N° Ordre...../Faculté/UMBB/2021

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté des **H**ydrocarbures et de la **C**himie

Thèse de Doctorat

Présentée par
Khebli Abdelmalek
Filière : Génie Electrique

Option : Automatique Appliquée et Traitement du Signal

REPRESENTATIONS D'IMAGES POUR LA RECHERCHE ET LA CLASSIFICATION D'IMAGES

Devant le jury :

ACHELI	Dalila	Prof	UMBB	Président
GROUNI	Said	Prof	UMBB	Examinateur
BENAMMAR	Abdessalem	DDR	CRTI	Examinateur
GUIATNI	Mohammed	Prof	EMP	Examinateur
MEGLOULI	Hocine	Prof	UMBB	Encadreur
BENTABET	Layachi	Prof	U/Canada	Co- Encadreur
AGUIB	Salah	MC/A	UMBB	Invité

Année Universitaire : 2020/2021

DéDicaces

A mes êtres les plus chers,

en guise de gratitude,

pour leurs amour et sacrifices :

mes parents,

A Ma chère femme , pour la patience et le soutien dont elle a fait preuve pendant

toute la durée de cette thèse,

et mes deux adorables filles : Ratil et Tasnime

A mes frères et sœurs,

A Toute ma famille ainsi que la famille de ma femme à Jijel

et mes chers amis, Amer, Mohamed et Salah

je dédie cette thèse.

Remerciements

A l'issue de la rédaction de ce manuscrit, je suis convaincu que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail de Doctorat sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifesté à l'égard de ma recherche qui m'a permis de progresser dans cette phase délicate de « Doctorant Abdelmalek ».

En premier lieu, je tiens à remercier mon directeur de thèse, monsieur le professeur MEGLOULI Hocine, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail de Doctorat, pour ses multiples conseils et pour toutes les heures qu'il a consacrées à diriger cette recherche. J'aimerai également lui dire à quel point j'ai apprécié sa grande disponibilité et son respect sans faille de relecture des documents que je lui ai adressés. Enfin, j'ai été extrêmement sensible à ses qualités humaines d'écoute et de compréhension tout au long de ce travail doctoral.

Mes vifs remerciements vont à mon Co-Encadreur, le professeur Layachi BENTABET , pour son encadrement. Cette thèse n'aurait pas vu le jour sans son encouragement et sa patience .

Mes remerciements s'adressent également à la présidente du jury, Mme. le professeur ACHELI Dalila et aux membres de ce jury, Mr. GROUNI Said, Mr. BENAMMAR Abdessalem, Mr. GUIATNI Mohammed et Mr.AGUIB Salah qui ont accepté d'évaluer cette thèse de doctorat.

Je tiens également à remercier le Dr. CHIKH Noureddine et notre collègue Mr. DJEDID Toufik, pour leur participation à l'amélioration de la qualité de ce manuscrit, tant dans la forme que dans le fond.

Un grand merci à ma femme Sara qui a partagé toutes mes difficultés au cours de la rédaction de ce travail par ses soutiens et conseils d'encouragements les plus fructueux. Un grand merci !

Je remercie finalement toutes les personnes qui m'ont aidé de près ou de loin, à la réalisation de ce travail.

Résumé

La reconnaissance d'actions humaines dans des séquences vidéo est un problème important, actuellement au cœur de nombreux domaines de recherche comme la vidéo surveillance et les interfaces Homme-Machine. Le développement d'algorithmes de reconnaissance d'actions précis et efficaces est une tâche difficile à cause des fortes variabilités des formes humaines, des vêtements et du mouvement. Ce travail vise la reconnaissance de l'action humaine dans la vidéo. Les actions humaines dans la vidéo sont des signaux spatio-temporels tridimensionnels (3D) qui caractérisent à la fois l'apparence visuelle et la dynamique de mouvement des humains. Dans ce travail, nous nous intéresserons principalement à l'information dynamique. Nous utilisons des réseaux de neurones convolutionnels profonds 3D qui prennent des cartes de flot optiques en entrée. Le processus de traitement de notre approche se compose principalement de trois parties :

- Calcul des cartes de flot optique;
- Filtrage de ces cartes, pour augmenter le taux de classification et de réduire le temps du calcul, qui ne contiennent pas de l'action en utilisant l'entropie pour chaque séquence ;
- Les réseaux de neurones convolutionnels profonds 3D pour l'extraction de caractéristiques et la classification des actions humaines. Notre méthode est évaluée sur une base de données publique InfAR. En comparaison avec les méthodes existantes dans l'état de l'art, les résultats obtenus montrent l'efficacité de l'approche proposée avec un taux supérieur à 88% sur la même base de données.

Mots clés : Reconnaissance d'actions humaines ; flot optique; Les réseaux de neurones convolutionnels profonds 3D .

Abstract

Human action recognition in 3D video is a major problem in research areas such as video surveillance, human-machine interfaces, video indexing and video search. Human action is defined as a periodic activity performed by one person in a time interval. Different types of actions, such as walking, jumping, running, etc., can be identified in a video sequence. Considerable progress has been made in recent years in human action recognition. However, it remains a difficult challenge due to the strong variation of human actions.

The aim of this work is to recognize human action in infrared video by focusing mainly on dynamic information. We developed a new technique based on deep 3D convolutional neural networks (3D CNNs) that take optical flow maps as input. Our approach consists mainly of three parts :

- Computation of optical flow maps;
- Filtering of these maps, using an entropy measurement in order to increase the classification rate and reduce the run time by eliminating sequences that do not contain human action;
- Classification using 3D CNN. The experimental results obtained by our approach on the InfAR dataset show considerable improvement in comparison with results obtained by existing models.

Keywords : Recognition of human actions; optical flow; deep 3D convolutional neural networks.

ملخص

يعتبر التعرف على أفعال وحركة الإنسان من خلال الفيديو مشكلة بحد داتها. حيث توجد تطبقاتها حاليًا في العديد من مجالات البحث العلمي مثل المراقبة والتعرف على وجه الإنسان مباشرة من شريط الفيديو عن طريق الحاسوب و يعد تطوير خوارزميات التعرف على الإجراءات الدقيقة والفعالة في شريط الفيديو مهمة صعبة بسبب التباين الكبير في الأشكال البشرية والملابس والحركة. يهدف هذا العمل بشكل عام إلى التعرف على عمل و حركة الإنسان في الفيديو. الأفعال البشرية في الفيديو هي عبارة عن إشارات زمنية و مكانية ثلاثية الأبعاد.

في هذا العمل ، ركزنا بشكل خاص على العمل الزمني لمعرفة حركة الإنسان من خلال شريط الفيديو حيث استخدمنا شبكات عصبية ثلاثية الأبعاد بحيث تأخذ خرائط التدفق البصري كا مدخلات لها. كما ترتكز عملية المعالجة على الأجزاء التالية:

- حساب خرائط التدفق البصري
- تصفية تلك الخرائط ، لزيادة معدل التصنيف وتقليل وقت الحساب ، التي لا تحتوي على الحركات الإنسانية في شريط الفيديو وهذا باحتساب الانتروبي لكل تسلسل .
- استخدام شبكات عصبية عميقة ثلاثية الأبعاد لاستخراج ميزات وتصنيف الأفعال البشرية. كما قمن بتجربة هده الطريقة على قاعدة البياناتInfAR و بالمقارنة مع الأساليب والطرق الموجودة حاليا، أظهرت النتائج التي تم الحصول عليها فعالية الطريقة المقترحة بمعدل تصنيف يفوق 88٪ على نفس قاعدة البيانات InfAR.

كلمات مفتاحيه: خرائط التدفق البصري, شبكات عصبية, تصنيف الأفعال البشرية.

Acronymes

ASLP	Analyse Sémantique Latente Probabiliste
ART	Théorie de Résonance Adaptative, Adaptative Resonnance Theorie
CEC	Constant Error Carousel
CNN	Réseau de Neurones Convolutifs, Convolutionnal Neural Network
CNN3D	*Réseau de Neurones Convolutif 3D*, 3D Convolutional Neural Networks
eSURF	SURF étendu
GCA-LSTM	Global Context-Aware – LSTM
GPU	Processeur Graphique , Graphics Processing Unit
HOF	Histogramme de Flux Optique , Histogram of Optical Flow
HOG	Histogramme d'Orientations du Gradient, Histogram of Oriented Gradients
InfAR	Infrared Action Recognition
KNN	*k* plus proches voisins, k-Nearest Neighbors
KSC	Kinematic Spline Curves
LSTM	Longue Mémoire à Court Terme, Long Short-Term Memory
MAP	Probabilité de Maximum A posteriori, Maximum A posteriori Probability
MEI	Image d'Energie du Mouvement, Motion Energy Image
MHI	Image d'Historique du Mouvement, Motion History Image
MHV	Volume d'Historique du Mouvement, Motion History Volume
PCA	Analyse en Composantes Principales, Principal Component Analysis
PMC	Perceptron Multi Couches
RBF	Fonctions de Base Radiale, Radial Basis Function
ReLU	Unité Linéaire Rectifiée, *Rectified Linear Unit*
RNA	Réseau de Neurones Artificiels
SIFT	Scale Invariant Feature Transform
SOM	carte auto-organisatrices, *Self Organizing Map*
SURF	Speeded Up Robust Features
SVM	Machines à Vecteurs de Support, Support Vector Machine

Table des matières

Chapitre 03 : Apprentissage automatique : les réseaux de neurones

Chapitre 04 : Les réseaux de neurones convolutifs

Chapitre 05: Résultats et discussions

Table des figures

Liste des tableaux

INTRODUCTION GÉNÉRALE

La reconnaissance des actions est un sujet particulièrement complexe dans le domaine de la vision par ordinateur. Cela consiste en la classification automatique des actions ou des activités réalisées par un individu dans une séquence vidéo. L'objectif d'un système de reconnaissance d'actions est de reconnaitre les actions simples de la vie courante dans une vidéo (ex : marcher, répondre au téléphone, sauter, etc.) à partir de vidéos de référence. Ces actions répondent à des modèles de mouvements simples effectués par une seule et même personne durant un laps de temps court.

De nos jours, l'information extraite dans le domaine de la vision par ordinateur provient de diverses sources visuelles et sensorielles telles que la parole, la vidéo, une image. Les données vidéo permettent d'enregistrer les différents mouvements réalisés par l'être humain à l'aide d'un capteur d'acquisition, tel que le capteur infrarouge.

Les systèmes de vision exigent deux processus fondamentaux, le processus de bas niveau qui utilise des techniques ou des algorithmes de prétraitements, permettant l'extraction d'un ensemble de caractéristiques pertinentes comme la couleur, les différentes régions homogènes existantes dans l'image, la texture, le mouvement,... etc. dont le but est d'obtenir une représentation plus concise et facilement analysable résumant le contenu de la donnée vidéo et fournissant une quantité d'informations sur la scène. Dans une deuxième étape, l'exploitation de ces attributs par un processus de haut niveau plus complexe permet d'analyser, et de reconnaitre l'action qui se déroule dans la séquence vidéo.

La reconnaissance d'actions peut intervenir dans différents domaines, à savoir :

- **La vidéo-surveillance automatique :** Les systèmes de vidéo-surveillance automatique aident à assurer au mieux la sécurité de personnes, de sites sensibles ou de lieux publics et privés. Elle permet d'analyser automatiquement des scènes et de détecter des comportements suspects. Les systèmes de reconnaissance d'activités permettent dans ce contexte la détection d'activités inhabituelles telles que l'agression ou l'intrusion;
- **L'indexation et la recherche des vidéos :** Avec la multiplication des sites de partage de vidéos, il est devenu nécessaire de développer des outils d'indexation et de stockage fiables et efficaces afin d'améliorer l'expérience de l'utilisateur. Cela nécessite l'apprentissage de modèles à partir de vidéos brutes et résumer celles-ci selon leur contenu. Cette pratique a gagné un regain d'intérêt avec les progrès des applications de recherche d'image par le contenu;
- **Applications et environnements interactifs :** Comprendre l'interaction entre un ordinateur et un humain reste l'un des grands défis dans la conception d'interfaces homme-machine. Les repères visuels sont le mode le plus important de la communication non verbale. L'utilisation adéquate de ce mode peut amener à la création d'ordinateurs interagissant de manière efficace avec leur utilisateur. De même, les environnements interactifs tels que les maisons intelligentes réagissant aux gestes de l'utilisateur peuvent bénéficier de méthodes basées sur la vision par ordinateur;
- **Applications - santé :** La reconnaissance des activités joue aussi un rôle important dans les applications de suivi médical et d'assistance aux personnes âgées que ce soit dans les habitats intelligents pour la santé, les hôpitaux ou dans les résidences pour personnes âgées. La reconnaissance des activités permet par exemple de détecter les évènements pouvant être dangereux pour la personne âgée et prévenir toute situation à risque. Pour les applications de santé, la reconnaissance automatique des activités permet d'améliorer la sécurité des personnes âgées souhaitant vivre le plus longtemps possible dans leurs domiciles. La reconnaissance des activités permet dans ce contexte de détecter les cas d'urgence tels que la chute et la non prise de médicaments. De plus, elle constitue une étape primordiale dans les systèmes de détection de perte d'autonomie de la personne âgée [2,115];

- **Animation et synthèse d'images :** L'industrie de l'animation et du jeu vidéo repose sur la synthèse réaliste de l'humain et de ses mouvements. La synthèse de mouvement trouve une large application dans l'industrie du jeu où l'exigence est de produire une grande variété de mouvements avec quelques compromis sur la qualité. L'industrie du film d'autre part repose traditionnellement davantage sur des animateurs humains pour fournir des animations de haute qualité en terme de réalisme. Toutefois, ces tendances tendent à changer. Grâce à l'amélioration des algorithmes et du matériel, la synthèse de mouvements beaucoup plus réalistes est maintenant possible à partir de l'apprentissage. Une application possible est l'apprentissage dans des environnements de simulation.

Problématiques posées

Bien que le contenu vidéo soit assez informatif, la tâche de reconnaissance de l'action humaine reste problématique. Ceci est dû, d'une part, à l'ambiguïté interclasses puisqu'il existe des actions similaires telles que les actions "courir lentement" (jogging) et "marcher" qui ne diffèrent que par la vitesse de réalisation. Ceci rend difficile la classification et l'interprétation de l'action au cours du temps. Une autre difficulté vient au grand degré d'occlusion entre la personne qui fait l'action et une autre personne qui passe devant lui. De plus, il faut prendre en compte la grande variabilité intra-classe à laquelle une classe d'action peut s'exposer. En effet, de grandes variations de style peuvent être observées dans la reproduction de la même action selon les habitudes, le genre, la taille, l'ethnicité des personnes et selon aussi le contexte dans lequel l'action se déroule. Par exemple, l'action "marcher" varie selon la vitesse et la taille de la personne. Ainsi un système de reconnaissance d'actions fiable doit avoir une grande capacité de généralisation pour prendre en compte les variations intra-classes tout en étant capable de discriminer les actions ayant une petite variation inter-classes. D'autres difficultés additionnelles sont liées aux conditions d'enregistrement des séquences vidéo. Le changement des conditions d'illumination, le fond dynamique et la distance de la caméra peuvent influencer l'apparence des personnes effectuant l'action. De plus, l'apparence visuelle de l'action peut changer selon l'angle de vue d'enregistrement.

Contributions

Le but de cette thèse est de proposer une méthode de classification de l'action humaine à partir des séquences vidéo. Les contributions de cette thèse peuvent être résumées dans les points suivants :

1- Nous proposons un nouvel algorithme basé sur le calcul d'entropie pour filtrer les cartes de flot optique qui ne contiennent pas de l'action. Cet algorithme aide le réseau de neurone convolutionnel profond 3D proposé pour améliorer le taux de la classification et aussi de minimiser le temps du calcul;
2- Nous évaluons l'influence d'utiliser cet algorithme de filtrage sur les données d'apprentissage et de test de CNN3D, il montre que l'utilisation de cet algorithme améliore la précision de classification;
3- Nous appliquons notre approche à l'ensemble de données d'InfAR et signalons une amélioration significative par rapport aux résultats de référence sur la même base de données.

Plan de la thèse

Ce manuscrit est organisé en 5 chapitres :

Le chapitre 1 : Etat de l'art, présente les travaux de l'état de l'art concernant les principaux travaux portant sur les techniques d'extraction des caractéristiques utilisées en reconnaissance d'actions à partir des séquences vidéo;

Le chapitre 2 : Modélisation des actions par le flot optique, présente la technique de flot optique que nous avons utilisée dans notre travail pour exploiter l'aspect temporel de la séquence vidéo dans le système de classification et de suivre les points caractéristiques dans chaque trame;

Le chapitre 3 : Apprentissage automatique : les réseaux de neurones

Ce chapitre présente une technique d'apprentissage fondée au départ sur une analogie avec la physiologie de la transmission de l'information et de l'apprentissage dans les systèmes cérébraux : les modèles connexionnistes (on dit aussi les réseaux de neurones artificiels);

Le chapitre 4 : Réseaux de neurone convolutionnel 3D, présentera le modèle de classification de séquences vidéo parmi les plus populaires de l'état de l'art. Nous allons rappeler quelques définitions et fondements théoriques de ce modèle;

Le chapitre 5 : Résultats et discussions, détaillera la base de données étudiée et notre algorithme de filtrage, ainsi que les résultats expérimentaux trouvés et on termine par une conclusion générale et quelques perspectives.

CHAPITRE 01

ETAT DE L'ART

1.1 Introduction

La dernière décennie a été témoin d'un accroissement rapide de caméras vidéo de différents types, allant de la plus simple à la plus sophistiquée. Cela a donné lieu à une explosion de contenus vidéo. Plusieurs applications telles que la recherche et la classification de vidéos basées sur le contenu, l'extraction d'informations exigent la reconnaissance des activités se produisant dans celles-ci. Parmi ces activités, celle sur laquelle a porté notre travail est l'activité humaine dans les vidéos. L'analyse des activités humaines est un domaine dont les résultats sont de plus en plus considérables dans des secteurs aussi divers que la surveillance, la sécurité, la santé, …etc.

Dans ce chapitre, nous décrivons quelques techniques parmi les plus populaires utilisées en extraction des caractéristiques. Les approches existantes peuvent être divisées en trois groupes : les approches basées sur l'extraction des caractéristiques locales, les approches basées apparence et les approches d'extraction automatique basées sur l'apprentissage profond.

1.2 Approches basées sur des caractéristiques locales

L'extraction des caractéristiques locales est une étape de pré-traitement nécessaire à la classification des actions humaines dans la vidéo. Le principal avantage de ces approches est qu'aucune information sur le modèle du corps humain où la localisation des personnes n'est nécessaire. Les approches locales décrivent les observations comme une série de descripteurs locaux ou des ensembles de patchs indépendants. Les

représentations (primitives) locales sont insensibles au bruit, invariantes aux changements d'angle et elles ne nécessitent pas une soustraction de l'arrière plan. Ces approches peuvent être classées en deux grandes catégories que nous présentons dans ce qui suit.

1.2.1 Détecteurs de points d'intérêt spatio-temporels

Les systèmes de classification vidéo reposent généralement sur l'étape d'extraction des caractéristiques qui consistent à représenter le contenu à classer par un vecteur de description. Parmi les caractéristiques qui peuvent être extraites des séquences vidéo, les points d'intérêt spatio-temporels. Différentes méthodes ont été proposées dans l'état de l'art pour extraire les points d'intérêt spatiaux. Dans [76], Laptev et Lindeberg utilisent le détecteur de Harris et Stephens donné dans [51] en lui rajoutant la variable temporelle. Les primitives extraites sont les points dont le voisinage local est soumis à un changement significatif dans les plans spatiaux et temporels. Ce travail a été amélioré dans [75] pour compenser les mouvements relatifs de la caméra.

Dans [100], les auteurs ont utilisé la notion d'entropie dans le détecteur de région saillante 2D proposé dans [65]. L'inconvénient que comporte cette méthode est que le nombre de points d'intérêt stable est relativement faible par rapport aux zones contenant des mouvements significatifs. Afin d'y remédier à ce problème, une solution a été proposée dans l'état de l'art par Dollar *et al* [28], qui appliquent un filtre de Gabor sur les dimensions spatiales et temporelles individuelles, cette approche a été améliorée plus tard par Bregonzio *et al* [15] en appliquant un filtre de Gabor 2D au volume spatiotemporel. D'autres approches reposent sur la transformation en ondelettes. A titre d'exemple, nous pouvons citer les travaux de Rapantzikos et *al* [106] qui appliquent des transformations en ondelettes discrètes dans chacune des trois directions d'un volume vidéo. Quelques années *plus tard,* Rapantzikos et *al* [107], ont utilisé des informations sur la couleur, l'intensité et le mouvement pour déterminer des points d'intérêt. En 2008, Willems et ses collaborateurs [141] ont utilisé le détecteur Hessian pour définir la présence de points saillants dans le volume spatio-temporel. Zhen et Shao [148] ont

décomposé les volumes spatio-temporels en sous-volumes utilisant la pyramide orientable spatio-temporelle (STSP).

1.2.2 Descripteurs locaux

Hormis les approches basées sur les points d'intérêts spatio-temporels, d'autres travaux se sont intéressés à la description du voisinage spatio-temporel des points détectés. Plusieurs descripteurs ont été proposés dans la littérature, parmi lesquels nous pouvons citer le descripteur donné par Laptev et Lindeberg dans [75]. Les auteurs ont proposé de calculer l'histogramme de flot optique et d'orientations du gradient sur les volumes 3*D* locaux. En 2008, Klaser et *al* [69] ont proposé une extension du descripteur d'image HOG au domaine spatio-temporel 3D (HOG3D). En 2007, Scovanner et *al* dans [113], ont suggéré une extension spatiotemporelle du descripteur d'image SIFT (Scale Invariant Feature Transform) de Lowe [89,90]. Ce descripteur est basé sur les zones régulièrement réparties sur l'image et les gradients spatio-temporels. En 2008, Willems *et al.* [141] ont proposé le descripteur E-SURF (Extended SURF) extension du descripteur SURF 2D de [5] sur les vidéos.

1.3 Méthodes basées apparences

Les méthodes basées apparences proposent des caractéristiques qui encodent les images dans leur globalité. Le principe utilisé dans cette méthode est de générer une image (appelée aussi "carte") qui représente le contenu de la séquence vidéo, et de se baser sur cette image pour la classification. Nous allons présenter les approches les plus populaires pour chacune de ces catégories dans ce qui suit.

1.3.1 Analyse de la silhouette

Dans cette approche, Les images d'énergie (MEI : Motion Energy Image) et d'historique du mouvement (MHI : Motion History Image) ont été introduites par Bobick et Davis dans [13]. Les MEIs et les MHIs sont calculées en accumulant les images de différence sur un certain nombre d'instants consécutifs. La figure (1.1) représente la trace des mouvements de la silhouette.

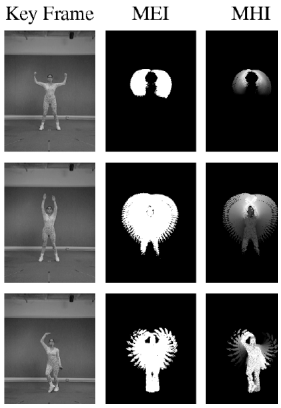


Figure 1.1. Illustration du modèle holistique : Images d'Énergie du Mouvement (MEI) et images de l'historique du mouvement (MHI) [13].

En 2005, Shechtman et Irani [116] ont proposé une solution de reconnaissance d'actions en utilisant des patchs au lieu de la silhouette complète. Ils extraient, pour chaque image de la vidéo, la silhouette de la personne filmée en soustrayant le fond. La figure 1.2 présente une illustration de ce principe.

Figure 1.2. Illustration du modèle holistique : volume spatio-temporel

Blank et *al* [12,46] ont appliqué l'équation de poisson pour déduire les caractéristiques de la forme et l'orientation d'un pixel par rapport à son voisinage.
Huang et Trivedi [60] ont créé le concept d'histogramme cylindrique, basé sur une reconstruction tridimensionnelle du personnage à partir de silhouettes multi-vues synchronisées. De même, Weinland et *al* [140] ont étendu le travail de Bobick et *al* [13] au volume de l'historique des mouvements 3D afin de combiner les silhouettes issues de plusieurs caméras pour construire une représentation basée sur les pixels en 3D. En 2008, Ragheb et *al* [104] ont transformé le MHV dans le domaine de Fourier. Ils

construisent les MHVs ; puis ce volume est divisé en sous-volumes. Le vecteur des caractéristiques correspond à la moyenne des fréquences obtenues. Xiong et Liu [145] utilisent les chaînes de Markov sur des descripteurs extraits des silhouettes. Yilmaz et Shah [146] ont proposé une méthode basée sur la création d'un volume 3D à partir des contours de silhouette. Ke et *al* [68] ont suggéré une modélisation du volume spatiotemporel des silhouettes. Ils segmentent les vidéos en des volumes 3D homogènes avec un algorithme de partitionnement non supervisé.

- **Méthodes basées sur le calcul du flot optique et du gradient**

Cette catégorie utilise le flot optique calculé entre les frames de la séquence vidéo. Parmi les travaux de l'état de l'art qui sont directement liés à cette catégorie, citons Efros et *al* [32] qui ont calculé le flot optique de manière dense dans les séquences centrées sur la personne et divisent la direction du flot en quatre champs scalaires différents (correspondant à la composante négative et positive, horizontale et verticale de flot). Cette approche a été utilisée plus tard par Robertson et *al* [109] et Wang et *al* [139].

L'inconvénient majeur de ces approches est que la différence des images peut être considérée comme une conséquence d'un mouvement.

1.3.2 Analyse du squelette

Cette technique sert à représenter la silhouette par des segments. En 1973, le psychophysicien suédois Gunnar Johansson [64] a représenté le mouvement du corps humain par des points lumineux décrivant les mouvements des articulations principales (Figure 1.3).

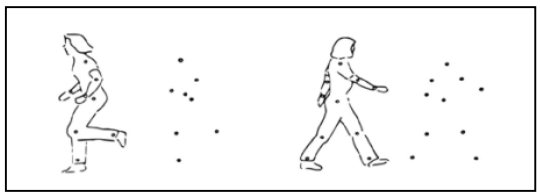
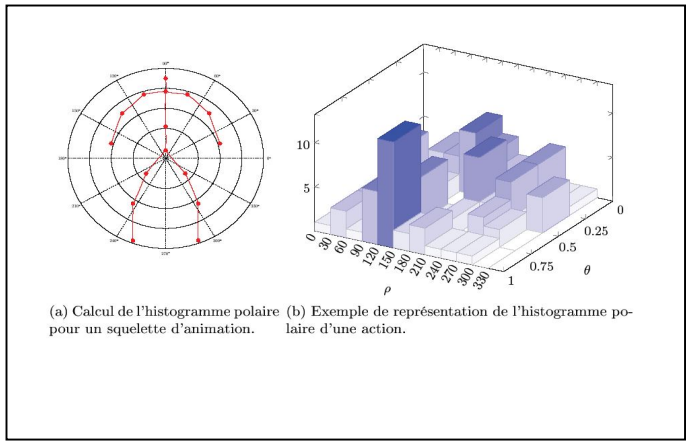


Figure 1.3 Illustration du mouvement des points repères [64]

L'une des utilisations les plus anciennes du squelette humain simplifié remonte à Gavrila et Davis [41]. Les auteurs procèdent à l'extraction d'un squelette simplifié à partir d'une capture de mouvements 3D, issus de plusieurs caméras. Quelques années *plus tard,* Han et *al* [49] exploitent la capture de mouvements pour représenter les trajectoires de chacune des articulations dans un espace manifold. En 2013, Raptis et *al* [108] ont utilisé la capture de mouvements de Shotton et *al* [117] et de la caméra Kinect™, pour interpréter l'action de danse. Fujiyoshi et Lipton [37] introduisent le concept de star skeletonization ou star skeleton pour représenter le corps humain par l'intermédiaire d'un squelette à cinq branches maximum, représentant les extrémités des jambes et des bras, ainsi que la tête. Ziaeefard et Ebrahimnezhad [150] ont utilisé l'histogramme polaire des positions des articulations au cours du temps, comme présenté sur la figure 1.4.



(a) Calcul de l'histogramme polaire pour un squelette d'animation. (b) Exemple de représentation de l'histogramme polaire d'une action.

Figure 1.4 Histogramme polaire du squelette d'animation [150]

En 2012, Xia et *al* [143] ont utilisé des histogrammes des positions 3D des articulations (Histograms Of 3D Joint HOJ3D) pour encoder l'occupation spatiale des articulations par rapport au centre de la silhouette (Figure 1.5). La phase qui suit consiste à construire le vecteur de primitive par une quantification vectorielle en utilisant l'algorithme de K-means. Une méthode de reconstruction de pose 3D en temps réel a été proposée par Ke et *al* [68]. Ils utilisent un algorithme d'optimisation pour aligner les vecteurs primitives 2D suivies avec les modèles humains 3D prédéfinis.

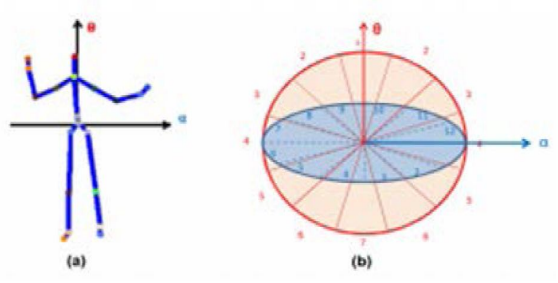


Figure 1.5 (a) les coordonnées de référence de HOJ3D.
(b) système de coordonnées sphérique [143]

Ramanan et *al.* [105] ont développé un modèle de forme articulé 3D pour décrire la configuration du corps humain.

En 2014, Vemulapalli et ses collaborateurs [131] ont représenté chaque squelette dans un groupe de Lie. Les squelettes constituant l'action sont modélisés par une courbe comme le montre la figure 1.6.

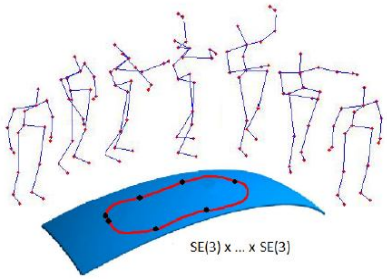


Figure 1.6 Représentation d'une action (séquence squelettique) sous forme de courbe dans le groupe de Lie SE (3)x...x SE (3) [131]

Dans des travaux plus récents, Ghorbel et *al* [42] ont utilisé un descripteur d'action humaine rapide et précis nommé Kinematic Spline Curves. Pour surmonter la variabilité anthropométrique et la vitesse d'exécution, les auteurs ont proposé l'utilisation d'une normalisation squelettique et d'une normalisation temporelle. Quelques années plus tard, Ghorbel et *al* [43, 44] ont utilisé les données du squelette fournies par les caméras RGB-D pour former les vecteurs primitives. Liu et *al* [87] ont proposé un réseau LSTM, Global Context-Aware Attention LSTM (GCA-LSTM), pour la reconnaissance d'action basée sur le squelette. Pour améliorer la capacité d'attention de

ce réseau, les auteurs utilisent un mécanisme d'attention récurrent, avec lequel les performances d'attention du réseau peuvent être améliorées progressivement. En 2020, Fan et *al* [33] proposent un module d'attention croisé qui se compose d'une branche d'auto-attention et d'une branche d'attention croisée pour la reconnaissance d'action basée sur le squelette. Ce module permet d'extraire les articulations qui sont hautement corrélées aux informations de contexte de scénario correspondant. De plus, le module d'attention croisé maintient la taille des variables d'entrées est intégré de manière flexible dans les frames works existants sans casser leurs comportements.

1.4 Approche d'apprentissage profond

Le Deep Learning (apprentissage profond) est une forme d'intelligence artificielle, dérivée du Machine Learning (apprentissage automatique). Parmi ces techniques, on compte les réseaux de neurones artificiels**.** Plus récemment, les approches basées sur CNN ont été étudiées pour apprendre les représentations vidéo pour la reconnaissance d'action. Cet intérêt est traduit par de nombreuses études bibliographiques proposées aux cours des années, LeCun et *al* [79], Krizhevsky et *al* [72], Simonyan et Zisserman [121] , Karpathy et *al* [66] , Feichtenhofer et *al* [34],Tran et Cheong [127]; Wang et *al*., [137], Hu et *al* [59], Carreira et Zisserman [21] .

Les réseaux convolutifs (Convolutionnal Neural Network – CNN) ont été démocratisés par LeCun et *al* [79] en 1995 qui les ont utilisés avec succès pour la reconnaissance de caractères. Le premier grand succès des CNN lors de ce concours date de 2012 par Krizhevsky et *al* [72], en utilisant la puissance des processeurs graphiques (Graphics Processing Unit – GPU), Krizhevsky et *al* [72] ouvraient la voie à de nombreux autres systèmes (VGG, ResNet, etc.) [66,134]. Pour explorer les informations spatio-temporelles dans les actions humaines, l'architecture à deux flot est d'abord proposée dans [121] où deux CNN 2D sont respectivement appliqués aux domaines d'apparence (trames RVB) et de mouvement (flot optique empilé). Les fonctionnalités des deux modalités sont fusionnées à l'étape finale et l'architecture à deux flot atteint une précision de reconnaissance d'actions élevées sur la base de cette architecture, plusieurs mécanismes sont proposés pour une meilleure fusion des deux réseaux de branches sur

l'apparence et le mouvement [30, 34, 66, 127,137, 138]. D'un autre côté, les premières tentatives qui incorporent le LSTM avec des caractéristiques traditionnelles ont montré le potentiel de l'architecture CNN-LSTM pour la modélisation des informations spatio-temporelles dans la reconnaissance des actions [31]. Les réseaux LSTM sont utilisés pour modéliser explicitement les relations spatio-temporelles. Le travail de [59] attire l'attention de différents canaux pour améliorer encore la qualité des représentations produites par un réseau. Dans [27], une couche de corrélation est appliquée dans l'architecture LSTM à deux flots pour résoudre le problème de corrélation entre les données d'apprentissage.

Le CNN 3D pour la reconnaissance de l'action est d'abord présenté dans [63] pour apprendre les caractéristiques discriminantes le long des dimensions spatiales et temporelles. Plus tard, la fonctionnalité C3D ainsi que les architectures CNN 3D correspondantes sont présentées dans [128]. Les auteurs utilisent des noyaux de convolution 3D pour modéliser à la fois les informations spatiales et temporelles. En 2017, Tran et *al* [129] ont proposé l'architecture Res3D pour faciliter le processus d'apprentissage. De même, Carreira et *al* [21] ont proposé le réseau Inception I3D donné par [62] pour représenter la vidéo. Récemment, plusieurs approches ont été proposées dans la littérature pour améliorer la convolution 3D [39, 52, 71, 130, 144,149]. Cependant, toutes les méthodes sont trop coûteuses en termes du temps de calcul que leurs concurrents 2D en raison de la nouvelle dimension temporelle, ce qui rend les modèles difficiles à former et peu pratique dans les applications du monde réel. Pour diminuer le nombre de paramètres, ces travaux [81, 103,123] décomposent un noyau de convolution 3D en une combinaison d'un noyau spatial 2D et d'un noyau temporel 1D.

1.5 Jeux de données pour la reconnaissance des actions

Nous allons présenter dans cette partie plusieurs bases de données de classification des actions humaines. Le tableau 1.1 illustre une liste d'ensembles de données existants en fonction du nombre d'actions et le nombre de clips par action triés par année.

Base de données	Référence	Année	Actions	Clips
KTH	[114]	2004	6	100
Weizmann	[12]	2005	9	9
Sport Dataset (UCF Sport)	[110]	2008	9	14-35
Hollywood Dataset 1	[77]	2008	8	30-129
Youtube Dataset (UCF YouTube)	[85]	2009	11	100
Hollywood Dataset 2	[92]	2009	12	61-278
TUM Kitchen Dataset	[126]	2009	4	21
TV Human Interactions Dataset	[102]	2010	4	300
HMDB51	[73]	2011	51	101
UESTC-SAR	[23]	2011	8	531
MSR Daily Activity 3D	[136]	2012	3	16
InfAR dataset	[40]	2016	12	50
Kinetics400	[67]	2017	400	400-1150

Tableau 1 .1 Liste d'ensembles de données

Dans nos expériences du chapitre 5, nous utilisons les ensembles de données InfAR dataset [40].

1.6 Conclusion

Dans ce chapitre, nous avons présenté les méthodes pour l'extraction des caractéristiques. Nous avons divisé ces méthodes en trois classes distinctes : les méthodes basées sur des caractéristiques locales, les méthodes basées apparence et les méthodes d'extraction automatique basées sur l'apprentissage profond. On a décrit les différents types de bases de données les plus populaires utilisées pour évaluer les méthodes de reconnaissance d'action.

Dans le chapitre suivant, on s'intéressera au calcul du flot optique afin d'exploiter l'aspect temporel entre deux images consécutives (et donc 2 instants de prise d'image successifs), et non entre une image de départ (t = 0) et une image prise à un instant donné *t*.

CHAPITRE 02

ESTIMATION DU FLOT OPTIQUE

2.1 Introduction

Avec la généralisation de l'utilisation de caméras vidéo, l'analyse du mouvement dans les vidéos s'est révélée être un outil indispensable pour des applications aussi diverses que la recherche et la classification de vidéos, la vidéo surveillance, l'analyse des activités humaines, la robotique, l'interaction homme machine et l'analyse de séquences sportives.

La perception et la mesure du mouvement sont effectuées à partir des variations temporelles des intensités lumineuses dans une séquence d'images, ces variations constituent le flot optique, c'est-à-dire le champ bidimensionnel des vitesses. Ainsi, nous définissons le flot optique comme un champ de déplacement visuel décrivant les changements d'intensité des pixels dans une image animée en fonction du déplacement de points images [3,54].

Dans ce chapitre, nous décrivons quelques techniques parmi les plus populaires utilisées en estimation du flot optique.

2.2 Les méthodes d'estimation du flot optique

2.2.1 Méthodes variationnelles

En 1966, Gibson [45, 88] a introduit le premier modèle d'estimation du mouvement. Quelques années plus tard, des algorithmes ont été proposés dans l'état de l'art par [1, 22, 50, 58,98]. Actuellement plusieurs méthodes traitent le problème d'estimation du mouvement sous la forme d'un système d'équations différentielles [16,

17,18] comme par exemple les méthodes par filtrage [9,10,29,122], les méthodes statistiques [118, 119, 120] et les méthodes variationnelles [16, 17,18].

2.2.1.1 Luminosité constante

Les méthodes variationnelles reposent sur le principe de la constance de l'intensité des pixels au cours du mouvement, c'est à dire, l'intensité d'un point reste constante le long de sa trajectoire. Ce principe est appelé la contrainte fondamentale du flot optique. Cette contrainte reste valable dans le cas des petits déplacements, car les changements de luminosité d'une image à l'autre étant faibles.

En admettant que *I(x, y, t)* soit l'intensité du pixel à la position *(x, y)* dans l'image au temps *t*. Cette intensité est donnée par :

$$\mathbf{f}: \boldsymbol{\Omega} \times [\mathbf{0}, \mathbf{T}] \rightarrow \mathbb{R}$$
$$(\mathbf{x}, \mathbf{y}), \mathbf{t} \mapsto \mathbf{I}(\mathbf{x}, \mathbf{y}, \mathbf{t})$$

La trajectoire d'un pixel à l' instant *t* est donnée :

$$\mathbf{t} \mapsto (\mathbf{x}(\mathbf{t}), \mathbf{y}(\mathbf{t}), \mathbf{t})$$

à l'instant $\mathbf{t_0}$ cette trajectoire est définie par:

$$(\mathbf{x}(\mathbf{t_0}, \mathbf{y}(\mathbf{t_0}), \mathbf{t_0}) = (\mathbf{x_0}, \mathbf{y_0}, \mathbf{t_0})$$

et

$$I(x(t), y(t), t) = I(x_0, y_0, t_0) \quad \forall\, t \tag{2.1}$$

En dérivant l'équation (2.1) par rapport à $\mathbf{t}$ à l'instant $\mathbf{t} = \mathbf{t_0}$, on obtient :

$$\frac{\partial \mathbf{I}}{\partial \mathbf{t}}(\mathbf{x_0}, \mathbf{y_0}, \mathbf{t_0}) + \frac{\partial \mathbf{I}}{\partial \mathbf{x}}(\mathbf{x_0}, \mathbf{y_0}, \mathbf{t_0}) \frac{\mathbf{dx}}{\mathbf{dt}}(\mathbf{t_0}) + \frac{\partial \mathbf{I}}{\partial \mathbf{y}}(\mathbf{x_0}, \mathbf{y_0}, \mathbf{t_0}) \frac{\mathbf{dy}}{\mathbf{dt}}(\mathbf{t_0}) = \mathbf{0} \tag{2.2}$$

Où

$\frac{dx}{dt}$ **et** $\frac{dy}{dt}$ représentent respectivement les déplacements suivant $\mathbf{x}$ et y .

Donc le flot optique est défini par :

$$\mathbf{u}(\mathbf{x_0}, \mathbf{y_0}) = (\mathbf{u_1}(\mathbf{x_0}, \mathbf{y_0}), \mathbf{u_2}(\mathbf{x_0}, \mathbf{y_0}))$$

Où

$$\mathbf{u_1(x_0, y_0)} = \frac{\mathbf{dx}}{\mathbf{dt}}\mathbf{(t_0)\ et\ u_2(x_0, y_0)} = \frac{\mathbf{dy}}{\mathbf{dt}}\mathbf{(t_0)}$$

En utilisant la notation $\mathbf{I_*} = \frac{\partial \mathbf{I}}{\partial *}$, dans l'équation (2.2) on obtient :

$$\mathbf{I_x u_1 + I_y u_2 + I_t = 0} \quad (2.3)$$

Cette équation est connue sous le nom d'équation de contrainte du mouvement apparent. Cette équation à elle seule ne peut suffire à estimer le flot optique dans la séquence d'images. En effet, (2.3) est une équation scalaire à deux inconnues $\boldsymbol{u_1}$ **et** $\boldsymbol{u_2}$. Le problème est mal posé. Il est communément appelé problème de l'ouverture. Afin d'y remédier à ce problème, plusieurs méthodes ont été proposées dans la littérature**.** Nous en détaillons dans la suite deux d'entre elles : la méthode locale et la méthode globale.

2.2.1.2 Méthode Locale

Cette approche consiste à prendre en compte des hypothèses supplémentaires sur un domaine de taille réduite pour particulariser le flot optique. En 1981, Lucas et Kanade [91] ont proposé une hypothèse locale. Les auteurs supposent que les points appartenant à un même voisinage ont un déplacement proche. Cette méthode suppose que le flot optique est constant dans un voisinage local du pixel considéré, et résout l'équation (2.4) pour tous les pixels dans ce voisinage par la méthode des moindres carrés**.**

$$\iint_{\boldsymbol{v}(\mathbf{x},\mathbf{y})} (\mathbf{I_{x'} u_1 + I_{y'} u_2 + I_t})^2 \mathbf{dx'\, dy'} \quad (2.4)$$

La minimisation de l'équation (2.4) peut être remplacée par la minimisation de :

$$\mathbf{1}_{\boldsymbol{v}(\mathbf{x},\mathbf{y})} * (\mathbf{I_x u_1 + I_y u_2 + I_t})^2$$

Pour donner un rôle plus important aux pixels les plus proches de $(\boldsymbol{x}, \boldsymbol{y})$ **,** les auteurs remplacent la fonction indicatrice par une fonction gaussienne de déviation standard $\boldsymbol{\rho}$ notée $\mathbf{K_\rho}$. Finalement, la méthode de Lucas et Kanade vise à minimiser l'expression (2.5) pour chaque pixel $(\boldsymbol{x}, \boldsymbol{y})$

$$\mathbf{Loc(u)} = \mathbf{K_\rho} * (\mathbf{I_x u_1 + I_y u_2 + I_t})^2 \quad (2.5)$$

En utilisant les équations d'Euler-Lagrange, le problème à résoudre est donné par

$$\begin{bmatrix} K_\rho * (I_x)^2 & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y)^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -K_\rho * (I_x I_t) \\ -K_\rho * (I_y I_t) \end{bmatrix} \quad (2.6)$$

La méthode locale est connue pour sa robustesse au bruit présent dans les données traitées. Le temps de calcul est également un avantage. En revanche, les méthodes locales souffrent de leur manque de précision de la solution au niveau des petits objets ainsi que sur les discontinuités (bords des objets).

2.2.1.3 Méthode Globale

En 1981, Horn et Schunck [58] ont proposé une approche globale pour résoudre le problème de l'ouverture. Ils supposent que le champ de vecteurs soit globalement régulier et considèrent la minimisation de la fonctionnelle

$$\mathbf{Glob(u)} = \iint_{\Omega} \underbrace{((\mathbf{I_x u_1} + \mathbf{I_y u_2} + \mathbf{I_t})^2}_{\mathbf{D}} + \underbrace{\boldsymbol{\alpha}(|\boldsymbol{\nabla} \mathbf{u_1}|^2 + |\boldsymbol{\nabla} \mathbf{u_2}|^2))}_{\mathbf{R}} \mathbf{dxdy} \quad (2.7)$$

Où $\boldsymbol{\alpha}$ est un paramètre de régularisation constant jouant le rôle de pénaliseur pour les gradients de la solution. Le but de cette approche est de déterminer les deux inconnues $(\mathbf{u_1}, \mathbf{u_2})$ vérifiant la contrainte fondamentale du flot optique (terme D) en négligeant le terme R.

Plusieurs travaux ont été proposés dans la littérature pour calculer les champs de vecteurs discontinus $\boldsymbol{u} = (\mathbf{u_1}, \mathbf{u_2})$. La plupart de ces travaux reposent sur la modification du terme de régularisation R [3].

En 1992, Black [11] a introduit les normes robustes pour modifier la fonctionnelle de Horn et Schunk. Il remplace le terme de régularisation par :

$$\iint_{\Omega} \left(\boldsymbol{\psi}(|\boldsymbol{\nabla} \mathbf{u_1}|) + \boldsymbol{\psi}(|\boldsymbol{\nabla} \mathbf{u_2}|)\right) \mathbf{dxdy}$$

où les $\boldsymbol{\psi}$ sont des fonctions permettant d'ôter le bruit et de conserver les bords des objets de l'image. Les auteurs Cohen [25] Kumar, Tannenbaum et Balas [74] ont proposé la variation totale de la fonction $(\boldsymbol{\psi}(\mathbf{t}) = \mathbf{t})$ [54]. Gupta et *al* [48] ainsi que Guichard et Rudin [47] ont modifié le terme de régularisation de Black en ajoutant des termes basés sur la divergence ou le rotationnel du flot optique en considérant

$$\iint_{\Omega} \boldsymbol{\varphi}(\mathbf{div}(\mathbf{u}), \mathbf{rot}(\mathbf{u}))\mathbf{dxdy}$$

Dans [47], le terme de régularisation est donné par :

$$\iint |\mathbf{div}(\mathbf{u})|\,\mathbf{dxdy}$$

En 1993, Nési [99] a modifié la formulation de Horn et Schunck en introduisant la longueur de l'ensemble des discontinuités de $\mathbf{u}$ noté $|\mathbf{S_u}|$. Le terme de régularisation est de la forme :

$$\iint_{\Omega} \boldsymbol{\alpha}(|\boldsymbol{\nabla}\mathbf{u_1}|^2 + |\boldsymbol{\nabla}\mathbf{u_2}|^2)\mathbf{dxdy} + \boldsymbol{\alpha}^{\mathbf{d}}|\mathbf{S_u}| \qquad (2.8)$$

Cette idée a été exploitée dans le domaine de la segmentation d'image par Mumford et Shah [97].

2.2.1.4 Combinaison Locale–Globale

En 2004, Bruhn *et al* [19,20] ont proposé la méthode Locale-Globale pour calculer le flot optique, cette méthode consiste à minimiser la quantité suivante :

$$CLG(\mathbf{u}) = \iint_{\Omega} (\mathbf{K_\rho} * (\mathbf{I_x u_1} + \mathbf{I_y u_2} + \mathbf{I_t})^2 + \boldsymbol{\alpha}(|\boldsymbol{\nabla}\mathbf{u_1}|^2 + |\boldsymbol{\nabla}\mathbf{u_2}|^2))\mathbf{dxdy} \qquad (2.9)$$

2.2.1.5 Méthodes adaptatives

Dans [6,7], les auteurs ont proposé une approche adaptative pour contrôler le paramètre de régularisation $\boldsymbol{\alpha}$ dans l'équation (2.9). Ils considèrent que le paramètre $\boldsymbol{\alpha}$ n'est plus une constante mais une fonction constante par morceaux dépendante des directions $\mathbf{x}$ et $\mathbf{y}$ [54]. La fonctionnelle à minimiser (2.9) devient

$$\iint_{\Omega} \underbrace{(\mathbf{K_\rho} * (\mathbf{I_x u_1} + \mathbf{I_y u_2} + \mathbf{I_t})^2}_{\text{données}} + \underbrace{\boldsymbol{\alpha}(\mathbf{x},\mathbf{y})(|\boldsymbol{\nabla}\mathbf{u_1}|^2 + |\boldsymbol{\nabla}\mathbf{u_2}|^2)}_{\text{régularisation}}\,\mathbf{dxdy} \qquad (2.10)$$

Au niveau des bords des différents objets, le paramètre $\boldsymbol{\alpha}$ est diminué afin de minimiser la diffusion et donc d'obtenir des angles mieux définis. Sur les zones homogènes, la fonction $\boldsymbol{\alpha}(\mathbf{x},\mathbf{y})$ conserve sa valeur initiale.

2.2.1.6 Mesure de grands déplacements

La méthode de Lucas & Kanade est considérée comme la méthode la plus facile à mettre en œuvre, cette méthode est inadaptée lorsqu'il s'agit de traiter de grands déplacements. En effet, la contrainte fondamentale du flot optique (2.3) peut être obtenue grâce à un développement de Taylor :

$$I(x + u_1, y + u_2, t + 1) = I(x, y, t + 1) + I_x . u_1 + I_y . u_2 \quad (2.11)$$

$$= I(x, y, t) \quad (2.12)$$

I_t étant en pratique égal à $I(x, y, t + 1) - I(x, y, t)$.

L'intégrale à minimiser (2.9) s'écrit donc sous la forme

$$\iint_{\Omega} \underbrace{\mathbf{K_\rho} * \left(\mathbf{I(x + u_1, y + u_2, t + 1) - I(x, y, t)}\right)^2 \mathbf{dxdy}}_{\text{non linéaire}} + \iint_{\Omega} \underbrace{\boldsymbol{\alpha}(\mathbf{x}, \mathbf{y})(|\nabla \mathbf{u_1}|^2 + |\nabla \mathbf{u_2}|^2) \mathbf{dxdy}}_{\text{linéaire}} \quad (2.13)$$

Pour linéariser ce problème, une des solutions possibles [20,124] consiste à utiliser la méthode de point fixe puis à faire la décomposition de la variable suivante

$$\mathbf{u^n} = (\mathbf{u_1^n}, \mathbf{u_2^n})$$

$\mathbf{u^{n+1}} = \mathbf{u^n} + \boldsymbol{\delta u^n}$ **où** $\boldsymbol{\delta u^n} = (\boldsymbol{\delta u_1^n}, \boldsymbol{\delta u_2^n})$ **et** $\mathbf{u^{n+1}} = \left(\mathbf{u_1^{n+1}}, \mathbf{u_2^{n+1}}\right) = (\mathbf{u_1^n} + \boldsymbol{\delta u_1^n}, \mathbf{u_2^n} + \boldsymbol{\delta u_2^n})$

On obtient alors

$$\mathbf{I(x + u^{n+1}) = I(x + u^n + \delta u^n)},$$
$$= \mathbf{I(x + u^n) + I_x(x + u^n) . \delta u_1^n + I_y(x + u^n) . \delta u_2^n}$$
$$= \mathbf{I(x)}.$$

Cette idée a notamment été exploitée par LeBesnerais et Champagnat dans [78]. Une année plus tard, Papenberg et ses collaborateurs [101] ont fait de même dans le cadre de la régularisation globale.

2.2.2 Méthodes fréquentielles

Les méthodes fréquentielles consistent à prendre en compte la transformée de Fourier de l'équation (2.3) du flot optique ce qui nous donne

$$\mathbf{f_x u_1 + f_y u_2 + f_t = 0} \quad (2.14)$$

Où f_x , f_y sont les fréquences spatiales et f_t la fréquence temporelle.

Les Filtres de Gabor sont des filtres 3D formés par le produit d'une gaussienne et d'une fonction trigonométrique, soit :

$$\mathbf{g(x, y, t) = \frac{1}{(2\pi)^3 \sigma_x \sigma_y \sigma_t} e^{\frac{-x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_x^2} - \frac{t^2}{2\sigma_x^2}} . \cos\left(2\pi(f_{x0}x + f_{y0}y + f_{t0}t\right)} \quad (2.15)$$

Où : $(\boldsymbol{\sigma_x, \sigma_y, \sigma_t})$ sont les ecarts types de la gaussienne et $\mathbf{(f_{x0}, f_{y0}, f_{t0})}$ sont les fréquences centrales du filtre**.** La transformée de Fourier de ce filtre est une gaussienne centrée en $\mathbf{(f_{x0}, f_{y0}, f_{t0})}$ et d'écart type $(\frac{1}{\sigma_x}, \frac{1}{\sigma_y}, \frac{1}{\sigma_t})$.

2.2.2.1 Méthodes par filtrage

Plusieurs méthodes proposent une combinaison de filtrages pour évaluer différents ordres de grandeur de vitesse, soit par une approche multi-résolution (pyramide de Heeger), soit en utilisant plusieurs filtres de Gabor sur la même image [93]. Parmi les travaux de l'état de l'art qui sont directement liés à cette catégorie, citons Fleet et Jepson [36] qui ont utilisé des filtres de Gabor sur les images obtenues après la convolution. Il existe également des approches utilisant des ondelettes particulières [142] (équivalentes à des projections sur des filtres particuliers).

L'inconvénient majeur de ces méthodes est l'utilisation excessive du filtrage, qui entraîne une perte d'information préjudiciable notamment en ce qui concerne les petits mouvements.

2.2.2.2 Méthode de corrélation de phase

Dans cette approche (f_x et f_y variables dans l'espace de Fourier), f_1 et f_2 étant deux blocs candidats, appartenant aux deux images successives que l'on considère :

$$\hat{f}_2(f_x, f_y) = \hat{f}_1(f_x, f_y) e^{-i(f_x u_1 + f_y u_2)}$$

Puis $$\frac{\hat{f}_2\ \hat{f}_1^*}{|\hat{f}_2\ \hat{f}_1^*|} = e^{-i(f_x u_1 + f_y u_2)} \quad (2.16)$$

La solution du problème est donnée par :

$$C_{t,t+1}(u_1, u_2) = F^{-1}\left(\frac{\hat{f}_2\ \hat{f}_1^*}{|\hat{f}_2\ \hat{f}_1^*|}\right) \quad (2.17)$$

Où F^{-1} désigne l'opérateur transformé de Fourier inverse.

Le mouvement estimé est donné par :

$$[u_1, u_2] = argmax(\Re(C_{t,t+1})) \quad (2.18)$$

Cette méthode ne renvoie pas un résultat dense (seuls les maximas locaux sont considérés) [93].

2.2.3 Block Matching

Dans cette approche, on recherche la "meilleure correspondance" entre deux fenêtres spatiales (ou blocs) situées dans des trames consécutives. On autorise pour cela un déplacement maximal et on cherche le bloc qui correspond au mieux au bloc initial, de manière à minimiser un critère d'erreur (corrélation).

Les critères de corrélation les plus utilisés dans la littérature sont la différence quadratique moyenne (Mean Square Difference MSD), la Différence Absolue Moyenne (Mean Absolute Difference MAD), la Somme Absolue des Différences Transformées ou Sum of Absolute Transformed Differences (SATD) et la somme des différences absolue (Sum Absolute Difference) SAD.

L'inconvénient majeur des approches de type block matching est qu'elles postulent un déplacement maximal sur une certaine fenêtre, on exclut ainsi les grands déplacements [93].

2.3 Approche retenue

Le choix de l'approche est basé sur les critères suivants : le résultat doit être dense (une estimation pour chaque pixel), le plus précis possible (subpixelique), dépourvu de la paramétrisation du flot et de filtrages excessifs. Du fait de ces

contraintes, les méthodes fréquentielles ne sont pas retenues. En effet, la corrélation de phase donne des pics locaux du flot et donc un résultat non dense et les méthodes par filtrage effectuent un lissage trop important et sont soumises à un réglage de paramètres conséquent. L'approche répondant à nos exigences est donc les méthodes variationnelles (globales et locales).

2.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés au problème de l'estimation du mouvement dans des séquences d'images par le calcul du flot optique. Nous avons décrit les méthodes les plus couramment utilisées pour extraire une information essentielle sur les séquences d'images : le mouvement apparent. Après un bref rappel des problèmes rencontrés pour estimer le flot optique, nous avons discuté brièvement des trois grands groupes de méthodes d'estimation : les méthodes variationnelles, les méthodes fréquentielles et la méthode block matching**.**

CHAPITRE 03

APPRENTISSAGE AUTOMATIQUE : LES RÉSEAUX DE NEURONES

3.1 Introduction

Depuis plus de trois décennies, l'utilisation des réseaux de neurones artificiels (RNA) s'est développée dans de nombreuses disciplines (sciences économiques, écologie et environnement, biologie et médecine…etc). Ils sont notamment appliqués pour résoudre des problèmes de classification, de prédiction, de catégorisation, d'optimisation et de reconnaissance des formes. Les RNA sont également applicables dans toutes les situations où il existe une relation non linéaire entre une variable prédictive et une variable prédite. Par leur nature et leur fonctionnement, les RNA peuvent détecter les interactions multiples non linéaires parmi une série de variables d'entrée, ils peuvent donc gérer des relations complexes entre les variables indépendantes et les variables dépendantes.

L'objectif général d'un RNA est de trouver la configuration des poids de connexion entre neurones pour qu'il associe à chaque configuration d'entrée, une réponse adéquate. L'utilisation d'un RNA se fait en deux temps. Tout d'abord une phase d'apprentissage qui est chargée d'établir des valeurs pour chacune des connexions du réseau, puis une phase d'utilisation proprement dite, où l'on présente au réseau une entrée et où il nous indique en retour « sa » sortie calculée.

Dans ce chapitre, nous allons décrire la similarité entre le modèle de neurone biologique et ce lui de neurone Formel. Nous verrons dans un premier temps l'historique de RNA en parcourant les différents travaux sur la modélisation du neurone biologique.

Par la suite, nous expliciterons les différents types des fonctions d'activation dont leur rôle est d'appliquer une transformation mathématique aux informations qui arrivent aux neurones. En d'autres termes, il s'agit des fonctions qui traitent les informations des neurones. Nous introduirons ainsi le principe de perceptron multi couches. En dernier lieu, nous décrirons les différentes techniques d'entraînement ou d'apprentissage de ces réseaux.

3.2 Historique

En 1890, le psychologue américain W. James [96] introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb. En 1943, J. Mc Culloch et W. Pitts [95] ont introduit une modélisation du neurone biologique (un neurone au comportement binaire). Ce sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique). Dans la même période, le physiologiste américain D. Hebb explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à cette heure précise même en l'absence de nourriture [96]. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux.

En 1958, F. Rosenblatt [111] développe le modèle du Perceptron. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception et une couche liée à la prise de décision. C'est le premier système artificiel capable d'apprendre par expérience. Plus tard, B. Widrow (1960) [96] présente le perceptron linéaire appelé ADALINE (adaptive linear element) qui peut donner

n'importe quelles valeurs de sorties, autre que zéro et un . Ce modèle sera par la suite le modèle de base de l'algorithme de rétropropagation de gradient très utilisé aujourd'hui avec les Perceptrons multicouches. Quelques années plus tard M. Minsky et S. Papert [96] publient un ouvrage (Le perceptron) qui met en évidence les limitations théoriques des Perceptrons, car ils sont incapables de résoudre des problèmes non linéaires comme le cas du XOR (le OU Exclusif) cette fonction logique qui n'est pas linéairement séparable, ce qui provoque un abandon des recherches jusqu'en 1972, où T. Kohonen [70] présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes. C'est en 1982 que J. Hopfield [57] présente l'étude d'un réseau complètement rebouclé, dont il analyse la dynamique.

La machine de Boltzmann (1983) est le premier modèle connu apte à traiter de manière satisfaisante les limitations recensées dans le cas du perceptron. Mais l'utilisation pratique s'avère difficile, la convergence de l'algorithme étant extrêmement longue (les temps de calcul sont considérables). Dans la même période, La rétropropagation de gradient apparaît en 1985. C'est un algorithme d'apprentissage adapté aux réseaux de neurones multicouches (aussi appelés Perceptrons multicouches). Sa découverte réalisée par trois groupes de chercheurs indépendants. Dès cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau en décomposant cette fonction en une suite d'étapes linéairement séparables. De nos jours, les réseaux multicouches et la rétropropagation de gradient reste le modèle le plus étudié et le plus productif au niveau des applications.

3.3 Du Neurone Biologique au Neurone Formel

3.3.1 Neurones Biologiques

En biologie, un neurone est une cellule nerveuse dont la fonction est de transmettre un signal électrique dans certaines conditions. Il agit comme un relai entre une couche de neurones et celle qui la suit.

Le corps d'un neurone est relié d'une part à un ensemble de dendrites (entrées des neurones) et d'autre part à un axone, partie étirée de la cellule, qui représentera pour nous sa sortie ; une illustration est présentée à la figure 3.1. Le neurone étudié est

connecté aux neurones qui l'environnent : il reçoit au niveau de ses dendrites les signaux électriques des neurones "en amont", propagés par les axones de ces derniers. Les charges électriques s'accumulent dans le neurone jusqu'à dépasser un certain seuil : à ce moment la transmission du signal électrique se déclenche via son axone vers d'autres neurones "en aval.

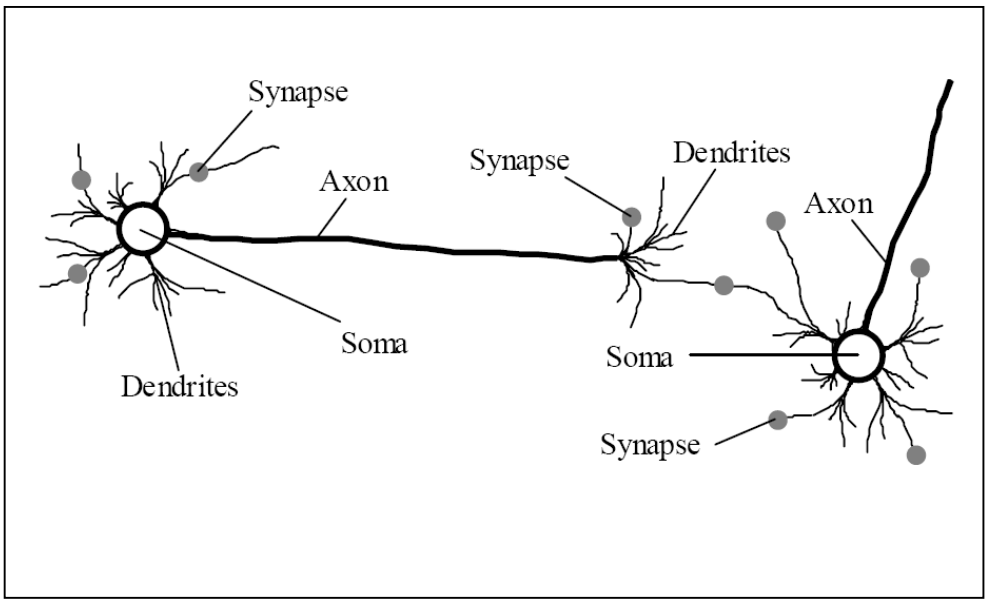


Figure 3.1 : Morphologie d'un neurone biologique.

On remarque que les liaisons axone/dendrite entre deux neurones (connexions synaptiques) ne sont pas toutes de la même efficacité. Ainsi l'entrée associée à une certaine dendrite du neurone pourra avoir plus d'influence qu'une autre sur la valeur de sortie. On peut représenter la qualité de la liaison par un *poids*, sorte de coefficient s'appliquant au signal d'entrée. Le poids sera d'autant plus grand que la liaison est bonne. Un poids négatif aura tendance à inhiber une entrée, tandis qu'un poids positif viendra l'accentuer.

Les chercheurs ont essayé de modéliser les neurones biologiques et de mettre au point le neurone formel. C'est donc une modélisation mathématique qui reprend les grands principes du fonctionnement du neurone biologique.

3.3.2. Le neurone formel

En 1943, McCulloch et Pitts [95] ont implémenté un système de réseaux neuronaux artificiels, qui est analogue aux neurones biologiques fondé sur une structure complexe (tableau 3.1). Le système des RNA est considéré comme un arrangement d'éléments de structure identique appelés neurones interconnectées par analogie avec cellules du système nerveux humain. Il est composé également d'une succession de couches connectées de manière à ce que chaque neurone tienne son entrée de la sortie du neurone précédant. Dans ce cas, chaque neurone fonctionne indépendamment par rapport aux autres afin que l'ensemble forme un système compact. L'information est emmagasinée de façon répartie dans le réseau sous forme de coefficients synaptiques. Le neurone formel calcule régulièrement un résultat qu'il transmet ensuite aux neurones suivant, chaque calcul est associé à un poids qui définit la force de la connexion.

Neurone artificiel	Neurone biologique
Poids de connexion	Synapses
Signal de sortie	Axones
Signal d'entré	Dendrite
Fonction d'activation	Soma

Tableau 3. 1 : Analogie entre le neurone biologique et le neurone formel

Mathématiquement, tel qu'illustré par la figure 3.2, chaque neurone reçoit des entrées sous forme vectorielle, puis il calcule une somme pondérée de ses entrées pour que le résultat passe ensuite par la fonction d'activation afin de créer une sortie.

Il est donc tout naturel d'assimiler un neurone à un triplet ($\overrightarrow{poids}$, *biais*, fonction d'activation Ψ) :

- on multiplie chaque valeur d'entrée par la composante des $\overrightarrow{poids}$ correspondante ce qui revient à faire le produit scalaire $\overrightarrow{entrées}.\overrightarrow{poids}$,
- on compare la valeur obtenue à une valeur de référence : le biais (b), ce qui revient à soustraire le scalaire biais,

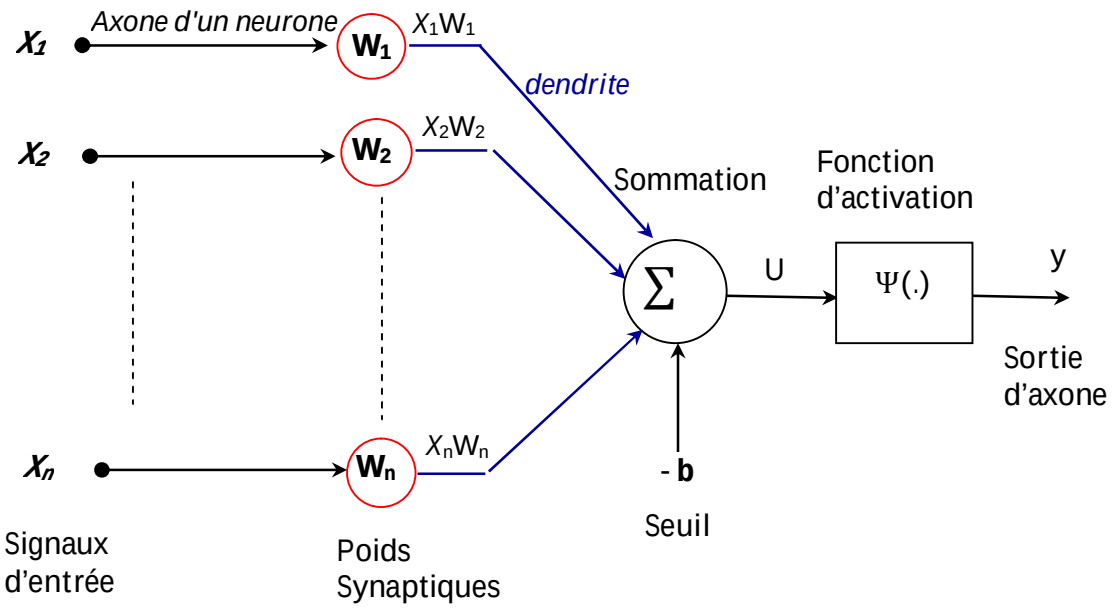


Figure 3. 2 : Modèle du neurone formel

Enfin, on applique la fonction d'activation à cette différence ; la fonction d'activation est souvent de façon à avoir une sortie comprise entre 0 et 1. Par exemple dans le cas d'une fonction d'activation de type seuil, la sortie sera :

$$y = \begin{cases} 1 \;\; si \;\; \overrightarrow{entrées}.\overrightarrow{poids} - biais > 0 \quad état\ actif \\ 0\ si \;\; \overrightarrow{entrées}.\overrightarrow{poids} - biais \leq 0 \quad état\ inactif \end{cases}$$

La valeur de sortie d'un neurone en fonction de ses entrées ($\mathbf{X} = [x_1, \ldots, x_n]$) devient donc ; avec $\mathbf{W} = [w_1, \ldots, w_n]$ le vecteur de poids associés à chacune des entrées et b le biais, ou seuil, du neurone artificiel:

$$y(\mathbf{X}) = \mathbf{\Psi}\left(\sum_{\mathbf{i=1}}^{\mathbf{n}} (w_i \times x_i) + b\right) \qquad (3.1)$$

Où Ψ représente la fonction d'activation.

3.3.3 Fonctions d'activation

La fonction d'activation (ou fonction de transfert) est la fonction mathématique qui permet de traiter l'information qui arrive à un neurone artificiel en machine learning, comme le fait ceux du cerveau avec les signaux électriques qu'ils reçoivent. Elle sert à

convertir le résultat de la somme pondérée des entrées d'un neurone en une valeur de sortie, cette conversion s'effectue par un calcul de l'état du neurone en introduisant une non-linéarité dans le fonctionnement du neurone [26]. Généralement, les fonctions d'activation ont un effet "d'aplatissement".

Le biais b joue un rôle de seuil, quand le résultat de la somme pondérée dépasse ce seuil, l'argument de la fonction de transfert devient positif ou nul; dans le cas contraire, il est considéré négatif. Finalement, si le résultat de la somme pondérée est:

- en dessous du seuil, le neurone est considéré comme non-actif;
- aux alentours du seuil, le neurone est considéré en phase de transition;
- au-dessus du seuil, le neurone est considéré comme actif.

Il y a plusieurs types de fonctions de transfert qui peuvent être utilisées dans les RNA, les fonctions d'activation souvent utilisées sont représentées ci-dessous:

3.3.3 .1 La fonction non linéaire sigmoïde

Aux débuts des réseaux de neurones artificiels l'une des premières fonctions d'activation a été la fonction sigmoïde aussi connue sous le nom de fonction logistique. Elle prend une valeur réelle en entrée et la transforme en une valeur réelle de sortie comprise entre 0 et 1. La fonction sigmoïde (figure 3.3) est définie par l'équation suivante:

$$\Psi(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

Il y a quelques inconvénients quant à utiliser la fonction logistique comme fonction d'activation.

- Elle sature à ses extrémités à 0 ou à 1 et donc le gradient (dérivée) en ces zones est nul. A cause de cette limitation, le modèle peut souffrir de ce que l'on appelle le *vanishing gradient,* qui signifie en d'autres termes que le gradient de la fonction coût diminue progressivement (peut même devenir nul) et l'apprentissage devient moins efficace car les paramètres ne sont plus mises à jour correctement.

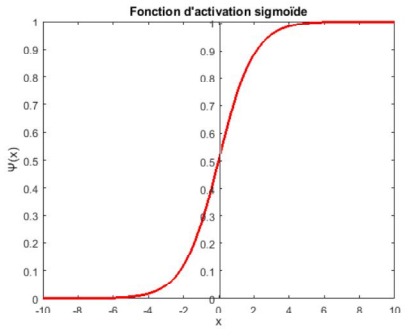


Figure 3.3 : Fonction d'activation sigmoïde

- Bien que à priori pas "primordial", les valeurs de sortie de la fonction sigmoïde ne sont pas centrées en 0. Le centrage à zéro des données de sortie des neurones est une propriété appréciable, car elles deviennent elles même centrées à zéro (moyenne nulle).

3.3.3 .2 Tangente hyperbolique

Une autre fonction d'activation qui fonctionne presque toujours mieux que la fonction logistique est la fonction tangente hyperbolique. Elle transforme toute entrée réelle en une valeur comprise entre -1 et 1. Autant ici nous avons un centrage à zéro, autant nous avons toujours le problème du vanishing gradient, car elle aussi, plafonne à ses extrémités et donc son gradient (dérivée) devient nul. Mais elle reste néanmoins très utilisée. La fonction Tangente hyperbolique (figure 3.4) est définie par :

$$\Psi(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1} = 2 \times sigmoid(2x) - 1 \qquad (3.3)$$

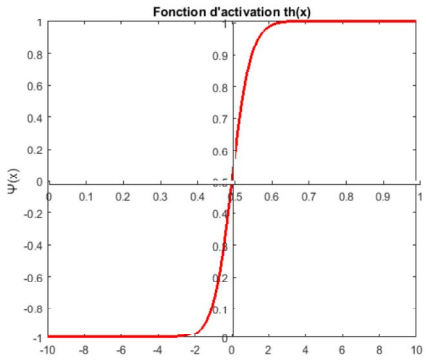


Figure 3. 4 : La fonction Tangente hyperbolique

3.3.3 .3 Unité linéaire rectifiée (ReLU)

La fonction d'activation la plus utilisée de nos jours est la ReLU - Rectified Linear Unit - définie par l'équation (3.4) et la figure 3.5 qui est une fonction linéaire par

morceaux. Son avantage réside dans le fait qu'elle remplace toute valeur d'entrée négative par 0 et toute valeur positive par elle même. Et pour son gradient, il devient nul pour les valeurs négatives et vaut 1 pour les valeurs positives. Cette propriété est très intéressante dans la phase d'apprentissage car elle permet d'éviter et de corriger le problème du vanishing gradient, mais sa principale limitation est que certains gradients peuvent être fragiles et donc s'annuler et cela est du à sa valeur nulle pour des valeurs négatives. Pour combler cela, une version modifiée du ReLU, le Leaky ReLU est proposée.

$$\Psi(x) = \max(0, x) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases} \tag{3.4}$$

La fonction Leaky ReLU est interprétée par l'équation (3.5). La fonction Leaky Relu essaye de corriger la fonction ReLU lorsque l'entrée est négative. Le concept de Leaky ReLU est lorsque l'entrée est négative, il aura une petite pente positive de 0,1. Cette fonction élimine quelque peu le problème d'inactivité de la fonction ReLU pour les valeurs négatives, mais les résultats obtenus avec elle ne sont pas cohérents. Elle conserve tout de même les caractéristiques d'une fonction d'activation ReLU, c'est-à-dire efficace sur le plan des calculs, elle converge beaucoup plus rapidement et ne sature pas dans les régions positives.

$$\Psi(x) = \max(0.1 * x, x) = \begin{cases} 0.1 * x & si \; x < 0 \\ x & si \; x \geq 0 \end{cases} \tag{3.5}$$

La valeur 0.1 a été choisie arbitrairement, elle peut parfois être utilisée comme paramètre du neurone.

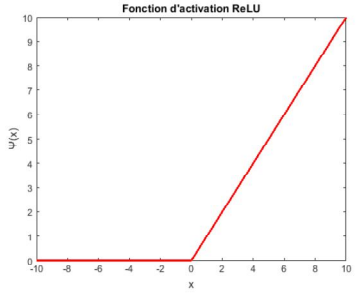


Figure 3.5 : Fonction d'activation ReLU

L'idée de la fonction Leaky ReLU peut être encore élargie. Au lieu de multiplier ***x*** par un terme constant, nous pouvons le multiplier par un hyperparamètre qui semble mieux

fonctionner que la fonction Leaky ReLU. Cette extension à la fonction Leaky ReLU est connue sous le nom de ReLU paramétrique.

La fonction ReLU paramétrique est interprétée par la formule :

$$\Psi(x) = \max(a * x, x) = \begin{cases} a * x & si\ x < 0 \\ x & si\ x \geq 0 \end{cases} \quad (3.6)$$

où "a" est un hyperparamètre. Cela donne aux neurones la possibilité de choisir quelle est la meilleure pente dans la région négative. Avec cette capacité, la fonction ReLU paramétrique a la possibilité de devenir une fonction ReLU classique ou une fonction Leaky ReLU.

3.3.3 .4 La fonction à seuil

On peut citer comme exemple la fonction Heaviside et la fonction Signe qui sont définies respectivement comme suit :

Fonction Heaviside

La fonction de Heaviside (ou fonction échelon unité), est une fonction discontinue prenant la valeur 0 pour tous les réels strictement négatifs et la valeur 1 partout ailleurs :

$$\Psi(x) = \begin{cases} 1 & si\ x \geq 0 \\ 0 & sinon \end{cases} \quad (3.7)$$

Cette fonction est représentée sur la figure 3.6.

Fonction Signe : est utilisée lorsque l'on souhaite modéliser une invariance par changement de signe (comme par exemple en reconnaissance d'objets, pour approcher une invariance en fonction des conditions d'illumination).

La fonction signe représentée sur la figure 3.7 est interprétée par la formule :

$$\Psi(x) = sgn(x) = \begin{cases} +1 & si\ x \geq 0 \\ -1 & si\ x < 0 \end{cases} \quad (3.8)$$

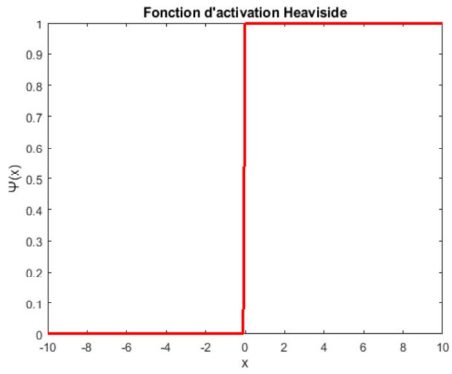


Figure 3.6 : Fonction Heaviside

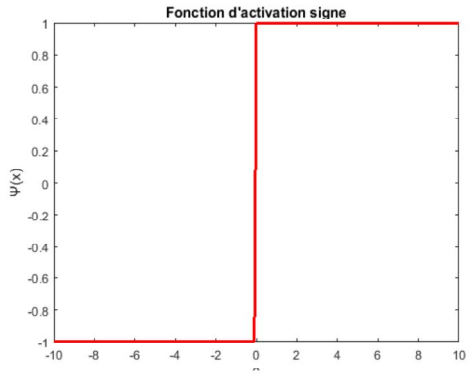


Figure 3.7 : Fonction signe

3.3.3 .5 La fonction non linéaire

On peut la définir comme suit (figure 3.8) :

$$\Psi(x) = \begin{cases} x & si\ u < x < v \\ v & si\ x \geq v \\ u & si\ x \leq u \end{cases} \tag{3.9}$$

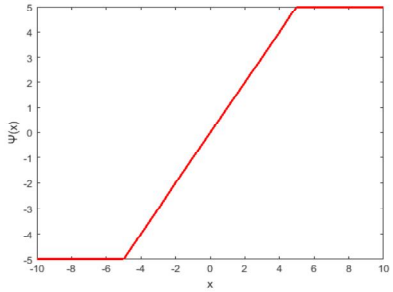


Figure 3.8 : Fonction non linéaire à seuil

3.3.3 .6 Fonction d'activation de sortie Softmax

La fonction d'activation de softmax est généralement utilisée dans la dernière couche du réseau. Il s'agit d'une généralisation de la sigmoïde, qui peut aussi s'écrire

$$softmax(u_k) = \Psi_k(u_k) = \frac{e^{u_k}}{\sum_{l=1}^{K} e^{u_l}} \tag{3.10}$$

Dans le cas d'un problème de classification multi-classe figure 3.9, nous modifions l'architecture du perceptron. Au lieu d'utiliser une seule unité de sortie, il va en utiliser autant que de classes. Chacune de ces unités sera connectée à toutes les unités d'entrée. On aura donc K.(n+1) poids de connexion, où K est le nombre de classes.

Si la sortie pour la classe k est suffisamment plus grande que celles des autres classes, son activation sera proche de 1 tandis que l'activation des autres sera proche de 0. On peut donc aussi considérer qu'il s'agit d'une version différentiable du maximum, ce qui nous aidera grandement pour l'apprentissage.

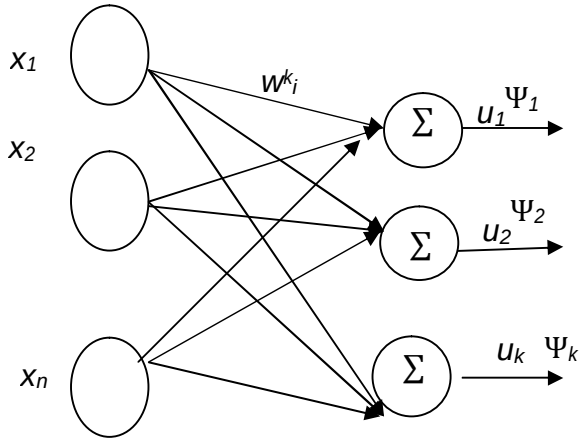


Figure 3.9 : Perceptron multi-classe.

3.4 Architecture des réseaux de neurones

L'architecture est un concept très important qui joue un rôle important dans la classification des réseaux de neurones artificiels (RNA). Chaque architecture a sa propre organisation qui est adaptée à des applications bien spécifiques.

3.4.1 Les réseaux récurrents« FEED-BACK »

Les réseaux récurrents *(ou RNN pour Recurrent Neural Networks)* sont des réseaux de neurones dans lesquels l'information peut se propager dans les deux sens, y compris des couches profondes aux premières couches. Les architectures les plus utilisées sont :

3.4.1.1 Les réseaux de Kohonen

Inspiré de l'organisation du cortex, les réseaux auto-organisés se distinguent par une connectivité locale. Ils sont surtout adaptés pour le traitement d'informations spatiales. Le modèle le plus connu de ce type est la carte auto-organisatrices de Kohonen appelée aussi carte auto-adaptative ou SOM pour *self organizing map*.

Ces réseaux utilisent des méthodes d'apprentissage non-supervisées. Ils peuvent être utilisés pour cartographier un espace réel ou encore étudier la répartition de données dans un espace de grandes dimensions comme dans le cas de problème de quantification vectorielle ou de classification.

3.4.1.2 Les réseaux de Hopfield

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés. Dans ce type de réseau, chaque neurone est connecté à chaque autre neurone et il n'y a aucune différenciation entre les neurones d'entrée et de sortie. Ils fonctionnent comme une mémoire associative non-linéaire et sont capables de trouver un objet stocké en fonction de représentations partielles ou bruitées. L'application principale des réseaux de Hopfield est la résolution de problèmes d'optimisation. Le mode d'apprentissage utilisé est le mode non-supervisé.

La Figure 3.10 présente le modèle général de Hopfield. Il n'y a pas de distinction entre neurone d'entrée et neurone de sortie ; seule importe la matrice des poids. Les neurones sont en effet reliés deux à deux par des connexions symétriques. Ces connexions supportent les motifs de connaissance que le réseau apprend. Les neurones sont représentés par les cercles blancs et les liaisons synaptiques par les traits pleins.

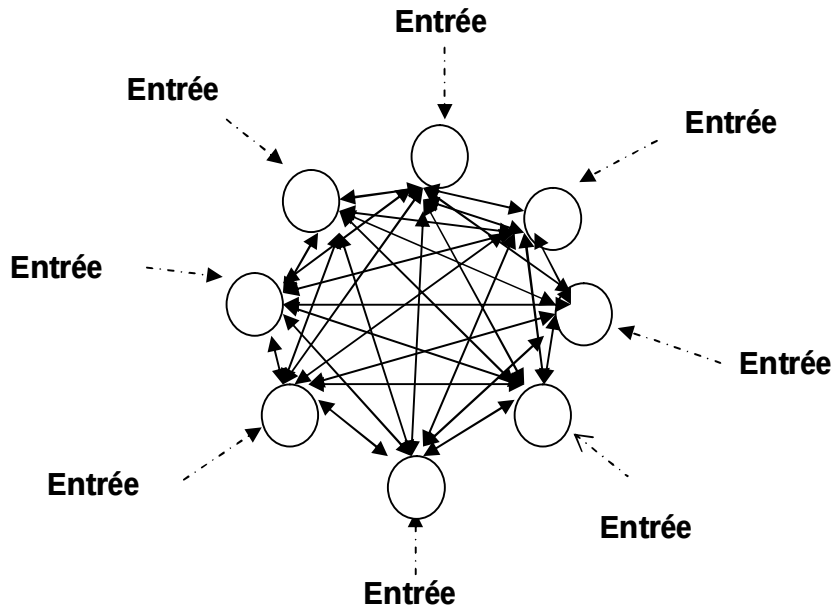


Figure 3.10 : Réseau de Hopfield complètement connecté à connexions symétriques

Le modèle de Hopfield présente quelques inconvénients. En premier lieu, un tel réseau n'est capable d'activer qu'un seul motif simultanément. Si des caractéristiques appartenant à deux motifs ne possédant aucun neurone commun sont activées, le réseau n'en reconnaîtra qu'un seul, selon un choix basé aléatoirement sur le temps de réponse et de mise à jour des neurones. Ensuite, les motifs doivent être appris simultanément. Un apprentissage incrémental donne des résultats chaotiques durant la phase de rappel des connaissances.

3.4.1.3 Les ART(Adaptative Resonnance Theorie)

C'est un modèle de réseau de neurones à architecture évolutive. Dans la plupart des réseaux de neurones, il existe deux étapes :

- La première est la phase d'apprentissage : les poids des connexions sont modifiés selon une règle d'apprentissage.
- La deuxième est la phase d'exécution (test) où les poids ne sont plus modifiés.

Avec le réseau ART, ces deux étapes sont réalisées simultanément. Le réseau en phase de test, s'adapte à des entrées inconnues en construisant de nouvelles classes (ajout de neurones) tout en dégradant au minimum les informations déjà mémorisées. Il existe plusieurs versions de réseaux (ART1, ART2, ART3).

3.4.1.4 Réseaux récurrents à longue mémoire à court terme (LSTM)

L'avantage des RNN réside dans leur capacité à prendre en compte le contexte passé lors de l'apprentissage. Toutefois, même si en théorie cette propriété les rend particulièrement adaptés au traitement des séquences, en pratique les RNNs classiques sont incapables de traiter des séquences faisant intervenir des écarts temporels supérieurs à 10 instants entre les entrées et les sorties désirées correspondantes [14].

En effet, avec des calculs cumulés sur le long terme, l'erreur obtenue avec la rétropropagation du gradient décroît ou, moins fréquemment, augmente d'une manière exponentielle par rapport à l'échelle du temps. Ces deux problèmes sont nommés respectivement la « dissipation du gradient » (vanishing gradient) et « l'explosion du gradient » (exploding gradient) [14,55]. Nous notons également que ce type de

problèmes existait aussi dans les architectures non bouclées profondes. La dissipation ou l'explosion du gradient s'aggrave dans ce cas en fonction du nombre de couches.

Plusieurs solutions ont été proposées pour remédier au problème de la décroissance exponentielle de l'erreur [8, 84], mais la plus utilisée dans l'état de l'art est celle des réseaux récurrents à longue mémoire à court terme (LSTM pour long short-term memory) [56].

L'architecture LSTM consiste en un ensemble de sous-réseaux récurrents particuliers (appelés "blocs de mémoire") situés au niveau de la couche cachée (Figure 3.11), et contenant chacun une ou plusieurs "cellules de mémoire"(Figure 3.12). Cette architecture est définie par deux idées clés que nous allons présenter dans ce qui suit.

La première idée clé architecturale des réseaux LSTM est l'introduction d'un nœud spécial appelé CEC (pour Constant Error Carousel) qui possède une connexion autorécurrente avec un poids constant égal à 1, 0. Ce nœud assure la rétro-propagation d'une erreur constante dans le temps en l'absence de "nouvelles" entrées. Ceci permet donc de résoudre en partie le problème de la décroissance exponentielle de l'erreur. De manière plus intuitive, le CEC peut être vu comme une unité qui permet de "collecter" et de "conserver" les informations jugées pertinentes tout au long de la séquence, et de les "présenter" au reste du réseau [4].

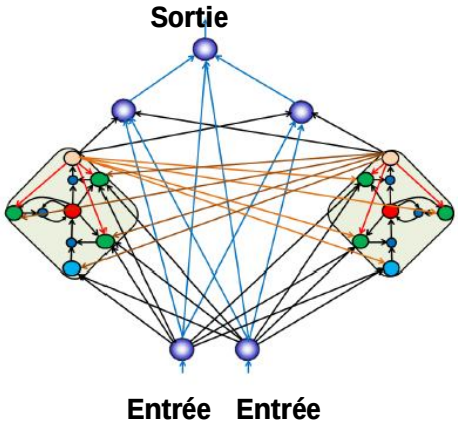


Figure 3.11: Réseau de longue mémoire à court terme. Ce réseau a deux blocs mémoires d'une cellule mémoires chacun.

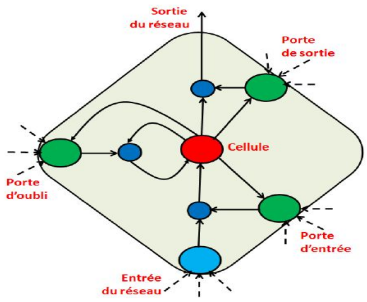


Figure 3.12: Exemple d'un bloc de mémoire. Les cellules bleues représentent les unités multiplicatives.

La seconde idée clé est l'utilisation de "portes" multiplicatives qui sont des fonctions d'activation dont la sortie est un coefficient multiplicatif permettant d'ouvrir ou de fermer une connexion donnée. Dans la première version de l'architecture LSTM introduite par Hochreiter et *al* [56], chaque bloc de mémoire comportait deux portes multiplicatives, une pour l'entrée et une pour la sortie. Le rôle de ces portes est, d'une part, protéger le contenu du CEC des activations provenant des nouvelles entrées (pour le cas des portes d'entrée), et d'autre part, protéger le reste du réseau du contenu du CEC si celui-ci n'est pas pertinent (pour le cas des portes de sortie). Typiquement, tant que la porte d'entrée reste fermée (c'est-à-dire que son activation est proche de 0), l'activation du CEC sera gardée constante et ne sera pas mise à jour en fonction des nouvelles activations arrivant à l'entrée du bloc de mémoire. De même, tant que le contenu du CEC est jugé pertinent pour le reste du réseau, la porte de sortie sera maintenue ouverte. L'ouverture et la fermeture des portes sont apprises automatiquement à partir des données d'apprentissage [4].

Les LSTM ont montré leur efficacité dans divers domaines d'application. Ils sont considérés actuellement comme l'approche état-de-l'art dans plusieurs tâches traitant des données séquentielles ainsi que dans les séquences d'événements assez longues [24, 35, 82,135, 147].

3.4.2 Réseaux propagation vers l'avant« FEED-FORWARD »

Appelés aussi "réseaux de type Perceptron", ce sont des réseaux dans lesquels l'information se propage de couche en couche sans retour en arrière. Ce genre de réseaux utilise un apprentissage supervisé, par correction des erreurs ou le signal d'erreur est rétro-propagé vers les entrées afin de mettre à jour les poids des neurones.

3.4.2.1 Le perceptron monocouche

C'est historiquement le premier RNA, c'est le Perceptron de Rosenblatt. C'est un réseau simple, puisque il ne se compose que d'une couche d'entrée et d'une couche de sortie (figure 3.13). Il est inspiré du système visuel et de ce fait a été conçu dans un but premier de reconnaissance des formes. Cependant, il peut aussi être utilisé pour faire de la classification et pour résoudre des opérations logiques simples (telle "ET" ou "OU"). Les sorties des neurones ne peuvent prendre que deux états (-1 et 1 ou 0 et 1). Seuls les poids des liaisons entre la couche d'association et la couche finale peuvent être modifiés. Il suit généralement un apprentissage supervisé selon la règle de correction de l'erreur (ou selon la règle de Hebb) : si la sortie du réseau (donc celle d'une cellule de décision) est égale à la sortie désirée, le poids de la connexion entre ce neurone et le neurone d'association qui lui est connecté n'est pas modifié.

Dans le cas contraire, le poids est modifié proportionnellement. À la différence entre la sortie obtenue et la sortie désirée : **w <= w + k (d - s)**, Où **s** est la sortie obtenue, **d** la sortie désirée et **k** une constante positive.

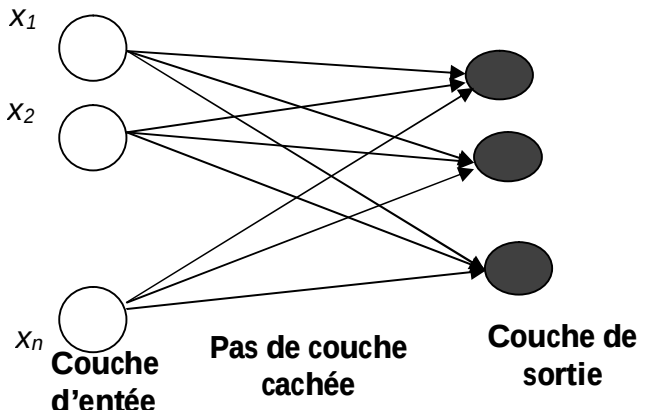


Figure 3.13 : Le réseau monocouche

3.4.2.2 Le perceptron multicouches «PMC»

C'est une extension du perceptron monocouche, avec une ou plusieurs couches cachées entre l'entrée et la sortie (figure 3.14).

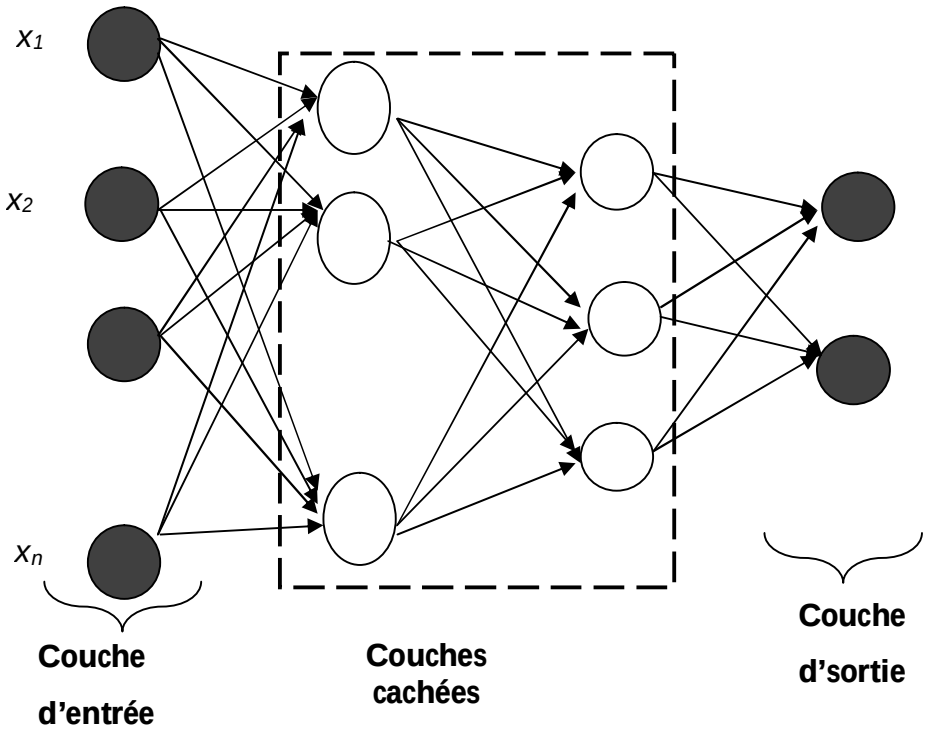


Figure 3.14 : Architecture d'un réseau multicouche

Chaque neurone, dans une couche, est connecté à tous les neurones de la couche précédente et de la couche suivante (excepté pour les couches d'entrée et de sortie), et il n'y a pas de connexions entre les cellules d'une même couche.

Le choix du nombre de couches cachées dépend généralement de la complexité du problème à résoudre. En théorie, une seule couche cachée peut être suffisante pour résoudre n'importe quel problème donné, mais il se peut que le fait de disposer de plusieurs couches cachées permette de résoudre plus facilement n'importe quel problème complexe.

Les PMC utilisent, pour modifier leurs poids, un algorithme d'apprentissage, il existe une centaine mais le plus populaire est la rétropropagation du gradient, qui est une généralisation de la règle de Widrow-Hoff. Il s'agit toujours de minimiser l'erreur

quadratique, on propage la modification des poids de la couche de sortie jusqu'à la couche d'entrée, donc cet algorithme passe par deux phases :

- Les entrées sont propagées de couche en couche jusqu'à la couche de sortie.
- Si la sortie du PMC est différente de la sortie désirée alors l'erreur est propagée de la couche de sortie vers la couche d'entrée en modifiant les poids durant cette propagation.

3.4.2.3 Réseaux à fonction radiale« RBF »

Le réseau RBF fait partie des réseaux de neurones supervisés. Il est constitué de trois couches (figure 3.15): une couche d'entrée qui retransmet les entrées sans distorsion, une seule couche cachée qui contient les neurones RBF qui sont généralement des fonctions gaussiennes et une couche de sortie dont les neurones sont généralement animés par une fonction d'activation linéaire. Chaque couche est complètement connectée à la suivante et il n'y a pas de connexions à l'intérieur d'une même couche.

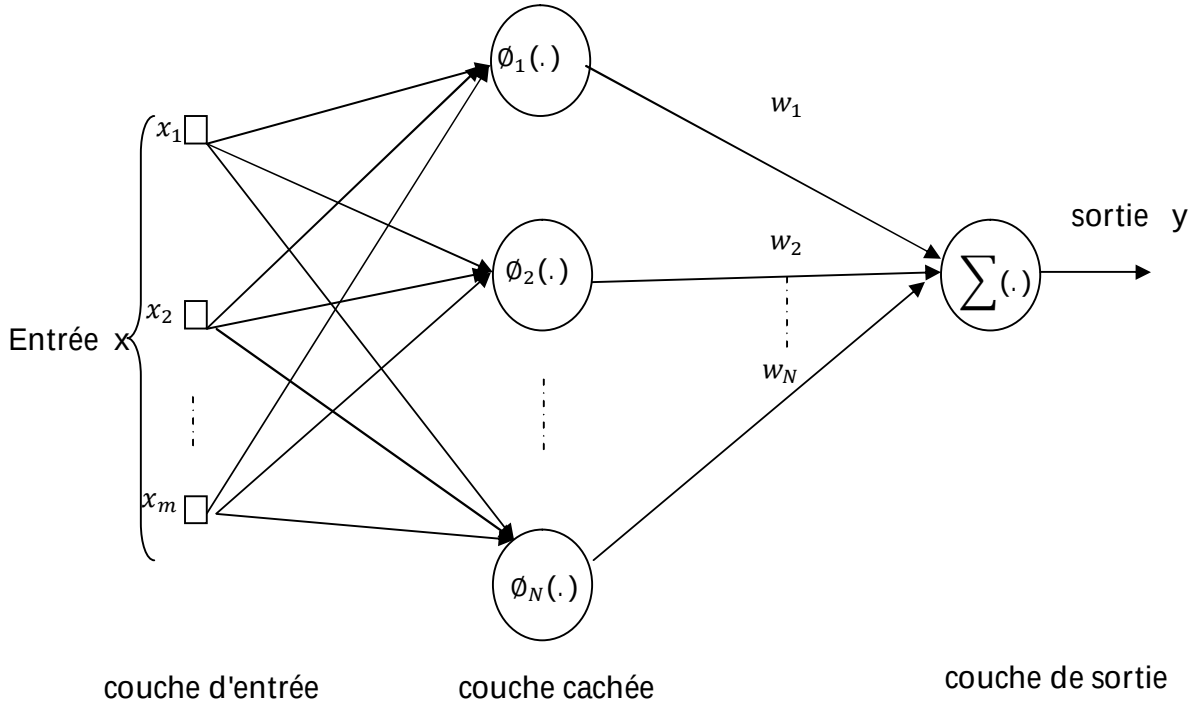


Figure 3.15: Architecture d'un réseau RBF

Comme un PMC, un RBF peut être utilisé dans la prédiction, l'identification et la classification, mais les réseaux RBF diffèrent des réseaux PMC, du fait que les fonctions d'activation des nœuds de la couche cachée sont des fonctions gaussiennes. Le modèle de réseau RBF est caractérisé par quatre paramètres principaux, qui doivent être réglés, lors de l'étape de construction du réseau. Toute modification d'un de ces paramètres entraîne directement un changement du comportement du réseau. Ces paramètres sont :

- Le nombre de neurones RBF dans l'unique couche cachée ou le nombre des gaussiennes;
- La position des centres des gaussiennes de chacun des neurones;
- La largeur de ces gaussiennes;
- Le poids des connexions entre les neurones RBF et le(s) neurone(s) de sortie.

Une fonction radiale de base (RBF) est une fonction symétrique autour d'un centre μ_i :

$$\emptyset_i(x) = \emptyset(\|x - \mu_i\|, \sigma_i) \tag{3.11}$$

où:

$\|.\|$ représente la norme euclidienne;

σ_i représente la largeur de la fonction.

La sortie d'un réseau RBF calcule une combinaison linéaire des fonctions radiales de la couche cachée. Dans le cas de fonctions gaussiennes, la sortie est définie par :

$$y(x) = \sum_{i=1}^{N} w_i e^{\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}\right)} \tag{3.12}$$

L'utilisation d'un modèle RBF nécessite la détermination de son architecture, c.à.d. le nombre *N* de fonctions radiales $\emptyset_i$, la fixation des paramètres de ces fonctions,μ_i et σ_i, et la détermination des poids des connexions w_i vers la couche de sortie.

3.5 Choix de réseau de neurones artificiels

Il existe plusieurs modèles de réseau de neurones artificiels. Chaque modèle est caractérisé par son architecture, son traitement et sa règle d'apprentissage. Pour choisir

le modèle le plus adapté à une application définie, il faut prendre en compte différents paramètres, parmi lesquels nous citons :

- La fonction désirée (classification, prédiction, diagnostic ou reconnaissance) ;
- La nature des données à traiter. Ces données peuvent être de nature dynamique, statique ou aléatoire et peuvent avoir différentes formes ;
- Ressources matérielles et/ou logicielles disponibles pour l'implémentation du réseau;
- Contraintes temporelles généralement liées à des applications temps réel;
- Les efforts de préparation de la base d'apprentissage ainsi que de la base de tests et validation en cas de besoin;
- Délais d'apprentissage correspondant au temps nécessaire avant de considérer le réseau comme expert et commencer la décision.

3.6 L'apprentissage des réseaux de neurones

Une fois que nous avons choisi l'architecture du réseau de neurones, c'est-à-dire le type de réseau de neurones et les fonctions d'activation, nous devons déterminer les autres paramètres ajustables du modèle que sont les poids qui permettent de connecter les entrées aux neurones cachées et les neurones cachés aux neurones de sortie. Le processus d'ajustement de ces paramètres de telle sorte que le réseau soit en mesure d'approcher la relation fonctionnelle sous-jacente entre les entrées et les cibles est connu sous le nom d'apprentissage.

L'apprentissage, pour les réseaux de neurones, consiste à calculer les paramètres de telle manière que les sorties du réseau de neurones soient, pour les exemples utilisés lors de l'apprentissage, aussi proches que possible des sorties « désirées », qui peuvent être le code de la classe à laquelle appartient la forme que l'on veut classer, la valeur de la fonction que l'on veut approcher ou celle de la sortie du processus que l'on veut modéliser, ou encore la sortie souhaitée du processus à commander. Les techniques d'apprentissage des réseaux de neurones sont des algorithmes d'optimisation : ils cherchent à minimiser l'écart entre les réponses réelles du réseau et les réponses désirées, en modifiant les paramètres par étapes (appelées « itérations ») successives. La

sortie du réseau de neurones s'adapte de mieux en mieux aux données au fur et à mesure que l'apprentissage se déroule.

3.6.1 Définitions et concepts de base

La fonction principale des algorithmes d'apprentissage automatique consiste à apprendre des paramètres θ pour modéliser un jeu de données $\mathcal{D}$ contenant *N* exemples, et représentatif du problème que l'on cherche à résoudre. Chacun de ces algorithmes possède également des hyper-paramètres qui contrôlent leur capacité, c'est-à-dire leur performance à apprendre des paramètres pour modéliser $\mathcal{D}$ mais aussi à généraliser cet apprentissage sur un nouveau jeu de données. Chacun des exemples de $\mathcal{D}$ sera dénoté par $x^{(i)} \in \mathbb{R}^D$, où D correspond au nombre de dimensions de l'exemple et *i* est compris entre [1,…,N] exemples. Des cibles peuvent également être présentes pour étiqueter le jeu de données $\mathcal{D}$. Chaque exemple $x^{(i)}$ peut donc être relié à sa cible correspondante $y^{(i)}$, pour former $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$.

3.6.1.1 Mode d'apprentissage

Les algorithmes d'apprentissage automatique apprennent selon des objectifs particuliers qui peuvent être divisés en trois principales catégories : l'apprentissage supervisé, l'apprentissage non-supervisé et l'apprentissage par renforcement. Il existe également un paradigme d'apprentissage semi-supervisé qui mélange les apprentissages supervisé et non-supervisé.

3.6.1.2 Apprentissage supervisé

La formulation du problème de l'apprentissage supervisé est simple : on dispose d'un nombre fini d'exemples d'une tâche à réaliser, sous forme de paires (entrée, sortie désirée), et on souhaite obtenir, d'une manière automatique, un système capable de trouver de façon relativement fiable la sortie correspondante à toute nouvelle entrée qui pourrait lui être présentée [132].

On distingue en général trois types de problèmes auxquels l'apprentissage supervisé est appliqué. Ces tâches diffèrent essentiellement par la nature des paires (entrée, sortie) qui y sont associées :

Classification

Dans les problèmes de classification, l'entrée correspond à une instance d'une classe, et la sortie qui y est associée indique la classe. Si le problème de classification est binaire, alors $y \in \{0, 1\}$. Si le problème de classification est multi classe, alors $y \in \{1, \ldots, L\}$ où L correspond au nombre de classes. Par exemple pour un problème de reconnaissance de visage, l'entrée serait l'image bitmap d'une personne telle que fournie par une caméra, et la sortie indiquerait de quelle personne il s'agit (parmi l'ensemble de personnes que l'on souhaite voir le système reconnaître).

Régression

Dans les problèmes de régression, l'entrée n'est pas associée à une classe, mais dans le cas général, à une ou plusieurs valeurs réelles (un vecteur). Par exemple, pour une expérience de biochimie, on pourrait vouloir prédire le taux de réaction d'un organisme en fonction des taux de différentes substances qui lui sont administrées.

Séries temporelles

Dans les problèmes de séries temporelles, il s'agit typiquement de prédire les valeurs futures d'une certaine quantité connaissant ses valeurs passées ainsi que d'autres informations. Par exemple le rendement d'une action en bourse. Une différence importante avec les problèmes de régression ou de classification est que les données suivent typiquement une distribution non stationnaire.

3.6.1.3 Apprentissage non-supervisé

L'apprentissage non-supervisé, consiste à apprendre sans superviseur. A partir d'une population, il s'agit d'extraire des classes ou groupes d'individus présentant des caractéristiques communes, le nombre et la définition des classes n'étant pas donnés a priori. A l'inverse de l'apprentissage supervisé (**Supervised Learning**) qui tente de

trouver un modèle depuis des données labellisées $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, l'apprentissage non supervisé prend uniquement des données $x^{(i)}$ sans label (pas de variable à prédire $y^{(i)}$).

3.6.1.4 Apprentissage semi-supervisé

L'apprentissage semi supervisé vise à résoudre des problèmes de reconnaissance des formes comportant un petit nombre de données étiquetées pour un grand nombre de points sans labels. Ce type de situation peut se produire quand l'étiquetage des données est coûteux, comme dans le cas de la classification de pages internet. La question qui se pose est alors de savoir si la seule connaissance des points avec labels est suffisante pour construire une fonction de décision capable de prédire correctement les étiquettes des points non étiquetés.

3.6.1.5 Apprentissage par renforcement

Dans le cadre de l'apprentissage par renforcement, le but est d'apprendre quelles actions effectuer, étant donné un contexte, afin de maximiser une mesure de récompense qui peut, elle même, dépendre des actions passées. Des prédictions sont donc effectuées pendant l'apprentissage pour prendre des décisions et explorer l'espace des solutions. Cette approche fait intervenir le compromis exploration-exploitation, c'est à dire, soit essayer de nouvelles stratégies pour trouver une meilleure solution, quitte à faire des erreurs, soit maximiser la récompense avec la solution la plus performante actuellement apprise .

3.6.2 La généralisation

3.6.2.1 Le problème

Pour formaliser le problème, nous devons tout d'abord choisir une modélisation mathématique de la tâche que nous voulons automatiser. Cette dernière va définir l'espace des fonctions $\mathcal{F}$ dans lequel nous allons chercher une solution. Nous devons aussi choisir une fonction de coût (ou objectif) C évaluant par un réel la performance d'une solution $f \in \mathcal{F}$ pour un échantillon des données. Si le coût est élevé, la

performance est faible, et inversement. Si D est la variable aléatoire correspondant à l'ensemble des données relatives à la tâche en considération, alors le but ultime de l'apprentissage machine est de trouver une fonction f^* telle que:

$$f^* = \text{argmin}_{f \in \mathcal{F}} \mathbb{E}_D [\text{C}(D, f)] \tag{3.13}$$

C'est à dire une fonction qui minimise le coût en généralisation. Donc, si l'on a accès à l'ensemble de toutes les données possibles, le problème d'apprentissage automatique se réduit au problème d'optimisation ci-dessus. Cela est rarement le cas: les données sont la plupart du temps limitées à un ensemble fini $\mathcal{D}$.

3.6.2.2 Le risque empirique et le sur-apprentissage

Envisageons de minimiser le risque empirique, c'est-à -dire la moyenne de la fonction de coût sur $\mathcal{D}$, un ensemble fini de données.

$$f' = \text{argmin}_{f \in \mathcal{F}} \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} [\text{C}(d, f)] \tag{3.14}$$

Où $|\mathcal{D}|$ est le cardinal de $\mathcal{D}$. En élargissant l'ensemble $\mathcal{F}$ de manière appropriée, on peut remarquer qu'il est possible d'atteindre un risque empirique arbitrairement proche de la solution optimale sur $\mathcal{D}$. Généralement, une minimisation excessive du risque empirique sur $\mathcal{D}$ résulte en une augmentation du risque sur des données n'appartenant pas à l'ensemble d'entrainement, ce phénomène est appelé le sur-apprentissage. Or, justement, le but de l'apprentissage machine est de trouver une solution qui généralise sur des exemples que l'on a jamais rencontrés. Il est donc nécessaire d'isoler un sous-ensemble des données que l'on n'utilise pas pendant l'apprentissage afin d'estimer cette performance en généralisation.

3.6.2.3 La régularisation

Pour limiter le sur-apprentissage et donc améliorer les performances en généralisation, on peut limiter la capacité du modèle mathématique, conformément au rasoir d'Ockham [94,151]. Cette procédure s'appelle la régularisation. On peut par

exemple réduire l'espace des solutions $\mathcal{F}$ ou rajouter une pénalité de régularisation $\mathcal{R}$ à la fonction de coût qui dépend du modèle et/ou des données. Dans ce dernier cas, à celle-ci est associée un coefficient λ fixant le poids de la régularisation durant l'optimisation, ce dernier est un hyper paramètre. Si le problème est trop contraint par la régularisation, on assiste, à l'inverse, au phénomène de sous-apprentissage: le modèle a trop peu de capacité pour apprendre la tâche convenablement. Il faut donc trouver un équilibre entre la capacité de l'espace des solutions $\mathcal{F}$, les contraintes apportés au problème d'optimisation par l'ensemble de données et celles apportées par la régularisation. Le manque de données nécessite de régulariser plus et inversement. On doit effectuer une sélection de modèle pour trouver cet équilibre.

3.6.2.4 La sélection de modèle

Pour faire la sélection de modèle et estimer la performance en généralisation, on divise l'ensemble $\mathcal{D}$ en trois sous-ensembles disjoints:

- L'*ensemble d'apprentissage*: $\mathcal{D}_{app}$, servant à la minimisation du risque empirique.

$$f^*_{\mathcal{F},\mathcal{R},\lambda} = arg\min_{f\in\mathcal{F}} \frac{1}{|\mathcal{D}_{app}|}\sum_{d\in\mathcal{D}_{app}}[\mathrm{C}(d,f) + \lambda\mathcal{R}(d,f)] \qquad (3.15)$$

- L'*ensemble de validation*: $\mathcal{D}_{val}$, servant à la sélection de modèle, c'est-`a-dire au choix de l'espace des solutions $\mathcal{F}$, de la pénalité de régularisation $\mathcal{R}$ et des autres hyper-paramètres (que nous incluons ici dans λ) pour ajuster le compromis sur-apprentissage/sous-apprentissage.

$$(\mathcal{F}^*,\mathcal{R}^*,\lambda^*) = \mathrm{argmin}_{(\mathcal{F},\mathcal{R},\lambda)} \frac{1}{|\mathcal{D}_{val}|}\sum_{d\in\mathcal{D}_{val}}\left[\mathrm{C}(d, f^*_{\mathcal{F},\mathcal{R},\lambda})\right] \qquad (3.16)$$

- L'*ensemble de test*: $\mathcal{D}_{tes}$, servant à estimer la performance en généralisation

$$\mathcal{G} = \frac{1}{\mathcal{D}_{tes}}\sum_{d\in\mathcal{D}_{tes}}[\mathrm{C}(d, f^*_{\mathcal{F}^*},\mathcal{R}^*,\lambda^*)] \qquad (3.17)$$

Pour obtenir un meilleur estimé de la performance en généralisation ainsi qu'une indication de la variance de cet estimateur, on peut effectuer une validation croisée ("cross-validation"). On découpe $\mathcal{D}$ en *K* sous-ensembles de tailles similaires.
On effectue *K* sélections de modèle, où l'on utilise à chaque fois un sous ensemble différent comme ensemble de test. On peut donc estimer la moyenne et la variance de l'erreur de généralisation. A l'extrême on peut fixer $K = |\mathcal{D}|$ (on ne retire qu'un exemple comme ensemble de test à chaque fois).

3.6.3 Le type de modèle

Il existe une grande variété de modèles en apprentissage machine. Nous intéressons dans cette thèse au modèle à fonction de coût convexe / non-convexe.

3.6.3 .1 A fonction de coût convexe / non-convexe

Nous avons vu, d'après l'équation 3.15, que l'apprentissage machine fait intervenir un problème d'optimisation. On veut minimiser une fonction de coût. Il y a deux possibilités. Soit la fonction que l'on veut minimiser est convexe, c'est à dire qu'elle satisfait la relation suivante :

$$\forall (a,b) \in X, \forall t \in]0,1[, f(t \times a + (1-t) \times b) \leq t \times f(a) + (1-t) \times f(b) \quad (3.18)$$

Dans ce cas, nous avons la garantie que tout minimum local est aussi un minium global (et unique si la fonction est strictement convexe). Soit elle ne l'est pas et dans ce cas on n 'a pas de garantie: on peut converger vers des solutions aux performances différentes suivant les conditions (initialisation, algorithme d'optimisation,...etc.). Dans cette thèse on s'intéresse aux méthodes d'apprentissage profond qui sont basées sur les problèmes d'optimisation non convexes. La figure 3.16 illustre ces deux cas de figure.

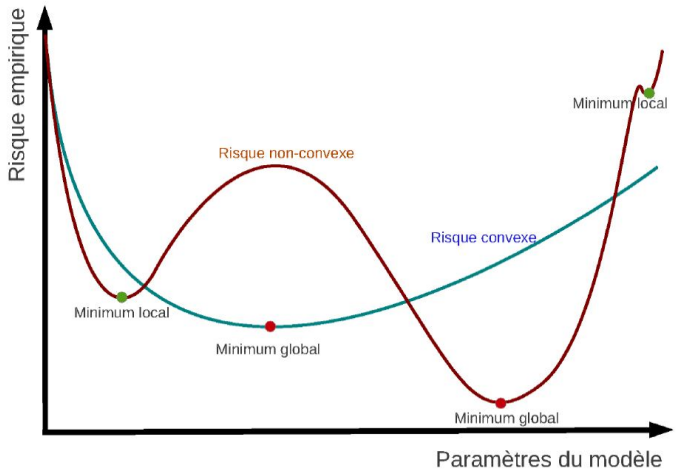


Figure 3.16: Les risques empiriques convexe et non convexe

3.7. Réseaux de neurones multicouches avec rétropropagation

La rétropropagation du gradient est un algorithme d'apprentissage applicable aux réseaux de neurones multicouches non-linéaires. Ce type de réseau est historiquement appelé perceptron. Cette méthode est basée sur les propriétés de classification des neurones non-linéaires, sans les limitations des réseaux mono-couche.

3.7.1 Propriétés d'un neurone à seuil

Un neurone à seuil à 2 entrées peut séparer un plan en 2 parties par une droite, et donc réaliser une classification en 2 classes.

En effet, la sortie de ce neurone est définie par :

$$y = \Psi(w_1 x_1 + w_2 x_2) \tag{3.19}$$

où Ψ est la fonction d'activation seuil. Les vecteurs d'entrée de ce neurone sont de dimension 2 et donc représentent des points dans un plan.

On peut distinguer 2 cas :

$$w_1 x_1 + w_2 x_2 > 0\ (1) \rightarrow y = 1$$
$$w_1 x_1 + w_2 x_2 < 0\ (2) \rightarrow y = 0$$

de (1) $\Leftrightarrow\ x_2 > -\frac{w_1}{w_2} x_1$

$x_2 = -\frac{w_1}{w_2} x_1$ est l'équation d'une droite passant par l'origine et de pente $\frac{-w_1}{w_2}$.

(2) $\Leftrightarrow$ le point est au dessus de la droite. En rajoutant un 3eme lien d'entrée à ce neurone (entrée x_0 et poids w_0), relié à une entrée constante, on peut avoir une droite quelconque

$$y = \Psi(w_0 x_0 + w_1 x_1 + w_2 x_2) \tag{3.20}$$

L'équation de la droite devient alors :

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2} x_0 \tag{3.21}$$

En prenant par exemple $x_0 = 1$ (choix arbitraire), l'ordonnée à l'origine de la droite est $\frac{-w_0}{w_2}$. La pente reste la même.

Un perceptron mono-couche permet donc de réaliser une classification en 2 classes dans le cas d'un problème linéairement séparable, c'est à dire pour lequel tous les exemples d'une classe peuvent être séparés de tous les exemples de l'autre classe par une droite.

3.7.2 Règle d'apprentissage du perceptron

Le but de l'apprentissage est d'obtenir automatiquement les droites de séparation, à partir d'exemples d'entrées/sorties. Les paramètres de ces droites (pente et ordonnée à l'origine) sont codés dans les poids du réseau. L'apprentissage est de type supervisé car il utilise l'information de classe correcte des exemples.

La règle dite d'apprentissage supervisé est définie par :

$$\Delta w_{ij} = \text{taux d'apprentissage} \times \text{erreur de sortie} \times \text{entrée}$$

soit

$$\Delta w_{ij} = \alpha\left(y_j^{(d)} - y_j\right).x_i \tag{3.22}$$

où

- $y_j^{(d)}$ est la valeur désirée (c'est à dire la valeur exacte) de la sortie du neurone j ;
- y_j est la valeur fournie par le neurone ;
- x_i est l'entrée i des neurones.

Elle peut être appliquée à un neurone unique ou plusieurs. L'erreur est locale à chaque neurone.

3.7.3 Apprentissage par descente de gradient

Il s'agit d'une autre méthode d'apprentissage, basée sur la diminution du gradient d'une erreur par rapport aux poids du réseau.

La règle d'apprentissage est définie par :

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{3.23}$$

où :

- α est le taux d'apprentissage, positif et compris entre 0 et 1 (la plupart du temps très inférieur à 1).
- E est l'erreur de sortie : dans le cas de plusieurs neurones et d'un apprentissage incrémental (voir paragraphe suivant), elle est égale à la somme des erreurs individuelles des neurones de sortie :

$$E = \frac{1}{2}\sum_{i=1}^{m}(y_i^{(d)} - y_i)^2 \tag{3.24}$$

Le coefficient 1 / 2 est arbitraire.

- $y_i^{(d)}$ est la sortie désirée du neurone *i* et y_i sa sortie effective. L'apprentissage est du type supervisé.
- *m* est le nombre de neurones.

Le calcul de la dérivée de $\partial E / \partial w_{ij}$ permet de déterminer l'expression de la règle d'apprentissage :

$$\Delta w_{ij} = \alpha(y_j^{(d)} - y_{ji})x_i \tag{3.25}$$

Elle est identique à la règle d'apprentissage du perceptron énoncée dans le paragraphe précédent. Ce qui signifie que réduire l'erreur locale à chaque neurone est équivalent à réduire la somme des erreurs des neurones de sortie.

3.7.4 Mode incrémental et mode "par cycles"

Selon que l'on modifie les poids des neurones après chaque présentation d'une entrée ou après la présentation de tous les exemples de la base d'apprentissage (en fonction de l'erreur cumulée), on est en mode d'apprentissage instantané ou différé.

L'algorithme est un peu différent dans les deux cas.

3.7.4.1 Apprentissage incrémental

Ce mode d'apprentissage est également appelé temps-réel, en ligne, ou instantané.

L'algorithme correspondant est le suivant :

```
Initialisation des poids
Pour chaque cycle d'apprentissage (jusqu'à convergence)
Début
    Pour chaque exemple de la base d'apprentissage
    Début
        Présentation de l'exemple au réseau
        Activation du réseau (=propagation d'activité directe=calcul de sa sortie)
        Calcul de l'erreur instantanée de sortie
        Modification des poids en fonction de l'erreur
    Fin
Fin
```

3.7.4.2 Apprentissage par cycles

Ce mode d'apprentissage est également appelé hors-ligne ou différé (batch).

La présentation de tous les exemples une fois est appelée cycle (epoch).

L'algorithme correspondant est le suivant :

```
Initialisation des poids
Pour chaque cycle d'apprentissage (jusqu'à convergence)
Début
    Pour chaque exemple de la base d'apprentissage
    Début
        Présentation de l'exemple au réseau
        Activation du réseau
        Calcul de l'erreur de sortie et cumul
    Fin
    Modification des poids en fonction de l'erreur cumulée
Fin
```

Dans l'apprentissage en temps différé, la règle d'apprentissage n'est appliquée qu'une fois que tous les vecteurs d'apprentissage ont été présentés au réseau. L'erreur est la valeur moyenne calculée sur tous les exemples de la base d'apprentissage :

$$E = \frac{1}{2F}\sum_{f=1}^{F}\sum_{i=1}^{m}(y_i^{(d,f)} - y_i^{(f)}) \quad (3.26)$$

où *F* est le nombre de formes présentes dans cette base.

3.7.5 Algorithme de rétropropagation du gradient

Les réseaux monocouches ne peuvent traiter que les problèmes linéairement séparables, ce qui n'est pas le cas de la plupart des problèmes réels. Par exemple, le problème pourtant simple du OU-Exclusif n'est pas linéairement séparable. L'extension de l'apprentissage par descente de gradient aux réseaux multi-couches a donc été développée pour traiter les problèmes dans lesquels les classes peuvent avoir des formes quelconques.

3.7.5.1 Principe

La rétropropagation du gradient de l'erreur (ou backpropagation) est un algorithme d'optimisation permettant d'ajuster les paramètres d'un réseau de neurones multicouches pour mettre en correspondance des entrées et des sorties référencées dans une base d'apprentissage.

Pour pouvoir entraîner ces systèmes, il faut savoir comment ajuster les paramètres de chaque couche de neurones. La rétropropagation permet de calculer le gradient de l'erreur pour chaque neurone, de la dernière couche vers la première. Le calcul de ce gradient se fait par la méthode de rétropropagation, pratiquée depuis le milieu des années 80. Cela permet de corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs. Ainsi, les poids synaptiques qui contribuent à engendrer une erreur importante se verront modifiés de manière plus significative que les poids qui ont engendré une erreur marginale. Moyennant quelques précautions lors de l'apprentissage, les procédures d'optimisation finissent par aboutir à une configuration stable, généralement un extremum local, au sein du réseau de neurones.

3.7.5.2 Formules de la rétro-propagation

Comme dans le cas mono-couche, la règle d'apprentissage par descente de gradient est définie par :

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{3.27}$$

Dans le cas des réseaux multicouches, elle peut donner naissance à la règle d'apprentissage dite "delta généralisée" (generalized delta-rule), définie par :

$$\Delta w_{ij}^{(c)} = \alpha . \delta_j^{(c)} . y_i^{(c-1)} \tag{3.28}$$

avec

$\delta_j = e_j . \mathrm{f}'(a_j)$ pour le neurone j de la couche de sortie

$\delta_j^{(c)} = \mathrm{f}'\left(a_j^{(c)}\right) . \sum_k \delta_k^{(c+1)} . w_{jk}^{(c+1)}$ pour le neurone j d'une couche cachée

et

- c est l'indice de la couche courante, $c-1$ l'indice de la couche directement en amont (en général à gauche) et $c+1$ l'indice de la couche directement en aval (en général à droite)
- e_j est l'erreur de sortie au neurone j, définie par : $e_j = y_j^{(d)} - y_j$ ($y_j^{(d)}$ sortie désirée)
- $a_j = \sum_i w_{ij} . y_i$
- $\delta_j^{(c)}$ est appelé gradient local au neurone j (caché ou de sortie).
- f(x) est une fonction non-linéaire pour au moins une des couches, mais ne peut pas être la fonction seuil

Remarque sur les notations : le réseau étant composé de plusieurs couches, les sorties des neurones sont toutes désignées par y ; l'indice c est ajouté pour préciser à quelle couche ces derniers appartiennent.

3.7.5.3 Démonstrations des formules

Le principe de base utilisé dans cet algorithme est la descente de gradient, qui s'exprime par :

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$$

Le principe de calcul pour obtenir ces formules est le suivant : puisque ce que l'on cherche est la dérivée de l'erreur de sortie E par rapport à w_{ij}, il faut faire apparaître la dépendance entre eux. En mode d'apprentissage en temps-réel (ou instantané), E est définie par exemple par :

$$E = \frac{1}{2}\sum_{j=1}^{n}(y_j^{(d)} - y_j)^2 \quad (3.29)$$

où *n* est le nombre de neurones de sortie, y_j la sortie du neurone de sortie *j*, définie par :

$$y_j = f\left(\sum_i w_{ij} . y_i\right) \quad (3.30)$$

où f est la fonction d'activation (dérivable) des neurones de la couche de sortie, w_{ij} les poids des neurones de sortie et y_i les sorties des neurones de la couche précédente.
On voit que la dépendance entre y_j et w_{ij} est à 2 niveaux : la fonction d'activation f() et la somme. On fait donc apparaître une variable intermédiaire : a_j, l'activation du neurone *j* :

$$a_j = \sum_i w_{ij} y_i \quad (3.31)$$

et

$$y_j = f(a_j) \quad (3.32)$$

On effectue la décomposition du gradient de l'erreur par rapport aux poids, en dérivées partielles faisant apparaître toutes ces dépendances :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} . \frac{\partial y_j}{\partial a_j} . \frac{\partial a_j}{\partial w_{ij}} \quad (3.33)$$

Les paramètres y_j, a_j et w_{ij} concernent la couche de sortie.
On peut également effectuer cette décomposition pour le cas de neurones de couches cachées. *E* représente toujours l'erreur de sortie, et les autres paramètres y_j, a_j et w_{ij} sont ceux de la couche cachée concernée.

Les 2 derniers termes de cette expression s'exprimeront de la même manière quelle que soit la couche que l'on est en train de calculer : y_j, a_j et w_{ij} sont exprimés par rapport à cette même couche, et donc la dépendance entre eux sera la même d'une couche à l'autre. Par contre le premier sera différent pour la couche de sortie et pour les autres couches (couches cachées), puisque *E* est exprimé par rapport à la couche de sortie.

Calcul du 2eme terme

Le lien entre $y_j \; et \; a_j$ est : y_j = f(a_j). On a donc :

$$\frac{\partial y_j}{\partial a_j} = \mathrm{f}'(a_j) \tag{3.34}$$

où le prime désigne la dérivée par rapport à l'argument :

$$\mathrm{f}'(a_j) = \frac{\partial \mathrm{f}(a_j)}{\partial a_j} \tag{3.35}$$

Calcul du 3eme terme

L'activation a_j est définie par (3.31) donc :

$$\frac{\partial a_j}{\partial w_{ij}} = y_j \tag{3.36}$$

Calcul du 1er terme

1er cas : neurone de sortie

Dans le cas d'un neurone de sortie, la dépendance entre *E* et y_j est :

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i^{(d)} - y_i)^2$$

ou si l'on fait apparaître l'erreur de sortie e_i du neurone *i* par:

$E = \frac{1}{2}\sum_{i=1}^{n} e_i^2$ avec $e_i = y_i^{(d)} - y_i$

donc

$$\frac{\partial E}{\partial y_j} = -e_j$$

$$= -\left(y_j^{(d)} - y_j\right)$$

avec $\frac{\partial E}{\partial y_j} = \frac{1}{2}\left(\frac{\partial e_1^2}{\partial y_j} + \frac{\partial e_2^2}{\partial y_j} + \cdots + \frac{\partial e_j^2}{\partial y_j}\right)$

si l'on réunit tous ces résultats partiels :

$$\frac{\partial E}{\partial w_{ij}} = -e_j . \mathrm{f}'(a_j) . y_j \tag{3.37}$$

l'équation (3.27) devient finalement :

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} = \alpha e_j . \mathrm{f}'(a_j) . y_j = \alpha . \delta_j y_j \tag{3.38}$$

avec :

$$\delta_j = e_j . \mathrm{f}'(a_j)$$

Cette grandeur est appelée gradient local au neurone *j*.

2eme cas : neurone de la dernière couche cachée

Pour un neurone de cette couche, le calcul est le même que le cas d'un neurone de sortie. avec :

$$\frac{\partial E}{\partial y_j} = \frac{1}{2} \sum_k \frac{\partial e_k^2}{\partial y_j} \tag{3.39}$$

k est un indice sur tous les neurones de la couche de sortie. y_j représente maintenant la sortie d'un neurone de la couche cachée, ce qui change tout : contrairement au cas d'un neurone de sortie, où seule la dérivée de e_j par rapport à y_j était non-nulle, maintenant aucune des dérivées n'est nulle : en effet, e_k représente toujours une erreur de sortie (celle du $k^{\text{ième}}$ neurone de sortie), alors que y_j représente la sortie d'une couche en amont. Comme tous les neurones de cette couche sont connectés aux neurones de la couche de sortie, l'erreur de sortie du $k^{\text{ième}}$ neurone e_k dépend de toutes les sorties des couches en amont de la couche de sortie. Nous devons donc écrire :

$$\frac{\partial E}{\partial y_j} = \sum_k e_k \frac{\partial e_k}{\partial y_j} \tag{3.40}$$

On doit encore décomposer l'équation (3.40) car la dépendance entre e_k et y_j est à deux niveaux :

$$\begin{aligned} e_k &= y_k^{(d)} - y_k \\ &= y_k^{(d)} - \mathrm{f}(a_k) \\ &= y_k^{(d)} - \mathrm{f}\left(\sum_{\mathrm{j}} \mathrm{w_{jk}\, y_j}\right) \end{aligned}$$

où *j* est un indice sur tous les neurones de la dernière couche cachée.

La décomposition de l'équation (3.40) est donc :

$$\frac{\partial E}{\partial y_j} = \sum_k e_k \frac{\partial e_k}{\partial a_k} . \frac{\partial a_k}{\partial y_j} \tag{3.41}$$

avec

$\frac{\partial e_k}{\partial a_k} = -\mathrm{f}'(a_k)$ et $\frac{\partial a_k}{\partial y_j} = w_{jk}$ car $a_k = \sum_j w_{jk}\, y_j$

On obtient finalement :

$$\frac{\partial E}{\partial y_j} = -\sum_k e_k\, \mathrm{f}'(a_k) w_{jk} \tag{3.42}$$

donc

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}.\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}.\mathrm{f}'(a_j).y_j = -\left[\sum_k e_k\, \mathrm{f}'(a_k) w_{jk}\right].\mathrm{f}'(a_j).y_j \tag{3.43}$$

On remarque que le gradient local apparaît dans l'équation (3.43).

Posons que :

$\delta_k = e_k.\mathrm{f}'(a_k)$

On peut écrire :

$$\frac{\partial E}{\partial y_j} = -\sum_k \delta_k w_{jk}$$

et

$$\frac{\partial E}{\partial w_{ij}} = -\mathrm{f}'(a_j).y_j.\sum_k \delta_k w_{jk} \tag{3.44}$$

On a aussi, d'après la définition du gradient local δ_j :

$$\frac{\partial E}{\partial w_{ij}} = -\delta_j.y_j$$

donc, par identification : $\delta_j = \mathrm{f}'(a_j).\sum_k \delta_k w_{jk}$

On a donc obtenu une récurrence : pour la dernière couche cachée (donc l'avant dernière couche de neurones) le gradient local est exprimé en fonction des gradients locaux de la couche de sortie. On retrouvera cette récursivité entre les autres couches, et donc la méthode permet d'adapter les poids de n'importe quelle couche (en partant de la dernière et en remontant jusqu'à la première) du réseau, menant à une diminution progressive de l'erreur de sortie.

3.7.6 Paramètres de l'algorithme

Dans le cas de la rétropropagation du gradient comme dans le cas des réseaux de neurones en général, la plupart des paramètres sont déterminés de manière empirique.

3.7.6.1 Structure du réseau

Nombre de couches cachées

Une seule couche cachée permet de traiter la plupart des problèmes.
Le fait d'en utiliser 2 ou 3, voire plus, permet de définir des régions plus complexes dans l'espace des entrées.

Nombre de neurones dans les couches cachées

Si le nombre de neurones dans les couches cachées est trop grand, le réseau va avoir tendance à réaliser un apprentissage par coeur des données d'apprentissage, et donc à mal généraliser à de nouvelles données. S'il est trop petit, il ne possédera pas assez de variables internes pour résoudre le problème à traiter.
Le choix du nombre de neurones est donc un compromis entre ces 2 aspects.

Connectivité

En général, on utilise une connectivité complète entre les entrées et les différentes couches, c'est à dire que les neurones sont connectés à tous les neurones de la couche en amont, et à toutes les entrées pour le cas de la 1ère couche cachée.

3.7.6.2 Fonction d'activation

La fonction d'activation des neurones peut a priori être quelconque pourvu qu'elle soit dérivable, ce qui exclut la fonction seuil. La dérivée de la fonction d'activation intervient en effet dans la règle de modification des poids. De plus, il faut au moins qu'il y ait une couche avec une fonction d'activation non linéaire, car si toutes les couches étaient linéaires, cela serait équivalent à un réseau à une seule couche linéaire ; or un réseau monocouche ne peut pas traiter un problème non linéairement séparable.

En général, c'est la sigmoïde qui est utilisée. Elle peut être uniquement positive ou positive et négative (symétrique par rapport au centre du repère). Dans le 2eme cas, l'apprentissage est plus rapide car les poids sont modifiés même pour des valeurs d'entrées de la sigmoïde négative.

3.7.6.3 Taux d'apprentissage

Il s'agit du paramètre α présent dans la règle d'apprentissage. S'il est très faible, l'apprentissage est lent. S'il est grand, on risque d'avoir des oscillations des valeurs des poids. On peut le faire diminuer en cours d'apprentissage (comme dans le cas des cartes de Kohonen), pour améliorer la stabilisation du réseau.

3.7.6.4 Valeurs initiales des poids

Si les valeurs initiales des poids sont très faibles, l'apprentissage est lent car ils interviennent dans la règle de modification. Si elles sont grandes, l'apprentissage est également lent car la somme pondérée des entrées d'un neurone est grande, et la dérivée de la fonction de transfert (fonction d'activation) est faible. Or celle-ci intervient dans la règle de modification des poids.

3.7.6.5 Ajout d'un terme de moment

Une méthode pour accélérer la convergence de l'apprentissage est d'ajouter un terme appelé "moment" (momentum) dans la règle d'apprentissage, défini par :

$$\mu \Delta w_{ij}(n-1) \tag{3.45}$$

La règle d'apprentissage devient donc :

$$\Delta w_{ij}(n) = -\alpha \frac{\partial E}{\partial w_{ij}} + \mu . \Delta w_{ij}(n-1) \tag{3.46}$$

où *n* est l'indice de l'itération d'apprentissage actuelle (et donc *n*-1 l'indice de l'itération précédente) et avec :

$0 < \mu < 1$ (typiquement : $\mu \cong 0.5 - 0.99$)

Ce terme permet d'augmenter le taux d'apprentissage sans provoquer d'oscillations des poids, et donc d'éviter des minima locaux.

La variation des poids ne dépend donc plus que du gradient mais également de leur variation précédente.

Ce moment introduit une certaine "douceur" dans l'apprentissage, par le biais d'une récurrence : pour un exemple donné, si la variation des poids est grande, elle aura tendance à le rester à l'exemple suivant.

Il est nécessaire que μ soit inférieur à 1 en valeur absolue pour éviter une divergence. Il permet ainsi de s'affranchir des petites discontinuités de la surface d'erreur, et donc d'éviter de rester bloqué dans un minimum local. En général le moment est diminué au cours de l'apprentissage.

3.8 Conclusion

Ce chapitre a permis de rappeler les propriétés principales des réseaux de neurones utilisés dans la suite de cette thèse. Les réseaux de neurones présentent donc une très grande diversité, en effet, un type de réseau neuronal est défini par sa topologie, sa structure interne et son algorithme d'apprentissage. Selon la nature des connexions, plusieurs architectures des réseaux de neurones peuvent être obtenues et différents types d'apprentissage peuvent être utilisés. On a présenté l'architecture neuronale MLP par laquelle, nous avons concentré notre travail. Puis, nous avons décrit la technique dite de rétro-propagation du gradient qui est très souvent utilisée dans les réseaux de neurones.

Dans le prochain chapitre, nous allons aborder les réseaux neuronaux convolutionnels.

CHAPITRE 04
LES RESEAUX DE NEURONES CONVOLUTIFS

4.1 Introduction

En apprentissage automatique, un réseau de neurones convolutifs ou réseau de neurones à convolution (en anglais ConvNet pour *Convolutional Neural Networks*) est un type de réseau de neurones artificiels à propagation directe (*feed-forward*) dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des animaux. Les neurones de cette région du cerveau sont arrangés de sorte qu'ils correspondent à des régions qui se chevauchent lors du pavage du champ visuel. Leur fonctionnement est inspiré par les processus biologiques, ils consistent en un empilage multicouche de perceptrons, dont le but est de prétraiter de petites quantités d'informations.

Les réseaux à convolution (CNN) sont très utilisés dans les applications graphiques (traitement et reconnaissance d'images ou vidéos). Les deux caractéristiques principales des réseaux convolutifs sont qu'ils utilisent des filtres (kernel) et mettent en œuvre du pooling.

Dans ce chapitre, nous allons présenter les différents types de couches classiquement utilisés dans les CNN. Nous verrons dans un premier temps les origines de CNN en parcourant les différents travaux sur le cortex visuel. Par la suite, Nous expliciterons chacune de ces couches pour en déduire leurs différents paramètres et fonctionnements. Nous introduirons ainsi le principe de la profondeur de couche, de pas, et le zéro pading.

4.2 Origine du principe des réseaux de neurones convolutionnel

L'origine des CNN est liée aux travaux sur les mécanismes biologiques de la vision. Ainsi on peut remonter dans les années 1950 - 1960 avec les travaux de caractérisation du cortex visuel de Hubel et Wiesel [61] qui ont identifié deux types de cellules dans le cortex visuel, les cellules simples et les cellules complexes. Les couches de cellules simples sont utilisées pour l'extraction de motifs et les couches de cellules complexes réunissent les motifs reconnus, rendant l'application moins sensible aux variations de position [133]. Ils ont ainsi introduit les concepts de champs réceptifs 2D des neurones et de pooling. Dans les années 1980, la notion de Neocognitron inspirée par les travaux de Hubel et Wiesel, fut introduite par Kunihiko Fukushima [38]. La structure Neocognitron est une structure en couches, où celles ci sont interconnectées en cascade. Ces couches sont composées de cellules simples ou complexes et chacune possède son propre jeu de pondérations. Les couches de cellules simples sont utilisées pour l'extraction de motifs et les couches de cellules complexes réunissent les motifs reconnus, rendant l'application moins sensible aux variations de position. Finalement, cette structure de Neocognitron fut utilisée dans une application de reconnaissance de chiffres [133].

En 1986, la structure du Neocognitron a été réutilisée avec l'idée de partage des mêmes pondérations entre plusieurs neurones d'une même couche associée aux principes de base de l'apprentissage formel par optimisation du gradient, la rétropropagation du gradient (backpropagation, BP) [112]. Ainsi, les bases structurelles des réseaux convolutionels ont été posées avec une première formalisation d'un algorithme d'apprentissage. Par la suite, en 1998, Yann LeCun et *al* ont proposé une structure de réseaux de convolutions, LeNet-5 [80]. Un exemple d'architecture type de réseau de neurone convolutif est présenté en figure 4.1.

Les CNN sont constitués successivement de couches de convolutions, de couches de regroupements (**Pooling Layers**), et de couches connectées. Le terme d'apprentissage profond deep learning se réfère aux nombreuses couches qui doivent être apprises au fur de l'entraînement.

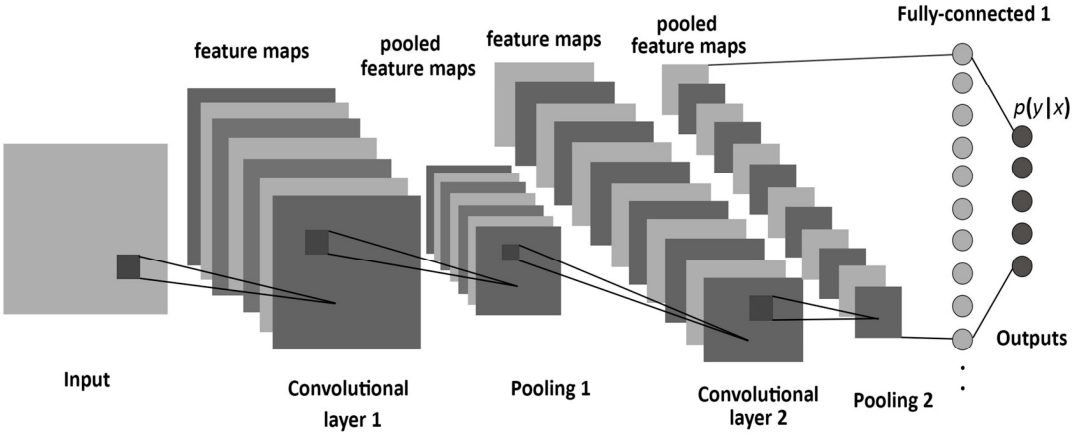


Figure 4.1 : Architecture classique d'un réseau de neurones convolutif. Une image est fournie en donnée d'entrée (input) et est convoluée avec des filtres (première couche de convolution) et dont les cartes d'activation sont regroupées et concaténées

Au vu de cette structure globale, nous allons par la suite définir la connectivité d'un CNN par la définition de la connectivité de chaque couche. Nous allons donc maintenant nous intéresser aux détails des connexions dans les couches.

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution**,** la couche de poolin**g**, la couche de correction (Non-linéarité) **ReLU** et la couche de classification (**fully-connected)**.

4.3 L'image, une matrice

Il faut savoir qu'une image est une matrice de valeurs correspondant aux pixels. Mais une image en couleur possède trois canaux. C'est à dire qu'on a 3 valeurs pour chaque pixels. Ces trois valeurs représentent le niveau de rouge, de vert et de bleu. On peut imaginer 3 matrices constituées de valeurs de pixels superposées, donnant une matrice en 3D. Par exemple, pour une image d'une taille de 200 par 200 pixels, sa matrice correspondante aura une taille de 200 (longueur) par 200 (largeur) par 3 (nombre de canaux).

En revanche, pour une image en noir et blanc, sa matrice correspondante ne sera qu'en 2D car possédant qu'un seul canal. Ici, la valeur d'un pixel indique le niveau de

gris, cela suffit pour une image en noir et blanc. La figure 4.2 montre un sous-tableau de 4×5 pixels extrait d'une image.

Les valeurs d'un pixel sont mappées sur 256 bits, c'est-à-dire allant de 0 à 255.

112	153	145	211	167
154	197	213	254	211
130	208	173	143	115
193	230	198	159	143

Figure 4.2: Exemple simpliste des valeurs des pixels d'une image 4x5

4.4 Le filtre

Le filtre est un élément important dans les réseaux de neurones convolutifs. Comme une image, le filtre est une matrice de valeurs mais elle a généralement de petites dimensions et bien souvent carrées. La taille de filtre la plus utilisée est de 3 par 3. La figure 4.3 montre un exemple de valeurs d'une matrice utilisée comme filtre

1	0	1
0	1	0
1	0	**1**

Figure 4.3: Exemple de valeurs d'une matrice utilisée comme filtre

Un filtre sert à faire ressortir certaines caractéristiques d'une image donnée (couleur, contour, luminosité, netteté,... etc.). Ce filtre va être déplacé par pas successifs sur l'image. La figure 4.4 montre le parcours de la fenêtre de filtre sur l'image.

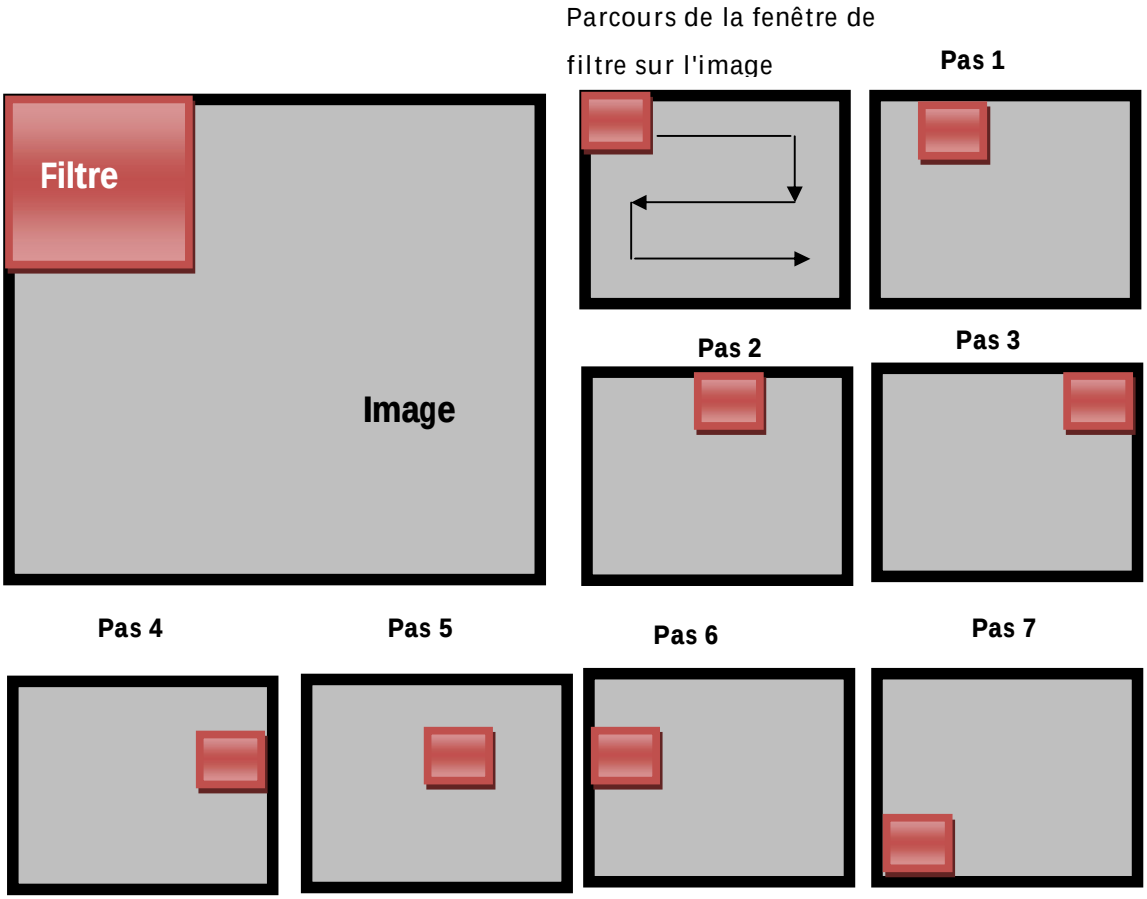


Figure 4.4 : Parcours de la fenêtre de filtre sur l'image

Pour chaque position du filtre, les valeurs des deux matrices en superposition (filtre et image à traiter) sont multipliées. Chaque valeur ainsi inférée est projetée dans une nouvelle matrice. Cette matrice représente une nouvelle image qui fait ressortir les caractéristiques recherchées au travers du filtre.

4.5 Les grands types de couches

4.5.1 La couche de convolution

La convolution consiste à appliquer un filtre sur l'image. Pour cela un kernel se déplace sur la totalité de l'image d'entrée et agit comme un filtre afin de produire une image en sortie. Le kernel a pour but de détecter des formes particulières . La figure 4.5 montre la convolution d'une image de dimension NxNx3 (le 3 correspond aux 3 channels R,G,B et N=32 dans notre exemple) avec un filtre.

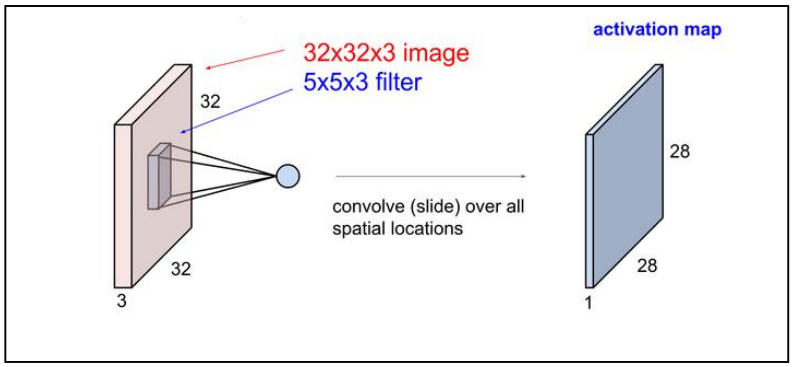


Figure 4.5 : Convolution d'une image de dimension NxNx3 avec un filtre de 5x5x3

La couche de convolution correspond à un filtre (ici de taille 5×5(x3)) que l'on va balayer sur l'ensemble de notre image. En sortie on obtient une "image" de taille 28x28x1 Qu'on appelle une "activation map". En général, lors d'une couche de convolution, on n'applique pas qu'un seul filtre, mais un ensemble de k filtres. On obtient alors une pile de k "activation maps" qui constitue notre nouvelle "image".

Un exemple de convolution de (04) filtres par une image d'entrée donne quatre cartes d'activation (Feature maps) distinctes est présenté en figure 4.6

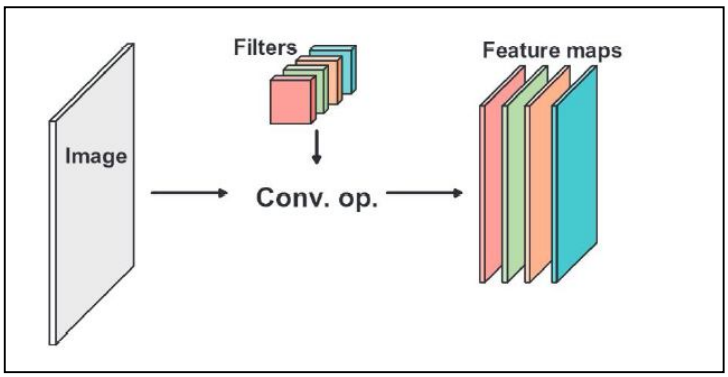


Figure 4.6 : **C**onvolution d'une image avec 4 filtres

4.5.1.1 Calcul de la convolution

Le calcul de la convolution est relativement simple : pour une convolution de taille 3×3, on va sélectionner dans l'image d'entrée les 3×3 premiers pixels pour en créer un nouveau dans l'image de sortie. La valeur de ce nouveau est égale à « pixel 1 de la

sélection dans l'image d'entrée» * « pixel 1 du filtre » + « pixel2 de la sélection dans l'image d'entrée» * « pixel 2 du filtre » + etc… jusqu'au neuvième pixel.

Ensuite, pour calculer la deuxième valeur de l'image de sortie, on va utiliser le paramètre « stride » de la convolution. Il représente de combien de pixels dans l'image d'entrée on va se décaler pour ré-appliquer la convolution . L'opération de convolution est illustrée dans la figure 4.7. On la calcule par l'´equation 4.1 où H est le produit de convolution résultant.

$$\mathbf{H} = \sum_{k=0}^{M} \sum_{l=0}^{N} \sum_{i=0}^{R} \sum_{j=0}^{C} \boldsymbol{I}_{(i+k),(j+l)} \, \boldsymbol{F}_{(R+i-1),(C-j+1)} \qquad (\mathbf{4.1})$$

avec

I : matrice de pixels représentant l'image

F : matrice du filtre de convolution à appliquer sur l'image

H : résultante du produit de convolution , appelée **feature map**

R : nombre de lignes de la matrice filtre

C : nombre de colonnes de la matrice filtre

$M = \frac{M'-R}{S} + 1$, avec M' est le nombre de lignes de la matrice d'images, et S le nombre de pas du filtre suivant la ligne et colonne

$\boldsymbol{N} = \frac{\boldsymbol{N'}-\boldsymbol{R}}{\boldsymbol{S}} + \mathbf{1}$, avec $\boldsymbol{N'}$ le nombre de colonnes de la matrice d'images.

La figure 4.7 donne un exemple de convolution discrète. La grille bleu clair est la matrice de pixels représentant l'image en entrée. Pour que le dessin reste simple, une seule carte de caractéristiques en entrée (matrice de pixels représentant l'image**)** est représentée, mais il n'est pas rare que plusieurs cartes de caractéristiques soient empilées les unes sur les autres. Un noyau (filtre) de valeur :

0	1	2
2	1	0
1	0	1

est glissé sur la carte des caractéristiques d'entrée. À chaque emplacement, le produit entre chaque élément du noyau et l'élément d'entrée qu'il chevauche est calculé et les résultats sont additionnés pour obtenir la sortie à l'emplacement actuel. La procédure

peut être répétée en utilisant différents noyaux pour former autant de cartes de caractéristiques de sortie que vous le souhaitez.

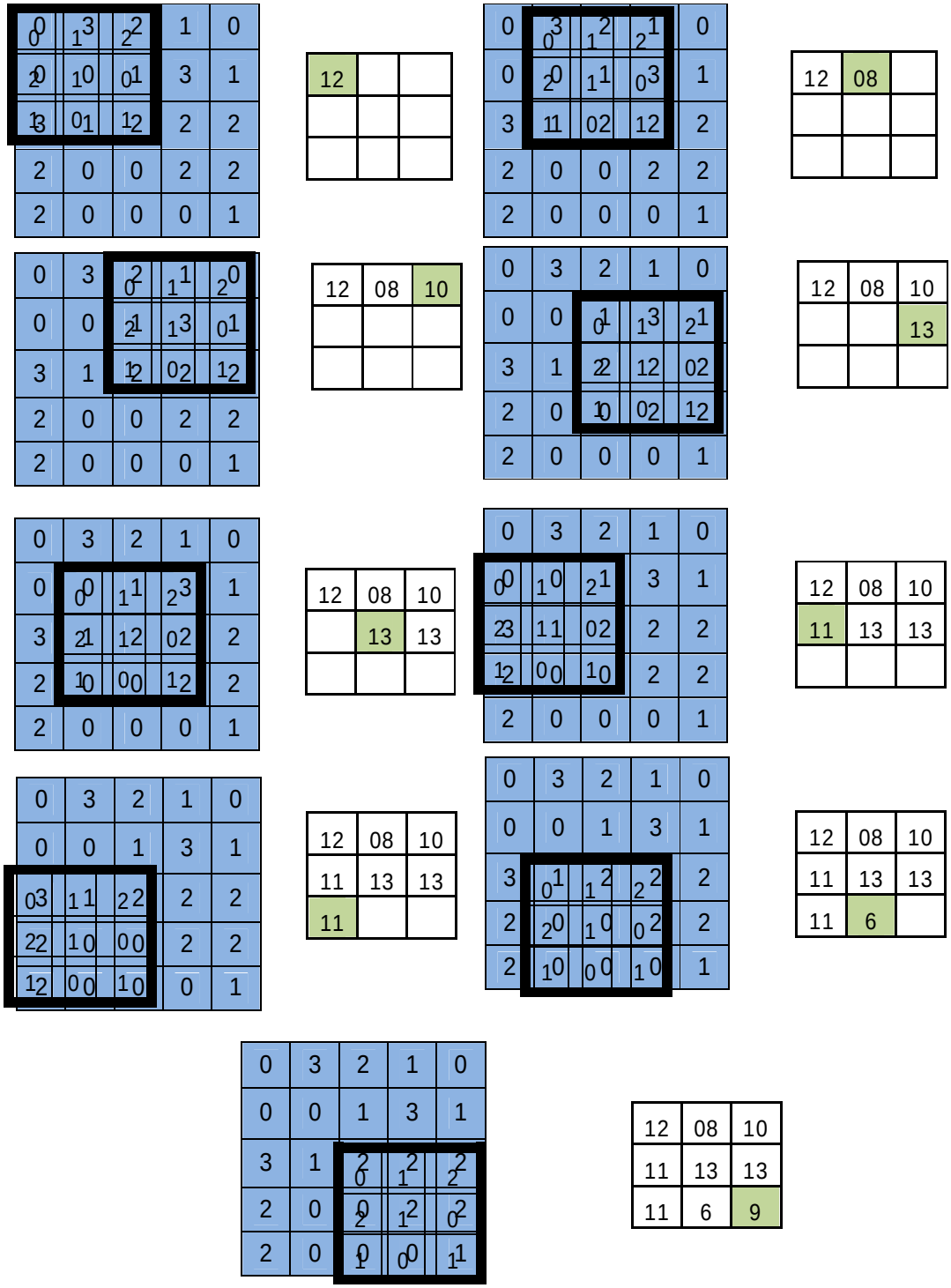


Figure 4.7: Calcul des valeurs de sortie d'une convolution discrète

Les résultats finaux de cette procédure sont appelées cartes de caractéristiques de sortie. S'il existe plusieurs cartes de caractéristiques en entrée, le noyau devra être tridimensionnel - ou, de manière équivalente, chacune des cartes de caractéristiques sera convertie en noyau distinct - et les cartes de caractéristiques résultantes seront résumées élément par élément pour produire la carte de caractéristiques en sortie.

La convolution représentée à la figure 4.7 est un exemple de convolution 2D, mais elle peut être généralisée aux convolutions N-D.

Il est nécessaire d'introduire quelques concepts afin de comprendre comment la convolution est utilisée dans le cadre d'un réseau convolutif. Les trois hyperparamètres suivants permettent de dimensionner le volume de la couche de convolution :

1. **Profondeur de la couche** : nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur);
2. **Le pas** : lorsque l'on effectue une convolution, on choisit ce qu'on appelle le pas «stride», c-a-d le pas auquel on déplace le noyau à travers l'entrant. Le pas horizontal représente le pas auquel on déplace horizontalement le noyau à travers l'image alors que le pas vertical représente le pas auquel on déplace verticalement le noyau à travers l'image. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand. Afin de simplifier, pour la suite des choses, on utilisera un pas horizontal égal à notre pas vertical afin de simplifier le modèle;
3. **La marge (à 0) ou zero padding** : parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce zero-padding est le troisième hyperparamètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée.

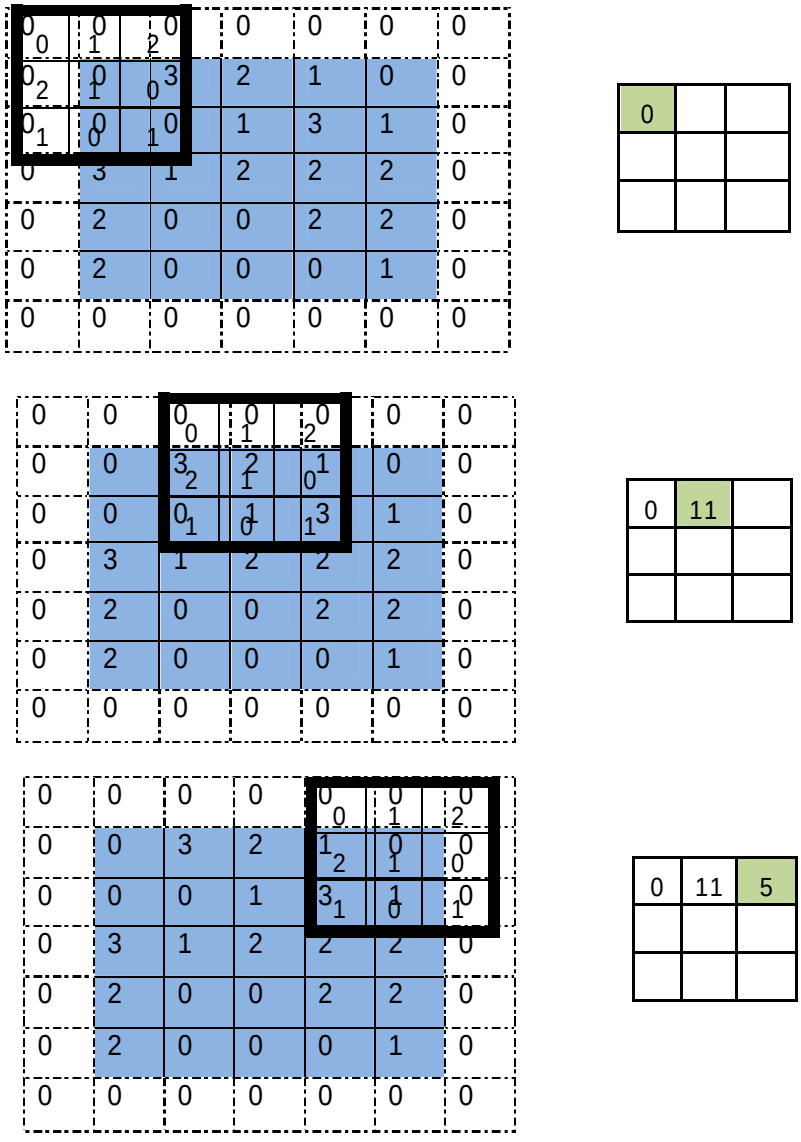
La taille spatiale du volume de sortie W_0 peut être calculée en fonction de la taille du volume d'entrée W_i , la surface de traitement K (nombre de champs récepteurs), le pas S avec lequel ils sont appliqués et la taille de la marge ou zero padding P. Elle est calculée selon l'équation suivante :

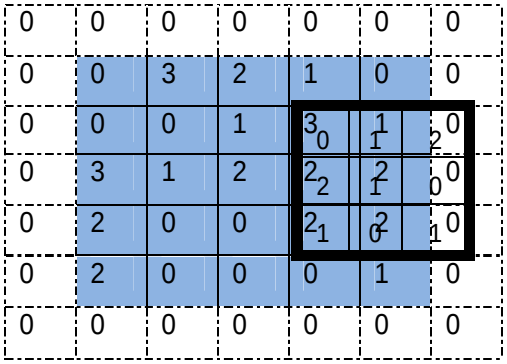
$$W_0 = \frac{W_i - K + 2P}{S} + 1 \qquad (4.2)$$

Souvent, on considère un pas $S = 1$, on calcule la marge de la manière suivante :

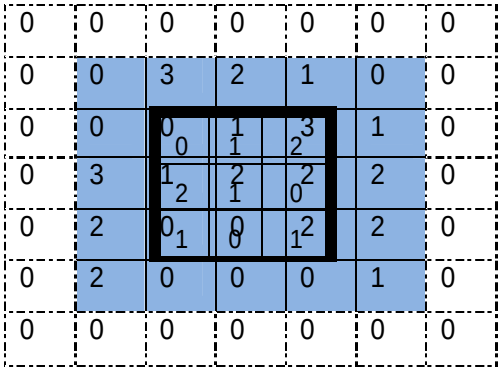
$$P = \frac{K-1}{2} \tag{4.3}$$

Si on souhaite un volume de sortie de même taille que le volume d'entrée. Dans ce cas particulier la couche est dite "connectée localement". Par exemple, la figure 4.8 montre un noyau $(kernel)3 \times 3$ appliqué à une entrée 5×5 complétée par une bordure (la marge) 1×1 de zéros utilisant un pas (strides) 2×2. Dans ce cas , on a $W_0 = 3, W_i = 5, K = 3, P = 1 \; et \; S = 2$.

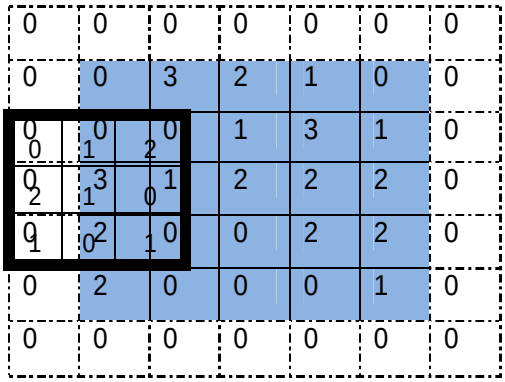




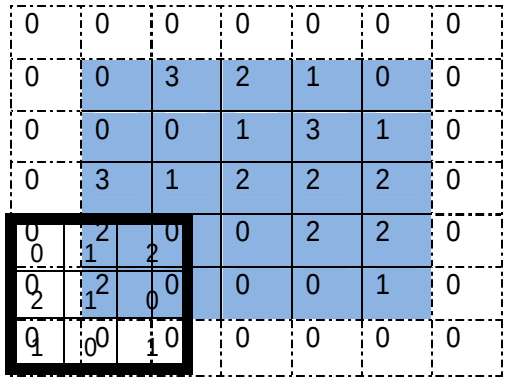
0	11	5
		9



0	11	5
	13	9



0	11	5
3	13	9



0	11	5
3	13	9
4		

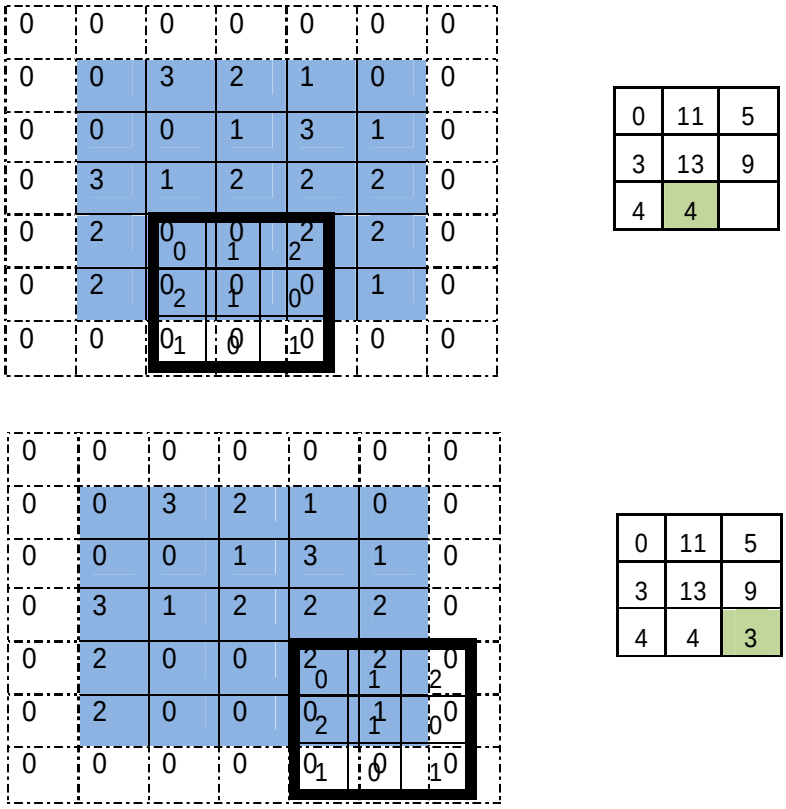


Figure 4.8: Exemple d'entrant 5 x 5 x 1 avec couche de marge à zéro de taille 1 auquel on applique un noyau 3 x 3 x 1. La sortie de notre convolution est de dimension 3 x 3 x 1. On a utilisé un pas de 2 pour les déplacements du noyau.

La figure 4.9 représente la convolution appliquée sur une image en format RVB. On utilise un seul filtre de dimension ***K***. Les trois carrés de gauche représentent l'image en RVB et le carré de droite représente la sortie (**feature map)** qui sera de dimension **:**

$$W_0 = \frac{W_i - K + 2P}{S} + 1$$

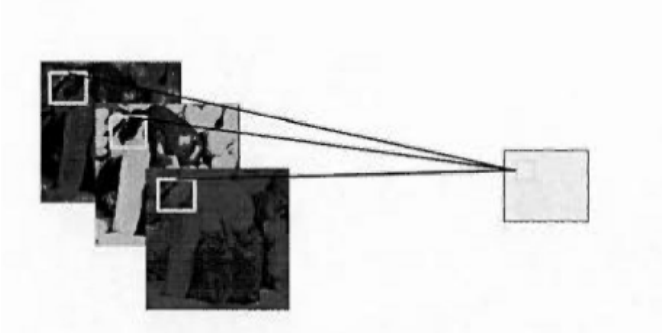


Figure 4.9 : Convolution appliquée sur une image en format RVB

4.5.1.2 Les différentes types de convolutions

Il existe plusieurs types de convolutions, à savoir, la convolution classique, la dilated convolution, la transposed convolution et la separable convolution.

- La **convolution classique**, dont nous avons expliqué au paragraphe 4.5.1.1. Elle a trois paramètres : la taille de filtre de convolution (appelé **kernel**), le **stride** qui représente le pas de décalage du kernel entre chaque calculs, et le **zero padding** qui est la manière dont on peut « dépasser » de l'image pour appliquer la convolution;
- La **dilated convolution**, identique à la convolution à ceci-près que le kernel est éclaté (on prend, par exemple, un pixel sur deux pour calculer la convolution). Il y a un paramètre supplémentaire : le **dilation rate**, qui est le nombre de pixels à ignorer;
- La **transposed convolution**, qui construit la sortie comme si on inversait une convolution sur l'image;
- La **separable convolution**, qui est une convolution décomposable en convolutions plus simples.

4.5.2 Couches de correction (RELU)

La couche de correction joue le rôle de fonction d'activation. Elle est appliquée après la convolution et laisse la taille volume de sortie inchangé. Dans ce cadre, on trouve par exemple la fonction d'activation ReLU (*Rectified Linear Units*) désigne la fonction réelle non-linéaire définie par ReLU(x)=max(0,x).

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Souvent, la correction Relu est préférable, mais il existe d'autre forme :

- La correction par *tangente hyperbolique* : $f(x) = tanh\,(x)$,
- La correction par la *tangente hyperbolique saturante* : $f(x) = |tanh\,(x)|$,
- La correction par la *fonction sigmoïde:* $f(x) = (1 + e^{-x})^{-1}$.

La figure 4.10 démontre la correction d'une feature map en utilisant la fonction d'activation ReLU(x)=max(0,x).

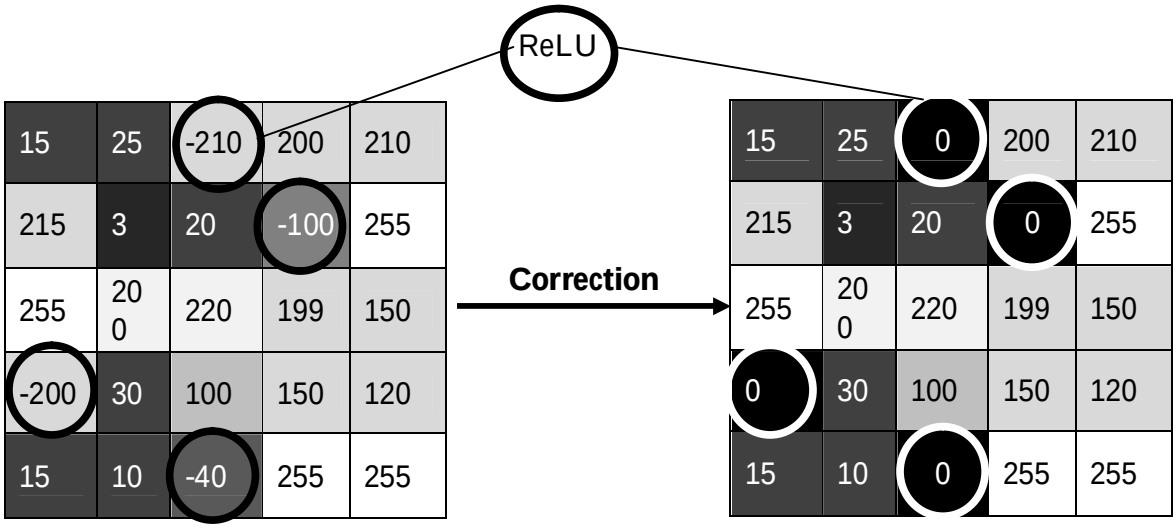


Figure 4.10: Correction d'une feature map

4.5.3 Couche de mise en commun: pooling

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs *feature maps*, et applique à chacune d'entre elles l'opération de *pooling*. L'opération de pooling (ou sub-sampling) consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Pour cela, on découpe l'image en cellules régulières, puis on garde au sein de chaque cellule la valeur maximale. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2 × 2 pixels qui ne se chevauchent pas. On parle dans ce cas de « Max-Pool 2x2 », ou des cellules de taille 3 × 3 pixels, distantes les unes des autres d'un pas de 2 pixels (qui se chevauchent donc). On obtient en sortie le même nombre de *feature maps* qu'en entrée, mais celles-ci sont bien plus petites.

La couche de *pooling* permet de réduire le nombre de paramètres et de calculs dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage.

Il est courant d'insérer périodiquement une couche Pooling entre les couches Conv successives dans une architecture ConvNet. Sa fonction est de réduire progressivement la taille spatiale de la représentation afin de réduire la quantité de paramètres et de calculs dans le réseau, et donc de contrôler également le sur-ajustement.

La couche de pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne spatialement, en utilisant l'opération MAX.
Il est possible d'utiliser d'autres fonctions de pooling que le maximum (Max-pooling). On peut utiliser un « average pooling » (la sortie est la moyenne des valeurs du patch d'entrée), du « L2-norm pooling ». Dans les faits, même si initialement l'average pooling était souvent utilisé il s'est avéré que le max-pooling était plus efficace car celui-ci augmente plus significativement l'importance des activations fortes. En d'autres circonstances, on pourra utiliser un pooling stochastique.

La figure 4.11 donne un exemple de Max-pooling, et la figure 4.12 fait de même pour average pooling.

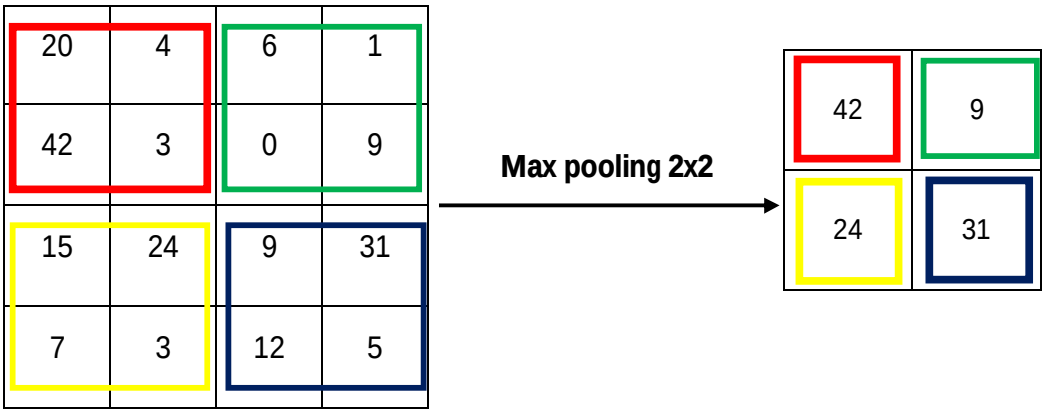


Figure 4.11: Exemple de max pooling (2x2), pas de deux

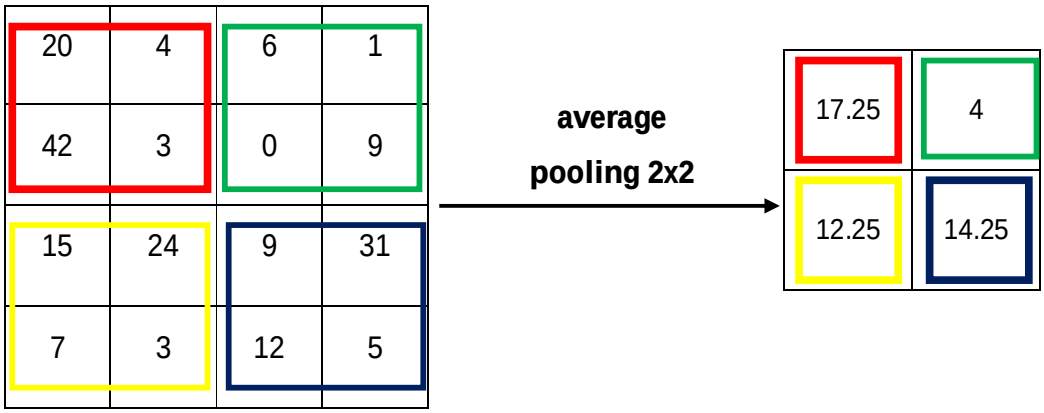


Figure 4.12: Exemple d' average pooling (2x2), pas de deux

4.5.3 .1 La différence entre max pooling et mean pooling

La différence est importante : de par sa nature, max-pooling va avoir tendance à retenir les caractéristiques (features) les plus marquées et simples de la sélection de pixels, comme par exemple une arête verticale. A l'inverse, mean étant une moyenne, seules les features moins marquées ressortiront.

De manière générale, il est recommandé d'utiliser max-pooling, car il se distingue de mean-pooling sur les cas extrêmes et est quasiment équivalent à mean-pooling dans les autres cas.

4.5.4 Mise à plat (ou Le flattening)

La mise à plat est la dernière étape de la partie d'extraction de caractéristiques « extraction des informations » qui consiste simplement à mettre bout à bout toutes les images (matrices) que nous avons pour en faire un (long) vecteur. Les pixels (en réalité ce ne sont plus des images ou des pixels, mais des matrices de nombres, donc les pixels sont ces nombres) sont récupérés ligne par ligne et ajoutés au vecteur final. La mise à plat des images finales en sortie des filtres est illustrée à la figure 4.13.

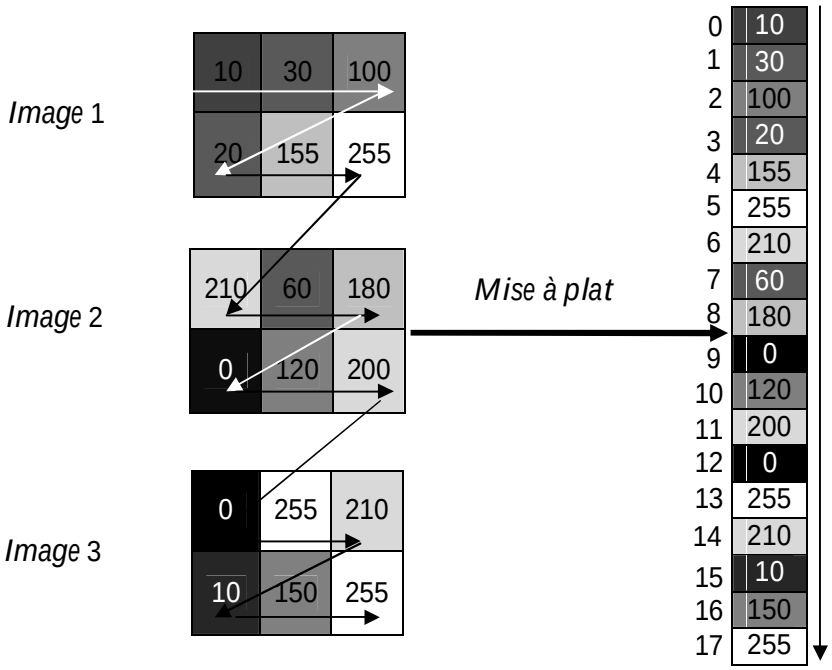


Figure 4.13 : Mise à plat des images finales en sortie des filtres

L'intérêt de cette étape est que , le réseau de neurones à la couche entièrement connectée (FC) (étape 5) prend simplement en entrée un vecteur (à chaque neurone d'entrée on envoie une seule valeur).

4.5.5 La couche fully-connected

La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones convolutif . Ce type de couche reçoit le vecteur de mise à plat en entrée et produit un nouveau vecteur en sortie (vecteur de classes) . Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée. La dernière couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes, par exemple pour le problème de classification d'images. Chaque élément du vecteur final indique la probabilité pour l'image en entrée d'appartenir à une classe.

Pour calculer les probabilités, la couche *fully-connected* multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique la fonction d'activation *softmax* . Cette dernière permet de normaliser le vecteur de sortie (vecteur de classes) pour que les valeurs de sortie soient simplement ramenées entre 0 et 1 de sorte à ce que leur somme fasse 1). Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme fully-connected.

4.6 Entraînement d'un réseau de neurone convolutif

4.6.1 La base de données

Lorsque l'on crée un modèle de réseaux de neurones convolutifs ou tout simplement quand on utilise un modèle déjà existant, nous devons entraîner ce modèle avec des données afin qu'il puisse apprendre et être capable de reconnaître une action ou quelque chose en particulier. Pour cela on utilise une base de données contenant toutes nos séquences vidéo. Une partie de cette base de données servira à entraîner le modèle CNN, c'est ce qu'on appelle «la base de données d'entraînement ou d'apprentissage». La partie de la base de données restante sert pour tester le modèle, on

l'appelle la «base de données de validation**».** Ce qui est important de savoir c'est que toutes les séquences vidéo que nous utilisons doivent êtres préalablement étiquetées**,** c'est à dire que chaque séquence vidéo que nous utilisons doit appartenir à une classe en particulier. Par exemple si nous souhaitons reconnaître des voitures, des vélos et des motos, cela nous donne 3 classes. Chaque séquence vidéo de notre base de données doit appartenir à une de ces trois classes. C'est ce que l'on appelle l'apprentissage supervisé.

4.6.2 Apprentissage du CNN

L'entraînement d'un CNN consiste à calculer la valeur de chacun de ses poids**.** Le principe est le suivant : le CNN traite une séquence de vidéo (de la base de données d'entraînement) et en sortie il fait une prédiction**,** c'est-à-dire qu'il dit à quelle classe il pense que cette séquence appartient. Sachant qu'on connaît préalablement la classe de chacune des séquences d'entraînement**,** on peut vérifier si ce résultat est correct.

En fonction de la véracité de ce résultat, on met à jour tous les poids du CNN selon un algorithme qui s'appelle la rétropropagation du gradient de l'erreur. Lors de la phase d'entraînement du modèle, le processus de mis à jour des poids du CNN est répété plusieurs fois et avec la totalité des séquences de la base de données d'entraînement**.** Le but étant que le modèle classifie au mieux ces données. Lorsque le modèle a fini de mettre à jour ses poids, on évalue le modèle en lui présentant la base de données de validation**.** Il classe toutes ces séquences (qui sont des séquences que le modèle n'a jamais vues**)** et on calcule son taux de bonne classification**,** c'est ce qu'on appelle la précision du modèle**.**

4.6.2.1 Sur-apprentissage et sous-apprentissage

A la fin du processus d'apprentissage trois cas peuvent se présenter :

- Le modèle est aussi performant sur les données d'entraînement (séquences sur lesquelles il s'entraîne) que sur les données de validation (séquences qu'il n'a jamais vues). Cela est le cas idéal, ça signifie que le modèle a très bien fait son travail et qu'il reconnaît aussi bien les images qu'il connaît que celles qu'il n'a jamais vues;

- Le modèle reconnaît très bien les séquences d'entraînement et moins bien celles de validation. Le modèle aura une faible capacité prédictive, il n'arrive pas à généraliser. On parle alors de sur-entraînement. Dans ce cas là on peut ajouter davantage des séquences pour pallier ce problème;
- Le modèle ne reconnaît pas très bien les séquences d'entraînements et pas très bien les séquences de validation. On parle alors de sous-apprentissage. Dans ce cas là ajouter plus des séquences vidéo ne servira à rien, c'est généralement le modèle choisie qui ne convient pas, il faudrait utiliser un modèle de CNN plus complexe.

4.7 Évaluation de la performance de la classification

Une fois les différents descripteurs extraits et la classification réalisée, la technique d'évaluation de performance par la matrice de confusion permettant de vérifier la fiabilité des descripteurs utilisés et la pertinence de l'approche de classification.

4.7.1 La matrice de confusion

La matrice de confusion ou tableau de contingence sert à évaluer la qualité d'une classification. C'est un tableau 2D de taille K×K – où K est le nombre total de classes utilisé pour disposer les résultats d'une expérience de classification. Cette matrice permet aussi de comprendre de quelle façon le modèle de classification est confus lorsqu'il effectue des prédictions. Ceci permet non seulement de savoir quelles sont les erreurs commises, mais surtout le type d'erreurs commises. Les utilisateurs peuvent les analyser pour déterminer quels résultats indiquent comment les erreurs sont commises. Ainsi, la valeur à la ligne i et à la colonne j indique le nombre de fois qu'un objet dont la vraie classe est i, est classifié comme appartenant à la classe j. La diagonale principale de cette matrice indique le nombre de cas pour lesquels le classifieur a bien réussi sa tâche ; un classifieur parfait devrait afficher zéro pour tous les éléments situés en dehors de la diagonale principale. La figure 4.14 montre un exemple d'une matrice de confusion de la classification des actions dans les classes marcher, courir et sauter.

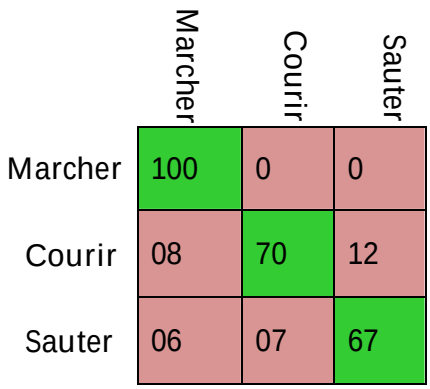


Figure 4.14 : Exemple d'une matrice de confusion de la classification

Dans la matrice de confusion de la figure 4.14, on observe que les 100 séquences vidéo d'action marcher , ont été parfaitement bien classifiées ce qui équivaut à une sensibilité de 100% ; par contre sur les 90 séquences vidéo d'action courir, 70 ont été correctement classifiées tandis que 08 séquences ont été rangées comme séquences d'action marcher et 12 séquences ont été rangées comme action sauter, ce qui donne une sensibilité de 77.77% et pour l'action sauter sur les 80 séquences vidéo , 67 ont été correctement classifiées tandis que 06 séquences ont été rangées comme l'action marcher et 07 séquences ont été rangées comme action courir, ce qui donne une sensibilité de 83.75%.

4.8 Conclusion

Un réseau de neurones convolutifs se compose donc d'une ou plusieurs étapes de convolution (filtrage, Pooling) suivi d'une classification. Lors de la phase d'entrainement d'un modèle CNN, en fonction de chaque séquence vidéo entrante dans le réseau et du résultat produit par l'étape de classification, le taux d'erreur est calculé. La valeur de ce dernier est utilisée pour mettre à jour les paramètres du réseau afin de l'optimiser.

De nombreux paramètres sont mis en place : nombre et taille des filtres, méthode de Pooling, nombre de couches de neurones, nombre de neurones par couche,.. etc. L'optimisation d'un réseau de neurones convolutifs requiert un minimum d'expérience. En général, il ne faut pas utiliser de filtres trop petits et de ne pas user du nombre de

niveaux de convolution ni du nombre de filtres causant un sur-apprentissage du réseau. Mais, les bonnes valeurs, dépendent du problème traité.

Les réseaux à convolution, malgré leur complexité, sont fortement conseillés pour la classification d'images et de vidéos.

Le prochain chapitre, traite les détails de l'architecture CNN 3D , ainsi que l'algorithme de filtrage et les outils utilisés pour la classification des actions humaines dans les séquences vidéo.

CHAPITRE 05
RESULTATS ET DISCUSSIONS

5.1 Introduction

Ce chapitre vise la reconnaissance de l'action humaine dans la vidéo. Les actions humaines dans la vidéo sont des signaux spatio-temporels tridimensionnels (3D) qui caractérisent à la fois l'apparence visuelle et la dynamique de mouvement des humains. Dans ce chapitre, nous nous intéresserons principalement à l'information dynamique. Nous utiliserons des réseaux de neurones convolutionnels profonds 3D qui prennent des cartes de fot optiques en entrée. Le processus de traitement de notre approche se compose principalement de trois parties :

- Calcul de cartes de flot optique;
- Filtrage de ces cartes, pour augmenter le taux de classification et de réduire le temps du calcul, qui ne contiennent pas de l'action en utilisant l'entropie pour chaque séquence ;
- Les réseaux de neurones convolutionnels profonds 3D pour l'extraction de caractéristiques et la classification des actions humaines.

Enfin, nous comparerons les résultats de classification trouvés par l'approche proposée avec les meilleurs travaux connexes sur la même base de données.

5.2. Description de l'approche

Pour reconnaitre l'action humaine, nous proposons une approche dont les principales étapes sont illustrées dans la figure 5.1.

Étape 1: Calcul des cartes de fot optique : elle permet de calculer des cartes de flot optique. Chaque carte représente le déplacement de chaque pixel entre deux images successives.

Étape 2: Suppression des cartes de flot optique qui ne contiennent pas de l'information (filtrages) : elle permet de calculer l'entropie de chaque séquence de 30 cartes et de supprimer les séquences qui ne contiennent pas de l'information.

Étape 3: Classification de l'action : elle permet de classer l'action dans une vidéo en utilisant des ConvNets 3D (3D CNN).

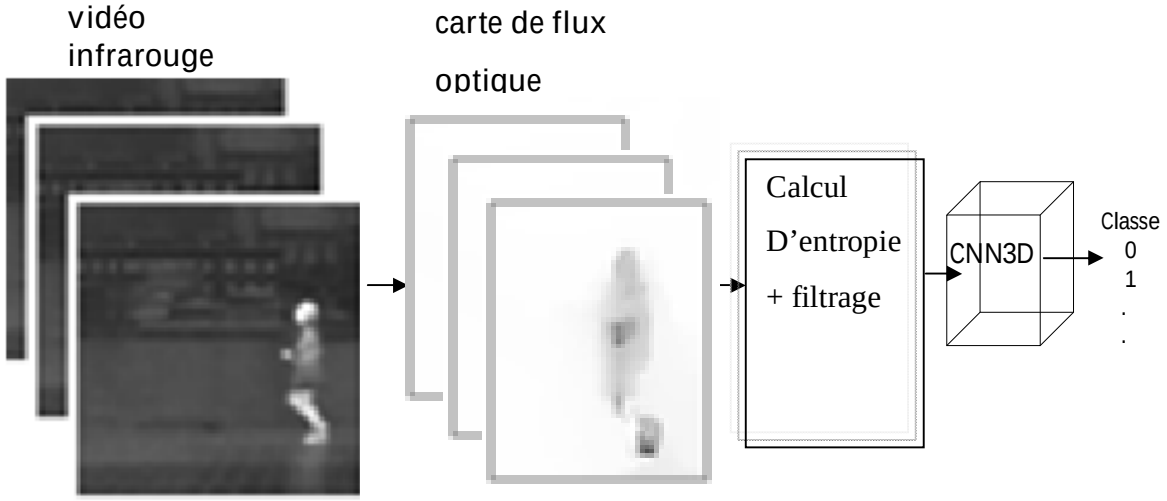


Figure 5.1 Structure générale du système de reconnaissance d'action

L'entropie *H* d'une image est définie par :

$$H = -\sum_{k=1}^{L} P_r(X_k) log_2 P_r(X_k) \quad (5.1)$$

où

L est le nombre de niveaux de gris et $P_r(X_k)$ est la probabilité associée au niveau de gris *k*.

5.3 Estimation du flot optique

Le calcul du flot optique consiste à évaluer le déplacement de points dans une séquence d'images au cours du temps. L'évaluation du flot optique consiste à mettre en correspondance un point entre deux images successives, correspondant au même point

de la scène 3D se déplaçant dans une séquence. En règle générale, il s'évalue entre deux images consécutives (et donc 2 instants de prise d'image successifs), et non entre une image de départ (t = 0) et une image prise à un instant suivant t quelconque [83]. Le flot optique suppose une constance de luminosité, ce qui donne :

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{5.2}$$

En prenant l'approximation de la série Taylor du côté droit de (5.2), en supprimant les termes communs et en les divisant par dt on obtient l'équation suivante :

$$f_x u + f_y \vartheta + f_t = 0 \tag{5.3}$$

où

$$f_x = \frac{\partial f}{\partial x} \quad ; \quad f_y = \frac{\partial f}{\partial y} \tag{5.4}$$

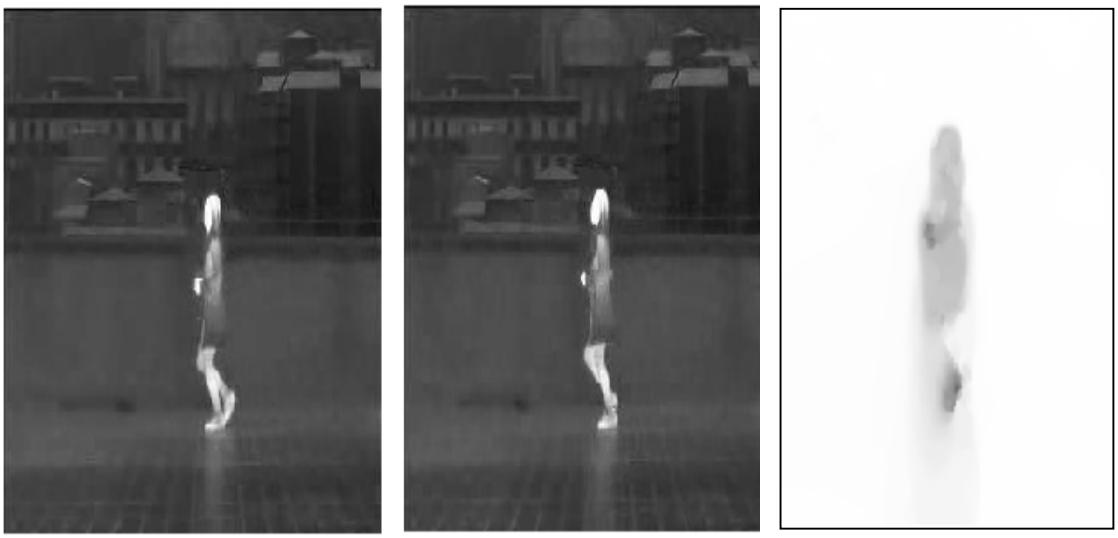
$$u = \frac{dx}{dt} \; ; \; \vartheta = \frac{dy}{dt} \tag{5.5}$$

Les équations ci-dessus décrivent le flot optique en termes de gradient d'image spatiale avec deux inconnuesu, ϑ. f_x et f_y sont des gradients d'image et f_t est le gradient dans le temps. Cette équation avec deux variables inconnues ne peut pas être résolue.
Diverses méthodes ont été suggérées pour résoudre ce problème (voir chapitre 2). Nous sommes basés sur le calcul du flot optique donné par [19].

La figure 5.2 montre la carte de flot optique calculée entre des paires de trames consécutives.

5.4 Base de données InfAR

Nous avons appliqué notre technique basée sur le modèle CNN 3D sur l'ensemble de données InfAR [40]. Cet ensemble de données comprend 12 classes d'actions humaines. Chacune de ces classes est représentée par 50 clips vidéo de longueurs variables, allant de 80 images à 280 images. La fréquence d'images est de 25 images par seconde et la résolution est de 256 x 293. Chaque vidéo contient une ou plusieurs personnes effectuant une ou plusieurs actions. Pour chaque classe, nous avons 35 vidéos pour l'apprentissage et 15 vidéos pour le test.



a b c

Figure 5.2: Flot optique. (a), (b): une paire de trames vidéo consécutives. (c) : flot optique pour (a) et (b)

5.5 Calcul d'entropie et filtrage de carte de flot optique

Les étapes de calcul d'entropie en utilisant l'équation (5.1) de chaque 30 carte de flot optique ainsi que le filtrage des cartes qui ne contiennent pas de l'information sont donnés par l'algorithme 1.

Algorithme 1 : Filtrage des séquences du flot optique

Entrée : les séquences S{i} et leurs entropies E(i)
Sortie : les séquences à supprimer S{i}

1. Q : le nombre de séquences dans la vidéo
2. **For** i=1 :Q-1
   $y(i) = |E(i+1) - E(i)|$ **(**Calcul de la différence d'entropie entre deux séquences consécutives)
3. $if\ y(i) < seuil$
   Save(S{i},s{i+1})
   End if
   End for

Les exemples ci-dessous montrent les résultats trouvés par l'algorithme 1

Exemple 1 :

La figure 5.3 montre les séquences de flot optique de l'action course, chaque séquence contienne 30 frames. D'après cette figure, on a constaté visuellement que les séquences 1,4 et 5 ne contiennent pas de l'action et dans ce cas, ces séquences sont considérés comme un bruit et doit être filtrés pour réduire le temps de calcul et d'augmenter aussi la précision de l'algorithme de classification des actions (CNN3D).

Le tableau 5. 1 donne les valeurs de l'entropie $E(i)$, ainsi que la différence d'entropie entre deux séquences consécutives $y(i) = |E(i+1) - E(i)|$ et aussi les séquences sauvegardées après le filtrage. Tout d'abord, nous pouvons constater que $y(2) = |E(3) - E(2)| = 6.1579 < 10(seuil)$, l'algorithme 1 sauvegardera les séquences $s\{2\}, s\{3\}$ et de supprimer les autres séquences automatiquement.

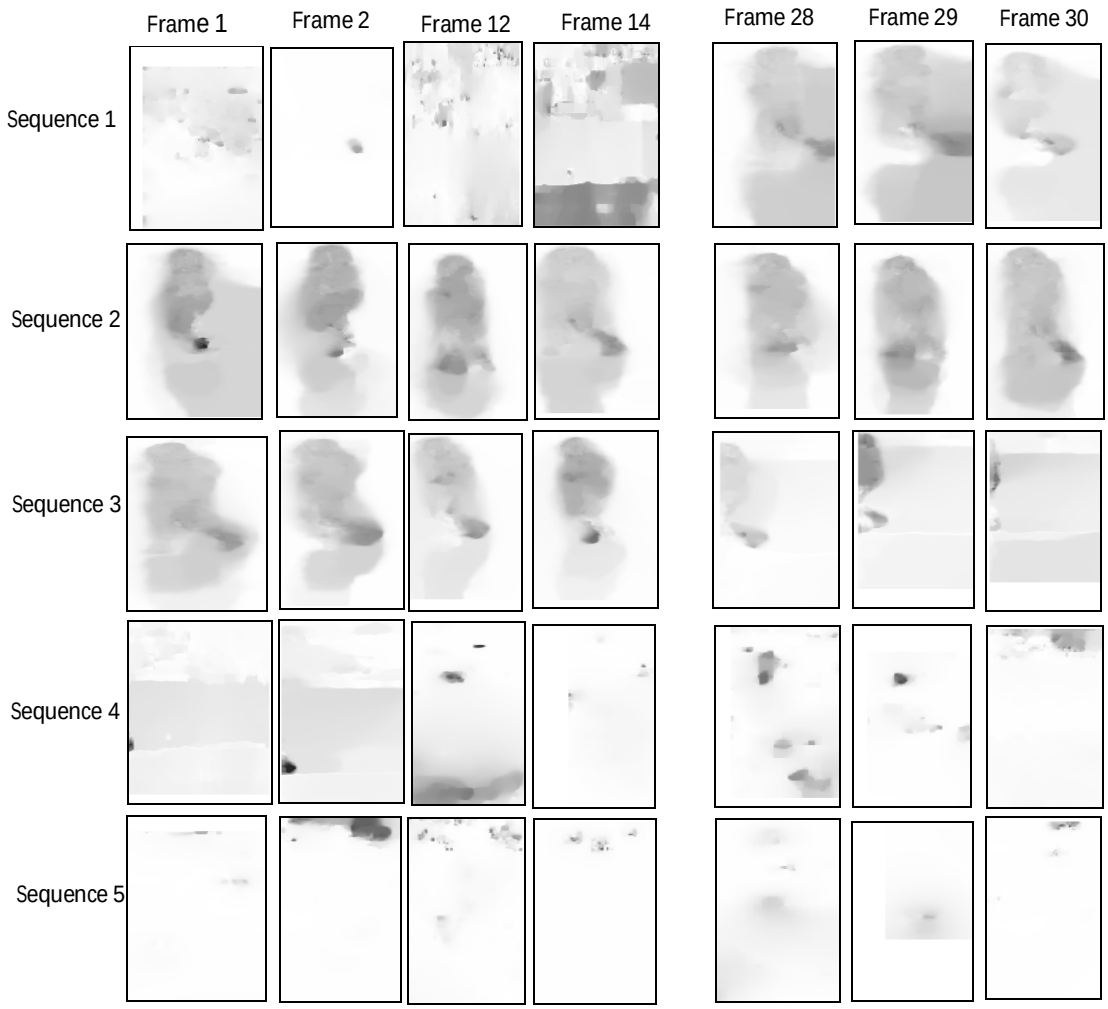


Figure 5.3 : Séquences de flot optique de l'action course

Séquence (S)	Entropie (E)	$y(i)$	Seuil	s{i} et s{i+1} sauvegarder
s {1}	79.7371	12.5513	10	
s {2}	67.1857	**6.1579**		**s{2}** et **s{3}**
s {3}	73.3436	33.0443		
s {4}	106.3879	46.7350		
s {5}	59.6529			

Tableau 5. 1 : Entropie pour chaque séquence de flot optique de l'action course

Exemple 2 :

La figure 5.4 montre les séquences de flot optique de l'action mouvement de deux mains. D'après cette figure, on a constaté visuellement que les séquences 1 et 2 contiennent de l'action.

Le tableau 5.2 donne les valeurs de l'entropie $E(i)$. Nous pouvons constater que $y(1) = |E(2) - E(1)| = 6.1039 < 10(seuil)$ et l'algorithme 1 sauvegardera les deux séquences $s\{1\}\ et\ s\{2\}$.

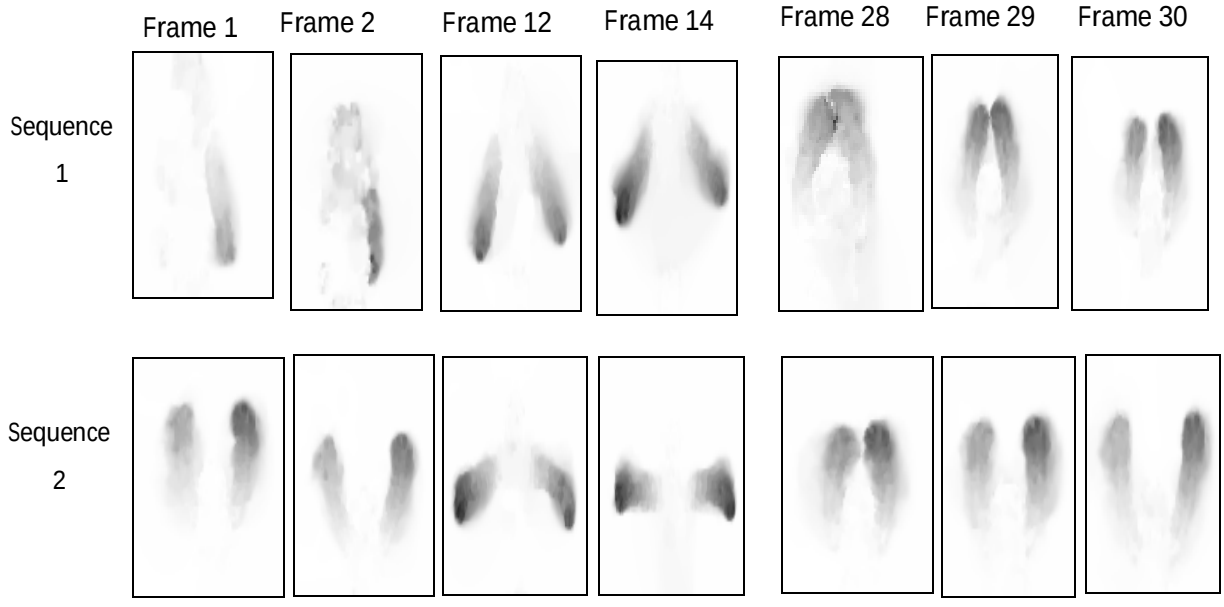


Figure 5.4 : Séquences de flot optique de l'action mouvement de deux mains

Séquence (S)	Entropie (E)	$y(i)$	Seuil	s{i} et s{i+1} sauvegarder
s {1}	29.0483	**6.1039**	10	
s {2}	35.1522			**s{1}** et **s{2}**

Tableau 5. 2: Entropie pour chaque séquence de flot optique de l'action mouvement de deux mains

5.6 Réseaux de neurones convolutifs 3D

Avant d'introduire l'architecture proposée, nous présentons les réseaux de neurones convolutionnels 3D (CNN3D). Le modèle CNN3D est une variante de 2DCNN développée pour les tâches de reconnaissance des images fixes (voir chapitre 04). Les CNN 3D traitent la séquence de données d'entrée sous la forme d'une séquence fixe et appliquent des couches de convolution 3D pour calculer les cartes de caractéristiques.

Généralement les CNN3D ont deux couches : la couche de convolution 3D et la couche de sous-échantillonnage 3D [53,63,125] . Les informations spatiales et temporelles sont abstraites couche par couche [86]. Les couches neuronales de mise en commun 3D sont utilisées pour réduire le nombre de paramètres dans le levier spatio-temporel. La fonction de la couche de convolution 3D est d'appliquer plusieurs filtres convolutionnels sur les volumes d'entrée. La couche de pooling 3D sélectionne la meilleure caractéristique extraite par la couche de convolution 3D [53].

5.6.1 Notation

- $a_{ij}^{x,y,z}$ désigne la valeur du pixel à la position (x, y, z) de la carte de caractéristiques $\boldsymbol{j}$ au niveau de la couche $\boldsymbol{i}$.
- f désigne la fonction d'activation, qui est le **tanh** (voir chapitre 03) dans la couche de convolution 3D.
- m désigne l'ensemble des cartes de caractéristiques dans la couche $(\boldsymbol{i-1})$ qui sont connectés à la carte des caractéristiques actuelle $\boldsymbol{j}$.
- b_{ij} représente le terme de biais de la $j^{ème}$ unité de la couche i.

- P, Q, R désignent la largeur, la hauteur de l'image et la dimension temporelle, respectivement sur lequel est calculée la convolution.
- $w_{ijm}^{p,q,r}$ représente le paramètre du noyau à la position (p, q, r) et à l'unité m de la couche précédente.

5.6.2 Couche de convolutions 3D

La convolution 3D est réalisée par la convolution d'un noyau 3D avec l'ensemble de multiples trames contiguës. Grâce à cette construction, les cartes des caractéristiques dans la couche de convolution sont connectées à plusieurs images contiguës dans la couche précédente, ce qui permet de capturer les informations de mouvement. La couche de convolution 3D peut être calculée comme suit :

$$a_{ij}^{xyz} = f\left(b_{ij} + \sum_m \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} \sum_{r=0}^{R-1} w_{ijm}^{pqr}\, a_{(i-1)m}^{(x+p)(y+q)(z+r)}\right) \quad (5.6)$$

5.6.3 Couche de pooling 3D

Pour réduire la dimension de la carte caractéristique, une étape de sous-échantillonnage est effectuée sur ces cartes de caractéristiques. La méthode de max-pooling est utilisée dans ce travail (voire chapitre 04). Le max-pooling consiste à transférer en entrée de la couche suivante les valeurs des maxima locaux, pour un voisinage donné. Aucun paramètre ne doit être appris. La couche de pooling 3D est calculée comme suit :

$$a_{ij}^{xyz} = max_{i,j,k \in \{1,2,\ldots,g\}} a_{(i-1)j}^{(gx+i)(gy+j)(gz+z)} \quad (5.7)$$

Où g est la longueur de la région de pooling.

5.6.4 Configuration du réseau

La figure 5.5 présente l'architecture globale de CNN3D pour la reconnaissance d'action humaine. Cette architecture comporte 6 couches. Il y a deux couches alternées de convolution et de sous-échantillonnage C1, S2 et C3, S4 suivies de deux couches de neurones entièrement connectées FC1 et Softmax. La taille de la couche d'entrée 3D est de 256 × 293 × 30, correspondant à 30 images de flot optique successives de 256 × 293

pixels chacune. La première couche est une couche de convolution (C1) composée de 08 cartes caractéristiques de 250 × 288 ×26 unités, chacune des unités est connectée avec un 7 × 6 × 5 dans la couche précédente, appelée champ réceptif local. La couche de sous-échantillonnage suivante (S2) est composée de 08 cartes de caractéristiques de taille 125 × 144 × 26, chacune étant connectée à une carte de fonctionnalité dans C1. La couche de convolution (C3) contient 16 cartes de caractéristiques de taille 117 × 136 ×18 pixels et chaque unité est connectée à un 9 × 9 ×9 champ réceptif dans la couche précédente. La couche S4 suit le même principe que S2. Enfin, l'information d'entrée est encodée dans un vecteur de taille 128 dans la couche FC1. Ce vecteur peut être interprété comme un descripteur de l'information spatio-temporelle extraite de la séquence d'entrée. La couche Softmax contient un perceptron multicouche classique avec un neurone par une action humaine dans la couche de sortie. La configuration détaillée est indiquée dans le tableau 5.3.

C1	**S2**	**C3**	**S4**	**FC1**	**Softmax**
8x7x6x5	Pool 2x2x1	16x9x9x9	Pool 4x4x4	128x1x1	12x1
Pad –	-	1x1x2	-	-	-
Dropout –	-	0.8	-	0.8	-
Stride 1	Stride 2	Stride 1	Stride 4	-	-

Tableau 5.3: configuration du réseau de neurones convolutionnel 3D

La première ligne désigne le nombre de filtres, la hauteur, la largeur et la dimension temporelle de chaque filtre. Stride indique les intervalles pour appliquer le noyau de convolution à l'entrée. Pad indique le nombre de pixels à ajouter de chaque côté de l'entrée.

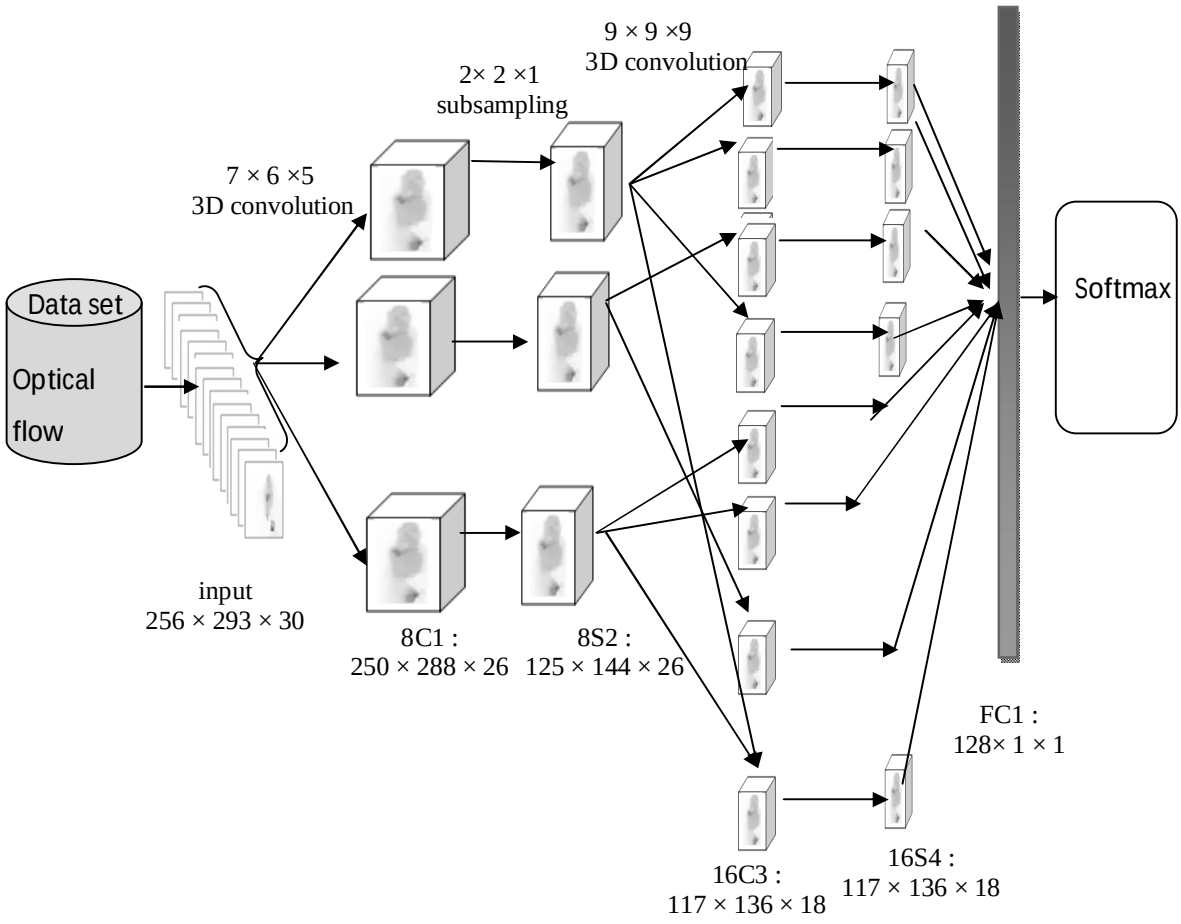


Figure 5.5 : Architecture 3D CNN pour la reconnaissance de l'action humaine

5.6.5 Apprentissage des réseaux de neurones convolutionnels 3D

L'apprentissage du modèle *CNN3D*, à chaque itération, est effectué en prenant en entrée 90 séquences de flot optiques de longueur trente. L'apprentissage est réalisé à l'aide de l'algorithme du gradient stochastique avec momentum adapté au partage de poids (voir chapitre 03). Le momentum utilisé égal à 0,9 et le taux d'apprentissage égal à 10^{-5} à l'instant t = 0, puis divisons les taux d'apprentissage de 2 et formons le modèle pour 50 autres itérations. L'apprentissage est arrêté lorsque le taux d'apprentissage est inférieur à $5x10^{-6}$. Pour réduire l'effet de sur-apprentissage du réseau, nous utilisons la technique dropout. Pour l'évaluation, nous utilisons la fonction softmax sur le vecteur de caractéristiques de la couche FC1 .

5.6.6 Procédure d'évaluation et résultats de classification

Le premier test que nous expliquons est basé sur l'ensemble de données d'action utilisé dans Gao et *al* [40]. Cet ensemble de données contient douze types d'actions humaines : one hand wave (wave1), multiple hands wave (wave2), handclap, walk, jog, jump, skip, handshake, hug, push, punch et fight.

Pour attribuer une classe d'action à chaque séquence de test, nous avons effectué la classification par les réseaux de neurones convolutionnels 3D, nous avons formé les classificateurs d'actions sur les 1680 séquences vidéo et nous avons testé sur les 60 séquences vidéo. Ensuite, nous avons répété cette procédure pour toutes les actions et nous avons calculé la moyenne pour obtenir le taux de classification moyen.

La figure 5. 6 illustre la matrice de confusion obtenue après l'utilisation de l'algorithme de filtrage (algorithme 1) sur l'ensemble de données d'apprentissages et de tests. La précision moyenne est de 88,19% lorsque nous utilisons 30 images espacées uniformément dans le temps pour chaque séquence.

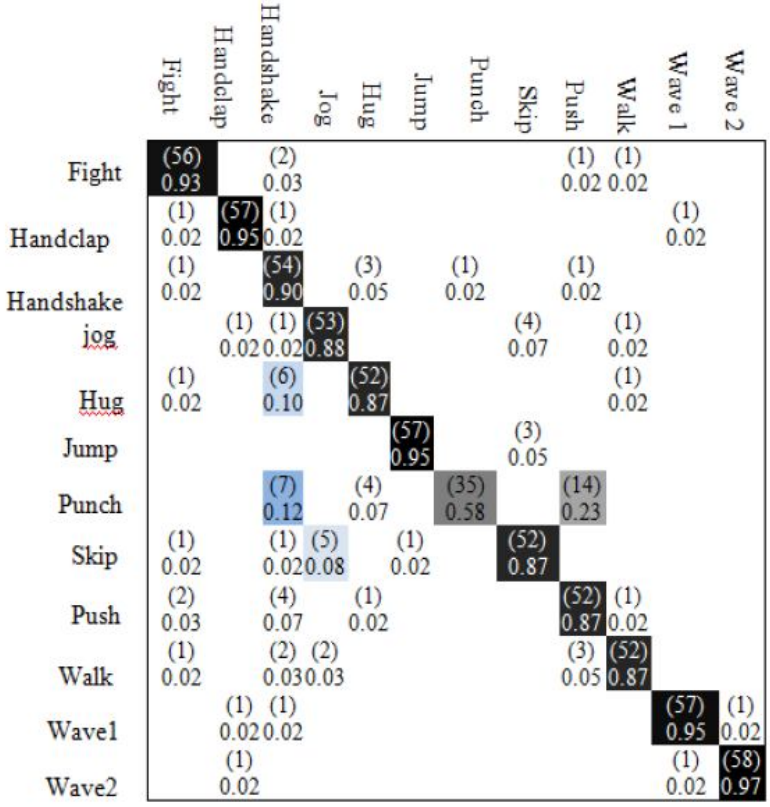


Figure 5.6 Matrice de confusion sur le jeu de données InfAR filtré

On peut constater que l'action punch a une précision faible, car cette action est parfois confondue avec les actions push, handshake et hug. Le taux de classification obtenu dans ce travail est considérablement plus élevé que celle obtenue dans [40], qui était de 76.66%. Cette amélioration est due au filtrage des cartes de flot optique sur les données de tests et d'apprentissages (suppression des séquences vidéo qui ne contiennent pas de l'action).

Dans la matrice de confusion représentée sur la figure 5.7, on peut remarquer que lorsque les données d'apprentissage et de test ne sont pas filtrés on a atteint un taux de classification de 53,88% qui est considéré plus faible par rapport au données d'apprentissage filtrés et de test qui ne sont pas filtrés (figure 5. 8), ce qui donne un taux de classification de 77,5% de même que les données de test sont filtrés et d'apprentissage ne sont pas filtrés (figure 5.9) , qui donne aussi un taux de classification de 73,61% .

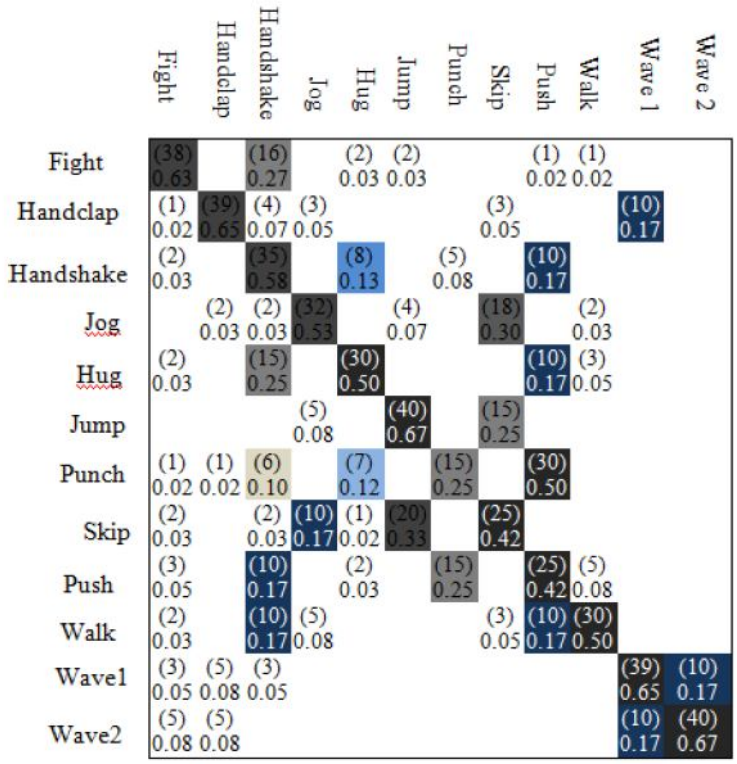


Figure 5.7 Matrice de confusion pour les données de test et d'apprentissage non filtrés

On peut aussi remarquer que le taux de classification a augmenté de 3,89%, atteignant 77,5% (figure 5.8) lorsque les données d'apprentissages sont filtrées par rapport à la figure 6.9 qui représente la matrice de confusion lorsque les données d'apprentissage ne sont pas filtrées.

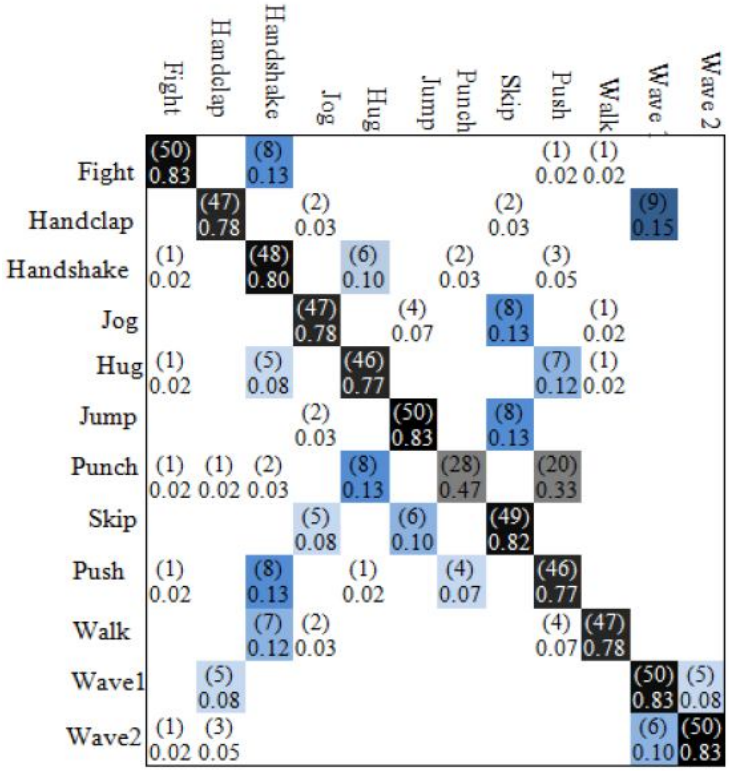


Figure 5.8 Matrice de confusion pour les données d'apprentissage filtrés

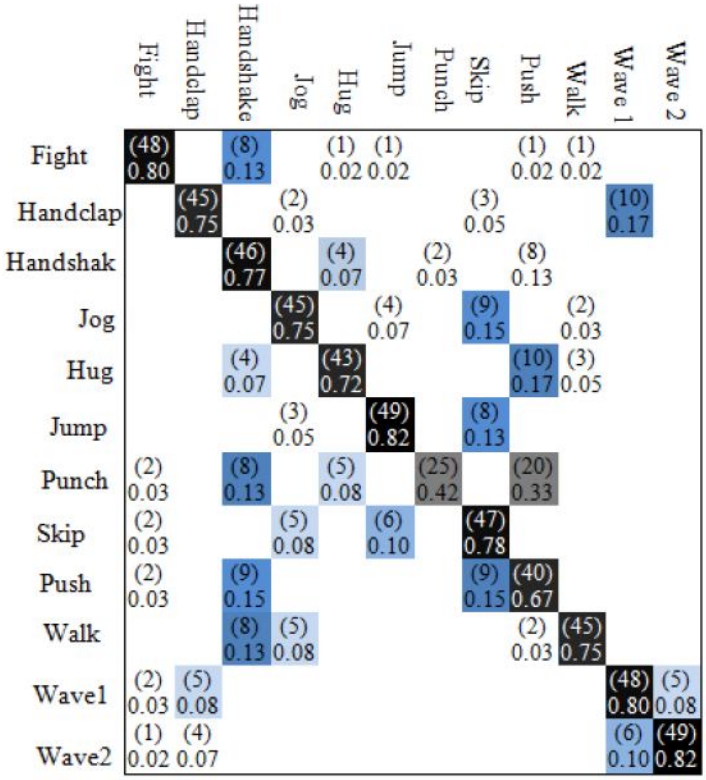


Figure 5.9 Matrice de confusion pour les données de test filtrés

5.7 Étude comparative

Le tableau 5.4 représente une étude comparative entre nos résultats et les approches de l'état de l'art données par [40] (appliquées sur la même base de données) . Nous observons que le taux de classification est le plus élevé (88,19%), soit environ 11,53 points par rapport au Tow- stream CNN (76,66%). Le filtrage des cartes de flot optiques est important pour la classification des séquences vidéo.

Méthode	Taux de classification (%)
STIP [40]	49.16
3DSIFT [40]	49.50
HOF [40]	68.58
Two-stream without OFMHI [40]	32.08
Tow- stream CNN [40]	76.66
Notre méthode	**88.19**

Tableau 5.4 : Taux de classification des différentes méthodes.

5.8 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode pour filtrer les cartes de flot optique et un système de classification de l'action humaine basé sur les réseaux de neurones convolutionnels (CNN) 3 D qui exploite l' information de mouvement après le filtrage des cartes de flot optique. La méthode de filtrage est basée sur le calcul de l'entropie entre deux séquences de 30 trames consécutives. Nous avons également testé l'algorithme de filtrage sur les cartes de flot optique pour filtrer les cartes qui ne contiennent pas de l'information (l'action). La technique proposée a donné de très bons résultats en classant toutes les actions de l'ensemble de données InfAR, atteignant un taux de classification de 88,19%.

CONCLUSION GENERALE ET PERSPECTIVES

La recherche de l'information utile dans une vidéo est importante. Diverses recherches ont été entreprises dans le domaine de traitement des données multimédia en vue de faciliter l'accès aux données importantes. Nous citons la détection des séquences d'informations intéressantes, la création des résumés et le filtrage du contenu.

Un document vidéo est une production de l'activité humaine d'où la nécessité de développer certains outils pour faciliter son traitement. Les données multimédia sont composées de données audio, visuelles et textuelles synchronisées. Par conséquent, le traitement de la vidéo est réalisé par des outils spécifiques qui respectent sa structure spatio-temporelle.

Dans ce travail, nous nous sommes intéressés à la reconnaissance d'actions à partir de séquences vidéo. Pour traiter ce problème, nous avons développé une approche de reconnaissance fondée sur un algorithme de filtrage en suivant une démarche hiérarchique à trois niveaux. Le premier niveau concerne l'exploitation de l'aspect temporel dans la séquence vidéo en utilisant l'estimateur de flot optique calculé entre deux images consécutives. Le niveau intermédiaire permet de filtrer les cartes de flot optique en se basant sur l'algorithme de filtrage proposé dans le chapitre 05. Le dernier niveau a pour but de classifier les actions en utilisant l'approche des réseaux de neurones convolutionnels 3D.

Dans l'étape d'exploitation de l'aspect temporel dans les séquences vidéo, nous avons utilisé la technique de flot optique qui consiste à évaluer le déplacement de points dans une séquence d'images au cours du temps. L'évaluation du flot optique

consiste à mettre en correspondance un point entre deux images successives, correspondant au même point de la scène 3D se déplaçant dans une séquence.

L'étape intermédiaire permet de filtrer les cartes de flot qui ne contient pas de l'information en se basant sur le calcul de l'entropie afin de réduire le temps de calcul et d'augmenter le taux de classification. Softmax est utilisé comme classifieur à la sortie de CNN3D afin de convertir le vecteur FC en vecteurs de probabilités conditionnelles des classes d'actions. La technique proposée a donné un taux de classification de 88,19%. La comparaison avec les méthodes de l'état de l'art a montré que notre méthode est prometteuse, car malgré la simplicité de l'algorithme de filtrage, les résultats obtenus sont comparables avec des méthodes basées sur des représentations de CNN à deux flot (two stream cnn) plus complexes.

En perspective pour la suite des travaux, nous avons proposé des améliorations qui pourront être intégrées pour améliorer le taux de reconnaissance et de réduire le temps de calcul. L'une des voies d'amélioration serait l'intégration d'un Réseau convolutif à deux flot pour améliorer le taux de reconnaissance.
Nous proposons aussi d'utiliser des bases plus grandes et difficiles telles que Hollywood2 [92] et HMDB [73]. De plus une extension de nos travaux vers la localisation et détection des actions sur la base InfAR est envisageable.

Bibliographie

[1] Anandan, P. A computational framework and an algorithm for the measurement of visual motion, dans *International Journal of Computer Vision*, vol 2 pp. 283–310, 1989.

[2] Ange Mikaël Mousse. Reconnaissance d'activités humaines à partir de séquences multi-caméras : application à la détection de chute de personne. Thèse de doctorat , Université du Littoral Côte d'Opale France, 2016.

[3] Aubert , G., Deriche, R., Kornprobst, P.Computing optical flow via variational techniques, dans *SIAM Journal on Applied Mathematics*, vol 60(1) , pp. 156–182, 1999.

[4] Baccouche, M. Apprentissage neuronal de caractéristiques spatio-temporelles pour la classification automatique de séquences vidéo, dans *thèse doctorat*, Institut National des Sciences Appliquées de Lyon, France, 2013.

[5] Bay, H., Tuytelaars, T., and Gool, L. V. Surf : Speeded up robust features, dans *ECCV*, pp 404–417, 2006.

[6] Belhachmi, Z., Hecht, F. Control of the effects of regularization on variational optic flow computations, dans *Journal of Mathematical Imaging and Vision*, vol 40, pp.1–19, 2011.

[7] Belhachmi, Z ., Hecht, F. An adaptive approach for segmentation and TV denoising in the optic flow estimation, dans *Journal of Mathematical Imaging and Vision*, vol 54, 2015.

[8] Bengio, Y., Simard, P., Frasconi, P. Learning long-term dependencies with gradient descent is difficult, dans *IEEE Transactions on Neural Networks*, vol 5(2), pp.157–166, 1994.

[9] Bernard, C. P., Wavelets and ill-posed problems: optic flow estimation and scattered data interpolation, dans *PhD Thesis*, Ecole Polytechnique, France, 1999.

[10] Bernard, C.P. Discrete wavelet analysis for fast optic flow computation, dans *Applied and Computational Harmonic Analysis*, vol11(1), pp.32–63, 2001.

[11] Black , M.J. Robust incremental optical flow, dans PhD thesis, Yale university, 1992.

[12] Blank, M., L. Gorelick, E. Shechtman, M. Irani et R. Basri. Actions as space-time shapes, *dans Internationale Conference on Computer Vision*, pp. 1395-1402, 2005.

[13] Bobick, A. F. and Davis, J. W. The recognition of human movement using temporal templates, dans *IEEE Trans. Pattern Anal. Mach. Intell*, vol 23(3), pp 257–267, 2001.

[14] Bouaziz, M. Réseaux de neurones récurrents pour la classification de séquences dans des flux audiovisuels parallèles, dans *thèse doctorat* , Université d'Avignon, France ,2017.

[15] Bregonzio, M., Gong, S., and Xiang, T. Recognising action as clouds of space-time interest points, dans *CVPR*, 2009.

[16] Brox, T., Malik, J. Large displacement optical flow : Descriptor matching in variational motion estimation, dans *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol 33, pp.500–513, 2011.

[17] Bruhn, A., Weickert, J., Feddern, C., Kohlberger, T., Schnörr, C. Real-time optic flow computation with variational methods, dans *Computer Analysis of Images and Patterns*, Springer Berlin Heidelberg, vol 2756, pp. 222–229, 2003.

[18] Bruhn, A., Weickert, J., Feddern, C., Kohlberger, T., Schnorr, C. Variational optical flow computation in real time, dans *IEEE Transactions on Image Processing*, vol 14(5), pp. 608-615, 2005.

[19] Bruhn, A., Weickert, J., Schnorr, C. Lucas/Kanade meets Horn/Schunck : Combining local and global optic flow methods, dans *International Journal of Computer Vision*, vol 61(3), pp.211–231, 2005.

[20] Bruhn, A.Variational optic flow computation : Accurate modelling and efficient numerics , dans *PhD thesis*, Department of Mathematics and Computer Science, Saarland University, Saarbrucken, 2006.

[21] Carreira**, J .,** Zisserman**,** A. Quo vadis, action recognition? a new model and the kinetics dataset, dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 6299-6308, 2017.

[22] Chen, Z., Jin, H., Lin, Z., Cohen, S., Wu, Y. Large displacement optical flow from nearest neighbor fields, dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2443–2450, 2013.

[23] Cheng, H., Liu, Z., Zhao, Y., Ye, G. Real world activity summary for senior home monitoring, dans *International Conference Multimedia and Expo*, pp. 1-4, 2011.

[24] Cheng, J., Dong, L., Lapata, M. Long short-term memory networks for machine reading, dans *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp.551–561, 2016.

[25] Cohen, I. Nonlinear variational method for optical flow computation, dans *Proceedings of the Scandinavian conference on image analysis*, vol 1, pp. 523–530, 1993.

[26] Cybenko, G. Approximation by superpositions of a sigmoidal function, dans *Math. Control Signal Systems* vol 2**,** pp. 303–314, 1989.

[27] Dai, C., Liu, X., Lai, J. Human action recognition using two-stream attention based LSTM networks, dans *Applied Soft Comput,* vol 86, 2020.

[28] Dollar, P., Rabaud, V., Cottrell, G., and Belongie, S. Behavior recognition via sparse spatio-temporal features,dans *Proceedings of the 14th International Conference on Computer Communications and Networks, ICCCN '05,* pp 65–72, 2005.

[29] Dérian, P., Héas, P., Herzet, C., Mémin, E. Wavelets and Optical Flow Motion Estimation, dans *Numerical Mathematics: Theory, Methods and Applications*, *Global Science Press*, vol 6, pp.116-137, 2013.

[30] Diba, A., Fayyaz, M., Sharma, V., Karami, A.H., Arzani, M.M., Yousefzadeh, R.,Van Gool, L. Temporal 3D convnets : New architecture and transfer learning for video classification , dans *arXiv preprint*, arXiv:1711.08200, 2017.

[31] Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M.,Venugopalan, S., Saenko, K., Darrell, T. Longterm recurrent convolutional networks for visual recognition and description, dans *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol39(4), pp. 677-691, 2017.

[32] Efros, A. A., Berg, A. C., Mori, G., and Malik, J. Recognizing action at a distance, dans *Proceedings of the Ninth IEEE International Conference on Computer Vision* ,Vol 2, p 726, 2003.

[33] Fan, Y., Weng, S., Zhang, Y., Shi, B., Zhang, Y. Context-Aware Cross-Attention for Skeleton-Based Human Action Recognition, dans *IEEE Access*, vol 8, pp. 15280-15290, 2020, doi: 10.1109 / ACCESS.2020.2968054.

[34] Feichtenhofer, C., Pinz, A., Wildes, R.P. Spatiotemporal multiplier networks

for video action recognition, dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 4768-4777,2017.

[35] Fischer, T., Krauss, C. Deep learning with long short term memory networks for financial market predictions, dans *European Journal of Operational Research,* vol 270(2), pp. 654-669, 2018.

[36] Fleet, D.J., Jepson, A.D. Computation of Component Image Velocity from Local Phase Information, dans *International Journal of Computer Vision,* vol 5, pp. 77–104, 1990.

[37] Fujiyoshi, H. and Lipton, A. J. Real-time human motion analysis by image skeletonization, dans *Proceedings of IEEE WACV98*, pp 15–21, 1998.

[38] Fukushima, K. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, dans *Biological Cybernetics*, vol 36(4), pp. 193–202, 1980.

[39] Funke**,** I., Bodenstedt**,** S., Oehme, F., Bechtolsheim, F.V., Weitz, J., Speidel, S. Using 3D Convolutional Neural Networks to Learn Spatiotemporal Features for Automatic Surgical Gesture Recognition in Video, dans *arXiv preprint* arXiv: 1907.11454. 2019.

[40] Gao, C., Du, Y., Liu, J., Lv, J.,Yang, L.,Meng, D., Hauptmann, A.G. InfAR dataset : Infrared action recognition at diferent times,dans *Neurocomputing*, vol 212, pp. 36–47, 2016.

[41] Gavrila, D. M. et L. S. Davis. Towards 3-d model-based tracking and recognition of human movement : a multi-view approach, dans *International Workshop on Automatic Face and Gesture-Recognition*, pp 272-277, 1995.

[42] Ghorbel, E., Boutteau, R., Boonaert, J., Savatier, X., Lecoeuche, S. A fast and accurate motion descriptor for human action recognition applications, dans 23rd *International Conference on Pattern Recognition (ICPR)*, pp 919-924, 2016.

[43] Ghorbel , E., Boonaert, J., Boutteau, R., Lecoeuche, S . An extension of kernel learning methods using a modified Log-Euclidean distance for fast and accurate skeleton-based Human Action Recognition, dans *Computer Vision and Image Understanding* Vol 175, pp 32-43, 2018.

[44] Ghorbel, E., Boutteau, R., Boonaert, J., Savatier, X., Lecoeuche, S. Kinematic spline curves: A temporal invariant descriptor for fast action recognition. dans *Image and Vision Computing* vol 77, pp 60–71, 2018.

[45] Gordon, D.A. Static and dynamic visual fields in human space perception,

dans *journal of the optical society of america*, vol 55(10) , pp. 1296–1303, 1965.

[46] Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. Actions as space-time shapes, dans *Transactions on Pattern Analysis and Machine Intelligence*, vol 29(12) , pp 2247–2253, 2007.

[47] Guichard, F., Rudin, L. Accurate estimation of discontinuous optical flow by minimizing divergence related functionals, dans IEEE *Image Processing, Proceedings, International Conference*, vol 1, pp. 497–500, 1996.

[48] Gupta, S.N., Prince, J.L. On div-curl regularization for motion estimation in 3-d volumetric imaging, dans *Image Processing, Proceedings., International Conference*, vol 1, pp. 929–932 , 1996.

[49] Han, L., Wu, X., Liang, W., Hou, G., and Jia, Y. Discriminative human action recognition in the learned hierarchical manifold space, dans *Image and Vision Computing*, vol28(5), pp 836 – 849, 2010.

[50] Haralick, R.M., Lee, J.S. The facet approach to optic flow, dans *Technical report,Virginia Polytechnic Inst and State Univ Blacksburg Dept Of Computer Science*, 1983.

[51] Harris, C. et Stephens, M. A Combined Corner and Edge Detection, dans *Proceedings of The Fourth Alvey Vision Conference*, pp 147-151, 1988.

[52] He, D., Zhou, Z., Gan, C., Li, F., Liu, X., Li, Y., Wang, L., Wen, S. Stnet: Local and global spatial-temporal modeling for action recognition, dans *arXiv preprint* arXiv:1811.01549, 2018.

[53] He, T., Mao, H., Yi, Z . Moving object recognition using multi-view three-dimensional convolutional neural networks, dans *Neural Computing and Applications*, *Springer*, vol 28(12), pp 3827– 3835, 2017.

[54] Hirtz, D.G. Techniques variationnelles et calcul parallèle en imagerie : Estimation du flot optique avec luminosité variable en petits et larges déplacements, dans *thèse de doctorat*,Université de Haute Alsace - Mulhouse, france, 2016.

[55] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J. Gradient flow in recurrent nets : the difficulty of learning long-term dependencies, dans *A field guide to dynamical recurrent neural networks. IEEE Press*, vol 1, pp. 237–243,2001.

[56] Hochreiter, S., Schmidhuber, J. Long Short-Term Memory, dans *Neural computation*, vol9(8), pp.1735–1780, 1997.

[57] Hopfield, J. J. Neural networks and physical systems with emergent collective

computational abilities, dans *Proceedings of the National Academy of Sciences* , vol 79, pp. 2554-2558, 1982.

[58] Horn, B.K., Schunck , B.G. Determining optical flow, dans *Artificial Intelligence*, vol 17, pp. 185–203, 1981.

[59] Hu, J., Shen, L., Sun, G. Squeeze-and-Excitation Networks, dans *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, pp. 7132-7141, 2018.

[60] Huang, K. S. and Trivedi, M. M. 3d shape context based gesture analysis integrated with tracking using omni video array, dans *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2005.

[61] Hubel , D. H., Wiesel, T. N. Receptive fields and functional architecture of monkey striate cortex, dans *The Journal of physiology*, vol 195(1) , pp. 215–243, 1968.

[62] Ioffe , S., Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift, dans *arXiv preprint* arXiv:1502.03167, 2015.

[63] Ji, S., Xu, W., Yang, M., Yu, K. 3d convolutional neural networks for human action recognition, dans *IEEE transactions on pattern analysis and machine intelligence* vol 35(1), pp.221–231, 2013.

[64] Johansson, G. Visual perception of biological motion and a model for its analysis, dans *Perception & Psychophysics*, vol 14 , pp 201–211, 1973.

[65] Kadir, T., Brady, M. Scale saliency : A novel approach to salient feature and scale selection, dans " *International Conference on Visual Information Engineering*., pp 25–28, 2003.

[66] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei , L. Large-scale video classication with convolutional neural networks, dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[67] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A. The kinetics human action video dataset, dans *arXiv preprint* arXiv:1705.06950, 2017.

[68] Ke, Y., R. Sukthankar et M. Hebert. 2010, Volumetric features for video event detection, dans *International Journal of Computer Vision*, vol 88, pp 339-362, 2010.

[69] Kläser,A., Marszałek, M and Schmid, C. A spatio-temporal descriptor based on 3Dgradients. In BMVC, 2008.

[70] Kohonen, T . Self-organized Formation of Topologically Correct Feature Maps, dans *Biol. Cybern* , vol 43, pp . 59-69 , 1982.

[71] Köpüklü, O., Kose, N., Gunduz, A., Rigoll, G . Resource Efficient 3D Convolutional Neural Networks, dans *arXiv preprint* arXiv: 1904.02422. 2019

[72] Krizhevsky, A., Sutskever, I., Hinton, G.E. Imagenet classification with deep convolutional neural networks, dans *Advances in neural information processing systems*, pp 1097–1105, 2012.

[73] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T. HMDB:A large video database for human motion recognition, dans *Internationale Conference on Computer Vision,* pp. 2556-2563, 2011.

[74] Kumar, A., Tannenbaum, A.R.., Balas, G.J. Optical flow : a curve evolution approach, dans *Image Processing, IEEE Transactions*, vol 5, pp.598–610, 1996.

[75] Laptev, I., Caputo, B., Schüldt, C., and Lindeberg, T. Local velocity adapted motion events for spatio-temporal recognition, dans *Computer Vision and Image Understanding*, Vol. 108(3) , pp 207–229, 2007.

[76] Laptev, I., Lindeberg,T. Space-time interest points, dans *Internationale Conference on Computer Vision*, pp 432-439, 2003.

[77] Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B. Learning realistic human actions from movies, dans *Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[78] LeBesnerais, G., Champagnat, F. Dense optical flow by iterative local window registration, dans *IEEE International Conference on Image Processing*, Genova, Italy , pp. 137-140 , *2005 .*

[79] LeCun, Y., Bengio, Y. Convolutional networks for images, speech, and time series, dans *The handbook of brain theory and neural networks*, vol 3361(10), p. 1995, 1995.

[80] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. Gradient-based learning applied to document recognition, dans *Proceedings of the IEEE*, vol 86(11), pp. 2278–2324, 1998.

[81] Leong, M.C., Prasad, D.K., Lee, Y.T., Lin, F. Semi-CNN Architecture for Effective Spatio-Temporal Learning in Action Recognition, dans *Applied*

Sciences, 10(557) doi:10.3390/app10020557, 2020.

[82] Li , D., Qian, J. Text sentiment analysis based on long short term memory. Dans *les actes de International Conference on Computer Communication and the Internet (ICCCI), IEEE* , pp. 471–475, 2016.

[83] Lim, A., Ramesh, B., Yang, Y., Xiang, C., Gao, Z., Lin, F. Real-time optical flow-based video stabilization for unmanned aerial vehicles, dans *Journal of Real-Time Image Processing*, *Springer*, vol 16(6), pp. 1975-1985,2019.

[84] Lin, T., Horne, B.G., Tino, P., Giles, C.L. Learning long-term dependencies in NARX recurrent neural networks, dans *IEEE Transactions on Neural Networks*,vol7(6), pp.1329–1338, 1996.

[85] Liu, J., Luo, J., Shah, M. Recognizing realistic actions from videos in the wild dans *Conference on Computer Vision and Pattern Recognition*, pp. 1996-2003, 2009.

[86] Liu, H., Tu, J., Liu ., M. Two-Stream 3D Convolutional Neural Network for Human Skeleton-Based Action Recognition. dans *arXiv preprint: 1705.08106*, 2017.

[87] Liu, J ., Wang , G., Duan , L-Y., Abdiyeva, K., Kot, A. Skeleton-Based Human Action Recognition With Global Context-Aware Attention LSTM Networks, dans *IEEE Transactions on Image processing*, vol 27(4), pp 1586–1599 , 2018.

[88] Longuet-Higgins, H.C., Prazdny, K. The interpretation of a moving retinal image, dans *Proceedings of the Royal Society of London B : Biological Sciences,* vol 208 (1173) , pp. 385–397, 1980.

[89] Lowe, David G. Object recognition from local scale-invariant features, dans *Proc. 7th International Conference on Computer Vision (ICCV'99)* , vol 2, pp 1150-1157, 1999.

[90] Lowe, David G. Distinctive image features from scale-invariant key points, dans *International Journal of Computer Vision* , vol 60(2), pp 91-110, 2004.

[91] Lucas , B.D., Kanade, T. An iterative image registration technique with an application to stereo vision, dans *Proceedings of the 7th International Joint Conference on Artificial Intelligence* , Morgan Kaufmann Publishers , vol 2, pp. 674–679, 1981.

[92] Marszalek, M., Laptev, I., Schmid, C. Actions in context, dans *Conference on Computer Vision and Pattern Recognition*, pp. 2929-2936, 2009.

[93] Marzat, J (2008). Estimation temps réel du Flot Optique, dans *rapport technique d' Institut National Polytechnique de Lorraine*, 2008.

[94] Matricon, A. Regroupement de compétences robotiques en compétences plus générales, dans *thèse doctorat,* université de bordeaux, France, 2018.

[95] McCulloch, W., Pitts, W. A logical calculus of the ideas immanent in nervous activity, dans *Bulletin of Mathematical Biophysics*, vol 5, pp. 115-133, 1943.

[96] Mehdi ,A. Réalisation d'un réseau de neurones "SOM" sur une architecture matérielle adaptable et extensible à base de réseaux sur puce "NoC". Micro et nanotechnologies / Microélectronique, dans *thèse doctorat* Université de Lorraine Français; Université du Centre (Sousse, Tunisie), 2018.

[97] Mumford, D., Shah, J .Optimal approximations by piecewise smooth functions and associated variational problems, dans *Communications on pure and applied mathematics*, vol 42(5), pp. 577–685, 1989.

[98] Nagel, H.H. On the estimation of optical flow : Relations between different approaches and some new results, dans *Artificial Intelligence*, vol 33(3), pp. 299–324, 1987.

[99] Nesi, P. Variational approach to optical flow estimation managing discontinuities, dans *Image and Vision Computing*, vol 11(7), pp. 419–439, 1993.

[100] Oikonomopoulos, A., Patras, I., and Pantic, M.. Spatiotemporal saliency for human action recognition, dans *IEEE Trans. Syst., Man Cybern*. B, vol. 36, pp. 710–719, 2006.

[101] Papenberg , N., Bruhn, A., Brox, T., Didas, S., Weickert, J. Highly accurate optic flow computation with theoretically justified warping, dans *International Journal of Computer Vision*, vol 67(2), pp. 141–158, 2006.

[102] Patron-perez, A., Marszalek, M., Zisserman, A., Reid, I. High five : Recognising human interactions in tv shows, dans *British Machine Vision Conference*, pp. 1-11, 2010.

[103] Qiu, Z., Yao, T., Mei, T. Learning spatio-temporal representation with pseudo-3d residual networks, dans Proceedings of the *IEEE International Conference on Computer Vision (ICCV)*, pp. 5534–5542, 2017.

[104] Ragheb, H., Velastin, S. A., Remagnino, P., and Ellis, T, dans *ICIP, IEEE* , pp. 753–756, 2008.

[105] Ramanan, D. and Forsyth, D. A. Automatic annotation of everyday movements, dans *Thrun, S., Saul, L. K., and Schölkopf, B., editors, NIPS. MIT Press*, 2003.

[106] Rapantzikos K., Tsapatsoulis N., Avrithis Y., "Spatiotemporal Visual Attention Architecture for Video Analysis, dans *Proc. of IEEE International Workshop On Multimedia Signal Processing (MMSP'04)*, Sienna, 2004.

[107] Rapantzikos K., Avrithis Y., Kollias S., Dense saliencybased spatiotemporal feature points for action recognition, dans *IEEE Conference on Computer Vision and Pattern Recognition*, pp 1454–1461, 2009.

[108] Raptis, M. and Sigal, L. Poselet key-framing : A model for human activity recognition, dans *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pp 2650–2657,2013.

[109] Robertson, N. M. and Reid, I. D. (2005). Behaviour understanding in video : A combined method, dans *IEEE International Conference on Computer Vision (ICCV'05)* Vol 1, pp 808–815, 2005.

[110] Rodriguez, M.D., J. Ahmed ., Shah, M. Action mach : A spatio-temporal maximum average correlation height filter for action recognition, dans *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[111] Rosenblatt, F.The Perceptron: a probabilistic model for information storage and organization in the brain, dans *Psychological Review*, vol 65, pp. 386-408, 1958.

[112] Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning internal representations by error propagation, dans *Parallel distributed processing: explorations in the microstructure of cognition*, vol 1, pp. 318–362, 1986.

[113] Scovanner, P., Ali, S., and Shah, M. A 3-dimensional sift descriptor and its application to action recognition, dans *Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA '07*, pp 357–360, 2007.

[114] Schuldt, C., Laptev, I., Caputo, B. Recognizing human actions: a local svm approach, dans Pattern Recognition,Proceedings of the 17th IEEE International Conference vol 3, pp. 32–36, 2004.

[115] Selmi Mouna. Reconnaissance d'activités humaines à partir de séquences vidéo. Thèse de doctorat, Institut National des Télécommunications France, 2014.

[116] Shechtman, E. et M. Irani. Space-time behavior based correlation, dans *Conference on Computer Vision and Pattern Recognition*, pp 405-412, 2005.

[117] Shotton, J., A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman et A. Blake. Real-time human pose recognition in parts from a single

depth image, dans *Conference on Computer Vision and Pattern Recognition*, pp 1297-1304, 2011.

[118] Simoncelli, E. P. Bayesian multi-scale differential optical flow, dans *Handbook of Computer Vision and Applications* (Haussecker, Jähne, and Geissler, Eds.), Academic Press, San Diego, 1998.

[119] Simoncelli, E.P,. Adelson , E.H. Computing optical flow distributions using spatio-temporal filters. Technical Report, 1991.

[120] Simoncelli, E.P., Adelson, E.H., Heeger, D.J. Probability distributions of optical flow, dans *Proc. Conf. Comput. Vis. Patt. Recog.* Maui, pp. 310–315, 1991.

[121] Simonyan, K., Zisserman, A. Two-stream convolutional networks for action recognition in videos, dans *Advances in neural information processing systems*, pp 568–576, 2014.

[122] Spinei, A. Estimation du mouvement par triades de filtres de Gabor. Applicationau mouvement de transparence, dans *PhD thesis*, Institut national polytechnique de Grenoble, France, 1998.

[123] Sun, L., Jia, K., Yeung, D.Y., Shi, B.E. Human action recognition using factorized spatio-temporal convolutional networks, dans *IEEE International Conference on Computer Vision (ICCV)*, pp. 4597-4605, 2015.

[124] Sundaram, N., Brox, T., Keutzer, K. Dense point trajectories by GPU accelerated large displacement optical flow, dans *Computer Vision – ECCV 2010, Springer Berlin Heidelberg* , vol 6311, pp. 438–451., 2010.

[125] Taylor G.W., Fergus R., LeCun Y., Bregler C. Convolutional learning of spatiotemporal features, dans *Proc. Inernational. Conference*. *IEEE European Conference on Computer Vision*, *Heraklion, Crete, Greece*, vol 6316, pp. 140–153,2010.

[126] Tenorth, M., Bandouch, J., Beetz, M. The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition, dans *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV*, pp. 1089-1096, 2009.

[127] Tran, A., Cheong, L. Two-Stream Flow-Guided Convolutional Attention Networks for Action Recognition, dans *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Venice, pp. 3110-3119, 2017.

[128] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M. Learning spatiotemporal features with 3d convolutional networks, dans *Proceedings of*

the IEEE International Conference on Computer Vision, pp. 4489-4497, 2015.

[129] Tran, D., Ray, J., Shou, Z., Chang, S., Paluri, M. Convnet architecture search for spatiotemporal feature learning, dans *arXiv preprint* arXiv:1708.05038, 2017.

[130] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M. A closer look at spatiotemporal convolutions for action recognition, dans *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6450-6459, 2018.

[131] Vemulapalli,R ., Arrate, Fand Chellappa, R., Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group, dans *IEEE Conference on Computer Vision and Pattern Recognition*, pp 588-595, 2014.

[132] Vincent, P. Modèles à noyaux à structure locale, dans *thèse doctorat*, Département d'informatique et de recherche opérationnelle Faculté des arts et des sciences, Université de Montréal, Canada, 2003.

[133] Vincent, L. Etude et conception de circuits innovants exploitant les caractéristiques des nouvelles technologies mémoires résistives, dans *Thèse de doctorat* de l'Université Paris-Saclay, préparée à l'Université Paris-Sud, France, 2018.

[134] Wang, H., Raj, B. On the Origin of Deep Learning , dans *arXiv preprint arXiv:1702.07800*, 2017.

[135] Wang, J.H., Liu, T.W., Luo, X., Wang, L. An LSTM Approach to Short Text Sentiment Classification with Word Embeddings, dans *The Conference on Computational Linguistics and Speech Processing ROCLING*, pp. 214-223, 2018.

[136] Wang, J., Liu, Z., Wu, Y., Yuan, J. Mining actionlet ensemble for action recognition with depth cameras, dans *Conference on Computer Vision and Pattern Recognition*, pp. 1290-1297, 2012.

[137] Wang, L., Li, W., Li, W., Van Gool, L. Appearance-and-Relation Networks for Video Classification, dans *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, pp. 1430-1439, 2018.

[138] Wang, Y., Long, M., Wang, J ., Yu, P.S. Spatiotemporal Pyramid Network for Video Action Recognition, dans *arXiv preprint* arXiv: :1903.01038v1 . 2019 .

[139] Wang, Y., Sabzmeydani, P., and Mori, G. Semi-latent dirichlet allocation : A hierarchical model for human action recognition, dans *2nd Workshop on Human Motion Understanding, Modeling, Capture and Animation*, 2007.

[140] Weinland, D., Ronfard, R., and Boyer, E. Free viewpoint action recognition using motion history volumes, dans *Computer Vision and Image Understanding,*

Elsevier, vol 104 (2-3), pp 249-257, 2006.

[141] Willems, G., Tuytelaars, T., and Gool, L. An efficient dense and scale-invariant spatio-temporal interest point detector, dans *Proceedings of the 10th European Conference on Computer Vision : Part II, ECCV '08*, pp 650–663, 2008.

[142] Wu, Y.T., Kanade, T., Cohn, J., Li, C-C. Optical Flow Estimation Using Wavelet Motion Model, dans *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Bombay, India, pp. 992-998, 1998.

[143] Xia, L., Chen, C.-C., and Aggarwal, J. K. View invariant human action recognition using histograms of 3d joints, dans *CVPR Workshops*, pp 20–27, 2012.

[144] Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification, dans *European Conference on Computer Vision (ECCV)* (Springer), pp. 305–321, 2018.

[145] Xiong, J. et Z. Liu. Human motion recognition based on hidden markov models, dans *Advances in Computation and Intelligence*, pp 464-471, 2007.

[146] Yilmaz, A. et M. Shah. A differential geometric approach to representing the human actions, dans *Computer Vision and Image Understanding*, vol 109(3), pp 335-351, 2008.

[147] Zhang, R., Chen, Z., Chen, S., Zheng, J., Büyüköztürk, O., Sun, H. Deep long short-term memory networks for nonlinear structurl seismic response prediction, dans *Computers and Structures* , vol 220, pp. 55-68, 2019.

[148] Zhen, X. and Shao, L. Spatio-temporal steerable pyramid for human action recognition, dans *10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2013.

[149] Zhou, Y., Sun, X., Zha, Z.J., Zeng, W. Mict. Mixed 3d/2d convolutional tube for human action recognition, dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* ,pp.449– 458, 2018.

[150] Ziaeefard, M. and Ebrahimnezhad, H. Hierarchical human action recognition by normalized-polar histogram, dans I*CPR'10*, pp 3720–3723, 2010.

[151] Zimmer, M. Apprentissage par renforcement développemental, dans *thèse doctorat* ,Université de Lorraine, France, 2018.