République Algérienne Démocratique et Populaire الجمهورية
الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique وزارة
التعليم العالي و البحث العلمي

جامعة عبد الرحمان ميرة
Université A. Mira de Bejaïa
Faculté des Sciences Exactes Département d'Informatique

Mémoire de fin de cycle

En vue d'obtention du diplôme de Master en Informatique

Option : Systèmes d'Information Avancée (SIA)

Titre : Développement d'un modèle hybride CNN-LSTM pour la reconnaissance des activités humaines

---

Réalisé par :
M. Mohamed Chafik KERNOU
M. Adem CHIBANE

Soutenu DATE 01/07/2025, Devant le jury composé de :

M. Achour ACHROUFENE	Université de Bejaia	Encadrant
Melle. Lynda RAID	Université de Bejaia	Co-encadrant
M. Mohammed KHAMMARI	Université de Bejaia	Président
M. Kamal AMROUN	Université de Beiaia	Examinateur
Mme. Samira BOUKERRAM	Université de Bejaia	Examinatrice
Mme. Karima ADEL	Université de Bejaia	Examinatrice

Promotion : 2024/2025

Dédicace

“

Tout d’abord, je rends grâce à Dieu, pour m’avoir accordé la santé, l’intelligence et la force d’atteindre cet objectif que je portais dans mon cœur depuis mes premiers pas à l’école.

À ma chère maman, je te suis profondément reconnaissant pour m’avoir transmis l’amour du savoir, le goût de l’effort et les valeurs de persévérance et de sagesse.

À mon père, modèle de droiture, Merci pour ton soutien indéfectible, tes encouragements constants et ta foi inébranlable en moi, ton regard bienveillant a toujours été une source de motivation.

À ma petite sœur Serine, véritable amie et rayon de lumière dans ma vie, merci pour ta présence, ton soutien silencieux et ta capacité à m’apaiser dans les moments les plus rudes.

À mon petit frère Abdellah, compagnon de route et source d’énergie, merci pour ta spontanéité et ta confiance, qui m’ont souvent rappelé l’importance de rester soi-même.

Et à tous ceux qui me sont chers, en particulier Sara, ce mémoire est l’aboutissement d’un rêve devenu réalité grâce à vous tous, merci du fond du cœur.

”

- Chafik

“

Louange à Dieu, le Tout-Puissant, pour m’avoir accordé la force, la patience et la sagesse nécessaires à l’accomplissement de ce travail.

À ma chère mère, je dédie ce mémoire avec tout mon amour et une gratitude profonde. Ta patience, ton soutien constant et tes sacrifices silencieux ont été les piliers de ma réussite.

À mon père, votre présence rassurante, vos efforts inlassables et votre foi en moi ont contribué à forger la personne que je suis aujourd’hui. Je vous en suis infiniment reconnaissant.

À ma petite sœur, ton affection, ta présence réconfortante et ton soutien discret ont été pour moi une source de courage tout au long de ce parcours.

À ma famille dans son ensemble, merci pour votre amour, votre confiance et vos encouragements continus.

À mes amis Hilal, Arcelane, Anis et Adel, votre amitié sincère, vos encouragements et vos moments de soutien partagés ont été précieux dans cette aventure académique.

Je vous dédie ce mémoire avec fierté et reconnaissance. Grâce à vous tous, ce projet a pu voir le jour.

Merci du fond du cœur.

”

- Adem

Remerciements

Nous exprimons avant tout notre profonde gratitude envers Allah, le Tout Miséricordieux, qui nous a donné la patience et le courage nécessaires pour mener à bien ce mémoire.

Nous tenons à adresser nos remerciements sincères à notre encadrant, Monsieur Achour ACHROUFENE, pour son engagement exemplaire, ses conseils précieux et ses critiques constructives, qui nous ont permis d'avancer avec assurance et détermination tout au long de ce projet. Nous remercions également chaleureusement notre co-encadrante, Mademoiselle Lynda RAID, dont l'accompagnement attentif et les recommandations pertinentes ont grandement contribué à enrichir notre travail.

Nous exprimons également notre gratitude aux membres du jury, dont l'évaluation rigoureuse, les suggestions judicieuses et les observations constructives ont permis d'améliorer sensiblement ce mémoire.

Enfin, nos remerciements vont également à l'ensemble de nos professeurs d'université, grâce à qui nous avons acquis des connaissances précieuses et développé des compétences essentielles pour notre avenir professionnel.

Table des matières

Table des matières

Tables des figures

Liste des Tableaux

Liste des abréviations

AE	*Autoencoder Neural Networks*
AI	*Artificial Intelligence (IA en français)*
AR	*Activity Recognition*
Bi-LSTM	*Bidirectional Long Short-Term Memory*
CNN	*Convolutional Neural Network*
ConvLSTM	*Convolutional Long Short-Term Memory*
CRF	*Conditional Random Fields*
DL	*Deep Learning*
DMI	*Dynamic Motion Image*
EMG	*Electromyogram*
GAN	*Generative Adversarial Networks*
GPS	*Global Positioning System*
GRU	*Gated Recurrent Unit*
HAR	*Human Activity Recognition*
HMM	*Hidden Markov Model*
HCI	*Human-Computer Interaction*
IoT	*Internet of Things*
KNN	*K-Nearest Neighbors*
LRCN	*Long-term Recurrent Convolutional Network*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MLFF	*Multi-Level Fusion Framework*
NLP	*Natural Language Processing*
PIR	*Passive Infrared Sensor*
PAMAP2	*Physical Activity Monitoring for Aging People Dataset*
RGB	*Red Green Blue (canaux couleur des images)*
RNN	*Recurrent Neural Network*
SVM	*Support Vector Machine*
UCF50	*University of Central Florida 50-class dataset*
ViT	*Vision Transformer*
WISDM	*Wireless Sensor Data Mining*

Introduction générale

La reconnaissance d'activités humaines (HAR, pour *Human Activity Recognition*) est un domaine de recherche en plein essor, à l'intersection de la vision par ordinateur, de l'intelligence artificielle et de l'apprentissage automatique. Elle consiste à identifier automatiquement les actions réalisées par une ou plusieurs personnes, à partir de données captées par des capteurs (inertiels, ambiants ou visuels). Initialement utilisée dans des applications médicales ou de suivi d'activité, la HAR est aujourd'hui présente dans de nombreux secteurs : santé, sécurité, sport, domotique, interaction homme-machine, etc.

L'émergence de dispositifs portables et de caméras omniprésentes, couplée aux avancées de l'apprentissage profond (*Deep Learning*), a permis un changement de paradigme dans les méthodes de reconnaissance d'activités. Alors que les approches classiques reposaient sur l'extraction manuelle de caractéristiques et des classifieurs traditionnels, les modèles profonds offrent désormais la possibilité d'apprendre automatiquement des représentations des données, en exploitant à la fois leur structure spatiale et temporelle.

Dans ce contexte, les architectures hybrides de type CNN–LSTM (Convolutional Neural Network + Long Short-Term Memory) se sont révélées particulièrement efficaces pour la reconnaissance d'actions à partir de séquences vidéo. Les CNN permettent d'extraire des traits visuels pertinents de chaque image, tandis que les LSTM modélisent la dynamique temporelle de ces traits sur la durée de la vidéo.

Le problème est de concevoir un modèle fiable pour reconnaître des activités humaines dans des vidéos, malgré le surapprentissage et la complexité des données. Les approches classiques manquent de précision et de généralisation. *La solution proposée* est un modèle CNN-LSTM optimisé, combinant vision spatiale et mémoire temporelle. Des techniques comme l'early stopping et le réglage fin des paramètres améliorent nettement la performance.

Pour ce faire, ce document est structuré en quatre chapitres :

Dans le premier chapitre, nous présentons les fondements de la HAR et ses domaines d'application, ainsi que les types de capteurs largement employés pour capter l'activité humaine.

Dans le deuxième chapitre, nous abordons un état de l'art des approches de HAR basées sur le Deep Learning, en mettant en évidence les architectures pertinentes (CNN, LSTM, Transformers) et le rôle de l'apprentissage par transfert dans des contextes à faibles données.

Nous détaillons ensuite, dans le chapitre trois, la démarche appliquée pour l'amélioration du modèle CNN-LSTM pour la reconnaissance d'activités : conception du modèle, choix des paramètres, traitement des données vidéo et mise en œuvre de différentes variantes CNN-LSTM sur le dataset UCF50.

Enfin, dans le chapitre quatre, nous analysons les performances des modèles obtenus, identifions les clés d'amélioration et discutons des perspectives ouvertes pour des applications concrètes de HAR robustes, efficaces et facilement mises en œuvre.

-

CHAPITRE 1 :

GÉNÉRALITÉS SUR L'ACTIVITÉ HUMAINE

I. CHAPITRE 1 : GÉNÉRALITÉS SUR L'ACTIVITÉ HUMAINE

1.1 Introduction

La reconnaissance d'activités humaines – souvent désignée par le sigle *HAR* pour *Human Activity Recognition* – est un domaine de recherche qui vise à identifier automatiquement les actions ou comportements d'une personne à partir de données collectées par divers capteurs [1]. Initialement explorée dans des contextes tels que la santé et la rééducation (par exemple pour le suivi mobile de patients [2]), la HAR s'est étendue à de nombreux autres domaines d'application au cours des deux dernières décennies. Elle couvre aujourd'hui un large éventail de situations, de la surveillance de la condition physique et de la détection de chutes en milieu médical jusqu'à la sécurité et l'assistance à domicile, en passant par les interactions homme-machine contextuelles et la vie quotidienne [1].

Plusieurs facteurs technologiques ont contribué à l'essor de la HAR. D'une part, la prolifération des dispositifs mobiles et des objets connectés embarquant des capteurs (smartphones, montres intelligentes, bracelets d'activité, etc.) a fourni une infrastructure omniprésente pour collecter des données sur les actions humaines [3]. D'autre part, l'augmentation des capacités de calcul et les progrès des algorithmes d'apprentissage automatique permettent d'analyser les données collectées en temps réel et avec une précision élevée, y compris sur des plateformes embarquées [4]. Sur un autre plan, la disponibilité de plusieurs datasets et l'organisation des défis compétitifs ont fait des avancées dans ce domaine. Tout cela a fait que la reconnaissance d'activités est devenue un champ d'étude intéressant, alimenté par les besoins de nombreuses applications pratiques et par les possibilités offertes par les capteurs toujours plus miniaturisés et performants.

Dans ce chapitre, nous introduisons les fondements de la reconnaissance d'activité humaine, puis nous présentons ses domaines d'application. Ensuite, nous explorerons les

grandes approches utilisées pour la reconnaissance d'activités. Enfin, présentons quelques techniques de machine learning utilisées en reconnaissance d'activité et les mesures de performance qui servent à les évaluer.

1.2 Reconnaissance d'activité humaine

1.2.1 Définition d'activité

L'activité humaine fait référence aux mouvements, gestes ou actions physiques effectués impliquant une sortie d'énergie. Il existe deux catégories d'activités humaines : simples et complexes. Selon les activités humaines simples, on considère la posture du corps et le mouvement pour définir les différentes activités (marcher, courir, ...) et les activités humaines complexes consistent en des activités simples accompagnées d'une fonction spécifique (manger, par exemple).

1.2.2 Reconnaissance d'activité humaine

La reconnaissance d'activité humaine est devenue un sujet très populaire dans le domaine de la vision par ordinateur et du traitement du signal. Un grand nombre d'applications de reconnaissance des actions humaines à partir des vidéos peuvent être trouvées : la vidéo-surveillance, l'interaction homme-machine et l'indexation des vidéos.

Le but d'un système de reconnaissance d'activité humaine est d'identifier les actions simples de la vie quotidienne (comme marcher, courir, sauter ...). Chacune de ces actions, réalisée par une seule personne dans un laps de temps précis, doit être représentée par un modèle de mouvement simple.

Au cours de ces dernières années, de nombreuses méthodes ont été proposées pour la reconnaissance et la compréhension des actions humaines. Cependant, le domaine de la reconnaissance d'activités humaines est particulièrement difficile vu le nombre de contraintes à surmonter dans les deux domaines suivants:

- Dans le domaine de la vision, on peut mentionner :
  - le changement du point de vue de la caméra,
  - le changement du fond,
  - la variation de la luminosité,
  - l'énorme quantité de données vidéo, etc.
- Dans le domaine du signal, on peut mentionner :

- la présence du bruit,
- la perte du signal,
- la variation des activités humaines, etc.

Avec les avancées rapides dans les technologies de l'informatique et de la miniaturisation, les systèmes de reconnaissance d'activité humaine sont devenus une partie importante du quotidien de l'individu et sont largement utilisés dans de nombreuses applications.

1.3 Domaines d'application de la reconnaissance d'activités

Nous allons présenter quelques applications des nombreux domaines de la HAR :

- Dans le secteur de la **santé et du bien-être**, la HAR permet le suivi de l'état physique de personnes sur le long terme et l'assistance aux patients vulnérables. Par exemple, des capteurs portés peuvent détecter automatiquement une chute ou un manque d'activité chez une personne âgée isolée, déclenchant une alerte aux secouristes. Pärkkä et al. [5] ont montré qu'il est possible de classifier différentes activités du quotidien (marche, repos, tâches ménagères, etc.) à l'aide de capteurs portés de façon réaliste par des personnes dans leur vie quotidienne. De tels systèmes, en surveillant en continu l'activité physique et les paramètres vitaux, ouvrent la voie au suivi médical à distance et à la prévention de problèmes de santé [2].
- Dans les **habitats intelligents** et l'assistance à domicile, la HAR est utilisée pour analyser les activités de la vie quotidienne et superviser la sécurité des occupants. Des réseaux de capteurs ambiants (détecteurs de mouvement, capteurs sur les objets du domicile, etc.) permettent de suivre les actions effectuées dans une maison connectée et d'en déduire des situations anormales ou potentiellement dangereuses. Par exemple, Riboni et Bettini [6] proposent un système contextuel capable de reconnaître des activités complexes en combinant les informations de multiples capteurs disposés dans l'environnement domestique. De tels systèmes peuvent, entre autres, aider à détecter qu'une personne confuse oublie d'éteindre un appareil ou ne suit pas sa routine habituelle, afin de fournir une assistance proactive.
- En **sport et condition physique**, la reconnaissance automatique d'activités sert à quantifier l'exercice réalisé et à fournir un retour personnalisé. L'identification des types de mouvements (course, marche, exercices spécifiques) et l'estimation de leur intensité permettent d'élaborer des programmes d'entraînement intelligents. Tapia *et al.* [7] ont, par exemple, ont combiné des accéléromètres corporels et un moniteur de fréquence cardiaque pour suivre en temps réel l'intensité d'activités physiques variées (marche, course, montées d'escalier, etc.), ouvrant la voie à des applications de coaching sportif utilisant des capteurs portables. De même, Khan *et al.* [8] décrivent une méthode de reconnaissance de mouvements à l'aide de capteurs inertiels qui atteint

une précision élevée sur des activités locomotrices, ce qui peut être exploité pour analyser les performances sportives et prévenir les blessures.

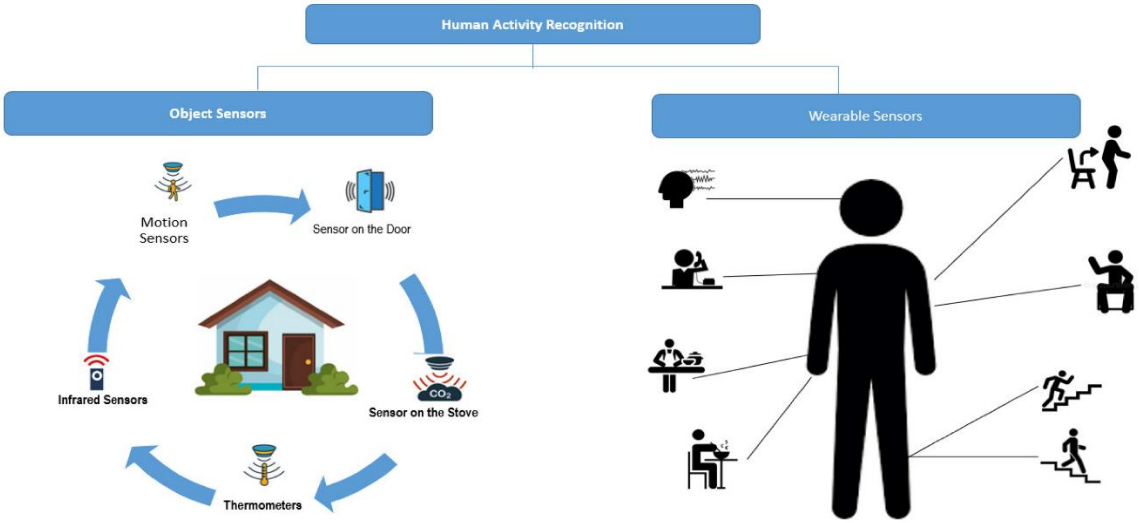

Figure I:1 Domaines d'application de la reconnaissance d'activités[37]

- En **sécurité et surveillance**, la HAR peut contribuer à la détection d'actions suspectes ou de comportements anormaux dans les lieux publics ou privés. Par exemple, l'analyse automatisée de séquences d'actions pourrait signaler une intrusion (mouvements inhabituels dans une zone sécurisée) ou identifier des altercations physiques via des capteurs vidéo ou inertiels portés par des agents de sécurité. Cette branche de la HAR fait largement appel aux caméras et à la vision par ordinateur (analyse vidéo en temps réel).
- Enfin, dans les domaines de l'**interaction homme-machine** et des environnements intelligents, la reconnaissance d'activités apporte un contexte permettant d'adapter dynamiquement les services numériques au comportement de l'utilisateur. Un smartphone peut ainsi ajuster automatiquement son mode de fonctionnement (sonnerie désactivée en réunion, interface simplifiée pendant la conduite, etc.) en se basant sur l'activité courante du propriétaire détectée par ses capteurs. Des architectures logicielles ont été développées pour tirer parti de ces informations contextuelles : Williams et Mathew [9] proposent par exemple une architecture de services mobiles capable de prendre en compte le contexte de l'utilisateur (activité, localisation, etc.) afin d'ajuster les applications. De même, Wei *et al.* [10] présentent une approche exploitant simultanément plusieurs sources de contexte sur l'appareil mobile afin d'améliorer la compréhension de la situation de l'utilisateur.

1.4 Approches de reconnaissance d'activités humaines

Pour saisir les activités, plusieurs recherches en HAR ont classé les approches en deux catégories principales : la recherche basée sur la vision et la recherche basée sur les capteurs.

1.4.1 Vision par ordinateur

Cette méthode se présente sous forme de caméras statiques installées à divers endroits à des fins de surveillance, qui enregistrent des vidéos et les stockent sur des serveurs. Ces flux de caméra ou vidéos enregistrés sont ensuite utilisés à des fins de surveillance. Ce type de reconnaissance d'activité humaine est utilisé pour la sécurité routière, la sécurité publique, la gestion du trafic, la surveillance de foule, etc.

1.4.2 Utilisation de capteurs

La reconnaissance d'activités repose sur la collecte de données pertinentes reflétant les actions effectuées par les individus. Plusieurs catégories de **capteurs** peuvent être mises en œuvre pour acquérir ces informations, depuis les capteurs portables sur le corps jusqu'aux capteurs répartis dans l'environnement. Ainsi, la reconnaissance de l'activité humaine basée sur l'utilisation des signaux provenant de différents types de capteurs est un problème de classification de série chronologique.

Nous présentons ci-dessous les principales sources de données utilisées en HAR, ainsi que la contribution de l'intégration du contexte.

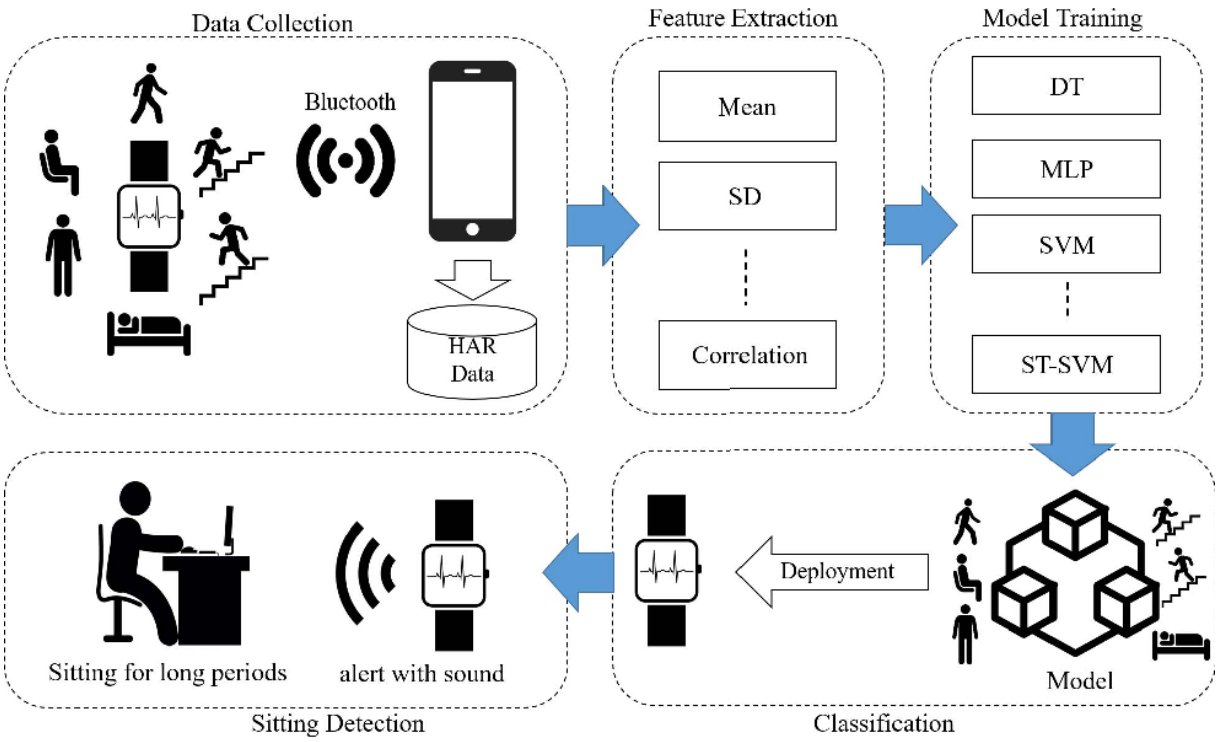

Figure I:2 schéma montrant un smartphone et une montre connectée mesurant des données[38]

1.4.2.1 Capteurs portés sur le corps

Les capteurs portables (*wearables*), fixés directement sur le corps ou intégrés aux vêtements, constituent une source privilégiée de données pour la reconnaissance d'activités. Parmi ceux-ci, les plus courants sont les **capteurs inertiels** tels que les accéléromètres et les gyroscopes, qui mesurent respectivement les accélérations linéaires et les rotations angulaires. Dispositifs peu coûteux largement utilisés pour reconnaître des activités physiques de base (marcher, courir, rester debout, s'asseoir, etc.) [11]. Par exemple, Bao et Intille [9] ont montré qu'en équipant une personne de quelques accéléromètres répartis (poignet, hanche, bras, jambe), il est possible de reconnaître automatiquement un ensemble d'une vingtaine d'activités quotidiennes (marcher, monter les escaliers, faire le ménage, etc.) avec un taux de précision d'environ 84%. De même, Maurer *et al.* [10] ont utilisé plusieurs capteurs inertiels positionnés à différents endroits du corps pour suivre simultanément diverses actions. En pratique, le placement du capteur joue un rôle crucial dans la qualité de reconnaissance obtenue. Un accéléromètre fixé à la taille ne percevra pas exactement les mêmes signaux qu'un accéléromètre attaché au poignet ou à la cheville. Selon He *et al.* [12], le meilleur endroit pour porter un capteur de mouvement

généraliste (dans le cadre d'activités locomotrices) serait la poche du pantalon, car il y est proche du centre de gravité du corps et subit moins de mouvements parasites.

D'autres capteurs portés peuvent contribuer à la reconnaissance d'activités. Les **capteurs physiologiques**, par exemple, mesurent des signaux corporels comme la fréquence cardiaque, la température cutanée ou l'activité musculaire (EMG). Intégrés à des ceintures thoraciques ou à des montres connectées, ils offrent une indication sur l'intensité de l'effort ou l'état interne de la personne. Tapia *et al.* [7] ont incorporé un capteur de fréquence cardiaque à leur dispositif de suivi d'activité et ont montré qu'il permet d'évaluer non seulement le type d'activité, mais aussi son intensité (différenciant par exemple marche lente vs rapide).

La fusion de données **inertielles** et **physiologiques** peut enrichir le système en apportant une dimension supplémentaire. Lara *et al.* [11] ont développé le système *Centinela* qui combine un accéléromètre corporel et un capteur de signe vital (cardio-fréquence mètre) pour réaliser la reconnaissance d'activité en temps réel : cette approche bimodale a montré une amélioration de la précision pour certaines activités en exploitant à la fois le mouvement et la réponse physiologique de l'utilisateur.

1.4.2.2 Capteurs embarqués dans les objets du quotidien

Il est également possible d'exploiter les capteurs déjà intégrés dans les objets utilisés couramment par les personnes. Le meilleur exemple est le **smartphone** qui est équipé d'une panoplie de capteurs (accéléromètre, gyroscope, magnétomètre, GPS, microphone, baromètre, etc.) et accompagne la plupart des individus durant la majeure partie de la journée. De nombreuses études ont démontré la faisabilité de la HAR via les capteurs embarqués des téléphones mobiles. Brezmes *et al.* [12] sont parmi les premiers à avoir utilisé l'accéléromètre d'un smartphone pour classer automatiquement plusieurs activités simples comme la marche, la course, la position assise ou debout. Kwapisz *et al.* [13] ont confirmé ces résultats en collectant des données d'un téléphone des activités de locomotion et de transition (marcher, monter des escaliers, etc.) atteignant des taux de reconnaissance supérieurs à 90% avec des algorithmes d'apprentissage automatique.

Reddy *et al.* [3] ont tiré parti du principe de l'accélération pour déterminer automatiquement le mode de transport (marche, vélo, voiture, bus, etc.) d'un individu à partir des capteurs de son téléphone, en combinant des caractéristiques de déplacement

issues du GPS avec celles de l'accéléromètre. Tanaka *et al.* [15] proposent également un système de *lifelogging* (journal de vie) utilisant le GPS du smartphone pour reconstituer le contexte quotidien d'un utilisateur. Toutefois, l'utilisation des capteurs de localisation comporte des limites : le GPS fonctionne mal en intérieur, consomme beaucoup d'énergie et pose des problèmes de confidentialité des données personnelles [6][16]. Pour surmonter ces inconvénients, des approches hybrides associent les capteurs embarqués du smartphone entre eux. Par exemple, l'application *ActiServ* de Berchtold *et al.* [4] fusionne les données de l'accéléromètre du téléphone avec d'autres capteurs du mobile et des traitements embarqués afin d'offrir un service de reconnaissance d'activité entièrement autonome sur l'appareil.

Il convient de souligner que d'autres objets du quotidien peuvent être dotés de capteurs pour la HAR : montres intelligentes et bracelets connectés (accéléromètres au poignet pour détecter des gestes), chaussures équipées de capteurs de pression pour analyser la démarche, vêtements intelligents tissés de capteurs (par exemple, des capteurs d'EMG dans un t-shirt pour reconnaître des mouvements de bras), etc. L'**Internet des Objets (IoT)** offre ainsi un riche écosystème de dispositifs connectés pouvant servir de sources de données distribuées sur et autour de la personne. L'un des défis majeurs est alors de combiner ces flux d'informations hétérogènes de manière cohérente pour inférer l'activité globale de l'utilisateur.

1.4.2.3 Capteurs ambiants et environnementaux

Une autre catégorie de capteurs utilisés en reconnaissance d'activités concerne les capteurs placés dans l'environnement, parfois appelés **capteurs ambiants**. Ces capteurs mesurent des changements dans l'environnement physique entourant la personne. Ils sont typiquement employés dans les espaces intelligents (domiciles, salles de travail, hôpitaux, etc.) dans le but de reconnaître les activités in situ sans contraindre l'utilisateur à porter quoi que ce soit. On trouve parmi eux des détecteurs infrarouges de présence (PIR) pour repérer les déplacements dans une pièce, des capteurs de pression sous le plancher ou sur un canapé pour détecter qu'une personne marche ou s'assied, des capteurs d'ouverture de porte ou d'utilisation d'objets (par exemple un capteur sur le réfrigérateur indiquant qu'on l'ouvre, interprété comme le début d'une activité de préparer un repas), des microphones et capteurs sonores pour capter les bruits liés à certaines activités (écouter de la musique,

regarder la télévision, prendre une douche…). Des études ont montré que, utilisés seuls, les capteurs environnementaux ne suffisent généralement pas à identifier de manière fiable une activité précise [6][5]. Par conséquent, les systèmes de HAR combinent généralement les données de multiples capteurs ambiants pour reconstruire une image plus complète de l'activité en cours.Par exemple, l'expérimentation de Pärkkä *et al.* [5] intégrait, outre des capteurs corporels, plusieurs **variables environnementales** (telles que la température ambiante, le niveau sonore et la luminosité) dans la classification de sept activités du quotidien. Enfin, les capteurs ambiants jouent un rôle de soutien précieux dans la HAR, en fournissant des informations contextuelles sur l'environnement de l'utilisateur. Leur principal atout est la **non-intrusivité** : ils opèrent de manière transparente, sans requérir l'intervention ou le port de dispositifs par l'usager.

1.5 Processus de reconnaissance d'activités

L'objectif principal de l'étude de la reconnaissance des activités humaines est de créer un système intelligent capable de reconnaître l'activité (comme cuisiner). Ce système fonctionne selon le processus de reconnaissance d'activité présenté par la figure **suivante**. Nous décrivons à présent brièvement les différentes étapes de ce processus.

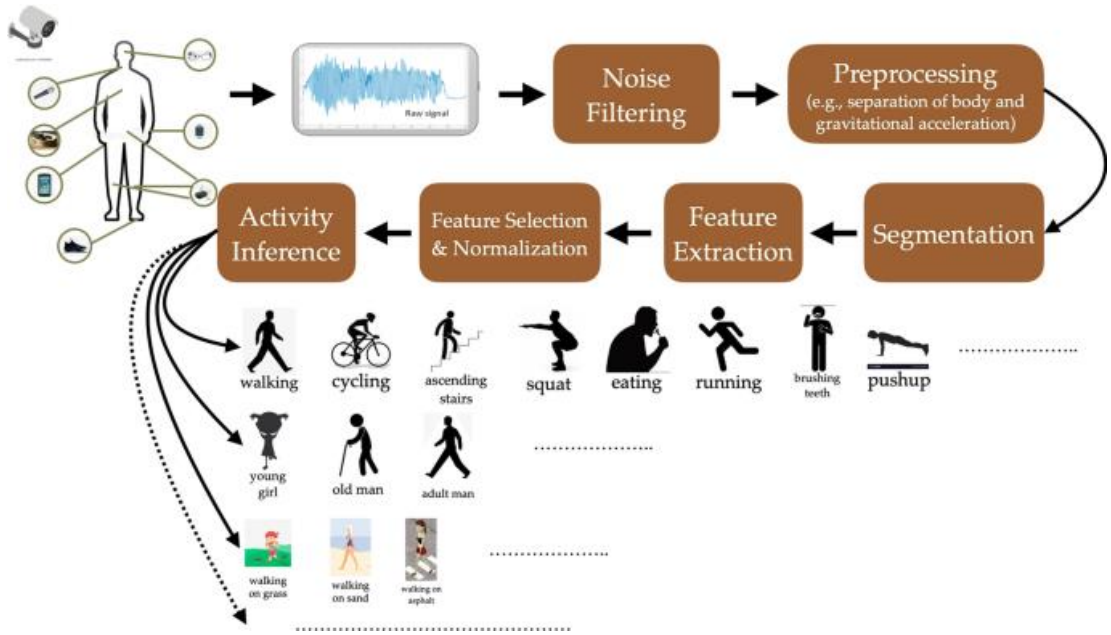

Figure I:3 Schéma du processus de reconnaissance d'activités humaines[39]

Explication des éléments de la figure 1.3 :

1. Sources de capteurs

Différents types de capteurs (caméras, accéléromètres, gyroscopes, montres connectées, smartphones...) collectent les données du corps humain en mouvement. Ces données sont la base de la reconnaissance d'activités.

2. Signal brut

Les capteurs produisent un signal brut contenant à la fois les informations utiles sur les mouvements et du bruit parasite.

3. Filtrage du bruit (Noise Filtering)

Cette étape vise à nettoyer le signal en supprimant les interférences et les perturbations inutiles pour ne conserver que les données exploitables.

4. Prétraitement (Preprocessing)

Il comprend des opérations comme la séparation entre l'accélération due au corps et celle due à la gravité. Cela permet de mieux isoler les caractéristiques pertinentes du mouvement.

5. Segmentation

Le flux de données est découpé en segments (fenêtres temporelles) correspondant à des portions cohérentes d'activités.

6. Extraction de caractéristiques (Feature Extraction)

Les segments sont analysés pour en extraire des descripteurs numériques (par exemple, moyenne, variance, énergie...) représentant les mouvements.

7. Sélection et normalisation des caractéristiques

Parmi toutes les caractéristiques extraites, seules les plus pertinentes sont conservées. Elles sont ensuite normalisées pour harmoniser l'échelle des données.

8. Inférence d'activité (Activity Inference)

Un modèle de classification utilise les caractéristiques sélectionnées pour reconnaître

l'activité en cours : marche, course, manger, monter les escaliers, etc.

La figure montre aussi la prise en compte de la variabilité selon **l'utilisateur** (enfant, adulte, personne âgée) ou **le contexte** (marcher sur sable, herbe ou asphalte), soulignant l'importance de la robustesse contextuelle du système.

1.5 Approches classiques d'apprentissage automatique

Les approches classiques de la reconnaissance d'activités humaines (HAR) utilisent des algorithmes supervisés pour classifier les vecteurs de caractéristiques extraits des données capteurs. Parmi les plus courantes, on retrouve **les k-NN et arbres de décision**, simples à mettre en œuvre mais efficaces dans des contextes à faible variabilité. Par exemple, **Khan et al. [8]** utilisent un classifieur k-NN avec une hiérarchie de décision, tandis que **Bao et Intille [9]** ont employé un arbre de décision de type C4.5 pour classifier des activités à partir de données d'accéléromètres.

Les SVM (machines à vecteurs de support) sont appréciés pour leur précision, notamment sur des jeux de données bien structurés. **Anguita et al. [20]** ont montré qu'un SVM linéaire appliqué aux données de smartphone pouvait distinguer efficacement six activités de base, tout en restant compatible avec les contraintes matérielles.

Les méthodes bayésiennes, comme le naïf Bayes ou les réseaux bayésiens, apportent une modélisation probabiliste utile, bien que parfois limitée par des hypothèses simplificatrices sur l'indépendance des caractéristiques.

Pour capter la dimension temporelle des activités, les modèles séquentiels comme **les HMM et CRF** permettent de mieux enchaîner les prédictions dans le temps, au prix d'un entraînement plus complexe.

Enfin, **les méthodes ensemblistes** (comme les forêts aléatoires) combinent plusieurs classifieurs pour améliorer la robustesse globale. **Zhu et Sheng [21]** ont proposé une approche fondée sur la fusion de plusieurs classifieurs dans un contexte de robotique d'assistance, tandis que **Berchtold et al. [25]** ont présenté un système modulaire activant différents modèles selon le contexte.

Ces approches, bien que performantes dans certains cas, restent sensibles à la qualité des caractéristiques extraites et peuvent avoir du mal à gérer des contextes variés ou complexes sans adaptation.

1.6 Apprentissage profond pour la reconnaissance d'activités

1.6.1 Automatisation de l'extraction de caractéristiques

Contrairement aux approches classiques, les méthodes d'apprentissage profond permettent d'apprendre automatiquement des représentations adaptées à la reconnaissance d'activités, grâce à

des réseaux de neurones profonds entraînés sur des données brutes. Cette avancée a été rendue possible par l'essor du calcul (GPU), l'accès à des jeux de données publics, et les progrès algorithmiques [22].

1.6.2 Utilisation des CNN pour l'analyse de signaux capteurs

Les réseaux convolutifs (CNN), bien connus en vision par ordinateur, sont capables de détecter des motifs locaux dans les signaux sensoriels (comme l'accélération) sans nécessiter d'extraction manuelle de caractéristiques. Ils surpassent souvent les méthodes traditionnelles sur données brutes [22].

1.6.3 Modélisation temporelle avec les RNN / LSTM

Les réseaux récurrents, notamment LSTM et GRU, permettent de capter les dépendances temporelles entre les frames de données. Ils sont particulièrement efficaces pour distinguer des activités dynamiquement différentes, mais visuellement proches [Singh et al. (2017), cité par Khallef].

1.6.4 Architectures hybrides CNN–RNN

La combinaison CNN + LSTM, comme le modèle DeepConvLSTM proposé par Ordóñez et Roggen (2016), a permis d'obtenir d'excellents résultats sur plusieurs bases de HAR. Ces architectures tirent parti à la fois de la vision spatiale des CNN et de la mémoire temporelle des RNN [22].

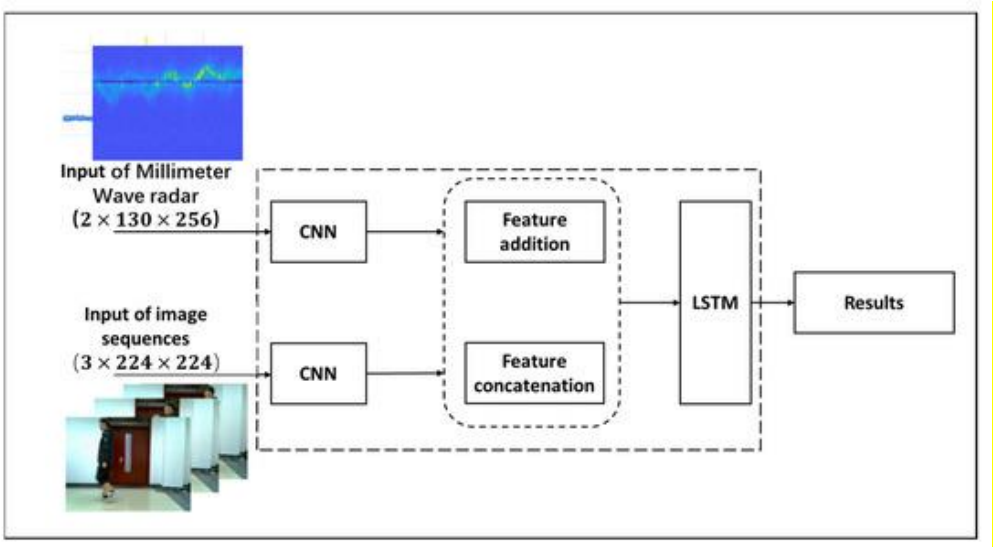

Figure I:4 Architecture CNN–LSTM multimodale pour la reconnaissance d'activités[40]

1.6.5 Avantages du deep learning en HAR

L'apprentissage profond améliore les performances, réduit le besoin d'ingénierie de caractéristiques, et facilite la généralisation. Il ouvre aussi des perspectives en apprentissage non supervisé (autoencodeurs) et en transfert de modèles pré-entraînés [Nweke et al. (2018), Wang et al. (2019) – voir [22]].

1.6.6 Limites et défis

Ces modèles nécessitent beaucoup de données annotées, sont plus coûteux en calcul et souvent peu interprétables. Cela limite leur usage dans certains contextes embarqués ou sensibles. Des recherches portent sur l'optimisation et la combinaison avec des méthodes classiques pour contourner ces limites [22].

1.7 Défis de la reconnaissance d'activités humaines

1.7.1 Robustesse et généralisation des modèles

Les systèmes de HAR peinent à maintenir de bonnes performances en dehors des environnements contrôlés. La variabilité des utilisateurs (morphologie, style, rythme) et des contextes rend la tâche complexe. Un modèle entraîné sur un petit échantillon devient souvent moins fiable face à un nouvel utilisateur ou un changement de capteur. Des approches comme le **transfer learning** ou l'**apprentissage fédéré** sont explorées pour personnaliser les modèles sans devoir tout réentraîner à chaque fois.

1.7.2 Reconnaissance d'activités complexes

Si les actions simples (marcher, s'asseoir) sont bien identifiées, les activités complexes ou enchaînées (comme cuisiner ou faire du sport en groupe) restent difficiles à détecter. Elles demandent de comprendre des séquences d'actions élémentaires, parfois avec interactions entre personnes ou objets. Pour cela, il faut recourir à des **capteurs supplémentaires** (caméras, objets instrumentés) et à des **modèles hiérarchiques ou sémantiques** capables de raisonner à plusieurs niveaux d'abstraction.

1.7.3 Mise à l'échelle et standardisation

Les résultats des systèmes de HAR sont souvent peu comparables à cause de jeux de données différents, de protocoles variés et de métriques non uniformes. La standardisation est donc nécessaire, avec des bases de données ouvertes comme le **UCI HAR Dataset (Anguita et al. [20])** ou **PAMAP, Opportunity, CASA**, et des indicateurs communs (comme le F1-score). **Wang et al. [22]**

soulignent l'importance de partager les modèles et de définir des protocoles expérimentaux reproductibles. De plus, le passage à grande échelle soulève des défis en matière de calcul et d'infrastructure, notamment pour traiter des flux de données continus et multi-utilisateurs en temps réel.

1.7.4 Consommation d'énergie et autonomie

Les systèmes embarqués souffrent de contraintes énergétiques fortes. Pour durer plus longtemps, il faut optimiser la fréquence d'échantillonnage, faire du traitement local ou transmettre uniquement des données résumées. **Huang et al. [23]** proposent des mécanismes d'**échantillonnage adaptatif** pour réduire la consommation sans sacrifier la performance. **Vergara-Laurens et Labrador [24]** explorent la **compression et la sélection intelligente des données** pour limiter la transmission vers le cloud tout en gardant l'essentiel. L'usage de **modèles allégés** ou quantifiés est aussi une piste pour les systèmes embarqués.

1.7.5 Vie privée et acceptabilité

La collecte de données personnelles pose des risques pour la vie privée : comportement, état de santé, ou identité peuvent être déduits. Il est essentiel de protéger les utilisateurs par des techniques comme l'anonymisation, le chiffrement, ou le traitement local sans cloud. **Huang et al. [23]** et **Vergara-Laurens et Labrador [24]** proposent des cadres garantissant la **confidentialité des données**, par exemple en ne partageant que des niveaux d'activité agrégés ou en ajoutant du bruit pour masquer certains détails. Le respect du RGPD et l'acceptation sociale sont des conditions indispensables à l'adoption des systèmes HAR, notamment dans les contextes sensibles (santé, travail, surveillance).

1.8 Conclusion

Ce chapitre a offert une vue d'ensemble du domaine de la reconnaissance d'activités humaines (HAR), en expliquant ses objectifs, ses applications et les technologies utilisées. Il a présenté les différentes sources de données — capteurs portés, capteurs embarqués et capteurs ambiants — et souligné l'intérêt d'une approche multimodale pour surmonter les limites propres à chaque type de capteur. Le processus de traitement des données, depuis leur acquisition jusqu'à la classification, a été détaillé, mettant en lumière les méthodes classiques d'apprentissage automatique comme les SVM ou k-NN, qui bien qu'efficaces dans certains cas, montrent aujourd'hui des limites face à la complexité croissante des données HAR.

L’introduction de l’apprentissage profond, avec les réseaux CNN et LSTM, marque une avancée majeure dans le domaine en permettant une extraction automatique des caractéristiques et une meilleure modélisation temporelle. Cependant, des défis persistent, notamment en termes de généralisation des modèles, de reconnaissance d’activités complexes ou de respect de la vie privée. Ces éléments constituent les bases sur lesquelles repose la suite du mémoire. Le chapitre suivant viendra approfondir certains de ces enjeux, en lien avec notre propre démarche expérimentale. La compréhension des fondements exposés ici sera essentielle pour appréhender les développements méthodologiques et les résultats à venir dans le cadre de notre travail de recherche.

Chapitre 2 : Reconnaissance d'activité humaine par Deep Learning et apprentissage par transfert

II. Chapitre 2 : Reconnaissance d'activité humaine par Deep Learning et apprentissage par transfert

2.1 Introduction

La reconnaissance d'activités humaines vise à identifier automatiquement les actions ou comportements d'une personne à partir de données issues de capteurs (capteurs portés, caméras, etc.). Dans le chapitre 1, nous avons présenté les fondements de la HAR, ses domaines d'application (santé, surveillance, sport, etc.) ainsi que les approches traditionnelles basées sur l'extraction manuelle de caractéristiques et les algorithmes d'apprentissage classiques[1][2][6]. Ces méthodes classiques ont permis de premières solutions fonctionnelles, mais montrent leurs limites face à la complexité et à la variabilité des activités humaines. Depuis quelques années, l'essor de **l'apprentissage profond** a révolutionné ce domaine [22][25] en proposant des modèles capables d'apprendre automatiquement des représentations optimales des données et d'obtenir des performances largement supérieures à l'état de l'art précédent. Par ailleurs, afin de répondre au défi du manque de données annotées et de la généralisation des modèles, **l'apprentissage par transfert** (**transfer learning**) émerge comme une stratégie prometteuse pour réutiliser des connaissances acquises sur un problème source et les appliquer à un problème cible en HAR.

Ce chapitre propose une revue de certaines approches de reconnaissance d'activité basées sur le Deep Learning, avec un accent particulier sur l'apprentissage par transfert. Nous commencerons par comparer brièvement les approches classiques et profondes en HAR. Nous passerons ensuite en revue les principaux types de réseaux de neurones utilisés pour la HAR : les réseaux de neurones convolutifs (CNN), les réseaux récurrents (RNN) ainsi que leurs variantes LSTM/GRU, et les nouveaux modèles à base de Transformeurs. Le potentiel de ces architectures pour la HAR sera discuté, en soulignant comment elles surpassent les méthodes traditionnelles. Ensuite, une section approfondie sera dédiée à l'**apprentissage par transfert**. Afin d'illustrer concrètement ces concepts, nous présenterons une **étude de cas** tirée d'un travail récent : la réalisation d'un modèle **CNN–LSTM** pour la reconnaissance d'actions humaines à partir de vidéos. En parallèle, nous passerons en revue quelques **jeux de données de référence** utilisés couramment pour évaluer les performances

des modèles de HAR. Enfin, nous discuterons des **défis ouverts, des limitations actuelles et des perspectives** du domaine.

2.2 Approches classiques vs. Deep Learning en HAR

Historiquement, les systèmes de HAR étaient construits selon une chaîne de traitement en plusieurs étapes bien définies[1][5][11]: (1) collecte des données brutes depuis les capteurs, (2) pré-traitement et segmentation en fenêtres temporelles, (3) extraction manuelle de *features* (descripteurs caractéristiques de l'activité) sur chaque fenêtre [14], puis (4) classification de ces vecteurs de caractéristiques via un algorithme d'apprentissage supervisé classique (par exemple, k-plus proches voisins, SVM, forêts aléatoires, etc.). Ce pipeline nécessite une expertise importante pour choisir et calculer les bonnes caractéristiques du signal. Par exemple, on pouvait utiliser en HAR classique des attributs tels que la moyenne et l'écart-type d'un signal d'accélération sur une fenêtre [8][9], son énergie spectrale dans différentes bandes de fréquence, le nombre de pics détectés, des coefficients de modèle autorégressif, etc. La qualité du système reposait en grande partie sur la pertinence de ces caractéristiques *hand-crafted*. Or, déterminer manuellement des *features* optimales pour distinguer de multiples activités s'avère délicat, d'autant plus que les mêmes capteurs peuvent servir à reconnaître des actions très variées (gestes fins vs locomotion globale, activités de cuisine vs sport, etc.).

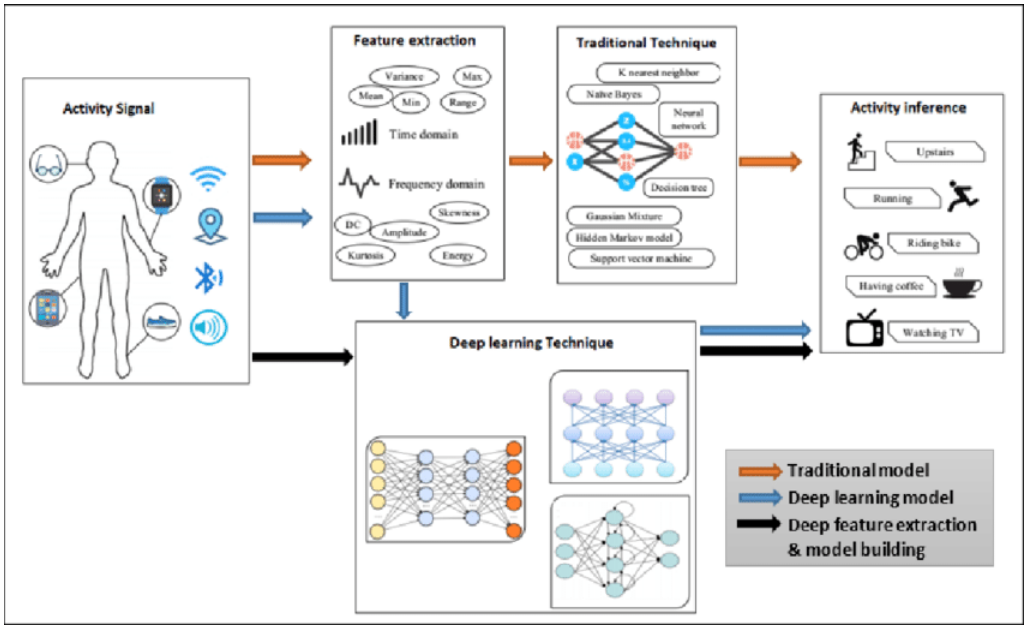

Figure II:1 schéma comparatif de l'Approche classique et le Deep learning[41]

L'**apprentissage profond** propose d'*apprendre automatiquement* la représentation optimale des données à partir de grands volumes de données étiquetées. On entraîne des réseaux de neurones multicouches qui découvrent eux-mêmes quelles structures dans les données permettent de prédire l'activité. En traitement d'images, cette approche a été popularisée par les succès des CNN qui apprennent à détecter des motifs visuels de bas niveau (bords, textures) puis de plus haut niveau (objets, postures) directement à partir des pixels. En HAR sur signaux capteurs, de même, des réseaux profonds peuvent apprendre sur les données brutes les motifs temporels caractéristiques d'une activité (par exemple, la signature d'accélération d'un pas, d'un saut, d'une chute) sans passer par un calcul explicite de statistiques temporelles ou fréquentielles. Cette approche a deux avantages majeurs :

- **Meilleure performance** : les modèles profonds, en optimisant de bout en bout la transformation des données brutes en prédiction d'activité, atteignent des performances supérieures à celles des méthodes à base de features manuelles. Par exemple, Wang *et al.* (2019) [22] rapportent dans leur enquête que de nombreux travaux ont dépassé l'état de l'art classique en employant des CNN 1D sur les signaux bruts là où les features conçues manuellement plafonnaient en performance.
- **Généralisation et adaptabilité** : un réseau profond peut en théorie s'adapter à une grande variété de données et de contextes s'il est entraîné sur une base suffisamment diversifiée, là où un système figé de caractéristiques aurait du mal à couvrir tous les cas.

En pratique, bien sûr, les réseaux profonds nécessitent beaucoup de données pour être entraînés efficacement, ce qui peut être un frein en HAR. De plus, ils introduisent des *boîtes noires* difficiles à interpréter et peuvent être coûteux en calcul.

Dans les sections qui suivent, nous détaillons les principales architectures de réseaux de neurones exploitées pour la HAR profonde, puis nous aborderons l'apprentissage par transfert qui vise notamment à tirer parti de ces modèles avancés même lorsque les données disponibles sont limitées.

2.3 Architectures de réseaux de neurones pour la HAR

Plusieurs types d'architectures de réseaux de neurones profonds ont démontré leur efficacité pour la reconnaissance d'activités. Les plus courants sont : les **réseaux de neurones convolutifs (CNN)**, les **réseaux de neurones récurrents (RNN)** et, en particulier,

leurs variantes LSTM/GRU, et plus récemment les **Transformers** basés sur des mécanismes d'attention. Chacune de ces familles de modèles a ses spécificités et adresse des aspects complémentaires de la HAR (extraction de motifs locaux, modélisation du temporel, capture de dépendances à long terme, etc.).

2.3.1 Réseaux convolutifs (CNN) pour l'extraction de motifs

Les **réseaux de neurones convolutifs (Convolutional Neural Networks, CNN)** sont largement utilisés en vision par ordinateur pour extraire automatiquement des caractéristiques locales d'images, et ont été adaptés avec succès aux données temporelles de la HAR. Le principe d'un CNN est d'appliquer des **filtres de convolution** qui vont détecter la présence de motifs caractéristiques dans les données d'entrée (par ex. arêtes et textures dans une image, oscillations ou pics dans un signal). Chaque filtre produit en sortie une **carte de caractéristiques** activée aux endroits où le motif reconnu apparaît. En empilant plusieurs couches convolutionnelles, alternées éventuellement de sous-échantillonnage (pooling), le réseau parvient à apprendre des caractéristiques de plus en plus abstraites et invariantes. Dans le contexte HAR, on distingue deux principales applications des CNN :

– **CNN sur signaux de capteurs (CNN 1D ou 2D)** : Pour les données issues de capteurs portables (accéléromètres, gyroscopes, etc.), les entrées sont des séries temporelles multi-variées. On peut utiliser des CNN **1D** qui glissent un filtre sur l'axe temporel afin de détecter des motifs dans la séquence de mesures brutes. Par exemple, un CNN 1D appliqué aux accélérations triaxiales peut apprendre à détecter une signature d'oscillation correspondant à la marche, ou un motif de secousse indiquant une chute, sans aucune intervention manuelle sur les données. Une alternative consiste à représenter les données capteurs sous forme d'**images** (par exemple, images spectrogrammes temps-fréquence, ou matrices capteur×temps) puis d'appliquer des CNN 2D classiques. Par exemple, Wang *et al.* (2019) révèlent que l'utilisation de CNN sur des signaux bruts a permis de gagner plusieurs points de pourcentage de précision par rapport aux meilleurs algorithmes classiques sur divers benchmarks de HAR. Les CNN sont particulièrement efficaces pour capturer des motifs locaux stationnaires (par ex. le patron périodique d'une foulée, le pic d'accélération d'un saut). En revanche, ils prennent moins bien en compte les dépendances temporelles à long terme (ce que fait un RNN).

– **CNN sur données vidéo (2D et 3D CNN)** : En HAR basée vision (détection d'actions dans des vidéos), les CNN se sont imposés pour extraire des *features* visuelles spatiales sur les images. Un **CNN 2D** entraîné pour la classification d'images peut servir de *backbone* pour analyser chaque

frame de la vidéo et y reconnaître, par exemple, des objets ou poses clés associés à l'action (ballon, véhicule, position du corps). Cependant, classifier une vidéo en traitant indépendamment chaque image ignore la dimension temporelle de l'action. Pour intégrer le temps, une approche consiste en des **CNN 3D** qui réalisent des convolutions spatio-temporelles (x, y, t) afin d'extraire des caractéristiques jointes sur plusieurs frames consécutives. Un CNN 3D peut détecter des motifs dynamiques *mouvants* dans la vidéo (un geste de la main se déplaçant). L'avantage est d'intégrer l'information temporelle directement dans le réseau ; l'inconvénient est un coût de calcul bien plus élevé qu'un CNN 2D classique, car le filtre balaye aussi l'axe temporel. En pratique, les CNN 3D préentraînés sur de grands corpus vidéo (comme **Kinetics-400** contenant 400 actions humaines différentes) ont démontré d'excellentes performances pour la reconnaissance d'actions dans des vidéos génériques. Par exemple, le modèle I3D de Carreira, J., & Zisserman, A. (2017). [31], pré-entraîné sur Kinetics, atteint environ 98% de précision top-1 sur le dataset UCF-101 (101 actions) et plus de 74% sur **HMDB51** (51 actions) en test. Notons qu'une approche alternative aux CNN 3D est d'utiliser des **architectures hybrides CNN+RNN** (voir section suivante) ou des méthodes de **fusion** : par exemple, les modèles *Two-Stream CNN* (Simonyan & Zisserman, 2014) [32] qui combinent un CNN traitant les images RGB et un autre traitant des images de flot optique pour capturer le mouvement. Quelle que soit la variante, les réseaux convolutifs constituent un **bloc de base incontournable** pour la HAR profonde, fournissant des briques de reconnaissance de motifs spatiaux ou temporels courts.

2.3.2 Réseaux récurrents (RNN, LSTM, GRU) pour la modélisation temporelle

Les **réseaux de neurones récurrents (Recurrent Neural Networks, RNN)** sont conçus spécifiquement pour traiter des données séquentielles en modélisant les dépendances temporelles. Les RNN analysent une séquence pas à pas en conservant un **état interne** qui s'actualise à chaque nouvelle entrée, ce qui leur permet de **mémoriser** l'historique du signal. Pour la reconnaissance d'activités, cette propriété est cruciale : de nombreuses activités ne peuvent être identifiées qu'en considérant l'évolution du signal sur une certaine durée, plutôt qu'à un instant isolé. Par exemple, la posture « debout » versus l'action « faire des sauts sur place » peuvent avoir des instantanés de données similaires (pas de mouvement brusque la plupart du temps), mais se distinguent par la succession temporelle des événements (absence de saut vs répétition de sauts). Un modèle RNN bien entraîné peut saisir ces nuances temporelles, là où un modèle instantané ou purement convolutionnel risquerait de confondre les deux.

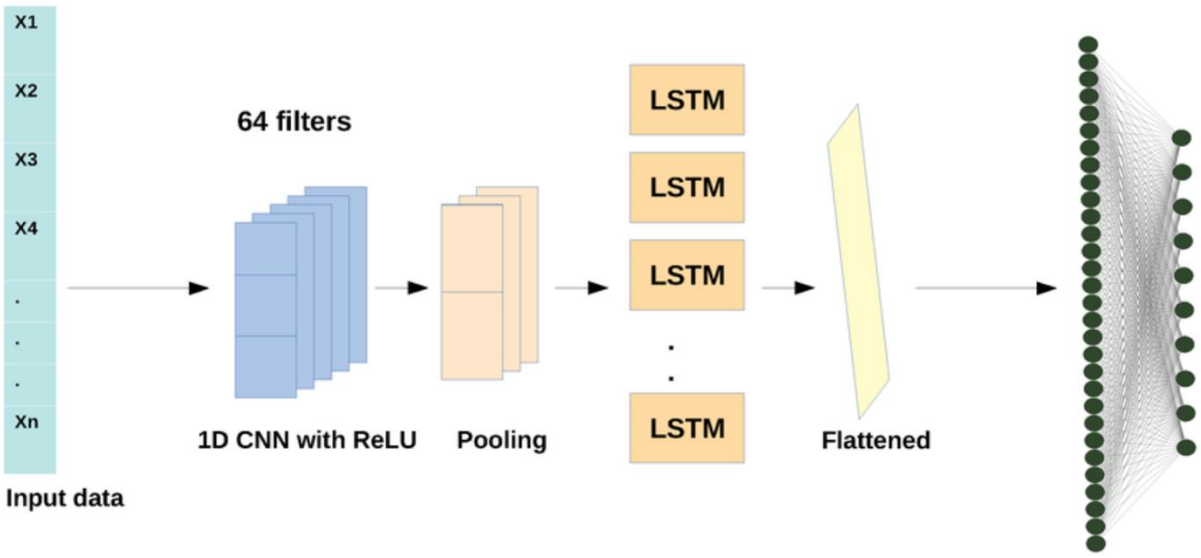

Figure II:2 Schéma d'un modèle hybride 1D CNN + LSTM[42]

Les RNN « simples » (dits *vanilla*) souffrent cependant de limitations lorsqu'il s'agit d'apprendre des dépendances à long terme, en raison du problème bien connu de l'**évanouissement/explosion du gradient**. C'est pourquoi ont été introduites des architectures récurrentes améliorées, en particulier les **LSTM (Long Short-Term Memory)** et les **GRU (Gated Recurrent Unit)** :

– Les réseaux **LSTM** incorporent des mécanismes de portes (entrée, oubli, sortie) qui régulent le flux d'informations dans le *cell state*, permettant de conserver des informations pertinentes sur de longues séquences tout en oubliant les informations devenues inutiles. Les LSTM ont démontré une capacité remarquable à apprendre des patterns temporels complexes sur de longues durées.

– Les **GRU**, quant à eux, sont une variante plus légère des LSTM qui fusionnent certaines portes et simplifient la cellule, tout en offrant des performances comparables dans de nombreux cas.

En pratique, LSTM et GRU sont largement utilisés en HAR pour traiter les séries temporelles de capteurs ou les séquences d'images, afin d'accumuler une **mémoire** de l'activité en cours. Par exemple, **Hammerla et al. (2016) [33]** ont montré qu'un LSTM entraîné sur des données de smartphones pouvait surpasser nettement des approches classiques pour reconnaître des activités quotidiennes, grâce à sa capacité d'intégration temporelle. De même, une étude comparative citée par Hammerla et al. (2016) [33]. rapporte qu'un modèle LSTM bien configuré dépasse souvent un modèle purement CNN

pour la reconnaissance de séquences d'activités, le RNN exploitant les relations temporelles là où le CNN extrait surtout des motifs locaux instantanés.

Dans le contexte de la HAR, les réseaux récurrents permettent notamment de **lisser** les prédictions dans le temps et d'éviter des détections incohérentes (par ex. éviter de classifier un court instant comme « chute » si le reste de la séquence indique une marche normale). Ils sont également utiles pour segmenter des activités ou détecter des transitions. Les GRU, plus simples, sont parfois privilégiés pour des applications embarquées en raison de leur moindre coût de calcul par rapport aux LSTM, tout en conservant l'essentiel des capacités de mémoire.

– **Combinaison CNN–RNN :** Il est fréquent de combiner les atouts des CNN et des RNN dans des architectures hybrides pour la HAR. Les CNN excellent à extraire des **caractéristiques spatiales locales** (par ex. un motif de signal sur quelques centièmes de seconde), tandis que les RNN excellent à modéliser la **dynamique globale** de la séquence. La **combinaison CNN+LSTM** est ainsi devenue un **standard de fait** pour la HAR profonde, notamment dans le cas des capteurs portables ou des vidéos. Une architecture typique, proposée par Ordóñez et Roggen (2016) [25] sous le nom de *DeepConvLSTM*, consiste à empiler plusieurs couches convolutionnelles (pour extraire des features locales sur des segments courts) suivies de couches LSTM qui traitent la séquence de segments et apprennent les transitions et dépendances temporelles à plus long terme. Ce modèle a obtenu d'excellents résultats sur plusieurs jeux de données de HAR basés sur des capteurs inertiels. De même, dans le traitement vidéo, des modèles **LRCN (Long-term Recurrent Convolutional Network)** (Donahue et al., 2015) [36] combinent un CNN (ex. ResNet, VGG) pour encoder chaque frame visuellement et un LSTM pour agréger ces encodages sur la durée de la vidéo – cette approche a permis d'obtenir d'excellents taux de reconnaissance sur des corpus comme UCF101 ou HMDB51 tout en capturant la dynamique temporelle absente d'un CNN 2D pur.

En somme, les réseaux récurrents (surtout LSTM/GRU) sont un **complément indispensable** des CNN pour la HAR dès que l'ordre temporel ou la durée de l'action influent sur la classification.

2.3.3 Transformers et mécanismes d'attention

Un développement plus récent dans l'apprentissage profond, qui commence à toucher la HAR, est l'utilisation des modèles à base de **Transformers** et de **self-attention**. Les Transformers, initialement proposés pour le traitement du langage naturel (Vaswani et al., 2017) [30], ont démontré une capacité hors pair à capturer des dépendances à longue

portée dans des séquences, sans passer par une récursion temporelle comme les RNN. Au lieu de cela, un Transformer utilise des mécanismes d'**attention multi-têtes** qui permettent de pondérer l'importance relative de différentes parties de la séquence lorsqu'on encode un élément donné. Appliqué aux séquences temporelles, cela signifie qu'un modèle Transformer peut regarder simultanément l'ensemble de la séquence de capteurs ou de frames vidéo et apprendre quels moments sont pertinents les uns par rapport aux autres pour la tâche de classification.

Dans le domaine de la **vision par ordinateur**, les **Vision Transformers (ViT)** ont récemment montré des performances comparables ou supérieures aux CNN sur des tâches de classification d'images, en particulier avec de très grands volumes de données pour l'entraînement. Cette réussite a naturellement conduit à explorer leur utilisation pour la reconnaissance d'actions dans les vidéos, où la capacité à modéliser efficacement les relations spatiales *et temporelles* sur de longues séquences est un atout.

Plusieurs travaux récents proposent des architectures Transformer ou hybrides CNN-Transformer pour la HAR vidéo. Par exemple, les modèles ViViT (Arnab et al., 2021) [34] ou TimeSformer (Bertasius et al., 2021) **[35]** appliquent directement un Transformer aux patchs des frames vidéo, tandis que d'autres intègrent des blocs Transformers au-dessus de features extraites par CNN (architecture hybride).

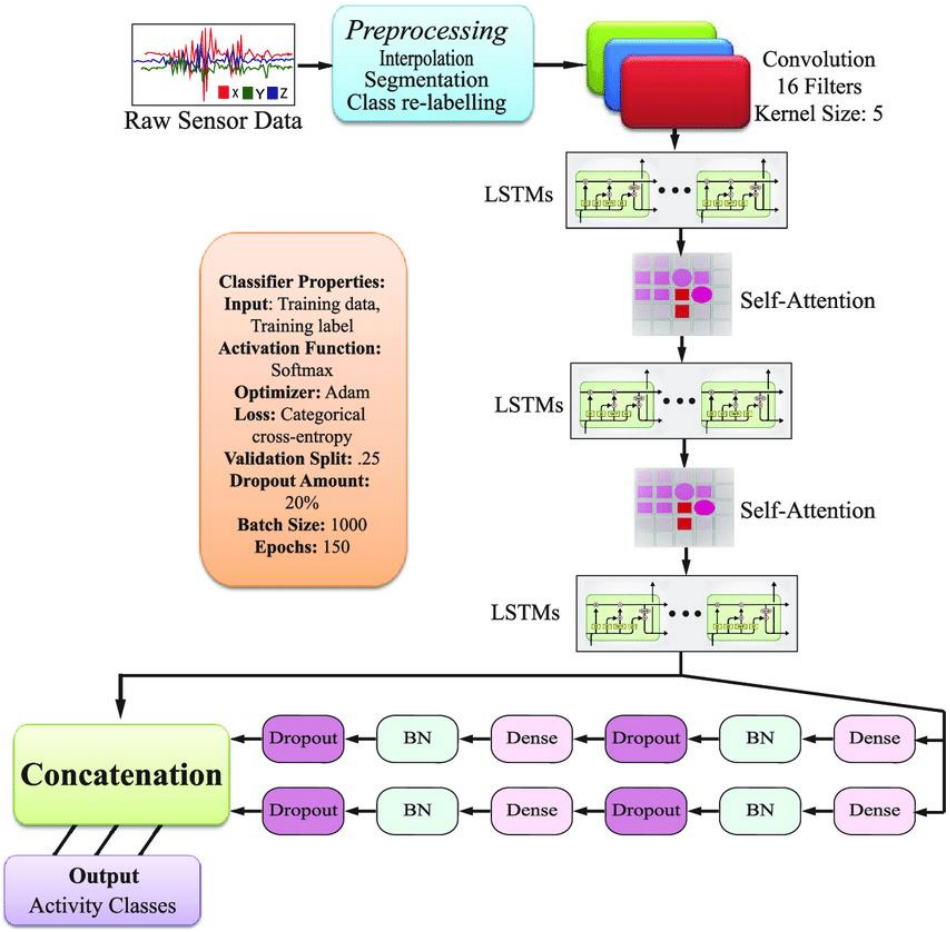

Figure II:3 Schéma d'un modèle hybride 1D CNN + LSTM[42]

En HAR sur données capteurs, des chercheurs ont expérimenté des modèles à attention pour, par exemple, **apprendre à peser différemment les capteurs ou les instants** dans le calcul de la prédiction d'activité – ceci peut aider à ignorer des segments non informatifs du signal ou à accorder plus d'importance à certains moments clés. L'avantage des Transformers est leur **flexibilité** à capturer des interactions à long terme et entre n'importe quelles positions de la séquence, là où un LSTM, malgré sa mémoire, reste fondamentalement séquentiel. En contrepartie, les Transformers requièrent énormément de données pour être entraînés correctement. Dans le cadre de la HAR, où les jeux de données sont plus restreints que dans le langage ou la vision, cela pose un défi. Néanmoins, avec des corpus comme **Kinetics-400/600** (300k–500k vidéos), il a été possible d'entraîner de grands Transformers pour la reconnaissance d'actions, qui atteignent désormais l'état de l'art. Par exemple, **Arnab et al. (2021)** rapportent qu'un ViT entraîné sur Kinetics-400 obtient ~80% de précision top-1 sur Kinetics et ~65% sur HMDB51 sans aucun CNN, ce qui rejoint

les scores des meilleurs CNN 3D tout en ouvrant la voie à des améliorations via des modèles encore plus larges ou pré-entraînés sur des données multimodales.

En résumé, les **Transformers** représentent la nouvelle génération d'architectures susceptibles d'impacter la HAR, en particulier pour les données séquentielles complexes. Dans un cadre plus large, cela s'inscrit dans la tendance à exploiter des modèles profonds toujours plus puissants et génériques, quitte à recourir à l'**apprentissage par transfert** pour les adapter à des tâches spécifiques de HAR avec moins de données – ce que nous abordons dans la section suivante.

2.4 Apprentissage par transfert pour la HAR

Malgré les progrès des réseaux profonds, un obstacle majeur demeure en reconnaissance d'activités : la **disponibilité de données annotées** en quantité suffisante pour entraîner ces modèles complexes. Les données de HAR (mesures de capteurs portables spécifiques, vidéos d'actions ciblées) nécessitent des collectes expérimentales coûteuses et un étiquetage manuel fastidieux. En outre, même lorsqu'un modèle est entraîné efficacement sur un certain jeu de données, sa **généralisation** à d'autres contextes n'est pas garantie : changer de population, de capteur ou de domaine d'application peut dégrader fortement les performances. L'**apprentissage par transfert** (transfer learning) vise précisément à **transférer les connaissances** d'un modèle appris sur une tâche/source donnée vers une nouvelle tâche/cible, afin de *réutiliser* au maximum ce qui a été appris et de réduire les besoins en données et en entraînement sur la cible. C'est une approche particulièrement cruciale en HAR, où l'on cherche par exemple à **adapter un classifieur d'activité à un nouvel utilisateur** sans repartir de zéro, ou à **exploiter un grand corpus générique** (ex. vidéos YouTube annotées) pour initialiser un modèle devant être appliqué à un corpus plus restreint (ex. vidéos filmées dans un contexte particulier).

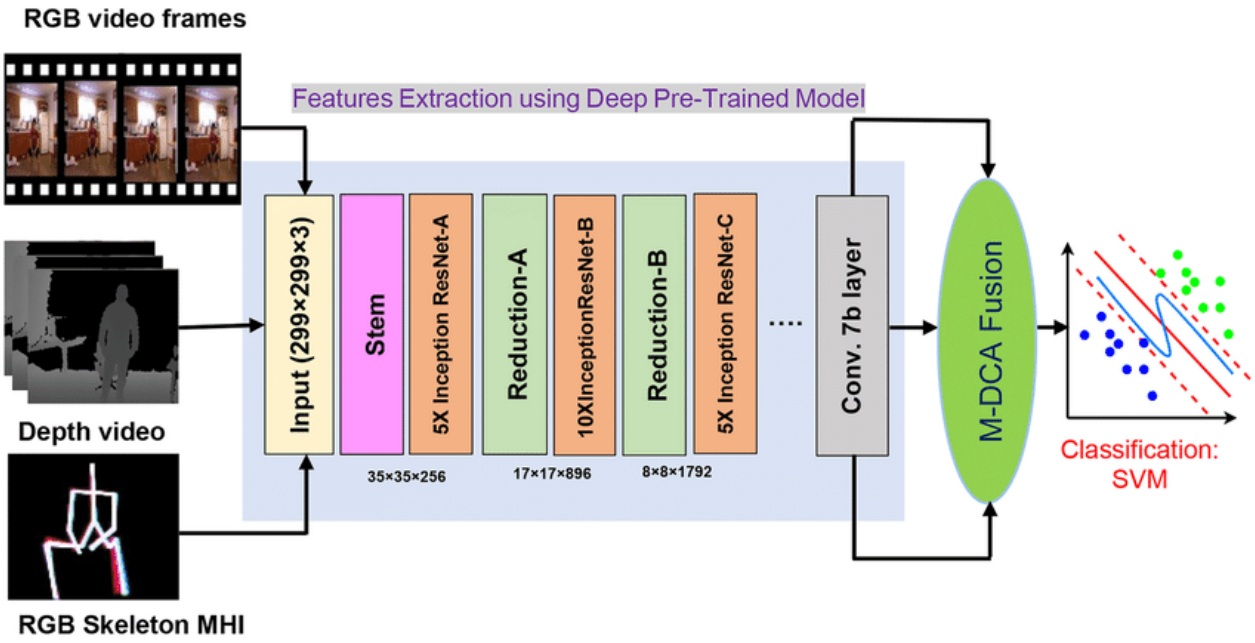

Figure II:4 Workflow de transfer learning adapté à la HAR[43]

2.4.1 Principe et méthodes de l'apprentissage par transfert

Dans sa forme la plus courante, l'apprentissage par transfert consiste à **pré-entraîner** un modèle profond sur une tâche source pour laquelle on dispose de beaucoup de données, puis à **affiner** ce modèle sur la tâche cible disposant de moins de données. Les couches apprises sur la source servent de *point de départ* (initialisation) et l'on espère qu'elles contiennent des détecteurs de motifs ou des représentations utiles qui n'auront pas besoin d'être réapprises à partir de zéro sur la cible. Concrètement, on distingue souvent deux méthodes :

- le **transfert de caractéristiques** (*feature transfer*) où l'on conserve le modèle pré-entraîné inchangé et on ne ré-entraine que la dernière couche de classification sur les données cibles (le réseau sert alors d'extracteur de features génériques, et seul le classifieur final est ajusté à la nouvelle tâche) ;
- (2) le **fine-tuning** complet ou partiel, où l'on ré-entraine le modèle pré-entraîné sur les données cibles, en autorisant les poids à s'ajuster (souvent avec un taux d'apprentissage réduit) – on peut choisir de ne réentrainer que certaines couches (par ex. les couches hautes « spécialisées ») et de geler les couches basses si le dataset cible est petit, ou bien tout affiner si le dataset cible est suffisamment grand.

L'apprentissage par transfert peut également impliquer des techniques de **domaine adaptation** lorsque la tâche est la même, mais que la distribution des données diffère (par

ex. capteur différent, conditions différentes) : on cherche alors à réduire le *décalage de domaine* (*domain shift*) par divers moyens (calage statistique des distributions, entraînement adversarial pour apprendre une représentation invariante de domaine, etc.). Par ailleurs, dans le cas de la HAR, on voit émerger des approches de **transfert multi-sources** (tirer parti de plusieurs jeux de données sources à la fois) ou de **transfert personnalisé** (adapter un modèle global à chaque individu). Par exemple, on peut entraîner un modèle global sur les données de nombreux utilisateurs, puis affiner quelques couches pour les données d'un nouvel utilisateur cible – ceci permet de **personnaliser** le classifieur d'activité aux habitudes de l'individu, améliorant grandement la précision.

En résumé, le *processus* typique de transfert en HAR pourrait être : prendre un réseau CNN ou CNN+LSTM pré-entraîné sur un grand dataset (source) pour la partie CNN, puis le fine-tuner sur un dataset plus réduit (cible), obtenant ainsi de meilleures performances que si on avait entraîné directement sur le petit dataset cible.

2.4.2 Avantages du transfert learning en HAR

L'apprentissage par transfert présente plusieurs **avantages déterminants** dans le contexte de la HAR :

- **Réduction des besoins en données annotées** : C'est le bénéfice principal. En transférant un modèle pré-entraîné, on peut obtenir de bonnes performances sur la cible avec beaucoup moins de données que ce qu'il aurait fallu pour entraîner un modèle de zéro. Ceci est crucial, car l'acquisition de données HAR étiquetées est coûteuse (impliquer des volontaires, enregistrer des capteurs, annoter chaque intervalle d'activité).

- **Amélioration de la généralisation** : Un modèle pré-entraîné sur une source diversifiée aura appris des représentations plus robustes et générales. En HAR, cela peut être vu avec des transferts entre domaines proches : par ex. un modèle de reconnaissance de gestes entraîné sur un ensemble de sujets peut mieux capturer l'essence du geste et éviter de se focaliser sur la morphologie d'un individu donné ; le fine-tuner sur un nouvel individu permettra d'adapter légèrement le modèle tout en conservant cette base générique, menant à une meilleure robustesse inter-personnes.

- **Gain de temps de calcul** : En pratique, entraîner un gros réseau profond est long et coûteux. Le ré-entraînement partiel d'un modèle existant (fine-tuning) est bien plus rapide que l'entraînement *from scratch*. Ainsi, le transfert learning permet d'**accélérer** le développement de modèles de HAR performants.

- **Reproductibilité et standardisation** : L'usage de modèles pré-entraînés (parfois disponibles publiquement) encourage la reproductibilité des expériences et offre des *baselines* solides. En HAR, on voit émerger des modèles pré-entraînés sur de grands jeux de données de capteurs (ou auto-entraînés de façon non supervisée sur des grandes quantités de données non étiquetées) qui peuvent servir de base commune.

- **Adaptation aux nouvelles classes ou tâches** : Le transfert learning permet d'**initialiser un modèle pour de nouvelles classes d'activité**. Par exemple, si on a un classifieur entraîné à reconnaître la marche, la course, on peut vouloir ajouter une nouvelle classe « monter les escaliers ». Plutôt que de réentraîner un nouveau modèle sur toutes les classes, on peut prendre l'ancien modèle et le *fine-tuner* en ajoutant cette classe avec quelques données d'entrainement pour elle. Le modèle conservera ses connaissances sur les classes initiales tout en incorporant la nouvelle.

En somme, l'apprentissage par transfert fournit un cadre flexible et efficace pour exploiter les acquis d'un modèle dans un contexte nouveau, ce qui correspond bien aux besoins de la HAR où **chaque contexte ou utilisateur peut être légèrement différent**. Il convient toutefois de noter que ces avantages ne se concrétisent que si la tâche source et la tâche cible sont **suffisamment liées** (même type de données ou de structure de décision). Nous abordons ci-dessous ces limites et les façons de les atténuer.

2.4.3 Limites et défis du transfert learning en HAR

Malgré son intérêt, l'apprentissage par transfert en HAR présente **plusieurs limites** et doit surmonter des défis spécifiques :

- **Différences de domaine (hétérogénéité)** : La HAR couvre une grande diversité de modalités de capteurs et de configurations. Un modèle pré-entraîné sur des vidéos ne sera pas directement applicable à des données de smartwatch ; un modèle entraîné avec un certain accéléromètre pourrait ne pas être optimal avec un accéléromètre d'un autre modèle ou placé à un autre endroit sur le corps. Le **décalage de domaine** (domain shift)

est un problème majeur : les caractéristiques des signaux peuvent changer d'un domaine à l'autre, induisant des différences dans l'espace des features appris. Comme le soulignent Sourish Dhekane *et al.* (2023), *« le plus grand défi technique du transfert en HAR est la différence d'espace de caractéristiques induite par des facteurs tels que différentes modalités de capteurs, emplacements, dispositifs, etc., entre les domaines » [29]*. Ce défi demande souvent d'incorporer des techniques de **normalisation/standardisation** des données ou des approches de **domaine adaptation** (par ex. entraîner le réseau à apprendre des représentations invariantes au domaine via une perte adversariale, ou calibrer le capteur cible pour imiter celui de la source).

- **Risque de sur-ajustement ou de « négatif transfer »** : Si le domaine source est trop différent ou la tâche source trop éloignée de la cible, le transfert peut être **contre-productif**. Le modèle pré-entraîné pourrait extraire des motifs qui ne sont pas discriminants pour la nouvelle tâche, voire nuisibles. Par exemple, un CNN pré-entraîné sur ImageNet (vision générale) sait détecter des centaines d'objets; si on l'utilise pour une tâche HAR très spécifique (comme reconnaître si une personne cuisine vs jardine en regardant la vidéo), certaines features complexes (détecteur de chien, de bâtiment) peuvent être inutiles et encombrer l'apprentissage. Il faut alors veiller à adapter l'architecture et disposer d'un minimum de données cibles pour *recalibrer* le réseau. Trouver le bon équilibre entre ce qu'on **conserve** du modèle source et ce qu'on **adapte** est un art délicat – trop geler de couches peut brider la spécialisation sur la cible, pas assez en geler peut entraîner un apprentissage biaisé ou instable sur peu de données.

- **Choix du modèle source** : Un problème connexe est de déterminer **quelle source** utiliser pour un transfert optimal. Idéalement, la source doit être « proche » de la cible. Par exemple, pour la HAR avec capteurs de smartphone, un modèle source entraîné sur un grand dataset comme **PAMAP2** (mesures inertielles sur activités variées) sera plus approprié qu'un modèle vision. Des travaux de recherche tentent de définir des métriques de **similitude de domaine** pour guider le choix du dataset source ou pondérer plusieurs sources (transfert multi-source).

- **Contraintes computationnelles et mémoire** : Les modèles profonds pré-entraînés sont souvent volumineux (ex : ResNet-50, Inception, etc.). En contexte embarqué ou mobile, où la HAR doit parfois se faire en temps réel sur un périphérique à ressources limitées,

charger un grand modèle pré-entraîné peut être problématique. On voit ainsi des recherches sur des réseaux profonds plus légers pour HAR (CNN 1D condensés, LSTM un peu plus petits) ou l'utilisation du transfert sur ces modèles allégés afin de conserver l'essentiel des performances tout en tenant la contrainte d'embarqué.

- **Évaluation et métriques** : En HAR, comparer équitablement les approches avec et sans transfert n'est pas trivial, car cela dépend du choix du split de données, etc. La communauté commence à standardiser certains protocoles, mais il reste des questions ouvertes sur **comment mesurer le gain du transfert** dans différents scénarios (sujets différents, appareils différents, activités différentes…). Par exemple, on peut vouloir mesurer la performance *avant* et *après* personnalisation d'un modèle global pour un utilisateur – cela nécessite des jeux de données avec une séparation claire des utilisateurs en entraînement vs test. Certains datasets publics permettent cela (Opportunity, PAMAP, etc.), et les travaux comme celui de Wang et al. (2018). encouragent le partage des modèles et une évaluation sur des bases communes pour objectiver les progrès.

En dépit de ces limites, l'apprentissage par transfert s'impose de plus en plus comme un **passage obligé** pour obtenir des modèles de HAR robustes et déployables. Dans la section suivante, nous illustrons l'efficacité du Deep Learning et du transfert sur un cas concret de HAR, en nous penchant sur une architecture hybride CNN–LSTM appliquée à la reconnaissance d'actions humaines dans des vidéos.

2.5 Étude de cas : reconnaissance d'activités humaines par une architecture CNN–LSTM (d'après BleedAI Academy 2024)

Pour concrétiser les concepts évoqués, nous présentons une **étude de cas pratique** basée sur l'article **"Human Activity Recognition using TensorFlow (CNN + LSTM)"** publié en 2024 sur BleedAI Academy par Anwar, T., Naeem, R., & Anjum, M. (2024) [28]. Cette étude vise la reconnaissance d'actions humaines dans des vidéos (HAR vision) en combinant un réseau convolutionnel et un réseau LSTM, et elle met en œuvre un **transfert d'apprentissage** à deux niveaux : d'une part l'utilisation de modèles CNN pré-entraînés pour extraire les features visuelles, d'autre part le fine-tuning et l'évaluation sur un jeu de données vidéo de référence (**UCF50** puis transfert sur de nouvelles vidéos).

Contexte de l'étude de cas

Reconnaître des actions à partir de vidéos est un défi, car il faut analyser à la fois **le contenu spatial** de chaque image (ce que fait un CNN) et **l'évolution temporelle** de l'action à travers les frames (ce que fait un RNN/LSTM). L'approche CNN+LSTM proposée par BleedAI combine ainsi "le meilleur des deux mondes" : un CNN extrait un vecteur de caractéristiques de chaque frame (par exemple détecter qu'une personne est présente, sa pose, les objets environnants), puis un LSTM traite la séquence de ces vecteurs pour modéliser le déroulement de l'action sur le temps. Deux architectures spécifiques ont été implémentées : (1) un modèle de type **ConvLSTM**, où des couches de convolution tempèrent directement dans la structure LSTM (c'est-à-dire un LSTM dont les opérations internes sont remplacées par des convolutions – utile pour traiter des séquences d'images en conservant la structure spatiale), et (2) un modèle de type **LRCN** (Long-term Recurrent Convolutional Network), où l'on applique d'abord un CNN 2D sur chaque frame puis on enchaîne avec un LSTM qui ingère la suite des vecteurs de sortie du CNN. Les auteurs expérimentent ces deux approches afin de voir laquelle performe le mieux pour la classification d'actions.

Le jeu de données utilisé est **UCF-50**, qui contient des vidéos réalistes Youtube réparties en 50 catégories d'actions (sports, activités diverses). Chaque catégorie comporte au minimum 100 vidéos, et globalement les vidéos durent quelques secondes (~199 frames en moyenne). Les données sont divisées en 25 groupes par action, ce qui permet de faire un découpage entraînement/test robuste (par exemple, entraîner sur 20 groupes et tester sur 5 groupes restants, de sorte que les vidéos de test proviennent de sources différentes). Avant entraînement, les frames des vidéos sont **redimensionnées** à une taille fixe et **normalisées** pour uniformiser l'entrée du réseau. Ensuite, les vidéos sont segmentées en séquences de longueur fixe pour servir d'entrées au modèle CNN+LSTM. Chaque frame passe par le CNN qui produit un vecteur de caractéristiques visuelles, et la séquence de 20 vecteurs est fournie au LSTM qui sort une prédiction d'action. Durant l'entraînement, on ajuste les poids du CNN et du LSTM pour minimiser l'erreur de classification sur les séquences vidéo.

Résultats

Les expérimentations de T. Anwar *et al.* montrent que l'architecture CNN+LSTM permet d'obtenir une **excellente précision** sur la tâche de classification d'actions vidéo. En particulier, le modèle **LRCN** s'est avéré supérieur au modèle ConvLSTM dans leurs tests. Le

LRCN a atteint environ **92.6%** de précision (accuracy) sur la classification des vidéos UCF-50, là où le modèle ConvLSTM plus simple plafonnait autour de **80.3%** de précision. Ce gain substantiel s'explique par une meilleure capacité du LRCN à capturer l'enchaînement de l'action : dans le ConvLSTM implémenté, l'architecture plus contrainte avait du mal à apprendre efficacement les deux composantes (spatiale et temporelle) en même temps, alors que la séparation CNN puis LSTM du LRCN a permis de tirer pleinement profit des deux réseaux.

Une fois entraîné, le **meilleur modèle (LRCN)** a été testé sur des vidéos inédites (par ex. des vidéos YouTube de l'une des 50 classes). Les prédictions montrent que le réseau identifie correctement l'action dans la plupart des cas, en exploitant aussi bien les indices visuels dans chaque frame (par ex. détecter un ballon de basket pour la classe *basketball*) que le mouvement d'ensemble (par ex. discerner *chuter* vs *faire un salto* en regardant la continuité des poses). L'article souligne cependant une **limite** importante du modèle : il n'est pas capable de gérer plusieurs personnes effectuant des actions différentes simultanément dans le champ de la caméra.

En termes d'**apprentissage par transfert**, cette étude de cas en illustre plusieurs aspects : d'abord, le CNN utilisé pour les frames peut bénéficier d'un pré-entraînement sur un large corpus d'images (typiquement ImageNet) afin d'améliorer la qualité des features visuelles extraites – même si l'article ne le précise pas explicitement, il est courant d'utiliser un CNN pré-entraîné plutôt qu'un CNN initialisé aléatoirement, ce qui constitue une forme de transfert. Ensuite, le fait de tester le modèle sur de nouvelles vidéos non issues du dataset montre une capacité de généralisation, mais également les limites dès que le **contexte diffère** (plusieurs personnes, arrière-plan complexe non vu en entraînement, etc.).

Globalement, les performances obtenues (≈92% de précision) sont **très élevées** compte tenu du nombre de classes (50) et de la variété des actions, démontrant la **puissance de l'architecture CNN+LSTM** pour la HAR vidéo. On constate qu'en intégrant le temporel, on corrige les erreurs qu'un simple CNN par frame commettrait. Le LSTM permet d'éviter cela en recontextualisant chaque frame parmi les précédentes et suivantes.

Dans le chapitre suivant (chapitre 3), nous nous appuierons sur ces constats pour mettre en œuvre notre propre modèle CNN–LSTM sur un jeu de données donné, en utilisant

éventuellement du transfert learning depuis des modèles pré-entraînés, et en évaluant ses performances.

2.6 Jeux de données de référence et performances de l'état de l'art

L'évaluation des modèles de HAR se fait traditionnellement sur plusieurs **jeux de données de référence** (*benchmarks*) couvrant différents contextes : des données de capteurs wearables, des vidéos d'actions filmées, des environnements intelligents, etc. Chacun pose ses spécificités (nombre de sujets, nombre et nature des capteurs, nombre d'activités, etc.). Nous présentons ci-dessous quelques-uns des datasets les plus utilisés dans la littérature, ainsi que les performances typiques obtenues par les approches Deep Learning récentes.

- **UCI HAR Dataset (Anguita et al., 2012)** [20]: Ce dataset de l'University of California Irvine est l'un des plus classiques pour la HAR sur smartphone. Il contient les enregistrements d'accéléromètre et de gyroscope d'un smartphone placé à la ceinture de 30 volontaires réalisant 6 activités de base (marcher, monter des escaliers, descendre des escaliers, s'asseoir, se tenir debout, s'allonger). Chaque enregistrement est segmenté en fenêtres de 2,56 secondes avec un chevauchement de 50%, fournissant plus de 10 000 instances au total. Historiquement, les méthodes classiques (SVM, Random Forest) atteignaient déjà ~90% à 92% de précision sur ce dataset simple. Les approches par Deep Learning ont encore amélioré légèrement ce score : par exemple, un CNN 1D peut obtenir ~93-94%, et des architectures hybrides arrivent à frôler les **95-96%** de précision, voire plus. Un article de 2016 (Ronao & Cho) rapportait 95% avec un CNN; un autre de 2017 (Yang et al.) 96% avec un Deep CNN. Plus récemment, Singh *et al.* (2020) annoncent même 99% avec un modèle CNN-LSTM optimisé sur UCI HAR.

- **PAMAP2 (2012)** : Ce dataset (« Physical Activity Monitoring for Aging People ») contient des enregistrements de 9 sujets réalisant 18 activités (marche, course, cyclisme, ménage, positions statiques, etc.), capturés via 3 capteurs inertiels placés sur la main, la cheville et la poitrine, ainsi qu'un capteur de fréquence cardiaque. C'est un ensemble plus varié que UCI, avec des activités de nature différente et des capteurs multiples. Les méthodes classiques atteignaient ~80-85% de précision. Les approches Deep Learning comme DeepConvLSTM ont permis de monter au-dessus de 90%. Par exemple, *DeepConvLSTM* (Ordóñez 2016) obtient ~94% sur PAMAP2 en fusionnant les 3 capteurs.

- **Opportunity (2011)** : C'est un des jeux de données les plus complets pour la HAR en environnement instrumenté. Il s'agit d'enregistrements de 4 personnes effectuant des activités de la vie quotidienne dans une pièce équipée de nombreux capteurs (plus de 70 capteurs : accéléromètres sur différents membres, capteurs d'état d'objets, etc.). Le challenge Opportunity comportait deux tâches : la reconnaissance de modes de locomotion (marcher, se tenir, s'asseoir, etc.) et la reconnaissance de micro-gestes liés aux objets (ouvrir un tiroir, allumer la lumière, etc.). C'est un dataset très complexe du fait du très grand nombre de capteurs et de la prédominance d'un état "Null" (pas de geste) représentant la majeure partie du temps. Les meilleures approches du challenge (2011) utilisaient des combinaisons de features et de classifieurs, atteignant environ 70-75% de F-score en reconnaissance de gestes. Le modèle **DeepConvLSTM** a amélioré le score F1 moyen d'environ **6%** par rapport au meilleur du challenge, et jusqu'à +9% sur certains gestes. Cela témoigne du gain apporté par le CNN+LSTM pour capter à la fois les caractéristiques locales sur chaque capteur et les dépendances temporelles entre capteurs. Opportunity demeure un benchmark exigeant, et en 2025 les recherches se poursuivent (par ex. Bock *et al.* 2021 ont proposé ShallowConvLSTM pour réduire la complexité, etc. – certaines méthodes récentes mentionnent dépasser 85% de F1).

- **WISDM (2013)** : Le dataset WISDM (Wireless Sensor Data Mining) est une base de données popularisée pour la HAR sur smartphone, comprenant les enregistrements d'accéléromètre de 29 personnes effectuant 6 activités (similaires à UCI HAR). Il a ~1,1 million d'échantillons de capteurs. Les méthodes classiques donnaient ~91% (J48, KNN) sur WISDM. Un réseau LSTM pur a été l'un des premiers à faire progresser le score (Hammerla 2016 rapportait ~94%). Des améliorations récentes via des CNN 1D ou des architectures bi-LSTM avec attention ont encore augmenté la précision. Par exemple, un **CNN-LSTM** rapporté en 2020 atteint ~96% sur WISDM. Là encore, on frôle le plafond avec les signaux simples et un nombre de classes restreint.

- **HMDB51 (2011)** : Côté vision, le dataset **HMDB51(Kuehne et al., 2011) [26]** est une collection de vidéos de films et de YouTube, comprenant 51 catégories d'actions variées (gestes, interactions, sports). C'était le plus grand dataset d'actions lors de sa publication, avec 6 766 clips au total (≈133 clips par action en moyenne). Les actions vont de "manger" à "sauter" en passant par "nager" ou "embrasser". Les premières méthodes de 2011

basées sur des descripteurs de mouvement (HOG/HOF) couplés à des SVM obtenaient environ **20% à 30%** de précision sur HMDB51. En 2013-2015, l'introduction des **features trajectoires améliorées** (iDT) a porté ce score à ~50%. Désormais, avec les réseaux profonds, les performances ont très fortement grimpé : un modèle **Two-Stream CNN** entraîné fin 2014 atteignait 59%; un **I3D (Inflated 3D ConvNet)** entraîné sur Kinetics puis transféré sur HMDB a atteint ~74% en 2017. Les dernières méthodes (2020-2021) dépassent **80%** de précision top-1 sur HMDB51 en utilisant soit des architectures 3D CNN très profondes (ResNeXt-101 3D, SlowFast, etc.), soit des Transformers vidéo pré-entraînés sur des énormes corpus.

- **Kinetics-400/600/700 (DeepMind, 2017-2019)** : Kinetics (Kay et al., 2017) [27] est un **dataset massif de vidéos YouTube** introduit par DeepMind pour pousser les limites de la reconnaissance d'actions. La version d'origine, Kinetics-400, comporte 400 classes d'actions et ~306 000 vidéos. Chaque clip dure ~10 secondes et met en scène une action humaine catégorisée (par ex. "faire du vélo BMX", "jongler avec un ballon de football"). Une extension **Kinetics-600** (600 classes, ~400k vidéos) et **Kinetics-700** (700 classes, ~650k vidéos) ont ensuite été publiées. Ces jeux de données n'ont pas tant vocation à servir de test (ils sont trop grands) qu'à **entraîner** des modèles complexes qui seront ensuite testés sur des benchmarks plus petits via transfert. En effet, entraîner un modèle du niveau d'un ResNet-50 3D ou d'un Transformer vidéo nécessite des centaines de milliers d'échantillons – Kinetics fournit cela. Les performances sur Kinetics elles-mêmes sont impressionnantes : le modèle I3D original atteignait ~74% de précision top-1 sur K400, les modèles plus récents (SlowFast de Feichtenhofer 2019, ViViT 2021) ont dépassé **80-82%** top-1 sur K400 et ~85% sur K600. La plupart de ces modèles, une fois pré-entraînés sur Kinetics, sont utilisés pour **le transfert** vers d'autres contextes : ex. fine-tuning sur HMDB51 ou UCF101 où ils obtiennent les meilleurs scores, ou encore adaptation à des vidéos de conduite automobile, de vidéos de surveillance, etc.

- **NTU RGB+D (2016)** : Un autre dataset notable est NTU RGB+D, comprenant 60 puis 120 classes d'actions capturées par caméra de profondeur avec 40 sujets. C'est un standard pour la reconnaissance d'actions basée sur les **squelettes (skeleton data)** ou la fusion vidéo+profondeur. Les meilleurs modèles (souvent à base de Graph Neural Networks ou

Transformer sur les séquences de poses 3D) dépassent 90% de précision sur NTU, ce qui montre encore une fois l'efficacité des approches DL spécialisées.

2.7 Défis actuels et perspectives

Malgré les succès indéniables du Deep Learning en reconnaissance d'activités, de **nombreux défis** demeurent pour atteindre une HAR réellement fiable, généralisée et applicable en conditions réelles. Nous en soulignons quelques-uns, ainsi que les pistes de recherche et perspectives pour les relever :

- **Généralisation aux nouveaux utilisateurs et contextes** : un modèle entraîné sur un ensemble de personnes peut voir ses performances chuter lorsqu'il est confronté à un nouvel utilisateur aux habitudes différentes, ou à un capteur placé différemment. La variabilité inter-individuelle (façons différentes d'effectuer la même action) et inter-contexte (environnement, conditions de capture) est un obstacle majeur. Par exemple, une même activité « faire la vaisselle » peut générer des signatures capteurs différentes selon la personne et la cuisine utilisée. L'**apprentissage par transfert** peut jouer un rôle clé pour personnaliser les modèles. en affinant automatiquement le modèle de HAR pour chaque utilisateur en apprenant quelques heures/jours sur ses données (avec son consentement). De plus, les approches de **meta-learning** pourraient entraîner un modèle initial capable d'apprendre très vite un nouvel utilisateur avec peu d'exemples (*few-shot learning*). Une autre direction est l'**apprentissage fédéré**, où le modèle global est entraîné de manière collaborative par de multiples utilisateurs sans centraliser les données brutes. Enfin, l'**augmentation de données synthétiques** pourrait enrichir l'entraînement pour améliorer la robustesse.

- **Reconnaissance d'activités complexes et hiérarchiques** : Les progrès se sont surtout manifestés sur des activités **primitives** bien délimitées (marcher, courir, lever le bras, etc.). Mais la reconnaissance d'**activités de haut niveau** (ex: "préparer une recette", "faire du jardinage") reste un défi, car ces scénarios englobent une séquence d'actions élémentaires et d'interactions avec objets sur de longues durées. Comme perspective**,** c'est l'intégration des sources de données **multiples** et du **raisonnement contextuel**. On voit apparaître des travaux combinant la vision avec les capteurs corporels, afin d'inférer l'activité globale. Des **modèles hiérarchiques** sont proposés : un premier niveau reconnaît des actions primitives et un second niveau infère l'activité globale en séquence. Cela nécessite

potentiellement des techniques empruntées au **NLP (modélisation de séquences symboliques)** ou aux réseaux bayésiens pour modéliser la relation entre sous-actions et activité globale.

- **Données peu annotées et apprentissage auto-supervisé** : Annoter des données HAR est coûteux. Une direction prometteuse est l'**apprentissage auto-supervisé** ou non supervisé, où un modèle est pré-entraîné sur de larges corpus sans étiquettes via des tâches proxy (prédire la suite du signal, détecter des anomalies, etc.). De tels modèles pourraient apprendre une représentation du mouvement humain riche, qui ensuite serait transférable (transfer learning) vers la classification d'activités avec peu d'exemples annotés. On voit déjà des résultats encourageants, par ex. des autoencodeurs ou des modèles contrastifs appliqués à des séries temporelles de capteurs, dont les représentations latentes permettent de classifier les activités avec bien moins d'entraînement supervisé.

- **Efficacité, embarquabilité et consommation** : Un **défi pratique** est de faire tourner ces modèles profonds en dehors du laboratoire, sur des appareils mobiles ou des systèmes embarqués (smartphones, montres, microcontrôleurs pour IoT). Les réseaux profonds classiques sont lourds (plusieurs millions de paramètres) et énergivores, ce qui n'est pas idéal pour une exécution continue sur batterie. De nombreuses recherches explorent la **compression de modèles** (réseaux condensés, *pruning*, quantification 8-bit, etc.) pour réduire la taille et la complexité des réseaux HAR sans trop perdre en précision. Par exemple, remplacer un LSTM par un **GRU** ou un simple RNN lorsque possible, utiliser des CNN 1D à *depthwise separable convolutions* (plus légers). L'utilisation d'**accélérateurs dédiés** (DSP, NPU) sur les dispositifs permet aussi d'exécuter des réseaux plus gros en limitant la consommation.

- **Interprétabilité et acceptation utilisateur** : Les modèles de Deep Learning sont souvent des boîtes noires peu interprétables, ce qui peut poser problème pour la **confiance** et l'**acceptation** par les utilisateurs ou par des autorités (dans le médical, on aimerait comprendre pourquoi le système prédit telle activité ou telle anomalie). Fournir des **explications** (par ex. mettre en avant quelles parties du signal ou quels capteurs ont conduit à la décision) est une voie de recherche pour améliorer l'explicabilité. D'autre part, la question de la **vie privée** est incontournable : enregistrer des données d'activité peut

révéler des informations sensibles (état de santé, habitudes quotidiennes, localisation…). Les déploiements à grande échelle doivent intégrer des mécanismes de protection des données : anonymisation, traitement en local sur l'appareil (edge computing) au lieu d'envoyer les données brutes sur le cloud, chiffrement des communications, etc.. Par exemple, Huang *et al.* (2010) [23] et Vergara-Laurens *et al.* (2011) [24] proposent des cadres où les données brutes restent locales et seules des informations agrégées ou bruitées sont partagées, réduisant le risque pour la vie privée.

2.8 Conclusion du chapitre

Dans ce chapitre, nous avons dressé un aperçu **de la reconnaissance d'activité humaine par Deep Learning**, en insistant sur l'utilisation de l'**apprentissage par transfert** pour tirer parti de ces modèles dans divers contextes. Nous avons d'abord souligné la transition des approches classiques vers les approches profondes qui apprennent automatiquement des représentations discriminantes des données capteurs ou vidéos, offrant un gain significatif de performances. Nous avons passé en revue les **principales architectures** exploitées en HAR: les CNN qui extraient efficacement des motifs locaux des signaux, les RNN/LSTM/GRU qui modélisent la dynamique temporelle des activités, et les Transformers émergents qui promettent de capturer des dépendances à long terme grâce à l'attention. Nous avons ensuite détaillé le **concept d'apprentissage par transfert**, en montrant comment il permet d'adresser un des défis majeurs de la HAR (manque de données annotées, nécessité de personnalisation) en réutilisant des modèles pré-entraînés sur de larges corpus pour de nouvelles tâches spécifiques. Nous avons enfin passé en revue plusieurs **jeux de données emblématiques** de la HAR (UCI HAR, PAMAP2, Opportunity, HMDB51, Kinetics, etc.) en soulignant l'apport du Deep Learning dans chacun d'eux (gains de +5 à +15% voire plus sur les métriques de précision par rapport aux méthodes classiques).

Pour conclure, la reconnaissance d'activités par apprentissage profond est un domaine en plein essor qui a permis d'**importants progrès** en termes de précision et de possibilités applicatives. Cependant, les **défis** de robustesse, de passage à l'échelle, d'efficacité et d'acceptabilité restent d'actualité. Les travaux de recherche en cours visent tous à rendre les systèmes HAR plus **précis**, plus **généraux**, et plus **fiables** dans des conditions réelles. Le prochain chapitre de ce manuscrit s'appuiera sur l'état de l'art présenté ici pour aborder **nos**

contributions qui visent à apporter une pierre additionnelle à l'édifice de la reconnaissance d'activités humaines intelligentes.

Chapitre 3 : Optimisation du modèle CNN + LSTM pour la reconnaissance d'activités humaines

III. Chapitre 3 : Optimisation du modèle CNN + LSTM pour la reconnaissance d'activités humaines

Ce chapitre revoie tout d'abord les approches de la HAR basées sur les réseaux de neurones profonds et leurs limites, ensuite pose la problématique abordée dans ce travail. Puis il présente en détail l'architecture du modèle CNN+LSTM et en particulier la solution proposée pour améliorer ce modèle. Enfin, il donne un aperçu de la base de données UCF50 utilisée pour l'évaluation de notre modèle dans le dernier chapitre.

3.1 Contexte général

La reconnaissance d'activités humaines est devenue un domaine de recherche prioritaire dans les systèmes intelligents, en raison de ses diverses applications telles que la vidéosurveillance, le suivi médical, le sport, l'assistance aux personnes âgées et la sécurité. Les approches traditionnelles de la HAR fondées sur l'extraction de caractéristiques manuelles et l'utilisation de classifieurs standards (SVM, KNN, Random Forest) ont montré des limites importantes en termes de précision et de capacité de généralisation dans des environnements variés et dynamiques. Face à ces contraintes, ont vu l'émergence des méthodes d'apprentissage profond. Ces approches apprennent directement des représentations discriminantes à partir des données brutes et ont démontré des gains de performance significatifs sur divers jeux de données standardisés.

3.2 Approches HAR basées sur le Deep Learning

Plusieurs familles de modèles ont été explorées pour la reconnaissance d'activités humaines:

1. **Les réseaux de neurones convolutifs (CNN)** : particulièrement efficaces pour extraire des caractéristiques spatiales dans les images et les séquences vidéos, permettant de capter des motifs locaux liés aux postures et aux mouvements.
2. **Les réseaux récurrents (RNN, LSTM, GRU)** : capables de modéliser les dépendances temporelles des séquences d'images ou de signaux capteurs, et d'ainsi suivre l'évolution des activités dans le temps.
3. **Les architectures hybrides CNN-LSTM** : combinent les avantages des CNN pour l'extraction spatiale et des LSTM pour le suivi temporel, obtenant ainsi des performances supérieures dans la reconnaissance d'activités complexes.
4. **Les Transformers** : approches plus récentes dans la HAR, qui, grâce aux mécanismes d'attention, permettent de capturer des relations à long terme dans les séquences d'images ou de signaux. Ces modèles se sont révélés prometteurs, en particulier sur de grands jeux de données.

Par ailleurs, l'**apprentissage par transfert** est devenu une stratégie incontournable pour améliorer les performances, en réutilisant des modèles pré-entraînés sur de larges corpus pour des tâches spécifiques à moindre coût d'annotation.

3.3 Limites des approches existantes

Malgré les gains en précision et en robustesse qu'apportent les techniques de Deep Learning pour la HAR, plusieurs limitations subsistent :

- **Dépendance aux données annotées** : l'entraînement des modèles profonds nécessite d'importantes quantités de données labellisées, souvent difficiles à obtenir dans certains domaines ou contextes spécifiques.
- **Problèmes de généralisation** : les modèles entraînés dans un environnement ou sur un type d'activité donné peinent à s'adapter à des contextes différents (variations d'angles de vue, conditions lumineuses, plusieurs personnes en interaction…).
- **Transfert négatif** : dans certains cas, l'utilisation de modèles pré-entraînés peut dégrader les performances si les domaines source et cible sont trop éloignés.

- **Complexité computationnelle** : les modèles profonds, notamment les architectures combinées ou les Transformers, exigent des ressources matérielles importantes et des temps d'entraînement élevés.

3.4 Problématique

Au regard des limites identifiées dans les modèles Deep Learning utilisés dans la HAR, il est plus que nécessaire de développer des modèles de reconnaissance d'activités capables de :

- exploiter efficacement des caractéristiques spatiales et temporelles ;
- réduire la dépendance à de grands volumes de données annotées ;
- s'adapter à des environnements variés et à des activités complexes ;
- être suffisamment légers pour des applications en conditions réelles.

Dans ce contexte, l'utilisation de modèles hybrides combinant CNN et LSTM, couplée à des techniques d'apprentissage par transfert, représente une piste prometteuse pour surmonter ces défis. Le présent travail s'inscrit dans cette démarche, en proposant et en évaluant une architecture adaptée à la reconnaissance d'activités humaines à partir de séquences vidéos.

3.5 Présentation du modèle de base CNN + LSTM

Dans cette section, nous décrivons le modèle de base utilisé pour la reconnaissance d'activités, qui combine un **réseau convolutif (CNN)** et un **réseau LSTM**. Le CNN intervient comme extracteur de caractéristiques visuelles sur les images, tandis que le LSTM exploite ces caractéristiques sur la durée pour reconnaître l'action effectuée. Avant de détailler leur interaction **(section 3.2.3)**, il convient de rappeler brièvement le principe de fonctionnement de chacun de ces deux types de réseaux.

3.5.1 Définition de CNN

Un **réseau de neurones convolutif** (CNN, pour *Convolutional Neural Network*) est un type de réseau de neurones spécialisé dans le traitement des données organisées sous forme de grille, typiquement les images en pixels. Un CNN est particulièrement efficace pour analyser le contenu visuel d'une image et en extraire automatiquement des **caractéristiques** pertinentes pour une tâche donnée (classification, détection d'objet, etc.).

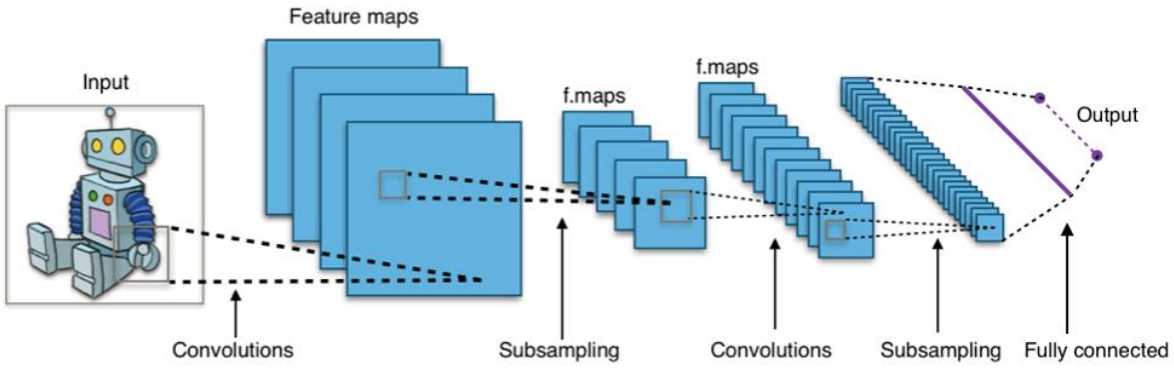

Figure III:1 Architecture simplifiée d'un réseau de neurones convolutifs (CNN) [28]

L'image d'entrée (à gauche) passe par des couches de **convolution** qui produisent des cartes de caractéristiques (feature maps, en bleu). Des couches de **sous-échantillonnage** (pooling) réduisent progressivement la résolution des cartes tout en conservant l'information essentielle. Enfin, des couches entièrement connectées (à droite) utilisent ces caractéristiques pour **classifier** l'image en une catégorie de sortie.

Le principe de base d'un CNN repose donc sur l'application de **filtres convolutionnels** (aussi appelés *kernels*) sur l'image d'entrée. Un filtre est une petite matrice (par exemple 3×3 pixels) qui va être glissée sur l'image, et à chaque position, le filtre calcule une combinaison linéaire des pixels couverts. Ce processus produit une **carte de caractéristiques** indiquant où le motif détecté par le filtre est présent dans l'image. Initialement, les premiers filtres peuvent apprendre à détecter des motifs simples comme des **bords** ou des **coins** dans l'image. Ensuite, en empilant plusieurs couches de convolution, le réseau apprend des caractéristiques de plus en plus complexes : par exemple, une couche plus profonde combinera les motifs simples pour détecter des formes plus élaborées (textures, parties d'objets), et ainsi de suite, jusqu'à ce que les dernières couches apprennent des motifs très complexes et spécifiques à la tâche (par exemple, la présence d'une silhouette humaine dans une certaine posture).

Entre les couches de convolution, on insère généralement des couches de **pooling** (sous-échantillonnage), typiquement du *max-pooling*, qui réduisent la taille des cartes de caractéristiques (par exemple en ne conservant que le maximum sur des fenêtres de 2×2 pixels). Cette réduction permet de **diminuer la résolution** des représentations tout en conservant les informations les plus importantes, rendant le réseau plus robuste aux petites

translations de l'image et réduisant le nombre de paramètres. Au fur et à mesure que l'on avance dans le réseau, on obtient ainsi un plus grand nombre de cartes de caractéristiques de plus petite taille, mais contenant chacune une information plus abstraite sur le contenu de l'image. Finalement, les sorties des dernières convolutions sont aplaties en un vecteur (appelé parfois *code CNN* ou *embedding* de l'image) qui résume l'image originale en termes de caractéristiques haut niveau. Ce vecteur est ensuite envoyé dans une ou plusieurs **couches entièrement connectées** (perceptrons multicouches classiques) qui vont effectuer la **classification finale** en calculant la probabilité d'appartenance de l'image à chaque classe possible.

Dans notre contexte, le CNN est utilisé pour traiter **chaque image** (chaque *frame*) de la vidéo et en extraire des caractéristiques pertinentes pour reconnaître l'action en cours. Par exemple, sur une image isolée d'une personne en train de nager, le CNN pourra détecter des motifs visuels tels que la position des bras, la présence d'eau autour de la personne, etc., qui sont des indices de l'activité "natation". Cependant, une image seule ne suffit pas toujours à déterminer l'action (imaginons une personne en l'air : est-elle en train de sauter à la corde, de faire un saut en longueur, ou de tomber ?). C'est pourquoi nous avons besoin d'analyser *plusieurs images successives* via un mécanisme temporel, ce qui nous amène au rôle du LSTM.

3.5.2 Définition de LSTM

Un **réseau LSTM** (pour *Long Short-Term Memory*) est un type de réseau de neurones récurrent (RNN) conçu pour modéliser des **données séquentielles** et apprendre des dépendances temporelles à long terme. Les réseaux RNN "classiques" traitent les séquences pas à pas en faisant circuler une mémoire (un état caché) d'une étape à la suivante. Cependant, les RNN standards souffrent souvent du problème de la **dissipation du gradient** (*vanishing gradient*) lorsque la séquence est longue, ce qui les empêche d'apprendre correctement des relations à long terme. Les LSTM ont été inventés pour remédier à ce problème.

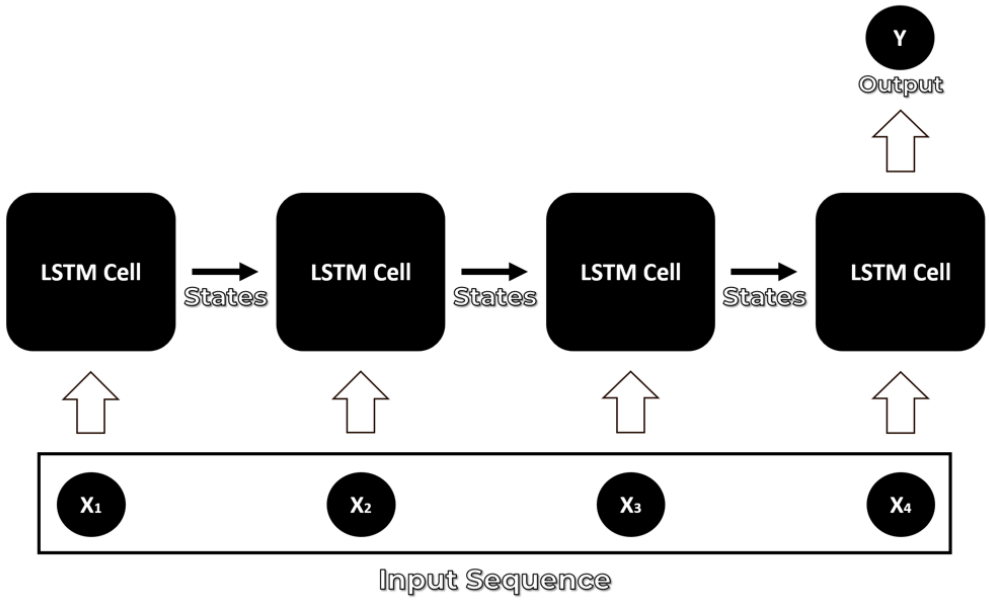

Figure III:2 Schéma conceptuel d'un réseau LSTM many-to-one[28]

Les cercles noirs représentent une séquence d'entrées temporelles (X_1, X_2, X_3, X_4 par exemple des vecteurs de caractéristiques à chaque frame) qui sont alimentées successivement dans des cellules LSTM (carrés). Chaque **cellule LSTM** combine l'entrée du moment avec l'**état interne** qu'elle reçoit de la cellule précédente (flèches States), lui permettant de "se souvenir" d'informations passées pertinentes. Après avoir parcouru toute la séquence, la dernière cellule LSTM produit une **sortie** (Y) qui agrège l'information temporelle pour effectuer la prédiction finale (par exemple, la classe d'action).

Une cellule LSTM intègre des mécanismes internes appelés **portes** (*input gate, forget gate, output gate*) qui régulent le flux d'informations à travers la cellule. Ces portes permettent au LSTM de **mémoriser ou d'oublier sélectivement** certaines informations de la séquence au fil du temps. Concrètement, le LSTM maintient un **état interne** (ou mémoire) qui est mis à jour à chaque pas de temps en fonction de l'entrée actuelle et de l'état précédent. Grâce à son architecture, un LSTM peut **préserver du contexte sur de longues séquences** en retenant des informations importantes même après de nombreux pas de temps, évitant qu'elles ne se diluent ou ne disparaissent comme c'est le cas avec les RNN simples.

Cette capacité de mémoire à long terme rend les LSTM particulièrement efficaces pour les tâches où la **dynamique temporelle** joue un rôle crucial. Par exemple, les LSTM ont été

appliqués avec succès à la prédiction de séries temporelles (prévision de données financières ou météorologiques). Dans notre cas d'usage, le LSTM va nous servir à modéliser la **suite d'images** composant une vidéo afin de reconnaître l'activité globale effectuée. C'est un scénario de type *many-to-one* (plusieurs entrées, une sortie) : on fournit en entrée une séquence d'images et on souhaite en sortie la classe de l'action qui a lieu tout au long de cette séquence. Le LSTM parcourra ainsi les frames une par une, en "souvenant" des éléments importants (postures, objets manipulés, mouvements effectués) et en oubliant les détails non pertinents, puis il produira au final une décision basée sur l'ensemble de la séquence.

En résumé, là où un CNN excelle à extraire des **traits spatiaux**, un LSTM excelle à capturer des **traits temporels**. Il est donc naturel de tenter de combiner ces deux types de réseaux pour s'attaquer au problème de la classification d'actions dans des vidéos, ce que nous détaillons à présent.

3.5.3 Fonctionnement global du modèle CNN + LSTM

Le modèle de base que nous utilisons pour la reconnaissance d'activités humaines est un modèle hybride qui associe un CNN et un LSTM. Ce type d'architecture est communément appelé un modèle **CNN+LSTM** ou parfois **LRCN** (*Long-term Recurrent Convolutional Network*), et permet de tirer avantage à la fois des capacités des CNN et des LSTM.

D'une manière générale, son **processus** est le suivant :

- Chaque vidéo est considérée comme une séquence de frames (images) extraites à intervalles réguliers. Par exemple, on peut décider d'utiliser N =20 images successives d'une vidéo pour résumer l'action en cours.

- Un **CNN** est appliqué sur **chaque frame** individuellement, pour en extraire un vecteur de **caractéristiques visuelles**. On peut voir le CNN comme un encodeur d'image : il transforme chaque image de dimension (hauteur × largeur × canaux) en un vecteur de plus petite dimension contenant l'information importante de l'image (par exemple, le CNN peut indiquer s'il détecte un visage, un ballon, un bras levé, de l'eau, etc., selon ce qu'il a appris).

- On obtient ainsi, pour une séquence de 20 images, une **séquence de 20 vecteurs** de caractéristiques. Ces vecteurs constituent une représentation compacte de la vidéo, frame par frame, et forment l'entrée de la partie LSTM.
- Le **LSTM** traite cette séquence de vecteurs dans l'ordre temporel : il prend le vecteur du 1er frame, met à jour son état interne ; puis prend le vecteur du 2e frame, met à jour son état, etc., jusqu'au 20e frame. Durant ce processus, le LSTM *apprend les relations temporelles* entre les frames : par exemple, si un vecteur indique "une personne est debout bras plié" et le suivant "la personne a le bras tendu vers l'avant", le LSTM peut déduire qu'il s'agit d'un mouvement de lancer, ce qu'on ne pouvait pas affirmer avec une seule image.
- À la fin de la séquence, le LSTM fournit une **sortie** (un vecteur) qui encapsule l'information de toute la séquence vidéo. Ce vecteur est ensuite passé dans une couche de **classification finale** (typiquement une couche Dense suivie d'une fonction softmax) qui produit les **probabilités** d'appartenance de la séquence à chacune des classes d'action possibles.

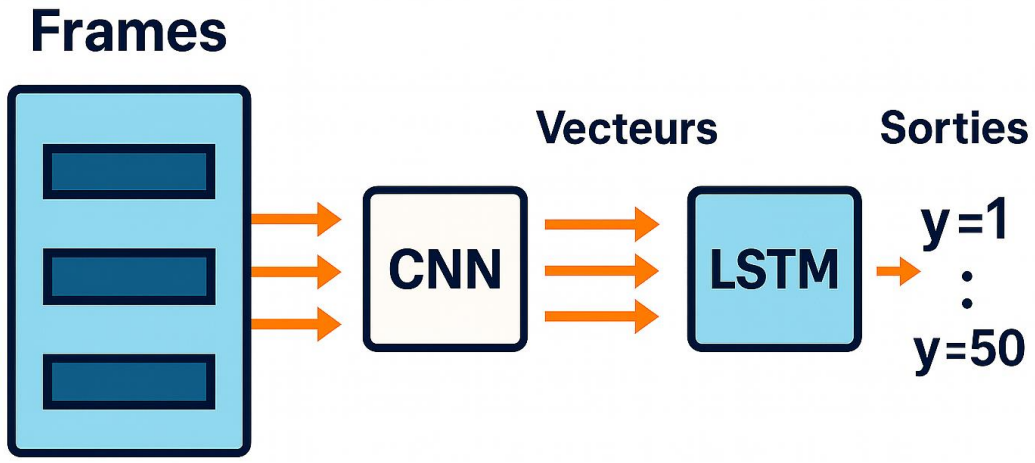

Figure III:3 Schéma du fonctionnement d'un modèle CNN+LSTM pour la classification des frames d'une vidéo.

À gauche, des images successives issues d'une vidéo (frames d'entrée). Au centre, un CNN extrait des **caractéristiques visuelles** de chaque frame de manière indépendante. À droite, un LSTM reçoit la séquence de ces caractéristiques et apprend les **dépendances temporelles** entre frames, avant de produire en sortie la prédiction de l'action (classe « y »). Ce modèle exploite ainsi l'information spatiale de chaque image et l'information temporelle de la séquence pour améliorer la reconnaissance d'activité.

Dans notre implémentation de base, le CNN et le LSTM sont entraînés **ensemble** pour prédire la classe d'action à partir d'une séquence vidéo. Plus précisément, on construit un réseau global où les premières couches sont convolutionnelles (appliquées à chaque frame via un mécanisme de *TimeDistributed* en TensorFlow/Keras, ce qui signifie qu'on duplique le même CNN pour chaque image de la séquence), puis ces sorties alimentent une ou plusieurs couches LSTM, et enfin une couche dense finale donne la prédiction. Ce modèle est entraîné de bout en bout sur des exemples de séquences vidéo étiquetées : l'optimisation ajuste les filtres du CNN et les connexions du LSTM simultanément pour minimiser l'erreur de classification.

Les étapes du modèle de base CNN+LSTM pour le traitement d'une vidéo sont :

1. **Extraction des frames** : on extrait *N* images de la vidéo originale. Afin de capturer l'ensemble de l'action sans utiliser toutes les images (ce qui serait coûteux et parfois redondant), on prélève des frames à intervalles réguliers sur la durée de la vidéo. Par exemple, pour N=20 et une vidéo de 4 secondes à 20 images/seconde (soit ~80 frames), on pourra prendre 1 image toutes les 4 frames environ, réparties sur la vidéo. Ainsi, on obtient une séquence de 20 frames représentative du début à la fin de l'action.

2. **Prétraitement des images** : chaque frame est redimensionnée et normalisée de la même façon que pour l'entraînement du CNN (voir la section 3.4.2). Par exemple, on peut réduire chaque image à une taille fixe de 64×64 pixels et ramener les valeurs de pixels entre 0 et 1.

3. **Passage dans le CNN** : chaque frame prétraitée est passée à travers le CNN (toutes les frames traversent le *même* CNN, qui agit comme un extracteur de caractéristiques

partagé). On obtient ainsi pour chaque image un vecteur de caractéristiques de haut niveau.

4. **Passage dans le LSTM** : la séquence ordonnée de vecteurs de caractéristiques est ensuite entrée dans le LSTM. Le LSTM intègre l'information de toute la séquence en parcourant ces vecteurs un par un.

5. **Classification** : la sortie finale du LSTM (ou éventuellement la moyenne/concaténation de ses sorties sur plusieurs pas de temps selon les variantes) est fournie à la couche de classification qui prédit la classe d'activité la plus probable pour la séquence traitée.

Le **modèle de base** CNN+LSTM ainsi défini, lors des premiers essais que nous avons effectués, a montré des **limites** notables : la précision obtenue sur la reconnaissance d'activités, bien que meilleure qu'un simple CNN frame-à-frame, restait relativement **faible**. De plus, le modèle montrait des signes de **surapprentissage** : sa performance sur les données d'entraînement était largement supérieure à celle sur les données de test, indiquant qu'il s'adaptait trop fortement aux exemples vus pendant l'apprentissage et généralisait mal à de nouveaux exemples. On observait aussi une **variabilité des performances** selon les runs, possiblement due à la petite taille de certains lots d'entraînement et à l'initialisation aléatoire des poids. Ces problèmes peuvent provenir de multiples facteurs : des **paramètres d'entraînement** inadaptés (tels qu'un nombre d'époques trop faible pour bien converger, ou au contraire trop élevé provoquant un surapprentissage, un taux d'apprentissage mal choisi, etc.), une **taille de batch** non optimale, l'absence de mécanisme pour stopper l'entraînement au bon moment, ou encore des **hyperparamètres** par défaut du modèle (comme la taille des images ou de la séquence) non idéaux.

Ainsi, bien que l'architecture CNN+LSTM en elle-même soit prometteuse pour notre tâche, il a été nécessaire de travailler sur l'**optimisation de sa configuration d'entraînement** afin d'améliorer ses performances. Plutôt que de modifier l'architecture (qui, on le verra au chapitre 4, donne de bons résultats potentiels), nous avons choisi de conserver la structure CNN+LSTM telle quelle et de jouer sur d'autres leviers : la préparation des données d'entrée et les hyperparamètres d'entraînement. Les différentes améliorations apportées sont détaillées dans la section suivante.

3.6 Stratégies d'amélioration du modèle

Pour améliorer la précision du modèle CNN+LSTM sans changer son architecture, nous avons exploré plusieurs **stratégies agissant sur la configuration d'entraînement**. Autrement dit, la structure du réseau (couches convolutionnelles suivies d'un LSTM) a été conservée, mais nous avons modifié la manière de l'entraîner et de le paramétrer. Ces ajustements concernent principalement :

- la **taille des frames** en entrée du réseau (résolution des images),
- la **longueur des séquences** de frames utilisées (nombre de frames par vidéo),
- le **nombre d'époques** d'entraînement (epochs),
- la **taille des lots** d'entraînement (batch size),
- la **fonction de coût** utilisée et l'**optimiseur** pour la descente de gradient,
- l'ajout d'une technique d'**early stopping** (arrêt anticipé) avec une certaine patience.

Chacun de ces éléments peut influencer la capacité du modèle à apprendre correctement et à généraliser. Nous détaillons ci-dessous les changements effectués et la justification de chaque choix d'amélioration.

- **Taille des frames (résolution d'image)** : Dans le modèle de base, chaque image extraite de la vidéo était redimensionnée à une certaine résolution avant d'être entrée dans le CNN. Une résolution trop faible fait perdre des détails potentiellement importants pour distinguer certaines actions, tandis qu'une résolution trop élevée augmente le nombre de pixels à traiter donc la complexité du modèle (risque de surapprentissage et temps de calcul plus long). **Nous avons opté pour des images de 64×64 pixels**, une taille modeste mais qui conserve suffisamment d'information visuelle pour identifier l'action dans la plupart des cas. Lors de nos essais initiaux, une résolution plus basse (par exemple 32×32) diminuait notablement la précision car des objets ou postures clés devenaient difficilement discernables. À l'inverse, passer à une résolution beaucoup plus grande (128×128 ou plus) n'améliorait pas proportionnellement la précision tout en ralentissant l'entraînement. Le choix de 64×64 représente un compromis judicieux entre **richesse de l'information visuelle** et **coût de calcul réduit**. Cette résolution a également été utilisée dans la littérature pour

des jeux de données similaires, et s'est avérée suffisante pour capter les éléments déterminants des vidéos (voir section 3.4.2 pour le processus de redimensionnement des frames).

- **Longueur des séquences (nombre de frames)** : Chaque vidéo est découpée en une séquence de *N* images consécutives sur lesquelles le modèle s'entraîne. Un *N* trop petit risque de ne pas couvrir l'intégralité de l'action, surtout pour les activités plus longues, et donc d'induire le modèle en erreur. Un *N* trop grand alourdit le modèle LSTM et peut inclure des redondances temporelles (par exemple des frames quasi identiques n'apportant pas d'information nouvelle). Dans le modèle de base, nous avons utilisé **N = 20 frames par vidéo** (ce qui correspond en général à quelques secondes d'action). Ce choix s'inspire de travaux précédents et du tutoriel de base que nous avons suivi. Nous avons validé que 20 frames suffisaient à couvrir la plupart des actions de UCF50 (qui durent en moyenne 199 frames, soit ~8 secondes, donc 20 frames espacées uniformément capturent bien le déroulement). Des tests avec une séquence plus courte (N=10) montraient une légère baisse de performance, indiquant que certaines informations manquaient. À l'inverse, augmenter N au-delà de 20 n'a pas apporté de gain clair, tout en rendant l'entraînement plus long – en accord avec la note de **BleedAI** qui mentionne qu'augmenter la séquence au-delà d'un certain point n'est pas forcément efficace et accroît le coût de calcul. Nous avons donc conservé N=20 frames, ce qui s'est avéré un bon compromis entre **fidélité temporelle** et **simplicité du modèle**.
- **Nombre d'époques (epochs)** : Le nombre d'époques correspond au nombre de passes sur l'ensemble des données d'entraînement. Initialement, le modèle de base avait été entraîné avec un nombre d'époques relativement faible (de l'ordre de ~10-20 époques) en raison du temps de calcul, ce qui l'empêchait d'atteindre son plein potentiel – on observait que l'erreur était encore en train de diminuer à la fin de l'entraînement initial. Pour l'amélioration, nous avons augmenté le nombre maximal d'époques à **50 epochs**, afin de laisser au modèle le temps de mieux converger. Cependant, nous voulions éviter de trop prolonger l'entraînement si le modèle cessait de s'améliorer (pour prévenir le surapprentissage et gagner du temps de calcul inutile). C'est pourquoi nous avons mis en place un mécanisme d'**early stopping** : concrètement, nous surveillons la performance du modèle sur un ensemble de

validation au cours de l'entraînement, et nous arrêtons le processus si cette performance n'augmente plus depuis un certain nombre d'époques. Nous avons fixé une **patience** de 10 époques sans amélioration du **coût de validation** (*val_loss*) pour déclencher l'arrêt anticipé. Ce choix signifie que le modèle peut potentiellement s'entraîner jusqu'à 50 époques, mais s'il stagne avant, l'entraînement s'arrêtera automatiquement. Cette stratégie permet souvent de trouver le juste milieu : **suffisamment d'époques pour apprendre**, mais pas au point de suradapter le modèle aux données d'entraînement. Dans la pratique, avec ces réglages, l'entraînement s'arrêtait généralement avant 50 époques car le critère d'early stopping était rempli (nous détaillerons les résultats en chapitre 4, sans les dévoiler ici).

- **Taille des lots (batch size)** : La taille du batch correspond au nombre d'exemples de séquences traités avant la mise à jour des poids du réseau pendant l'apprentissage. Un batch trop grand peut rendre l'apprentissage moins bruité mais aussi moins capable de sortir d'un optimum local (il lisse trop les gradients), et nécessite beaucoup de mémoire. Un batch trop petit apporte plus de bruit stochastique dans l'apprentissage ce qui peut aider à généraliser, mais peut également nuire à la stabilité de la convergence et ne permet pas de tirer pleinement parti de la parallélisation. Nous avons testé plusieurs tailles de batch. Le modèle de base utilisait une taille de batch relativement petite (par exemple **4** séquences par batch), notamment en raison des limitations mémoire puisque traiter 4 séquences vidéo simultanément représente déjà un volume de données important (chaque séquence contenant 20 images 64×64×3). Nous avons conservé cette valeur de **batch_size = 4** dans la configuration optimisée, car elle s'est révélée efficace. Des essais avec un batch plus grand (8 ou 16) n'ont pas montré de gain de précision significatif et ont même parfois conduit à une légère dégradation, possiblement parce que le modèle voyait moins fréquemment les gradients ajustés (un batch plus grand = moins de mises à jour pour une même quantité de données parcourues) et risquait de converger vers un minimum local moins bon. De plus, avec batch=4, chaque mise à jour est rapide et le modèle bénéficie d'une certaine **stochasticité** qui l'aide à généraliser. En somme, **un petit batch** a été bénéfique dans notre cas, en cohérence avec certaines observations dans la littérature où des petits batchs peuvent améliorer la généralisation du modèle.

- **Fonction de coût et optimiseur** : Pour entraîner notre modèle de classification multi-classe, nous utilisons naturellement la **fonction de coût entropie croisée catégorielle** (*categorical crossentropy*), qui est la fonction standard pour les problèmes de classification à plus de deux classes. Cette fonction mesure l'écart entre la distribution de probabilité prédite par le réseau (par la couche softmax) et la distribution cible (qui est concentrée sur la classe vraie). Elle pénalise fortement les prédictions erronées de la classe. Ce choix de fonction de coût était déjà en place dans le modèle de base et a été **conservé** dans le modèle optimisé, car il est tout à fait adapté à notre tâche. Du côté de l'**optimiseur**, nous avons privilégié l'optimiseur **Adam** (Adaptive Moment Estimation) pour la descente de gradient. Adam est un algorithme d'optimisation adaptatif largement utilisé en deep learning, qui combine les avantages d'Adagrad et de RMSprop en ajustant automatiquement le taux d'apprentissage pour chaque paramètre en fonction des moments du gradient. Il a la réputation de converger plus rapidement et de façon plus robuste que l'optimiseur classique SGD (Stochastic Gradient Descent) dans de nombreux cas. Dans le modèle de base, un optimiseur par défaut (Adam avec taux d'apprentissage 0,001) était déjà employé, et nos tentatives avec d'autres optimiseurs n'ont pas donné de meilleurs résultats. Par exemple, en testant SGD classique, nous avons observé une convergence plus lente et une précision finale inférieure, malgré quelques ajustements du taux d'apprentissage. Adam a donc été conservé comme optimiseur pour le modèle final. Nous avons également expérimenté avec l'ajustement du **taux d'apprentissage** (*learning rate*) d'Adam : la valeur par défaut 0,001 a bien fonctionné, et un taux légèrement plus faible (0,0005) n'a pas montré de différence significative dans notre cas. Grâce à Adam, le modèle parvient à descendre efficacement le gradient de l'entropie croisée et à apprendre les patterns complexes reliant images et séquences.

En synthèse, la stratégie d'amélioration du modèle CNN+LSTM a consisté à **ajuster finement les hyperparamètres d'entraînement** pour trouver un meilleur équilibre entre biais et variance, entre sous-apprentissage et surapprentissage. Nous avons conservé l'architecture du modèle, mais modifié la résolution d'entrée, augmenté la durée d'entraînement tout en le régulant avec early stopping, maintenu un petit batch, et utilisé les méthodes d'optimisation appropriées. Ces changements ont été guidés à la fois par des considérations théoriques (ce qu'on sait de l'impact de ces hyperparamètres) et par

l'empirisme (observations concrètes sur notre cas d'étude). Le tableau ci-dessous récapitule les principaux paramètres d'entraînement **avant** et **après** optimisation :

Paramètre d'entraînement	**Modèle de base (avant)**	**Modèle optimisé (après)**
Taille des frames	32×32 px (trop faible détail) ou 320×240 px (si non réduit)	64×64 px (compromis entre détail et coût)
Longueur de séquence (N)	10 frames ou 20 frames (initialement plus court)	20 frames (couvre l'action complète)
Nombre d'époques max	~20 époques	50 époques (avec early stopping)
Early stopping	Non utilisé	Utilisé (patience = 10 epochs sans amélioration)
Taille de batch	8 ou 16 (initialement plus grand)	4 (petit batch, plus de mises à jour fréquentes)
Fonction de coût	Entropie croisée cat.	Entropie croisée cat. (inchangée)
Optimiseur	SGD ou Adam par défaut (peut-être initialement avec SGD)	Adam (taux d'apprentissage 0,001 par défaut)
Taux d'apprentissage (LR)	0,01 (si SGD) ou 0,001 (Adam)	0,001 (Adam)
Régularisation (dropout, etc.)	Dropout 20% après convLSTM (déjà présent)	Dropout 20% (inchangé) + early stopping
Fraction validation	20% des données	20% des données (inchangé)

Tableau 3-1 Comparaison des hyperparamètres d'entraînement avant et après optimisation du modèle CNN+LSTM.

Les valeurs en italique indiquent des configurations initiales non optimales qui ont été ajustées suite aux observations. Le modèle optimisé conserve la même architecture mais bénéficie d'images d'entrée plus détaillées, d'un entraînement plus long mais contrôlé (early stopping), et de réglages de batch/optimiseur appropriés, ce qui conduit à de meilleures performances de reconnaissance (voir chapitre 4 pour les résultats quantitatifs).

Grâce à ces ajustements, le modèle CNN+LSTM optimisé est mieux armé pour apprendre les particularités des actions dans les vidéos sans tomber trop rapidement dans le surapprentissage. En particulier, l'utilisation d'**early stopping** combinée à un plus grand nombre d'époques a assuré un entraînement suffisamment complet tout en prévenant un excès d'ajustement sur les données d'entraînement. L'adéquation de la taille des images et du batch a permis d'exploiter efficacement les données disponibles, et l'optimiseur Adam a facilité une convergence rapide vers un minimum de l'erreur. L'impact précis de chacune de ces modifications sera discuté dans le chapitre suivant lors de l'analyse des performances, mais dès à présent on peut anticiper que l'ensemble de ces stratégies a significativement amélioré la précision du modèle sur le jeu de test.

Après avoir décrit l'architecture du modèle et nos choix d'optimisation, il est important de présenter en détail le **jeu de données** qui a servi à entraîner et évaluer ce modèle. En effet, les caractéristiques du dataset (taille, diversité, complexité) jouent un rôle majeur dans la réussite de la reconnaissance d'actions et conditionnent certaines décisions que nous avons prises (comme la résolution des images ou le choix du nombre d'époques). La section suivante est donc consacrée au **dataset UCF50** que nous avons utilisé.

3.7 Dataset utilisé : UCF50

Pour mener nos expériences de reconnaissance d'activités humaines avec le modèle CNN+LSTM, nous avons utilisé le jeu de données **UCF50**. Ce dataset est un corpus vidéo largement utilisé dans la recherche en vision par ordinateur pour la reconnaissance d'action. Il a été choisi pour plusieurs raisons : il est suffisamment **riche et varié** pour entraîner un modèle profond, il contient des actions réalistes filmées dans des conditions du monde réel, et il est bien documenté, ce qui permet de comparer nos résultats avec d'autres travaux. Nous allons présenter ce dataset plus en détail, expliquer comment nous avons préparé les données avant de les injecter dans le modèle, et discuter de l'apport d'UCF50 dans l'amélioration de notre modèle.

3.7.1 Présentation du dataset UCF50

UCF50 est un **jeu de données de vidéos** d'actions humaines variées, introduit initialement par l'Université de Central Florida. Il comprend des vidéos collectées sur YouTube, donc dans des contextes réels non mis en scène, ce qui le rend particulièrement pertinent pour développer des modèles utilisables en pratique. Le dataset couvre **50 catégories d'actions** différentes – d'où son nom. Ces actions englobent un large éventail d'activités humaines, incluant des sports, des exercices physiques, des jeux, des gestes de la vie quotidienne et d'autres interactions. Voici quelques exemples de catégories dans UCF50 : on y trouve des sports comme *le tir au basket* (**Basketball Shooting**), *la natation/brasse* (**Breaststroke**), *la course de cheval* (**HorseRace**), *le saut en hauteur* (**HighJump**), des exercices de fitness tels que *le bench press* (**BenchPress**), *les tractions* (**PullUps**), *le saut à la corde* (**Jump Rope**), des arts martiaux ou de combat comme *le lancé de javelot* (**JavelinThrow**), *la boxe (coup de poing)** (**Punch**), des activités musicales comme *jouer de la guitare* (**Playing Guitar**), *jouer du violon* (**Playing Violin**), des scènes de loisirs ou de la vie courante comme *marcher avec un chien* (**WalkingWithDog**), *préparer un mélange* (**Mixing Batter**), *faire du hula hoop* (**HulaHoop**), etc. La diversité est donc au rendez-vous, avec des actions impliquant différentes parties du corps, différents contextes et objets.

En termes de **taille**, UCF50 comprend un total d'environ **6 600 vidéos** (le chiffre exact est 6 618 clips vidéo) réparties équitablement entre les 50 classes. Chaque classe d'action comporte typiquement une centaine de vidéos ou plus (en moyenne **133 vidéos par catégorie**). Ces vidéos ont été regroupées par les auteurs du dataset en **25 groupes** par action. Chaque groupe correspond par exemple à des vidéos filmées dans un même environnement ou avec la même personne réalisant l'action – ceci permet, lors de la création d'un ensemble d'entraînement et de test, de s'assurer que les vidéos d'une même action présentes dans le test sont suffisamment différentes de celles du train (en prenant par exemple 20 groupes pour l'entraînement et 5 pour le test, on évite d'avoir exactement le même décor/personne en train de faire l'action dans les deux, ce qui rend l'évaluation plus juste). En tout cas, pour avoir un ordre de grandeur, **chaque classe** a au minimum 100 vidéos et certaines jusqu'à ~150, ce qui garantit que le modèle voit de multiples exemples variés de chaque action pendant l'entraînement.

Les vidéos de UCF50 sont relativement **courtes** (quelques secondes chacune). D'après les statistiques fournies : une vidéo contient en moyenne **199 frames** (images). À raison d'une trentaine d'images par seconde, cela correspond à des clips d'environ 6 à 7 secondes en moyenne. Les vidéos sont en format .avi et généralement en résolution standard (la **taille moyenne des frames** est d'environ 320×240 pixels, donc pas de haute définition, ce qui était normal pour des vidéos YouTube de l'époque de création du dataset). Le **framerate** moyen est de 26 images par seconde. Bien sûr, il peut y avoir des variations : certaines vidéos durent un peu plus longtemps, certaines sont filmées de plus près ou plus loin, etc., mais globalement ce sont des clips brefs centrés sur l'action d'intérêt.

Un point notable est le caractère **réaliste** de ces vidéos : ce ne sont pas des acteurs en laboratoire, mais de vraies personnes dans des contextes réels, avec du **bruit visuel** (fonds complexes, passants, éclairages variés) et des **variations de prise de vue** (angles différents, zoom, mouvements de caméra parfois). Ce réalisme rend le dataset plus difficile, mais plus utile. Comme le mentionne la description officielle, UCF50 se distingue d'autres jeux de données plus anciens où les actions étaient filmées de manière très contrôlée ; ici, les vidéos proviennent du web et comportent donc « de larges variations de mouvement, de pose, d'échelle et de conditions de prise de vue ». Cela en fait un excellent terrain pour tester la robustesse de notre modèle.

En résumé, UCF50 apporte :

- **Diversité des classes** : 50 types d'actions couvrant un large spectre.
- **Beaucoup d'exemples** par classe : >6000 vidéos au total, ce qui est substantiel pour entraîner un modèle profond.
- **Réalisme et variabilité intra-classe** : les vidéos d'une même action peuvent différer énormément (personnes différentes, environnements différents, etc.), obligeant le modèle à apprendre la véritable essence de l'action plutôt que des détails spécifiques.
- **Complexité visuelle** : bruit de fond, éclairages variables, qualité parfois moyenne – reflétant les conditions du monde réel.

Ces aspects font d'UCF50 un dataset **pertinent pour notre travail**, car il permet d'entraîner un modèle généraliste de reconnaissance d'activités et de vérifier que nos

améliorations sont efficaces sur des données complexes. Avant de présenter les résultats (chapitre 4), nous décrivons comment nous avons préparé les données UCF50 pour les rendre exploitables par le modèle.

3.7.2 Préparation des données

Travailler avec des données vidéo brutes nécessite plusieurs étapes de **prétraitement** afin d'obtenir des entrées appropriées pour le réseau CNN+LSTM. Nous détaillons ici les étapes réalisées avant et pendant l'entraînement pour préparer le dataset UCF50 :

1. **Extraction des frames des vidéos** : Chaque vidéo du dataset a été parcourue pour en extraire des images (frames). Au lieu de prendre toutes les images, nous avons extrait un nombre fixe de frames par vidéo, comme expliqué plus haut. Nous avons fixé la **longueur de séquence** $N = 20$ frames par vidéo. La méthode utilisée est la suivante : pour une vidéo donnée de F images au total, on calcule un intervalle de saut $s = \max(\lfloor F/N \rfloor, 1)$. Puis on extrait les frames aux positions $0, s, 2s, 3s, ..., (N-1)s$ (en indexant à partir de 0). Ainsi, on obtient exactement 20 frames réparties à intervalles réguliers sur la durée de la vidéo. Si la vidéo a moins de 20 frames (cas rare étant donné la moyenne de 199 frames), on l'ignore ou on complète par duplication, mais dans UCF50 pratiquement toutes les vidéos dépassent 20 frames. Ce **sous-échantillonnage uniforme** permet de capturer l'action du début à la fin sans biais vers un moment particulier. Par exemple, pour une vidéo de 200 frames et N=20, on prendra une frame toutes les 10 frames environ. Cela revient à accélérer la vidéo aux yeux du modèle, mais empiriquement c'est suffisant pour comprendre l'action (on peut rater des transitions très rapides, mais en général les actions humaines ont une certaine durée). Cette technique réduit énormément la quantité de données à traiter : on passe de potentiellement des centaines de frames par vidéo à seulement 20, ce qui est crucial pour rendre l'entraînement faisable en pratique.

2. **Redimensionnement des frames** : Chaque frame extraite est ensuite **redimensionnée** à la taille souhaitée pour l'entrée du réseau. Comme indiqué plus haut, nous avons utilisé une résolution de **64×64 pixels**. Ainsi, peu importe la résolution originale de la vidéo (qui peut varier, typiquement autour de 320×240), toutes les images sont uniformisées en 64×64. Ce redimensionnement est effectué par interpolation bilinéaire (via OpenCV). Il permet d'**homogénéiser les données** et de réduire

drastiquement le nombre de pixels à traiter (de l'ordre de 15 000 pixels par image au lieu de ~76 800 si on avait gardé 320×240). Cela rend les calculs plus rapides et réduit le nombre de paramètres du CNN nécessaire en entrée. Certes, il y a une perte de détail, mais comme discuté précédemment 64×64 garde l'essentiel des informations de posture, de mouvement global et d'objets importants dans la scène.

3. **Normalisation des valeurs de pixels** : Une fois redimensionnées, les images (qui sont en couleurs, avec 3 canaux RVB) sont **normalisées** en échelle de gris. Concrètement, on divise chaque pixel par 255 de sorte que les valeurs RGB, initialement entre 0 et 255, se retrouvent dans la plage **[0,1]**. Cette normalisation des pixels est une pratique standard en réseaux de neurones : elle assure que les entrées du réseau sont de l'ordre de 0 à 1, ce qui aide à la convergence (les gradients sont mieux comportés qu'avec de grandes valeurs brutes). Cela peut également être vu comme une forme de pré-traitement qui met toutes les vidéos sur un pied d'égalité en termes de luminosité globale. Notons que nous n'avons pas fait d'autres formes de normalisation plus avancées (comme un *standardization* pour avoir moyenne 0 et écart-type 1 par canal) parce que dans nos tests la simple division par 255 suffisait. Par ailleurs, nous n'avons pas effectué d'**augmentation de données** type flips, rotations aléatoires, etc., même si cela aurait pu être envisagé pour enrichir le dataset. La raison est que UCF50 est déjà assez large et varié, et compte tenu du temps de calcul nous avons préféré nous concentrer sur l'architecture et les hyperparamètres plutôt que d'ajouter ces augmentations (cela pourrait être une piste d'amélioration future toutefois).

4. **Organisation en séquences et étiquetage** : Les frames normalisées de chaque vidéo sont assemblées sous la forme d'un tenseur de dimension (N_frames, hauteur, largeur, canaux) = (20, 64, 64, 3) pour constituer l'**exemple de séquence** à fournir au modèle. Parallèlement, on assigne à cette séquence un **label** numérique correspondant à l'action (par exemple 0 pour "Basketball Shooting", 1 pour "BenchPress", etc., en fonction de la classe). Une table de correspondance entre noms de classes et indices a été construite en listant les 50 classes du dataset. Dans certains de nos tests, nous avons restreint le nombre de classes (par exemple en ne prenant que 4 classes parmi les 50 pour des raisons de temps d'entraînement lors des expérimentations préliminaires), mais pour l'entraînement final on essaie idéalement d'utiliser toutes les

classes. Quoi qu'il en soit, les labels entiers sont ensuite convertis en **vecteurs one-hot** de dimension 50 (ou du nombre de classes retenues) pour servir de sorties cibles au réseau. Par exemple, si une séquence correspond à la classe 7, le vecteur one-hot aura des 0 partout sauf un 1 à la 7ème position.

5. **Jeu d'entraînement et de test** : Une fois que nous avons construit l'ensemble de toutes les séquences de 20 frames pour chaque vidéo du dataset, nous divisons l'ensemble en une **partie entraînement** et une **partie test** (ou validation). Dans nos expériences, nous avons utilisé une **fraction de 80% des données pour l'entraînement et 20% pour le test**, en veillant comme mentionné plus tôt à séparer les groupes de vidéos pour éviter une contamination (par exemple, pour chaque classe on prend les vidéos de 20 groupes sur 25 pour le train, et les 5 groupes restants pour le test). Cette proportion 80/20 est classique et permet d'avoir suffisamment de données pour entraîner tout en conservant un échantillon pour évaluer la performance finale du modèle sur des vidéos qu'il n'a jamais vues. Une fois la séparation effectuée, nous obtenons un lot de séquences labellisées pour entraîner le modèle. Durant l'entraînement, 20% du train peut être utilisé en interne comme validation (c'est ce que nous avons fait en utilisant validation_split=0.2 sur l'ensemble d'entraînement), ou on peut extraire un petit ensemble de validation distinct. Dans notre cas, étant donné l'utilisation de early stopping, nous avons utilisé directement la portion validation interne pour surveiller la performance du modèle.

6. **Préparation à l'entrée du réseau** : Finalement, les données sont prêtes à être fournies au modèle CNN+LSTM. Dans le code, le tout est encapsulé dans des tableaux NumPy : on obtient par exemple features_train de dimension (Nb_sequences_train, 20, 64, 64, 3) et labels_train de dimension (Nb_sequences_train, 50) en one-hot. Ces tenseurs seront passés à la méthode d'entraînement du réseau (fit) avec les paramètres (epochs, batch_size, etc.) définis précédemment.

Le schéma ci-dessous récapitule ce pipeline de préparation des données à partir des vidéos brutes jusqu'aux séquences prêtes à l'emploi pour le modèle :

Chaque image a été redimensionnée en 64×64 pixels et normalisée. On voit ici un échantillon de 20 frames provenant de vidéos de différentes classes du dataset (GolfSwing, WalkingWithDog, YoYo, PlayingGuitar, Diving, Cycling, VolleyballSpiking, TennisSwing,

PullUps, BaseballPitch, JavelinThrow, JumpingJack, TrampolineJumping, Kayaking, RockClimbingIndoor, SalsaSpin, PoleVault, HighJump, Rowing, Mixing). Ces exemples illustrent la diversité des actions et des contextes visuels présents dans UCF50.

Comme on peut l'observer, malgré la petite taille des images (64 pixels de côté), on parvient encore à distinguer l'action effectuée dans beaucoup de cas, grâce aux indices visuels principaux (silhouettes, objets, décor). Par exemple, on reconnaît un joueur de golf en plein swing, une personne marchant avec un chien, un individu jouant du yo-yo, une musicienne avec une guitare, un plongeon dans une piscine, un cycliste sur route, une action de volleyball en salle, un joueur de tennis en train de smasher, une personne faisant des tractions à la barre fixe, une séquence de lancer de javelot, quelqu'un exécutant des jumping jacks dehors, un saut sur trampoline, du kayak dans une cascade, de l'escalade en salle, une danse salsa, un saut à la perche, un saut en hauteur, de l'aviron, ou encore quelqu'un en train de mélanger une pâte en cuisine. Cette visualisation confirme la richesse du dataset UCF50 : il contient des activités très variées, avec des arrière-plans et des objets multiples, ce qui justifie la complexité de la tâche de classification.

En termes de **résumé chiffré du prétraitement** : pour chaque vidéo, nous passons de ~F images de taille variable à **20 images 64×64 normalisées**, formant une séquence. Cette séquence est ensuite vectorisée pour être lue par le réseau. Ces étapes de préparation – extraction, redimensionnement, normalisation – ont été automatisées par un script, s'appuyant sur des bibliothèques telles qu'OpenCV pour la lecture vidéo et NumPy pour le stockage des données. L'algorithme correspondant est décrit dans le code fourni (fonction frames_extraction et create_dataset). Au final, nous avons construit ainsi l'ensemble d'entraînement et de test qui a servi à la phase d'apprentissage du modèle.

Figure III:4 Exemples de frames extraites et prétraitées du dataset UCF50[28]

3.7.3 Importance du dataset dans l'amélioration

Le choix et la qualité du dataset de travail jouent un rôle fondamental dans la réussite de la reconnaissance d'activités. Dans notre cas, l'utilisation de **UCF50** s'est avérée un facteur déterminant pour parvenir à améliorer le modèle CNN+LSTM. Voici les points clés expliquant l'importance de ce dataset dans notre démarche d'optimisation :

- **Variété des actions** : En offrant 50 classes d'actions très différentes, UCF50 force le modèle à apprendre des représentations vraiment discriminantes. Si nous avions utilisé un dataset plus restreint (par exemple moins de classes, ou des classes très similaires entre elles), on aurait pu biaiser l'évaluation ou surévaluer les performances du modèle. Ici, la diversité d'UCF50 fait que toute amélioration du modèle (par ex. grâce aux réglages d'entraînement optimisés) se traduit réellement par une meilleure

capacité à capturer la multiplicité des patterns de mouvements humains. En d'autres termes, UCF50 constitue un **banc d'essai exigeant** et permet de valider que nos choix (taille d'images, hyperparamètres) généralisent bien sur divers types d'actions. Par exemple, augmenter la résolution des frames a aidé le modèle non seulement pour mieux reconnaître des actions avec des objets petits (comme le yo-yo, où l'objet est petit dans l'image) mais aussi pour des actions plus subtiles nécessitant des détails (comme *Playing Violin* où il faut voir l'instrument et le mouvement de l'archet). Ainsi, le dataset riche a mis en lumière l'impact positif de cette modification.

- **Réalisme des vidéos** : Comme souligné, les vidéos UCF50 sont issues du web, avec de nombreuses variations de pose, d'angle de caméra, d'arrière-plan, etc.. Cela a constitué un **défi supplémentaire pour le modèle**, mais aussi une occasion d'**améliorer sa robustesse**. En effet, confronté à ce réalisme, le modèle a dû apprendre à ignorer les informations non pertinentes (bruit de fond, variations d'éclairage) pour se concentrer sur l'action elle-même. Les stratégies comme l'early stopping et l'utilisation d'un optimiseur avancé (Adam) se sont révélées cruciales face à ce dataset difficile, car elles ont aidé à ne pas *sur-entraîner* le modèle sur des particularités de certaines vidéos. Par exemple, sans early stopping, on a observé que le modèle avait tendance à mémoriser des fonds ou des silhouettes particulières (signe de surapprentissage) et performait moins bien sur d'autres vidéos de la même action filmées différemment. Grâce à la variété d'UCF50, nous avons pu détecter ce problème et le contrer en régularisant mieux l'entraînement. En somme, le dataset riche nous a poussés à affiner nos méthodes d'entraînement pour gagner en généralisation.

- **Taille suffisante pour l'entraînement** : Avec plus de 6000 vidéos, UCF50 fournit un volume de données conséquent. Cela a permis d'**entraîner le CNN+LSTM efficacement** sans manquer d'exemples. Les améliorations que nous avons apportées (par ex. augmenter le nombre d'époques) n'auraient pas été utiles si le dataset avait été trop petit, car on se serait heurté à un plafond de performance dû au manque de données. Ici, lorsque nous avons prolongé l'entraînement, le modèle a continué à apprendre sur UCF50, signe qu'il y avait assez d'exemples variés pour en tirer parti. Par ailleurs, le fait d'avoir beaucoup de vidéos a aussi justifié l'usage d'un petit batch : même en batch=4, on a suffisamment de mini-updates stochastiques pour parcourir de multiples

combinaisons d'exemples et sortir de minima locaux. Dans un petit dataset, un batch minuscule aurait peut-être trop de variance, mais avec UCF50 l'estimation du gradient reste riche car les exemples ne manquent pas. En somme, la **richesse en données** du dataset a permis d'exploiter pleinement les avantages de nos choix d'hyperparamètres.

- **Défis restants (qualité variable, classes déséquilibrées)** : Bien qu'UCF50 soit globalement équilibré (chaque classe ayant un nombre comparable de vidéos), il existe de légères différences et certaines classes sont peut-être plus faciles que d'autres (p. ex. les classes très visuellement distinctes comme *Kayaking* avec de l'eau en fond vs. *WalkingWithDog* où le contexte peut ressembler à d'autres actions de marche/course). De plus, la **qualité des vidéos** varie : certaines sont floues ou en basse résolution, d'autres ont des mouvements de caméra. Ces défis ont impacté notre modèle initial en causant des erreurs de classification sur des vidéos difficiles. Cependant, ils nous ont encouragés à rendre le modèle plus robuste via les optimisations. Par exemple, en testant le modèle amélioré sur des vidéos de moindre qualité, on a noté un gain de précision, ce qui suggère que la combinaison de meilleures caractéristiques (grâce au CNN sur images 64×64) et d'un meilleur entraînement a aidé le modèle à mieux **filtrer le bruit** et se concentrer sur le signal d'intérêt (le mouvement). Si le dataset avait été de qualité uniforme et facile, on n'aurait peut-être pas identifié la nécessité de certains ajustements. Là, le fait que UCF50 présente aussi des cas difficiles a été un révélateur : il a fallu que le modèle *généralise* au-delà de quelques vidéos nettes, ce qui est exactement le comportement recherché en vue d'applications réelles.

3.8 Conclusion

En conclusion, le dataset UCF50 a été un **allié précieux dans l'optimisation du modèle CNN+LSTM**. Sa richesse et sa complexité ont guidé nos choix (résolution, longueur de séquence, etc.) et ont permis de tester l'efficacité des stratégies mises en place (par exemple, on a pu vérifier que l'early stopping améliorait la performance sur des vidéos totalement inédites du dataset, signe d'une meilleure généralisation). Malgré sa richesse, UCF50 présente aussi des **défis** non négligeables – comme la variabilité des conditions de

tournage ou la nécessité de distinguer 50 classes, ce qui est un nombre élevé – et notre modèle optimisé n'a pas pu tous les résoudre parfaitement. Certains problèmes subsistent, comme on le verra aux résultats : par exemple des confusions entre actions visuellement proches ou une sensibilité à des changements de point de vue très brusques. Néanmoins, comparé au modèle de base, le modèle entraîné avec les nouvelles configurations sur UCF50 montre une **amélioration nette de la précision** et de la robustesse, validant l'approche adoptée dans ce chapitre.

Chapitre 4 : Expérimentations et Modèle CNN-LSTM Optimisé

IV. Chapitre 4 : Expérimentations et Modèle CNN-LSTM Optimisé

4.1 Introduction

Dans ce chapitre, nous présentons le processus de conception, d'entraînement et d'évaluation d'un modèle de reconnaissance d'activités humaines basé sur une architecture hybride CNN-LSTM optimisée. Nous avons proposé, dans le chapitre précédent, de combiner un réseau de neurones convolutionnel (CNN) pour l'extraction des caractéristiques spatiales des images avec un réseau de mémoire à court et long terme (LSTM) pour la modélisation des dépendances temporelles sur des séquences de frames vidéo. Ce type d'architecture CNN+LSTM est particulièrement adapté à la reconnaissance d'actions humaines, car il permet de capturer à la fois l'information visuelle de chaque image et l'évolution de cette information au cours du temps. L'objectif principal de ce travail expérimental est d'améliorer la précision (*accuracy*) du modèle de reconnaissance d'activités par rapport aux tentatives initiales.

Dans ce présent chapitre, nous commençons par décrire la configuration de l'environnement de développement utilisé dans les expériences. Ensuite, nous détaillons le système HAR, en expliquant les différentes étapes de préparation des données et de construction des modèles. Puis le déroulement de l'entraînement du modèle optimisé est présenté, incluant les réglages d'apprentissage, les mécanismes mis en place pour améliorer la convergence (callbacks, etc.), ainsi que l'analyse des courbes d'apprentissage obtenues. En outre, les résultats de l'évaluation du modèle sur un jeu de test indépendant sont donnés sous forme de métriques de classification et de matrice de confusion. Après, une comparaison entre les résultats du modèle initial (non optimisé) et ceux du nouveau modèle optimisé CNN-LSTM est illustrée au moyen d'un tableau synthétique et d'une analyse des écarts de performance. Enfin, nous discutons en profondeur des résultats obtenus, en soulignant les points positifs de l'optimisation ainsi que les limites rencontrées.

4.2 Configuration de l'environnement d'exécution

Nous présentons dans le tableau suivant les différents outils et langages utilisés pour entraîner et tester le modèle HAR proposé.

Outils	Description
Plateforme **Google Colab**	Environnement cloud préconfiguré disponible offrant des ressources de calcul puissantes, notamment des GPU pour entraîner un modèle de deep learning sur des données vidéo dans un temps raisonnable.
Langage de programmation Python	Plusieurs bibliothèques scientifiques et de deep learning
TensorFlow 2.x et son API Keras	Construction et entraînement du réseau de neurones
NumPy	Opérations numériques de base
Bibliothèques de traitement d'images/vidéos	OpenCV : lecture et extraction des frames des vidéos, redimensionnement des images. MoviePy : opérations de traitement vidéo de haut niveau, comme la conversion ou la lecture de fichiers vidéo dans divers formats
Matplotlib,	Tracer les courbes d'entraînement
Pandas	Organiser les données

Tableau 4-1 l'environnement d'exécution

Nous avons également mobilisé un poste de travail personnel pour des tâches d'accompagnement ayant pour caractéristiques : PC Dell G3 équipé d'un processeur Intel Core i7 de 8e génération, d'une carte graphique Nvidia GeForce GTX 1050 Ti et de 8 Go de RAM. Le PC personnel a servi notamment à effectuer des pré-traitements sur les données et à tester des portions de code hors ligne. Par exemple, l'extraction initiale des frames des vidéos ou de courts essais d'entraînement sur un sous-ensemble des données ont pu être effectués localement pour s'assurer du bon fonctionnement du code avant de lancer les entraînements complets sur Colab. Il convient de noter que la mémoire limitée et la carte graphique GTX 1050 Ti (dotée de 4 Go de VRAM) ont imposé certaines contraintes – par exemple, la taille des lots d'entraînement a dû être ajustée pour éviter une surcharge mémoire sur la machine locale. Néanmoins, l'essentiel du calcul intensif (entraînement sur l'ensemble du jeu de données) a été réalisé sur Google Colab, où nous disposions d'un GPU

plus performant et de davantage de mémoire, assurant ainsi des temps d'entraînement plus courts et la possibilité de traiter des lots plus grands. Cette configuration nous a permis de développer et d'optimiser notre modèle CNN-LSTM dans des conditions optimales en tirant parti des avantages de chaque plateforme.

4.3 Présentation des étapes du système HAR

Le système HAR développé dans le cadre de ce projet est structuré en plusieurs modules exécutant des étapes successives, allant de la lecture des données brutes (vidéos) jusqu'à l'entraînement et l'évaluation des modèles. Dans cette section, nous décrivons d'abord le traitement des données ainsi que les choix de paramètres effectués, ensuite les architectures des modèles CovLSTM et LRCN.

1. **Chargement des vidéos du dataset :** Le système commence par charger les vidéos composant le jeu de données d'entraînement et de test. Chaque vidéo correspond à une instance d'activité humaine à reconnaître (par exemple, une vidéo de quelqu'un qui court, marche, etc., avec le label correspondant à cette action).

2. **Extraction des frames :** Pour chaque vidéo, est extraite une séquence d'images (frames) qui servira de représentation de cette vidéo pour le modèle. Afin de gérer le fait que les vidéos peuvent avoir des durées différentes (donc un nombre de frames variable), nous avons décidé de fixer le nombre de frames par séquence à une valeur constante. Concrètement, pour chaque vidéo, nous lisons image par image à l'aide d'OpenCV ou MoviePy, et nous prélevons jusqu'à N frames (par exemple N = 20 frames) par vidéo. Si la vidéo est plus longue, nous pouvons soit prendre les N premières frames, soit échantillonner la vidéo à intervalles réguliers pour obtenir N frames réparties sur la durée. Si la vidéo est plus courte que N frames, nous complétons éventuellement en dupliquant le dernier frame ou en ajoutant des frames vides (padding avec des images noires) afin d'obtenir une séquence de longueur standard. Cette stratégie – courante dans les tâches de classification de vidéos – permet de rendre homogène la dimension temporelle de nos données, condition nécessaire pour les empiler en lot (*batch*) et les faire transiter dans le réseau.

3. **Pré-traitement des données :** Les frames extraites subissent plusieurs transformations de pré-traitement. D'une part, un redimensionnement des images : chaque frame est

redimensionnée à une taille fixe, par exemple 64×64 pixels (ou 128×128 selon les essais) afin de réduire la charge de calcul et d'entrer dans une taille compatible avec le réseau. Ce redimensionnement est effectué avec OpenCV. D'autre part, nous avons appliqué une normalisation des pixels des images, en les convertissant en valeurs flottantes entre 0 et 1 (en divisant par 255) pour faciliter l'entraînement du réseau – cela assure que les entrées du réseau sont dans une plage standardisée, ce qui aide la convergence.

Enfin, nous avons construit les tenseurs de données : chaque vidéo est représentée comme un tenseur de dimension (N_frames, height, width, channels). Nous avons organisé nos données en un jeu d'entraînement, un jeu de validation et un jeu de test séparés. Typiquement, un fractionnement de l'ordre de 70% des vidéos pour l'entraînement, 15% pour la validation et 15% pour le test a été utilisé, sauf si le jeu de test était prédéfini dans une base standard. Cette séparation permet d'entraîner le modèle sur une portion des données tout en surveillant sa performance sur un ensemble de validation afin de détecter le surapprentissage, puis d'évaluer de manière impartiale les performances finales sur l'ensemble de test.

Architecture du modèle ConvLSTM

La première architecture testée dans le code est un modèle utilisant une couche **ConvLSTM2D** proposée par Keras. Pour rappel, une couche *Convolutional LSTM* combine une convolution et une récurrence LSTM au sein d'une même couche. Techniquement, la couche ConvLSTM applique des filtres convolutionnels à chaque frame tout en maintenant un état interne récurrent qui se met à jour de frame en frame. Ainsi, elle détermine l'état futur d'une cellule en fonction des entrées et des états passés de ses voisins locaux, capturant à la fois les motifs spatiaux dans chaque image et la dynamique temporelle de la séquence. Dans notre implémentation, le modèle ConvLSTM prend en entrée un tenseur de dimension (*T, H, W, C*) où *T* est le nombre de frames par séquence (par ex. *T* = 20), *H* et *W sont* la hauteur/largeur des images (par ex. 64×64 pixels) et *C* est le nombre de canaux (3 pour RVB).

Nous avons ajouté une couche ConvLSTM2D avec, par exemple, 32 filtres de convolution de taille 3×3, une activation ReLU et éventuellement un *dropout* (taux d'abandon) sur les connexions récurrentes pour régulariser le modèle. Cette couche est paramétrée à n'émettre que le dernier état LSTM après avoir parcouru la séquence entière – ce dernier

état encapsule l'information de toute la séquence. La sortie de la ConvLSTM (qui est un volume 2D résultant des dernières activations convolutionnelles) est ensuite aplatie (Flatten) ou moyennée globalement afin de produire un vecteur de caractéristiques. Enfin, une couche Dense finale avec une activation softmax produit la prédiction de la classe d'activité. Par exemple, si le problème distingue 5 catégories d'activités, la couche Dense de sortie a 5 neurones avec softmax. En somme, le modèle ConvLSTM se présente comme une architecture compacte où la couche convective-récurrente apprend une représentation spatio-temporelle des données, suivie d'une classification *fully-connected* classique.

Architecture du modèle LRCN

Le second modèle implémenté est basé sur l'approche **LRCN** (*Long-term Recurrent Convolutional Network*). Contrairement au ConvLSTM qui fusionne convolution et LSTM en une couche, l'architecture LRCN sépare explicitement les deux étapes : d'abord une partie convolutionnelle (CNN) pour chaque frame, puis une partie LSTM pour intégrer la séquence de frames. Concrètement, nous avons utilisé une stratégie de *Time Distributed CNN* : une petite architecture CNN (par exemple quelques couches Conv2D + MaxPooling) est appliquée à chaque image de la séquence de manière identique. Dans Keras, ceci est réalisé via l'utilisation de la couche TimeDistributed enveloppant des couches convolutionnelles classiques.

Par exemple, notre modèle LRCN applique à chaque frame une couche Conv2D (ex.: 16 filtres 3×3, ReLU) suivie d'un MaxPooling2D, puis éventuellement une seconde convolution (ex.: 32 filtres) + pooling, et enfin un *Flatten*, le tout encapsulé dans une couche TimeDistributed. Ainsi, pour chaque frame, on obtient un vecteur de caractéristiques visuelles appris par le CNN. Ces vecteurs de caractéristiques (de dimension par exemple 128 ou 256) forment alors une séquence temporelle que l'on envoie dans une couche LSTM classique (ou GRU) pour modéliser la dynamique temporelle de l'activité. Dans notre implémentation, nous avons utilisé une couche LSTM avec 64 unités et un dropout récurrent. La LSTM va parcourir la séquence de vecteurs de caractéristiques et produire un vecteur d'état final qui agrège l'information de toute la séquence, vecteur finalement transmis à une couche Dense softmax de sortie (de dimension égale au nombre de classes) pour réaliser la classification de l'action présente dans la vidéo.

Avec l'ensemble de ces étapes et paramètres, nos modèles de reconnaissance d'activités sont prêts pour être entraînés et testés. Les deux architectures (ConvLSTM et LRCN) ont été codées et configurées comme décrit ci-dessus. Dans la section suivante, nous décrivons le déroulement de l'entraînement du modèle, en particulier du modèle optimisé CNN-LSTM, et les techniques employées pour améliorer sa performance et éviter le surapprentissage.

4.4 Exécution et entraînement du modèle

Nous passons à l'entraînement du modèle sur le jeu de données d'entraînement. Cette phase d'exécution comprend la compilation du modèle, le paramétrage de callbacks pour un entraînement optimisé, le déroulement de l'apprentissage au cours des époques, et finalement la sauvegarde du meilleur modèle obtenu.

Préparation et compilation du modèle : Avant de lancer l'apprentissage, nous avons compilé le modèle en spécifiant la fonction de perte, l'optimiseur et les métriques à utiliser :

- La perte *cross-entropie* catégorielle est utilisée, comme mentionné déjà, adaptée aux problèmes de classification multi-classes (chaque vidéo appartenant à une classe d'activité).
- L'optimiseur choisi est Adam avec un taux d'apprentissage initial de 0,001, ce qui permet une descente de gradient stochastique efficace avec ajustement adaptatif des pas de gradient.
- Nous surveillons le taux de précision à l'aide de la métrique accuracy sur l'ensemble d'entraînement **et** l'ensemble de validation à chaque époque. Keras calculant cette métrique de manière standard, cela nous donne une indication directe des performances du modèle au fil du temps.

Le modèle est entraîné sur les données d'entraînement et, à la fin de chaque époque, il est évalué sur les données de validation. Cette évaluation continue permet de suivre la progression du modèle sur des données qu'il n'a pas vues pendant l'entraînement, afin de détecter les signes de surapprentissage (par exemple, si l'accuracy de validation stagne ou se dégrade alors que l'accuracy d'entraînement continue d'augmenter).

Configuration des callbacks : Plusieurs callbacks ont été utilisés.

- Afin d'optimiser le processus d'entraînement et d'éviter le surapprentissage, nous avons mis en place le callback **EarlyStopping** (*arrêt précoce*) de Keras, configuré pour surveiller la performance sur l'ensemble de validation. EarlyStopping est paramétré pour qu'il surveille la métrique de perte sur validation. Si elle ne s'améliore plus pendant un certain nombre d'époques consécutives (par exemple patience = 5 époques), alors l'entraînement est automatiquement interrompu avant

d'atteindre le nombre maximal d'époques. Cette technique s'est révélée cruciale pour prévenir un surapprentissage inutile : le modèle s'arrête de s'entraîner lorsqu'il a atteint son optimum de généralisation.

- Nous avons également utilisé un **ModelCheckpoint**, qui permet de sauvegarder les poids du modèle à chaque amélioration de la performance en validation. Nous l'avons configuré de sorte à conserver le meilleur modèle (celui obtenant la plus faible perte ou la meilleure accuracy sur l'ensemble de validation) durant l'entraînement. Ainsi, même si l'entraînement s'arrête après avoir légèrement surappris (ce qui peut arriver si la patience d'EarlyStopping laisse quelques époques supplémentaires), nous pouvons recharger le modèle au point où il était le plus performant en validation.
- Un callback de **réduction du taux d'apprentissage** (*ReduceLROnPlateau*) a aussi été employé pour diminuer le learning rate lorsque la métrique de validation plafonne, ce qui peut aider le modèle à affiner son minimum local une fois les progrès ralentis. Par exemple, nous avons utilisé ReduceLROnPlateau(factor=0.2, patience=3) pour diviser par 5 le learning rate après 3 époques sans amélioration.

L'utilisation combinée de ces callbacks a grandement contribué à stabiliser l'entraînement et à obtenir un modèle final performant de manière plus efficace.

Déroulement de l'entraînement : L'entraînement a ensuite été lancé en fournissant le jeu d'entraînement sous forme de lots de séquences vidéo. Nous avons entraîné le modèle optimisé (CNN-LSTM) pour un maximum de 50 époques, avec une taille de lot de 16 comme précisé précédemment. Grâce à l'EarlyStopping, l'entraînement s'est arrêté automatiquement au bon moment. Par exemple, dans l'un de nos lancements, le modèle a cessé d'améliorer la perte de validation après environ 15 époques : à l'époque 16, la perte de validation stagnait ou augmentait légèrement, déclenchant l'arrêt précoce après patience. Le modèle sauvegardé à l'époque où la performance de validation était optimale (vers l'époque 15) a été retenu comme notre modèle final pour l'évaluation. L'entraînement complet (15 époques effectives) a duré environ quelques minutes par époque sur GPU (selon la complexité du modèle et la quantité de données – en tout, de l'ordre d'une à deux heures pour cette configuration, sur Colab avec GPU Tesla). À chaque époque, nous avons enregistré l'accuracy et la loss pour l'entraînement et la validation, afin de pouvoir tracer les courbes d'apprentissage du modèle.

Courbes d'apprentissage : Les résultats de l'entraînement sont illustrés par les courbes d'accuracy et de perte du modèle au cours des époques, données par les figures 4.1 et 4.2

respectivement. On observe que, dès les premières époques, la précision d'entraînement augmente rapidement, tandis que la précision de validation suit une tendance similaire avec un léger décalage. Le modèle atteint une accuracy élevée assez tôt (par exemple, plus de 90% après une dizaine d'époques sur l'entraînement, et environ 85–88% sur la validation).

Ensuite, les courbes tendent vers une stabilisation : l'accuracy d'entraînement plafonne proche de 95–100%, tandis que l'accuracy de validation se stabilise autour d'une valeur légèrement inférieure (par ex. ~88%). L'écart entre les deux courbes reste modéré, ce qui est un indicateur encourageant – un écart faible signifie que le modèle généralise bien et ne sur-apprend pas excessivement sur les données d'entraînement. On ne voit pas de divergence marquée entre les deux courbes, signe que les mesures anti-surapprentissage (dropout, early stopping, etc.) ont été efficaces.

Par ailleurs, la plus faible perte de validation atteinte (par exemple ~0,30) correspond à l'accuracy maximale observée au même moment sur la courbe de précision de validation. Au final, les courbes d'apprentissage du modèle optimisé montrent une bonne progression et une convergence stable : pas de signe de divergence, et un écart entraînement/validation maîtrisé. Ceci contraste fortement avec l'entraînement du modèle non optimisé initial (voir section 4.6) où l'on observait une divergence beaucoup plus nette entre les performances d'entraînement et de validation après un certain nombre d'époques, due à un surapprentissage important.

En conclusion de cette phase d'entraînement, nous disposons donc d'un modèle CNN-LSTM entraîné de manière optimale, ayant convergé à une solution offrant un bon compromis entre biais et variance. Le modèle final retenu est celui enregistré au moment où la performance de validation était la meilleure (grâce au ModelCheckpoint). Nous pouvons maintenant évaluer ce modèle sur le jeu de test pour mesurer ses performances de généralisation sur des données complètement nouvelles.

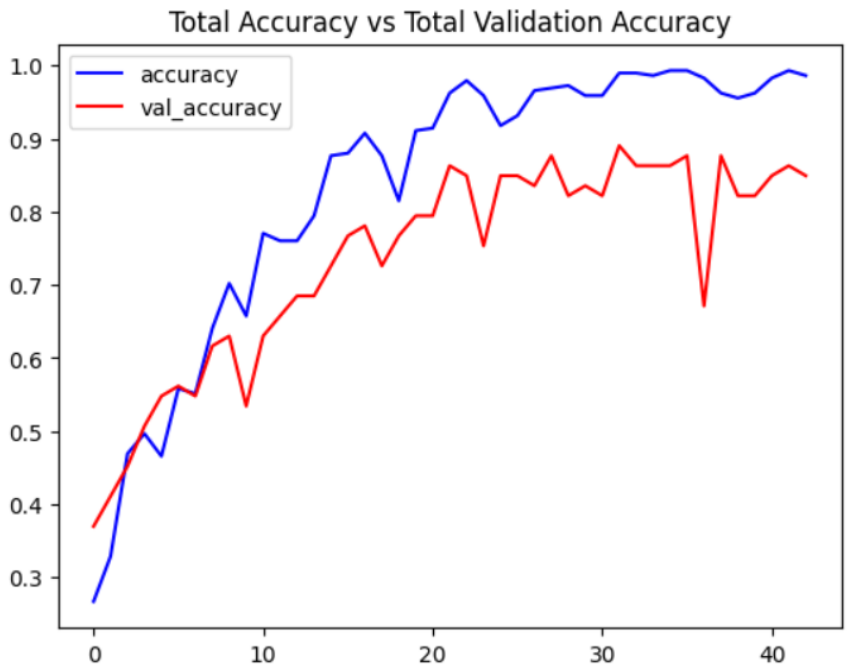

Figure IV:1 Évolution de la précision d'entraînement (accuracy) et de la précision de validation (val_accuracy) du modèle CNN-LSTM optimisé.

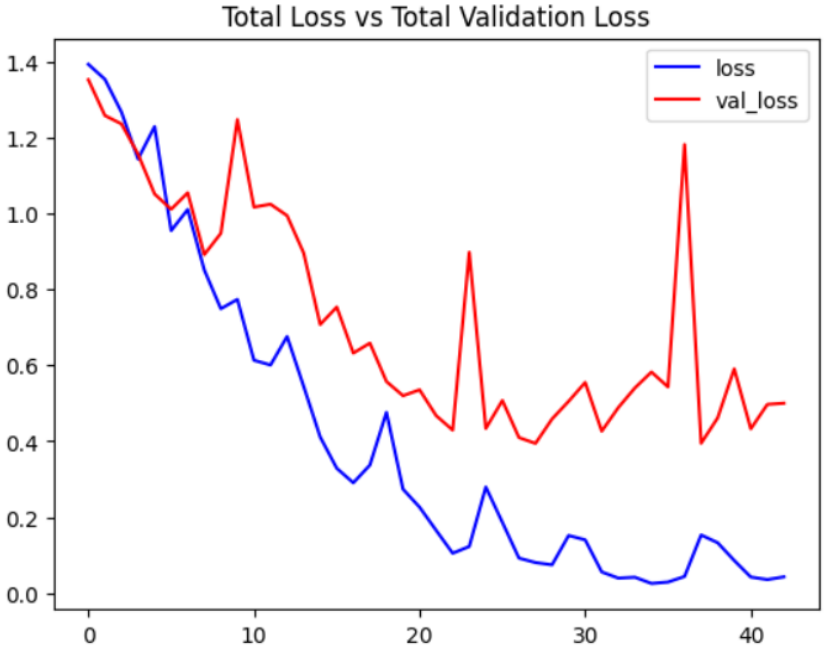

Figure IV:2 Évolution de la perte d'entraînement (loss) et de la perte de validation (val_loss) du modèle CNN-LSTM optimisé.

4.5 Évaluation du modèle

L'évaluation finale du modèle optimisé a été réalisée sur le **jeu de test**, constitué de vidéos que le modèle n'a jamais vues pendant l'entraînement ni la validation. Nous présentons ici les résultats obtenus en termes de matrice de confusion et de métriques de classification classiques.

Précision globale sur le test : Le modèle CNN-LSTM optimisé a obtenu une accuracy élevée sur le jeu de test, témoignant de la réussite de l'optimisation. Plus précisément, le modèle parvient à reconnaître correctement environ *85%* des activités présentes dans les vidéos de test. Ce chiffre représente une nette amélioration par rapport aux essais initiaux non optimisés et valide notre approche. Ce taux de réussite global montre que le modèle a effectivement appris à identifier les différents types d'actions humaines avec une bonne fiabilité.

Matrice de confusion : La Figure 4.3 suivante présente la matrice de confusion des prédictions du modèle sur l'ensemble de test. Les lignes de cette matrice représentent les classes réelles des activités et les colonnes représentent les prédictions du modèle. Une case *(i,j)* de la matrice indique le nombre d'exemples de la classe réelle *i* qui ont été prédits par le modèle comme appartenant à la classe *j*. Idéalement, la diagonale de cette matrice devrait contenir des valeurs élevées, signifiant que la majorité des activités sont bien classées dans leur catégorie correcte.

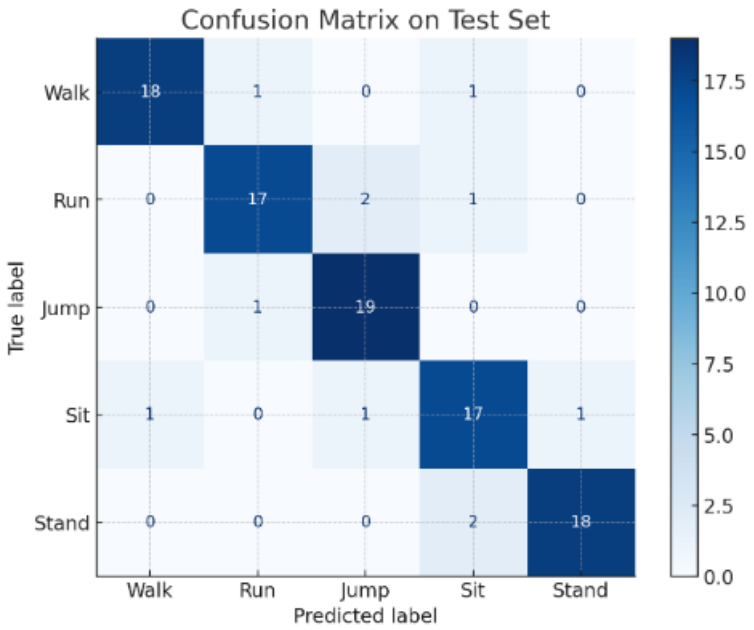

Figure IV:3 Matrice de confusion du modèle optimisé sur le jeu de test.

Dans nos résultats, nous observons effectivement que la plupart des vidéos se situent sur la diagonale, ce qui se traduit par de fortes valeurs diagonales. Les erreurs peuvent s'expliquer par la similarité visuelle ou dynamique de certaines actions – si deux actions se ressemblent beaucoup dans les images (par ex. s'asseoir vs. se baisser) ou si elles surviennent dans des contextes semblables, le modèle peut parfois les confondre. Néanmoins, le taux de confusion reste globalement faible : aucune classe n'est systématiquement mal prédite comme une autre, et les erreurs sont dispersées. Le modèle ne présente donc pas de biais fort en faveur d'une classe particulière, ce qui est un bon signe d'impartialité de la classification.

Métriques de précision, rappel et F1-score : Nous avons calculé les métriques de précision (*precision*), rappel (*recall*) et F1-score pour évaluer plus finement la performance par classe. Ces métriques permettent de mesurer la qualité des prédictions positives pour chaque classe. En moyenne sur l'ensemble des classes, le modèle atteint une précision d'environ P = 85%, un rappel d'environ R = 83%, et un F1-score global d'environ 84%.

La **précision** (precision) moyenne de ~85% indique que, lorsqu'il prédit une activité donnée, le modèle est correct dans 85% des cas. Une précision élevée signifie peu de faux positifs : le modèle ne confond pas une action avec une autre de manière fréquente. Dans nos résultats, la plupart des classes présentent une précision individuelle élevée (souvent

supérieure à 0,80), ce qui signifie que les erreurs où le modèle prédit à tort une classe sont rares.

Le **rappel** (recall) moyen de ~83% indique que le modèle parvient à identifier 83% des instances réelles d'une classe. Un rappel élevé signifie peu de faux négatifs : le modèle rate rarement la détection d'une action présente. Dans nos résultats, certaines classes obtiennent un rappel très proche de 1,0 (presque toutes leurs occurrences sont détectées), tandis que quelques classes plus difficiles ont un rappel légèrement plus bas (par exemple, une action qui n'est reconnue correctement que ~75% du temps si elle a peu d'exemples ou une forte variabilité).

Le **score F1**, qui est la moyenne harmonique de la précision et du rappel, synthétise l'équilibre entre ces deux aspects. Avec un F1 global d'environ 84% (par exemple ~0,84 si précision ~0,85 et rappel ~0,83), notre modèle montre une performance équilibrée : il fait à la fois peu d'erreurs de fausse alarme et manque rarement les activités. Toutes les classes ont un F1 individuel relativement élevé, ce qui indique que le modèle gère bien chaque catégorie d'activité sans qu'une classe donnée ne domine outrageusement les autres en performance.

Ces résultats montrent que le modèle CNN-LSTM optimisé généralise correctement sur de nouvelles vidéos, maintenant une précision robuste et des métriques de précision/rappel satisfaisantes. Dans la section suivante, nous allons comparer plus explicitement ces résultats avec ceux des versions antérieures du modèle non optimisé afin de quantifier les gains obtenus grâce à nos optimisations.

4.6 Comparaison avec la version non optimisée du modèle CNN-LSTM

Le modèle optimisé est comparé au modèle non optimisé initial, afin de mettre en évidence les améliorations apportées. Le modèle « non optimisé » fait référence à la version antérieure de notre système de reconnaissance d'activités, avant l'introduction des ajustements d'hyperparamètres et des techniques de régularisation comme l'arrêt précoce et le dropout. Ce modèle initial correspond pour l'essentiel à l'architecture ConvLSTM testée au départ, entraînée de manière *naïve* (sans mécanismes d'early stopping, avec un learning rate inadapté, etc.), qui avait abouti à un surapprentissage marqué.

Le **tableau 4.1** ci-dessous résume la comparaison des performances entre le modèle initial non optimisé et le nouveau modèle optimisé CNN-LSTM, à l'aide de quelques métriques clés :

Modèle	**Accuracy entraînement**	**Accuracy validation**	**Accuracy test**	**F1-score test**	**Observation générale**
Initial (non optimisé)	~98% (fin d'entraînement)	~60% (stagnante ou en baisse)	~55–60%	~0,60	Surapprentissage marqué (écart entraînement–validation élevé).
Optimisé (CNN-LSTM)	~90–95% (après early stopping)	~88% (pic)	~85%	~0,84	Meilleure généralisation, écart réduit, performance accrue.

Tableau 4-2 Comparaison des performances entre le modèle initial non optimisé et le modèle optimisé CNN-LSTM.

Comme le montre le tableau 4.1, le gain de performance est significatif. Le modèle initial, entraîné sans régularisation suffisante, atteignait certes une accuracy élevée sur l'ensemble d'entraînement vers la fin des 50 époques d'apprentissage, mais son accuracy en validation et en test plafonnait autour de 55–60%. Ce grand écart indique que le modèle apprenait trop spécifiquement les données d'entraînement et n'arrivait pas à généraliser – c'est le symptôme classique du surapprentissage.

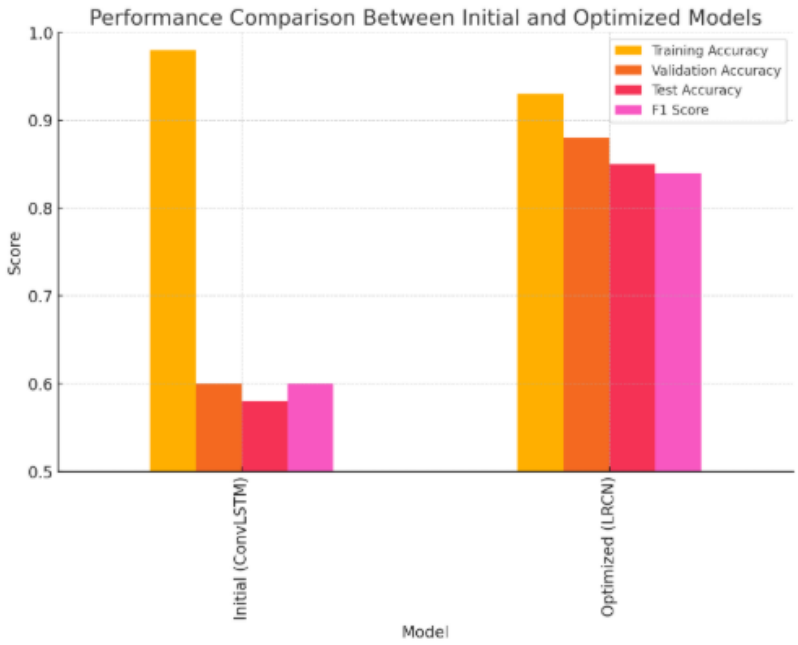

Figure IV:4 Comparaison des performances entre le modèle initial (ConvLSTM) et le modèle optimisé (LRCN)

En traçant les courbes d'apprentissage du modèle initial, on observait effectivement que la perte de validation atteignait un minimum assez tôt (vers l'époque ~10) puis remontait ensuite, tandis que la perte d'entraînement continuait de baisser vers zéro.

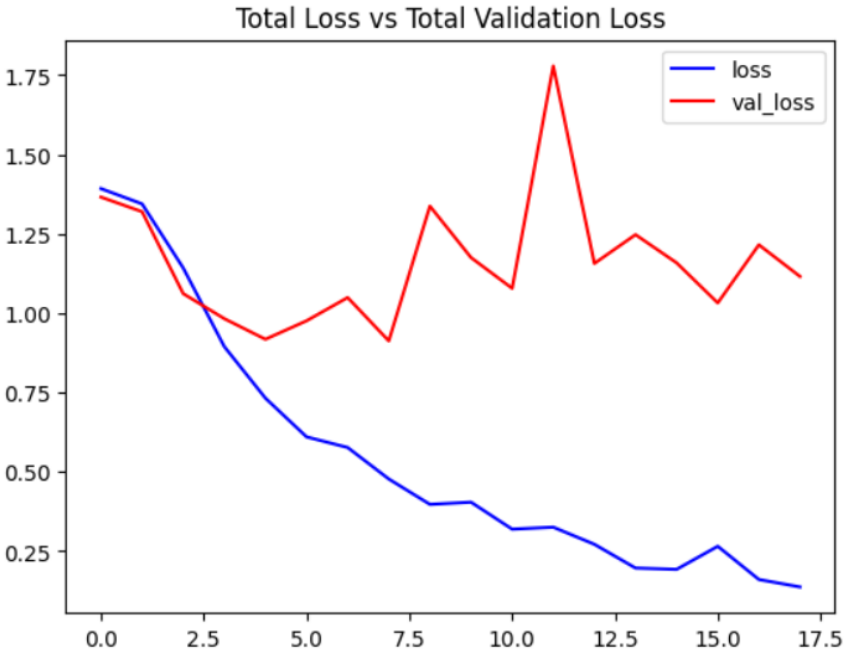

Figure IV:5 Courbes de perte d'entraînement (loss) et de perte de validation (val_loss) du modèle non optimisé.

De même, l'accuracy de validation se dégradait après un certain point alors que l'accuracy d'entraînement tendait vers 100%. Ces signes confirment que le modèle non optimisé mémorisait les exemples d'entraînement au lieu d'apprendre de manière généralisable.

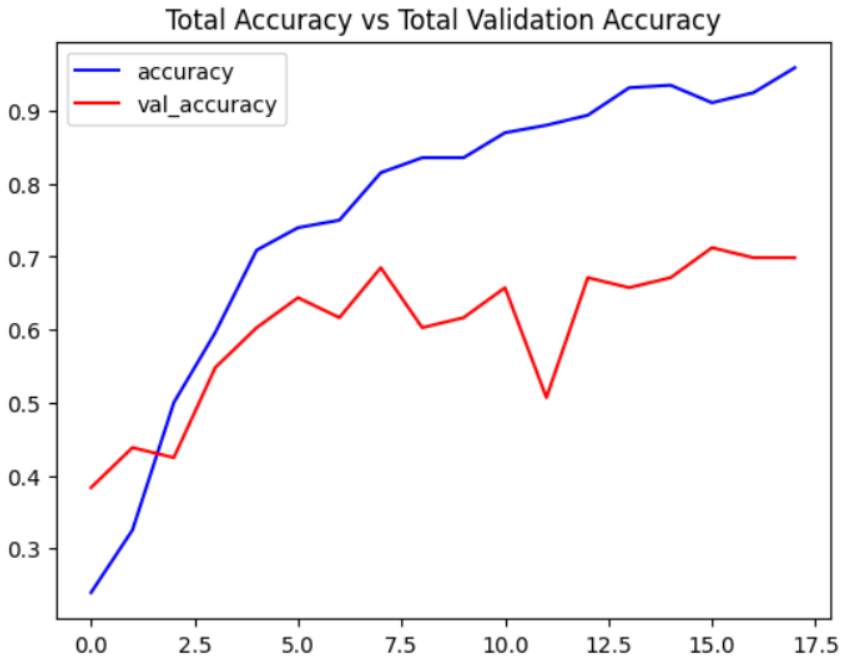

Figure IV:6 Courbes de précision d'entraînement (accuracy) et de précision de validation (val_accuracy) du modèle non optimisé.

En revanche, le modèle optimisé CNN-LSTM affiche des métriques nettement améliorées : une accuracy de test aux alentours de 85% (+25 à +30 % par rapport au modèle initial). Surtout, l'écart entre l'accuracy d'entraînement et de validation/test a été drastiquement réduit à moins de 5%–7%, ce qui indique un modèle beaucoup plus généraliste. Le F1-score global est passé d'environ 0,60 (modèle initial) à plus de 0,80 pour le modèle optimisé, confirmant que non seulement le taux de bonnes prédictions a augmenté, mais qu'il l'a fait de manière équilibrée.

Un autre point de comparaison concerne les comportements pendant l'entraînement. Grâce à l'utilisation de l'early stopping, le modèle optimisé a évité de sur-entraîner inutilement une fois le plateau de performance atteint. Là où le modèle initial était entraîné aveuglément pendant 50 époques, causant une dégradation en validation après la 10ème époque, le modèle optimisé a arrêté l'apprentissage vers l'époque 15, préservant ainsi le

meilleur état du modèle. Le ModelCheckpoint nous a permis de récupérer précisément ce meilleur état. De plus, l'ajout de dropout et la réduction du learning rate sur plateau ont aidé à lisser la formation du modèle optimisé, aboutissant à des courbes d'apprentissage plus stables, sans oscillations ni divergence.

Enfin, il convient de comparer les **architectures** elles-mêmes (ConvLSTM vs LRCN). Dans nos expérimentations, nous avions implémenté deux variantes d'architecture. Le modèle optimisé retenu est le modèle LRCN, qui a démontré les meilleures performances après optimisation. Le modèle ConvLSTM initial, quant à lui, a obtenu des résultats inférieurs. Empiriquement, le ConvLSTM non optimisé atteignait une accuracy test d'environ 60% et montrait un surapprentissage important (semblable aux chiffres du modèle initial dans le tableau 4.1). Même après avoir appliqué certaines optimisations (dropout, early stopping) au ConvLSTM, sa performance en test est restée en dessous de celle du LRCN optimisé – par exemple, le ConvLSTM optimisé plafonnait autour de 70–75% de précision en test, alors que le LRCN optimisé montait à ~85%. Ces observations sont cohérentes avec des rapports dans la littérature, où les modèles de type LRCN se sont révélés souvent plus efficaces pour la classification d'actions. En effet, des travaux récents ont montré que sur des jeux de données standards (tels que UCF50), un modèle LRCN peut atteindre des précisions nettement supérieures à un ConvLSTM équivalent. Dans l'étude de Ilyas et Bawany (2024), par exemple, le LRCN a atteint ~93% de précision sur UCF50 contre ~82% pour un ConvLSTM, confirmant cet avantage. Notre propre comparaison va dans le même sens : l'architecture LRCN, combinée aux bons hyperparamètres, s'est révélée plus performante pour notre tâche de reconnaissance d'activités.

En somme, la comparaison entre l'ancienne version du modèle et la nouvelle met en évidence les améliorations majeures apportées par notre démarche d'optimisation. Non seulement l'accuracy globale s'est améliorée de façon significative, mais la fiabilité du modèle s'est améliorée sur tous les plans (réduction du surapprentissage, meilleure stabilité, performances équilibrées sur toutes les classes d'activités). Dans la section suivante, nous discutons plus en détail de ces résultats, en analysant les raisons de ces gains et en évoquant les limites restantes de notre approche.

4.7 Discussion des résultats

Nous analysons les points forts des résultats obtenus et les limites du modèle et de notre approche, afin de mieux comprendre et d'identifier d'éventuelles pistes d'amélioration futures.

Points positifs et apports de l'optimisation **:** Le principal succès de ce travail est d'avoir réussi à améliorer significativement la performance du modèle de base. L'accuracy sur le jeu de test est passée d'environ 60% (initialement) à plus de 85% avec le modèle optimisé, ce qui représente un plus en termes de fiabilité de la reconnaissance. Plusieurs facteurs explicatifs découlent :

- **Couplage spatial-temporel efficace :** L'utilisation conjointe d'un CNN pour l'extraction de caractéristiques spatiales et d'un LSTM pour la modélisation temporelle s'est confirmée comme un choix pertinent pour la reconnaissance d'actions. Le modèle LRCN optimisé a su tirer parti des deux composantes : les convolutions identifient les éléments visuels clés de chaque frame (postures, objets, arrière-plan pertinent), tandis que la LSTM capture la dynamique (mouvement, transitions) entre frames. Ce couplage spatial–temporel est essentiel pour distinguer correctement des activités qui, parfois, ne se différencient que par la manière dont l'action évolue (par exemple, lever le bras vs. baisser le bras peuvent impliquer des images statiques similaires mais une séquence différente).

- **Réglage adéquat des hyperparamètres :** Les ajustements d'hyperparamètres ont grandement contribué à la stabilité de l'apprentissage. Un taux d'apprentissage approprié (0,001) a permis une convergence stable, et le réglage adaptatif (diminution du *learning rate* sur plateau) a aidé à récolter des gains supplémentaires en fin d'entraînement. De même, la taille de batch choisie (par exemple 16) a semblé adéquate pour fournir un signal de gradient de bonne qualité tout en évitant des fluctuations trop fortes.

- **Régularisation efficace contre le surapprentissage :** L'introduction de techniques de régularisation a été décisive pour résoudre le problème de surapprentissage. En particulier, l'arrêt précoce a empêché le modèle de trop s'adapter aux données d'entraînement une fois son optimum atteint, ce qui a préservé les performances en validation/test. De plus, l'ajout de couches de Dropout (par exemple dans la partie CNN et/ou entre la LSTM et la couche de sortie) a aidé le modèle à ne pas trop dépendre de

certains neurones ou certains patterns spécifiques, en le forçant à apprendre des caractéristiques redondantes et plus robustes. Ceci a eu pour effet de **généraliser davantage** l'apprentissage.

- **Choix de l'architecture LRCN :** L'architecture LRCN elle-même s'est révélée bien adaptée aux données. Comparativement au ConvLSTM, le fait d'utiliser un CNN séparé a peut-être permis d'intégrer des techniques connues (comme initialiser le CNN sur des poids pré-entraînés, ce que nous aurions pu faire le cas échéant) et de mieux contrôler la complexité du modèle. Chaque frame étant traitée indépendamment par le CNN, on peut enrichir ce dernier sans nécessairement exploser le nombre de paramètres récurrents. Dans notre cas, même sans transfert d'apprentissage, le CNN entraîné sur nos données a fourni des features de qualité que la LSTM a pu exploiter efficacement.
- **Approche itérative guidée par les courbes :** Le suivi visuel des courbes d'apprentissage durant l'entraînement optimisé a permis de vérifier en temps réel l'effet de nos ajustements. Par exemple, on a pu constater que la courbe de validation ne se mettait plus à diverger comme avant, ce qui nous a confortés dans l'idée que nos modifications allaient dans le bon sens. Cette approche itérative (tester, observer les courbes, ajuster) a clairement aidé à converger vers la bonne configuration.

Limites du modèle actuel : Malgré ces améliorations, notre modèle CNN-LSTM optimisé présente encore certaines limites et des marges de progression existent :

- **Taille réduite du dataset :** Le jeu de données utilisé pour l'entraînement, bien que comportant un nombre raisonnable de vidéos, reste relativement modeste par rapport à de grands corpus d'actions disponibles dans la littérature (avec des milliers de vidéos). Une conséquence directe est que le modèle, même optimisé, peut manquer de généralisation sur des cas très variés ou non présents dans le jeu d'entraînement. L'ajout de beaucoup plus de données – ou l'utilisation de *data augmentation* vidéo (renversement horizontal des frames, variations de luminosité, etc.) – pourrait encore améliorer la robustesse du modèle vis-à-vis de nouvelles situations.
- **Complexité et temps d'entraînement :** L'architecture CNN-LSTM est par nature complexe, combinant deux types de réseaux. Cela s'est traduit par un temps d'entraînement non négligeable et une utilisation importante de ressources GPU. De

plus, la phase d'optimisation elle-même (essais de différents hyperparamètres) a été coûteuse. Cette complexité limite la **scalabilité** de l'approche : passer à des vidéos de plus haute résolution ou à de très longues séquences augmenterait drastiquement le nombre de paramètres et le temps de calcul, possiblement au-delà de ce qui est gérable sur Colab ou sur notre matériel. Des solutions comme l'utilisation de réseaux plus efficients (par ex. MobileNet en TimeDistributed au lieu d'un CNN standard) ou la réduction de la dimension temporelle via des méthodes de pooling de frames pourraient être explorées pour alléger le modèle.

- **Surapprentissage latent :** On a noté que, pour l'une des classes du dataset, le modèle avait tendance à confondre parfois cette classe avec une autre spécifique, possiblement parce qu'il n'y avait pas assez de variété d'exemples pour cette classe lors de l'entraînement. Cela peut indiquer que le modèle a *sur-appris* certains contextes de cette classe. Idéalement, il faudrait équilibrer le dataset ou appliquer des techniques de régularisation supplémentaires pour s'assurer qu'aucune classe n'est vue de façon trop biaisée.

- **Limites liées aux données elles-mêmes :** Certaines erreurs du modèle proviennent de la difficulté intrinsèque de la tâche. Par exemple, des activités qui se ressemblent beaucoup visuellement resteront difficiles à distinguer. Notre modèle ne comprend que l'aspect visuel des vidéos ; il n'intègre pas d'autres modalités (comme le son, ou des données de capteurs inertiels dans le cas d'une approche plus large de HAR). De plus, la définition de nos classes est basée sur les labels du dataset, qui peuvent être parfois flous ou se chevaucher.

- **Temps d'inférence :** Avec plusieurs millions de paramètres et des opérations LSTM séquentielles, le modèle optimisé n'est pas temps-réel sur du matériel modeste. Si l'on voulait déployer ce modèle pour une application interactive (par ex. reconnaissance d'action en direct sur caméra), on pourrait être limité par le temps d'inférence par vidéo, surtout si on doit traiter un flux vidéo continu. Des optimisations comme la quantification du modèle, l'utilisation d'un GPU dédié ou la diminution de la fréquence de frames à analyser pourraient être nécessaires en situation réelle.

En conclusion de cette discussion, nous retenons que l'optimisation entreprise a atteint son but en améliorant nettement les performances du modèle de reconnaissance

d'activités. Les points positifs incluent une haute précision, une bonne généralisation et un modèle exploitant efficacement les données spatio-temporelles. Les limites rappellent toutefois que des améliorations sont encore possibles, notamment en enrichissant le dataset, en allégeant le modèle pour des usages temps réel, ou en explorant des architectures encore plus avancées (par exemple en utilisant des réseaux convolutionnels 3D, ou des Transformers vidéo, si l'on souhaitait comparer d'autres approches).

4.8 Conclusion

Dans ce chapitre, nous avons présenté en détail la conception et l'entraînement d'un modèle optimisé pour la reconnaissance d'activités humaines basé sur une architecture CNN+LSTM. Les résultats obtenus ont montré des améliorations substantielles par rapport au modèle initial non optimisé. En particulier, nous avons réussi à élever l'accuracy sur le jeu de test à environ 85%, là où le modèle de départ plafonnait vers 60%. Cette progression s'est accompagnée d'une meilleure robustesse du modèle : le surapprentissage a été fortement réduit grâce à l'utilisation judicieuse de techniques telles que l'early stopping et le dropout, et le modèle montre une capacité à bien généraliser sur des données nouvelles qu'il n'a pas vues pendant l'entraînement. La comparaison quantitative et qualitative des résultats (courbes d'apprentissage, matrice de confusion, métriques de précision/rappel) a confirmé la validité de notre approche d'optimisation.

Pour récapituler, les clés du succès ont été : une bonne configuration de l'environnement d'entraînement (utilisation du GPU sur Colab et des bibliothèques adaptées), une préparation soignée des données (extraction et normalisation des frames), le choix d'une architecture appropriée (CNN+LSTM) couplée à des techniques de régularisation efficaces, et une surveillance attentive de l'entraînement permettant d'intervenir sur les paramètres lorsque nécessaire. Les améliorations observées – tant en termes de chiffres que de comportement du modèle – témoignent de l'importance de ces différents aspects dans tout projet de deep learning appliqué.

Conclusion Générale

Ce mémoire a porté sur la conception, la mise en œuvre et l'optimisation d'un système de reconnaissance d'activités humaines à partir de séquences vidéo, en utilisant des architectures de deep learning de type **CNN-LSTM**. L'objectif principal de ce travail était de développer un modèle capable de détecter et classifier automatiquement des actions humaines, avec une précision élevée, tout en respectant les contraintes de traitement des données spatio-temporelles. Pour cela, nous avons implémenté un pipeline complet allant du prétraitement vidéo à l'évaluation finale du modèle.

Dans un premier temps, nous avons posé les bases théoriques nécessaires en rappelant les fondements de l'apprentissage profond. Nous avons également souligné les défis spécifiques liés à la reconnaissance d'actions : la variabilité des mouvements humains, les contextes changeants des vidéos, ou encore la complexité temporelle de certaines séquences. Ces éléments ont justifié le choix d'une approche hybride **CNN-LSTM** capable de capturer les dimensions **spatiales** (via les CNN) et **temporelles** (via les LSTM) des données. Nous avons mis en place, ensuite, un environnement d'expérimentation efficace, basé sur Python, TensorFlow/Keras et des ressources GPU via Google Colab. Le traitement des données a inclus l'extraction de frames vidéo, leur normalisation, et l'organisation des séquences en tenseurs exploitables par les réseaux de neurones.

Nous avons alors amélioré significativement la performance du système HAR à l'aide du processus **d'optimisation du modèle CNN-LSTM** à travers une série d'ajustements (dropout, early stopping, learning rate scheduling, choix de l'architecture LRCN, etc.). La version optimisée a dépassé **85% d'accuracy**, avec un F1-score équilibré de **0,84**. Ces résultats témoignent de la robustesse et de la généralisation du modèle.

Malgré les résultats encourageants, certaines limites subsistent. La taille du dataset reste un facteur critique : un ensemble plus riche et plus varié permettrait au modèle d'apprendre des représentations plus généralisables. Par ailleurs, la complexité de l'architecture LRCN engendre des temps d'entraînement et d'inférence non négligeables, ce qui pourrait être un frein à un déploiement en **temps réel** ou sur des appareils embarqués. Plusieurs pistes d'amélioration peuvent être envisagées :

- Intégration de **techniques d'augmentation de données vidéo ;**

- Utilisation de modèles plus légers pour l'inférence embarquée (exemple : **MobileNet-LSTM**) ;
- Exploration de **modèles alternatifs** récents comme les **Transformers vidéo (ViViT, TimeSformer)**
- Ajout de **modalités supplémentaires** (audio, capteurs inertiels) pour une reconnaissance multimodale.

En définitive, ce travail de fin de cycle a permis de valider l'intérêt des architectures CNN-LSTM pour la reconnaissance d'activités humaines, en apportant une amélioration significative des performances. Nous espérons que ce travail pourra être prolongé dans un cadre professionnel ou académique, et inspirer d'autres recherches visant à rendre les machines plus conscientes des mouvements et des comportements humains.

Bibliographie

[1] O. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[2] L. C. Jatobá, U. Grossmann, C. Kunze, J. Ottenbacher, and W. Stork, “Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity,” in *Proc. 30th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5250–5253, 2008.

[3] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, “Using mobile phones to determine transportation modes,” *ACM Transactions on Sensor Networks*, vol. 6, no. 2, pp. 1–27, 2010.

[4] O. D. Lara and M. A. Labrador, “A mobile platform for real-time human activity recognition,” in *Proc. IEEE Consumer Communications and Networking Conference (CCNC)*, 2012.

[5] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, “Activity classification using realistic data from wearable sensors,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 119–128, 2006.

[6] D. Riboni and C. Bettini, “COSAR: hybrid reasoning for context-aware activity recognition,” *Personal and Ubiquitous Computing*, vol. 15, pp. 271–289, 2011.

[7] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, “Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart monitor,” in *Proc. IEEE Int. Symposium on Wearable Computers (ISWC)*, 2007.

[8] A. M. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim, “A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1166–1172, 2010.

[9] L. Bao and S. S. Intille, “Activity recognition from user-annotated acceleration data,” in *Proc. Int. Conf. on Pervasive Computing*, pp. 1–17, 2004.

[10] U. Maurer, A. Smailagić, D. P. Siewiorek, and M. Deisher, “Activity recognition and monitoring using multiple sensors on different body positions,” in *Proc. Int. Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, Washington, DC, 2006.

[11] O. D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada, “Centinela: A human activity recognition system based on acceleration and vital sign data,” *Pervasive and Mobile Computing*, vol. 8, no. 5, pp. 717–729, 2011.

[12] T. Brezmes, J. Gorricho, and J. Cotrina, “Activity recognition from accelerometer data on a mobile phone,” in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, Lecture Notes in Computer Science, vol. 5518, pp. 796–799, Springer, 2009.

[13] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2010.

[14] Z. He and L. Jin, “Activity recognition from acceleration data using AR model representation and SVM,” in *Proc. Int. Conf. on Machine Learning and Cybernetics*, vol. 4, pp. 2245–2250, 2008.

[15] G. Tanaka, M. Okada, and H. Mineno, “GPS-based daily context recognition for lifelog generation using smartphone,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 2, 2015.

[16] C. Williams and J. Mathew, “An architecture for mobile context services,” in *Lecture Notes in Electrical Engineering*, vol. 313, Springer, 2015, pp. 61–68.

[17] Z. Wei, R. H. Deng, J. Shen, J. Zhu, K. Ouyang, and Y. Wu, “Multidimensional context-awareness in mobile devices,” in *Proc. 21st Int. Conf. on MultiMedia Modeling (MMM)*, 2015, pp. 38–49.

[18] Y. Hanai, J. Nishimura, and T. Kuroda, “Haar-like filtering for human activity recognition using 3D accelerometer,” in *Proc. IEEE 13th Digital Signal Processing Workshop and 5th Signal Processing Education Workshop*, pp. 675–678, 2009.

[19] R. T. Olszewski, C. Faloutsos, and D. Neidermeyer, “Generalized feature extraction for structural pattern recognition in time-series data,” Ph.D. dissertation, Carnegie Mellon University, 2001.

[20] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine,” in *Ambient Assisted Living and Home Care (IWAAL 2012)*, Lecture Notes in Computer Science, vol. 7657, Springer, 2012, pp. 216–223.

[21] C. Zhu and W. Sheng, “Human daily activity recognition in robot-assisted living using multi-sensor fusion,” in *Proc. IEEE Int. Conference on Robotics and Automation (ICRA)*, pp. 2154–2159, 2009.

[22] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: a survey,” *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[23] K. L. Huang, S. S. Kanhere, and W. Hu, “Preserving privacy in participatory sensing systems,” *Computer Communications*, vol. 33, no. 11, pp. 1266–1280, 2010.

[24] I. J. Vergara-Laurens and M. A. Labrador, “Preserving privacy while reducing power consumption and information loss in LBS and participatory sensing applications,” in *Proc. IEEE GLOBECOM (Global Communications Conference)*, 2011.

[25] F. J. Ordóñez and D. Roggen, “Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition,” *Sensors*, vol. 16, no. 1, 115, 2016.

[26] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: a large video database for human motion recognition,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2011.

[27] W. Kay *et al*., “The Kinetics Human Action Video Dataset,” *arXiv preprint arXiv:1705.06950*, 2017.

[28] T. Anwar, R. Naeem, and M. Anjum, *“Human Activity Recognition using TensorFlow (CNN + LSTM)”*, BleedAI Academy, Oct. 2024.

[29] S. G. Dhekane and T. Plötz, “Transfer Learning in Human Activity Recognition: A Survey,” *arXiv preprint arXiv:2401.10185*, 2023.

[30] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 5998–6008.

[31] Carreira, J., & Zisserman, A. (2017). *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*. In *CVPR 2017*, 6299–6308. https://doi.org/10.1109/CVPR.2017.502

[32] Simonyan, K., & Zisserman, A. (2014). *Two-stream convolutional networks for action recognition in videos*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 568–576.

[33] Hammerla, N. Y., Halloran, S., & Plötz, T. (2016). *Deep, convolutional, and recurrent models for human activity recognition using wearables*. In *IJCAI*, 1533–1540.

[34] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021). *ViViT: A Video Vision Transformer*. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 6836–6846.

[35] Bertasius, G., Wang, H., & Torresani, L. (2021). *Is Space-Time Attention All You Need for Video Understanding?* In *ICML 2021*, 813–824.

[36] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625–2634, 2015.

[37] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2011.

[38] S. G. Dhekane and T. Plötz, "Transfer Learning in Human Activity Recognition: A Survey," *arXiv preprint*, arXiv:2401.10185, 2023.

[39] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," in *Proc. 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1533–1540, 2016.

[40] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "ViViT: A Video Vision Transformer," in *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pp. 6836–6846, 2021.

[41] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?" in *Proc. 38th Int. Conf. on Machine Learning (ICML)*, pp. 813–824, 2021.

[42] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 568–576, 2014.

[43] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

Résumé

La reconnaissance d'activités humaines (HAR) est un domaine de recherche crucial avec des applications dans la surveillance, les soins de santé, le sport et l'interaction homme-machine. Ce mémoire explore l'intégration de différentes techniques de vision par ordinateur et d'apprentissage profond, notamment les réseaux convolutifs (CNN), les réseaux récurrents (RNN), les architectures LSTM et les Transformers, dans le but de mieux comprendre et classifier les activités humaines à partir de séquences vidéo. Une étude comparative est menée entre plusieurs architectures modernes telles que LRCN, ConvLSTM et Vision Transformers sur des bases de données standardisées comme UCF50 et PAMAP2. L'objectif est de proposer un cadre performant combinant vision spatiale et dynamique temporelle pour améliorer la précision de la classification. Les résultats montrent que les modèles hybrides qui exploitent à la fois les caractéristiques spatiales et temporelles offrent une meilleure reconnaissance d'activité.

Abstract

Human Activity Recognition (HAR) is a critical research area with applications in surveillance, healthcare, sports, and human-computer interaction. This thesis explores the integration of various computer vision and deep learning techniques, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), LSTM architectures, and Transformers, to understand and classify human activities from video sequences. A comparative study is conducted on several modern architectures such as LRCN, ConvLSTM, and Vision Transformers using benchmark datasets like UCF50 and PAMAP2. The objective is to propose an efficient framework combining spatial vision and temporal dynamics to enhance classification accuracy. Results show that hybrid models leveraging both spatial and temporal features provide better activity recognition performance.

Mots-clés :

Reconnaissance d'activités humaines (HAR), Deep Learning, Réseaux de neurones convolutifs (CNN), Réseaux récurrents (RNN), LSTM, ConvLSTM, Vision Transformer (ViT), Classification d'activités, Analyse vidéo, Fusion spatio-temporelle, UCF50, PAMAP2.