REPUBLIQUE ALGERIENNE
DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
École Nationale Polytechnique

Département d'Electronique

Mémoire du Projet de Fin d'Études

Pour l'obtention du diplôme d'ingénieur d'état en Electronique

Classification des arythmies cardiaquesen utilisant les réseaux de neurones profonds

Réaliser par :

Mr SEGHAIRI Issam et Mr BAAMARA Slimane

Présenté et soutenu publiquement le 07 juillet 2020

Devant le jury :

Dr. Sid-Ahmed BERRANI :	ENP Alger	- Président
Pr. Mourad ADNANE :	ENP Alger	- Encadreur
Dr. Nesrine BOUADJENEK :	ENP Alger	- Examinatrice

ENP 2020

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie

REPUBLIQUE ALGERIENNE
DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
École Nationale Polytechnique

Département d'Electronique

Mémoire du Projet de Fin d'Études

Pour l'obtention du diplôme d'ingénieur d'état en Electronique

Classification des arythmies cardiaquesen utilisant les réseaux de neurones profonds

Réaliser par :

Mr SEGHAIRI Issam et Mr BAAMARA Slimane

Présenté et soutenu publiquement le 07 juillet 2020

Devant le jury :

Dr. Sid-Ahmed BERRANI :	ENP Alger	- Président
Pr. Mourad ADNANE :	ENP Alger	- Encadreur
Dr. Nesrine BOUADJENEK :	ENP Alger	- Examinatrice

ENP 2020

10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie

ملخص

مخطط كهربية القلب (ECG) هو أداة تشخيصية مهمة لتقييم عدم انتظام ضربات القلب. إن تصنيف هذه الإشارات مهم للغاية في التشخيص التلقائي لأمراض القلب. تستخدم حاليا العديد من حلول التعلم الآلي لتحليل وتصنيف بيانات تخطيط القلب. ومع ذلك ، فإن أحد عيوب هذه الأساليب تكمن في صعوبة استخراج الميزات الحاسمة التي تسمح بالحصول على دقة عالية. أحد الحلول المقترحة في الأدبيات هو استخدام بنيات التعلم العميقة التي تستغل الطبقات الدستورية لاستخراج المعالم ، تليها طبقات متصلة بالكامل لصنع القرار.

الكلمات الرئيسية : تخطيط القلب ، الذكاء الاصطناعي ، الشبكات العصبية التلافيفية ، التعلم العميق

Abstract

The electrocardiogram (ECG) is an important diagnostic tool for the evaluation of cardiac arrhythmias. The classification of this signals is very important for the automatic diagnosis of heart disease. Currently many machine learning solutions are used to analyze and classify ECG data. However, one of the drawbacks of these methods lies in the difficulty of extracting the crucial features that allow attaining high precision. One of the solutions proposed in the literature is to use deep learning architectures that exploit convolutional layers for feature extraction, followed by fully connected layers for decision making.

keywords : Electrocardiogram, Artificial Intelligence, Convolutional Neural Networks, Deep Learning.

Résumé

L'électrocardiogramme (ECG) est un outil de diagnostic important pour l'évaluation des arythmies cardiaques. La classification de ces signaux est très importante pour le diagnostic automatique des maladies cardiaques. Actuellement, de nombreuses solutions d'apprentissage automatique peuvent être utilisées pour analyser et classer les données ECG. Cependant, l'un des inconvénients de ces méthodes réside dans la difficulté de trouver les caractéristiques les plus appropriées permettant d'avoir des précisions élevées. L'une des solutions proposées consiste à utiliser des architectures d'apprentissage profond dans lesquelles les premières couches de neurones convolutifs se comportent comme des extracteurs de caractéristiques automatiques et à la fin, des couches entièrement connectées sont utilisés pour la décision finale.

Mots clés : Electrocardiogramme, ECG, Intelligence Artificielle, Réseaux de neurones convolutifs, Apprentissage approfondi

Remerciement

MERCI !
C'est un petit mot tout simple mais qui pèse lourd
Un grand Merci, un petit Merci
Peu importe sa taille Il n'a pas de dimension...
Que ce soit dans la joie ou dans la tristesse !
C'est un signe de reconnaissance qui ne connait pas l'indifférence !
Merci !
Un petit mot qui fait du bien quand on le prononce, un petit mot gracieux qui calme et réjouit
Merci ! Merci !
Merci de nous avoir permis de vous dire : Merci !

Je tiens à remercier Dieu Tout-Puissant, de nous avoir aidés à terminer ce mémoire.

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon travail et qui m'ont aidée lors de la rédaction de ce mémoire.

Je voudrais dans un premier temps, remercier mon encadreur de mémoire M. Adnane Mourad, professeur à l'école nationale polytechnique, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Nous souhaitons adresser nos remerciements les plus sincères au corps professoral et administratif de l'école nationale polytechnique, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée

Un grand merci à ma mère et mon père, pour leur amour, leurs conseils ainsi que leur soutien inconditionnel, à la fois moral et économique, qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ECG : Electrocardiogram.
NOR : Normal Beat.
LBB : Left Bandle Beat.
RBB : Right Bandle BePACat.
PAC : Atrial Premature Beat.
PVC : Premature Ventricular Contraction.
VF : Ventricular Flutter Wave.
VE : Ventricular Escape Beat.
OD : Oreillette Droite.
OG : Oreillette Gauche.
VG : Ventricule Gauche.
VD : Ventricule Droit.
TRI : valve Tricuspide.
QRS : Complex formed by Q, R, and S waves of ECG signals.
P : Wave P.
NAV : Nœud Auriculo-Ventriculaire.
EMI : Electromagnetic Interference.
AV : Nœud Atrioventriculaire.
aVR : mesure unipolaire sur le bras droit.
aVL : mesure unipolaire sur le bras gauche.
aVF : mesure unipolaire sur la jambe gauche.
V1,V2V3,V4,V5,V6 : Dérivations précordiales.
DI,DII,DIII : Les dérivations périphériques bipolaires.
VL,VR,VF : Le V signifiant vecteur et R, L, F : droite, gauche et pied (en anglais).
ESV : Extrasystole Ventriculaire.
AI : Artificial Intelligence.
NLP : Natural Language Processing.
Relu : Rectified linear unit.
Tanh : Hyperbolic Tangent Functi MSE : Mean square error.
CE : Cross entropy.
MLP : Multilayer perceptron.
CNN : Convolutional Neural Network.
AE : Auto-encoder.
RNN : Recurrent neural network.
LSTM : Long short term memory.
GD,SGD : Gradient and stochastic gradient descent.
BLSTM : bidirectional long short term memory.
DWT : Discrete wavelet transform.

Introduction générale

Le signal de l'électrocardiogramme (ECG), qui est l'enregistrement de l'activité électrique cardiaque, fournit des informations importantes sur l'état du cœur. La détection des arythmies cardiaques est nécessaire pour le traitement des patients afin de diagnostiquer la maladie cardiaque à un stade précoce. Il est très difficile pour les médecins d'analyser de longs enregistrements ECG en peu de temps, et l'œil humain est également mal adapté pour détecter la variation morphologique du signal ECG, d'où la nécessité d'un système efficace d'aide au diagnostic assisté par ordinateur. L'importance de la classification automatique d'ECG est très importante en raison des nombreuses applications médicales actuelles où ce problème peut être rencontré. Citons l'exemple de « Smart Running Track » qui est une application de réseau de capteurs visant à améliorer la santé des personnes par l'observation des paramètres corporels des coureurs, tels que l'ECG pour observer son état cardiaque. Les applications Home care (systèmes de santé à domicile) représentent d'autres exemples d'applications. Dans ce cas, un appareil ECG câblé standard peut avoir un accès sans fil au centre médical, ce qui permettra une surveillance à distance [1].
L'analyse automatique du signal ECG est confrontée à un problème de taille en raison de la grande variation des caractéristiques morphologiques et temporelles des formes d'ondes ECG des déférents patients ainsi que des mêmes patients [2]. Les formes d'ondes ECG peuvent être différentes pour le même patient à différents moments et peuvent être similaires pour différents patients ayant différents types de battements. Il faut ajouter que l'enregistrement d'un ECG sur de longues périodes (24h) engendre un nombre conséquent de battements par dérivation (100000 battements pour un sujet sain). La tâche d'analyse manuelle devient très complexe pour le corps médical.
Actuellement, il existe de nombreuses solutions d'apprentissage machine(ML) qui peuvent être utilisées pour analyser et classifier les données ECG. Cependant, le principal inconvénient de ces méthodes ML est l'utilisation de caractéristiques extraites manuellement. Le problème réside dans la possibilité de ne pas trouver les caractéristiques les plus appropriées qui donneront une grande précision de classification. Une des solutions proposées consiste à utiliser des architectures d'apprentissage profond où les premières couches de neurones convolutionnelles se comportent comme des extracteurs de caractéristiques spatiales, et en fin de compte, certaines couches entièrement connectées (FCN) sont utilisées pour prendre la décision finale concernant les classes des battements cardiaques.
Une autre solution consiste à utiliser des architectures d'apprentissage profond où les premières couches sont les couches à longue mémoire LSTM pour la détection de caractéristiques temporelle présente dans le signal d'entrée, ainsi que des couches entièrement connectées (FCN) sont utilisées pour prendre la décision finale concernant les classes des battements cardiaques. Dans ce travail, trois architectures d'apprentissage profond pour la classification de l'ECG dans l'une des catégories d'arythmies cardiaques sont présentées. La première est basée sur les couches convolutionnelles 1D, tandis que la deuxième est basée sur les couches LSTM, et la dernière est une combinaison des deux architectures CNN-LSTM. Un autre critère est utilisé dans cette étude qui est le nombre de battements dans le signal d'entrée.
Le mémoire est organisé comme suit : le chapitre 1 est consacré au principes de l'élec-

trocardiogramme, ainsi que les différentes arythmies cardiaques. le chapitre 2 présente la partie théorique sur l'apprentissage profond. Des explications sur le fonctionnement des différentes architectures citées ci-dessus sont fournies. Dans la partie expérimentale, présentée en chapitre 3, les résultats d'entrainement et de test des différents modèles sélectionnés sont exposés ainsi que des métriques pour évaluer la performance de chaque modèle. Une synthèse de tout ce que nous avons pu faire et apprendre dans le cadre de cette étude fera l'objet d'une conclusion générale.

Chapitre 1

Principe de l'électrocardiogramme

1.1 Introduction

L'électrocardiogramme (ECG) est un examen médical qui enregistre l'activité électrique cardiaque et permet de détecter le cas échéant des anomalies. Le cœur produit de minuscules impulsions électriques qui se propagent à travers des fuseaux et stimulent les muscles cardiaques, ces impulsions sont affichées par l'ECG sous forme de tracé sur un papier, ce qui permet leurs interprétations par les médecins. À partir de l'aspect du tracé, les médecins peuvent trouver ou affirmer des diagnostics cardiaques chez des patients présentant ou non des symptômes ou des signes d'appelles cardiaques.

Les pathologies peuvent être diagnostiquées par l'ECG sont : les troubles de rythme cardiaque, les cardiopathies ischémiques.

Toute irrégularité du rythme cardiaque ou tout dommage au muscle cardiaque peut modifier l'activité électrique du cœur et modifier également l'aspect du tracé et donne un aspect au tracé caractéristique de chaque pathologie.

1.2 Le fonctionnement cardiaque

1.2.1 Anatomie du cœur

Le cœur est un organe formé de deux compartiments droit et gauche complètement séparés par le septum interauriculaire et interventriculaire. Chaque compartiment est composé d'une oreillette et d'un ventricule. l'oreillette droite (OD) et le ventricule droit (VD) se communiquent par l'orifice tricuspidien (TRI). L'oreillette gauche (OG) et le ventricule gauche (VG) se communiquent par l'orifice mitral, La configuration interne et les connexions vasculaires sont illustrées dans la figure (1.1) [3].

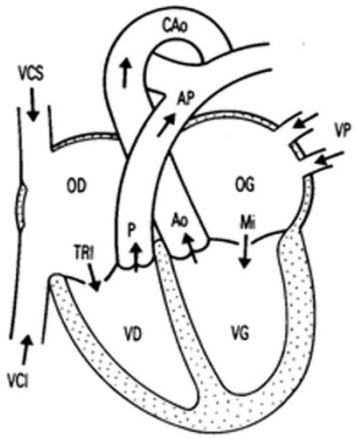


FIGURE 1.1: La configuration interne du cœur.

1.2.2 Activité mécanique du cœur

L'activité mécanique du cœur est générée par le flux de signaux électriques générés dans le nœud sinusal, se propage ensuite par le biais de fuseaux aux oreillettes puis aux ventricules. Ce flux de signaux électriques déclenche une série de contractions.
Les oreillettes se remplissent de sang venu de tout l'organisme pour l'oreillette droite, et des poumons pour l'oreillette gauche, ce qui augmente leur pressions. Cette forte pression ouvre les valves des orifices inter auriculo-ventriculaire, le signal électrique stimule les

oreillettes qui se contractent (Cela correspond à l'onde P sur le tracé de la figure (1.2)) ce qui provoque le flux sanguin vers les ventricules (intervalle de temps PR sur le tracé de la figure (1.2)). Ensuite les ventricules remplis de sang venant des oreillettes commencent à se contracter et leur pression augmente et dépasse la pression auriculaire, ce qui ferme les valves atrioventriculaires.
Afin d'ouvrir les valves aortique et pulmonaire, les ventricules continuent à se contracter (complexe QRS sur le tracé de la figure (1.2)), toujours grâce à la stimulation électrique qui vient du nœud sinusal et se propage à travers les fuseaux. Ainsi les pressions augmentent et deviennent supérieures à celles des troncs artériels aortique et pulmonaire ce qui permet d'ouvrir leurs valves, et le sang est expulsé dans les vaisseaux sanguins. Cette phase de contraction est appelée systole ventriculaire (segment QT). Les ventricules commencent à se détendre et à se relaxer (onde T), la pression baisse et la valve se referme. Lorsque toutes les valves sont fermées pendant un certain temps, la relaxation est iso volumétrique. Ensuite les oreillettes se remplissent de nouveau de sang et le cycle recommence. Toute la période de relaxation est appelée diastole ventriculaire.

1.2.3 Activité électrique du cœur

Le courant se produit en un point précis du cœur, appelé nœud sinusal (où naît l'automatisme cardiaque), situé au sommet de l'oreillette droite. Ce courant est généré par un échange d'ions spontané au travers de la membrane des cellules du tissu nodal permettant d'atteindre le potentiel seuil qui déclenche le potentiel d'action. Il s'ensuit une repolarisation. Chaque impulsion électrique (potentiel d'action) commence au niveau du nœud sinusal et se propage aux deux oreillettes jusqu'à leur base, provoquant leur contraction. Elle converge ensuite vers la cloison qui sépare les oreillettes des ventricules au niveau d'un relais électrique appelé le nœud auriculo-ventriculaire (NAV). À partir de ce dernier : l'influx progresse dans les deux ventricules, simultanément avec la progression du flux sanguin des oreillettes aux ventricules. L'activité électrique se propage du NAV en utilisant des voies de conduction extrêmement rapides (le faisceau His et le réseau de Purkinje) aux deux ventricules jusqu'à la pointe du coeur, ce qui provoque la contraction des ventricules. La figure (1.2) illustre le chemin de la conduction électrique cardiaque, ainsi que la position des différentes ondes sur le graphe [4].

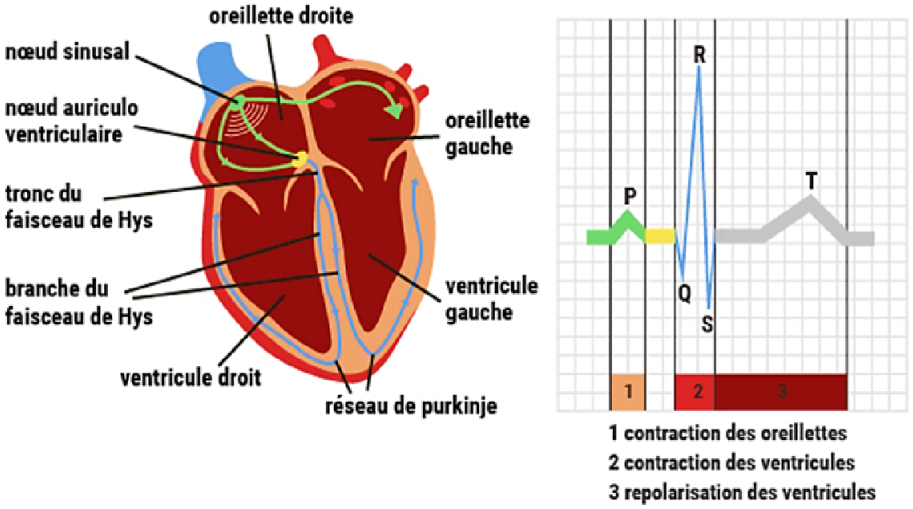


FIGURE 1.2: Chemin de la conduction électrique cardiaque, et la position des différentes ondes sur le graphe.

1.3 L'électrocardiographie de surface

1.3.1 Principe technique

L'électrocardiographe doit être capable de détecter non seulement des signaux extrêmement faibles allant de 0.5 mV à 5.0 mV, mais aussi une composante continue jusqu'à plus ou moins 300 mV (résultant du contact entre l'électrode et la peau) et une composante de mode commun jusqu'à 1.5 V, résultant du potentiel entre les électrodes et la terre.
La largeur de bande utile d'un signal ECG dépend de l'application et peut varier entre 0.5 Hz et 100 Hz, parfois jusqu'à 1 kHz. Il existe des sources de bruit affectant le signal ECG, citons les mouvements affectant l'interface peau-électrodes, les contractions musculaires ou les pics d'électromyographie, la respiration, les interférences électromagnétiques (EMI) et le bruit d'autres appareils électroniques connectés à l'entrée.
Tout d'abord, un amplificateur de bio potentialité est mis en place pour traiter l'ECG. Ensuite, des électrodes sont placées sur le patient pour mesurer la différence de potentiel entre deux bras. La fonction principale d'un amplificateur de bio potentialité est de prendre un faible signal électrique d'origine biologique et d'augmenter son amplitude afin de pouvoir le traiter, l'enregistrer ou l'afficher ultérieurement.

1.3.2 Les dérivations électrocardiographiques

En électrocardiographie, la dérivation est définie par deux points d'observation de l'activité électrique du cœur à partir desquels une différence de potentiel électrique est mesurée. Habituellement, les appareils d'électrocardiographie peuvent enregistrer simultanément différentes différences de potentiel, en fonction de l'emplacement et du nombre d'électrodes réparties sur le corps. Chaque mesure de ces potentiels correspond alors à une dérivation de l'ECG.
L'emplacement de ces électrodes a été choisi pour explorer presque tout le champ électrique du cœur résultant de la contraction du myocarde.

Les dérivations périphériques

Les dérivations périphériques permettent d'étudier l'activité électrique du cœur sur le plan frontal. Elles sont obtenues au moyen de 4 électrodes appliquées sur le bras droit, le bras gauche, la jambe gauche et l'électrode de la jambe droite comme étant une électrode neutre destinée à éliminer les parasites électriques. On distingue deux types :

1. Les dérivations périphériques bipolaires :
   Les dérivations bipolaires ont été déterminées par Einthoven [5]. Elles utilisent trois électrodes placées sur le sujet, sur les bras droit et gauche et sur la jambe gauche pour former un triangle (triangle d'Einthoven). Ces dérivations sont dites bipolaires parce qu'elles mesurent une différence de potentiel entre deux électrodes. Chaque côté du triangle formé par les trois électrodes représente une dérivation en utilisant une paire d'électrodes différente pour chacune des dérivations.
   Les trois dérivations sont :
   - DI : entre le bras droit et le bras gauche
   - DII : entre le bras droit et la jambe gauche
   - DIII : entre le bras gauche et la jambe gauche
2. Les dérivations périphériques unipolaires :
   Les dérivations unipolaires ont été introduites par Wilson [6,7]. Dans son système, les dérivations sont obtenues entre une électrode exploratrice située au sommet du triangle d'Einthoven et une borne centrale (électrode neutre ou indifférente,

telle que le potentiel est la moyenne des potentiels des trois sommets du triangle d'Einthoven). Cela a donné les dérivations unipolaires VL VR et VF. Plus tard, Goldberg [6,7] a modifié le système de Wilson pour obtenir trois dérivations unipolaires augmentées aVL, aVR et aVF, ces nouvelles dérivations amplifient les variations de potentiel des dérivations par un facteur de 1,5 par rapport à celle de Wilson.
La figure (1.3) montre les dérivations unipolaires des membres et bipolaires [8].

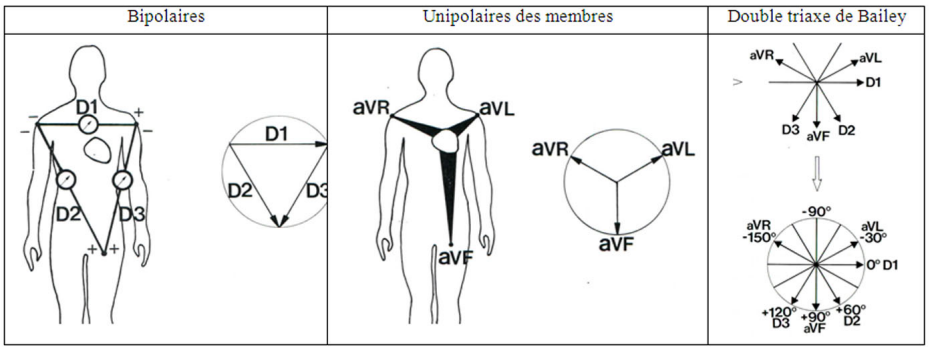


FIGURE 1.3: Les dérivations des membres.

Les dérivations précordiales

Wilson a introduit les dérivés dans le plan horizontal de V1, V2, V3, V4, V5 et V6 pour mesurer les potentiels proches du cœur [6,7]. Ces six dérivations sont situées sur le côté gauche du thorax figure(1.4) [9]. Les potentiels sont enregistrés à partir d'une électrode exploratrice (pôle positif) placée sur le thorax et l'électrode de référence (pôle négatif) connectée à la borne centrale de Wilson.
Ces dérivations sont positionnées comme suit :

- V1 : 4ème espace intercostal, bord droit du sternum (ligne parasternale)
- V2 : 4ème espace intercostal, bord gauche du sternum (ligne parasternale)
- V3 : à mi-distance entre V2 et V4
- V4 : 5ème espace intercostal, ligne médio-claviculaire gauche
- V5 : à mi-distance entre V4 et V6, sur la ligne axillaire antérieure
- V6 : même niveau horizontal que V4 et V5, ligne axillaire moyenne.

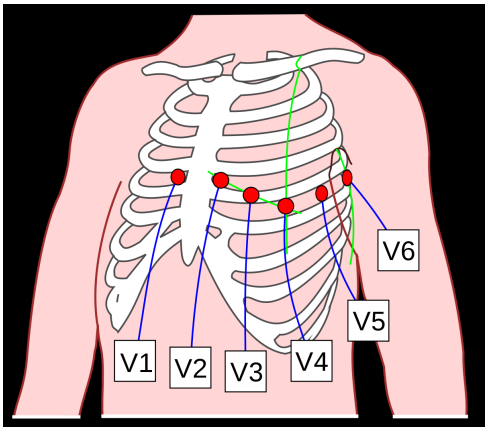


FIGURE 1.4: Les dérivations précordiales.

1.3.3 Description du tracé de l'électrocardiogramme (cycle cardiaque)

Le tracé de l'électrocardiogramme correspond à un cycle cardiaque et se déroule comme suit :

- Les impulsions électriques prennent naissance dans le nœud sinusal de l'oreillette gauche et se propagent ensuite dans l'oreillette droite, où une onde P positive est enregistrée dans la ligne gauche V4.
- L'impulsion électrique atteint alors le nœud atrioventriculaire (AV) puis le faisceau de HIS. À ce niveau, aucune partie du cœur n'est dépolarisée dans cet espace, que l'on appelle l'espace isoélectrique (correspondant à 0 mv) ou l'espace PQ ou PR.
- Au sommet du septum interventriculaire, le faisceau de HIS [10] se divise en deux, où la dépolarisation des ventricules commence, avec une onde négative appelée onde Q dans la dérivation gauche et une onde R positive dans la dérivation droite.
- La dépolarisation des deux ventricules commence alors, marquée par l'enregistrement d'une grande onde positive R sur les dérivations gauche et une grande onde négative S en dérivations droites.
- Enfin, la dépolarisation de l'infundibulum pulmonaire permet d'enregistrer une petite onde négative, l'onde S en dérivation gauche et une petite onde positive en dérivation droite qui est invisible sur l'ECG. Cette phase correspond à la dépolarisation des oreillettes et des ventricules.

Ainsi, le tracé de l'électrocardiogramme est composé des ondes P, QRS, T et U, ainsi que des intervalles qui sont importants pour l'interprétation de l'ECG, à savoir :

- Intervalle RR : Correspond à la durée entre deux contractions ventriculaires et permet le calcul de la fréquence cardiaque.
- Intervalle PP : Correspond à la durée entre deux dépolarisations des oreillettes.
- Intervalle PR : Correspond au délai entre la fin de la dépolarisation auriculaire et le début de la dépolarisation ventriculaire.
- Intervalle QT : Se réfère au temps entre le début de la dépolarisation des ventricules et la fin de leur repolarisation.
- Segment ST : Correspond à la durée pendant laquelle tous les tissus des ventricules sont dépolarisés.

1.3.4 Lecture et interprétation de l'ECG

Que faut-il observer sur l'électrocardiogramme ?

- Le Rythme : Regarder si l'intervalle R-R est quasi-constant sur tout le tracé, avec des complexes QRS similaires, alors le rythme est régulier.
- La Fréquence cardiaque, obtenue en hertz, et correspond à l'inverse de l'intervalle R-R en secondes, ou en battements par minute, on multipluant le resultat par 60.
- Les Ondes : Devant chaque trace, il faut regarder la présence et l'aspect de toutes les ondes :
  - L'onde P correspond à la contraction auriculaire, elle est généralement positive.
  - L'intervalle PR correspond à la conduction auriculo-ventriculaire (le passage des impulsions électriques des oreillettes aux ventricules).
  - Le QRS correspond à la contraction ventriculaire, il est généralement fin.
  - Le segment ST est toujours isoélectrique.
  - L'onde T correspond à la repolarisation ventriculaire, elle est souvent positive et asymétrique.

1.4 Les arythmies cardiaques

L'arythmie représente l'absence de régularité des battements cardiaques, elle est souvent associe à un trouble de la conduction électrique. On en recense plusieurs types détaillés ci-dessous :

1.4.1 Contraction auriculaire prématurée (PAC)

Une contraction auriculaire prématurée se produit lorsqu'un foyer dans l'oreillette (et non le nœud sinusal) génère un potentiel d'action avant le prochain potentiel d'action du nœud sinusal prévu. Elle présentent des caractéristiques telle que :
Prématurée, se produisant plus tôt que prévu si on la mesure par rapport aux intervalles P-P précédents. Ectopique, provenant de l'extérieur du nœud SA, et donc, la morphologie de l'onde P serait différente de la morphologie normale de l'onde P sinusale.
Dans la figure (1.5), les positions des pics PAC sont indiquées par des flèches, ainsi que la morphologie normale du pic P et l'interval R-R normale [11].

FIGURE 1.5: Contraction auriculaire prématurée.

1.4.2 Extrasystole ventriculaire

Lorsque le foyer ectopique se manifeste au niveau du ventricule, l'arythmie est dite extrasystoles ventriculaires (ESV). Sur l'ECG ceci se traduit par un QRS large isolé sans onde P, habituellement suivi d'une pause compensatrice, comme illustré dans la figure (1.6) [12].

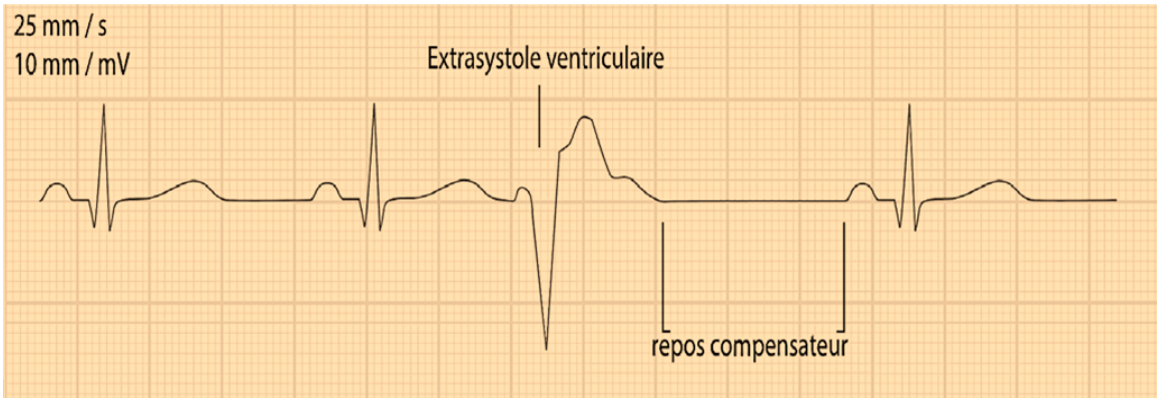


FIGURE 1.6: Extrasystole ventriculaire.

1.4.3 Le flutter ventriculaire

Le flutter ventriculaire est une forme de tachycardie ventriculaire très rapide associé à une conduction intraventriculaire anormalement élevée. Il est caractérisé par l'absence

de l'onde P, une morphologie du complexes QRS élargis (> 0,10 seconde) de forme sinusoïdale, cela est illustré dans la figure (1.7) [13].

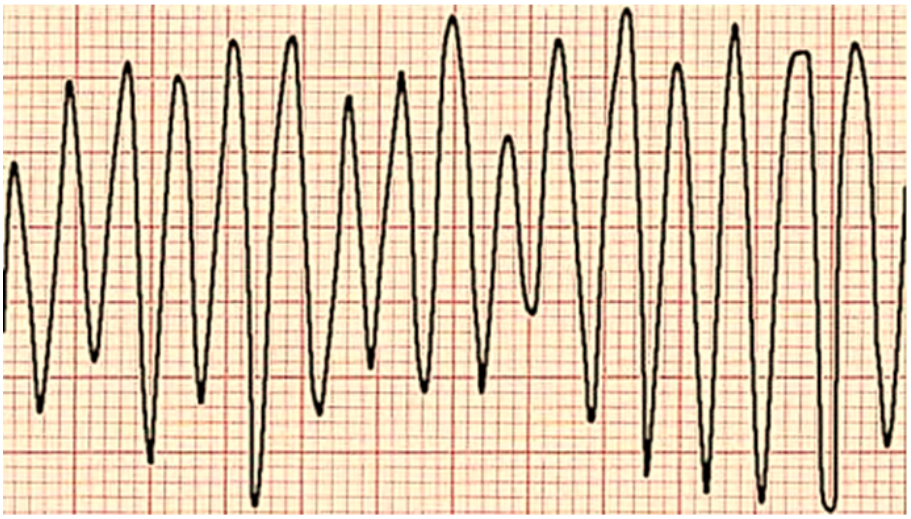


FIGURE 1.7: Le flutter ventriculaire.

1.4.4 Battement ventriculaire d'échappement

Un battement d'échappement ventriculaire est une décharge électrique auto-générée initiée par les ventricules du cœur et provoquant leur contraction. Le rythme d'échappement ventriculaire est suivi d'une longue pause dans le rythme ventriculaire, ce qui indique un échec du système de conduction électrique du cœur à stimuler les ventricules. Cela est traduit sur le tracé de l'ECG, par une forme plus large du complexe QRS. Dans la figure (1.8) les 2.5 premières secondes montrent un cycle cardiaque normal. Il est suivi d'une période d'activité sinusale retardée qui déclenche une réponse de prise en charge par les cellules du stimulateur ventriculaire, entraînant un battement ventriculaire d'échappement. Deux battements d'échappement sont affichés entre 5 et 8 secondes dans la figure (1.8) [14].

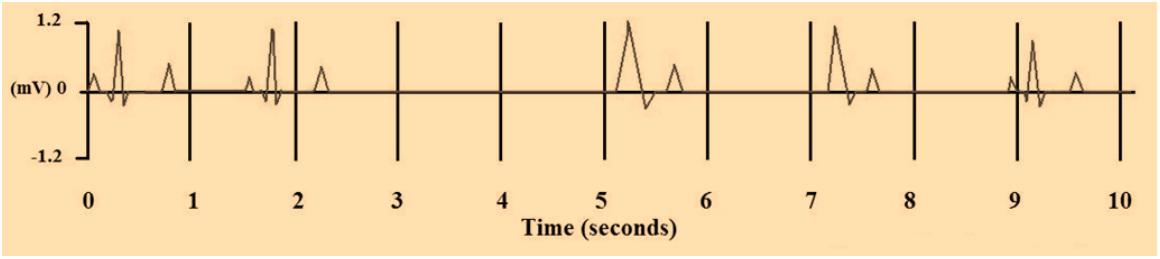


FIGURE 1.8: Battement ventriculaire d'échappement.

1.4.5 Bloc de branche

L'obstruction des branches peut se manifester par un ralentissement de la conduction électrique dans les branches ou par un blocage de celles-ci. Si l'influx ne passe pas à travers les branches, une dépolarisation se produira progressivement entre les cellules même du myocarde, entraînant une dépolarisation des ventricules sur une plus longue période, engendrant ainsi un QRS plus large sur le tracé de l'ECG.
On distingue les 2 types suivants :

Bloc de branche droit

Le trouble de la conductivité affecte la branche droite, dans ce cas le ventricule droit est dépolarisé après le ventricule gauche, ceci est reflété dans le tracé de l'ECG sur la figure (1.9) par un grand complexe rSR' large en dérivation V1, un complexe QRS en dérivation v6 et une onde T inversée sur les dérivations précordiales v1, v2 et v3 [15].

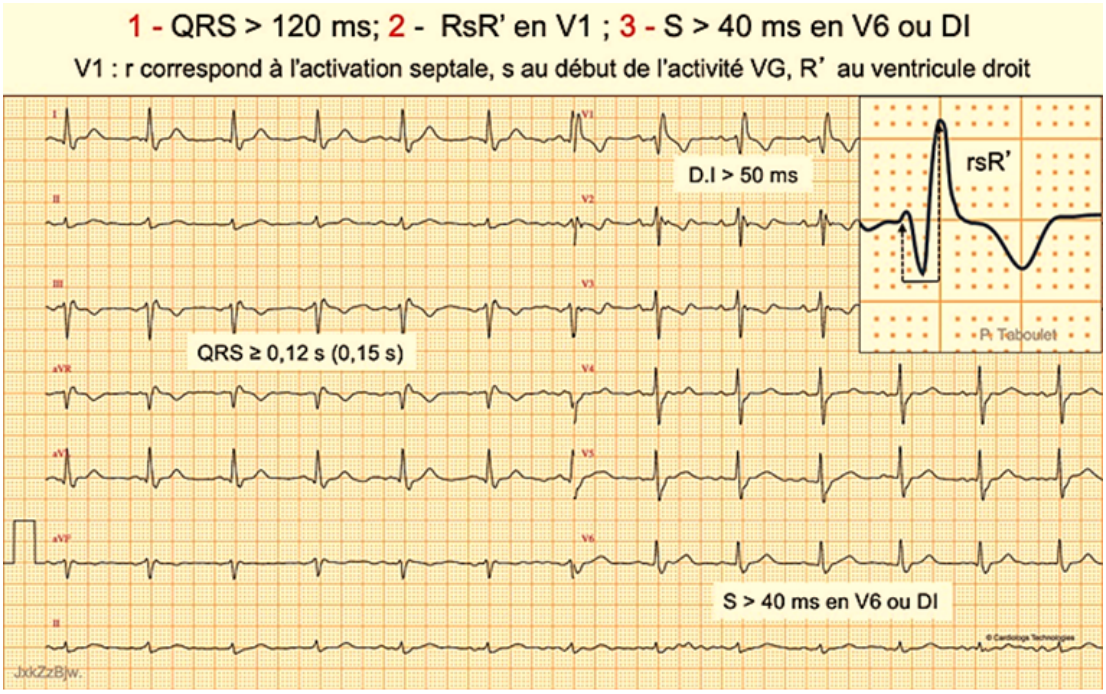


FIGURE 1.9: Bloc de branche droit.

Bloc de branche gauche

C'est la branche gauche qui est affectée par le problème de conduction, le ventricule gauche est dépolarisé après le ventricule droit, cela se traduit dans la trace de l'ECG sur la figure (1.10) par une durée des QRS supérieure ou égale à 120 ms, un retard en V5-V6 et DI-VL avec une onde R large, disparition de l'onde q septale en DI, V5-V6 [16].

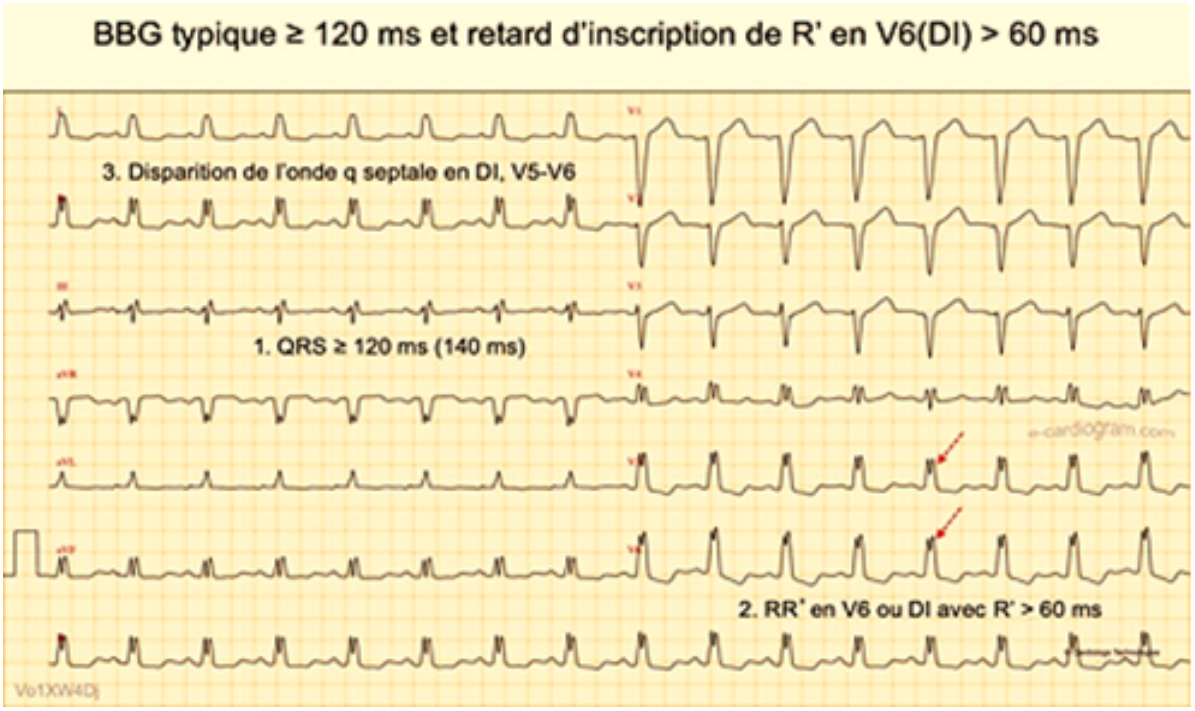


FIGURE 1.10: Bloc de branche gauche.

1.5 L'enregistrement du Holter

Est un type d'appareil d'électrocardiographie ambulatoire,il est portable, et utilisé pour surveiller l'activité cardiaque ECG pendant au moins 24 à 48 heures. Pendant ce temps, l'appareil enregistre tous vos battements cardiaques. Sa période d'enregistrement prolongée est parfois utile pour observer des arythmies cardiaques occasionnelles qui seraient difficiles à identifier dans une période plus courte. Il est d'une grande utilité pour les patients présentant des symptômes plus transitoires.

1.6 Conclusion

A travers ce chapitre nous avons présenté une description des principes du fonctionnement cardiaques, où on a introduit l'activité mécanique du cœur engendrée par son activité électrique et enregistrée sur l'ECG. La lecture ainsi que l'interprétation de l'ECG permettent la détection des anomalies cardiaques telles que les troubles du rythme dits arythmies. Cela permet aux spécialistes de faire un diagnostic des pathologies possibles. Cependant, en raison de la variabilité morphologique du signal ECG chez un même individu ou chez des individus différents, cette détection est une tâche difficile à traiter, tant manuellement qu'automatiquement.

Chapitre 2

Apprentissage profond

2.1 Introduction

L'intelligence artificielle est un domaine qui s'intéresse à produire des systèmes artificiels aussi intelligents que l'homme, des systèmes avec des capacités cognitives et qui savent prendre la décision sans intervention humaine. Cela peut recouvrir plusieurs domaines comme la classification des images en plusieurs catégories, traduire un texte d'une langue à une autre, la détection des objets et son application pour la vidéo-surveillance, la navigation d'un robot dans son environnement de façon autonome, créer des systèmes basés sur ordinateur qui aident les médecins pour le diagnostic des patients, etc.
L'apprentissage automatique (machine learning en anglais) est un sous-domaine de l'IA, il se repose sur le principe de reproduire des systèmes qui sont capables d'analyser des données, à apprendre de ces données et à appliquer ensuite ce qu'ils ont appris pour prendre une décision.
Un réseau de neurones est une technique d'apprentissage automatique inspirée des réseaux de neurones biologiques, c'est donc un sous-domaine de l'apprentissage automatique (et aussi de l'IA). Les réseaux de neurones artificiels sont composés de neurones artificiels simples qui sont des petites fonctions mathématiques montés en réseau et qui nous permettent de former des fonctions complexes.
Un réseau de neurones profond est un réseau de neurones multicouches (en général plus de 4 couches) qui peut comporter des millions de neurones, répartis en plusieurs dizaines de couches. Les réseaux de neurones profonds peuvent apprendre des caractéristiques à différents niveaux d'abstraction, des contours (aux couches inférieures) aux caractéristiques très complexes (aux couches plus profondes).

2.2 Les applications du deep learning

2.2.1 La reconnaissance faciale

En se besant sur un ensemble d'images donner à l'algorithme de deep learning, il sera en mesure de détecter des caractéristiques comme Les yeux, le nez, la bouche et autant d'autres qui l'aide a détecter le visage [17].

2.2.2 La détection d'objets

Sur des images, on peut trouver plusieurs éléments. Les algorithmes de détection d'objets sont faits pour détecter ces éléments-là et les localiser. Ces algorithmes sont appliqués dans divers domaines ; par exemple pour identifier une personne sur une photo dès lors qu'elle est chargée sur les réseaux sociaux [18].

2.2.3 Traitement du langage naturel(NLP)

Le traitement du langage naturel (Natural Language Processing en anglais) est une technique qui aide les ordinateurs à comprendre le langage naturel de l'homme. Le but du traitement du langage naturel est de lire, comprendre et extraire le sens des mots du langage humain. On trouve le NLP dans plusieurs domaines, comme par exemple la traduction d'un texte d'une langue à une autre, ou permettre à des systèmes de communiquer avec des personnes via des chatbots [19].

2.2.4 Application médicale

L'apprentissage profond apporte plusieurs solutions pour le domaine médical, il permettre au médecin de gagner du temps et d'optimiser le diagnostic des patients. Un

exemple d'application est la détection automatique de tumeurs sur des radiographies médicales. Une autre application est la classification de ces tumeurs (bénignes ou malignes) [20].

2.2.5 Les voitures autonomes

Les voitures autonomes sont des véhicules aptes à rouler sur route sans intervention d'un conducteur. Pour construire des telles voitures, il faut apprendre à un ordinateur à maîtriser certaines parties essentielles de la conduite à l'aide d'un système de capteurs au lieu de la cognitive humaine. Pour ce faire, il faut entraîner des algorithmes d'apprentissage profond sur un grand ensemble d'image et autres informations provenant des capteurs pour rendre ces ordinateurs aptes de détecter les autres voitures ou les objets qui les entourent et ce afin de respecter le trajet à suivre [21].

2.3 Neurone biologique

Un neurone (figure 2.1) est une cellule du système nerveux qui assure le traitement et la transmission d'informations. Chaque neurone est composé de :

- Péricaryon qui est un corps cellulaire, il comporte le noyau.
- De nombreuses ramifications de type dendritiques. Ces ramifications représentent la source de l'information.
- Un axone (par lequel sont diffusées les informations) dont la longueur peut atteindre 1 mètre pour seulement 1 à 15 micromètres de diamètre. Il est entouré par des cellules de Schwann (séparées par les nœuds de Ranvier) qui confèrent une gaine de myéline protectrice tout le long de l'axone.

Les informations électriques (provenant du système nerveux) arrivent aux dendrites qui sont les entrées du neurone. Un signal électrique peut être émis le long de l'axone si les signaux électriques arrivant par les dendrites excitent assez le neurone (dépasser un seuil d'excitation) [22].

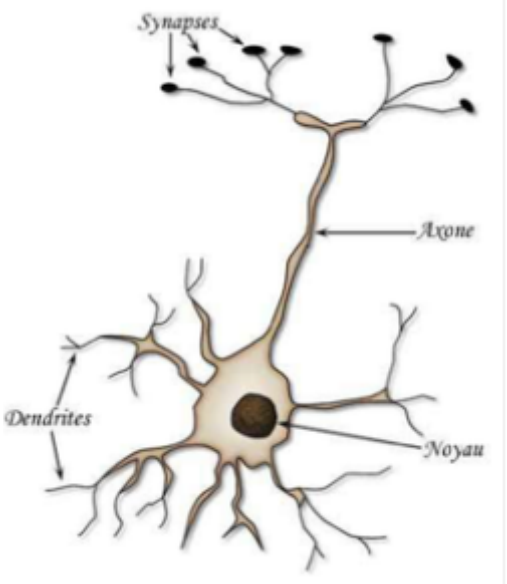


FIGURE 2.1: Neurone biologique.

La synapse est une zone de contact fonctionnelle qui assure le lien entre la sortie d'un neurone (l'axone) et l'entrée d'un autre neurone (dendrite). Si l'amplitude du signal arrivant à la synapse est assez élevée, alors elle laissera passer le signal, sinon il sera inhibé. Le neurone biologique est illustré dans la figure (2.1)

2.4 Le réseau neuronal

Les réseaux de neurones (neural network en anglais) sont des algorithmes de traitement de l'information inspirés du fonctionnement du système nerveux biologique. Ces réseaux s'inspirent de la façon dont le cerveau traite et manipule l'information au niveau du fonctionnement. Tous les réseaux de neurones sont constitués de neurones interconnectés qui sont organisés en couches.

2.4.1 Neurone formel (Perceptron)

La conception des réseaux de neurones se base sur les neurones artificiels qui sont inspirés du vrai neurone existant dans notre cerveau. La figure (2.2) montre une représentation d'un neurone artificiel [22].

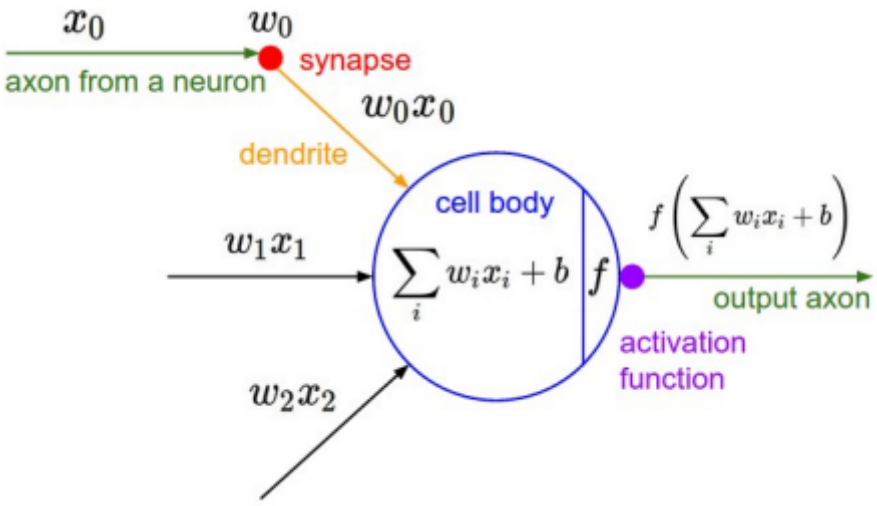


FIGURE 2.2: Neurone artificiel.

- Les x_i représentent soit les données d'entrée, soit les valeurs de sorties d'autres neurones.
- Les poids w_i représentent les valeurs de puissance des entrées.
- Le neurone artificiel fera un produit entre les poids w_i et les valeurs d'entrée x_i, puis ajouter un biais b.
- Le résultat est transmis à une fonction d'activation f qui ajoutera une certaine non-linéarité.

2.4.2 Les fonctions d'activation

Après la sommation pondérée de ses entrées, le neurone applique également une non-linéarité sur ce résultat. Cette fonction non-linéaire s'appelle la fonction d'activation. La fonction d'activation est essentielle pour le fonctionnement des réseaux de neurones, c'est elle qui fait que si le neurone est activé ou non. La sortie de cette fonction est envoyée à la couche suivante. Sans la fonction d'activation, un réseau de neurones n'est devenu qu'un simple modèle linéaire. Il existe de nombreux types de ces fonctions, parmi lesquelles nous trouvons :

La fonction Sigmoïde

La fonction sigmoïde ou la fonction logistique est l'une des fonctions les plus utilisées en apprentissage profond notamment dans la couche de sortie pour un problème de

classification binaire. Elle est bornée entre 0 et 1 et elle peut être interprétée comme la probabilité que le neurone s'active [23]. La sortie de la fonction sigmoïde est donnée par l'équation et la figure suivante :

$$y = \frac{1}{1 + \exp^{-z}} \tag{2.1}$$

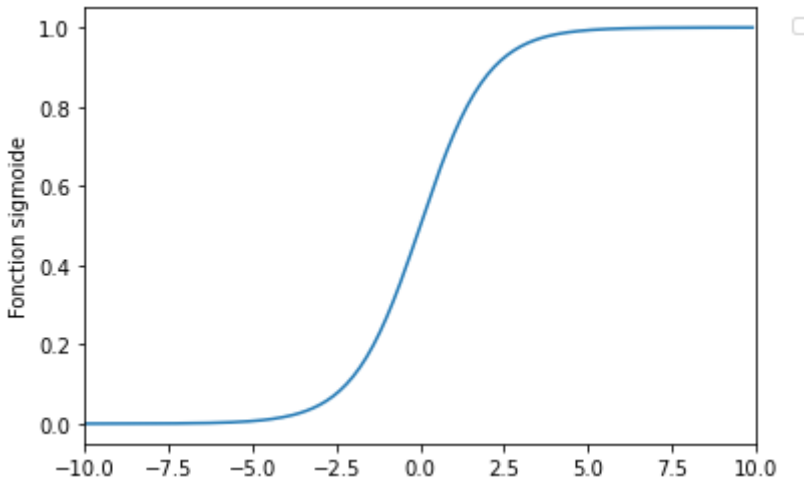


FIGURE 2.3: Représentation graphique de la fonction Sigmoïde.

La fonction ReLu

La fonction RELU (en anglais rectified linear unit) est récemment devenue un choix incontournable pour les réseaux de neurones [23]. Cette fonction nous renvoie 0 si l'entrée est inférieure à 0 et retourne l'entrée elle-même si elle est plus grande que 0. La sortie de la fonction relu est donnée par l'équation et la figure suivante :

$$y = \max(0, z) \tag{2.2}$$

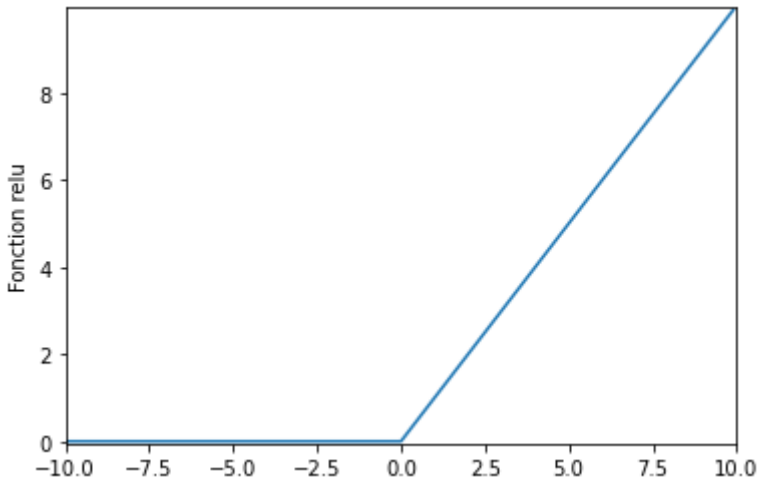


FIGURE 2.4: Représentation graphique de la fonction Relu.

La fonction Softmax

La fonction softmax ou régression Softmax est une généralisation de la fonction sigmoïde que nous pouvons l'utiliser en couche de sortie dans le cas d'une classification multi-classes. La sortie de la fonction softmax est donnée par l'équation suivante :

$$y_i = \frac{\exp^{-z_i}}{\sum_{j=1}^{c} \exp^{-z_j}} \tag{2.3}$$

c : représente le nombre de classes de la sortie.

- La sortie d'un neurone utilisant la fonction softmax dépend des sorties de tous les autres neurones de sa couche.
- La somme de toutes les sorties est égale à 1

La fonction Tanh (Hyperbolic Tangent Function)

La fonction tangente hyperbolique connue sous le nom de fonction tanh, est une fonction centrée sur le zéro, plus douce que la fonction sigmoïde et dont la plage se situe entre -1 et 1 [23]. La sortie de la fonction tanh est donnée par l'équation et la figure suivante :

$$y_i = \frac{\exp^{z} - \exp^{-z}}{\exp^{z} + \exp^{-z}} \tag{2.4}$$

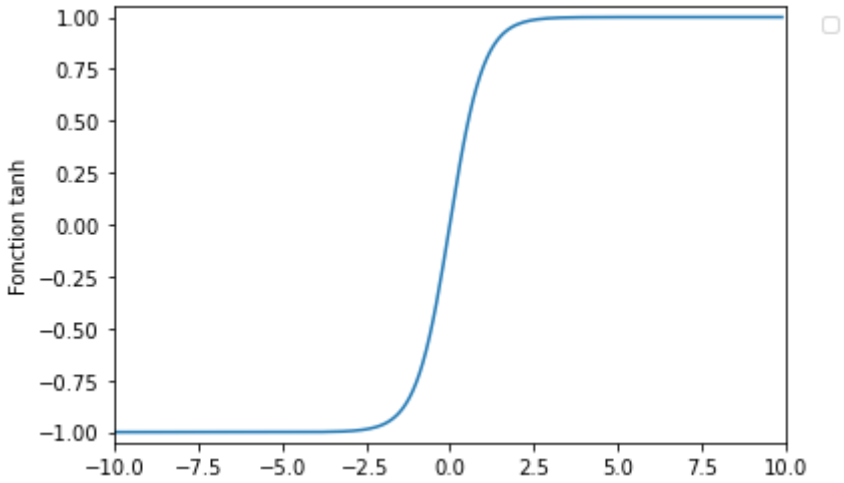


FIGURE 2.5: Représentation graphique de la fonction tanh.

La différence entre ces fonctions d'activation

- La fonction sigmoïde n'est pas centrée au zéro et elle est toujours positive. Le gradient sera toujours positif ou négatif et sa convergence va prendre beaucoup de temps.
- Les fonctions sigmoïde et tanh sont des fonctions saturées, quand on a des grandes valeurs à leurs entrées la sortie sera proche de 0 ou de 1 dans le cas de la sigmoïde et -1 et 1 dans le cas de tanh. Des grandes variations en entrée de ces fonctions seront équivalentes à des petites variations en sortie, comme la dérivée représente la vitesse de variation de la sortie, donc elle sera petite et le gradient tend vers le zéro, les paramètres changent très lentement et l'apprentissage est donc très lent. (problème de vanishing) [23].

- La fonction relu est la meilleure solution pour éviter le problème du vanishing et zéro centrée, c'est pour cette raison qu'elle est la fonction d'activation la plus utilisée [23].

2.4.3 La fonction de perte (cost function)

La fonction de perte (cost function en anglais) est une mesure de l'erreur du modèle en termes de sa capacité à estimer la relation entre les sorties prédites et les vraies sorties. Elle est utilisée afin que les paramètres puissent être mis à jour pour réduire la perte lors de la prochaine évaluation. Le choix de la fonction de perte est directement lié à la fonction d'activation utilisée dans la couche de sortie de votre réseau neuronal et le problème à prédire [24].

Problème de régression linéaire :
Prédiction d'une quantité à valeur réelle.

- La couche de sortie : un nœud avec une unité d'activation linéaire [24].
- La fonction de perte utilisée : l'erreur quadratique moyenne (MSE), elle est donnée par l'équation suivante :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 \tag{2.5}$$

y_{i} : C'est la vraie sortie.
$y_{\text{i}}^{'}$: C'est la sortie prédite.
n : C'est la taille de la sortie.
Classification binaire :
Un problème où vous classifiez un exemple comme appartenant à une des deux classes [24].

- La couche de sortie : Un nœud avec une unité d'activation sigmoïde.
- La fonction de perte utilisée : l'entropie croisée, elle est également appelée perte logarithmique. Cette fonction de perte est donnée par l'équation suivante :

$$CE = -\sum_{i=1}^{2} y_i log(y_i') = -(y_1 log(y_1') + (1 - y_1) log(1 - y_1')) \tag{2.6}$$

Classification multi-classes :
Un problème où vous classifiez un exemple comme appartenant à une des plusieurs classes [24].

- La couche de sortie : Un nœud pour chaque classe utilisant la fonction d'activation softmax.
- La fonction de perte utilisée : l'entropie croisée, elle est donnée par l'équation suivante :

$$CE = -\sum_{i=1}^{c} y_i log(y_i') \tag{2.7}$$

c : représente nombre de classes.

2.5 Les types d'apprentissage des réseaux de neurones

L'apprentissage est une phase du développement du réseau de neurones dans laquelle les poids des neurones sont mis à jour afin que les sorites du réseau soient aussi proches

que possible des sorties souhaitées. Il existe deux types d'apprentissages : apprentissage supervisé et apprentissage non supervisé, dans notre cas d'étude, on s'intéressera au premier type d'apprentissage [25].

2.5.1 L'apprentissage supervisé

L'apprentissage supervisé est une technique d'apprentissage automatique dans laquelle les données fournies à nos algorithmes de traitement des données sont étiquetées (en anglais on dit labeled data), c'est-à-dire que les classes correspondantes à ces données-là sont connues. Les algorithmes doivent s'en servir pour prédire un résultat en vue de pouvoir le faire plus tard lorsque les données ne seront plus étiquetées [25].

2.5.2 L'apprentissage non supervisé

L'apprentissage non supervisé est une technique d'apprentissage où on a pas besoin d'utiliser des données étiquetées. On doit plutôt laisser le modèle fonctionner tout seul pour découvrir les caractéristiques et les relations qui lient ces données. C'est une tâche de traitement plus complexe par rapport à l'apprentissage supervisé. Cependant, l'apprentissage non supervisé peut être plus imprévisible que les autres méthodes d'apprentissages [25].

2.6 Perceptron multicouches

2.6.1 Introduction

Perceptron multicouches (en anglais multilayer perceptron « MLP ») est un type du réseau de neurones artificiel organisé en plusieurs couches au sein desquelles une information se propage de la couche d'entrée vers la couche de sortie. Chaque couche est constituée d'un nombre variable de neurones, les neurones de la dernière couche étant les sorties du réseau [26]. Un perceptron à deux couches est donné par la figure suivante :

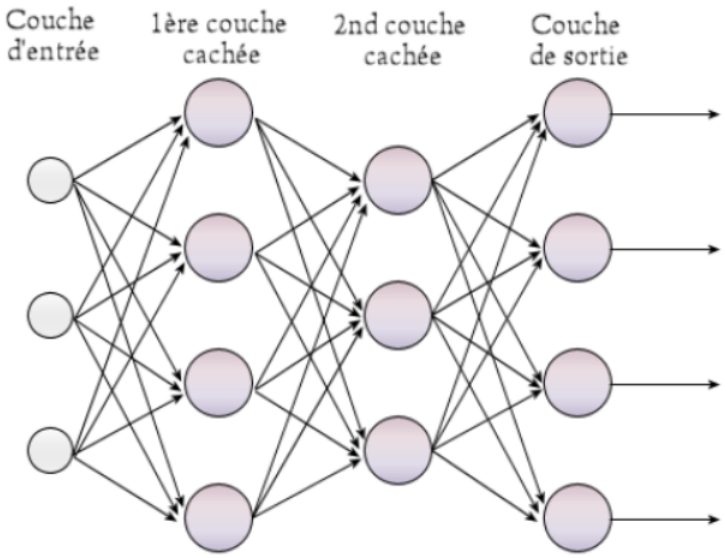


FIGURE 2.6: Représentation graphique d'un perceptron multicouche.

2.6.2 Fonctionnement des perceptrons multicouches.

Propagation vers l'avant(Forward propagation)

Pour les perceptrons multicouches l'information se propage de la couche d'entrée jusqu'a la couche de sortie en passant par les différentes couches intermediaires [27]. Pour bien comprendre comment ça fonctionne on prend l'exemple de la figure (2.7) :

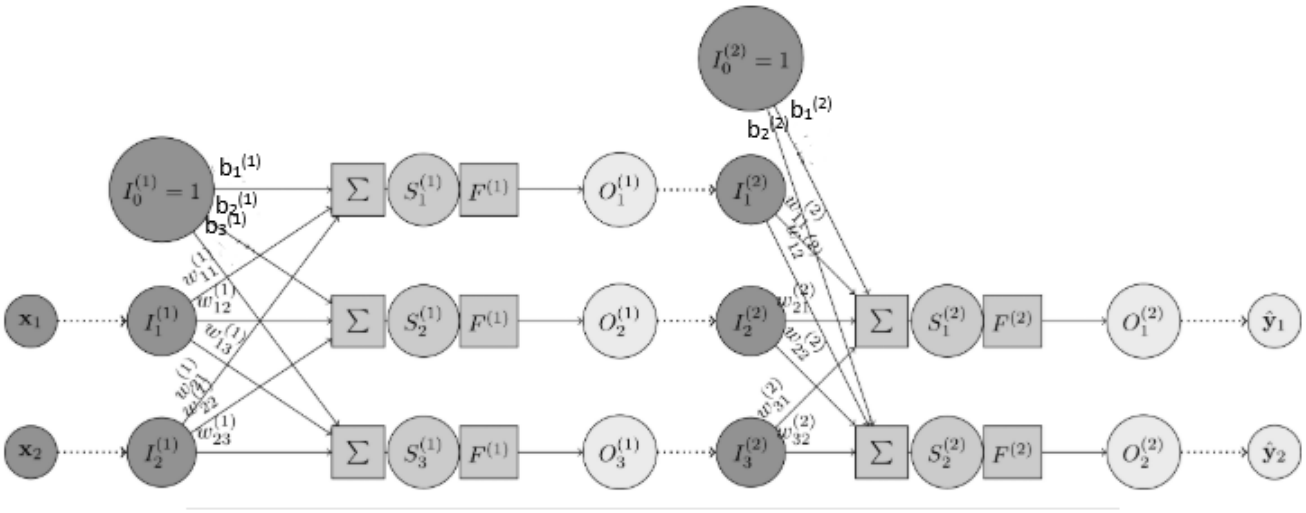


FIGURE 2.7: Un réseau de neurones à deux couches.

- Les I_i^l représentent les entrées des neurones et (l) représente le numéro de la couche sur laquelle on se trouve.
- Les S_j^l représentent les sorties lineaires et les O_i^l représentent les sorties non lineaires des neurones.
- y_i représente la sortie du réseau, les b_i^l représentent les biais.
- F^l représente la fonction d'activation, pour la dernière couche on utilise une fonction sigmoïde si la classification est binaire ou la fonction softmax dans le cas d'une classification multiclasse et pour les autres couches on utilise souvent la foncion relu.
- Les sorties des couches seront des entrées pour les couches suivantes et pour la premiere couche l'entrée est égale au vecteur caracteristique
- Les sorties lineaire et non lineaire des neurones sont obtenues par les relations :

$$S_j^l = \sum_{i=1}^{i=n_i} W_{ji}^l I_i^l + b_j^l \tag{2.8}$$

$$O_j^l = F^l(S_j^l) \longrightarrow I^{l+1} \tag{2.9}$$

j : indique le neurone de la couche sur laquelle on se trouve.
i : indique la iéme entrée du neurone de la couche actuelle.
n_i : représente le nombre des entrées.

Optimisation du perceptron multicouche par l'algorithme "Rétropropagation du gradient"(minimisation des erreurs)

Pour l'apprentissage du perceptron, on va commencer par calculer le gradient des paramètres de la dernière couche, puis propager le gradient vers les entrées de la dernière

couche (ce qui correspond au gradient de la sortie de la couche précédente) [27]. Ainsi, on peut calculer le gradient des paramètres de l'avant-dernière couche, puis le gradient de son entrée. Et ainsi de suite, couche à couche, de la dernière à la première. C'est ce qu'on appelle la rétropropagation du gradient.

- Les gradients de la perte par rapport aux paramètres de la couche de sortie sont alors donnés par :

$$dS_j^L = \frac{\partial J}{\partial S_j^L} = \frac{\partial J}{\partial O_j^L}\frac{\partial O_j^L}{\partial S_j^L} \tag{2.10}$$

$$dW_{ji}^L = \frac{\partial J}{\partial W_{ji}^L} = \frac{\partial J}{\partial O_j^L}\frac{\partial O_j^L}{\partial S_j^L}\frac{\partial S_j^L}{\partial W_{ji}^L} = \frac{\partial J}{\partial O_j^L} f'^L(S_j^L) I_i^L \tag{2.11}$$

$$db_j^L = \frac{\partial J}{\partial b_j^L} = \frac{\partial J}{\partial O_j^L}\frac{\partial O_j^L}{\partial S_j^L}\frac{\partial S_j^L}{\partial b_j^L} = \frac{\partial J}{\partial O_j^L} f'^L(S_j^l) \tag{2.12}$$

J : représente la fonction de perte (cost function).
L : c'est l'indice de la dernière couche.

- Les gradients des autres couches se calculent avec le même principe des dérivées partielles :

$$dS_j^l = \frac{\partial J}{\partial S_j^l} = \frac{\partial J}{\partial O_j^L}\frac{\partial O_j^L}{\partial S_j^L}.....\frac{\partial S_j^{l+1}}{\partial O_j^l}\frac{\partial O_j^l}{\partial S_j^l} \tag{2.13}$$

$$dW_{ji}^l = \frac{\partial J}{\partial W_{ji}^l} = \frac{\partial J}{\partial O_j^L}\frac{\partial O_j^L}{\partial S_j^L}.....\frac{\partial S_j^{l+1}}{\partial O_j^l}\frac{\partial O_j^l}{\partial S_j^l}\frac{\partial S_j^l}{\partial W_{ji}^l} \tag{2.14}$$

$$db_j^l = \frac{\partial J}{\partial b_j^l} = \frac{\partial J}{\partial O_j^L}\frac{\partial O_j^L}{\partial S_j^L}.....\frac{\partial S_j^{l+1}}{\partial O_j^l}\frac{\partial O_j^l}{\partial S_j^l}\frac{\partial S_j^l}{\partial b_j^l} \tag{2.15}$$

l : c'est l'indice de la couche sur laquelle on se trouve.

- A la fin les poids et les biais seront mettre à jour avec les relations suivantes :

$$W_{ji}^l = W_{ji}^l - \alpha dW_{ji}^l \tag{2.16}$$

$$b^l = b^l - \alpha db^l \tag{2.17}$$

α : représente le pas d'apprentissage.
La figure (2.8) donne une illustration du fonctionnement de l'algorithme de gradient descent.

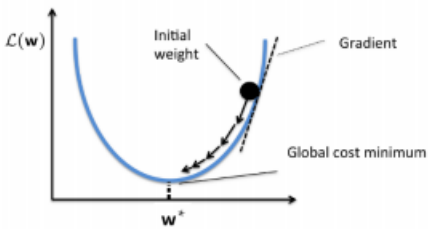


FIGURE 2.8: Descente du gradient.

2.7 Les réseaux convolutifs (CNN)

2.7.1 Introduction

Les réseaux de neurones convolutifs (CNN pour Convolutional Neural Networks) sont des réseaux de neurones multicouches qui sont trés utilisés dans le domaine du computer vision. Ils sont connus par leur robustesse aux faibles variations d'entrée, le faible taux de prétraitement nécessaires à leur fonctionnement, et l'extraction automatique des caractéristiques. Ils sont composés par des couches de convolutions, couches pooling et des couches entièrement connectées [28]. Ces réseaux de neurones profonds sont inspirés des travaux de Hubel et Wiesel sur le cortex visuel primaire du chat. La première partie des réseaux consistant en une succession de couches de convolution et pooling, elle est dédiée à l'extraction automatique des caractéristiques, tandis que la seconde partie, est composée de couches de neurones entièrement connectés, elle est dédiée à la classification. Un réseau CNN est donné par la figure suivante :

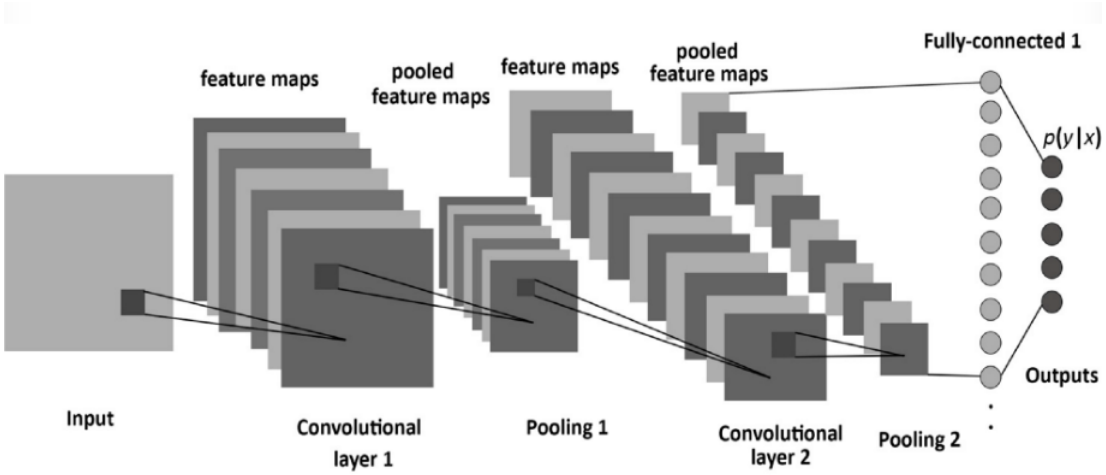


FIGURE 2.9: Représentation graphique d'un réseaux convolutifs (CNN).

Couches de convolution

Ces couches représentent le cœur du réseau convolutif, elles effectuent la partie la plus importante des calculs. Elles sont formées par un ensemble de filtres (on dit aussi kernel) qui assurent l'extraction des vecteurs caractéristiques qui caractérisent notre entrée par une opération de convolution avec l'entrée du réseau. Plus on a des couches de convolution plus on extrait des caractéristique plus complexes [28].

Couches de pooling

Le Pooling est une technique utilisée souvent dans les réseaux CNN, elle permet de prendre une large entrée (par exemple une image) et d'en réduire la taille tout en gardant les informations les plus importantes qu'elle contient. Pour se faire, il suffit de faire glisser une petite fenêtre pas à pas sur toutes les parties de l'image et de prendre la valeur maximum (max pooling) ou la moyenne (average pooling) de cette fenêtre à chaque pas. En pratique, on utilise souvent une fenêtre de 2 pixels de côté et une valeur de 2 pixels (stride=2) comme pas pour glisser la fenêtre [28].

Couches entièrement connectés

Les couches entièrement connectées sont des perceptrons multicouches. Elles reçoivent les vecteurs caractéristiques en sortie des couches de convolutions et de pooling pour les traiter et générer une décision finale et attribue à l'entrer la classe correspondante [28]

2.7.2 Fonctionnement d'un réseaux de neurones convolutif.

Propagation vers l'avant (Forward propagation)

Pour les réseaux de neurones convolutifs l'information se propage de la couche d'entrée (la première couche de convolution) jusqu'à la couche de sortie (la dernière couche entièrement connectée) en passant par les différentes couches intermédiaires.

1. **Propagation vers l'avant pour la couche de convolution**

   Les couches convolutives sont caractérisées par une entrée X, un ensemble de filtres h et des biais b. La sortie de ces couches de convolution se calcule par une opération de convolution entre l'entrée X et les filtres h, puis un ajout d'un biais b et application de non-linéarité [29].
   Pour bien comprendre comment ça fonctionne, on prend l'exemple donné par la figure suivante :

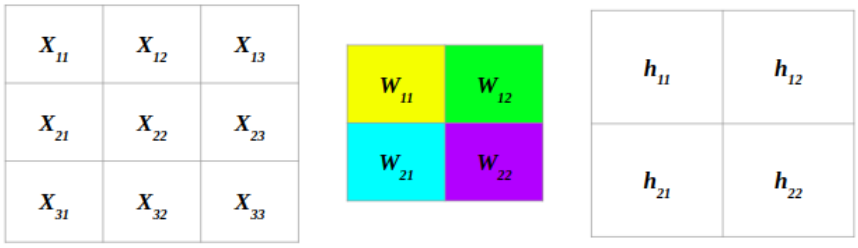


FIGURE 2.10: Illustration de l'opération de convolution.

- Nous avons une image de taille 3 x 3 x 1.
- Nous souhaitons lui appliquer un filtre dont la taille est un paramètre noté f : $f = 2$ signifie que le filtre est de taille 2 x 2.
- Pour effectuer la convolution, le filtre va parcourir l'entrée X de gauche à droite et de haut en bas avec un pas de un (stride=1). À chaque pas, le filtre est multiplié avec la fenêtre de l'image correspondante puis sommée [29].

- Au résultat de filtrage, on ajoute un biais b et ensuite on applique une fonction d'activation.

$$o_{11} = F(h_{11} + b1) = F(W_{11} * X_{11} + W_{12} * X_{12} + W_{21} * X_{21} + W_{22} * X_{22} + b1) \tag{2.18}$$

$$o_{12} = F(h_{12} + b2) = F(W_{11} * X_{12} + W_{12} * X_{13} + W_{21} * X_{22} + W_{22} * X_{23} + b2) \tag{2.19}$$

$$o_{21} = F(h_{21} + b3) = F(W_{11} * X_{21} + W_{12} * X_{22} + W_{21} * X_{31} + W_{22} * X_{32} + b3) \tag{2.20}$$

$$o_{22} = F(h_{22} + b4) = F(W_{11} * X_{22} + W_{12} * X_{23} + W_{21} * X_{32} + W_{22} * X_{33} + b4) \tag{2.21}$$

- Les éléments du filtre (valeurs) peuvent ne pas être connus en avance, car le Back-Propagation pourra permettre de les apprendre [30].
- La convolution réduit la taille de l'image initiale. Si l'image est de taille (n , n) et le filtre de taille (f , f) alors la sortie sera de taille ([(n-f+2p)/s+1] , [(n-f+2p)/s+1]](les crochets [] signifient la partie entière). [30].

Dans la pratique, les couches convolutives sont caractérisées par une entrée X de taille (m,W,H,D), un ensemble de filtres K de taille (f,f,D) et des biais b de taille(1,1,1,D_C).

- W : Largeur de l'entrée.
- H : Hauteur de l'entrée.
- D : Profondeur de l'entrée (dans le cas d'une image, D=3 qui est le nombre des canaux RGB, il peut aussi être le nombre des filtres de la couche précédente).
- m : Représente le nombre d'exemples dans notre base de données.
- fxf : C'est la taille des filtres de la couche actuelle.
- D_C : Nombre des filtres de la couche actuelle.
- La taille de la sortie est (m,W_C,H_C,D_C) avec :

$$W_C = [\frac{W - f + 2P}{S} + 1], \tag{2.22}$$

$$H_C = [\frac{H - f + 2P}{S} + 1] \tag{2.23}$$

- P : Représente le rembourrage par zéro "zero-padding".
- S : Le pas utilisé pour glisser les filtres sur l'image (stride en anglais).

La convolution entre l'entrée et les filtres sera une convolution volumique 3D (voir figure 2.11), c'est-à-dire une convolution sur un volume et non pas sur une surface 2D [30].

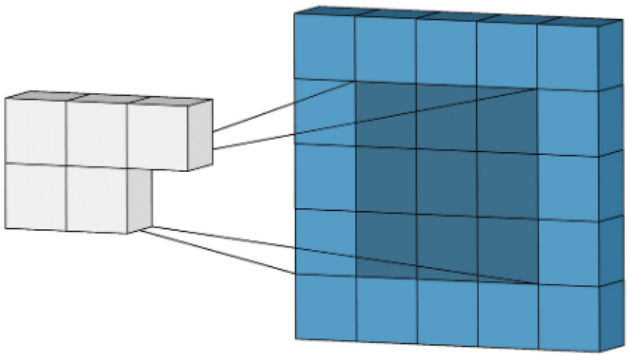


FIGURE 2.11: Convolution 2D.

rembourrage par zéro "zero-padding" (P) : Quand le filtre parcours l'entrée pour faire l'opération de convolution, les pixels sur les bords sont moins utilisés que ceux dans l'intérieur de l'entrée, ça cause une perte d'informations sur ces zones. Pour éviter ce problème, on utilise le padding qui consiste à ajouter une bordure d'un pixel (dans le cas P=1) à l'image avec une valeur de pixel de zéro. Pour contrôler l'épaisseur des bordures de padding, on ajuste le paramètre P de la couche correspondante [30].

le pas ou stride (S) : C'est le pas utilisé pour glisser le filtre sur une entrée. Il

indique de combien de pixels un filtre effectue un saut pour balayer l'image. Un stride S=1 signifie que le filtre va parcourir l'image avec un pas de 1 pixel [30].

2. **Propagation vers l'avant pour la couche de pooling**

Le principe de fonctionnement de la couche pooling est de faire balayé une fenêtre sur notre entrée (de la même manière qu'une couche convolutive), à chaque pas, on calcule la valeur maximale (pour un max pooling) ou la moyenne (pour un average pooling) dans la zone sélectionnée par la fenêtre et on stocke les résultats dans la matrice de sortie. La figure (2.12) montre un exemple de "max pooling" avec une fenêtre de taille (2,2).
Comme pour les couches de convolutions dans le cas des entrée 3D le pooling se fait sur un volume 3D et non pas sur une surface 2D [31].

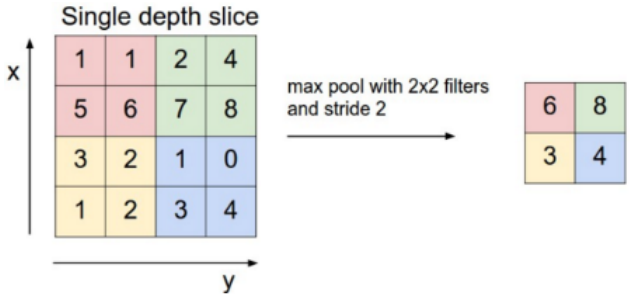


FIGURE 2.12: Illustration de l'opération pooling.

3. **Propagation vers l'avant pour la couche entièrement connectée**

L'information se propage de la couche d'entrée jusqu'à ce qu'elle arrive aux couches entièrement connectées (dense) en passant par toutes les couches convolutives et pooling. Pour faire passer l'information aux couches entièrement connectées, il faudrait une opération d'aplatissement (flattening operation).
L'aplatissement (figure 2.13) consiste simplement à mettre bout à bout toutes les entrées matricielles que nous avons pour en faire un long vecteur. Ce vecteur va être traité par les couches entièrement connectées pour générer une décision finale [28].
La propagation de l'information dans ces couches est comme pour les perceptrons multicouches.

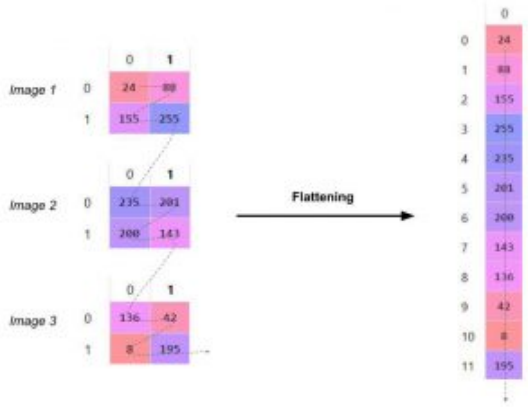


FIGURE 2.13: Illustration de l'opération flatening.

Rétropropagation du gradient

Pour l'apprentissage, nous allons commencer par calculer le gradient des paramètres de la dernière couche entièrement connectée, puis propager le gradient vers les entrées de cette couche (ce qui correspond au gradient de la sortie de la couche précédente). Ainsi, on peut calculer le gradient des paramètres de l'avant-dernière couche, puis le gradient de son entrée. Et ainsi de suite, couche à couche jusqu'à ce qu'on arrive aux couches polling et convolutives et ainsi de suite jusqu'à la première couche de convolution.

1. **Rétropropagation du gradient pour les couches entièrement connectées**

   Les couches entièrement connectées sont des perceptrons multicouches, alors la rétropropagation du gradient se fait comme la rétropropagation des perceptrons multicouches.

2. **Rétropropagation du gradient pour les couches convolutives**

   Comme nous l'avons décrit précédemment, l'opération de convolution peut être explicitée comme décrit dans la figure ci-dessous [29].

$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

$$O_{11} = F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22}$$
$$O_{12} = F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23}$$
$$O_{21} = F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32}$$
$$O_{22} = F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}$$

FIGURE 2.14: Illustration de l'opération de convolution.

Maintenant, pour calculer les gradients du filtre «F» par rapport à l'erreur «J»,

les équations suivantes doivent être résolues :

$$\frac{\partial J}{\partial F_{11}} = \frac{\partial J}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial J}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial J}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial J}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{11}} \tag{2.24}$$

$$\frac{\partial J}{\partial F_{12}} = \frac{\partial J}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial J}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial J}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial J}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{12}} \tag{2.25}$$

$$\frac{\partial J}{\partial F_{21}} = \frac{\partial J}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial J}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial J}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial J}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{21}} \tag{2.26}$$

$$\frac{\partial J}{\partial F_{22}} = \frac{\partial J}{\partial O_{11}}\frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial J}{\partial O_{12}}\frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial J}{\partial O_{21}}\frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial J}{\partial O_{22}}\frac{\partial O_{22}}{\partial F_{22}} \tag{2.27}$$

Cela est équivalent à résoudre les équations suivantes :

$$\frac{\partial J}{\partial F_{11}} = \frac{\partial J}{\partial O_{11}}X_{11} + \frac{\partial J}{\partial O_{12}}X_{12} + \frac{\partial J}{\partial O_{21}}X_{21} + \frac{\partial J}{\partial O_{22}}X_{22} \tag{2.28}$$

$$\frac{\partial J}{\partial F_{12}} = \frac{\partial J}{\partial O_{11}}X_{12} + \frac{\partial J}{\partial O_{12}}X_{13} + \frac{\partial J}{\partial O_{21}}X_{22} + \frac{\partial J}{\partial O_{22}}X_{23} \tag{2.29}$$

$$\frac{\partial J}{\partial F_{21}} = \frac{\partial J}{\partial O_{11}}X_{21} + \frac{\partial J}{\partial O_{12}}X_{22} + \frac{\partial J}{\partial O_{21}}X_{31} + \frac{\partial J}{\partial O_{22}}X_{32} \tag{2.30}$$

$$\frac{\partial J}{\partial F_{22}} = \frac{\partial J}{\partial O_{11}}X_{22} + \frac{\partial J}{\partial O_{12}}X_{23} + \frac{\partial J}{\partial O_{21}}X_{32} + \frac{\partial J}{\partial O_{22}}X_{33} \tag{2.31}$$

Cette équation ci-dessus peut être écrite sous la forme d'une opération de convolution donnée par la figure (2.15).

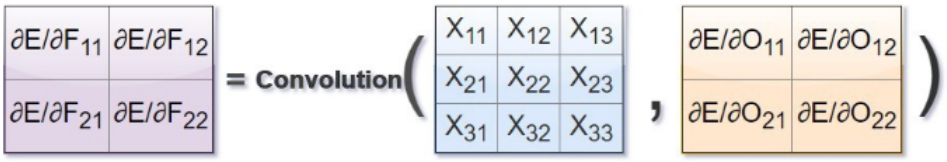


FIGURE 2.15: Calcul du gradient des filtres sous forme de convolution.

Nous pouvons trouver les gradients de l'entrée «X»' par rapport à l'erreur «J» de la même facon.

$$\frac{\partial J}{\partial X_{11}} = \frac{\partial J}{\partial O_{11}}F_{11} + \frac{\partial J}{\partial O_{12}}0 + \frac{\partial J}{\partial O_{21}}0 + \frac{\partial J}{\partial O_{22}}0 \tag{2.32}$$

$$\frac{\partial J}{\partial X_{12}} = \frac{\partial J}{\partial O_{11}}F_{12} + \frac{\partial J}{\partial O_{12}}F_{11} + \frac{\partial J}{\partial O_{21}}0 + \frac{\partial J}{\partial O_{22}}0 \tag{2.33}$$

$$\frac{\partial J}{\partial X_{13}} = \frac{\partial J}{\partial O_{11}}0 + \frac{\partial J}{\partial O_{12}}F_{12} + \frac{\partial J}{\partial O_{21}}0 + \frac{\partial J}{\partial O_{22}} \tag{2.34}$$

$$\frac{\partial J}{\partial X_{21}} = \frac{\partial J}{\partial O_{11}}F_{21} + \frac{\partial J}{\partial O_{12}}0 + \frac{\partial J}{\partial O_{21}}F_{11} + \frac{\partial J}{\partial O_{22}}0 \tag{2.35}$$

$$\frac{\partial J}{\partial X_{22}} = \frac{\partial J}{\partial O_{11}}F_{22} + \frac{\partial J}{\partial O_{12}}F_{21} + \frac{\partial J}{\partial O_{21}}F_{12} + \frac{\partial J}{\partial O_{22}}F_{11} \tag{2.36}$$

$$\frac{\partial J}{\partial X_{23}} = \frac{\partial J}{\partial O_{11}}0 + \frac{\partial J}{\partial O_{12}}F_{22} + \frac{\partial J}{\partial O_{21}}0 + \frac{\partial J}{\partial O_{22}}F_{11} \tag{2.37}$$

$$\frac{\partial J}{\partial X_{31}} = \frac{\partial J}{\partial O_{11}}0 + \frac{\partial J}{\partial O_{12}}0 + \frac{\partial J}{\partial O_{21}}F_{21} + \frac{\partial J}{\partial O_{22}}0 \tag{2.38}$$

$$\frac{\partial J}{\partial X_{32}} = \frac{\partial J}{\partial O_{11}}0 + \frac{\partial J}{\partial O_{12}}0 + \frac{\partial J}{\partial O_{21}}F_{22} + \frac{\partial J}{\partial O_{22}}F_{21} \tag{2.39}$$

$$\frac{\partial J}{\partial X_{33}} = \frac{\partial J}{\partial O_{11}}0 + \frac{\partial J}{\partial O_{12}}0 + \frac{\partial J}{\partial O_{21}}0 + \frac{\partial J}{\partial O_{22}}F_{22} \tag{2.40}$$

Le calcul ci-dessus peut être obtenu par un type différent d'opération de convolution appelée convolution complète. Afin d'obtenir les gradients de l'entrée, nous devons faire pivoter le filtre de 180 degrés et calculer la convolution complète (full convolution) du filtre pivoté par les gradients de la sortie par rapport à l'erreur, comme représenté dans les figures (2.16) et (2.17) [29].

FIGURE 2.16: Calcul du gradient de l'entrée sous forme de convolution complète.

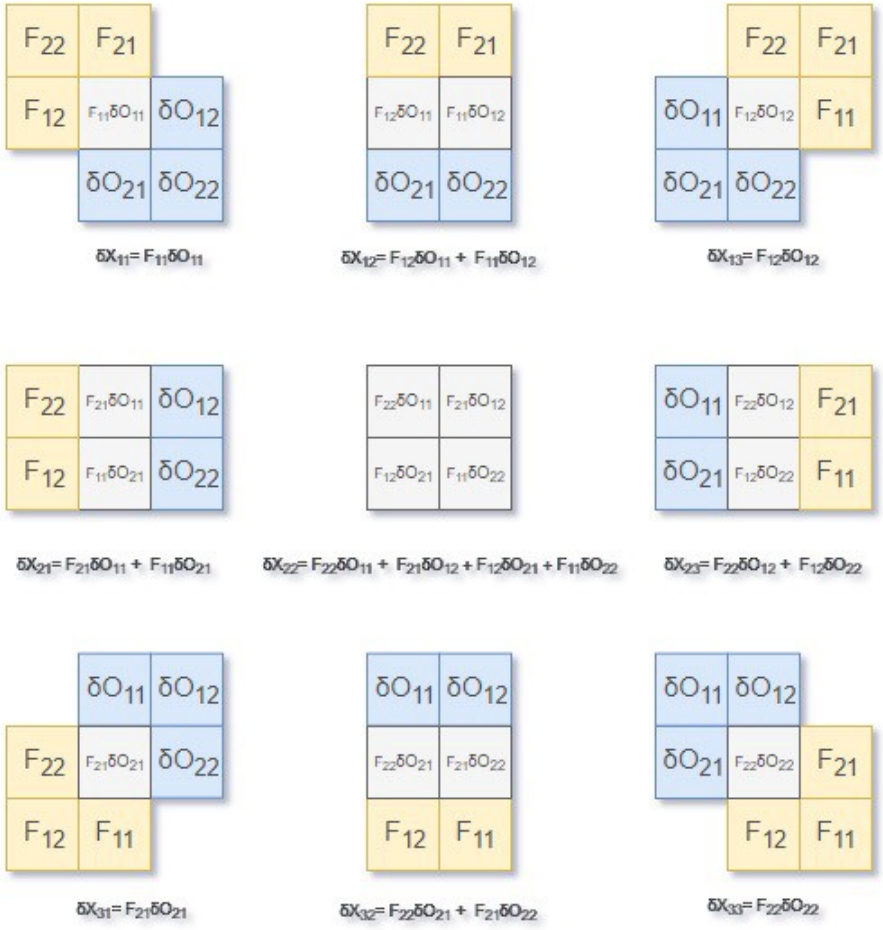


FIGURE 2.17: Convolution complète.

3. **Rétropropagation du gradient pour les couches pooling**

Même si la couche de pooling n'a pas de paramètres à mettre à jour, on doit toujours faire la rétropropagation à travers cette couche afin de calculer les gradients pour les couches qui l'ont précédée.
Pour calculer la rétropropagation à travers cette couche de pooling on aura besoin d'un Mask qui nous permettra de sélectionner les valeurs concernées par cette retropagation.

Le cas d'un «max pooling» : Dans ce cas, toute l'influence sur la sortie provenait d'un seul élément de la zone sélectionnée par la fenêtre en entrée de la couche pooling "la valeur maximale". On crée un masque qui indique la position du maximum dans l'entrée par Vrai(1) et le reste par Faux (0) [32].

$$X = \begin{Bmatrix} 1 & 3 \\ 4 & 2 \end{Bmatrix} \rightarrow M = \begin{Bmatrix} 0 & 0 \\ 1 & 0 \end{Bmatrix} \tag{2.41}$$

Le cas d'un «average pooling» : Dans ce cas, chaque élément de la zone sélectionée par la fenêtre en entrée de la couche pooling a une influence égale sur la sortie [32]. Par exemple, pour un filtre 2x2, le masque de rétropropagation à utilisé ressemble à :

$$M = \begin{Bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{Bmatrix} \tag{2.42}$$

Finalement la rétropropagation pour la couche pooling se calcul en multipliant le mask avec la rétropropagation de la couche qui vient après.

$$\frac{\partial E}{\partial I} = \frac{\partial E}{\partial O} * Mask \tag{2.43}$$

Avec I c'est l'entrée de la couche pooling et O c'est sa sortie.

2.8 Les réseaux neuronaux récurrents (RNN) et les réseaux récurrents à mémoire court et long terme (LSTM)

2.8.1 Introduction

Les réseaux récurrents (RNN) (figure 2.18) sont un type des réseaux de neurones qui sont adaptés à plusieurs tâches de traitement automatique des langues (en anglais Natural language processing « NLP »), notamment la prédiction des informations séquentielles.
Pour les réseaux RNN, l'information est donnée par une connexion en boucle ce qui permet de prendre en compte à l'étape courante les informations prédites dans les étapes précédentes. Ils sont appelés récurrents car ils effectuent la même tâche pour chaque bloc RNN [33].
Les réseaux récurrents à mémoire court et long terme (LSTM) (figure 2.19) sont un type des réseaux récurrents (RNN) capable de prendre en compte des dépendances à long terme et se souvenir des informations pendant une longue période. C'est l'architecture de réseau de neurones récurrents la plus utilisée en pratique et qui permet de répondre au problème de disparition de gradient (gradient vanishing) [33]. Ils ont été introduits par Hochreiter et Schmidhuber en 1997 [34].

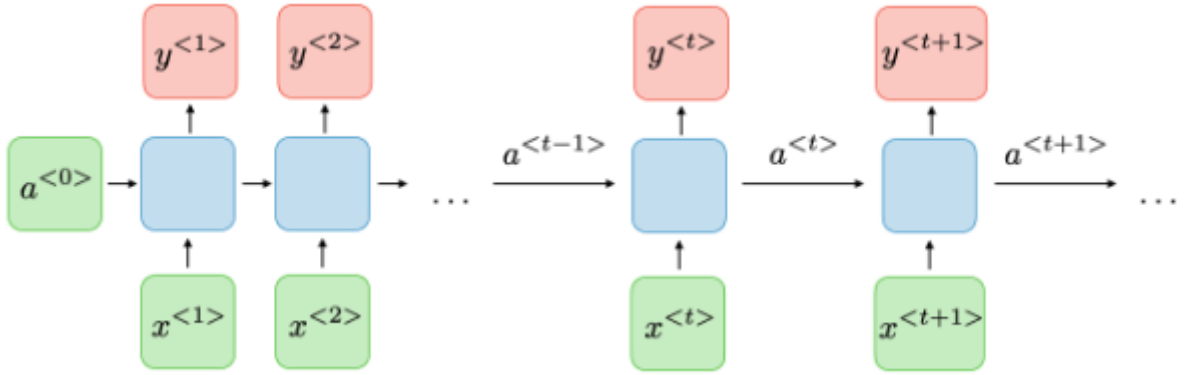


FIGURE 2.18: Un réseaux RNN

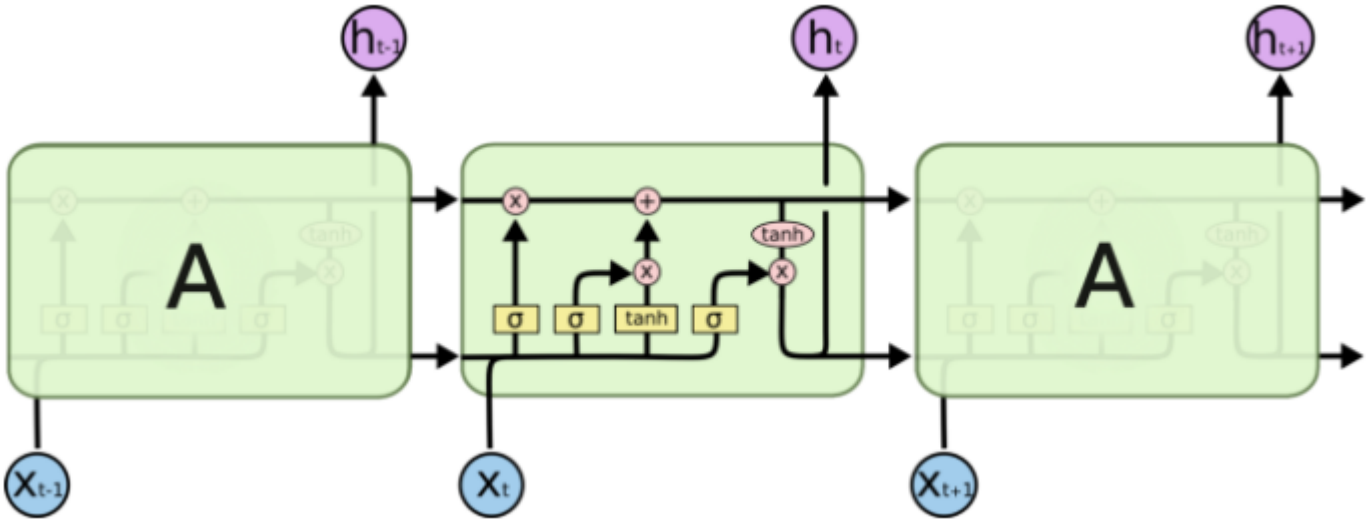


FIGURE 2.19: Un réseaux LSTM

Différents types de RNN

1. Un par un
   Cela est également appelé les réseaux de neurones Plain / Vanilla. Il traite des entrées et sorties de taille fixe et il n'y a pas une dépendance aux étapes précédentes [35].

   Exemple d'aplication : classification des images.

2. Un à plusieurs
   Il traite des entrées de taille fixe qui donne une séquence de données en sortie [35].

   Exemple d'aplication : sous-titrage d'image

3. Plusieurs à un
   Il prend une séquence de données en entrée et il donne une sortie de taille fixe [35].

   Exemple d'aplication : analyse de sentiment où une phrase donnée en entrée est classée comme exprimant un sentiment positif ou négatif.

4. Plusieurs à plusieurs
   Il prend une séquence de données en entrée qui donne une séquence de données en sortie [35].

   Exemple d'aplication : traduction automatique.
   La figure (2.20) illustre l'architecture de chaque type de réseau RNN :

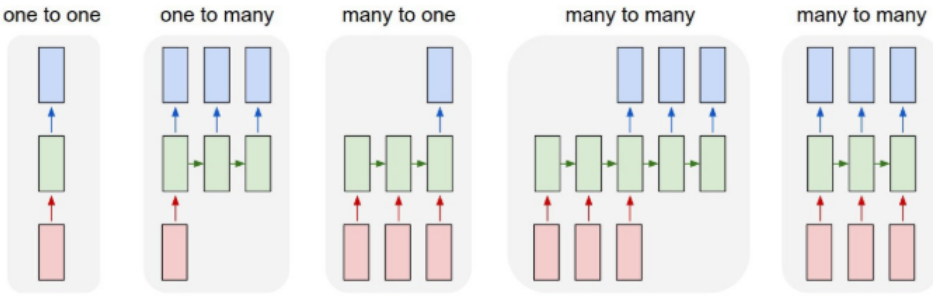


FIGURE 2.20: Différents types de RNN

2.8.2 Fonctionnement d'un réseaux de neurones récurrents RNN et LSTM.

Propagation vers l'avant pour les réseau récurrent RNN

Un réseau neuronal récurrent (figure 2.21) est une répétition d'une seule cellule. Cette cellule représente le cœur du réseau, toutes les opérations qui assurent le bon fonctionnement du réseau se font dedans [32].

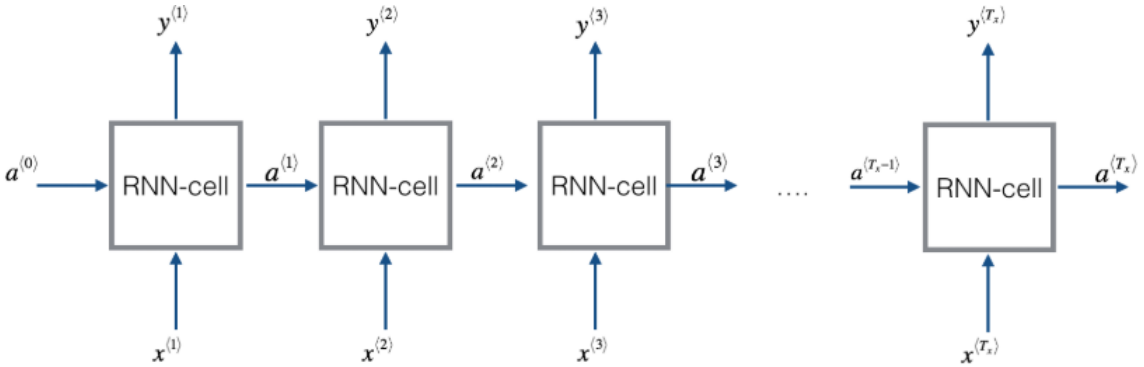


FIGURE 2.21: Exemple d'un réseaux RNN

- Les x représentent les entrées du réseau, peuvent être des mots dans le cas d'une translation.
- Les a représentent les sorties de la fonction d'activation, ils seront des entrées pour les blocs qui viennent après (prendre en considération les décisions prise par les blocs précédents 'effet séquentielle').
- Les y représentent les sorties du réseau (le nombre d'entrées et sorties peuvent être different).

  pour comprendre le fonctionnement de chaque cellule RNN, on donne la figure suivante :

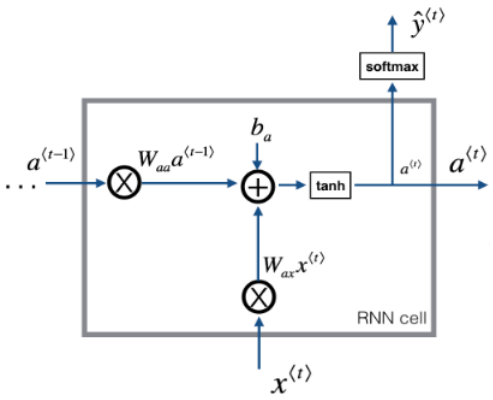


FIGURE 2.22: Un seul bloc RNN

Les sorties de la cellule RNN sont données par :

$$a^t = tanh(W_{ax}x^t + W_{aa}a^{t-1} + b_a) \tag{2.44}$$
$$y^t = softmax(W_{ya}a^t + b_y) \tag{2.45}$$

- $a^{t\text{-}1}$ représente la sortie de la fonction d'activation de la cellule précédente.
- Les W et b représentent respectivement les poids et les biais du réseau.

Les cellules RNN prennent en compte les informations courantes contenues dans l'entrée x, ainsi que les informations prédites dans les étapes précédentes contenu dans la fonction d'activation $a^{t\text{-}1}$.

Propagation vers l'avant pour les réseaux récurrents à mémoire court et long terme(LSTM)

Les LSTM sont conçus pour se souvenir des informations pendant de longues périodes, aussi pour éviter le problème que le gradient tend vers le zéro ou diverge(gradient vanishing) [32].
Dans la figure ci-dessous, on voit que chaque cellule LSTM a 3 portes : porte d'oubli, porte de mise à jour et porte de sortie.

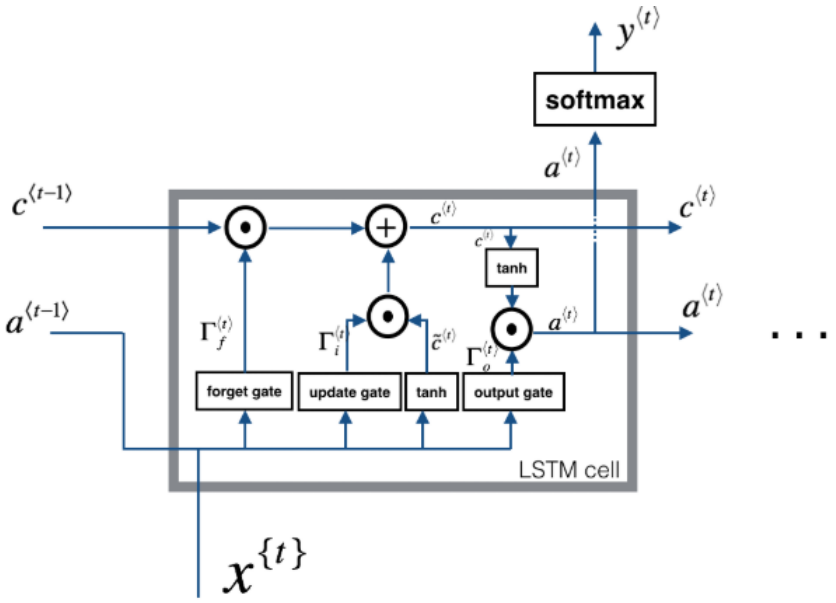


FIGURE 2.23: Une cellule LSTM

1. **Porte d'oubli (forget gate)**

   La porte d'oubli filtre les informations arrivant à la cellule LSTM provenant des cellules précédentes. Parmi ces informations, certaines vont être plus pertinentes que d'autre [32]. Pour décider quelles informations vont être autorisées à passer, nous allons calculer le masque donné par l'équation suivante :

$$T_f^t = \sigma(W_f.[a^{t-1}, x^t] + b_f) \tag{2.46}$$

   x^t correspond à l'entrée de la cellule LSTM, $a^{t\text{-}1}$ correspond à la valeur de sortie de la cellule LSTM précédente. D'un point de vue mathématique, [$a^{t\text{-}1}$,x^t] désigne la concaténation des deux matrices x^t et $a^{t\text{-}1}$. W_f et b_f représentent respectivement les poids et les biais de la porte d'oubli. Le résultat est passé comme paramètre à la fonction d'activation sigmoïde. En sortie, la fonction sigmoïde nous renvoie un entier compris entre 0 et 1.
   Pour savoir quelles valeurs de la cellule mémoire précédente $c^{t\text{-}1}$ vont être autorisées à passer, nous allons appliquer la multiplication terme à terme entre T_f et $c^{t\text{-}1}$. Si, lors de cette multiplication la valeur de T_f est proche de 1 alors les valeurs contenues dans $c^{t\text{-}1}$ sont autorisées à passer. Dans le cas contraire (la multiplication a donné une valeur proche de 0), les valeurs contenues dans $c^{t\text{-}1}$ sont ignorées. Toutes les valeurs qui ont réussi à passer sont stockées dans a^t [32].

2. **porte de mise à jour(update gate)**

   La porte de mise à jour va décider quelles nouvelles valeurs de l'entrée actuelle x^t vont être autorisées à passer. Certaines entrées vont être plus pertinentes que d'autres. Celles qui sont pertinentes vont être stockées dans la cellule mémoire qui va servir à la prise de décision à l'instant t et peut-être pour l'instant t+1 si la porte d'oubli lui autorise l'accès [32]. Comme pour la porte d'oubli, on doit calculer un masque T_u donné par l'equation (2.47), ce masque est nécessaires pour savoir quelles informations vont être autorisées à passer :

$$T_u^t = \sigma(W_u.[a^{t-1}, x^t + b_u]) \tag{2.47}$$

$$c'^t = tanh(W_u.[a^{t-1}, x^t] + b_u) \tag{2.48}$$

   c'^{t} contient les informations candidates à être stockées dans la cellule mémoire. La multiplication terme à terme entre T_u et c'^{t} va nous permettre de savoir quelles informations doivent être stockées dans la cellule mémoire.
   L'information présente dans la cellule mémoire c^t va dépendre de l'état interne antérieur $c^{t\text{-}1}$ et des nouvelles informations pertinentes c'^{t} [32].

$$c^t = T_f^t * c^{t-1} + \tau_u^t * c'^t \tag{2.49}$$

3. **Porte de sortie(output gate)**

   En fonction de l'entrée x(t) et les informations contenues dans la cellule mémoire précédente $c^{t\text{-}1}$, on calcule la fonction d'activation a^t pour la cellule LSTM par les équations suivantes[32]. :

$$T_o^t = \sigma(W_o.[a_{t-1}, x_t + b_o]) \tag{2.50}$$

$$a^t = T_o^t * tanh(c^t) \tag{2.51}$$

Rétropropagation du gradient pour les cellule RNN

Pour l'apprentissage du réseau RNN, nous allons commencer par calculer le gradient des paramètres de la dernière cellule, puis propager le gradient vers la cellule de l'entrée [32].
Chaque cellule reçoit le gradient par rapport à sa sortie et l'utilise pour calculer le gradient de ses paramètres.

1. D'abord, on doit calculer la fonction de perte pour le réseau RNN.

$$J = Loss(y, y') = \sum_{t=1}^{t=T} Loss(y^t, y'^t) \tag{2.52}$$

   Si on prend par exemple une fonction de perte logarithmique, les pertes pour chaque sortie du réseau seront de la forme :

$$J = Loss(y^t, y'^t) = -(y^t log(y'^t) + (1 - y^t) log(1 - y'^t)) \tag{2.53}$$

   Les pertes totales du réseau sont égales à :

$$Loss(y, y') = \sum_{t=1}^{t=T} -(y^t log(y'^t) + (1 - y^t) log(1 - y'^t)) \tag{2.54}$$

2. Ensuite, on passe aux gradients, les calculs sont montrés en détail d'aprés la figure (2.24) et les équations ci-dessous :

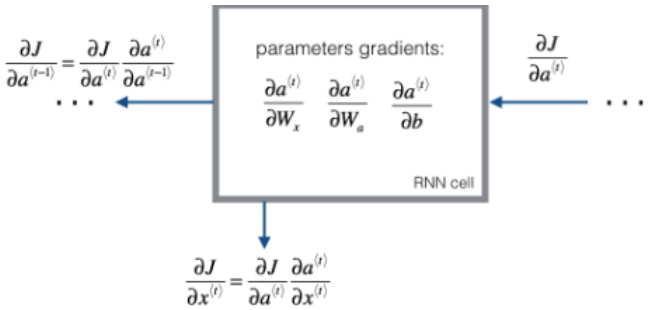


FIGURE 2.24: Gradient pour une cellule RNN

$$\frac{\partial J}{\partial W_{ax}} = \frac{\partial J}{\partial a^t} \frac{\partial a^t}{\partial W_{ax}} \tag{2.55}$$

$$\frac{\partial J}{\partial W_{aa}} = \frac{\partial J}{\partial a^t} \frac{\partial a^t}{\partial W_{aa}} \tag{2.56}$$

$$\frac{\partial J}{\partial b_a} = \frac{\partial J}{\partial a^t} \frac{\partial a^t}{\partial b_a} \tag{2.57}$$

$$\frac{\partial J}{\partial x^t} = \frac{\partial J}{\partial a^t} \frac{\partial a^t}{\partial x^t} \tag{2.58}$$

$$\frac{\partial J}{\partial a^{(t-1)}} = \frac{\partial J}{\partial a^t} \frac{\partial a^t}{\partial a^{(t-1)}} \tag{2.59}$$

- D'après la figure, les gradients des paramètres dépendent du gradient de la sortie de la fonction d'activation $\frac{\partial J}{\partial a^t}$. Ils sont calculés dans les équations ci-dessus en multipliant le gradient de cette sortie avec les gradients des paramètres.

- On sait que la sortie de la fonction d'activation de la cellule RNN est équivalente à :

$$a^t = tanh(W_{ax}x^t + W_{aa}a^{t-1} + b_a) \tag{2.60}$$

Et comme :

$$\frac{\partial(tanh(x))}{\partial x} = 1 - tanh(x)^2 \tag{2.61}$$

Alors :

$$\frac{\partial a^t}{\partial W_{ax}} = (1 - tanh(W_{ax}x^t + W_{aa}a^{t-1} + b_a)^2).(x^t)^T \tag{2.62}$$
$$\frac{\partial a^t}{\partial W_{aa}} = (1 - tanh(W_{aa}x^t + W_{aa}a^{t-1} + b_a)^2).(a^{t-1})^T \tag{2.63}$$
$$\frac{\partial a^t}{\partial b_a} = \sum((1 - tanh(W_{ax}x^t + W_{aa}a^{t-1} + b_a)^2) \tag{2.64}$$
$$\frac{\partial a^t}{\partial x^t} = (W_{ax})^T(1 - tanh(W_{ax}x^t + W_{aa}a^{t-1} + b_a)^2) \tag{2.65}$$
$$\frac{\partial a^t}{\partial a^{t-1}} = (W_{aa})^T(1 - tanh(W_{ax}x^t + W_{aa}a^{t-1} + b_a)^2) \tag{2.66}$$

Rétropropagation du gradient pour les cellules LSTM

La rétropropagation du gradient pour les cellules LSTM ressemble au cas des cellules RNN, mais elle est un peu compliquée. Les calculs sont démontrés ci-dessous.

Comme pour le cas d'une cellule RNN, les gradients des paramètres dépendent du gradient de la sortie de la fonction d'activation. Ils sont calculés en multipliant le gradient de cette sortie avec les gradients des paramètres par rapport à la sortie elle-même [32].

$$\frac{\partial J}{\partial T_o^t} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial T_o^t} \tag{2.67}$$
$$\frac{\partial J}{\partial c^t} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t} \tag{2.68}$$
$$\frac{\partial J}{\partial c'^t} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial c'^t} \tag{2.69}$$
$$\frac{\partial J}{\partial T_u^t} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial T_u^t} \tag{2.70}$$
$$\frac{\partial J}{\partial T_f^t} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial T_f^t} \tag{2.71}$$
$$\frac{\partial J}{\partial W_f} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial T_f^t}\frac{\partial T_f^t}{\partial W_f} \tag{2.72}$$
$$\frac{\partial J}{\partial W_u} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial T_u^t}\frac{\partial T_u^t}{\partial W_u} \tag{2.73}$$
$$\frac{\partial J}{\partial W_c} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial c'^t}\frac{\partial c'^t}{\partial W_C} \tag{2.74}$$
$$\frac{\partial J}{\partial W_o} = \frac{\partial J}{\partial a^t}\frac{\partial a^t}{\partial c^t}\frac{\partial c^t}{\partial T_o^t}\frac{\partial T_o^t}{\partial W_o} \tag{2.75}$$

Pour faciliter l'écriture, on note les dérivées partielles par rapport à l'erreur J par d.

$$dT_o^t = da_t * tanh(c_t) \tag{2.76}$$
$$dc^t = da_t * T_o^t * (1 - tanh(c^t)^2) \tag{2.77}$$
$$dc'^t = da_t * T_o^t * (1 - tanh(c^t)^2) * T_u^t \tag{2.78}$$
$$dT_f^t = da_t * T_o^t * (1 - tanh(c^t)^2) * c^{t-1} \tag{2.79}$$
$$dT_u^t = da_t * T_o^t * (1 - tanh(c^t)^2) * c'_t \tag{2.80}$$

les gradients des paramètres a mettre a jour sont donnés par :

$$dW_f = dT_f^t * T_f^t * (1 - T_f^t) * \begin{pmatrix} a_{t-1} \\ x^t \end{pmatrix}^T \tag{2.81}$$
$$dW_u = dT_u^t * T_u^t * (1 - T_u^t) * \begin{pmatrix} a_{t-1} \\ x^t \end{pmatrix}^T \tag{2.82}$$
$$dW_c = dc't * c't * (1 - tanh(c't)^2) * \begin{pmatrix} a_{t-1} \\ x^t \end{pmatrix}^T \tag{2.83}$$
$$dW_o = dT_o^t * T_o^t * (1 - T_o^t) * \begin{pmatrix} a_{t-1} \\ x^t \end{pmatrix}^T \tag{2.84}$$

2.9 Optimisation du réseau neuronal

2.9.1 Problème de sous-apprentissage (underfitting) et sur-apprentissage (overfitting)

Le sur-apprentissage (overfitting), et le sous-apprentissage (underfitting) (figure2.25) sont les causes principales des mauvaises performances des algorithmes de Machine Learning. Dans cette partie du chapitre, on va parler en détail sur ces deux problèmes et les solutions possibles [36].

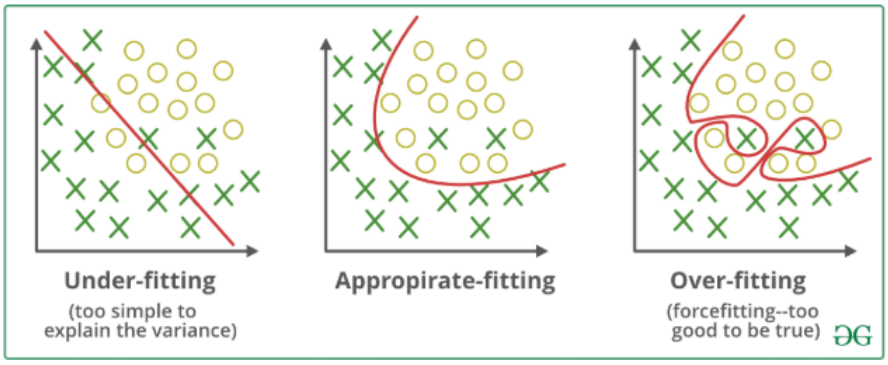


FIGURE 2.25: Exemple illustratif des problèmes sous-apprentissage et sur-apprentissage.

Sous-apprentissage(Underfitting)

Le problème de sous-apprentissage est caractérisé par un mal adaptation du modèle prédictif généré avec les données d'apprentissage lors de la phase d'apprentissage. Autrement dit, le modèle prédictif n'arrive même pas à capturer les corrélations des données d'apprentissage. Par conséquent, le coût d'erreur en phase d'apprentissage reste grand. On dit également qu'il souffre d'un grand biais (un grand écart entre la fonction de prédiction et les données d'apprentissage) [36].

Sur-apprentissage (Overfitting)

Sur-apprentissage (ou Overfitting en anglais) désigne le fait que le modèle prédictif produit par l'algorithme de Machine Learning s'adapte bien aux données d'apprentissage. Par conséquent, le modèle prédictif capturera tous les aspects et détails qui caractérisent les données d'apprentissage. Mais il prédira mal sur des données qu'il n'a pas encore vues lors de sa phase d'apprentissage, c.-à-d il donne des mauvaises prédictions sur les données du test. On dit que la fonction prédictive se généralise mal. Et que le modèle souffre de sur-apprentissage [36].
La figure 2.26 montre un exemple de sur-apprentissage. Le tracé en bleu représente une fonction de prédiction qui passe par toutes les données d'apprentissage (points en vert). On voit bien que la fonction est instable (grande variance) et qu'elle s'écarte beaucoup des points rouges qui représentent des données non vues lors de la phase d'apprentissage (les données de test).

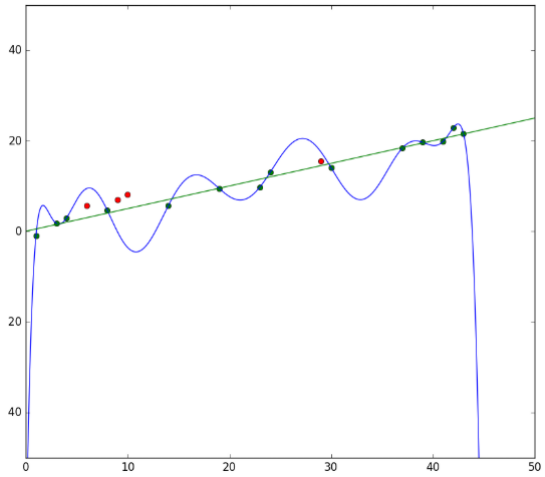


FIGURE 2.26: Exemple d'un sur-apprentissage

2.9.2 Solution pour sous-apprentissage et sur-apprentissage

- Sous-apprentissage se produit lorsqu'un modèle est trop simple. Les modèles simples sont représentés par des fonctions simples qui ont tendance à avoir moins de variances dans leurs prévisions, mais plus de biais. Pour résoudre ce problème de Sous-apprentissage il faudra plus de paramètres pour capturer toutes les fluctuations et variations aléatoires des données d'apprentissage, donc on aura besoin d'un modèle plus large (plusieurs couches).
- Sur-apprentissage se produit lorsque notre réseau de neurones représente une fonction trop complexe qui conduit à avoir un biais faible, mais une variance élevée. Pour résoudre ce problème, il faut diminuer la variance, donc on aura besoin d'un modèle plus simple.
- Il existe plusieurs techniques pour empêcher Sur-apprentissag (Overfitting), parmi ces techniques, on cite :

1. **Arrêt précoce (early stoping)**

   Lorsque vous entraînez un algorithme d'apprentissage de manière itérative, vous pouvez mesurer la performance de chaque itération du modèle. Jusqu'à un certain nombre d'itérations, de nouvelles itérations améliorent le modèle. Après ce point, la capacité du modèle à généraliser peut s'affaiblir car il commence à surcharger les données d'entraînement. L'arrêt précoce (figure 2.27) fait référence à l'arrêt du

processus de formation avant que l'apprenant dépasse ce point [36].

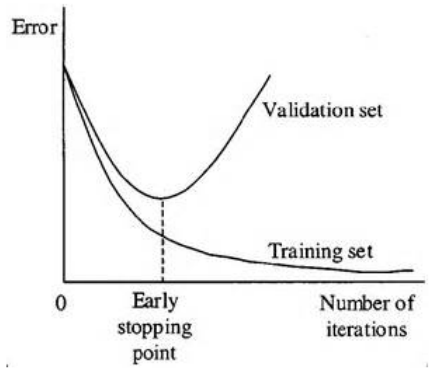


FIGURE 2.27: Arrêt précoce.

2. **Régularisation**

Afin d'éviter le problème de Sur-apprentissag (overfitting), certaines techniques de régularisation sont utilisées pour résoudre ce problème. Parmi ces techniques, on cite les suivantes :

Régularisation L1 et L2

- La régression Rigide ou régularisation L2, c'est une technique qui ajoute « l'amplitude au carré » des coefficients (poids W) comme terme de pénalité à la fonction de perte [37]. Cette technique est calculée par les équations suivantes :

$$L = \sum_{i=1}^{n}(y_i - y_i^{'})^2 + \frac{\lambda}{2m}\sum W_j^2 \tag{2.85}$$

$$dW^{'} = dW + \frac{\lambda}{m}W \tag{2.86}$$

$$W = W - \alpha dW^{'} = W(1 - \alpha\frac{\lambda}{m}) - \alpha dW \tag{2.87}$$

m : représente le nombre d'exemples utilisés a chaque itération d'apprentissage.

- La régression au lasso (Least Absolute Shrinkage and Selection Operator) ou régularisation L1 ajoute « la valeur absolue de l'amplitude » des coefficients comme terme de pénalité à la fonction de perte [37]. Cette technique est calculée par les équations suivantes :

$$L = \sum_{i=1}^{n}(y_i - y_i^{'})^2 + \frac{\lambda}{2m}\sum |W_j| \tag{2.88}$$

$$dW^{'} = dW + \frac{\lambda}{2m} \tag{2.89}$$

$$W = W - \alpha dW^{'} = W - \alpha(\frac{\lambda}{2m} + dW) \tag{2.90}$$

- si λ est égal à zéro, nous revenons à l'ancienne rétropropagation du gradient alors qu'une valeur très élevée rendra les coefficients nuls ce qui peut poser un

Problème d'underfitting. Donc il faut bien choisir la valeur de λ.

Drop-out
La régularisation "Dropout" élimine de façon probabiliste certains des nœuds d'une même couche afin de réduire les dépendances entre les couches. Le poids de la connexion sera exclu à la sortie du neurone, ce qui améliore considérablement la capacité de généralisation du modèle. Un modèle sans régularisation "Dropout" ajoute tous les poids au processus d'apprentissage pendant le processus d'entraînement, de sorte que la dépendance entre chaque couche du modèle est considérablement accrue, ce qui entraîne des problèmes de sur-apprentissage. "Dropout" peut être implémenté sur une ou toutes les couches cachées du réseau ainsi que sur la couche visible ou en entrée. Il n'est pas utilisé sur la couche de sortie.
Un nouveau hyperparamètre est introduit pour caractériser le "Dropout", cet hyperparamètre spécifie la probabilité à laquelle les sorties de la couche sont dropped out, ou inversement, la probabilité à laquelle les sorties de la couche sont conservées. Une valeur commune est une probabilité de 0,5 pour conserver la moitié des nœuds de la couche correspondante [38].

2.9.3 Normalisation par lots(Batchnormalization)

Souvent les couches d'entrée ont des plages de variation différentes. Pour accélérer l'apprentissage, il faut que ces entrées varient dans la même plage. Si les couches d'entrée en profite, pourquoi ne pas faire de même pour les valeurs des couches cachées, qui changent tout le temps, et obtenir 10 fois ou plus d'amélioration de la vitesse d'entraînement [39].
Pour augmenter la stabilité d'un réseau neuronal, la normalisation par lots normalise la sortie d'une couche d'activation en soustrayant ça moyenne et en la divisant par son écart-type (équation 2.93).
La normalisation par lots ajoute deux paramètres entraînables (équation 2.94) à chaque couche, de sorte que la sortie normalisée est multipliée par un paramètre «λ» et additionner avec un paramètre «β». En d'autres termes, la normalisation par lots permet à l'algorithme de descente de gradient de faire la normalisation en changeant ces 2 paramètres.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.91}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{2.92}$$

$$x_i^{'} = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.93}$$

$$y_i = \lambda x_i^{'} + \beta \tag{2.94}$$

2.9.4 Problèmes rencontrés avec descente de gradient et les solutions possibles

Il existe trois variantes de descente de gradient, qui diffèrent par la quantité de données que nous utilisons pour calculer le gradient des paramètres à mettre à jour.
La descente de gradient par lot calcule le gradient sur tout l'ensemble des données, ça peut être très lent parce qu'on aura moins d'itérations pour mettre à jour les paramètres.
Pour la descente de gradient stochastique "SGD", on utilise un seul exemple d'apprentissage avant de mettre à jour les gradients. Lorsque l'ensemble d'entraînement est grand,

la SGD peut être plus rapide, mais les paramètres "oscilleront" vers le minimum plutôt que de converger en douceur. La figure (2.28) montre la différence entre la SGD et la descente de gradient par lot.

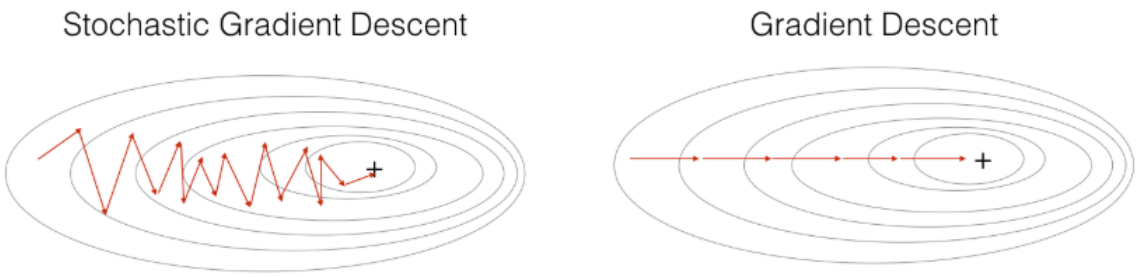


FIGURE 2.28: SGD vs GD.

En pratique, nous obtiendrons souvent des résultats plus rapides si nous n'utilisons ni l'ensemble complet de la formation, ni un seul exemple de formation pour effectuer chaque mise à jour. La descente en gradient en mini-lots utilise un nombre intermédiaire d'exemples pour chaque étape, ça limite la quantité de bruit propre aux SGD tout en restant plus efficace que le traitement de lots entiers. La figure (2.29) montre la différence entre la SGD et la descente de gradient par mini lot.

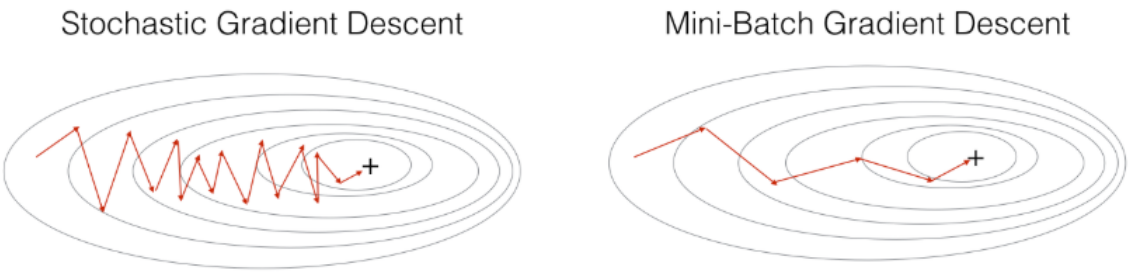


FIGURE 2.29: SGD vs SGD par lot.

Même avec la GD par mini-lot, on ne s'affranchit pas d'oscillation du gradient, on la minimise donc le problème du bruit reste

Le pas d'apprentissage influe beaucoup sur la convergence du gradient. Le choix d'un bon pas d'apprentissage peut être difficile. Un pas d'apprentissage trop faible conduit à une convergence douloureusement lente, tandis qu'un pas d'apprentissage trop important peut entraver la convergence et faire fluctuer la fonction de perte autour du minimum ou même diverger.

Momentum

Si on utilise la descente de gradient par mini-lot, A chaque itération, on se dirige vers les optimums avec des oscillations de haut en bas (figure 2.30). Si on utilise un pas d'apprentissage plus élevé, l'oscillation verticale aura une amplitude plus élevée. Ainsi, cette oscillation verticale ralentit notre descente de gradient et nous empêche d'utiliser un pas d'apprentissage beaucoup plus élevé. La descente de gradient avec momentum prend la moyenne des valeurs précédentes du gradient et les ajoutes à la valeur actuelle. Cela permet d'atténuer les oscillations verticales et l'algorithme va suivre le chemin plus

direct vers l'optimum [41].

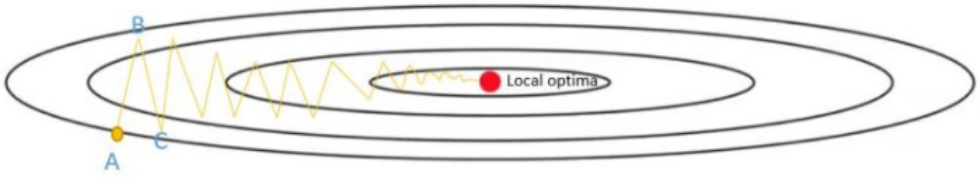


FIGURE 2.30: Illustration du phénomène d'oscillation pour la GD mini-lot.

Les calculs de descente de gradient se font comme ci-dessus :

$$V_{dW} = \beta * V_{dW} + (1 - \beta) * dW \tag{2.95}$$
$$V_{db} = \beta * V_{db} + (1 - \beta) * db \tag{2.96}$$
$$W = W - \alpha * V_{dW} \tag{2.97}$$
$$b = b - \alpha * V_{db} \tag{2.98}$$
$$\tag{2.99}$$

RMSprop

RMSprop et la descente de gradient avec momentum fonctionnent de la même manière, la seule différence, c'est la manière dont ils sont calculés [41].

$$V_{dW} = \beta * V_{dW} + (1 - \beta) * dW^2 \tag{2.100}$$
$$V_{db} = \beta * V_{db} + (1 - \beta) * db^2 \tag{2.101}$$
$$W = W - \alpha \frac{V_{dW}}{\sqrt{V_{dW}} + \epsilon} \tag{2.102}$$
$$b = b - \alpha \frac{V_{db}}{\sqrt{V_{db}} + \epsilon} \tag{2.103}$$

Adam

L'algorithme Adam (Adaptive Moment Optimization) combine les deux algorithmes Momentum et RMSProp pour optimiser la descente de gradient [42].
Les équations ci-dessous montrent comment cet algorithme est calculé :

$$V_{dW} = \beta * V_{dW} + (1 - \beta) * dW \tag{2.104}$$
$$S_{dW} = \beta * S_{dW} + (1 - \beta) * dW^2 \tag{2.105}$$
$$V_{db} = \beta * V_{db} + (1 - \beta) * db \tag{2.106}$$
$$S_{db} = \beta * S_{db} + (1 - \beta) * db^2 \tag{2.107}$$
$$W = W - \alpha \frac{V_{dW}}{\sqrt{S_{dW}} + \epsilon} \tag{2.108}$$
$$b = b - \alpha \frac{V_{db}}{\sqrt{S_{db}} + \epsilon} \tag{2.109}$$

2.10 Conclusion

Les progrès achevés par l'apprentissage en profondeur au cours des dernières années sont en grande partie dû aux énormes performances obtenus par les réseaux convolutifs CNN dans une variété de tâches de compréhension visuelle, telles que la détection d'objets, la reconnaissance faciale, la reconnaissance d'actions et d'activités, l'estimation de

la pose humaine, la récupération d'images et la segmentation sémantique, et aussi par les réseaux RNN dans des problèmes de reconnaissance vocale et traitement du langage naturel (NLP).
Les CNN ont la capacité d'apprendre automatiquement des fonctionnalités en fonction de l'ensemble de données fournit. Les CNN sont également invariants aux transformations, ce qui est un grand atout pour certaines applications de vision par ordinateur.
Les réseaux RNN et LSTM se souviennent de chaque information à travers le temps. Ils sont utilisés dans la prédiction de séries temporelles, en raison de la fonctionnalité permettant de mémoriser également les entrées précédentes. C'est ce qu'on appelle la mémoire à court terme dans le cas des RNN, long terme dans le cas des LSTM.

Chapitre 3

Classification des arythmies cardiaques avec un réseau CNN et un réseau CNN-LSTM

3.1 Introduction

Les réseaux de neurones convolutifs (CNN) qui effectuent à la fois des processus d'extraction de caractéristiques et de classification sont beaucoup utilisés dans les études d'apprentissage en profondeur. Un réseau CNN est composé par une succession de couches de convolution et de « pooling » pour entraîner un réseau profond permettant d'extraire les caractéristiques pertinentes de l'entrée. Comme les réseaux CNN ont donné des performances énormes dans le domaine de la vision par ordinateur, ils sont des candidats appropriés pour faire la reconnaissance de signaux ECG.
Les réseaux de mémoire à long et à court terme (LSTM) sont un autre type de réseaux de neurones profonds qui a été largement utilisé dans les récentes études d'apprentissage en profondeur notamment pour la prédiction des informations séquentielles.
Dans cette approche, nous utilisons deux réseaux d'apprentissage, le premier est le réseau de neurones convolutif (CNN 1D), le deuxième réseau est une architecture hybride basé sur l'ensemble (CNN + réseau récurrent LSTM).
Les deux réseaux sont formés et testés à l'aide de l'ensemble des signaux ECG dans le domaine temporel, qui sont obtenus à partir de la base de données MIT-BIH de Physionet pour classifier 7 types de battements.
Les caractéristiques automatiquement extraites remplacent spécifiquement les caractéristiques extraites manuellement.
Dans ce chapitre, nous présentons la classification des arythmies cardiaques avec un réseau CNN 1D et un réseau hybride basés sur un CNN-LSTM.

3.2 Classification des arythmies cardiaques

L'ECG joue un rôle important dans le diagnostic de diverses affections cardiaques. Le signal ECG à rythme irrégulier est connu sous le nom d'arythmie comme la contraction ventriculaire prématurée, la contraction prématurée auriculaire, bloc branche droit, etc. Le principal objectif de notre étude est de distinguer les différents types d'arythmie cardio-vasculaire, en utilisant l'algorithme d'apprentissage profond.
Dans cette approche, nous utilisons deux réseaux d'apprentissage, le premier est le réseau de neurones convolutif (CNN 1D) et le deuxième est basé sur l'ensemble (encoder CNN + réseau récurrent LSTM). Les deux réseaux sont formés et testés sur un ensemble de signaux ECG dans le domaine temporel, qui sont obtenus à partir de la base de données MIT-BIH de Physionet pour classifier 7 types de battements. Les caractéristiques automatiquement extraites remplacent spécifiquement les caractéristiques extraites manuellement.
Cette étude aidera les cardiologues à interpréter efficacement le signal ECG et à détecter le type d'arythmie cardiaque.

3.2.1 La base de données MIT-BIH Arrhythmia

La base de données MIT-BIH contient 48 extraits d'une demi-heure d'enregistrements d'ECG ambulatoires à deux canaux, obtenus de 47 sujets étudiés par le laboratoire d'arythmie BIH entre 1975 et 1979. Vingt-trois enregistrements ont été choisis au hasard parmi un ensemble de 4 000 enregistrements ECG ambulatoires de 24 heures, collectés auprès d'une population hétérogène de patients hospitalisés (environ 60%) et ambulatoires (environ 40%) à l'hôpital Beth Israel de Boston. Les 25 enregistrements restants ont été sélectionnés dans le même ensemble pour inclure des arythmies moins courantes mais cliniquement significatives. Les enregistrements ont été numérisés à 360 échantillons par

seconde et avec une résolution de 11 bits sur une plage de 10 mV. Deux cardiologues au moins ont annoté indépendamment chaque enregistrement ; les désaccords ont été résolus pour obtenir les annotations de référence lisibles par ordinateur pour chaque battement (environ 110 000 annotations au total) incluses dans la base de donnée [43].

3.2.2 L'extraction des données d'apprentissage de la base de données

La base de données MIT-BIH pourrait être téléchargée dans différents formats, adaptés à différents langages de programmation. Il comprend 48 enregistrements, chacun provenant d'un patient. Chaque enregistrement est constitué de quatre fichiers, le premier contient le signal (.dat file), le deuxième pour les annotations des battements de chaque enregistrement (.atr file), le troisième contient les différentes informations d'enregistrement (Le nom du patient, l'age, etc.".hea file") et le quatrième contient le lien d'une page web. Les annotations sont symbolisées par des lettres et leur signification est indiquée dans le Tableau (3.1). Chaque enregistrement dure plus d'une demi-heure, cette base de données est plus que suffisante pour l'apprentissage d'un réseau de neurones profond. Il est à noter que dans ce travail, on a pris en compte que les arythmies qui appartiennent aux classes dont on a parlé précédemment (on a négligé les arythmies qui ne se sont pas produites assez souvent.).

Types et occurrences des battements cardiaques dans la base de données MITBIH		
Annotation	Signification	Occurrence
N	Battement normal (un rythme sinusal normal)(NOR)	75052
A	Battement de contraction auriculaire prématurée (PAC)	2546
R	Battement de bloc de branche droit (RBB)	7259
L	Battement de bloc de branche gauche (LBB)	8075
V	Battement de contraction ventriculaire prématurée (PVC)	7130
F	Battement de flutter ventriculaire (VF)	472
!	Battement d'échappement ventriculaire (VE)	106

TABLE 3.1: La base de données MITBIH

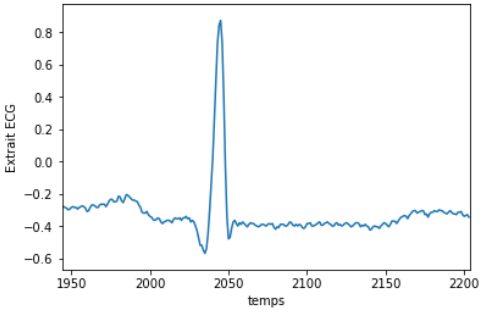


FIGURE 3.1: Battement APC

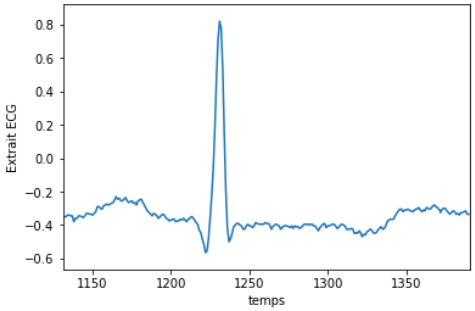


FIGURE 3.2: Battement NORM

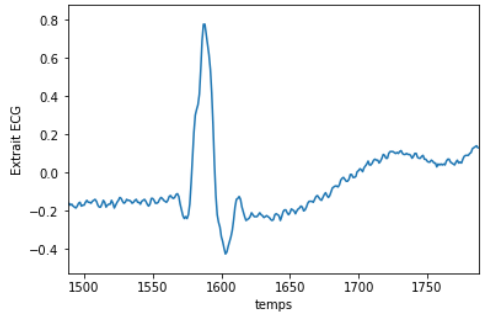


FIGURE 3.3: Battement RBB

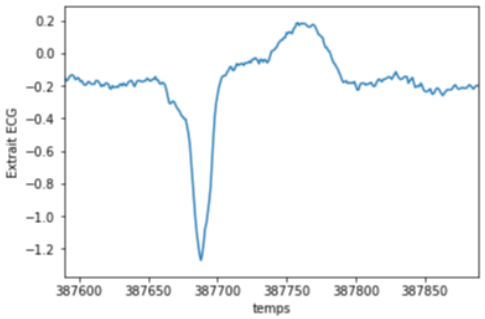


FIGURE 3.4: Battement LBB

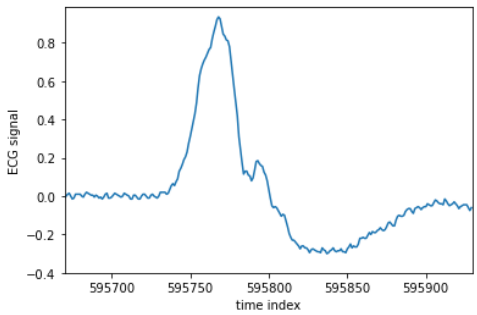


FIGURE 3.5: Battement VE

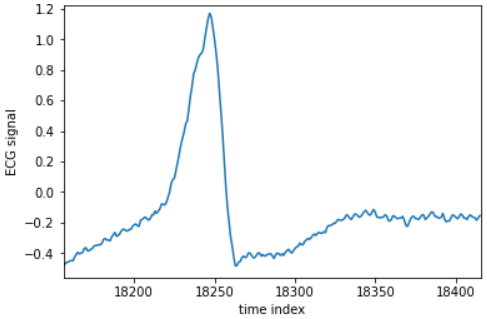


FIGURE 3.6: Battement VF

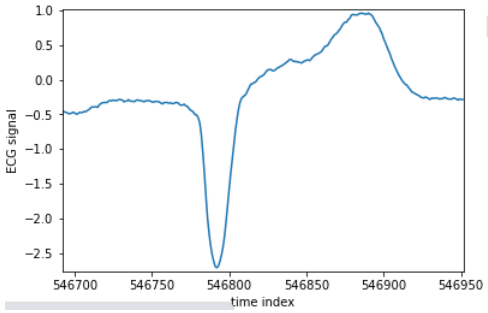


FIGURE 3.7: Battement PVC

Les figures précédentes représentent les différentes morphologies des arythmies à classifier pour une dérivation DII.

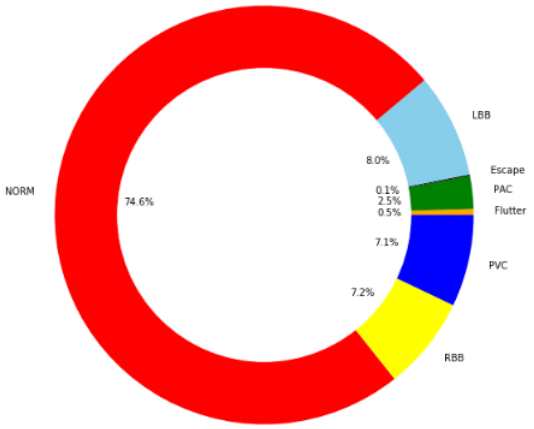
3.2.3 Traitement et préparation des données d'apprentissage

Pour exploiter les données contenu dans la base de données MIT-BIH on a besoin du framework wfdb, qui est un outil pour le traitement et l'analyse automatisée, la visualisation, l'annotation et l'analyse interactive des données de forme d'onde [44].

Chaque enregistrement ECG de la base de données "MIT-BIH" contient deux signaux différents, le premier est une dérivation DII, obtenue en plaçant les électrodes sur la poitrine. Le deuxième est une dérivation V1 (parfois V2 ou V5, V4). Comme le signal de la dérivation DII est de meilleur qualité par rapport à celui de la dérivation V1, on a choisi de l'utiliser pour la préparation de notre base de données.
Pour les données d'apprentissage, on doit segmenter ces enregistrements en petits extraits d'une durée de 6s (2160 échantillons). On a essayé avant avec des extraits qui contiennent le battement seul (260 échantillons) sans prendre les battements adjacents, mais on n'a pas obtenu des résultats assez bonnes pour le réseau CNN. Ces mauvais résultats sont dus

au fait qu'il existe des battements qui n'appartiennent pas à la même classe, mais qui ont des morphologies qui se ressemblent beaucoup. Par exemple, la classe NOR (battement normal) et la classe PAC (battement de contraction auriculaire) ont presque la même morphologie QRS, le seul moyen pour les différencier, c'est l'intervalle RR entre deux battements adjacents. Les battements de la classe PAC ont un intervalle RR plus petit que celui des battements de la classe NOR, c'est-à-dire que les battements de la classe PAC se produisent trop tôt par rapport aux battements de la classe NOR. Un autre exemple, c'est la répétitivité des battements de la classes PVC, cette caractéristique est importante pour les différenciers des battements des autres classes.
Les annotations indiquent la position du pic R et le type du battement, on utilise ces informations pour localiser la position de notre battement puis on fait la segmentation en prenant 1080 échantillons à gauche et a droite du pic R (2160 échantillons).
La base de données crée pour apprendre et tester notre réseau de neurones est composée de deux matrices, une pour les étiquettes (label) et une autre pour les battements à classifier. La matrice des battements est composée de 100 309 lignes représentant le nombre des segments dans la base de données et 2160 colonnes représentant la durée de chaque segment (6s). La matrice des étiquettes est aussi composée de 100 309 lignes représentant le nombre des segments et 7 colonnes représentant le codage one hot one des différents types à classifier.

FIGURE 3.8: La contribution de chaque classe dans la base de données crée.



Ce cercle est une illustration pour démontrer la contribution de chaque classe dans la base de données créer.

3.2.4 Le modèle d'apprentissage CNN 1D profond proposé

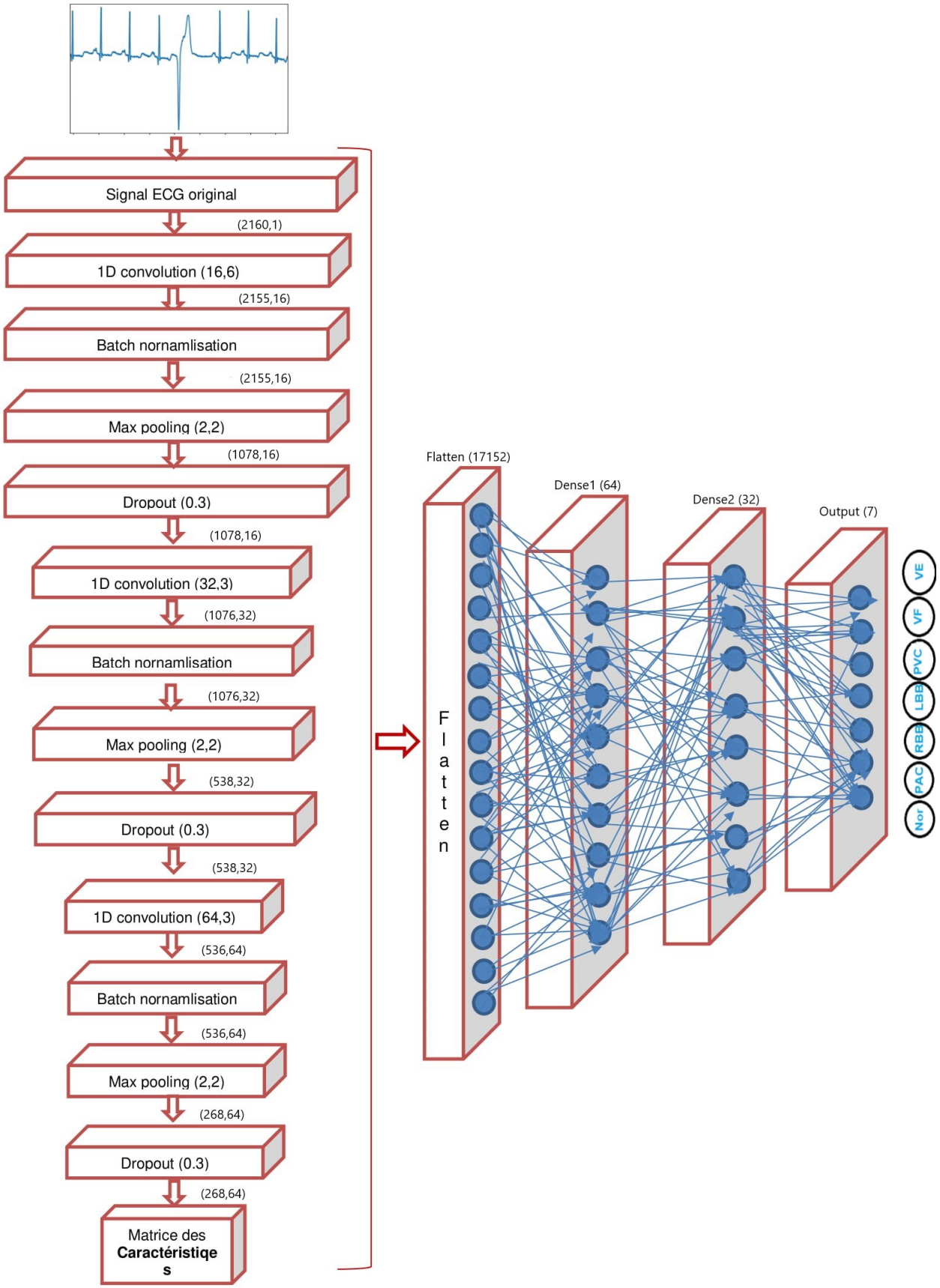


FIGURE 3.9: Architecture du réseau CNN utilisé

Dans ce travail, on va utiliser un réseau CNN à 1 dimension. Les CNN, qu'il s'agit de CNN 1D, 2D ou 3D, ils partagent les mêmes propriétés et la même approche. La plus grande différence est la dimensionnalité des données d'entrée et la façon dont les filtres de convolution ou détecteur de caractéristiques, sont utilisés pour parcourir les données.

L'architecture donnée par la figure (3.9) est composée de trois couches de convolution 1D qui servent à constituer les cartes des caractéristiques «The feature maps» avec une opération de convolution entre les données d'entrée et des filtres de taille spécifiée, toute en conservant la même taille d'entré à la sortie de cette couche (pas de "padding"). La taille de ces filtres ou kernels varie selon la complexité des caractéristiques à détecter. Pour la première couche, on utilise des filtres de taille grande (f=6) pour détecter des simples caractéristiques, plus on va en profondeur plus on doit détecter des caractéristique plus complexes, donc on utilise des filtres de taille petite (f=3) (détection des caractéristiques à différents niveaux d'abstraction, simples aux couches inférieures et caractéristiques très complexes aux couches plus profondes). Ces cartes des caractéristiques sont réduites par les couches de "pooling" (la méthode max pooling est utilisée dans cette étude). Par la suite, une couche de "Batchnormalization" a été utilisée pour normaliser les activations de la couche précédente.

La couche d'aplatissement (flatten) sert simplement à mettre bout à bout toutes les valeurs matricielles de la sortie des couches de convolution pour en faire un long vecteur et le passer aux couches entièrement connectées. Les couches entièrement connectées traitent ce vecteur pour générer une décision finale.

Toutes les couches de convolution dans le modèle et les deux avant-dernière couches entièrement connectée sont activées par des fonctions ReLU,la fonction softmax a été utilisé pour la dernière couche. la probabilité de dropout utilisé est $hp = 0.3$ pour éliminer 30% des neurones(couches entièrement connectée) ou des filtres (couches de convolution).

Les résultats obtenus par le réseaux CNN

Après traitement de la base de données, un ensemble de 100 309 segments de battement ont été choisis pour l'apprentissage du classifieur, 60% de ces derniers ont été utilisés pour l'apprentissage, 20% pour le test et les 20% restants pour la validation.
Pour l'optimisation on utilise l'algorithme adam, les données sont organisées en batch de taille 64. Après 10 époques d'apprentissage, on obtient les résultats suivantes :

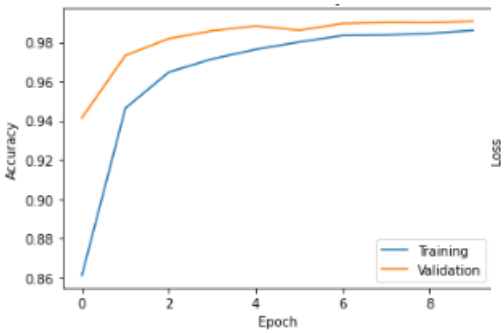


FIGURE 3.10: Battement NORM

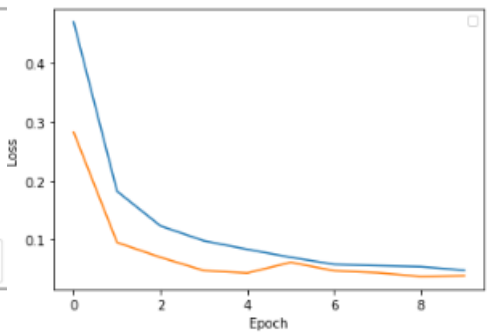


FIGURE 3.11: Battement NORM

La figure (3.10) montre la précision du réseau CNN à classifier les différents 7 classes pendant 10 époques d'apprentissage. La figure (3.11) est une mesure de l'erreur du même réseau en matière de sa capacité estimer la relation entre les sorties prédites et les vraies sorties.

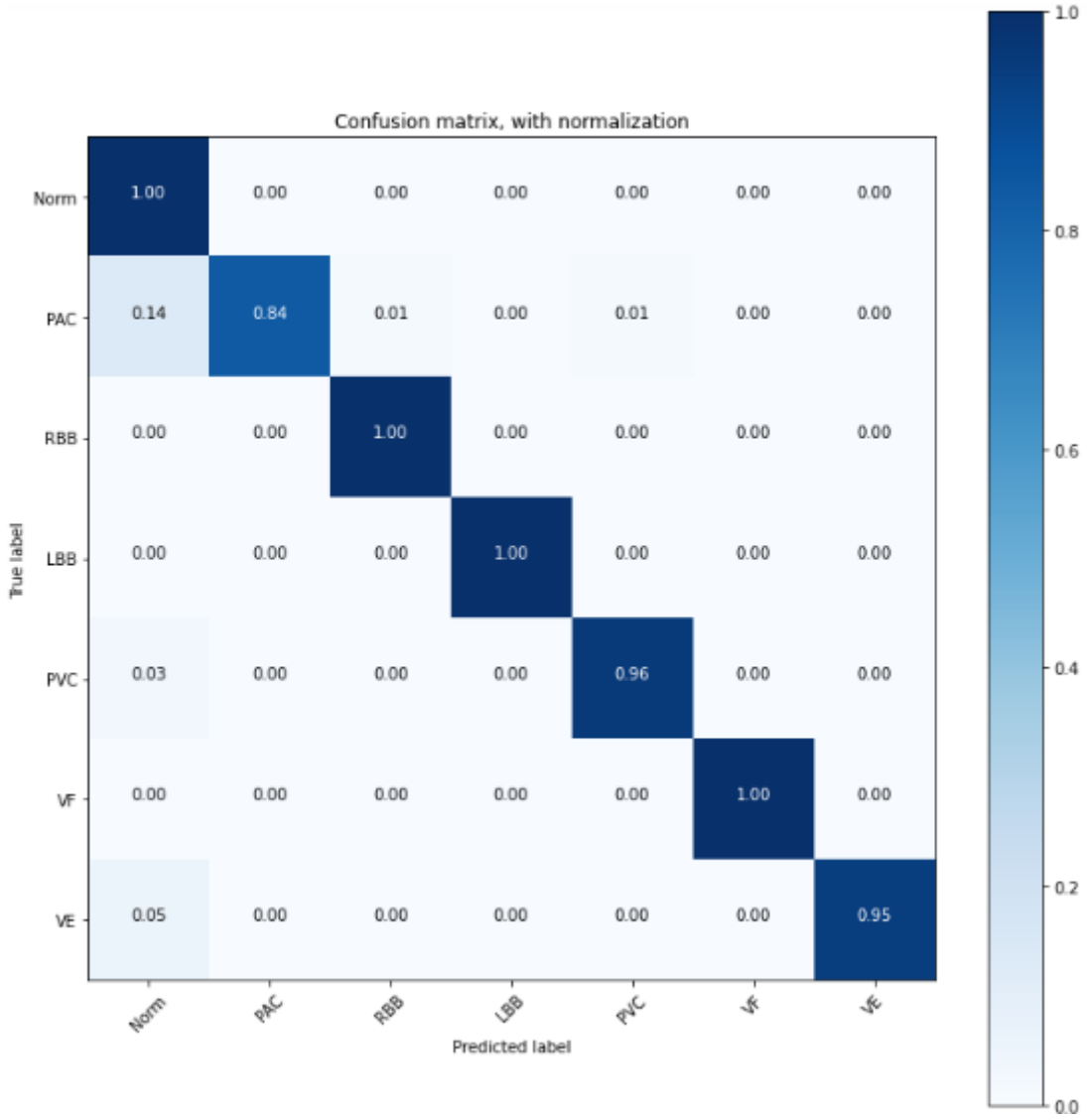


FIGURE 3.12: Matrice de confusion (Accuracy)

Certains critères d'évaluation tels que accuracy, la précision, le recall et le F1-score sont utilisé pour évaluer les performances de classification. Calculs de ces critères sont donnés comme ci-dessous :

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$
$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Pour ces équations, TP dénote le vrai positif, FP dénote le faux positif, TN désigne le vrai négatif et enfin FN désigne le faux négatif. Les performances de classement sur le 20 062 données de test et critère d'évaluation du réseau CNN, qui a utilisé les données brutes en entrée, est indiqué dans le tableau suivant.

Critère d'évaluation				
Classes	Precision	Recall	F1-score	Nombre des données de test
NOR	99.65%	99.58%	99.62%	14822
APC	83.99%	83.47%	83.73%	484
RBB	99.73%	99.66%	99.69%	1499
LBB	99.87%	99.64%	99.76%	1669
PVC	96.72%	96.13%	96.42%	1475
VF	99.99%	99.99%	99.99%	94
VE	94.73%	94.73%	94.73%	19

TABLE 3.2: Evaluation du réseau CNN

Selon le tableau, les performances les plus faibles ont été obtenues en reconnaissance des données de classe APC. Il est évident que les performances de reconnaissance des autres classes sont assez élevées. Des valeurs moyennes de 94% et plus sont mesurées pour tous les critères d'évaluation. Afin de mieux comprendre les résultats du test, la matrice de confusion obtenue pour les 20 062 données de test est donnée à la Figure (3.12).
Le réseau CNN profond avait une accuracy moyenne de 98.92% sur les 20 062 données de test des éléments. Le réseau CNN a mal classé 67 Données APC dans la classe NOR et 5 dans RBB et 5 dans PVC. De même, 44 signaux PVC ont été inclus dans la classe NOR et 1 signal VE a été inclus dans la classe NOR, entraînant les performances de reconnaissance les plus faibles entre ces trois classes.

3.2.5 Le modèle d'apprentissage hybride CNN-LSTM profond

Dans cette étude, deux approches ont été proposées pour la détection automatiquement de sept classes d'arythmies différentes présentes dans les signaux ECG. Les arythmies sont erratiques et irrégulières, elles peuvent se manifester en un ou plusieurs battements. Par conséquent, les différents réseaux proposés traitent de tels signaux de différentes tailles, petite (260 échantillons) et grande (2160 échantillons). Pour répondre à cette exigence, deux des quatre réseaux proposés sont des modèles hybrides d'apprentissage profond qui combinent les réseaux CNN et LSTM, et les deux autres sont des modèles d'apprentissage profond basés uniquement sur le réseau LSTM.

Segmentation des données

Pour la segmentation des impulsions ECG, on a utilisé deux types de segmentation. Dans le premier type, chaque segment contient un seul battement de 260 échantillons, avec 99 échantillons avant le pic R et 161 échantillons après le pic R. Dans le deuxième type, chaque segment contient le battement à classifier plus 6 à 7 de ses battement adjacents, d'une longueur de 2160 échantillons, avec 1080 échantillons avant le pic R du battement à classifier, et 1080 échantillons après le pic R.

Architecture de réseau CNN-LSTM profond proposé

La figure (3.13) illustre l'architecture du réseau CNN-LSTM dans les 2 cas : 260 échantillons et 2160 échantillons.

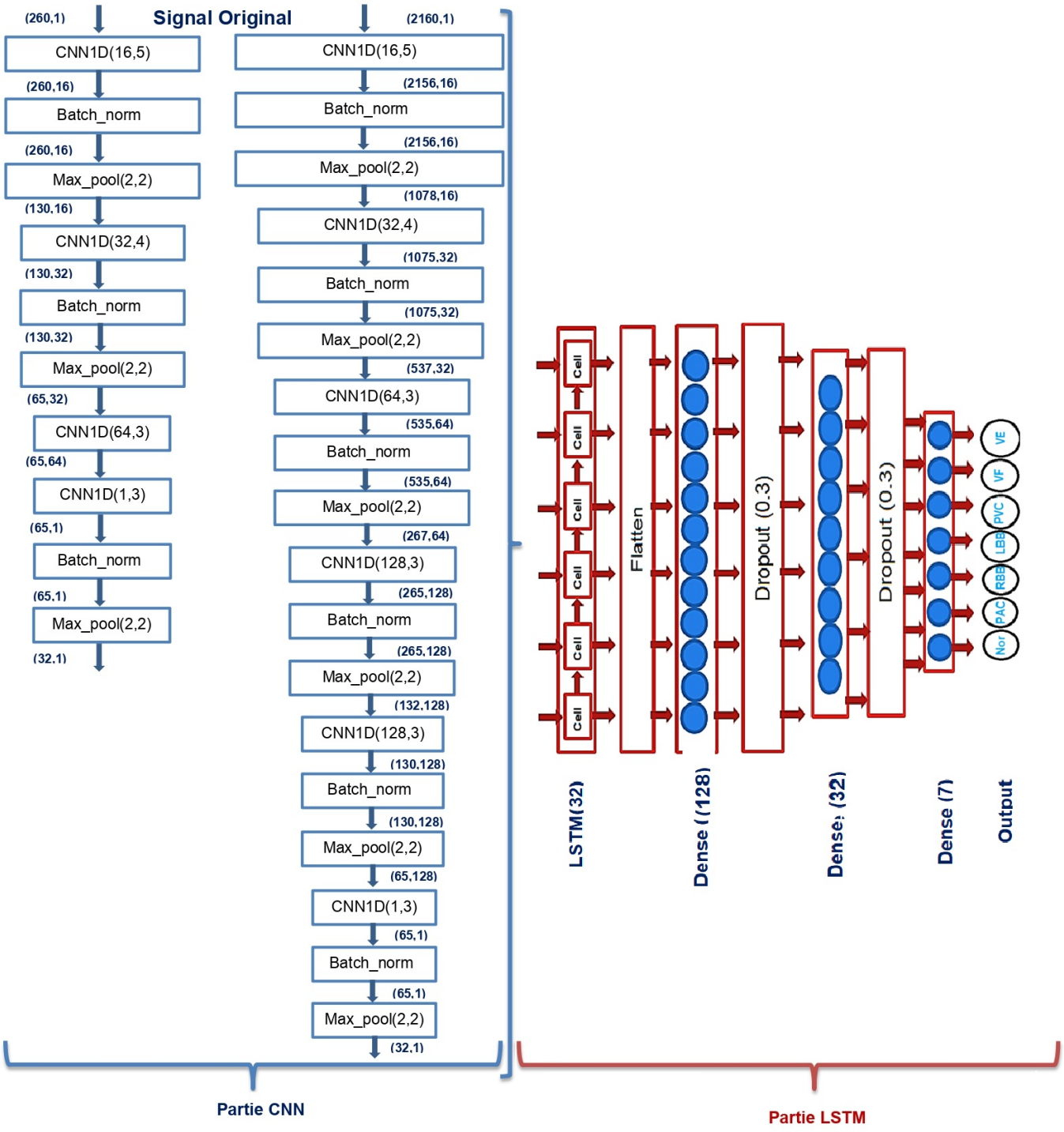


FIGURE 3.13: Architecture du réseau CNN-LSTM

Dans le cas du réseau LSTM, que la partie LSTM de l'architecture est utilisée.
La partie CNN de l'architecture est composée de 18 couches (cas de 2160 échantillons) et de 9 couches (cas de 260 échantillons), qui sont composées de couches de convolution avec des paramètres du kernel de taille spécifique (le nombre de filtres et leurs tailles), couplées à des couches de "Batchnormalization" pour normaliser les activations sortant des couches de convolution, suivaient des couches "max pooling" pour réduire la taille des cartes des caractéristiques. En conséquence, des caractéristiques de taille réduite 32*1 à partir des 2160*1 ou 260*1 signaux d'entrée segmentés ont été obtenus au niveau de la sortie de cette partie.
Ensuite, ces signaux de sortie sont présentés en entrée de la couche LSTM à 32 unités,

suivie d'une couche aplatie "flatten" et une série de couches entièrement connectées, utilisées pour prédire la sortie, et des couches "drop-out" pour empêcher les neurones de s'adapter excessivement bien aux données d'apprentissage, et donc empêcher le problème du sur-apprentissage.
Les couches de convolution permettent l'extraction des cartes des caractéristiques "feature maps" spatiales, tandis que la couche LSTM aide le modèle à capturer la dynamique temporelle présente dans ces cartes de caractéristiques. Les segments ECG sont classés en fonction des sorties de la couche LSTM aplatie via les couches entièrement connectées.

Entraînement et évaluation des données

Pendant la phase d'entraînement des modèles, 70% des signaux ECG ont été utilisés pour l'apprentissage, 15% pour la validation (La validation hors échantillon) et les 15% restants pour le test.
Les poids du réseau ont été initialisés en utilisant l'algorithme de Xavier (Glorot) [45], puis le modèle a été formé de bout en bout pour 50 époques d'apprentissage en utilisant l'algorithme de rétropropagation avec une taille de "batch" de 128. Le taux d'apprentissage du modèle est fixé à 0,001 et est utilisé en conjonction avec l'optimiseur Adam pour accélérer le processus d'apprentissage du réseau. Ensuite, les modèles formés ont été évalués avec les données de test.

Résultats de la classification

L'outil d'apprentissage en profondeur Keras [46], basé sur le langage de programmation Python, a été utilisé. Tensorflow [47] a été utilisé comme "back-end" de la bibliothèque Keras. Toutes les études expérimentales ont été effectuées sur un ordinateur doté d'un processeur Intel Core i7- 7500U 2,70 GHz, 16 Go de mémoire et 8 Go de carte graphique NVIDIA.
Quatre structures de réseau différentes ont été préparées pour l'évaluation des performances du système de classification proposé. La différence entre ces structures réside dans la taille des signaux d'entrée (260 échantillons ou 2160 échantillons) et leurs types (taille réduite ou brutes) et le type d'architecture (LSTM ou CNN-LSTM).
L'objectif était d'évaluer le succès de la structure convolutive comme extracteur de caractéristiques reduites, et pour évaluer l'influence de la taille du signal d'entrée (plusieur battements / un seul battement) et leurs types (taille réduite ou brute) sur la précision de classification.
Les figures (3.14),(3.15) montrent les performances d'entraînement et de validation des 4 réseaux pendant 50 époques d'apprentissage avec une taille du "batch" de 128. Dans chaque figure deux courbes sont tracées, la courbe de performance d'entrainement et la courbe de performance de validation. On remarque que les meilleures performances ont été obtenues pour le cas du réseau hybride avec 2160 échantillons, où les 2 courbes remontent d'une façon parallèle, ainsi que la courbe de validation présente moins de fluctuation, par rapport au autre cas.
Les figures(3.16), (3.17) montrent une mesure de l'erreur des 4 modèles en matière de leurs capacités à estimer la relation entre les sorties prédites et les vraies sorties pendant les 50 époques d'apprentissage. On remarque que pour les 4 réseaux l'erreur d'entrainement continue à diminuer pour converger vers le 0, tandis que l'erreur de validation diminue jusqu'à une valeur proche de 0.05 où elle demeure stable (le minimum de pertes a été obtenu pour le cas de 2160 échantillons).

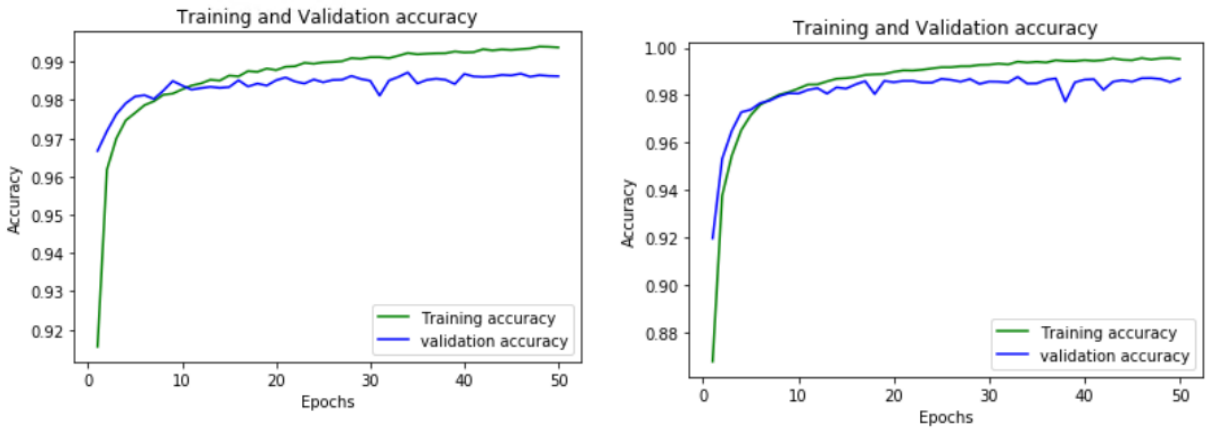


FIGURE 3.14: Cas de 260 échantillons à gauche avec un réseau LSTM et à droite avec un réseau CNN-LSTM

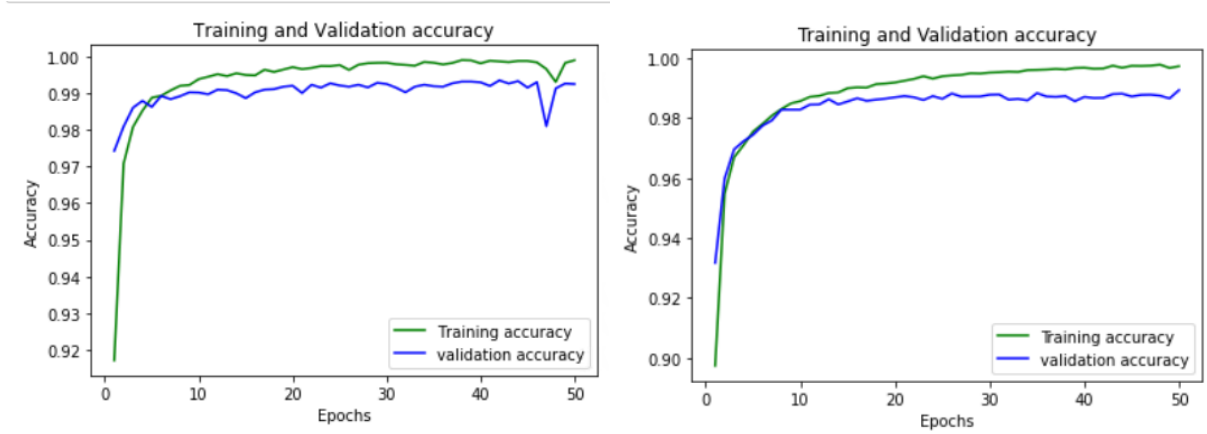


FIGURE 3.15: Cas de 2160 échantillons à gauche avec un réseau LSTM et à droite avec un réseau CNN-LSTM

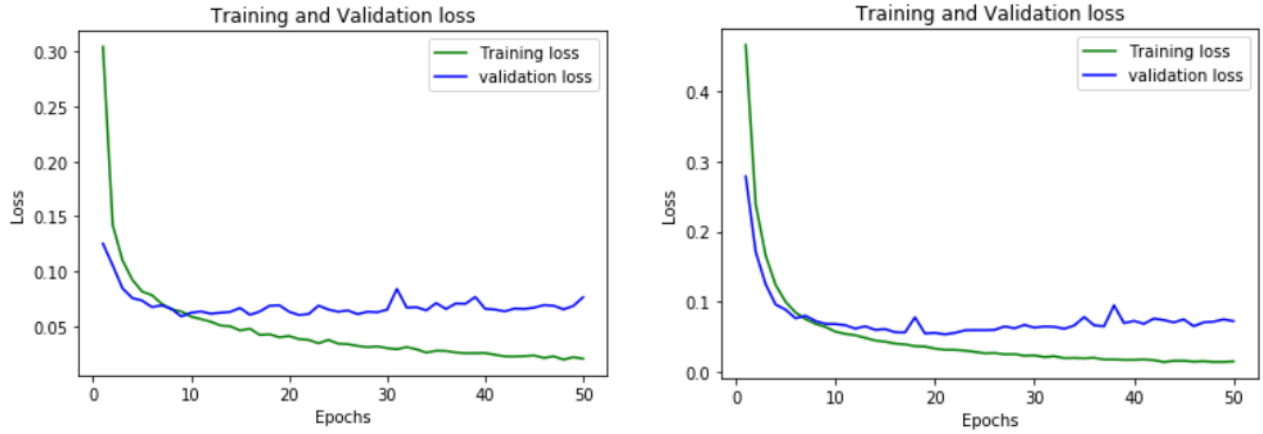


FIGURE 3.16: Cas de 260 échantillons à gauche avec un réseau LSTM et à droite avec un réseau CNN-LSTM

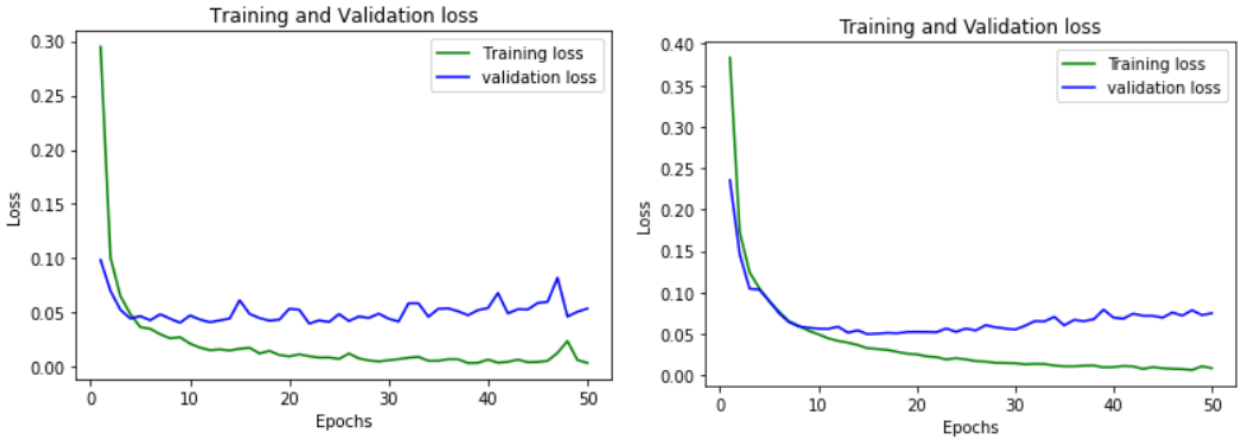


FIGURE 3.17: Cas de 2160 échantillons à gauche avec un réseau LSTM et à droite avec un réseau CNN-LSTM

Le tableau [3.3] montre les valeurs des performances de test et d'entrainement des différents réseaux utilisés et le temps nécessaire pour l'apprentissage de ces réseaux.

Les modèles	nombre d'échantillons	Temps d'apprentissage	performance de test	performance d'entrainement
LSTM	260	32500s (9.02h)	98.77%	98.72%
CNN+LSTM	260	11500s (3.19h)	98.82%	98.78%
LSTM	2160	152496s (42.36h)	99.34%	99.35%
CNN+LSTM	2160	23724s (6.59h)	99.50%	99.60%

TABLE 3.3: Comparaison entre les performances des quatre réseaux

Aucun des 4 réseaux n'a connu un vrai problème de sur-apprentissage mais une légère différence entre l'erreur d'entraînement et de validation, qui tend vers 0 dans le 1er cas et reste presque constante à une valeur proche de 0 pour le 2e cas.
Ainsi que, la comparaison des performances de test et d'entraînement cité dans le tableau [3.3], qui sont très proches et parfois les performances de test surmonte celle d'entrainement. Ce qui nous permet de confirmer la non-existence de problèmes de sur-apprentissage.
De plus, chacun des réseaux a fait preuve d'une performance élevée en matière d'apprentissage et de test, allant jusqu'à 99.60%. L'un des critères de performance les plus importants qui distingue ces réseaux a été le temps d'apprentissage. Le tableau[3.3] montre le temps nécessaire pour l'entrainement de chaque réseaux. Ce qui nous permet de conclure que l'entrainement sur des données de taille réduite réduit considérablement le temps de calcul. et les réseaux les plus rapide et bien efficace sont les réseaux hybride CNN-LSTM.
Les données de test, qui ne sont pas utilisées pour l'entrainement, sont fournies aux réseaux entraînés pour la tâche de reconnaissance automatique.
Certains critères d'évaluation tels que la précision, le rappel et le F1 score sont utilisés pour évaluer les performances de classification. Les résultats d'évaluation des performances de la classification sont présentés dans les tableaux [3.4],[3.5] et [3.6].

évaluation des différents réseaux				
Classes	260ech LSTM	260 ech CNN-LSTM	2160ech LSTM	2160 ech CNN-LSTM
NOR	99.6%	99.53%	99.74%	99.79%
APC	83.54%	85.29%	89.32%	93.42%
RBB	99.45%	99.55%	99.44%	99.86%
LBB	99.48%	99.66%	99.61%	99.49%
PVC	95%	95.43%	97.49%	98.60%
VF	91.55%	91.32%	95.77%	99.99%
VE	93.74%	96.77%	99.99%	92.86%

TABLE 3.4: Evaluation des quatre réseau selon le critère F1 score

évaluation des différents réseaux				
Classes	260ech LSTM	260 ech CNN-LSTM	2160ech LSTM	2160 ech CNN-LSTM
NOR	99.58%	99.52%	99.72%	99.79%
APC	83.54%	85.29%	89.32%	93.42%
RBB	99.36%	99.55%	99.44%	99.81%
LBB	99.48%	99.66%	99.61%	99.45%
PVC	94.92%	95.30%	97.49%	98.51%
VF	91.54%	91.32%	95.77%	99.99%
VE	93.74%	93.75%	99.99%	92.86%

TABLE 3.5: Evaluation des quatre réseau selon les critères rappel

évaluation des différents réseaux				
Classes	260ech LSTM	260 ech CNN-LSTM	2160ech LSTM	2160 ech CNN-LSTM
NOR	99.62%	99.55%	99.75%	99.79%
APC	83.54%	85.29%	89.32%	93.42%
RBB	99.45%	99.55%	99.44%	99.90%
LBB	99.48%	99.66%	99.61%	99.53%
PVC	95.09%	95.57%	97.49%	98.69%
VF	91.55%	91.32%	95.77%	99.99%
VE	93.75%	99.99%	99.99%	92.86%

TABLE 3.6: Evaluation des quatre réseau selon le critère précision

Selon les trois tableaux, Dans le cas de 260 échantillons en entrée, la performance la plus faible a été obtenue en reconnaissance des données de la classe APC. tandis que dans le cas de 2160 échantillons, la performance la plus faible a été obtenue en reconnaissance des données de la classe VE. Tandis que la performance de reconnaissance des autres classes est généralement meilleur. Afin de mieux comprendre les résultats des tests. Les matrices de confusion obtenue pour les données de test sont illustrés par les figures (3.18) et (3.19). Dans le cas de 260 échantillons, le réseau LSTM a classé par erreur 55 des 401 données APC dans la classe Normale. ce qui a entraîné la plus faible performance de reconnaissance pour cette classe APC. De même, le réseau CNN-LSTM a mal classifié 43 des 401 observations de la classe APC, mais avec des performances significatives pour les autres classes.

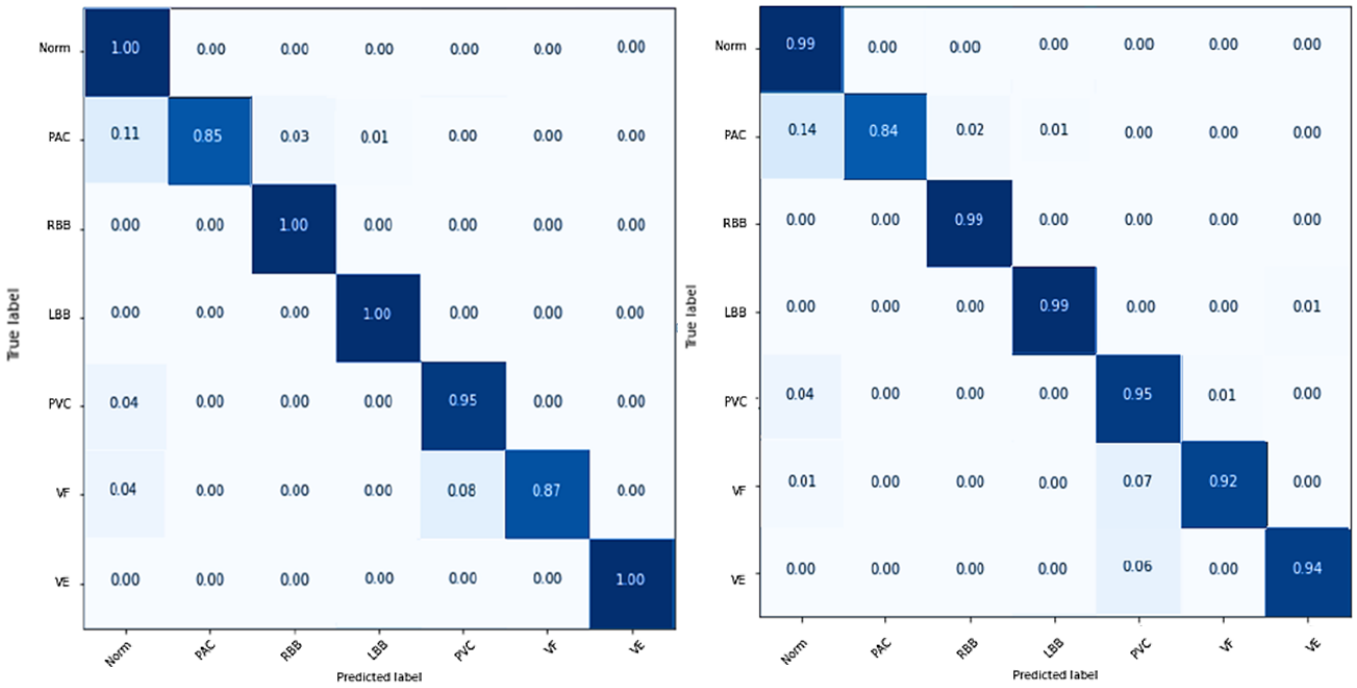


FIGURE 3.18: Cas de 260 échantillons à gauche avec un réseau LSTM et à droite avec un réseau CNN-LSTM

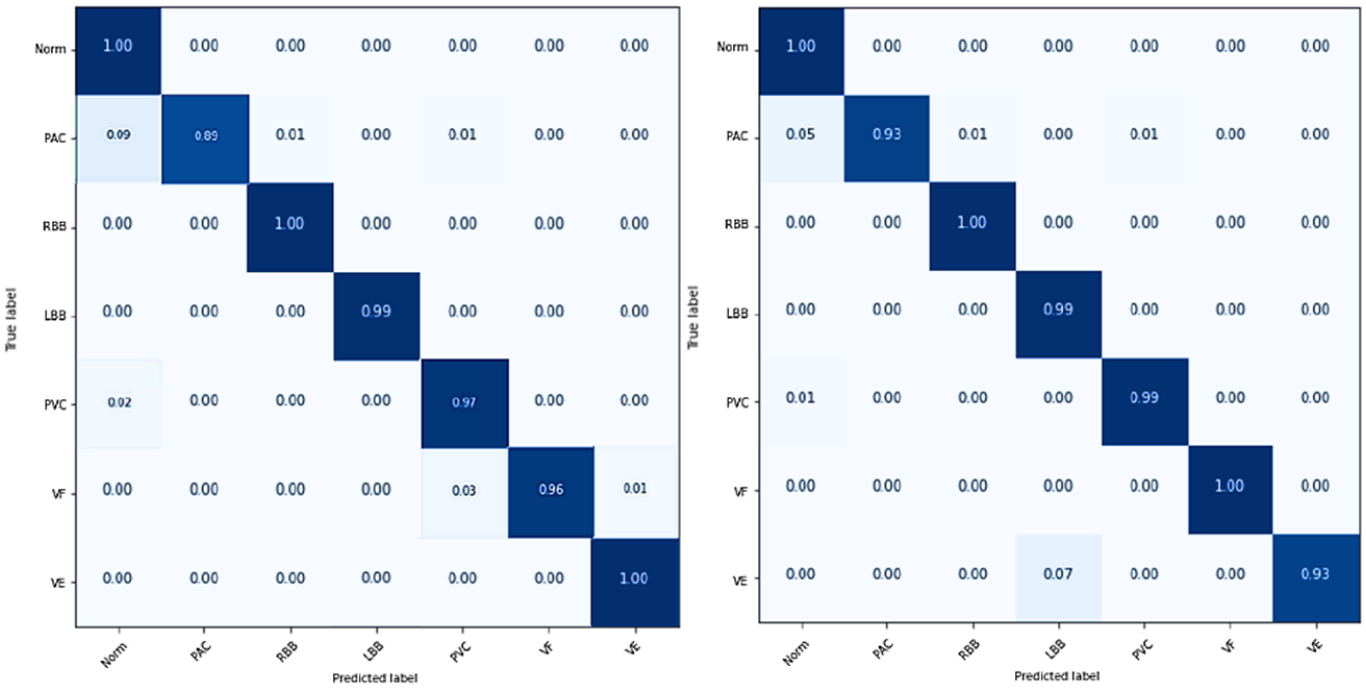


FIGURE 3.19: Cas de 2160 échantillons à gauche avec un réseau LSTM et à droite avec un réseau CNN-LSTM

Sur l'ECG, l'APC est semblable à un battement normal, tel que l'APC et le battement Normale ont une morphologie QRS similaire et peuvent être confondus. Une caractéristique distinctive est que l'intervalle de temps séparant un pic R d'APC et le battement qui le précède est court. De telles relations temporelles sont plus adaptées à l'analyse par des algorithmes multi-temps qui est le cas pour le 2e cas d'étude où 2160 échantillons sont pris, on remarque bien que la performance de classification du type APC a été améliorée de 84% à 89% dans le cas du réseau LSTM, et de 85% à 93% dans le cas du réseau CNN-LSTM. Le réseau LSTM dans les deux cas du nombre d'échantillons a montré une meilleure performance pour la reconnaissance de la classe VE par rapport à celle trouvé par le réseau CNN-LSTM. Sur l'ECG, le battement VE est semblable à un battement normal, la seule différence est la morphologie QRS qui est plus large dans le cas de VE. Une telle caractéristique temporelle est mieux détectée par les réseaux LSTM qu'avec les réseaux hybride CNN-LSTM, et elle est détectée à 100% dans le cas de 2160 échantillons.

Exemple de détection

Les figures (3.20 - 3.24) montrent quelques exemples de détection des arythmies cardiaques sur un signal ECG brut par un modèle entrainé par l'une des méthodes citées ci-dessus.

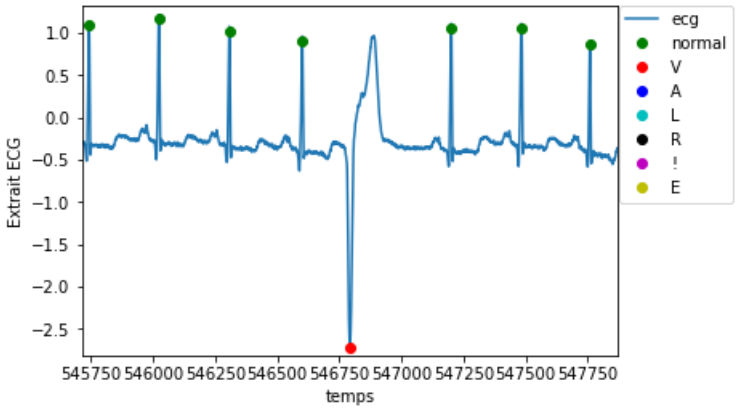


FIGURE 3.20: Exemple de détéction d'un battement PVC

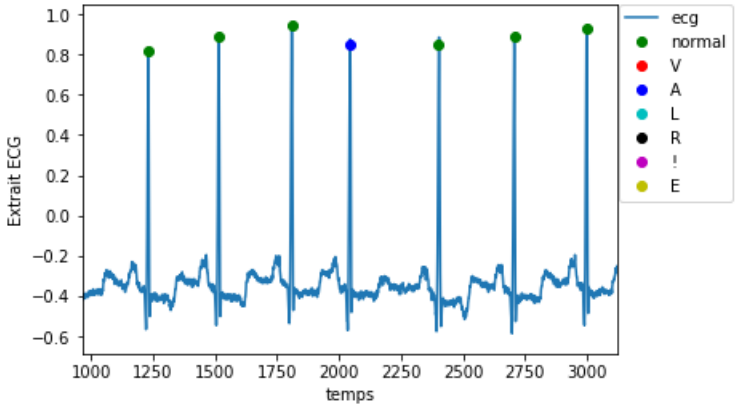


FIGURE 3.21: Exemple de détéction d'un battement APC

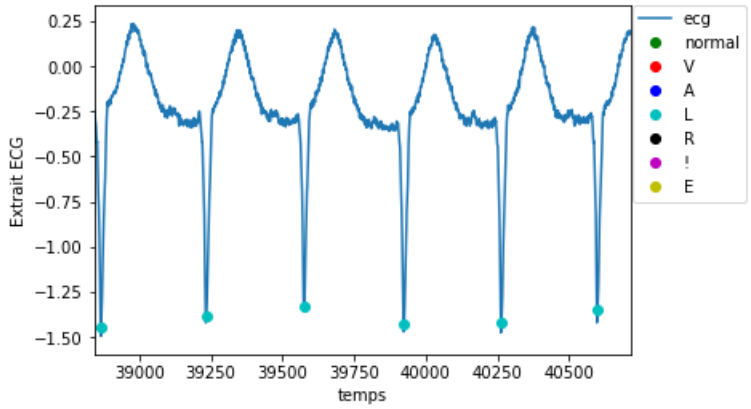


FIGURE 3.22: Exemple de détéction d'un battement LBB

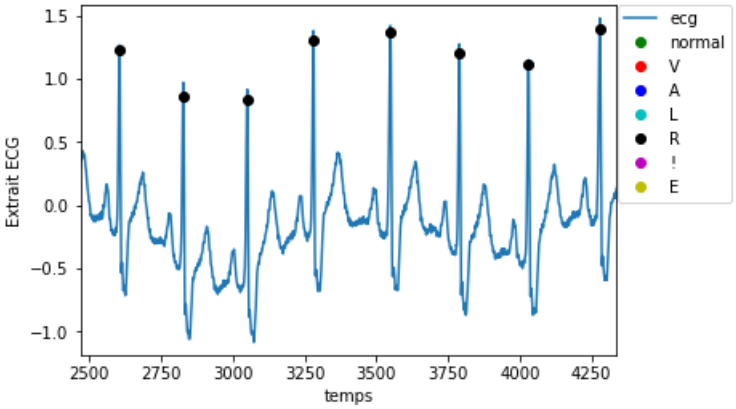


FIGURE 3.23: Exemple de détéction d'un battement RBB

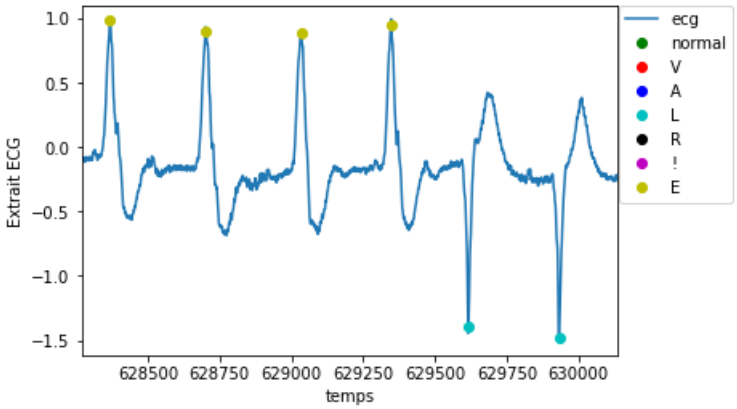


FIGURE 3.24: Exemple de détéction d'un battement d'échappement ventriculaire

3.3 Discussion

De nombreuses études ont été menées pour identifier les signaux ECG sur la base de données d'arythmie MIT-BIH. Le tableau (3.7) donne un résumé de certaines de ces études.

Comparaison					
Etude	Nombre de classes	Les donnés a l'entrée	Classifieur	Largeur des données	Accuracy
Yildirim [48] 2018	5	DWT sub bands	BLSTM	360 échantillons (1 s)	99.39
Yildirim et al. [49] 2018	17	Raw data	CNN	3600 échantillons (10 s)	91.33
Sellami et al. [50] 2019	5	Raw data	CNN	170 échantillons (0.47 s)	99.48
Acharya et al. [51] 2017	5	Raw data	CNN	60 échantillons (1 s)	94.03
Shu Lih Oh [52] 2018	5	Raw data	CNN-LSTM	variable	98.10
Yildirim et al. [53] 2019	5	Raw data	CNN-LSTM	100 échantillons	98.10
notre premier modèle	7	Raw data	CNN	2160 échantillons (6s)	98.92
notre deuxième modèle	7	Raw data	CNN-LSTM	260 échantillons et 2160 échantillons	99.27

TABLE 3.7: Résumé de certaines études d'identification des Signaux ECG sur la base « MIT-BIH ».

Yildirim a proposé l'utilisation d'un réseau LSTM pour ce problème de détection des arythmies, où des sous-bandes DWT de signaux ECG ont été utilisées comme entrées du réseau LSTM. Avec ce travail, Yildirim a pu atteindre 99,39% d'accuracy [48]. Une autre étude d'apprentissage en profondeur présentée par Yildirim est le modèle CNN pour les signaux ECG de longue durée, cette étude est basée sur la reconnaissance automatique de 17 classes des signaux ECG de taille de 3600 (10s) et elle a atteint une performance de 91,33% [49]. Sellami et al. Ont présenté un modèle CNN profond qui contient 9 couches de convolution pour la classification des battements cardiaques, ils ont obtenu un taux de performance de 99,48% [50]. Acharya et al. ont proposé un modèle de classification des arythmies basé sur un modèle CNN unidimensionnel et ils l'ont appliqué sur des signaux ECG pour la reconnaissance de 5 classes, cette méthode a montré une performance de 94,03% [51]. Shu Lih Oh [52] a pu atteindre 98,10% de performance en utilisant un réseau CNN-LSTM avec des segments ECG de longueur variable. La dernière étude présentée par Yildirim [53] où un modèle CAE+LSTM a été utilisé pour la classification des signaux ECG en 5 classes avec des tailles de 260 (2s). Un CAE (auto-encodeur) a été utilisé pour la compression des données et la construction du vecteur caractéristique des données ECG, puis ce vecteur a été passé au réseau LSTM et les couches entièrement connectées pour générer une décision.
Notre étude est basée sur la reconnaissance automatique de 7 classes des signaux ECG

de tailles de 2160 échantillons pour le réseau CNN 1D et 2160 et 260 échantillons pour le réseau CNN-LSTM. Avec le réseau CNN 1D, on a pu atteindre une performance de 98.92%, et pour le réseau CNN-LSTM 99.60% dans le cas de 2160 échantillons et 98.78%. Pour le cas de 260 échantillons, tandis qu'avec le reseau LSTM, 99.35% et 98.72% de performance sont atteintes pour les 2 cas respectivement.

3.4 Conclusion

Notre système automatisé développé, a été inspiré des diverses solutions aux problèmes de recherche propose dans l'état de l'art pour la reconnaissance automatique des signaux ECG, comme la réduction de la taille des données afin de minimiser les besoins de stockage et les coûts de transfert des données et la réduction du temps d'apprentissage du réseau LSTM profond avec des signaux de taille réduite.
Dans cette étude, on a entamé deux approches, la première consiste en un modèle basé sur un réseau CNN 1D pour l'extraction des caractéristiques et pour la reconnaissance des différentes classes. La deuxième approche basée sur une structure hybride composée d'un réseau CNN 1D pour l'extraction des caractéristiques spatiale et la réduction du vecteur caractéristique, et un réseau récurrent LSTM capturer la dynamique temporelle présente à son entrée, Ainsi que pour la reconnaissance des différentes classes. Dans cette approche quatre structures de réseaux différentes ont été utilisées, qui diffèrent par la taille des signaux d'entrée (260 échantillons ou 2160 échantillons), leurs types (taille réduite ou brute) et le type du modèle utilisé (LSTM ou CNN-LSTM)
La performance de classifications des données obtenues est : 98,92% dans la 1ère approche, et dans la 2ème approche, pour le cas de 260 échantillons 98.72% et 98.78% de performance ont été obtenus dans le cas du réseau LSTM et CNN-LSTM respectivement. Et pour le cas de 2160 échantillons 99.35% et 99.60% de performance ont été obtenus dans le cas du réseau LSTM et CNN-LSTM respectivement.

Conclusion générale

Le diagnostic précoce des anomalies cardiaques est important pour réduire les risques causés par les maladies cardiovasculaires. Plusieurs applications dans l'environnement hospitalier comme dans l'environnement externe à l'hôpital rendent le traitement automatique du signal ECG une nécessité. Ce traitement automatique vise à fournir des outils d'aide au diagnostic pour le personnel médical qualifié.
Comme nous l'avons vu en introduction et dans le survol de l'état de l'art, plusieurs techniques de machine learning (ML) classiques ont été utilisées pour traiter et extraire les classes des battements cardiaques.
Dans ce travail on a voulu expérimenter les dernières avancées dans le domaine du ML et les appliquer à la tache de classification des battements du cœur et plus particulièrement les classes d'arythmies. On a utilisé une combinaison de techniques de pointe (CNN et LSTM) pour la détection des arythmies en utilisant des longueurs de signal ECG différentes et de différents types. Ces différences s'expliquent par le fait que certaines anomalies (APC par exemple) sont mieux détectées (classées) si on associe au battement à traiter les battements adjacents et les durées entre battements adjacents. La force des méthodes présentées réside dans le fait que les caractéristiques pertinentes sont extraites automatiquement et sans aucun traitement préalable du signal ECG.
La meilleure solution qui répond aux exigences de notre étude, en matière de performance, précision, classification et temps de calcul a été obtenue en utilisant un réseau hybride CNN-LSTM avec 2160 échantillons à son entré. Ce travail peut encore être amélioré en utilisant des processeurs graphiques (GPU) en entrée et en étudiant en profondeur les architectures utilisées pour identifier les points à améliorer et qui influencent le plus les taches d'extraction des caractéristiques et de classification.

Bibliographie

[1] CRNJIN, Aleksandar. Application and Multidisciplinary Aspects of Wireless Sensor Networks : Concepts, Integration, and Case Studies. Macedonia : Liljana Gavrilovska, 2011. 293p. ISBN 978-1-84996-509-5

[2] CEYLAN, Rahime and ÖZBAY, Yüksel. Comparison of FCM, PCA and WT techniques for classification ECG arrhythmias using artificial neural network. 2007, vol.33, n2, pp. 286–295.[Consulté le 27/06/2020]. Disponible sur <https ://doi.org/10.1016/ j.eswa.2006.05.014>

[3] ANDRÉ-FOUËT, Xavier. Anatomie du coeur [en ligne]. [consulté le 27/06/2020]. Disponible sur : <http ://campus.cerimes.fr/cardiologieetmaladiesvasculaires/enseignement /cardio_1/site/html/1.html>.

[4] Fédération Française de Cardiologie. Activité mécanique du coeur [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://www.fedecardio.org/Je-m-informe/Le-coeur/lacti vite-electrique-du-coeur>

[5] WILLIS HURST, John. Naming of the Waves in the ECG, With a Brief Account of Their Genesis [en ligne]. 1998, vol.98, n18, pp.1937–1942. [consulté le 27/06/2020]. Disponible sur : <https ://doi.org/10.1161/01.CIR.98.18.1937>

[6] GARGIULO, Gaetano. True Unipolar ECG Machine for Wilson Central Terminal Measurements. BioMed Research International [en ligne]. 2015, Vol.2015, P.7. [consulté le 27/06/2020]. Disponible sur : <https ://doi.org/10.1155/2015/586397>.

[7] NORMAN WILSON, Frank and FRANKLIN, Johnston and FRANCIS, Rosenbaum and PAUL, Barker. On Einthoven's triangle, the theory of unipolar electrocardiographic leads, and the interpretation of the precordial electrocardiogram. American Heart Journal [en ligne], 1946, vol. 32, n 3, p.277–310. [consulté le 27/06/2020]. Disponible sur : <https ://doi.org/10.1016/0002-8703(46)90791-0>.

[8] ANDRÉ-FOUËT, Xavier. Les dérivations des membres [en ligne]. [consulté le 27/06/2020]. Disponible sur : < http ://campus.cerimes.fr/cardiologieetmaladiesvasculaires /enseignement/cardio_4/site/html/2. html :˜ :text=L'ECG%20standard%20comporte%20au, pr%C3%A9cordiales)%20%3A%20V1%20%C3%A0%20V6>.

[9] Jmarchn. Les dérivations précordiales [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://commons.wikimedia.org/wiki/File :Precordial$_L eads_2.svg$ >

[10] TABOULET, Pierre. faisceau de HIS [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://www.e-cardiogram.com/faisceau-de-his/>

[11] BURNS, Edward. Contraction auriculaire prématurée [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://litfl.com/premature-atrial-complex-pac/>

[12] GOY, Jean-jacques. Extrasystole ventriculaire [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://book.cardio-fr.com/fr/04-Troubles-du-Rythme/02-Extrasystoles.html>

[13] Anaesthetist.com and a hospital ECG. Le flutter ventriculaire [en ligne]. [consulté le 27/06/2020]. Disponible sur : <http ://www.anaesthetist.com/icu/organs/ heart/ecg/ images/e_ vflut.jpg>

[14] Jonram27. Battement ventriculaire d'échappement [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://commons.wikimedia.org /wiki/File :Ventricular_beat _cardiogram_diagram.jpg>

[15] TABOULET, Pierre. Bloc de branche droit [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://www.e-cardiogram.com/bloc-de-branche-2-droit-complet/>

[16] TABOULET, Pierre. Bloc de branche gauche [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://www.e-cardiogram.com/bloc-de-branche-3-gauche-complet/>

[17] Wikipedia. Facial recognition system [en ligne]. [consulté le 08/07/2020]. Disponible sur : < https ://en.wikipedia.org/wiki/Facial_recognition_system>.

[18] Wikipedia. Object detection [en ligne]. [consulté le 08/07/2020]. Disponible sur : < https ://en.wikipedia.org/wiki/Object_detection >.

[19] Wikipedia. Natural language processing [en ligne]. [consulté le 08/07/2020]. Disponible sur : < https ://en.wikipedia.org/wiki/Natural_language_processing >.

[20] BRESNICK, Jennifer. What Is Deep Learning and How Will It Change Healthcare [en ligne]. [consulté le 08/07/2020]. Disponible sur : < https ://healthitanalytics.com/features /what-is-deep-learning-and-how-will-it-change-healthcare >.

[21] PAL, Manajit. Deep Learning for Self-Driving Cars [en ligne]. [consulté le 08/07/2020]. Disponible sur : < https ://towardsdatascience.com/deep-learning-for-self-driving-cars-7f198ef4cfa2 >.

[22] Wordpress. L'INTELLIGENCE ARTIFICIELLE [en ligne]. [consulté le 03/06/2020]. Disponible sur : < https ://iatpe2015.wordpress.com/le-fonctionnement/le-reseau-de-neurones-artificiel/les-neuro nes-biologiques-et-artificiels/ >.

[23] KUMAR, Niranjan. Deep Learning Best Practices : Activation Functions and Weight Initialization Methods — Part 1 [en ligne]. [consulté le 03/06/2020]. Disponible sur : < https ://medium.com/datadriveninvestor/deep-learning-best-practices-activation-functions-weight-initialization-methods-part-1-c235ff976ed >.

[24] BROWNLEE, Jason. Loss and Loss Functions for Training Deep Learning Neural Networks [en ligne]. [consulté le 07/06/2020]. Disponible sur : < https ://machinelearningmast ery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/>.

[25] NKWAWIR, Beltus. Supervised and Unsupervised Learning For Everyone [en ligne]. [consulté le 08/07/2020]. Disponible sur : < https ://towardsdatascience.com/supervised-and-unsupervised-learning-for-everyone-526f9b746dd5>.

[26] Wikipedia. Perceptron multicouche [en ligne]. [consulté le 03/06/2020]. Disponible sur : < https ://fr.wikipedia.org/wiki/Perceptron$_m$*ulticouche* > .

[27] HERAULT, Romain and CHATELAIN, Clement. Initiez-vous au deep learning [en ligne]. [consulté le 07/06/2020]. Disponible sur : < https ://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning/5814616-explorez-les-reseaux-de-neurones-en-couches >.

[28] LAMBER, R. Le Réseau de Neurones Convolutifs [en ligne]. [consulté le 31/05/2020]. Disponible sur : < http ://penseeartificielle.fr/focus-reseau-neurones-convolutifs/ >.

[29] AGARWAL, Mayank. Back Propagation in Convolutional Neural Networks — Intuition and Code [en ligne]. [consulté le 31/05/2020]. Disponible sur : < https ://becominghu man.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199 >.

[30] LY, Racine. Deep Learning with Convolutional Neural Networks [en ligne]. [consulté le 04/06/2020]. Disponible sur : < https ://www.racinely.com/post/deep-learning-with-convolutional-neural-networks >.

[31] Deep Learning on Medium. Max Pooling vs Convolutions [en ligne]. [consulté le 31/05/2020]. Disponible sur : < https ://mc.ai/deep-learning-cage-match-max-pooling-vs-convolutions/ >.

[32] YAN-TAK NG, Andrew. Deep Learning convolution neural network [en ligne]. [consulté le 31/05/2020]. Disponible sur : < https ://www.coursera.org/specializations/deep-learning ? >.

[33] AMIDI, Afshine and AMIDI, Shervine. Recurrent Neural Networks cheatsheet [en ligne]. [consulté le 04/06/2020]. Disponible sur : < https ://stanford.edu/ shervine/-teachin g/cs-230/cheatsheet-recurrent-neural-networks >.

[34] HOCHREITER, Sepp and SCHMIDHUBER, Jürgen. Long short-term memory [en ligne]. November 15, 1997, vol. 9, n8 [consulté le 04/06/2020]. P.1735-1780. Disponible sur : < https ://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735>.

[35] GUDIKANDULA, Purnasai. Recurrent Neural Networks and LSTM explained [en ligne]. [consulté le 04/06/2020]. Disponible sur : < https ://medium.com/@purnasaigudikandula /recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9 >.

[36] KUMAR, Anuj. Underfitting and Overfitting in Machine Learning [en ligne]. [consulté le 04/06/2020]. Disponible sur : < https ://www.geeksforgeeks.org/underfitting-and-overfitti ng-in-machine-learning />.

[37] NAGPAL, Anuja. L1 and L2 Regularization Methods [en ligne]. [consulté le 28/05/2020]. Disponible sur : < https ://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c >.

[38] BROWNLEE, Jason. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks [en ligne]. [consulté le 28/05/2020]. Disponible sur : < https ://stanford.edu/ she rvine/teaching/cs-230/cheatsheet-recurrent-neural-networks >.

[39] F D , Batch normalization in Neural Networks [en ligne]. [consulté le 28/05/2020]. Disponible sur : < https ://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c >.

[40] RIZWAN, Muhammad. Gradient Descent with Momentum [en ligne]. [consulté le 06/06/2020]. Disponible sur : < https ://engmrk.com/gradient-descent-with-momentum/ >.

[41] GANDHI, Rohith. A Look at Gradient Descent and RMSprop Optimizers [en ligne]. [consulté le 06/06/2020]. Disponible sur : < https ://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b >.

[42] KATHURIA, Ayoosh. Intro to optimization in deep learning : Momentum, RMSProp and Adam [en ligne]. [consulté le 06/06/2020]. Disponible sur : < https ://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/ >.

[43] PhysioNet. The WFDB Software Package [en ligne]. [consulté le 14/02/2020]. Disponible sur : < https ://archive.physionet.org/physiotools/wfdb.shtml >.

[44] PhysioNet. MIT-BIH Arrhythmia Database [en ligne]. [consulté le 14/02/2020]. Disponible sur : < https ://physionet.org/content/mitdb/1.0.0/>.

[45] CHOLLET, François and BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research, 2010, vol.9, p.249–256. [consulté le 27/06/2020]. Disponible sur : <https ://www.researchgate.net/publication/215616968_Understanding_the_difficulty_of_training _deep_feedforward _neural_networks>.

[46] TABOULET, Pierre. Deep learning library for theano and tensorflowTen- sorFlow [en ligne]. [consulté le 27/06/2020]. Disponible sur : <https ://keras.io/>

[47] ABADI, Martín and BARHAM, Paul and CHEN, Jianmin and CHEN, Zhifeng and DAVIS, Andy and DEAN, Jeffrey and DEVIN, Matthieu and GHEMAWAT, Sanjay and IRVING, Geoffrey and ISARD, Michael and KUDLUR, Manjunath and LEVENBERG, Josh and MONGA, Rajat and MOORE, Sherry and G.MURRAY, Derek and STEINER, Benoit and TRICKER, Paul and VASUDEVAN, Vijay and wARDEN, Pete and WICKE, Martin and YU, Yuan and ZHENG, Xiaoqiang. Tensorflow : A system for large-scale machine learning. November, 2016, p.265–283. [consulté le 27/06/2020]. Disponible sur : <https ://www.usenix.org/conference/osdi16/technical-sessions /presentation/abadi>.

[48] YILDIRIM, Özal. A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. Computers in biology and medicine. 2018, vol. 96, p. 189–202. [consulté le 27/06/2020]. Disponible sur : <https ://doi.org/10.1016 /j.compbiomed.2018.03.016>.

[49] YILDIRIMA, Özal and PLAWAKB, Paweł and TAN, Ru-San and ACHARYA,

Rajendra. Arrhythmia detection using deep convolutional neural network with long duration ECG signals. Computers in biology and medicine. 2018, vol.102, p.411–420. [consulté le 27/06/2020]. Disponible sur : <https ://doi.org/10.1016/j.compbiomed.2018.09.009>.

[50] SELLAMI, Ali and HWANG, Heasoo. A robust deep convolutional neural network with batch-weighted loss for heartbeat classification. Expert Systems with Applications. 2019, vol.122, p.75–84. [consulté le 27/06/2020]. Disponible sur : <https ://-doi.org/10.1016/j.eswa.2018.12.037>.

[51] ACHARY, U.Rajendra and OH, Shu Lih and HAGIWARA, Yuki and TAN, Jen Hong and ADAM, Muhammad and GERTYCH, Arkadiusz and SAN TAN, Ru. A deep convolutional neural network model to classify heartbeats. Computers in biology and medicine. 2017, vol. 89, p. 389–396. [consulté le 27/06/2020]. Disponible sur : <https ://-doi.org/10.1016/j.compbiomed.2017.08.022>.

[52] OH, Shu Lih and NG, Eddie YK and SAN TAN, Ru and ACHARYA, U.Rajendra. Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats. Computers in biology and medicine. 2018, vol. 102, p. 278–287. [consulté le 27/06/2020]. Disponible sur : <https ://doi.org/10.1016 /j.compbiomed .2018.06.002>.

[53] YILDIRIM, Ozal and BALOGLU, Ulas Baran and TAN, Ru-San and CIACCIO, Edward J and ACHARYA, U Rajendra. A new approach for arrhythmia classification using deep coded features and LSTM networks. Computer methods and programs in biomedicine. 2019, vol. 176, p. 121–133. [consulté le 27/06/2020]. Disponible sur : <https ://-doi.org/10.1016/j.cmpb.2019.05.004>.