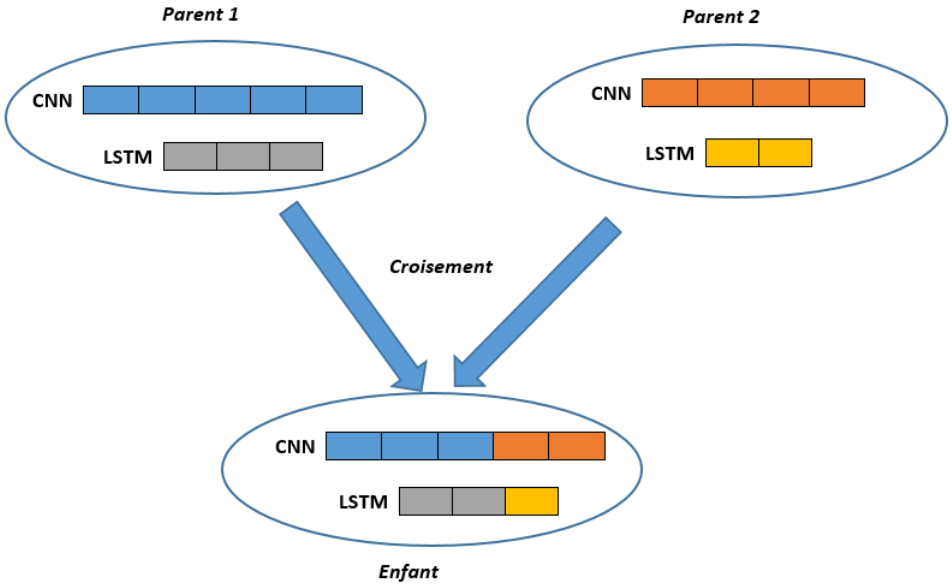

FIGURE 4 – Illustration du croisement entre deux solutions

vité humaine. La transformation du signal (données d'un capteur accéléromètre) était faite en appliquant un filtre, permettant de transformer les accélérations selon les trois axes, en valeurs RVB représentants les pixels de l'images, comme illustré dans la figure 5.

Nous proposons d'utiliser une approche différente, où les séries chronologiques sont transformées en courbes, représentant ainsi, une signature de l'activité. La courbe est construite en reliant les points correspondants à chaque instant t de l'entrée, initialement, on commence à l'origine (0,0,0), puis à chaque instant, l'accélération sur chaque axe est ajoutée à la composante lui correspondant du point précédent, comme illustré dans l'équation 11.

$$\begin{aligned} x_t &= x_{t-1} + a_{xt} \\ y_t &= y_{t-1} + a_{yt} \\ z_t &= z_{t-1} + a_{zt} \end{aligned} \tag{11}$$

Où a_{xi} représente l'accélération selon l'axe x à l'instant t. Une fois tout les points reliés, une courbe en 3D est obtenues, à l'exemple de la figure 6, représentant la signature 3D de l'activité «Se lever », tirée du data set Human Activity Recognition using smartphone.

En suite, dans le but de transformer les signatures 3D obtenues, en images en deux dimensions, nous faisons deux propositions, la première consiste à projeter les point de la courbes, selon les trois plans, obtenant ainsi 3 images par activités, ces images pourront ensuite être jointes (côte à côte par exemple) dans le but d'en former une seule, qui sera utilisée par le modèle d'apprentissage profond. La seconde proposition, consiste à projeter la courbe sur un seul plan, et utiliser une variation de couleur, pour exprimer le déplacement selon le troisième axe, comme l'illustre la figure 7.

L'utilisation de cette représentation devrait

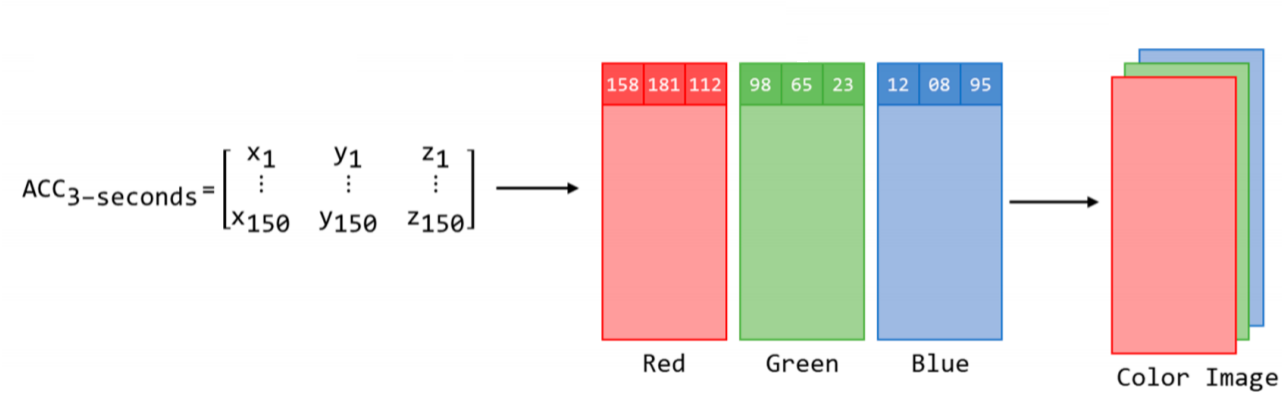

FIGURE 5 – Proposition de Hur et al. pour la représentation d'un signal de capteur [10]

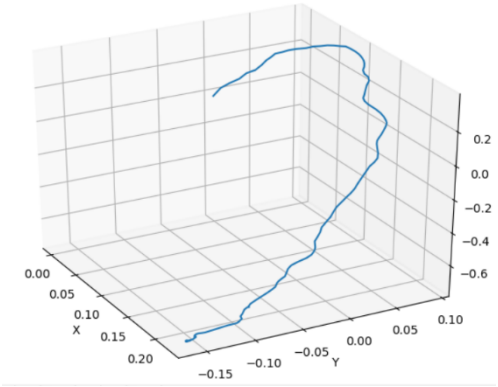

FIGURE 6 – Signature 3D de l'activité «Se lever » tirée du data set HAR using smartphones

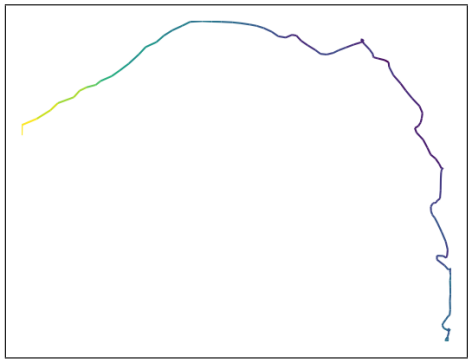

FIGURE 7 – Signature de la figure 6 représentée en une image en couleurs

permettre d'exploiter un CNN entraîné sur des données d'une certaine dimensions (série chronologique de taille n_1), pour la classification de données ayant une autre dimensions (série de taille n_2), chose qui n'est pas possible en utilisant des séries chronologiques représentées sous la forme de vecteurs. Cette avantage peut s'avérer très bénéfique dans le cadre d'une application en temps réel, où il est très difficile de déterminer le début et la fin d'une activité, à partir d'un flux de données continu.

6 Benchmark

L'évaluation des performances des différentes méthodes d'apprentissage automatique ou profond, ainsi que la comparaison entre elles, né-

cessite un ensemble de données d'essai, appelé *Benchmark*.

Dans ce qui suit, nous allons introduire les ensembles de données (*data set*), que nous avons utilisé dans le cadre des différentes expérimentations effectuées.

6.1 OPPORTUNITY

OPPORTUNITY est un ensemble de données collectées par Roggen et al., en utilisant un ensemble de capteurs portés combinés avec des capteurs de type ambiant et objet, placés respectivement dans l'environnement ainsi que sur les objets qui s'y trouvent, dans le but de fournir une description détaillé des activités de vie quotidiennes enregistrées [14].

Ce *data set* contient plusieurs annotations des activités recensées [12], nous nous intéresserons dans cette description de OPPORTUNITY à deux catégories citées dans [12] qui sont des activités de déplacement ainsi que des gestes.

Les gestes ainsi que les activités de déplacement qui sont recensés dans OPPORTUNITY sont enumérés dans le tableau 4 [12]

6.2 le *data set Human Activity Recognition using Smartphones*

Les données du *Human Activity Recognition using Smartphones data set* ont été récoltées par Anguita et al. [1] sur 30 individus, en utilisant des smartphones comme capteurs.

Les activités recensées dans cet ensemble de données sont : «Marcher, monter des escaliers, descendre des escaliers, s'asseoir, rester debout et s'allonger»[1].

Activités de déplacement	Gestes
1. Rester debout 2. Marcher 3. S'assoir 4. S'allonger	1. Ouvrir et fermer le réfrigérateur 2. Ouvrir et fermer le Lave-vaisselle 3. Ouvrir et fermer 3 tiroirs différents 4. Ouvrir et fermer la porte 1 5. Ouvrir et fermer la porte 2 6. Allumer et éteindre les lumières 7. Nettoyer la table 8. Boire (debout) 9. Boire (Assis)

TABLE 4 – Gestes et activités de déplacement recensés dans le *data set* OPPORTUNITY

6.3 le *data set* WISDM

L'ensemble de données WISDM a été recollé par Kwapisz et al., en utilisant des capteurs portés (smartphones et montres connectées) par 29 volontaires, durant l'accomplissement de leurs différentes activités quotidiennes [11].

Le *data set* traite cinq activités différentes qui sont : «Marcher, courir, monter ou descendre des escaliers, s'assoir, rester debout »

Le choix de ces activités revient au fait qu'elles soient effectuées régulièrement, par plusieurs personnes [11].

7 Expérimentation et résultats

Dans cette partie du travail, nous allons dans un premier temps introduire l'environnement de travail utilisé pour l'expérimentation. En suite, exposer les résultats obtenus par les tests effectués pour le paramétrage de la proposition. Enfin, nous comparerons les résultats obtenus avec d'autres travaux existants en utilisant les *data-sets* introduits précédemment.

7.1 Environnement de travail

L'implémentation de notre proposition a été faite sous Python 3, en utilisant la version 2.3.1 de Keras avec un backend Tensorflow-gpu, version 1.15.

L'ensemble des expérimentations ont été menées sur *Google Colaboratory* (GPU Tesla K80).

7.2 Paramétrage de la proposition

L'ensemble des tests conduits dans le but de paramétrer notre proposition, ont été effectués sur l'ensemble de données *Human Activity Recognition using Smartphones data set.*

7.2.1 Fonction d'évaluation d'une solution

Afin de pouvoir choisir la fonction d'évaluation la plus adaptée, parmi l'ensemble des fonctions proposées précédemment, un ensemble de tests a été mené, utilisant des modèles générés aléatoirement, 100 pour chacun des types suivants : Réseaux de neurones convolutifs (CNN), réseaux de neurones convolutifs suivi d'un perceptron multicouche (CNNMLP) et réseaux hybrides composés de couches convolutifs et de couches récurrentes (CNNLSTM).

L'évaluation s'effectue à la cinquantième itérations de l'apprentissage du modèle, puis, ce dernier poursuit son apprentissage jusqu'à 1000 itérations, avant d'être testé sur l'ensemble de test du *data set*. Dans les cas des fonctions f_5 et f_6, différentes valeurs du paramètre α (0.1 à 0.9 avec un pas de 0.1) ont été testées. L'optimiseur utilisé est « rmsprop »avec un taux d'apprentissage de 0.001

Pour pouvoir comparer entre les fonctions d'évaluation, les solutions sont d'abord triées par ordre décroissant de performance (la meilleure obtient le rang 1,etc...) selon les score obtenus par chacune des fonctions, puis, la même chose est faite en utilisant les scores obtenus sur l'ensemble de test. En suit, une moyenne des erreurs au carré (MSE) est calculée pour chaque fonction, en prenant en considération la différence entre les rangs obtenus par les fonctions (prédictions) et les rangs obtenus sur l'ensemble de test (valeurs réelles). La fonction qui minimise à la MSE serait la fonction qui prédit au mieux la qualité d'une solution une fois entraînée sur un plus grand nombre d'itérations. Les résultats obtenus par ces différents tests sont illustrés dans le tableau 5.

D'après les résultats illustrés dans le tableau 5, on peut remarquer que la fonction d'évaluation f_5 offre les meilleurs résultats sur les 3 types de réseaux testés. Le paramètre α offrant les meilleurs performances varie d'un type de réseau à un autre, afin de pouvoir trancher, on se réfère la moyenne des MSEs, qui est minimisée par $\alpha = 0.3$.

Dans la suite des tests, la fonction d'évaluation à utiliser sera f_5 avec $\alpha = 0.3$

fonction d'évaluation	MSE sur CNN	MSE sur CNNMLP	MSE sur CNNLSTM	Moyenne MSE
f_1	615.2	706.48	774.9	698.86
f_2	2191.56	2144.54	1910.32	2082.14
f_3	1740.54	1645.2	1731.74	1705.82667
f_4	336.58	507.98	628.6	491.0533
$f_5(\alpha = 0.1)$	314.38	**398.08**	616.18	442.88
$f_5(\alpha = 0.2)$	305.26	410.38	614.54	443.39333
$f_5(\alpha = 0.3)$	301.22	408.38	**610.38**	**439.99333**
$f_5(\alpha = 0.4)$	**298.34**	427.02	610.9	445.42
$f_5(\alpha = 0.5)$	301.48	445.72	612.82	453.34
$f_5(\alpha = 0.6)$	304.6	465.82	614.76	461.72667
$f_5(\alpha = 0.7)$	310.24	484.16	621.04	471.81333
$f_5(\alpha = 0.8)$	311.16	507.32	633.3	483.92667
$f_5(\alpha = 0.9)$	322.	522.66	645.16	496.60667
$f_6(\alpha = 0.1)$	336.92	512.08	631.06	493.35333
$f_6(\alpha = 0.2)$	338.66	512.64	631.16	494.15333
$f_6(\alpha = 0.3)$	334.76	516.16	632.5	494.47333
$f_6(\alpha = 0.4)$	336.16	517.82	626.3	493.42667
$f_6(\alpha = 0.5)$	337.94	515.7	623.6	492.41333
$f_6(\alpha = 0.6)$	341.16	522.16	624.12	495.81333
$f_6(\alpha = 0.7)$	345.74	527.36	622.14	498.41333
$f_6(\alpha = 0.8)$	349.98	530.9	622.38	501.08667
$f_6(\alpha = 0.9)$	350.36	536.74	623.58	503.56

TABLE 5 – MSE obtenues par les différentes fonctions testées

7.2.2 Nombre d'itération pour l'évaluation d'une solution

Dans le but de choisir le nombre d'itérations adéquat à l'évaluation d'une solution, le procédé précédent (en utilisant les MSEs) est réutilisé, en faisant varier le nombre d'itération cette fois ci, de 10 à 100, avec un pas de 10. Les résultats obtenus sont illustrés dans le tableau 5.

D'après les résultats obtenus, on peut remarquer que dans deux des trois cas (CNN et CNNLSTM), le nombre d'itérations le plus intéressant était de 50, tandis que dans le cas du CNNMLP, le meilleur résultat été obtenu à 90 itérations. Une fois encore, on se réfère à la moyenne des MSEs pour pouvoir trancher, ce qui nous permet de dire que, 50 itérations semble être le meilleur choix.

7.2.3 Algorithme firefly

Pour le choix de la taille de la population a utiliser pour l'algorithme firefly, plusieurs tests ont été effectués en variant ce paramètre de 10 à 40 avec un pas de 10, avec 30 itérations comme condition d'arrêt de la recherche. Quand

Nombre d'itérations	MSE sur CNN	MSE sur CNNMLP	MSE sur CNNLSTM	Moyenne MSE
10	484.92000	804.08000	1093.98000	794.32667
20	366.66000	705.80000	1077.46000	716.64000
30	360.10000	625.28000	962.78000	649.38667
40	429.34000	508.44000	820.04000	585.94000
50	**423.50000**	494.40000	**816.12000**	**578.00667**
60	479.34000	467.06000	831.70000	592.70000
70	523.64000	458.38000	902.66000	628.22667
80	555.40000	436.76000	850.66000	614.27333
90	618.64000	**436.26000**	886.44000	647.11333
100	660.62000	447.14000	900.60000	669.45333

TABLE 6 – MSE obtenues par les différents nombre d'itérations testées

aux autres paramètres, nous avons choisi de reprendre les même valeurs utilisé par Strumberger et al dans [18], à savoir $\alpha = 0.5$,$\beta = 0.2$ et $\gamma = 1.0$. Les résultats obtenu sont illustrés dans la tableau 7.

Taille de la population	Fitness obtenu
10	**0.9696699**
20	0.9645878
30	0.9577112
40	0.9668973

TABLE 7 – Résultats paramétrage de l'algorithme FireFly

D'après les résultats obtenus, on peut remarquer que la meilleure performance est donnée par une population de 10 solutions, même si les résultats obtenus par les autres valeurs ne sont pas très loin, à l'exemple de la population à 40 solutions, dont le score est à seulement 0.0028 derrière.

7.2.4 Algorithme génétique

De même que pour l'algorithme FireFly, le choix de la taille de la population à utiliser pour l'algorithme génétique a été fait par expérimentation, 30 itérations de l'algorithme pour une taille de population variant entre 10 et 40, avec un pas de 10 et une probabilité de mutation fixée à 0.3. Les résultats obtenus sont illustrés dans le tableau 8

Taille de la population	Fitness obtenu
10	0.9631918
20	**0.9750055**
30	0.9718211
40	0.9720857

TABLE 8 – Résultats paramétrage de l'algorithme génétique

D'après les résultats obtenus, on peut remarquer que la meilleure performance est donnée par une population de 20 solutions dans le cas de l'algorithme génétique, même si les résultats obtenus par les autres valeurs ne sont pas très loin, à l'exemple des populations à 30 et 40 solutions,

dont les scores sont à 0.003 seulement.

7.2.5 Hybridation génétique Firefly

Le paramétrage de l'hybridation entre l'algorithme firefly et l'algorithme génétique a été effectué de la même manière que les deux métaheuristiques précédente, en reprenant leurs paramètres, à savoir $\alpha = 0.5$,$\beta = 0.2$ et $\gamma = 1.0$ et une probabilité de mutation fixée à 0.3.

Dans le but de fixer le nombre de chance maximale avant l'exécution d'une itération de l'algorithme génétique, nous nous sommes basée sur le meilleure résultat obtenu par les tests effectués sur l'algorithme firefly (population de 10 solutions). En visualisant l'évolution de la meilleure solution en fonction du nombre d'itérations, illustré dans la figure 8, on peut remarquer que l'algorithme FireFly parvient à améliorer sa meilleure solution à la 6-ième et à la 10-ième itérations, avant de stagner sur ce résultat jusqu'à la fin de la recherche.

D'après cette visualisation, on peut remarquer que le nombre d'itération maximale avant une nouvelle amélioration est de 5 itérations, en se basant sur ce constat, nous avons choisi de fixer le nombre de chances de l'hybridation à 5 chances avant l'exécution de l'algorithme génétique. Les résultats obtenus sont illustrés dans le tableau 9

Taille de la population	Fitness obtenu
10	**0.9706549**
20	0.9702883
30	0.9634615
40	0.9693713

TABLE 9 – Résultats paramétrage de l'hybridation génétique firefly

D'après les résultats obtenus, on peut remarquer que la meilleure performance est donnée par

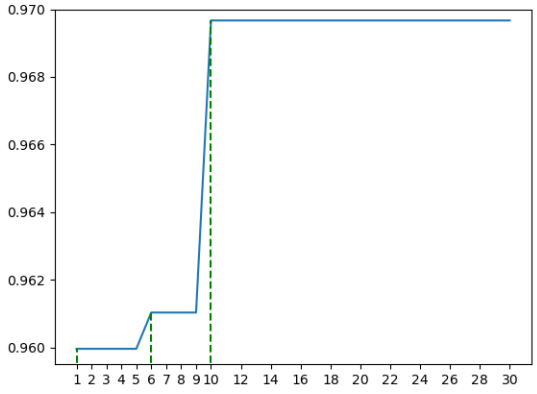

FIGURE 8 – Évolution de la meilleure solution en fonction des itérations pour l'algorithme FireFly avec population de 10 solutions

une population de 10 solutions, on peut aussi noter une amélioration des résultats comparé à ceux obtenu par la version standard de FireFly, ce qui nous confirme que l'utilisation de l'algorithme génétique comme processus de diversification peut apporter un plus à l'algorithme Firefly.

7.2.6 Comparaison entre les trois métaheuristiques

Le tableau 10 reprend le meilleur résultat obtenu par les trois métaheuristique, où on peut remarquer un léger avantage pour le résultat de l'algorithme génétique comparé à l'hybridation.

En se basant sur ces résultats, nous utiliserons un algorithme génétique avec une population de 20 solutions, pour le choix de l'architecture du modèle hybride CNN-LSTM, qui sera ré-entraînée sur 1000 itérations après son paramétrage, pour les comparaisons avec les autres travaux.

Métaheuristique	Fitness obtenu
FireFly	0.9696699
Génétique	0.9750055
Hybridation	0.9706549

TABLE 10 – Comparaison entre les trois métaheuristiques

Pour comparer notre proposition, dans le cas de l'hybridation entre le génétique et le Firefly, en intégrant un facteur de chance pour Firefly, nous avons effectué une exécution de l'hybridation proposée par Wahid et al dans [20]. Le score obtenu par cette méthode, en utilisant une population de 10 solution, est de 0.9496594, un résultat bien inférieur à notre proposition, qui permet aussi d'améliorer les résultats du FireFly, quelque soit la taille de la population.

7.3 Comparaison avec d'autres travaux

Dans cette partie du travail, nous effectuerons une comparaison entre les résultats obtenus par notre proposition, avec les autres méthodes utilisées dans la littérature, en exploiatant les différents data-sets introduits dans la section 6.

7.3.1 HAR using smartphones

Dans ce qui suit, nous allons effectuer une comparaison entre les performances des différentes propositions testées sur le *data set Human Activity Recognition using Smartphones* et notre proposition.

Après ré-entraînement du réseau obtenu précédemment, celui ci atteint une précision de 92.39% sur l'ensemble de test. En rajoutant une couche de Dropout, avant la couche d'inférence, nous parvenons à obtenir une précision de meilleure qualité, 94.60% sur l'ensemble de test, la matrice de confusion engendrée est donnée par le tableau 11

	Classe attribuée					
Classe réelle	1	2	3	4	5	6
1	438	29	26	0	3	0
2	2	456	7	2	1	3
3	0	13	406	0	0	1
4	0	0	0	436	32	23
5	3	0	2	30	477	20
6	0	0	1	45	20	471

TABLE 11 – Matrice de confusion du CNNLSTM obtenu pour le data set HAR using smartphones

Le tableau 12 illustre la précision obtenues par quelques méthodes testées sur cet ensemble de données, ces résultats sont rapportés dans l'article de Charissa et al. [6].

Méthodes utilisées	précision
PCA + MLP	57.10%
EM + NB	74.32%
EM + J48	83.02%
SDAE + MLP	87.77%
EM + ANN	91.08%
EM + SVM	94.61%
CNN [21]	93.21 %
LSTM [21]	89.14 %
CNN + MLP	**94.79%**
Notre proposition	94.60%

TABLE 12 – Comparaison entre les résultats obtenues par les différentes méthodes d'apprentissage automatique et d'apprentissage profond testées sur le data set HAR using Smartphones

D'après les résultats obtenus, on peut remarque que notre proposition ne parvient pas

à améliorer la meilleure précision connue, néanmoins, les performances du réseau obtenu se rapproche du meilleure résultat, avec un écart de 0.19% seulement.

7.3.2 WISDM

Dans ce qui suit, nous allons effectuer une comparaison entre les performances des différentes propositions testées sur le *data set* WISDM, avec notre proposition.

Nous présenterons ici le résultat obtenu après 15 itérations de l'algorithme génétique, après réentraînement sur 1000 itérations, ce réseau atteint 72.40 % sur l'ensemble de test.

Le tableau 13 illustre la précision obtenues par quelques méthodes testées sur cet ensemble de données.

Méthodes utilisées	précision
J48 [11]	85.10 %
Régression logistique [11]	78.10 %
Perceptron Multicouches [11]	91.70 %
RNN [17]	81.74 %
CNN [17]	**92.22 %**
Notre proposition	72.40%

TABLE 13 – Comparaison entre les résultats obtenues par les différentes méthodes d'apprentissage automatique et d'apprentissage profond testées sur le data set WISDM

D'après les résultats présentés dans le tableau 13, on peut constater que le réseau de neurones convolutif parvient à surpasser toutes les autres méthodes d'apprentissage, notre proposition quant à elle, se place en dernière position pour le moment, en attendant la fin de l'exécution de l'algorithme génétique, qui pourrait apporter un meilleur résultat. Il à noter que dans le cas du data-set WISDM, les données de capteurs ne sont pas pré-divisé en sous ensemble d'apprentissage et de test, contrairement au data-set HAR using smartphone, ce qui rend la comparaison avec ces méthodes plus délicate, étant donné que l'ensemble de test diffère entre les travaux.

7.3.3 OPPORTUNITY

Dans la littérature, la métrique utilisée pour l'évaluation des performances des propositions faites pour le data set OPPORTUNITY, est différente de la métrique standard (taux de précision). Pour cet ensemble de donnés, c'est le F-Score pondéré (donnée par l'équation 12) qui est utilisé, étant donné que les classes ne sont pas équitablement présentes (la classe null étant largement dominante), l'utilisation du F-Score pondéré permet de mieux évaluer la qualité d'une proposition [5].

$$FScore = \sum_i 2 \cdot \frac{n_i}{N} \cdot \frac{precision_i \cdot rappel_i}{precision_i + rappel_i} \quad (12)$$

Dans l'équation 12, n_i représente le nombre d'exemple dans la classe i et N le nombre total d'exemples [12].

Étant donné que OPPORTUNITY contient deux étiquetages possibles des activités, les tests sont effectués sellons ces deux possibilités.

Activité de déplacement :

Le résultat présenté ici, est obtenu après seulement 5 itérations de l'algorithme génétique, après ré-entraînement sur 300 itérations.

Le tableau 14 illustre les résultats obtenus en considérant l'étiquetage en activité de déplacement. Dans ce tableau, 1-NN et 3-NN sont des applications de *K-Nearest Neighbors* (Les K plus proches voisins), en utilisant respectivement 1 et

3 voisins. Quant à NCC, elle représente le *Nearest Centroid Classifier*, une méthode qui utilise la distance euclidienne entre l'exemple et le centroïde (un élément qui se rapproche le plus du centre de sa classe) des classes pour la classification [5].

Méthodes utilisées	F-Score
1-NN [5]	0.84
3-NN [5]	0.85
NCC [5]	0.54
CNN [12]	0.879
DeepConvLSTM [12]	**0.895**
Notre proposition	0.835

TABLE 14 – Comparaison entre les résultats obtenues par les différentes méthodes d'apprentissage automatique et d'apprentissage profond testées sur les activités de déplacement du data set OPPORTUNITY

D'après les résultats illustrés dans le tableau 14, nous pouvons constater que la proposition d'Ordonez et al. [12] offre les meilleures résultat, notre proposition obtient un résultat moins performant, cependant, poursuivre l'exécution de l'algorithme génétique, et effectué un apprentissage sur 1000 itérations pourrait l'améliorer.

Gestes :

Le résultat présenté ici, est obtenu après seulement 4 itérations de l'algorithme génétique, après ré-entraînement sur 300 itérations.

Le tableau 15 illustre les résultats obtenus en considérant l'étiquetage en activité de déplacement.

D'après les résultats illustré dans le tableau 15, On remarque une fois encore que la supporiorité de la proposition de Ordonez et al. [12], notre proposition obtient un résultat moins performant , mais ce dernier peut être améliorer une fois l'exécution de l'algorithme génétique achevée.

Méthodes utilisées	F-Score
1-NN [5]	0.87
3-NN [5]	0.85
NCC [5]	0.51
CNN [12]	0.883
DeepConvLSTM [12]	**0.915**
Notre propostion	0.865

TABLE 15 – Comparaison entre les résultats obtenues par les différentes méthodes d'apprentissage automatique et d'apprentissage profond testées sur la catégorie déplacement du data set OPPORTUNITY

‘

7.4 Représentation du signal en images

Dans cette partie de la section, nous allons présenter les différents résultats obtenus en utilisant la transformation des signaux capteurs en images. Les données utilisées proviennent du data set *Human Activity using smartphones*, la transformation en image (couleurs et 3 projections noir & blanc) a été effectuée en suivant le processus expliqué dans la Section 5.

Les résultats obtenus après paramétrage des réseaux CNNs en utilisant l'algorithme FireFly sont illustrés dans le tableau 16.

Type d'images	Précision
En couleurs	65.86%
Noir & blanc	67.49 %

TABLE 16 – Résultats obtenus avec en utilisant la représentation des signaux en images

D'après les résultats obtenus, on peut remar-

quer que les précisions obtenues sont très faibles comparé à ce qui à été vu dans la littérature, ainsi que les résultats obtenus précédemment.

Dans le but d'améliorer les performances, plusieurs pistes ont été envisagées, à l'exemple de la génération des solutions en suivant une distribution de probabilités, de changer l'épaisseur des lignes, ou l'utilisation de sous réseaux.

7.4.1 Génération des solutions

Afin de mieux cerner l'impact de la taille de filtres de convolution et de pooling sur la qualité de l'apprentissage, plusieurs tests ont été effectués, avec un apprentissage sur 50 itérations en variant la taille de ces filtres, de (3,3) à (25,25) avec un pas de 2 pour les filtres de convolutions, et de (3,3) à (11,11) avec un pas de 2 pour les filtres de pooling, la comparaison entre les différents résultats se fait en se basant sur l'erreur. Ces tests ont été menées sur les images en couleurs, le résumé des résultats obtenus est illustré dans le tableau 17

Dans les résultats obtenus, on peut remarquer que la taille des filtres de pooling qui donne les meilleurs résultats correspond aux filtres de grandes tailles (9,9) et (11,11). Inversement, pour les filtres de convolution, les meilleurs résultats sont donnés par les filtres de petites tailles (3,3) à (7,7).

Nous proposons d'exploiter ces résultats pour la génération des solutions, en joignant une probabilité à chaque taille de filtre pour le cas de la convolution et du pooling, de façon à obtenir une distribution de probabilités qui prévilégie les filtres de petites tailles dans le cas de la convolution, et de grande tailles dans le cas du pooling, comme suit :

- Convolution :

$$w_i = maxKernelSize - kernelSize_i + 1$$

Convolution	Pooling	Erreur
(3, 3)	(11, 11)	1,06064
(5, 5)	(9, 9)	**1,045586**
(7, 7)	(9, 9)	1,848606
(9, 9)	(9, 9)	2,376372
(11, 11)	(9, 9)	3,800796
(13, 13)	(9, 9)	9,417874
(15, 15)	(9, 9)	24,09441
(17, 17)	(9, 9)	34,52309
(19, 19)	(9, 9)	46,07142
(21, 21)	(9, 9)	70,49732
(23, 23)	(9, 9)	128,7633
(25, 25)	(9, 9)	358,2663

TABLE 17 – Résultats des tests effectués en variant la taille des filtres sur des images en couleurs

$$p_i = \frac{w_i}{\sum_{j=1}^{n}(w_j)}$$

- Pooling :

$$p_i = \frac{kernelSize_i}{\sum_{j=1}^{n}(kernelSize_j)}$$

Exploiter ces distributions de probabilités, lors de la génération des solutions durant l'exécution de la métaheuristique, permettrait d'augmenter les chances de générer de «bonnes solutions »dès le départ, ce qui pourrait améliorer la qualité du résultat final. En utilisant cette génération des solutions, lors de l'exécution de l'algorithme FireFly, on obtient un modèle dont la précision sur l'ensemble de test est de 67.86%, ce qui améliore de 2% le résultat obtenu précédemment sur les images en couleurs, même si la précision reste très faible comparé à l'utilisation des signaux.

7.4.2 Épaisseur des courbes

Comme seconde proposition d'amélioration, nous avons testé différentes valeurs d'épaisseur des courbes, pour les images en noir & blanc, ainsi que celles en couleurs, en exécutant un algorithme FireFly avec 10 solutions comme population. Dans le cas des images en noir & blanc, l'axe de jointure des images, est lui aussi pris en compte (Images regroupées verticalement, horizontalement, ou en profondeur). Les résultats des différents testes sont illustrés dans les tableaux 19 et 18

Largeur de la ligne	Précision
3	60.84%
4	63.75%
5	**67.45%**
6	61.75%
8	61.82%

TABLE 18 – Résultats obtenus en variant l'épaisseur des lignes pour les images en couleurs

Dans le cas des images en couleurs, on peut remarquer que l'épaisseur la plus adéquate semble être de 5, permettant d'atteindre une précision de 67.45%, qui demeure très faible comparé aux résultats obtenus en utilisant les donnés capteurs sans transformation.

Dans le cas des images en noir & blanc, représentant les projections sur les 3 plans de la courbe 3D, la meilleure combinaison semble être une jointure sur le premier axe (ce qui signifie que les images sont regroupées verticalement), avec une largeur de ligne égale à 6, faisant passer la précision sur l'ensemble de test à 74.68%, malgré cette amélioration considérable (plus de 7%), le résultat obtenu reste faible.

Largeur de la ligne	Axe de la jointure	Précision
3	1	71.83%
3	2	71.86%
3	3	71.25%
4	1	68.03%
4	2	72.34%
4	3	67.42%
5	1	74.14%
5	2	71.12%
5	3	67.49%
6	**1**	**74.68%**
6	2	72.44%
6	3	69.86%
8	1	70.0%
8	2	71.66%
8	3	64.4%

TABLE 19 – Résultats obtenus en variant l'épaisseur des lignes pour les images en noir & blanc

7.4.3 Utilisation de sous réseaux

Objectif de cette approche, est d'utiliser six sous réseaux, un par activité, dans le but d'obtenir des réseaux qui serait capables de reconnaître une seule activité bien précise, avant de les combiner entre elles pour reconnaître l'ensemble des activités. Six sous réseaux ont été mis en place, tous entraînés à reconnaître une seule et unique activité, le tableau 20 illustre l'activité reconnue par chaque réseau, ainsi que le taux de précision.

D'après les résultats illustrés dans le tableau 20, on peut remarquer que les activités "dynamiques" (marcher, monter et descendre les escaliers) sont reconnues avec un taux supérieur à 90%, alors que les activités "statiques" (s'asseoir, se metre debout et s'allonger) sont moins bien reconnue, dans les 80%, même si ces résul-

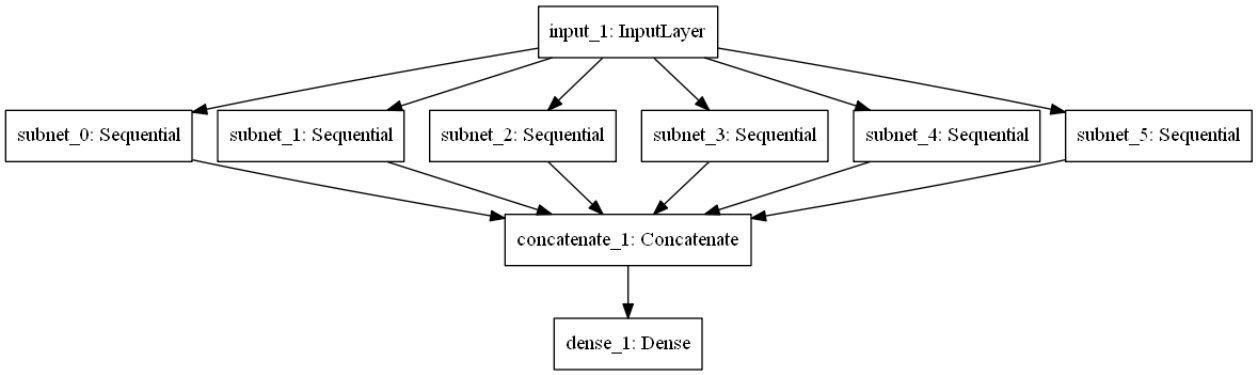

FIGURE 9 – Réseau de neurones englobant les 6 sous réseaux

Réseau	Activité reconnue	Précision
subnet_0	Marcher	95.18%
subnet_1	Monter les escaliers	91.14%
subnet_2	Descendre les escaliers	96.23%
subnet_3	S'asseoir	81.45%
subnet_4	Se mettre debout	85.82%
subnet_5	S'allonger	87.41%

TABLE 20 – Activité reconnue et taux de précision pour chaque sous réseau

tats demeurent plus intéressants que les résultats précédent en utilisant un CNN avec des images en entrée, pour reconnaître les 6 activités à la fois.

Dans un premier temps, nous avons proposer de placer ces 6 sous réseaux en parallèle, et d'exploiter la probabilité retournée par chacun d'eux dans le but d'effectuer la classification, par exemple, pour une image donnée, si la plus grande probabilité a été donnée par $subnet_1$, on classe l'activité comme étant "Monter les escalier". Cette approche a donné une précision sur l'ensemble de test de 61.08 %.

En suite, nous avons proposer de les placer en cascade, par ordre décroissant selon la précision obtenue par chacun,($subnet_2$ en premier, et $subnet_3$ en dernier), l'objectif étant de tester les sous réseaux un à un, et s'arrêter lorsqu'un sous réseau estime que l'instance donnée appartient à la classe dont il est responsable. Cette approche a donné une précision de 64.26 % sur l'ensemble de test.

Finalement, nous avons proposer de regrouper tous les sous réseaux, dans un seul réseau de neurones, comme l'illustre la figure 9, en connectant les sorties de chacun avec la couche d'inférence du "grand" réseau, avant de effectuer l'apprentissage sur cette couche uniquement, dans le but d'attribuer des poids à la sortie de chaque réseau. Après un apprentissage sur 1000 itérations, la précision obtenue sur l'ensemble d'apprentissage est de 64.46 %.

Il semble que l'utilisation des courbes, pour la représentation des série chronologique, engendre une perte d'information, en vu des résultats obtenus, contrairement à la proposition de Hur et al. qui obtient d'excellent résultats, en utilisant la représentation indiquée dans la figure 5 [10].

8 Cadre applicatif

Une application pour la reconnaissance d'activité humaine, en temps réel, a été mise en place, elle se compose d'une application mobile, permettant la récupération des données capteurs (accéléromètre et gyroscope), avant de les envoyer à un serveur, qui s'occupera de la classification (reconnaissance de l'activité) et l'affichage du résultat sur une page web.

8.1 Application mobile

L'application mobile a été développée en utilisant le framework Ionic version 5.4.16, son interface graphique est assez simple (illustrée dans la figure 10), elle comporte deux boutons, pour le début et la fin de la transmission, ainsi que l'affichage des données capteurs en temps réel. La transmission vers le serveur se fait via des requête HTTP, avec une fréquence de 50Hz, la même fréquence utilisée pour la récolte du data set Human Activity Recognition using smartphones, qui a été utilisé pour l'apprentissage du réseau de neurones utilisé par le serveur.

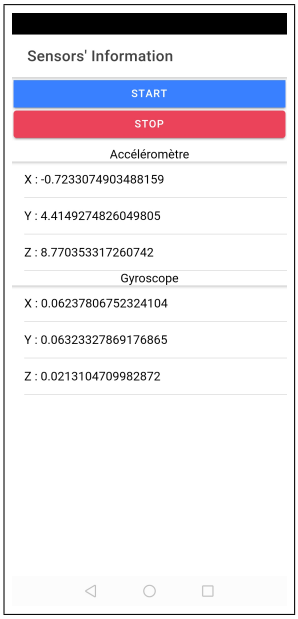

FIGURE 10 – Application smartphone mise en place

8.2 Serveur

Le serveur a été développé en utilisant Flask, son rôle consiste à récupérer les données transmises par l'application mobile, et les utiliser pour la reconnaissance de l'activité via le réseau hybride CNN-LSTM obtenu précédemment. En suite, le résultat obtenu est affiché dans une page web illustrée par la figure 11.

8.3 Problème rencontré

Un script python a été mis en place dans le but de simuler la transmission des données par l'application mobile avec pour objectif de s'assurer du bon fonctionnement de l'application côté serveur en utilisant les données de tests du data set HAR using smarphones. La reconnaissance des activités en utilisant cette simulation était quasiment parfaite. Cependant en utilisant les données envoyées par le smartphone, nous avons rencontré un problème pour la reconnaissance des activités.

Par exemple, pour la séquence d'activités suivante : « Se lever, marcher, descendre les escalier, marcher, monter les escaliers, s'allonger, se lever, marcher, s'asseoir », nous obtenons la séquence suivante : « Monter les escalier, s'allonger, se lever, descendre les escalier, se lever, descendre les escaliers, marcher,descendre les escalier, s'allonger, monter les escalier, etc... ».

Ce problème peut être dû à la segmentation du flux de données, car contrairement aux données

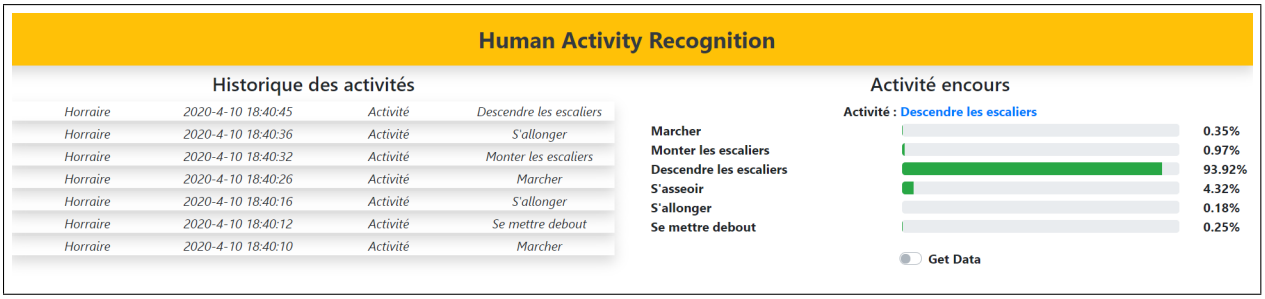

FIGURE 11 – Affichage du résultat de la classification dans la page Web

de tests, qui sont déjà segmentées, il est difficile de déterminer automatiquement, le début de chaque activité dans un flux continu. Nous avons testé différentes valeurs de chevauchement (jusqu'à 75%) pour la segmentation du flux, mais les résultats obtenus ne se rapprochent pas de la séquence réelle.

Une autre cause possible à ce problème est la durée de l'activité, les données du data set HAR using smartphones, sont segmentées sur une durée de 2.56s chacune, cette valeur ne reflète pas la durée réelle d'une activité, à l'exemple de l'activité s'asseoir ou se mettre debout. En suivant cette piste, nous avons segmenté les données du data set WISDM, qui sont fournies sans prétraitement et sans segmentation, en instances dont la durée est d'une seconde. Cette segmentation a permis de mieux reconnaître les activités dynamique (Marcher et Monter/Descendre les escaliers), même si une fois encore, la séquence réelle n'est pas complètement reconnue.

Une solution envisageable pour le problème de la segmentation du flux continu, est d'utiliser la dérivée du signal, comme proposé par Bhat et al dans [3], une approche qui a fournie des résultats très intéressants.

9 Conclusion

La reconnaissance de l'activité humaine représente un domaine de recherche important, notamment sur ses applications possibles pour l'aide à l'assistance de vie ou encore la sécurité basée sur la surveillance.

Suite à une étude bibliographique effectuée durant le premier semestre, nous avons pu constater que les méthodes d'apprentissage profond surpassent les méthodes d'apprentissage classiques et qu'une hybridation entre différentes techniques d'apprentissage profond pouvait améliorer la qualité de la reconnaissance. Nous avons donc choisi de nous orienter vers un modèle composé d'un réseau de neurones convolutifs (CNN), dont les mécanismes d'extraction automatique de caractéristiques résistent aux petites variations dans les données des capteurs, et d'un réseau de neurones récurrents (LSTM), qui permet d'exploiter les dépendances temporelles entre les séries chronologiques des différentes données de capteurs.

Le choix de l'architecture du modèle à utiliser représente une étape cruciale. Ce choix impactera directement les performances du réseau. Ce choix repose sur un problème combinatoire, nous avons opté pour l'utilisation de

trois méta-heuristiques : l'algorithme génétique, l'algorithme FireFly, et une hybridation entre les deux, dans le but de choisir l'architecture de notre modèle. Après expérimentation, il s'est avéré que notre proposition pour l'hybridation permettait d'améliorer les résultats de FireFly ainsi que l'hybridation de Wahid et al. [20], mais celle-ci reste devancée de peu par l'algorithme génétique, c'est donc ce dernier qui a été retenu pour la configuration de notre modèle, Ce choix nous a permis d'obtenir des résultats proches du meilleur résultat connu. Même si notre proposition ne parvient pas à améliorer les performances, elle représente néanmoins, un bon point de départ pour la configuration d'un réseau de neurones, dont les performances pourraient être améliorées en jouant sur d'autres paramètres, ce fut d'ailleurs le cas avec l'ajout d'une couche de Dropout.

A partir de ces résultats, nous avons proposé de représenter les séries chronologiques, sous forme de courbe 3D afin de simplifier la configuration d'un réseau de neurones hybride CNN-LSTM en un réseau de neurones convolutif simple. Nous avons donc transformé les données capteurs en image, selon deux propositions. La première consiste à représenter la profondeur suivant une valeur de couleur, la deuxième consiste à projeter la courbe 3D noir & blanc dans les trois plans. Le but étant précisément de représenter la dépendance temporelle présente dans les signaux, sous la forme d'une dépendance spatiale. Les précisions obtenues par cette approche se sont révélées très faibles, comparé à l'utilisation de signaux capteurs. Ces résultats pourraient être dû à une perte d'information lors de l'utilisation de cette représentation.

Enfin, nous retenons la difficulté de la mise en place d'une application pour la reconnaissance d'activité humaine en temps réel. Cette difficulté n'est pas dû à la reconnaissance de l'activité en elle-même, car les résultats obtenus sur les données de tests du data-set "Human Activity Recognition using smartphones", sont quasiment parfaits. La difficulté principale est essentiellement liée à la segmentation du flux continu de données capteurs, où il est difficile de déterminer le début et la fin d'une activité dans un enchaînement d'activités en continu.

Perspectives

Des exécutions de l'algorithme génétique sont toujours en cours, nous sommes dans l'attente de meilleurs résultats par rapport à ceux présentés dans ce rapport, qui sera mis à jour dès l'obtention de ces derniers.

L'ajout d'une couche de Dropout, nous a permis d'améliorer le résultat du réseau obtenu par l'algorithme génétique. Par la suite, il faudrait s'intéresser à d'autres hyperparamètres des réseaux de neurones, à l'exemple des couches de dropout, des fonctions d'activation, ou encore de la régularisation des poids, dans le but d'améliorer les résultats obtenus par les méta-heuristiques.

Pour améliorer l'application mise en place, pour la reconnaissance de l'activité humaine en temps réel, la piste la plus intéressante serait d'exploiter l'approche proposée par Bhat et al. dans [3], pour détecter automatiquement le début de chaque activité. Son approche consiste à prendre en compte la dérivée du signal temporel. Cette approche permettrait d'améliorer la segmentation du flux de données capteurs transmis par l'application mobile.

Ce travail a été effectué dans la perspective d'une collaboration avec le laboratoire DOMUS de l'université de Sherbrooke au Canada. Le but

de cette collaboration est de mettre en place un système permettant d'améliorer l'assistance de vie aux personnes atteintes de traumatismes crâniens en détectant les variations ou déviations dans l'exécution de leurs activités. Dans ce contexte, il sera intéressant de combiner la reconnaissance de l'activité humaine en temps réel, avec une détection de variation d'activité, à l'exemple de la proposition de Saives [15], qui consiste à représenter les activités via un automate, pour ensuite détecter les déviations du comportement habituel. Joindre la reconnaissance d'activité humaine en utilisant notre CNN-LSTM avec la méthode décrite précédemment, permettrait de mettre en place un système de détection automatique d'anomalies dans l'exécution d'une activité quotidienne.

Remerciements

J'adresse mes sincères remerciements à monsieur Alexis Nédélec, qui m'a honoré en acceptant de m'encadrer et de me guider durant ce stage de Master, je le remercie aussi pour son encadrement de qualité, sa disponibilité ainsi que ses conseils, qui ont fait de ce projet une expérience agréable et inoubliable. Je remercie également les membres du CERV, pour leur accueil chaleureux au sein du laboratoire, m'offrant ainsi la possibilité de vivre cette expérience. Je tiens à remercier aussi toute personne, qui de près ou de loin, a contribué à la réalisation de ce travail.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and J Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. 01 2013.

[2] Vina Ayumi, L.M. Rere, Mohamad Ivan Fanany, and Aniati Arymurthy. Optimization of convolutional neural network using microcanonical annealing algorithm. 10 2016. doi : 10.1109/ICACSIS.2016.7872787.

[3] Ganapati Bhat, Ranadeep Deb, Vatika Chaurasia, Holly Shill, and Umit Ogras. Online human activity recognition using low-power wearable devices, 08 2018.

[4] Adenilson Carvalho, Fernando Ramos, and Antonio Chaves. Metaheuristics for the feedforward artificial neural network (ann) architecture optimization problem. *Neural Computing and Applications*, 20, 10 2010. doi : 10.1007/s00521-010-0504-3.

[5] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundaratejaswi Digumarti, Gerhard Tröster, José del R. Millán, and Daniel Roggen. The opportunity challenge : A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34, Issue(15) :10. 2033–2042, 2013. doi : 10.1016/j.patrec.2012.12.014. URL http://infoscience.epfl.ch/record/182860.

[6] Charissa et al. "human activity recognition with smartphone sensors using deep learning neural networks". *Elsevier, Expert Systems With Applications*, 2016.

[7] Friday et. al. "deep learning algorithms for human activity recognition using mobile and wearable sensor networks : State of the art and research challenges". *Elsevier, Expert Systems With Applications*, pages 233–261, 2018.

[8] Wang et. al. "deep learning for sensor-based activity recognition : A survey". *Elsevier, Pattern Recognition Letters*, pages 3–11, 2019.

[9] Iztok Fister, Xin-She Yang, Dušan Fister, and Iztok Fister. *Firefly Algorithm : A Brief Review of the Expanding Literature*, pages 347–360. Springer International Publishing, Cham, 2014.

[10] Taeho Hur, Jae Bang, Thien Huynh-The, Jongwon Lee, Jee-In Kim, and Sungyoung Lee. Iss2image : A novel signal-encoding technique for cnn-based human activity recognition. *Sensors*, 18 :3910, 11 2018. doi : 10.3390/s18113910.

[11] Jennifer Kwapisz, Gary Weiss, and Samuel Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explorations*, 12 : 74–82, 11 2010. doi : 10.1145/1964897.1964918.

[12] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016. ISSN 1424-8220.

[13] L.M. Rere, Mohamad Ivan Fanany, and Aniati Arymurthy. Metaheuristic algorithms for convolution neural network. *Computational Intelligence and Neuroscience*, 2016, 05 2016. doi : 10.1155/2016/1537325.

[14] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Forster, Gerhard Troster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and Jose del R. Millan. Collecting complex activity datasets in highly rich networked sensor environments. pages 233 – 240, 07 2010. doi : 10.1109/INSS.2010.5573462.

[15] Jeremie Saives, Clement Pianon, and Gregory Faraut. Activity discovery and detection of behavioral deviations of an inhabitant from binary sensors. *IEEE Transactions on Automation Science and Engineering*, 12 :1211–1224, 09 2015. doi : 10.1109/TASE.2015.2471842.

[16] Marc Sevaux. Métaheuristiques : Stratégies pour l'optimisation de la production de biens et de services. (metaheuristics : strategies for the optimisation of the production of goods and services). 2004.

[17] Sarbagya Shakya, Chaoyang Zhang, and Zhaoxian Zhou. Comparative study of machine learning and deep learning architecture for human activity recognition using accelerometer data. 8, 11 2018. doi : 10.18178/ijmlc.2018.8.6.748.

[18] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko, and M. Tuba. Designing convolutional neural network architecture by the firefly algorithm. In *2019 International Young Engineers Forum (YEFECE)*, pages 59–65, 2019.

[19] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary Yen. Automatically designing cnn architectures using genetic algorithm for image classification, 08 2018.

[20] Fazli Wahid, Rozaida Ghazali, and Lokman Ismail. Improved firefly algorithm based on

genetic algorithm operators for energy efficiency in smart buildings. *Arabian Journal for Science and Engineering*, 44, 02 2019. doi : 10.1007/s13369-019-03759-0.

[21] Shaohua Wan, Lianyong Qi, Xiaolong Xu, Chao Tong, and Zonghua Gu. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, 25, 12 2019. doi : 10.1007/s11036-019-01445-x.

[34] Xin-She Yang. *Nature-Inspired Algorithms and Applied Optimization*. 01 2018. ISBN 978-3-319-67668-5. doi : 10.1007/978-3-319-67669-2.

References annexes

[23] Y. Bengio. *Learning Deep Architectures for AI*. now, 2009. ISBN null.

[24] Yoshua Bengio. Deep learning of representations : Looking forward. In Adrian-Horia Dediu, Carlos Martín-Vide, Ruslan Mitkov, and Bianca Truthe, editors, *Statistical Language and Speech Processing*, pages 1–37, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39593-2.

[25] Diane Cook, Kyle Feuz, and Narayanan Krishnan. Transfer learning for activity recognition : A survey. *Knowledge and information systems*, 36 :537–556, 09 2013.

[26] Thi Duong, Dinh Phung, Hung Bui, and Svetha Venkatesh. Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 173(7) :830 – 856, 2009. ISSN 0004-3702.

[27] Chen et al. Sensor-based activity recognition. *Trans. Sys. Man Cyber Part C*, 42(6) : 790–808, November 2012.

[28] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6) : 82–97, Nov 2012. ISSN 1558-0792.

[29] Lara and Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3) :1192–1209, 2013.

[30] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 :436–44, 05 2015.

[31] Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1729–1734, 2011.

[32] A. Sharma, Y. Lee, and W. Chung. High accuracy human activity monitoring using neural network. In *2008 Third International Conference on Convergence and Hybrid Information Technology*, volume 1, pages 430–435, Nov 2008. doi : 10.1109/ICCIT.2008.394.

[33] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, June 2015.

[34] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition, 2015.