Traitement de séries longitudinales pour l'imagerie médicale

Mohamed Fakhfakh

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Informatique et Télécommunication

Présentée et soutenue par :

M. MOHAMED FAKHFAKH

le vendredi 26 mai 2023

Titre :

Traitement de séries longitudinales pour l'imagerie médicale

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (IRIT)

Directeurs de Thèse :

M. LOTFI CHAARI

M. FAIEZ GARGOURI

Rapporteurs :

MME FLORENCE FORBES, INRIA GRENOBLE - RHONE ALPES

M. WALID BARHOUMI, ECOLE NATIONALE D'INGENIEURS DE CARTHAGE

Membres du jury :

M. JEAN-YVES TOURNERET, TOULOUSE INP, Président

M. YOUSRI KESSENTINI, CENTRE DE RECHERCHE EN NUMERIQUE SFAX, Membre

Je dédie ce mémoire à mes parents, Tijani et Chafia, ainsi qu'à mes frères Nizar, Lassaad et Walid. Leur amour et leur soutien permanent ont été inestimables pour moi durant la réalisation de ce travail. Je tiens également à remercier mes amis pour leur soutien indéfectible.

Remerciements

Les travaux présentés dans ce mémoire ont été menés à l'École Nationale d'Électronique et des Télécommunications de Sfax (ENET'Com – Sfax), à l'Institut National Polytechnique de Toulouse, au laboratoire IRIT – Toulouse (Institut de Recherche en Informatique de Toulouse), ainsi qu'au Laboratoire MIRACL - Sfax (Multimedia, Information systems and Advanced Computing Laboratory). Tout d'abord, je tiens à adresser mes remerciements à tous mes collègues des laboratoires IRIT et MIRACL pour leur accueil chaleureux et leur esprit de collaboration.

Je voudrais exprimer ma profonde reconnaissance à mes directeurs de thèse, Pr. Faiez GARGOURI, qui m'a soutenu pendant plusieurs années, partageant mes difficultés et mes préoccupations, m'encourageant à persévérer sans relâche et de qui j'ai beaucoup appris, ainsi qu'à Dr. Lotfi CHAARI, qui a consacré un temps et une disponibilité d'esprit considérables à la supervision de ma thèse. Je tiens à souligner publiquement à quel point j'ai apprécié sa passion pour la recherche et son encadrement efficace. Ses remarques pertinentes, ses critiques constructives et ses conseils avisés m'ont été d'une grande aide pour l'achèvement de cette thèse. Je le remercie également pour son humour et sa gentillesse. J'ai ainsi pu bénéficier de leur grande acuité scientifique et de leur enthousiasme indéfectible et communicatif. Je leur suis donc reconnaissant d'avoir pu achever ma thèse dans des conditions optimales.

Je remercie sincèrement Monsieur Walid BARHOUMI, professeur à l'université de Carthage et Madame Florence FORBES, directrice de recherche à INRIA Grenoble Rhône-Alpes, qui ont eu la gentillesse d'accepter de rapporter sur ma thèse, ainsi que Monsieur Jean-Yves TOURNERET et Monsieur Yosri Kessentini pour avoir accepté d'être examinateurs.

Je suis reconnaissant envers mon frère, Dr. Nizar FAKHFAKH, et mon meilleur ami, Ismail HADDAR, qui ont manifesté un grand intérêt pour l'ensemble de ma thèse en prenant la peine de lire en détail la thèse et de me faire part de leurs commentaires. Je leur exprime ma gratitude.

Un immense merci à mes parents, Tijani et Chafia Abid, ainsi qu'à mes frères Nizar, Lassaad et Walid, pour leur patience, leur amour et leur soutien continuel, sans lesquels je n'aurais pas pu arriver jusqu'ici.

Enfin, si j'ai omis de citer certaines personnes, qu'elles me pardonnent et trouvent ici toute ma sympathie.

Table des Matières

Table des Figures

Liste des Tableaux

Liste des Abbréviations

ABGSA – Adaptive Best-Mass Gravitational Search Algorithm.

AdaGrad – Gradient Adaptatif.

Adam – Adaptive Moment Estimation.

AOA – Algorithme d'Optimisation Arithmétique.

ChOA – Chimp Optimization Algorithm.

CNN – Convolutional Neural Network.

DA – Dragonfly Algorithm.

DL – Deep Learning.

DNN – Deep Neural Network.

ELU – Exponential linear unit.

FC – Fully Connected.

FL – Federated Learning.

FreLU – Flexible Rectified Linear Units.

GA – Genetic Algorithm.

HMC – Hamiltonian Monte Carlo.

IA – Intelligence Artificielle.

IRM – Imagerie par Résonance Magnétique.

IWT – Improved Whale Trainer.

LReLU – Leaky Rectified Linear Unit.

LSTM – Long Short Term Memory.

MALA – Metropolis-adjusted Langevin algorithm.

MAP – Maximum A Posteriori.

MCMC – Markov Cain Monte Carlo.

MeLU – Mexican Rectified Linear Unit.

MH – Metropolis Hastings.

ML – Machine Learning.

MLP – MultiLayer Perceptron.

ns-HMC – non-smooth Hamiltonian Monte Carlo.

PReLU – Parametric Rectified Linear Unit.

PSO – Particule Swarm Optimisation.

ReLU – Rectified Linear Unit.

RMSprop – Root Mean Square Propagation.

RNN – Recurrent Neural Network.

rw-MH – random-walk Metropolis-Hastings.

SELU – Scaled Exponential Linear Unit.

SGD – Descente de Gradient Stochastique.

SPECT – Single Photon Emission Computed Tomography.

SSA – Algorithme d'essaim de salpes.

SVM – Support-Vector Machine.

TDM – Tomodensitométrie.

TEP – Tomographie par Emission de Positons.

1

Introduction générale

Sommaire

1.1 Contexte du travail

La prise de décisions médicales découle des processus cognitifs complexes nourris par l'expertise des praticiens, et des données recueillies auprès des patients. Elle se base également sur des investigations réalisées dans la mesure où le praticien peut apprécier la pertinence des hypothèses dégagées, afin de les affirmer ou les infirmer. Une des étapes clés dans le processus de prise de décision est celle de la collection des données et l'interprétation y associée. Les nouvelles techniques d'acquisition et de représentation ont contribué à l'enrichissement des données collectées sur un sujet et ce dans le but d'aider le praticien à réaliser le meilleur diagnostic.

L'imagerie médicale [1], [2] est en plein essor grâce au développement de la numérisation et la digitalisation. Elle assure, d'ailleurs, une visualisation réaliste des parties et organes du corps humain. La représentation graphique et visuelle d'organes se fait au travers certains outils de quantification d'ondes électromagnétiques permettant de représenter l'invisible sous différents aspects. L'un des principaux avantages de l'imagerie médicale réside dans la capacité à détecter les maladies et évaluer avec précision.

Les praticiens commencent par recueillir le maximum de renseignements au sujet de l'état du patient, en fonction des symptômes qu'il manifeste. Pour mieux comprendre son état, ils procèdent à des examens cliniques via de diverses techniques d'examen afin d'obtenir une image de l'intérieur du corps du patient. Ces examens permettent aux médecins de visualiser la structure interne des organes susceptibles d'être affectés par des anomalies spécifiques. Cela constitue un atout important pour le diagnostic médical. En analysant différents aspects des organes observés, le médecin peut tester un panel d'hypothèses à partir des données initiales préalablement recueillies. Dans ce cas, le médecin peut être confronté à une ambiguïté consistant à entraver la prise d'une bonne décision. Il peut donc être amené à utiliser différentes techniques de recherche pour obtenir certaines informations afin de dégrossir le problème et mettre le doigt sur le bon diagnostic [3].

Les médecins s'appuient sur leur propre expérience et expertise pour poser le bon diagnostic sur les sujets à traiter, en fonction des imperfections caractérisant les données

obtenues. Ils se basent généralement sur l'ensemble des données collectées dans le passé et s'inspirent des décisions prises auparavant sur le même sujet. Cette procédure est appelée « justification des cas similaires ». Cependant, l'analyse et l'interprétation des données par un opérateur humain peut être chronophage, subjective et sujette à des erreurs ou des biais.

Des outils existants peuvent aider les médecins dans l'interprétation des images médicales et le diagnostic. L'intelligence artificielle [4]–[7] est désormais omniprésente dans la littérature scientifique de l'imagerie médicale. Il s'agit d'un domaine de recherche plus perpétuel et fécond qu'il possédera un bel avenir. Son application, touchant toutes les activités humaines, suppose une amélioration de la qualité des soins. Les progrès de la recherche ont contribué aux progrès de la pratique médicale en fournissant des systèmes d'aide au diagnostic médical [8].

En permettant l'accélération des processus cognitifs, ces systèmes intelligents aident les professionnels de la santé et les médecins à compléter le protocole de diagnostic par l'accès à une représentation visuelle et une interprétation basée sur l'analyse d'images. Ces systèmes améliorent par conséquent le pronostic dédiés aux cas les plus critiques notamment les organes dont les structures sont difficiles à analyser.

En plus de l'analyse spatiale de l'imagerie médicale, le diagnostic peut être réalisé aujourd'hui sur des séries longitudinales d'images. De telles séries peuvent fournir des informations temporelles qui reflètent de manière exhaustive les changements dynamiques d'une pathologie. Jusqu'à présent, seules quelques études ont exploré les informations temporelles issues de diverses modalités comme la tomodensitométrie [9] longitudinale, ou les mesures cliniques en utilisant un modèle basé sur l'intelligence artificielle. La dimension temporelle peut essentiellement intervenir sur deux plans: *i)* l'imagerie dynamique où des coupes d'images sont acquises dans le temps, et *ii)* le suivi longitudinal des patients en moyennant les acquisitions répétées sur la durée du traitement.

L'analyse des séries temporelles et le développement d'approches sophistiquées représentent un axe important de la recherche scientifique durant ces dernières années. Le suivi temporel d'images implique par conséquent une meilleur planification lors de la

prise en charge des patients par l'analyse de l'évolution de leur diagnostic. Cela permet également aux médecins d'envisager des actions plus efficaces, plus précises et moins arbitraires.

1.2 Problématique et objectifs

Dans le système de santé, il y a eu une augmentation spectaculaire de la demande des services d'imagerie médicale qui s'est généralisée ces deux dernières décennies. En plus, différentes utilisations de ces images par les médecins sont possibles telles que les tests de diagnostic et les surveillances du traitement en cours. En outre, les images médicales prennent souvent un temps considérable pour être analysées par les experts et ce compte tenu de l'accroissement des actes médicaux à réaliser.

De toute évidence, l'analyse longitudinale d'images est coûteuse, chronophage et compliquée à mener. De même, de nombreux problèmes sont implicites dans la conception de l'étude longitudinale, en particulier du fait que cela se produise sur des périodes de temps prolongées pour certaines raisons. De ce fait, il est à savoir qu'en pratique, l'analyse longitudinale d'images porte souvent sur un nombre réduit de sujets, ce qui rend la généralisation des résultats très difficile à réaliser.

Qui est plus, la différence significative de l'analyse des données longitudinales par rapport à l'analyse des données transversales réside dans le fait que si les données de différents sujets supposaient être indépendantes, les données du même sujet auraient tendance à être corrélées. Une telle dépendance dans l'ensemble de données invalide une hypothèse essentielle d'indépendance sur laquelle reposent de nombreuses techniques statistiques standards [10]. Le fait de ne pas spécifier correctement la corrélation entre les mesures répétées, peut entraîner des estimations incorrectes de la variabilité d'échantillonnage et peut amener à des inférences scientifiques trompeuses.

Les techniques d'apprentissage automatique, ou machine Learning ML [11]–[13] ont, entres autres, révolutionné le domaine médical. Cela a permis aux médecins d'être plus efficace lors de la réalisation du diagnostic des maladies et l'optimisation des protocoles cliniques. L'analyse d'images médicales aide à détecter certaines maladies à un niveau

précoce et également pour pouvoir interagir rapidement. Les algorithmes de ML incluent l'extraction de caractéristiques appropriées à apprendre pour un problème donné par le biais d'experts. Les techniques d'apprentissage profond résolvent le problème de la sélection des attributs à apprendre et à prédire [14]–[16]. C'est une sous-famille de l'apprentissage automatique à travers laquelle il est possible d'extraire automatiquement les caractéristiques essentielles des données d'entrée. Les avancées considérables des approches basées sur l'apprentissage profond ont montré que les modèles basés sur les réseaux de neurones convolutionnels (Convolutional Neural Network - CNN), sont des outils bien adaptés pour mieux gérer les données d'imagerie médicale.

Ces méthodes sont principalement adaptées à la classification, la segmentation, la détection d'objets, et d'autres applications [17]. Cependant, ces deux familles d'approches ne sont pas adaptées pour gérer correctement les dépendances temporelles longues et complexes [18], [19]. Une famille de méthodes d'apprentissage profond spécialement conçues pour faire face aux corrélations temporelles, sont les réseaux de neurones récurrents, (Recurrent Neural Network - RNN) [20], [21], notamment les réseaux à mémoire longue et courte [22].

Bien que les architectures CNN et RNN offrent une généralité en termes de capacité à appliquer le concept d'apprentissage, les méthodes d'apprentissage profond souffrent encore de quelques problèmes.

Nécessité d'une grande puissance de calcul :
La configuration matérielle requise pour les approches d'apprentissage profond peut également avoir des limites. Des unités de traitement graphique multicœurs à haute performance sont nécessaires pour réaliser de l'apprentissage profond sur un grand volume de données en u temps raisonnable. Cependant, ces unités sont coûteuses dans la proportion où leur coût d'acquisition est assez coûteux et elles consomment de grandes quantités d'énergie.

Lenteur de la convergence :
La convergence des réseaux de neurones est un stade d'entraînement d'un modèle après lequel les changements dans le taux d'apprentissage deviennent plus faibles et les erreurs produites par le modèle lors de l'entraînement sont réduites au mini-

mum. Malgré les avancées scientifiques de ces dernières années, les algorithmes d'apprentissage souffrent encore du problème de convergence qui est considéré comme lent [23].

Problème de la disparition du gradient :
La descente du gradient est l'approche d'optimisation la plus connue et utilisée dans les méthodes d'apprentissage. Lorsque les gradients de la fonction de perte d'un réseau sont très faibles, les paramètres du modèle ne convergent plus. En outre, comme de nombreux gradients sont très petits, ils ne contribuent pas assez à l'apprentissage. Cela constitue l'un des principaux problèmes des méthodes d'apprentissage qui peut affecter l'exactitude des résultats et conduire à de mauvaises performances.

Nous traiterons dans cette thèse les méthodes de classification d'images médicales par analyse de données longitudinales. Pour y parvenir à l'appréhender, la première partie consiste à procéder par des approches d'apprentissage profond pour la classification d'imageries médicales des séries temporelles. La deuxième partie consistera à explorer les modèles bayésiens pour optimiser le temps de convergence.

1.3 Plan du manuscrit

Le présent document s'organise de la façon suivante :

- **Chapitre 2 – L'image médicale : Analyses et modalités d'acquisition :**
  Nous introduisons dans ce chapitre la dualité des méthodes d'analyse transversale et longitudinale. Par la suite, la première partie décrit brièvement l'historique des imageries médicales. La deuxième partie de ce chapitre détaille les différentes modalités d'acquisition d'images médicales.

- **Chapitre 3 – Réseaux de neurones Artificiels : état de l'art:**
  La première partie de ce chapitre présente la structure des techniques d'apprentissage profond, ainsi que les différents frameworks utilisés. La deuxième partie est consacrée à un état de l'art ayant pour essence les méthodes d'optimisation, principalement les algorithmes du premier ordre, d'ordre élevé et sans dérivés. La troisième partie déchiffre les méthodes existantes des fonctions d'activation.

- **Chapitre 4 – Prognostic d'images médicales à l'aide de réseaux de neurones récurrents et convolutifs :**
  Nous présentons une nouvelle approche dédiée à la classification des séries temporelles dont elle est basée sur l'apprentissage profond. L'architecture proposée est basée sur un réseau de neurones convolutifs et récurrents en vue de classer des images X-ray COVID-19. L'objectif est de fournir un pronostic de l'évolution de la pathologie pulmonaire observée et d'améliorer les performances des structures médicales face à la pandémie COVID-19. Nous commençons par présenter le contexte applicatif suivi d'une explication détaillée de l'apprentissage profond. Nous présentons ensuite l'état de l'art des modèles de classification COVID-19 existants. La dernière partie consiste à exposer notre contribution ProgNet, montrer et discuter les résultats expérimentaux afin de clôturer le chapitre par une conclusion.

- **Chapitre 5 – Optimisation bayésienne non lisse pour les réseaux de neurones artificiels :**
  De nombreux travaux sur les méthodes d'optimisation en apprentissage automatique ont été proposés successivement tels que la méthode basée sur les gradients. En utilisant une base d'apprentissage annotée, l'un des principaux défis consistait à optimiser les poids du réseau. Dans un souci d'efficacité, la régularisation est généralement utilisée. Lorsque des méthodes de régulation non lisses sont utilisées notamment pour promouvoir des réseaux parcimonieux, comme la norme ℓ_1, l'optimisation devient, de ce fait, difficile et ce en raison des problèmes de non-différenciabilité du critère cible. Nous proposons un schéma d'optimisation basé sur MCMC formulé dans un cadre bayésien. Le schéma proposé résout le problème d'optimisation parcimonieuse et ce grâce à l'utilisation d'un schéma d'échantillonnage efficace et une dynamique hamiltonienne.

- **Chapitre 6 – Optimisation entièrement automatique avec fonctions d'activation entraînable :**
  Dans la littérature sur les réseaux de neurones profonds, on s'intéresse beaucoup à l'identification et à la définition des fonctions d'activation qui améliorent les performances des réseaux de neurones. Au cours de ces années, l'intérêt scientifique consistant à capter des fonctions d'activation qui peuvent être entraînées tout au long l'apprentissage, a émergé. Une variété de modèles hétérogènes

de fonctions d'activations entraînables a été présentée durant les dernières décennies. Ces méthodes sont conçues pour fournir un apprentissage fiable des paramètres tout en évitant le problème de surajustement. Nous étendons le schéma bayésien proposé dans le chapitre 5 pour estimer à la fois les paramètres de la fonction d'activation entraînable et les poids du modèle. L'objectif principal du schéma bayésien est de minimiser la fonction de coût cible relatif au modèle d'apprentissage grâce à l'ajustement des hyperparamètres.

- **Chapitre 7 – Conclusion :**
  Nous concluons ce manuscrit par un résumé des contributions et des procédures suivies tout au long de la thèse. Nous proposons également des perspectives. Cette section porte sur des aspects scientifiques et pratiques.

2

L'image médicale : Analyses et modalités d'acquisition

Sommaire

2.1 Introduction

L'imagerie médicale se réfère à diverses technologies utilisées pour visualiser les différents organes du corps humain. Elle joue un rôle crucial dans le diagnostic, la surveillance et le traitement de nombreuses maladies. Plusieurs modalités d'images offrent la possibilité de caractériser et de comparer les changements progressifs des pathologies.

Nous analysons durant ce chapitre les méthodes d'analyse d'images, aussi bien transversales que longitudinales, un bref historique ainsi que les diverses méthodes d'acquisition d'images médicales.

2.2 La dualité des méthodes d'analyses

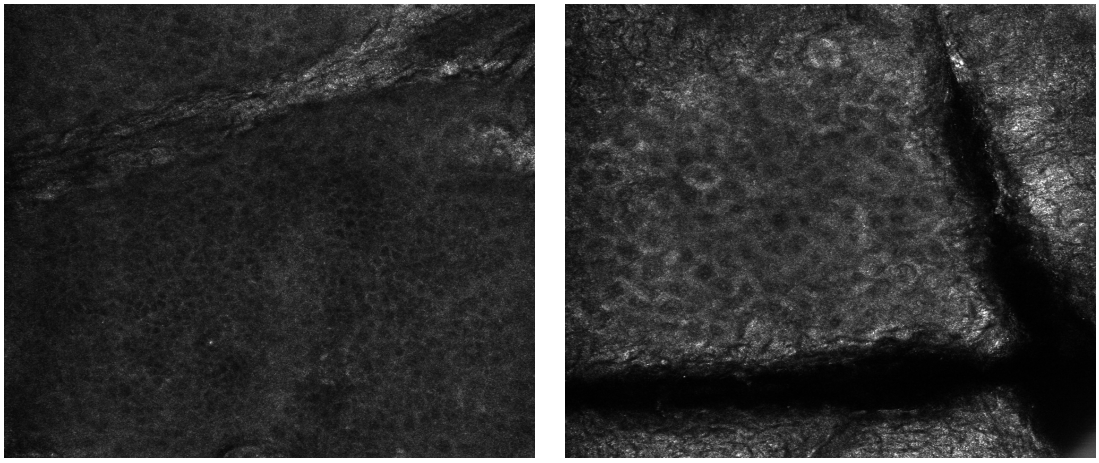
2.2.1 Analyse transversale

Une étude transversale est faite à un moment donné [24]. La collecte d'informations a marqué des moments précis dans la population cible. C'est la collecte des informations uniques qui nous permet de déterminer la fréquence des cas en cours à un moment donné et sur une courte période de temps. Les investigations transversales peuvent être répétées dans le temps, offrant une vision longitudinale du phénomène étudié.

Les chercheurs utilisent l'analyse transversale dans de nombreuses disciplines pour comprendre et analyser les résultats à un instant donné. Ainsi, parmi les domaines d'application on peut citer la détection d'une maladie pour les études médicales, la détection d'objet, etc. L'analyse transversale peut être répétée plusieurs fois dans le temps, fournissant une vue longitudinale de ce phénomène.

Une étude adoptant cette forme d'analyse pourrait être la classification des lentigos, connus encore sous le nom de taches de vieillesse, par microscopie à réflectance confocale (Reflectance confocal microscopy - *RCM*). Les lentigos sont des taches brunes typiquement trouvées sur les zones exposées au soleil telles que le visage, le cou, le dos des mains, les avant-bras et les épaules. Ils peuvent aussi se retrouver sur la peau ou sur les muqueuses. Les lentigos sont souvent liés à une forme particulière de lésion

précancéreuse et cancéreuse appelée lentigo malin. Le principe de base consiste à capturer une image médicale d'un patient et la classer ensuite en deux catégories : saines et lentigos, comme illustré sur la figure 2.1.



(a): sain (b): Lentigo

Figure 2.1: Classification des taches de vieillesse "lentigo" [25].

L'approche transversale présente de nombreux avantages qui constituent la méthode la plus populaire pour les chercheurs. Parmi ces avantages, on peut mentionner : *i)* la rapidité, étant donnée que la collecte des données se fait à un moment donné, l'analyse et l'interprétation de celles-ci sont réalisées rapidement ; *ii)* le coût est réduit par rapport à d'autres types d'approches et *iii*) la flexibilité qui offre la possibilité de mesurer plusieurs facteurs à la fois.

L'approche transversale présente certes des avantages, mais elle comporte également des limites et des inconvénients. Les études transversales ne peuvent pas déterminer l'ordre chronologique des événements car elles n'examinent que les données à un instant t. De plus, il est parfois difficile d'identifier le sens de la causalité en raison du manque d'une suite logique temporelle. En outre, le moment exact où les données sont collectées peut avoir une incidence significative sur les résultats et, par conséquent, les conclusions de l'étude peuvent ne pas être représentatives.

2.2.2 Analyse longitudinale

De nombreuses questions scientifiques peuvent être exprimées comme des changements ou encore d'altérations d'un processus dynamique [26]. La vidéosurveillance est la distinction entre un comportement normal et anormal dans une séquence vidéo. Les études cliniques visent à caractériser les modifications anatomiques ou fonctionnelles causées par la progression de la maladie, les interventions cliniques ou encore les traitements. Les neurosciences appréhendent le neurodéveloppement ou la neurodégénérescence du cerveau ainsi leurs structures connexes [27], [28]. L'imagerie cardiaque recherche des schémas anormaux dans les mouvements du cœur [29], [30]. De ce fait, la complexité des questions scientifiques soulevées est due à l'existence d'un intérêt évolutif qui change d'apparence dans les différentes situations susmentionnées.

Compte tenu de l'interprétation des données, ces problèmes peuvent être résolus par une analyse longitudinale. Ces données existent non seulement dans les milieux biomédicaux et cliniques, mais également dans de nombreux autres domaines de recherche tels que la télédétection [31]–[33]. Ces ensembles de données sont constitués de collections de séries chronologiques qui sont des participations d'un grand nombre de sujets dans une étude observationnelle [34]. Par conséquent, une caractéristique déterminante des données longitudinalement recueillies est que le résultat de chaque séquence est mesuré de manière itérative à plusieurs moments. Comparativement aux études transversales, les études longitudinales peuvent réduire les effets de confusion de la variation morphologique entre les séquences tout en utilisant chaque séquence comme son propre contrôle. Ainsi, l'analyse des études d'imagerie longitudinales est devenue très populaire dans les essais cliniques. Il est essentiel d'acquérir des évaluations morphologiques robustes en intégrant des informations temporelles pour le flux de traitement des séries longitudinales.

Comme exemple d'une série chronologique d'images médicales, considérant le développement d'un patient atteint d'Alzheimer. Cette série est caractérisée essentiellement par la perte de mémoire. Les souvenirs mémorisés à long terme sont conservés jusqu'aux derniers stades de la maladie. Toutefois, leur capacité à apprendre et à mémoriser de nouvelles informations semble être considérablement réduite. À des fins de recherche cliniques, des critères spécifiques ont été développés permettent d'identifier cette maladie probable ou certaine par rapport aux déficits cognitifs, mnésiques, de l'âge d'apparition et du syndrome démentiel [35]. Cependant, la précision de ces critères est

toujours limitée [36].

La neuroimagerie constitue un test adopté dans le but de soutenir le diagnostic clinique, mais sa valeur est encore mal comprise en pratique. La figure 2.2 présente des images prises dans une durée de 32 mois. Une dilatation des ventricules est observée après l'évaluation de la progression de la maladie. Par contre, l'inspection visuelle ne peut fournir qu'une analyse qualitative. L'analyse quantitative de la modification de formes dans le temps a le potentiel de révéler des pathologiques utiles abordant un diagnostic précoce.

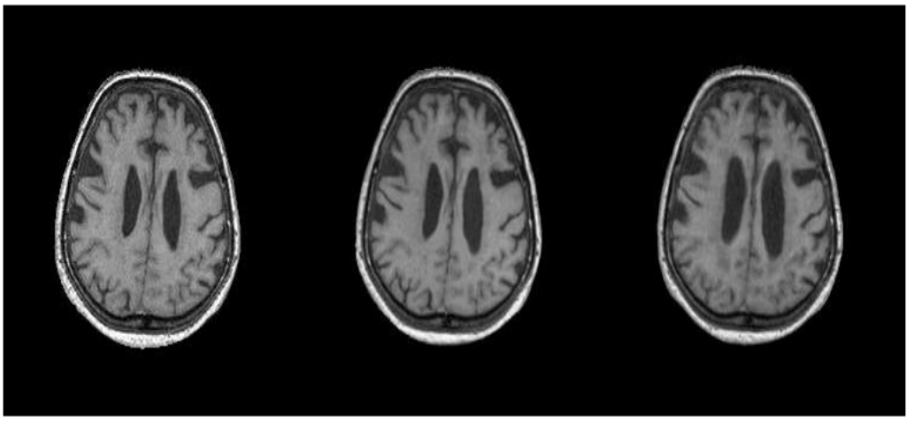


Figure 2.2: Un exemple d'images IRM d'un patient atteint de la maladie d'Alzheimer. L'image de gauche constitue le début de la maladie, l'image au milieu, est effectuée après 12 mois, celle à droite, est établie après 32 mois. L'examen visuel ne peut fournir que qualitativement l'évaluation de la maladie [37].

Comme autre exemple d'application des images temporelles, nous pouvons citer le diagnostic du COVID-19 [38]. La prédiction rapide du développement de cette maladie à un stade précoce est essentielle pour la prise de décision. Cependant, des études se sont concentrées sur les données obtenues à un moment donné de la progression de la maladie. L'analyse longitudinale de la progression du COVID-19 peut aider à mieux comprendre les changements dynamiques des lésions de pneumonie pendant le développement de la maladie et la progression post-traitement, permettant un pronostic plus précis.

2.3 Historique de l'imagerie médicale

La pratique médicale a véritablement été révolutionnée, des trépanations (Fig. 2.3[a]) aux robots chirurgicaux (Fig. 2.3[b]). La médecine traditionnelle comprend la pratique et les méthodes de santé impliquant l'utilisation de divers outils tels que la psychothérapie, les manuels pour le traitement, le diagnostic, fait référence aux connaissances ou encore aux croyances. Aujourd'hui, le processus de traitement s'améliore grâce aux nouvelles technologies d'imagerie modernisées. Les radiologues utilisant des techniques de tomodensitométrie peuvent diagnostiquer de manière quasi non invasive, ce qui permet une meilleure prise en charge des patients.

(a) (b)

Figure 2.3: Évolution de la pratique médicale. (a) la trépanation, (b) la robotique chirurgicale [39].

La première technique radiographie a été introduite par le physicien Wilhelm Konrad Roentgen (1845-1923) au 19e siècle [40]. Il a établi une expérience en 1895 en se basant sur les rayons cathodiques où il décharge le courant de la bobine de Rummukorf à travers un tube vide et en plaçant par la suite sa main devant le tube. Il a même pu obtenir un négatif en plaçant derrière lui une feuille de papier recouverte d'une substance photographique, de sorte qu'il est devenu possible de voir l'intérieur sans ouvrir le corps humain. Depuis lors, de nombreux progrès ont été réalisés à cette technique, donnant naissance à ce que l'on connaît aujourd'hui sous la connotation scientifique de

« radiographie » aux rayons X. Les ordinateurs et l'imagerie numérique ont contribué au développement du scanner à rayons X, appelé aussi tomodensitométrie, en 1972 [41].

Les ultrasons sont employés depuis 1915 pour le SONAR "Sound Navigation Ranging". La cardiologue suédoise Inge Edler (1911-2001) a réalisé la première échographie dans le but de diagnostiquer les sténoses mitrales en 1955. Ensuite, la résonance nucléaire, ou également résonance des noyaux atomiques, a été introduite en 1945. Après cela, il faut attendre jusqu'à l'année 1973 pour la création de la première Imagerie par Résonance Magnétique (IRM). La spectroscopie de résonance magnétique a également évolué à partir de la résonance des noyaux atomiques en 1980. Plus tard, une autre découverte basée sur la radioactivité naturelle a été faite. Les travaux de Irène et Frédéric Joliot-Curie ont favorisé le berceau d'une découverte scientifique éminente celle de la radioactivité artificielle en 1934, entraînant le développement de la médecine nucléaire utilisant la scintigraphie et la tomographie d'émission (TEP) en 1990.

L'imagerie n'est donc pas figée et des techniques que l'on croyait indispensables telles l'urologie intraveineuse, l'opacification gastro-intestinale ou certaines scintigraphies ont pratiquement disparu, remplacées par la tomodensitométrie (TDM), l'IRM, et de nouveaux radiopharmaceutiques en médecine nucléaire. Chaque technologie a ses points forts, et ses points de faiblesses. L'échographie est aujourd'hui incontournable dans le suivi de la grossesse et la cardiologie; la TDM et l'IRM ont chacune leur place en neuroradiologie, et dans toute l'imagerie; la médecine nucléaire est une approche moléculaire de la maladie et de son traitement.

2.4 Modalités d'acquisition d'Images Médicales

"Il existe actuellement plusieurs techniques en développement qui pourraient permettre d'établir un diagnostic médical [40]. Le choix de la méthode d'acquisition appropriée dépendra du problème médical, des performances et des spécifications de la technologie ainsi que de sa disponibilité. Par exemple, l'acquisition toujours plus rapide avec des images obtenues par IRM [42] a stimulé l'analyse sur le cerveau. Il existe un certain nombre d'appareils d'imagerie médicale qui peuvent obtenir des informations plus ou moins précises en fonction des procédures physiques utilisées pour visualiser l'anatomie humaine. Nous passons en revue les différentes techniques existantes et

leurs propriétés.

2.4.1 Les Rayons X

La radiographie a été la première technique d'imagerie médicale répandue, car son utilisation remonte à plus d'un siècle. Les rayons X [43] sont des rayonnements à ondes courtes découverts par Wilhelm Conrad Röntgen (figure 2.4). En décembre 1895, il a présenté, lors d'une communication à la Société Physico Médicale, la photo d'une radiographie de la main humaine comme le montre la figure 2.5. Ensuite, quelques années seulement après sa découverte, les rayons X sont utilisés en médecine [44].

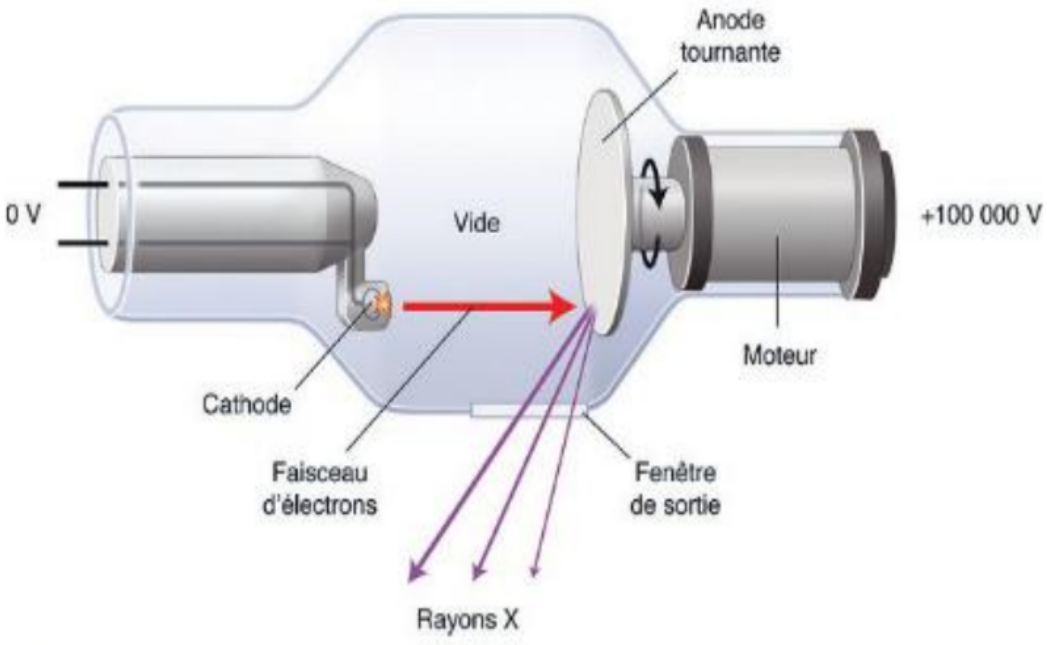


Figure 2.4: Schéma représentatif du tube à rayons X [45].

Les rayons X ont les mêmes propriétés que les ondes lumineuses mais avec plus d'énergie. Une propriété importante de ces rayons est qu'ils sont absorbés par les solides et affaibli par toutes les substances comme les liquides. Ils peuvent également traverser le corps humain selon la densité électronique des organes qu'ils traversent.

La lumière résiduelle, transmise à travers le corps, assombrit un film situé dans la table de radiographie. Les structures aérées comme les poumons apparaissent en noir alors que les structures denses comme l'os apparaissent en blanc. Dans ce cas, les rayons X

Figure 2.5: La première image radiographie. http://www.xray.hmc.psu.edu/rci/.

sont totalement absorbés. A ce jour, il est possible de changer les films radiographiques par des capteurs électroniques. De plus, les différents points du détecteur d'électrons permettent une numérisation et par la suite un traitement informatique des images obtenues par rayons X.

2.4.2 Échographie

Une autre technique est apparue celle de l'échographie qui consiste à visualiser des parties du corps à partir des ultrasons. Les sons exprimés à travers les organes sont interprétés par un terminal où des images sont affichées sur des écrans et des photographies. La tonalité est émise par un cristal vibrant rapidement à une fréquence d'environ 18-20 kHz. Une sonde attachée à la peau émet un son pour recevoir des échos. Presque toutes les radiations ultrasoniques sont absorbées par l'air, les tissus calcifiés et les os. Par conséquent, ce procédé ne permet pas de diagnostiquer des lésions osseuses ou pulmonaires. L'échographie est par conséquent principalement appliquée à l'analyse de certaines parties du corps [46], [47].

Contrairement aux rayons X, cette technique ne présente aucun danger durant la grossesse. Encore, la mesure de la circonférence de tête aide à déterminer l'âge exact du fœtus et à surveiller sa croissance (voir la figure 2.6).

Figure 2.6: Exemple d'une image échographique [48].

2.4.3 Tomodensitométrie

Depuis les années 1970, d'autres outils ont été développés qui utilisent les rayons X. Un scanner à rayons X, également appelé scanner X et un tomodensitomètre [49], pallie ainsi l'inconvénient majeur de la radiographie conventionnelle qui ne permet pas l'imagerie en trois dimensions. Les scanners à rayons X, quant à eux, créent des coupes minces en deux dimensions (2D), généralement de 1 mm d'épaisseur. Pour ce faire, la première étape consiste à produire une première image 2D à partir d'un faisceau de rayons X provenant d'une source parallèle examinant le corps du patient (figure 2.7). Le même processus est répété en tournant le dispositif pour avoir un autre angle de projection. Ensuite, un algorithme de reconstruction est utilisé pour estimer les quantités de coefficients d'atténuation à partir des données mesurées dans la projection. Les quatre générations de scanners avec différentes géométries de source et de détecteur ont permis de réduire progressivement le temps d'acquisition. Désormais, les nouveaux appareils de 5ème génération permettent de capturer des données 2D en très peu de temps, créant ainsi un système d'acquisition d'images en temps réel. De plus, des données 3D ont également été fournies en utilisant des scanners hélicoïdaux [45], [50].

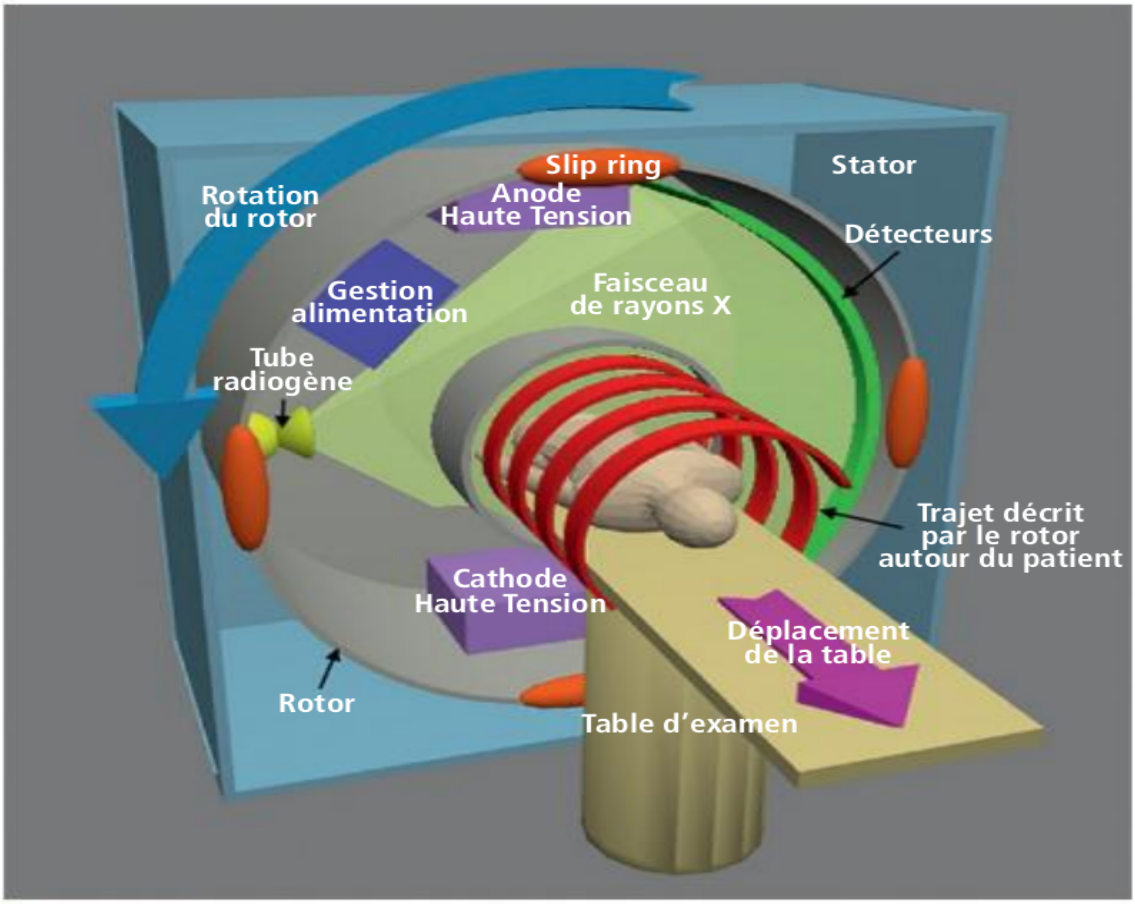


Figure 2.7: Schéma simplifié d'un scanner multidétecteurs de troisième génération [45].

Les scanners à rayons X ont des utilisations dans plusieurs domaines tels que la pathologie oculaire, l'examen thoracique et pulmonaire (figure 2.8). Ils offrent une résolution millimétrique à un prix raisonnable [51].

2.4.4 Images par Résonance Magnétique

Il s'agit d'une technique de diagnostic largement utilisée qui s'appuie sur le principe de résonance magnétique nucléaire [52]. Les principaux composants de IRM sont l'aimant, les antennes et les bobines de gradient comme illustré sur la figure 2.9. Le principe de base repose sur la distribution aléatoire des protons magnétiques. Ce processus comporte trois phases: premièrement, le corps est placé dans un champ magnétique de l'IRM qui force à mettre tous les protons dans le même sens. Les protons sont ensuite stimulés par des ondes radioélectriques qui changent de sens. Finalement, la stimulation s'arrête brusquement et le dispositif collecte les ondes résonnantes via une

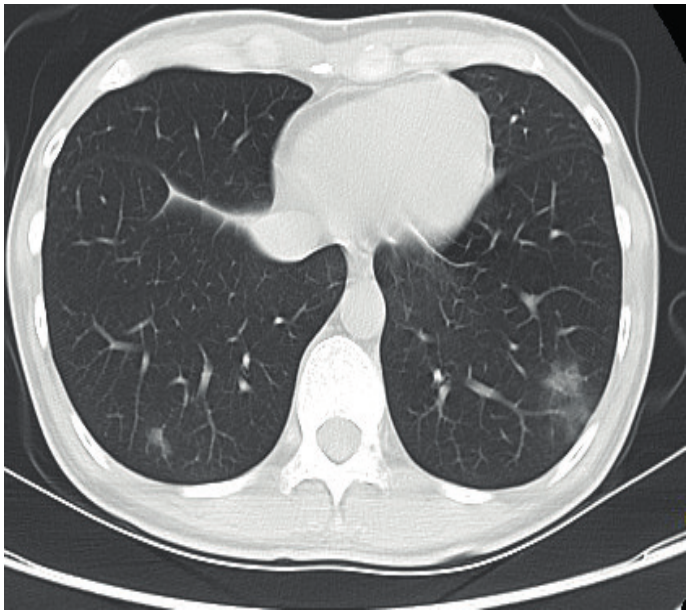


Figure 2.8: Exemple de tomographie thoracique.
https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset

antenne spécifique.

Ces méthodes d'imagerie médicale basées sur la résonance magnétique ont une variété d'applications. Les experts s'entendent pour dire que l'IRM est actuellement la méthode diagnostique la plus efficace. Des images de n'importe quel organe peuvent être obtenues en un temps relativement court. L'IRM a également été étendue pour montrer les artères et les veines en utilisant une technique appelée angiographie par résonance magnétique [53], [54].

Cette technique sert principalement à diagnostiquer les altérations subies par le cerveau et le système nerveux (voir figure 2.10). Les études IRM ont une résolution anatomique similaire aux scanners avec l'avantage d'un contraste élevé. C'est un test très coûteux, mais il a fait ses preuves et fournit des composants de diagnostic plus polyvalents et plus précis.

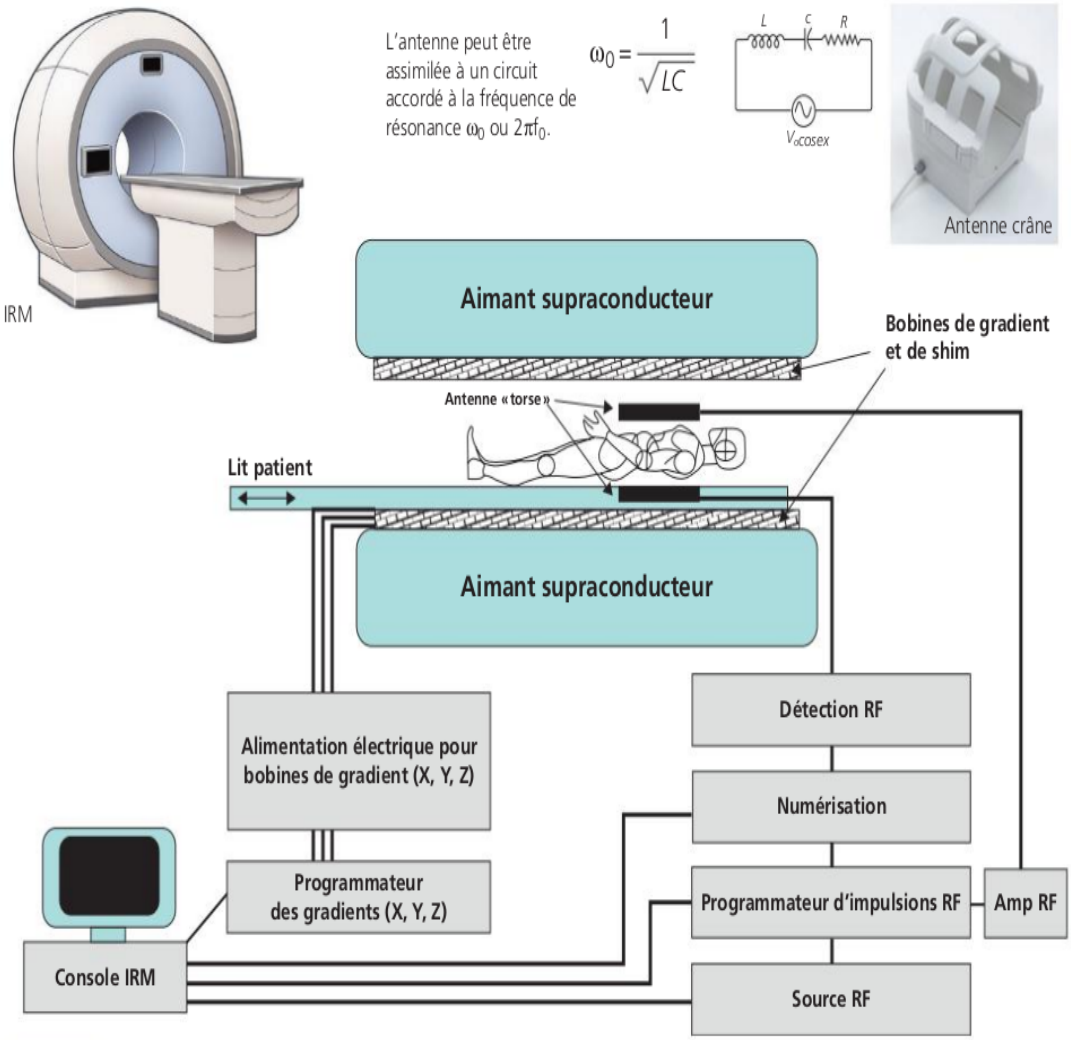


Figure 2.9: Représentation schématique des principaux éléments d'un appareil d'IRM [45].

(a): image Axiale (b): image Sagittal (c): image Coronal

Figure 2.10: Image IRM du cerveau [55].

2.4.5 Imagerie nucléaire

L'imagerie nucléaire repose sur la détection des radiotraceurs injectés dans le corps humain par voie intraveineuse ou orale. Ces traceurs se désintègrent en émettant des photons que l'on détecte. Pour cette raison, des radiomédicaments doivent être utilisés pour ce type d'imagerie. Ces radiomédicaments sont des composés fonctionnels pour lesquels on cherche à déterminer la voie ou le site de liaison, dont l'un des atomes est radioactif. Dans certains cas, c'est l'atome lui-même que l'on cherche à visualiser, qui fait office de traceur. Ainsi, en injectant de l'iode radioactif qui se lie à la thyroïde, on peut imager cet organe [56].

La médecine nucléaire a pour but principal de permettre le diagnostic précoce de certaines conditions médicales, mais elle est également utilisée pour déterminer la morphologie des organes et de suivre par la suite le cheminement des radio-isotopes le long des vaisseaux sanguins et lymphatiques. Nous distinguons :

- La tomographie à émission monophotonique unique (Single Photon Emission Computed Tomography - *SPECT*) applique une gamma-caméra pour enregistrer le rayonnement émis par un radio-isotope. La caméra peut tourner autour du patient dont elle reconstruit une coupe 2D du corps humain avec l'application d'un algorithme d'inversion. Elle a une mauvaise résolution spatiale, mais des informations fonctionnelles plutôt que morphologiques sont visualisées [57].

- La tomographie par émission de positons (*TEP*) est utilisée sous forme de sucre radioactif, fluorodésoxyglucose, pour produire des images de la fonction corporelle et du métabolisme [58]. L'imagerie TEP peut être utilisée pour évaluer la fonction biologique normale ou anormale des organes. Pendant la désintégration, les isotopes radioactifs génèrent des positrons qui, en s'annihilant avec des électrons, émettent deux photons se propageant en sens inverse. Le patient est placé dans un anneau de détecteurs de rayonnement. Lorsque deux détecteurs de chaque côté du patient détectent un photon en même temps, cela signifie qu'un positon a été émis quelque part le long de la ligne reliant les deux détecteurs. La figure 2.11 illustre le schéma de la TEP.

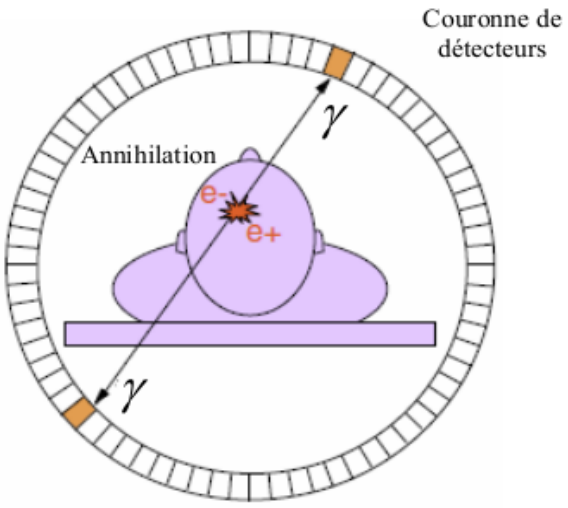


Figure 2.11: Fonctionnement de l'appareil TEP [55].

2.5 Conclusion

Dans ce chapitre, la méthode d'analyse d'image, principalement l'analyse longitudinale, a été présentée. Nous avons expliqué les principes de traitement des séries temporelles. La deuxième partie était consacrée à l'intérêt historique de l'imagerie médicale et à son évolution au fil du temps. Enfin, nous avons détaillé et expliqué le principe de fonctionnement des méthodes existantes pour différentes modalités d'acquisition d'images qui peuvent être utilisées pour le traitement et l'analyse d'images.

Comme nous l'avions mentionné, notre modèle d'analyse des séries longitudinales est fondé sur des méthodes d'apprentissage profond. Le chapitre suivant est associé à un état de l'art portant sur le déchiffrement de ces méthodes. Les contributions seront alors décrites en détail dans les chapitres 4, 5 et 6.

3

Réseaux de neurones Artificiels : État de l'art

Sommaire

3.1 Introduction

L'analyse d'images basée sur l'apprentissage profond a permis l'automatisation de la sphère dédiée à la recherche des anomalies de manière fiable, reproductible et robuste. Cette percée contribue bel et bien à révolutionner le rôle des radiologues modernes ayant tiré désormais des avantages de l'accessibilité des diagnostics assistés moyennant l'ordinateur des images médicales. Les techniques d'apprentissage profond combinent la flexibilité de l'oeil humain d'inspecteurs humains avec la rapidité et la fiabilité des systèmes informatiques. Ce chapitre présente les concepts de base liés à l'apprentissage profond qui sont utilisés pour résoudre les différents problèmes issus de la classification.

Dans un premier paragraphe, nous cherchons à déchiffrer les concepts et les mécanismes des réseaux de neurones et de l'apprentissage profond. Il est fondamentalement recommandé de maîtriser les principes fonctionnels susmentionnés et ce afin de pouvoir s'intéresser dans la deuxième et la troisième section à analyser en vue de mieux exposer les deux étapes cruciales de l'architecture d'apprentissage profond (Deep Learning - DL), à savoir la phase de l'optimisation et celle de la fonction d'activation.

3.2 Apprentissage profond

3.2.1 Définition

L'apprentissage profond (DL) est une approche de l'intelligence artificielle (IA). Plus précisément, c'est une branche de l'apprentissage automatique définie comme des techniques permettant d'améliorer les systèmes informatiques tout en alliant la persistance de l'expérience et la spécificité des données. Elle constitue également la seule approche viable, ayant comme substance la création de systèmes d'IA capables de fonctionner dans des environnements réels équipés inévitablement de certaines situations complexes. La figure 3.1 montre la différence entre IA, ML et DL.

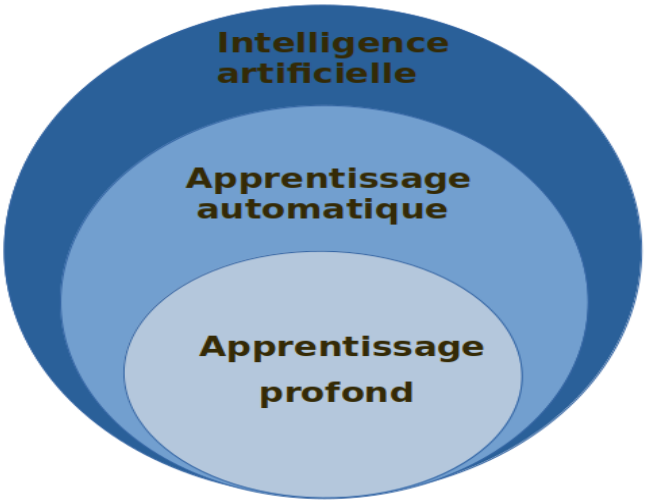


Figure 3.1: Relation entre l'Intelligence Artificielle, l'apprentissage automatique et l'apprentissage profond.

3.2.2 Réseaux de Neurones

3.2.2.1 Neurone biologique

Le neurone biologique est défini comme une cellule vivante qui prend différentes formes, telles que pyramidale, sphérique ou en forme d'étoile. Quelle que soit la forme observée, il est constitué d'une membrane qui isole l'intérieur et l'extérieur du neurone. En outre, le neurone contient des gènes avec un noyau et des protéines dans son cytoplasme. Il possède également des prolongements qui favorisent les connexions synaptiques avec d'autres cellules. Les prolongements qui reçoivent des signaux d'autres cellules sont appelés *dendrites*, tandis que le prolongement unique capable de transmettre et transposer les signaux des neurones vers d'autres cellules est appelé *axone*. Les contacts ou les interactions automatisées entre les deux neurones (entre axones et dendrites) s'établissent via les synapses. Le signal ne se comporte pas linéairement : effet de seuil [59].

Le neurone peut être décomposé en trois unités principales, en vertu de la figure suivante 3.2 :

- Un corps cellulaire, appelé péricaryon.
- Un ensemble de dendrites.
- Un axone.

Lorsque l'excitation du corps cellulaire dépasse un certain seuil, une impulsion nerveuse est émise le long de l'axone. L'excitation est quantifiée par modulation de fréquence. Plus l'excitation du neurone est forte, plus l'impulsion est proche de l'axone [60].

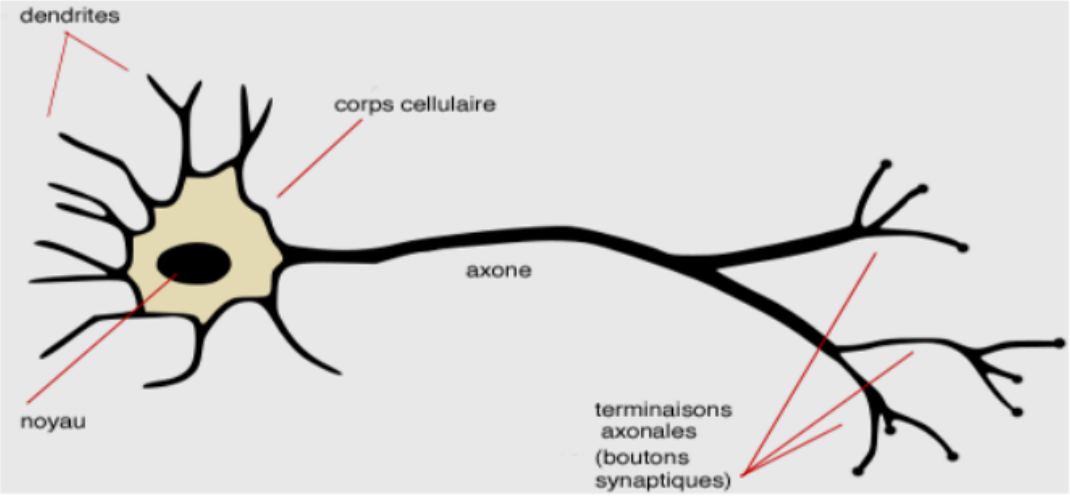


Figure 3.2: Représentation schématique d'un neurone biologique. Source https://commons.wikimedia.org/wiki/File:Neurone_-_commentÃl'.svg

Au sein la zone de contact, s'appelle *une synapse*, les terminaux axonaux peuvent interagir des connexions avec les dendrites d'un autre neurone. D'une manière abstraite, les informations sont transférées d'un neurone à un autre (voir figure 3.2).

3.2.2.2 Perceptron

Les perceptrons, également appelés neurones artificiels ou neurones formels, ont été introduits en 1958 par Franck Rosenblatt. Ils sont conçus pour imiter la fonction des neurones biologiques [61]. Il existe différents niveaux d'abstraction, tout en tenant compte du niveau de précision de modélisation requise.

Dans sa version la plus simple, le perceptron est composé de :

- Des entrées, x sous formes de vecteurs, représentant les dendrites
- Une sortie, y, représentant l'axone
- Deux paramètres, b et W, influençant le déroulement du neurone (voir figure 3.3).

Cette équation décrit comment la sortie $\widehat{y}$ est calculée. Chaque entrée est multipliée par un coefficient W, puis toutes les entrées sont additionnées avec un biais b et le résultat de la somme passe à travers une fonction de transfert f (le plus souvent non linéaire). Enfin, cette fonction produit la sortie souhaitée.

$$\widehat{y} = f((W, x) + b). \tag{3.1}$$

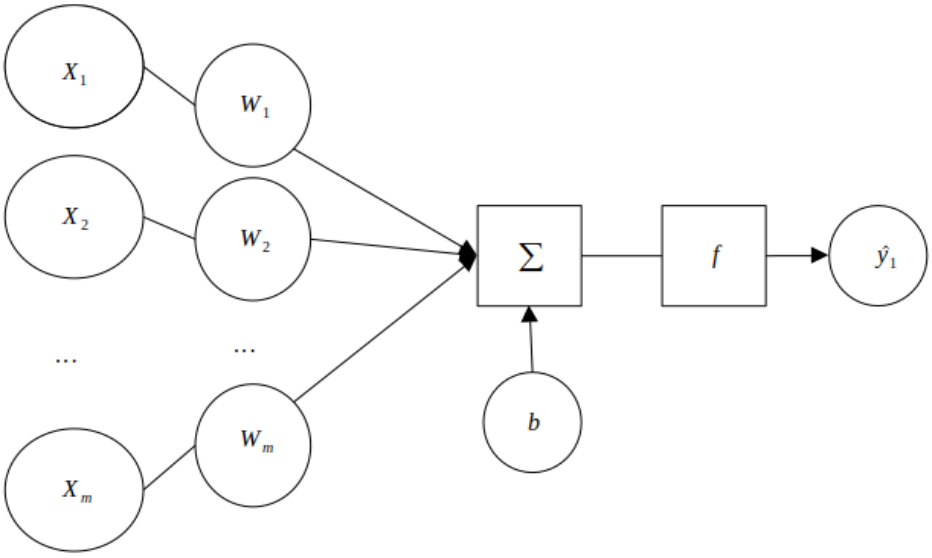


Figure 3.3: Modèle du perceptron.

3.2.2.3 Fonction d'activation

Une fonction d'activation décide de transmettre ou non le signal d'entrée vers la sortie d'un neurone. Elle est généralement équipée d'un ou plusieurs paramètres, qui modifient sa réponse en fonction du signal d'entrée. Au fur et à mesure que de nouvelles informations sont apprises, les poids des connexions (appelés probabilités de connexion) et les seuils des neurones sont ajustés, ce qui renforce ou affaiblit les connexions entre les neurones et permet au cerveau de traiter de nouvelles informations. Le but de l'activation est de transformer le signal d'entrée de manière à ce que le calcul complexe effectué entre les entrées produise une sortie pertinente [62].

La figure 3.4 montre le principe de fonctionnement de la fonction d'activation dans un neurone artificiel. La fonction $Z(x)$ (pré-activation), correspond à la somme des entrées.

Ensuite, le résultat z est analysé par la fonction d'activation $A(z)$ afin de produire y en sortie.

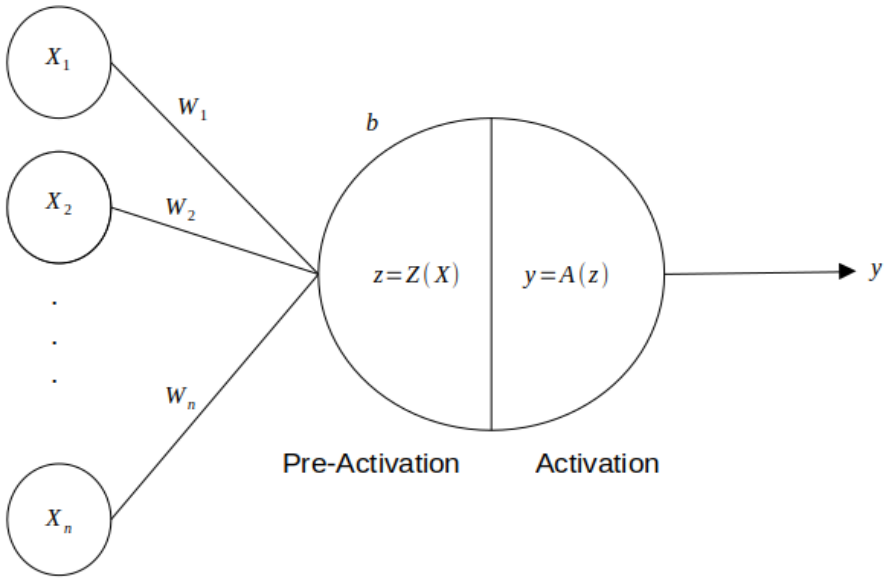


Figure 3.4: Fonction d'activation.

3.2.3 Les réseaux en couches

Les neurones communiquent entre eux moyennant des synapses pour former des réseaux. Les neurones artificiels peuvent être cartographiés de la même façon en ajustant le résultat d'un neurone à l'entrée des autres. Le flux d'informations commence toujours de la première à la dernière couche. Ces architectures peuvent être entraînés en utilisant une technique d'optimisation [63].

Dans la figure 3.5, les sorties des neurones sont agrégées pour former le vecteur de sortie y. Le vecteur d'entrée x est constitué des valeurs des différents attributs qui caractérisent les données à traiter par le réseau de neurones. Le calcul est basé sur la formule suivante :

$$S_j = \sum_i W_{ij} x_j \tag{3.2}$$

$$y_j = f(S_j) \tag{3.3}$$

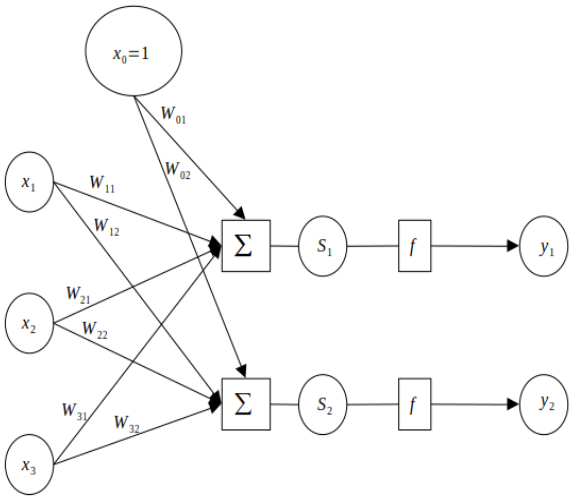


Figure 3.5: Réseau de neurones artificiel en couches.

3.2.3.1 Apprentissage des paramètres

Le processus d'apprentissage dans un réseau de neurones artificiels se fait généralement par la méthode de descente de gradient. Le fonctionnement des méthodes apprentissage se compose de deux phases : propagation avant, et la propagation arrière. Ce processus est décrit ci-dessous.

Propagation avant :
Soit x un vecteur dans $\mathbb{R}^m$, qui présente l'entrée de la première couche du réseau et puis se propage dans les couches cachées (voir figure 3.5[a]) afin de produit une sortie $\widehat{y}$. Le résultat obtenu est évalué par rapport à la vérité terrain y. Une fonction de perte $L(\widehat{y},y)$ calcule la différence entre la prédiction du modèle et la vérité-terrain pour trouver le coût total.

Propagation arrière :
Le but de la propagation en arrière est de réduire l'erreur de prédiction du réseau. Les poids W sont modifiés pour réduire l'erreur calculée par la fonction de perte. Pour ce faire, la première étape consiste à comparer la dérivée de la fonction de perte L en vertu de la sortie $\widehat{y}$ et de rechercher par la suite $\nabla_{\widehat{y}} L(\widehat{y}, y)$ qui constitue le gradient de $L(\widehat{y}, y)$ par rapport à $\widehat{y}$.

Dans le but de minimiser la fonction de perte L, on détermine la direction où L diminuera

rapidement. Étant donné que le gradient pointe dans la direction opposée à la pente, on met à jour $\widehat{y}$ à chaque itération en se déplaçant dans la direction opposée au gradient. Ce principe est appelé la descente de gradient :

$$w^1 = w - \epsilon \nabla_w L(x) \tag{3.4}$$

avec le taux d'apprentissage ϵ (appelé aussi le pas), déterminant la rapidité de l'algorithme de descente de gradient. Cette méthode converge quand le gradient tend vers 0. Par ailleurs, il est très efficace d'utiliser la méthode de rétropropagation pour calculer le gradient.

Rétropropagation :
Il s'agit de faire passer l'information de la dérivée de la fonction de perte, c'est-à-dire la différence entre le résultat réel et le résultat prédit, à la couche d'entrée. L'idée est d'exprimer la dérivée en fonction des données obtenues par les couches suivantes. De cette manière, les valeurs envisagées de la couche de sortie peuvent être rétro-propagées à couche d'entrée [64].

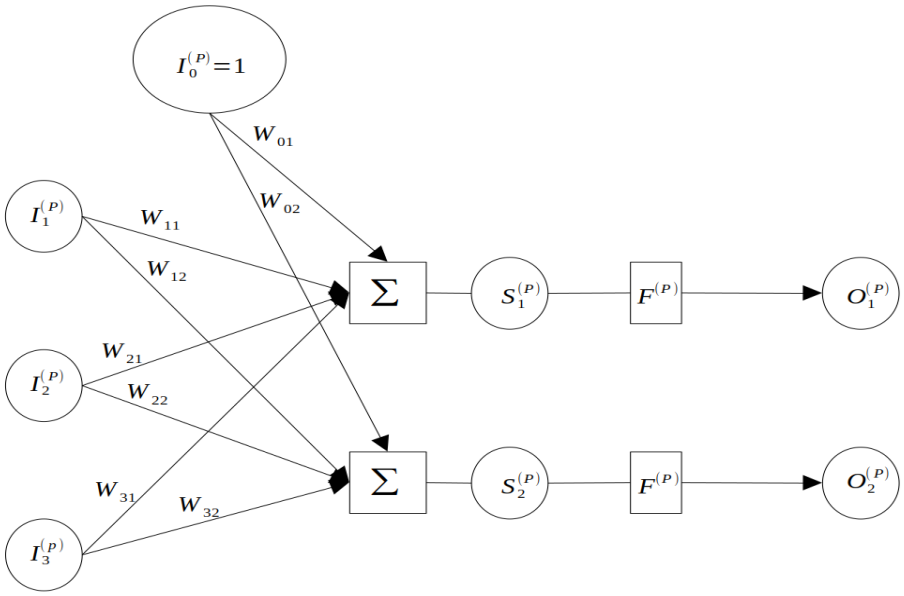


Figure 3.6: Représentation schématique d'une couche isolée.

Le schéma illustré dans la figure 3.6 représente une couche isolée à l'intérieur d'un réseau multicouche :

- (I) représente une entrée quelconque.
- (O) désigne une sortie quelconque.
- (P) prévoit le numéro de la couche.
- (S) est calculée en utilisant l'équation (3.2).

L'algorithme de rétropropagation se déroule de la manière suivante :

- On calcule le gradient de la sortie de la dernière couche :

$$\frac{\partial L}{\partial O_j} \leftarrow \frac{\partial L}{\partial \widehat{y_j}}. \tag{3.5}$$

- Ce processus est répété pour chaque couche. Le gradient d'une couche donnée est utilisé comme sortie de la couche d'avant :

$$\frac{\partial L}{\partial O_j^{P-1}} \leftarrow \frac{\partial L}{\partial I_i^{(P)}}. \tag{3.6}$$

La figure 3.7 montre le principe de fonctionnement de l'algorithme de rétropropagation.

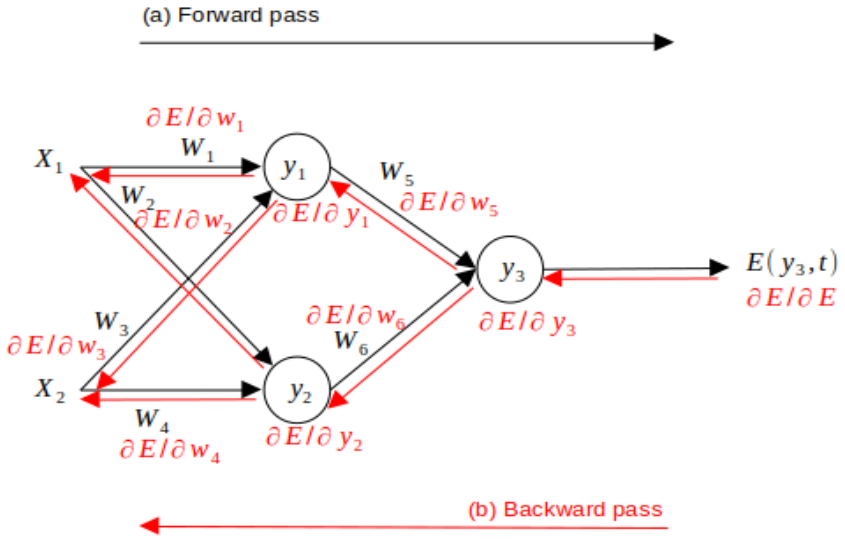


Figure 3.7: La rétropropagation [65].

3.2.4 Le perceptron multi-couche

Les perceptrons multicouches (*MLP*) sont les modèles les plus utilisés dans les applications de réseaux de neurones utilisant des algorithmes d'entraînement de rétropropagation. Définir l'architecture du réseau *MLP* est un point très important. Les sous-connexions peuvent empêcher le réseau de résoudre le problème dû à l'insuffisance des paramètres ajustables, tandis qu'un effet de sur-ajustement peut être créé dû en raison d'un excès de paramètres [66].

Les couches cachées se trouvent entre les couches d'entrée et de sortie, justifiant l'organisation structurale des neurones en couches. En outre, les connexions se font toujours de gauche à droite comme le montre la figure ci-dessous.

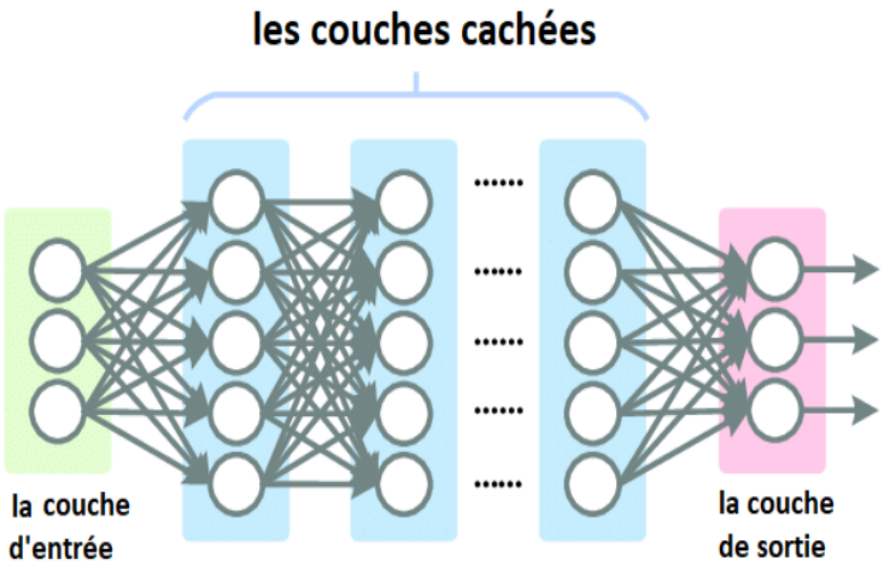


Figure 3.8: Les couches d'un perceptron multi-couches [67].

L'objectif principal consiste donc à obtenir voire à déterminer un meilleur réseau doté suffisamment de paramètres susceptibles de mieux généraliser efficacement la tâche de la solution souhaitée [67].

3.2.5 Réseaux de neurones convolutifs

Un réseau de neurones convolutif (Convolutional Neural Network - *CNN*) [68]–[71] est l'une des méthodes d'apprentissage profond les plus avancées, qui se distingue par l'utilisation d'opérations convolutives substituant l'usage des couches de perceptrons entièrement connectées. Les opérations de convolution sont appliquées séquentiellement à des régions limitées d'une image. La première phase d'un *CNN* est consacrée à l'extraction des caractéristiques les plus importantes de l'image à travers ces opérations. Une image passe par une série de filtres (ou opérations de convolution) pour former de nouvelles images appelées cartes de convolution (ou cartes de descripteurs). Un *CNN* est généralement composé de quatre couches principales comme le montre la figure 3.9 : la couche de convolution (section 3.2.5.1), la couche de pooling (Section 3.2.5.2) suivie par la couche de correction ReLU (Section 3.2.5.3), et la couche entièrement connectée (FC) (section 3.2.5.4).

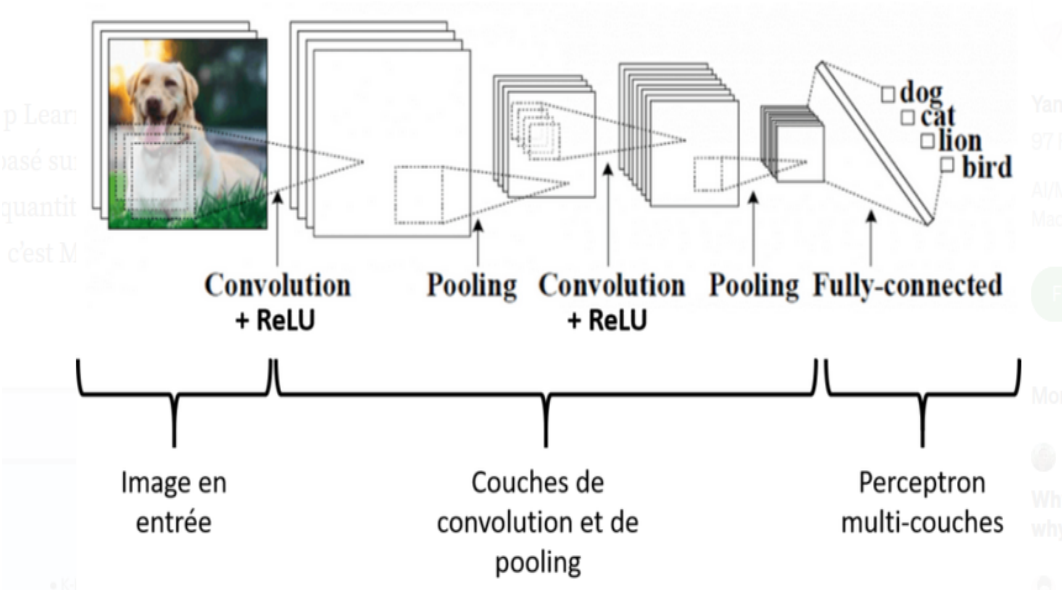


Figure 3.9: Réseau de neurones convolutif www.aspexit.com/.

3.2.5.1 La couche de convolution

Les couches convolutives sont les éléments constitutifs des *CNN*. Principalement, trois paramètres sont utilisés pour déterminer la taille de la couche de convolution [72] :

- Profondeur : corresponds aux noyaux de convolution.

- Le pas: contrôle les champs récepteurs qui se chevauchent. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
- La marge : parfois, il est utile de placer des zéros sur les bornes de l'image d'entrée. Cette marge (ou zero padding) contrôle la dimension spatiale de l'image de prédite (voir figure 3.10).

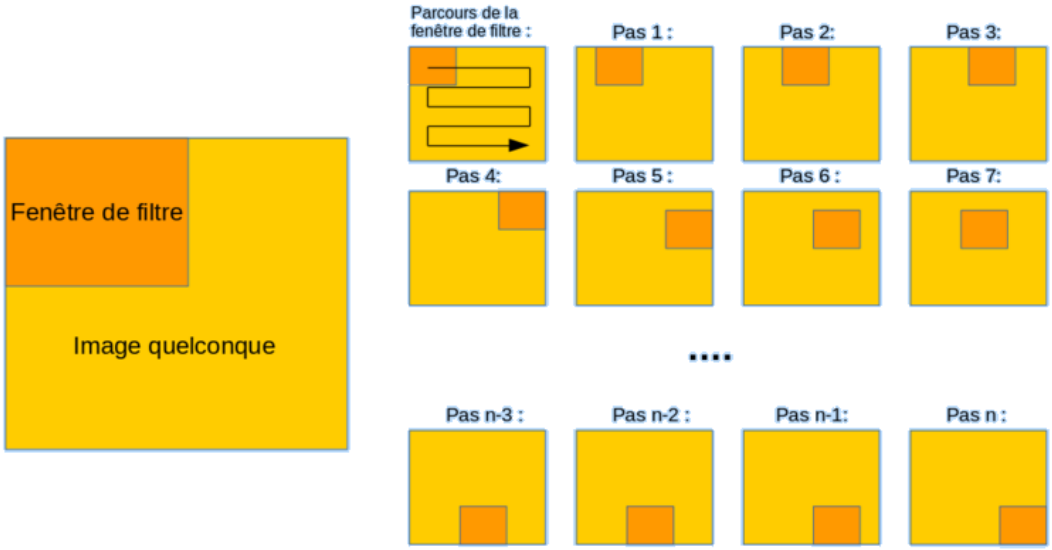


Figure 3.10: Exemple d'un noyau de convolution www.natural-solutions.eu/blog/la-reconnaissance-dimage-avec-les-rseaux-de-neurones-convolutifs.

3.2.5.2 La couche de pooling

C'est une partie clé dans les réseaux convolutifs. En extrayant les valeurs les plus importantes des pixels, ce processus aboutira à réduire l'image à partir de laquelle les caractéristiques pertinentes sont préservées. La méthode la plus couramment utilisée est le Max Pooling (figure 3.11). Cela permet de conserver les valeurs de pixel les plus élevées. Il y a aussi des tuiles qui se déplacent à la surface de l'image (comme des filtres). Ensuite, à chaque position de tuile, l'opération consiste à extraire seulement la valeur la plus élevée en tant qu'objet de conservation. De ce fait, cela créera une nouvelle image contenant uniquement les valeurs d'intérêt.

L'objectif de la couche de pooling est d'améliorer le calcul dans les méthodes d'apprentissage en réduisant la dimension des données. En effet, elle regroupe les informations à une

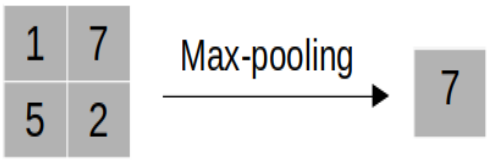


Figure 3.11: Un exemple d'illustration de la procédure de max-pooling.

échelle précise pour produire une représentation plus condensée et plus générale. Cela permet aux méthodes d'apprentissage d'accéder à des représentations abstraites qui sont plus faciles à traiter et à analyser que les données brutes initiales. De plus, cela permet d'accélérer le processus de calcul et de réduire la quantité de données à traiter.

3.2.5.3 La couche de correction (Rectified Linear Unit - *ReLU*)

La caractéristique de cette couche est de remplacer les valeurs négatives par zéro, et d'agir comme une fonction d'activation. Elle est définie par

$$Relu(x) = max(0, x). \tag{3.7}$$

Un schéma de la fonction ReLU est présenté dans la figure 3.12.

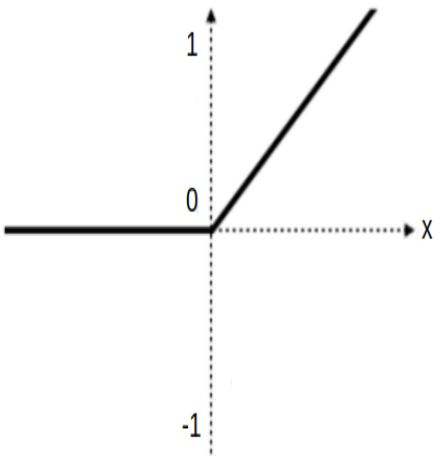


Figure 3.12: Allure de la fonction ReLU.

3.2.5.4 La couche Fully Connected

Cette couche prend un vecteur en entrée et produire un ensemble de valeurs à travers une fonction d'activation. Chaque élément du vecteur de la couche fully connected (FC) prévoit un score à attribuer aux neurones de la couche de perceptron multi-couches (voir figure 3.8) afin d'obtenir le vecteur de sortie contenant les probabilités d'appartenance aux classes données par le modèle.

3.3 Méthodes d'optimisation en apprentissage profond

Les performances des algorithmes d'apprentissage dépendent principalement de la méthode d'optimisation utilisée lors du processus d'apprentissage. L'essence de la plupart des architectures consiste à construire un modèle d'optimisation et d'apprendre les paramètres à partir des données d'apprentissage disponibles. Cependant, il n'existe pas de solution unique pour trouver les paramètres optimaux d'un modèle à complexité raisonnable. De nombreux travaux portent sur l'amélioration des algorithmes d'optimisation en terme précision et de temps de convergence.

En effet, les méthodes d'optimisation peuvent être divisées en trois catégories [73], [74] : *i)* méthodes d'optimisation du premier ordre, *ii)* méthodes d'optimisation d'ordre élevé, et *iii)* méthodes d'optimisation sans dérivées principalement les algorithmes heuristiques et métaheuristiques.

3.3.1 Algorithmes d'optimisation du premier ordre

Les techniques d'optimisation de premier ordre [75] les plus couramment utilisées sont principalement basées sur l'algorithme de descente. Il s'agit de minimiser la fonction objective paramétrée par le poids d'un modèle en mettant à jour les poids dans le sens inverse en utilisant l'algorithme de rétropropagation. La descente de gradient est facile à mettre en œuvre. Si la fonction objective est convexe, la solution est globalement optimale. Lorsque des fonctions non convexes ou non différentiables sont utilisées, ces méthodes peuvent être entachées de la lenteur et révélatrices de problèmes de minima locaux. De même, à mesure que les variables se rapprochent de la solution optimale, la convergence est souvent ralentie ainsi les itérations plus prudentes doivent

être effectuées.

Étant donné que l'algorithme de descente est complexe en termes de calcul à chaque itération pour les données à grandes dimensions, la descente stochastique a été proposée [76], [77]. Ceci est réalisé par la mise à jour du gradient à chaque itération en utilisant un échantillon aléatoire, plutôt de procéder directement à calculer la valeur exacte du gradient. Un gradient stochastique est une estimation sans biais du gradient [76]. Le coût de cet algorithme est indépendant du nombre d'échantillons qu'il peut atteindre une vitesse de convergence sous-linéaire[78]. L'algorithme de descente de gradient stochastique (Stochastic Gradient Descent - *SGD*) accélère considérablement les calculs en réduisant le temps de mise à jour et en supprimant la redondance de calcul pour traiter un grand nombre d'échantillons.

Bien que *SGD* soit populaire et répandu, le processus d'apprentissage peut être lent. Comment ajuster le taux d'apprentissage ? Comment accélérer la convergence ? et comment éviter de rester piégé dans les minima locaux ? Il s'agit de directions de recherche intéressantes. De nombreux travaux ont été présentés pour améliorer *SGD*. À titre d'exemple, l'idée du moment a été suggérée en vue de l'utiliser dans *SGD* [79]. Le concept de quantité de mouvement vient de la mécanique de la physique qui simule l'inertie d'un objet. L'idée d'appliquer un moment dans *SGD* est de conserver l'influence de la direction de la mise à jour de l'itération précédente sur la suivante dans une certaine mesure.

Par ailleurs, un taux d'apprentissage manuellement contrôlé, influence fortement l'efficacité de la méthode *SGD*. Fixer une valeur adéquate du taux d'apprentissage est réputé comme étant un problème difficile [80]–[82]. Plusieurs méthodes adaptatives ont été proposées pour ajuster automatiquement le taux d'apprentissage. Ces méthodes ne nécessitent aucun réglage de paramètres. L'extension la plus directe de *SGD* est *AdaGrad* [80]. Elle ajuste dynamiquement le taux d'apprentissage par rapport au gradient historique de quelques itérations précédentes.

La différence avec la méthode de descente est que la vitesse d'apprentissage n'est plus fixe pendant le déroulement du mécanisme de la mise à jour des paramètres. Elle est calculée à l'aide de tous les gradients historiques accumulés jusqu'à cette itération.

L'un des principaux avantages d'*AdaGrad* réside dans l'élimination du besoin de régler manuellement le taux d'apprentissage.

AdaGrad permet d'ajuster de manière adaptative le taux d'apprentissage. Toutefois, cette méthode présente deux problèmes. Premièrement, l'algorithme nécessite un réglage manuel du taux d'apprentissage global. Deuxièmement, le temps d'apprentissage augmente proportionnellement à l'accumulation des gradients, qui deviennent de plus en plus grands. Par conséquent, le taux d'apprentissage tend vers zéro, ce qui peut entraîner des mises à jour de paramètres inefficaces.

AdaGrad a ensuite été étendu à AdaDelta [77] et *RMSprop* [83] pour résoudre le problème du taux d'apprentissage qui finit par devenir trop petit. L'idée est de suivre le gradient dans une fenêtre en fonction du temps, ce qui permet d'éviter la possibilité d'accumuler tous les gradients passés, en utilisant une moyenne mobile exponentielle pour calculer la quantité de mouvement accumulée du second ordre.

L'estimation adaptative du moment [81] est une autre méthode de descente de gradient avancée et populaire qui introduit une vitesse d'apprentissage adaptatif appliqué à chaque paramètre. Elle combine les méthodes du taux d'apprentissage adaptatif et du moment. *Adam* (Adaptive Moment Estimation) fonctionne très bien et surpasse les autres algorithmes de taux d'apprentissage adaptatif. Une extension de la version d'*Adam*, *AdaMax* [81], est exploitée pour généraliser le processus d'optimisation des poids, permettant une optimisation plus efficace d'un problème particulier.

3.3.2 Algorithmes d'optimisation d'ordre supérieur

L'optimisation d'ordre supérieur [84], [85] est devenue un sujet très populaire et suscite un grand intérêt. Les méthodes d'ordre supérieur sont particulièrement utiles lorsque la fonction objectif est hautement non linéaire et mal conditionnée. Elles fonctionnent efficacement en introduisant des informations de courbure dans le processus d'optimisation, permettant une convergence plus rapide et plus stable vers le minimum global.

L'approche du gradient conjugué est une technique d'optimisation intéressante, ne nécessitant que des informations sur la dérivée première pour un programme quadra-

tique bien défini. Elle peut également être exploitée pour les problèmes non linéaires. [86]–[88]. Généralement, les méthodes de premier ordre sont simples mais lentes à converger, et les méthodes d'ordre supérieur consomment beaucoup de ressources.

Au début des années 1960, la méthode du gradient conjugué a été proposée pour résoudre les systèmes linéaires. C'est une alternative à l'élimination gaussienne [89]. En 1964, la méthode du gradient conjugué a été étendue pour gérer l'optimisation non linéaire des fonctions générales [90]. De nombreux algorithmes différents basés sur cette méthode ont été présentés au fil des années, dont certains sont largement utilisés dans la pratique.

La méthode de Newton [91] a également été développée pour résoudre le problème de lenteur de la descente de gradient. Elle consiste à approximer la fonction objective avec une fonction quadratique en utilisant la dérivée du premier et de seconde ordre. On répète ces étapes jusqu'à la convergence. Cette méthode peut être appelée méthode d'amortissement de Newton [92]. Géométriquement, le modèle Newton consiste à ajuster la surface locale de la position courante avec une surface quadratique, alors que la descente de gradient consiste à ajuster la surface locale courante avec un plan [93].

La méthode de Newton nécessite le calcul de la matrice hessienne inverse de la fonction objectif à chaque étape, ce qui rend le stockage et le calcul très coûteux. Pour pallier ces problèmes, une méthode approchée appelée méthode de quasi-Newton a été proposée. L'idée principale de cette méthode est d'utiliser une matrice définie positive pour approximer la matrice hessienne inverse, ce qui peut simplifier la complexité de l'opération. La méthode de quasi-Newton est l'une des méthodes les plus efficaces pour résoudre les problèmes d'optimisation non linéaire [94]. De plus, cette méthode est parfois plus efficace que la méthode de Newton car elle ne nécessite pas directement le calcul du gradient d'ordre supérieur.

De nombreux modèles d'apprentissage nécessitent l'utilisation des algorithmes d'approximation stochastiques basés sur un sous-ensemble d'apprentissage relativement petit [95], [96] à chaque étape de mise à jour. Les algorithmes stochastiques obtiennent les meilleures performances de généralisation dans les systèmes d'apprentissage à grande échelle [97]. Les méthodes quasi-Newton approximent la matrice hessienne en

utilisant uniquement des informations de gradient de premier ordre. Il est logique de combiner des méthodes quasi-Newton avec des méthodes stochastiques pour pouvoir traiter des problèmes à grande échelle [98].
D'autres méthodes basées sur Newton ont également été proposées comme la méthode sans hessian [99], la méthode sans hessian sous-échantillonnée [100], la méthode du gradient naturel [101], [102], et la méthode de région de confiance [103]–[105], etc.

Cependant, en raison du coût élevé par itération pour stocker la matrice hessienne inverse, cette famille de méthodes n'a pas été largement adoptée en apprentissage profond.

3.3.3 Algorithmes d'optimisation sans dérivés

Compte tenu de certains problèmes d'optimisation dans des applications pratiques, la dérivée de la fonction objective peut ne pas exister ou le cas échéant être difficile à calculer [106]. Dans ce cas, la solution consistant à faire apparaître le point optimal est appelée optimisation sans dérivée en tant que sphère de l'optimisation mathématique [107]–[109].

Il existe plusieurs types d'idées d'optimisation sans dérivée. L'une des idées consiste à utiliser des méthodes heuristiques. Elles se caractérisent par des règles empiriques, choisissant des méthodes qui fonctionnent d'une manière appropriée plutôt que de proposer systématiquement des solutions. Pour un problème d'optimisation donné, les heuristiques sont essentiellement des algorithmes de recherche qui prévoient une solution parmi toutes les solutions possibles. Ces algorithmes peuvent consommer des ressources temporelles limitées et produire des solutions de haute qualité, mais il n'y a aucune garantie théorique. En pratique, il n'est pas toujours nécessaire de trouver la solution optimale, car la recherche heuristique propose souvent une solution acceptable. Il existe de nombreux types de méthodes d'optimisation heuristiques, y compris les opérations classiques de recuit simulé, les algorithmes génétiques, les algorithmes de colonies de fourmis et l'optimisation des essaims de particules [110], [111].

Les métaheuristiques [112]–[114] représentent des classes d'algorithmes heuristiques qui peuvent être appliquées à tout problème d'optimisation. Ce sont généralement des algorithmes stochastiques itératifs qui visent des optima globaux, c'est-à-dire des extrema globaux de la fonction, en échantillonnant la fonction objective. Ils se comportent

comme des algorithmes de recherche (semblables aux algorithmes d'approximation) qui apprennent les propriétés du problème et tentent à se rapprocher de la meilleure solution. Certaines métaheuristiques nécessitent également des relations de voisinage ou des opérateurs de mutation (algorithmes évolutifs), mais ces composants sont considérés comme des "boîtes noires".

Il existe diverses méta-heuristiques non seulement pour la simple recherche locale mais également pour les algorithmes de recherche globale complexes. Cependant, ces approches exploitent un haut degré d'abstraction dont elles sont adaptables aux différents problèmes. En effet, la littérature récente contient de nombreuses études abordant de telles techniques aptes à entraîner des méthodes d'apprentissage profondes telles que l'optimisation de l'essaim de particules (Particle Swarm Optimization - PSO) [115], Algorithme génétique (Genetic Algorithm - GA) [116], improved wall trainer (IWT) [117], Algorithme d'optimisation Chimp (Chimp Optimization Algorithm - ChOA) [118], algorithme d'essaim de salpes (Salp Swarm Algorithm - SSA) [119], algorithme de recherche gravitationnelle adaptative de meilleure masse (Adaptive Best Mass Gravitational Search Algorithm - ABGSA) [120], algorithme de libellule (Dragonfly Algorithm - DA) [121], algorithme d'optimisation arithmétique (Arithmetic Optimization Algorithm - AOA) [122], etc. Cependant, la stratégie d'implémentation des algorithmes heuristiques et métaheuristiques dans des problèmes d'apprentissage profond à grande échelle est encore rarement étudiée [106].

En effet, les méthodes métaheuristiques peuvent fournir des solutions satisfaisantes dans un délai raisonnable, mais leurs principales limites se présument dans la difficulté à traiter des problèmes de grandes dimensions conjuguées par la présence des contraintes de convergence et de stabilité. Dans certains cas, l'optimisation fédérée est également prise en considération dans la littérature récente où des techniques d'apprentissage fédéré (Federated Learning - FL) ont été développées [123], [124]. Chaque nœud contient un élément de données où le but est de construire commun et unique qui correspond à l'ensemble de la distribution. La descente de gradient avec de petits lots est couramment envisagée pour l'optimisation du poids. FL est particulièrement utile lorsque certaines des données peuvent être conservées localement sur des nœuds collaborateurs. Certaines variantes, telles que le consensus flou, ont également été proposées [125].

Une autre idée d'optimisation sans dérivée consiste à ajuster une fonction appropriée basée sur des échantillons de la fonction objective. Ce type de méthode ajoute généralement des contraintes à l'espace de recherche pour dériver l'échantillon. La descente par coordonnée est un algorithme classique sans dérivée [126] qui peut être facilement étendu et appliqué aux algorithmes d'optimisation pour l'apprentissage automatique. Son idée gravite autour une recherche unidimensionnelle peut être effectuée le long de chaque direction d'axe tout en stimulant des valeurs mises à jour dédiées à chaque dimension. Cette technique est utile pour certains problèmes où la fonction de coût ne peut pas être différentiable.

La principale différence entre la descente par coordonnée et la descente de gradient est que dans la méthode de descente de gradient, chaque direction de mise à jour est déterminée par le gradient de la position actuelle et peut ne pas être parallèle aux axes de coordonnées. En descente de coordonnées, la direction d'optimisation est fixée du début à la fin. Il n'est pas nécessaire d'ajuster le gradient de la fonction de coût. À chaque itération, la mise à jour n'est exécutée que selon la direction d'un axe, de sorte que le calcul de la descente par coordonnée est trivial même pour les problèmes complexes [127]. La méthode de descente de coordonnées présente toujours des limitations lorsqu'elle est appliquée à un objectif non lisse qui peut se retrouver à un point non stationnaire.

3.3.4 Analyse et comparaison

Plusieurs approches ont été explorées pour résoudre des problèmes d'optimisation de l'apprentissage profond. Nous résumons certaines des méthodes d'optimisation mentionnées en termes d'années de publications, d'objectifs, d'avantages et d'inconvénients dans le tableau 3.1.

Les méthodes d'optimisation pour l'apprentissage profond sont encore confrontées à plusieurs défis et problèmes non résolus. Il existe deux principaux défis liés aux données et aux modèles. Le premier est l'insuffisance de données dans l'entraînement et la deuxième est optimisation non convexe dans un réseau de neurones profond :

3.3.4.1 Données insuffisantes :

En général, l'entraînement en apprentissage profond nécessite de grands volumes de données pour obtenir une bonne efficacité. Cependant, l'entraînement d'un modèle

dépourvut de données suffisantes ayant comme objet l'estimation des paramètres, peut engendrer une variance élevée, [128] et un surajustement [129]. Les réseaux de neurones sont investis de plusieurs techniques qui peuvent être utilisées en vue de réduire l'envergure de la variance, notamment les techniques de régularisation, le dropout [130], les techniques d'apprentissage par transfert [131], et les méthodes de méta-apprentissage [132], [133].

L'apprentissage par transfert et le méta-apprentissage sont susceptibles de surmonter les problèmes causés par des données d'entraînement insuffisants pour de nouvelles sources de données. Toutefois, ces méthodes introduisent généralement un grand nombre de paramètres ou encore des mécanismes complexes d'ajustement des paramètres nécessitant, de ce fait, l'intervention de certaines améliorations réputées fondamentales et ce afin de résoudre un problème donné. Par conséquent, l'entraînement avec des données insuffisantes reste un défi.

3.3.4.2 Optimisation non convexe :

L'optimisation convexe a de bonnes propriétés où de nombreux outils ont été publiés pour solutionner les problèmes d'optimisation. Cependant, certains problèmes d'apprentissage sont qualifiés comme étant des problèmes d'optimisation non convexes. À titre illustratif, la quasi-totalité des problèmes d'optimisation sont réputés non convexes.

Contrairement à l'optimisation convexe, les problèmes non convexes peuvent avoir un nombre infini de solutions optimales dans la région réalisable. La complexité des méthodes visant à trouver l'optimum global est NP-difficile. Il existe deux types de solutions d'optimisation non convexes : le premier type de méthodes consiste à transformer l'optimisation non convexe en un problème d'optimisation convexe [134]–[136] et le deuxième consiste à utiliser des méthodes d'optimisation pour régler directement des fonctions non convexes [137]–[140].

Table 3.1: Résumé des méthodes d'optimisation.

Catégories	Méthode	Années	Objectif	Advantages	Inconvénients
	SGD [76], [77]	2013	Les paramètres de mise à jour sont calculés à l'aide d'un mini-lot échantillonné au hasard. La méthode converge à un taux sous-linéaire.	Le temps de calcul pour chaque mise à jour ne dépend pas du nombre total d'échantillons d'apprentissage.	Il est difficile de choisir un taux d'apprentissage approprié, et utiliser le même taux d'apprentissage pour tous les paramètres n'est pas approprié. La solution peut être piégée au point saddle dans certains cas.
Premier ordre	Adam [81]	2014	Il est difficile de choisir un taux d'apprentissage approprié, et l'utilisation du même taux d'apprentissage pour tous les paramètres n'est pas approprié.	Le processus de descente de gradient est relativement stable. La méthode convient à la plupart des problèmes d'optimisation non convexes avec de grands ensembles de données.	La méthode peut ne pas converger dans certains cas.

	Adadelta [141]	2012	Cette méthode modifie le mode d'accumulation du gradient total en moyenne mobile exponentielle.	Adadelta améliore le problème d'apprentissage inefficace dans la phase tardive d'AdaGrad. Elle consiste à optimiser les problèmes non stationnaires et non convexes.	Dans la phase d'entraînement tardif, le processus de mise à jour peut être répété autour du minimum local.
	Adamax [81]	2017	Adamax est une généralisation d'Adam qui est basée sur l'estimation adaptative du moment d'ordre inférieur.	L'utilisation de la norme d'ordre infini rend l'algorithme plus stable et moins sensible aux valeurs aberrantes dans les gradients.	Le paramètre de pénalité est lié à la fois aux résidus originaux et duaux dont la valeur est difficile à déterminer.
	Méthode de Newton [142]	2003	Cette méthode calcule l'inverse de la matrice hessienne pour obtenir une convergence plus rapide que les approches du premier ordre.	La méthode de Newton a une convergence plus rapide que la méthode du gradient du premier ordre. De plus, elle a une convergence quadratique sous certaines conditions.	Elle nécessite un temps de calcul long et un espace de stockage important à chaque itération.

	Méthode quasi-Newton [90]	2006	Utilise une matrice approchée pour approximer la matrice hessienne ou sa matrice inverse.	Cette méthode n'a pas besoin de calculer la matrice inverse de la matrice hessienne, ce qui réduit le temps nécessaire de traitement.	Elle nécessite un espace de stockage important, ce qui n'est pas adapté à la gestion de l'optimisation à grande échelle.
Order supérieur	Méthode Hessian-Free (HF) [99]	2010	La méthode HF effectue une sous-optimisation en utilisant le gradient conjugué, ce qui évite le calcul coûteux de la matrice Hessienne inverse.	La méthode HF peut utiliser les informations de gradient d'ordre supérieur, mais n'a pas besoin de calculer directement les matrices hessiennes. La méthode Hessian-Free convient à l'optimisation de grandes dimensions.	Le coût du calcul du produit matrice-vecteur en méthode HF augmente linéairement avec les données d'apprentissage. Cela ne fonctionne pas bien pour les problèmes à grande échelle.

	Méthode Stochastique Quasi-Newton [143]	2018	La méthode stochastique quasi-Newton utilise des techniques d'optimisation stochastique : ex. online-LBFGS [98] et SQN [95].	Elle peut traiter des problèmes d'apprentissage automatique à grande échelle.	Par rapport à la méthode du gradient stochastique, le calcul de la méthode quasi-Newton stochastique est plus complexe.
	IWT [117]	2019	Utilisation d'une forme en spirale appropriée inspirée d'une baleine à bosse pour améliorer la phase d'exploitation de l'algorithme standard d'optimisation des baleines.	Cette méthode permet de renforcer la capacité de recherche globale et peut être utilisée pour surmonter des cas complexes d'optimisation sous contraintes.	Convergence lente et chute facile dans l'optimum local.

Sans dérivée "Méta-heuristique"	Algorithme d'essaim de salpes - (Salp Swarm Algorithm - SSA) [119]	2019	C'est un algorithme d'optimisation bio-inspiré basé sur le mécanisme d'essaim des salpes dans le but d'améliorer la précision et la fiabilité de la solution en explorant efficacement l'espace de recherche.	L'algorithme SSA est plus rapide à exécuter en raison de sa moindre complexité comparée à d'autres algorithmes d'optimisation. Encore, il permet d'améliorer la capacité à éviter les minima locaux, ce qui permet de trouver des solutions plus proches de l'optimum global.	Il peut rester bloqué dans la zone locale, ce qui entraîne l'échec de l'obtention de la solution optimale globale.
	Algorithme de libellule - (Dragonfly Algorithm - DA) [121]	2019	Il est inspiré des comportements d'essaim dynamiques et statiques des libellules pour résoudre la stagnation des optima locaux lors de la résolution de problèmes difficiles.	L'algorithme est simple et facile à mettre en œuvre, avec peu de paramètres à régler.	Il ne possède pas de mémoire interne pouvant conduire à une convergence prématurée vers l'optimum local.

Algorithme de recherche gravitationnelle adaptative de meilleure masse (ABGSA) [120]	2019	L'algorithme est utilisé pour résoudre le problème de précision de la classification inappropriée, éviter les minima locaux et améliorer la vitesse de convergence pour les réseaux de neurones multicouches perceptron.	ABGSA peut réduire la complexité et le temps de traitement des problèmes d'optimisation.	Cet algorithme peut avoir un taux d'échantillonnage insuffisant, ce qui peut affecter ses performances. De plus, il est difficile à optimiser en raison de son caractère stochastique et de sa grande sensibilité à la solution initiale.

Optimisation fédérée	consensus flou [125]	2021	La méthode d'apprentissage fédéré (Federated Learning - FL) consiste à entraîner un modèle global sur plusieurs appareils locaux, sans partager les données brutes, afin de préserver la confidentialité. LSGDM est une extension de FL qui utilise une méthode de consensus flou pour améliorer la prise de décision du groupe à grande échelle.	FL présente de l'avantage de pouvoir effectuer une classification rapide des échantillons, même pendant le processus d'apprentissage, car les modèles locaux sont mis à jour périodiquement.	L'adaptation des flux de travail d'entraînement centralisés telle que le réglage de l'hyperparamètre et les tâches d'interprétabilité au sens d'apprentissage fédéré présent des obstacles à l'adoption généralisée de FL dans des contextes pratiques.

3.4 Les fonctions d'activation

Les fonctions d'activation sont utilisées pour un fonctionnement sain des architectures de réseaux de neurones en tenant compte de la vitesse des modèles pendant la phase d'apprentissage, des minima locaux et de l'amélioration de la précision des modèles. Les fonctions d'activation constituent la base des réseaux de neurones pour modéliser et apprendre les relations complexes entre les variables et les couches. Une fonction d'activation bien conçue est efficace sur les performances des modèles d'apprentissage profond.

L'individualisation de nouvelles fonctions d'activation susceptible d'améliorer les résultats, reste un domaine ouvert. Une méthode explorée et prometteuse favorise la possibilité de déterminer la fonction d'activation appropriée par l'apprentissage. En d'autres termes, l'idée clé est d'inclure la fonction d'activation avec d'autres paramètres de réseau tels que les poids et les biais dans le processus d'apprentissage obtenant ainsi une fonction d'activation entraînée [144]–[146]. De nombreux modèles de fonction d'activation entraînables et hétérogènes ont été proposés dans la littérature. Ce sujet présente un intérêt particulier depuis des années, comme illustré dans la figure 3.13.

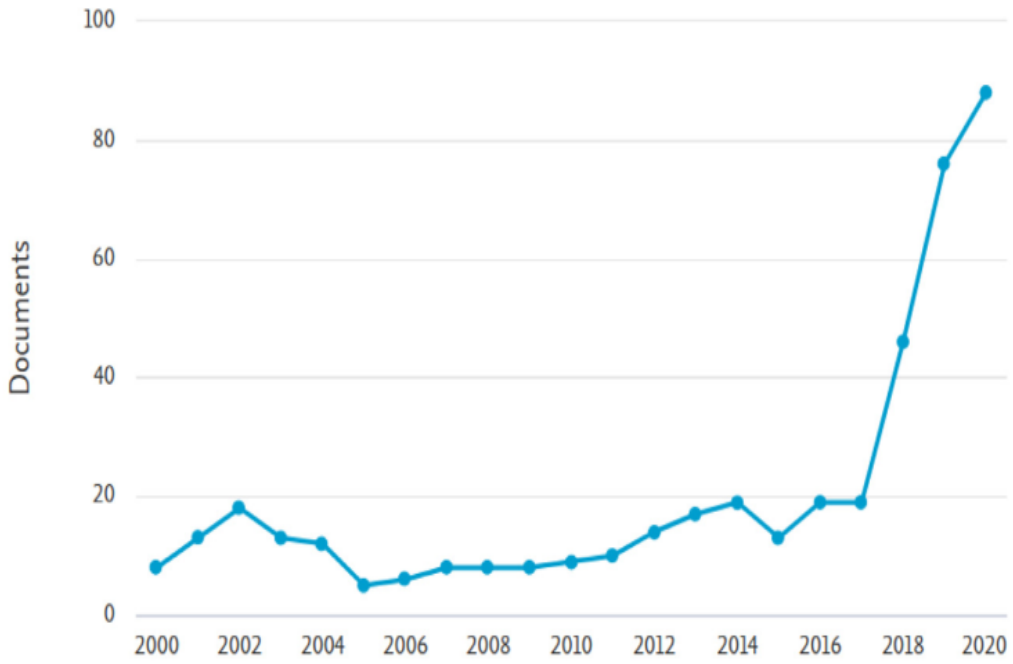


Figure 3.13: Nombre d'articles par an sur les fonctions d'activation entraînables [144].

Dans ce qui suit, nous proposons une taxonomie des principales fonctions d'activation trouvées dans la littérature. Puisque de nombreux modèles de réseaux neuronaux importants utilisant différentes méthodologies ont été décrits, la classification principale est basée sur la capacité à modifier la forme d'une fonction d'activation pendant l'entraînement. De ce fait, on peut distinguer deux grandes catégories [144].

Table 3.2: Certaines des fonctions d'activation fixes les plus utilisées, avec leur étendue.

Nom	Expression	Étendue
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	[0, 1]
Tangente hyperbolique	$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	[-1, 1]
Sigmoïde bipolaire	$\sigma_{\beta}(x) = \frac{1-e^{-x}}{1+e^{-x}}$	[-1, 1]
Tangente hyperbolique dure	$tanh_{H}(x) = max(-1, min(1, x))$	[-1, 1]
Bipolaire	$B(x) = \begin{cases} -1 & \text{si } x < 0 \\ +1 & \text{sinon} \end{cases}$	{-1, 1}
Fonction d'activation à seuil	$Th_{\geq 0}(x) = \begin{cases} 0 & \text{si } x < 0 \\ +1 & \text{sinon} \end{cases}$	{0, 1}
Identité	$id(x) = x$	$(-\infty, +\infty)$
Valeur absolue	$abs(x) = \lvert x \rvert$	$[0, +\infty)$
Cosinus	$cos(x)$	[-1, 1]

3.4.1 Fonctions d'activation à forme fixe

Par la fonction d'activation fixe, on désigne toutes les fonctions d'activation définies sans paramètres modifiables. Puisque de nombreuses fonctions d'activation entraînable sont proposées dans la littérature comme une combinaison ou une variation de fonctions d'activation fixe, cette section décrit les principales fonctions d'activation de forme fixe utilisées dans les réseaux de neurones. Au fil des années, plusieurs études ont été réalisées ayant comme objectif la comparaison de multiples fonctions d'activation fixes [145], [147].

Ce type de fonctions à forme fixe peut être divisé en deux parties : les fonctions d'activation classiques et ReLU et les améliorations potentielles en modifiant leur forme de base.

3.4.1.1 Fonctions d'activation classiques

Une liste courte et partielle des fonctions d'activation les plus couramment utilisées est présentée dans le tableau 3.2 et illustrée à la figure 3.14.

Dans [148], les auteurs montrent que quelle que soit la fonction spécifique continue, elle peut être arbitrairement approximée par un modèle feed-forward d'une seule couche cachée (réseau peu profond). Ceci se fait si le nombre de neurones cachés est

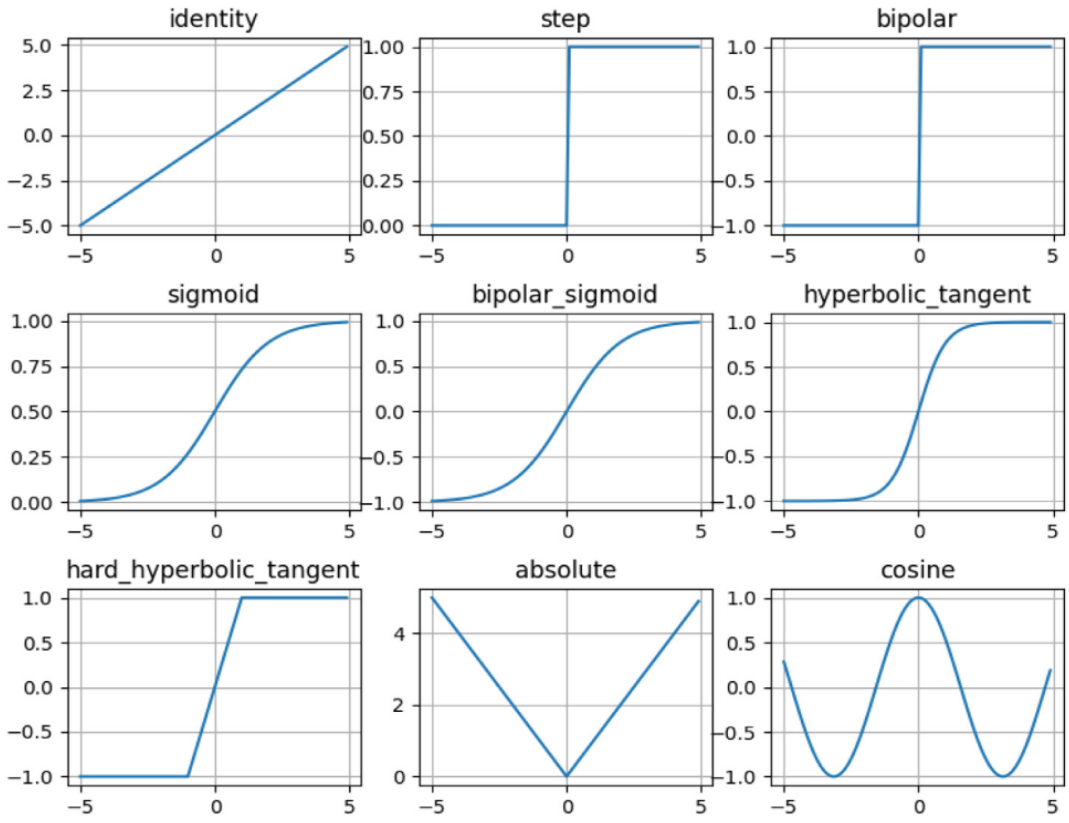


Figure 3.14: Certaines des fonctions fixes les plus couramment utilisées [144] .

assez grand en vue que la fonction d'activation ne soit pas une fonction continue constante, bornée et à croissance monotone. Ce théorème a envisagé que les fonctions d'activation, telles que la fonction d'identité ou d'autres fonctions linéaires utilisées dans le premier réseau de neurones [149], [150], ne peuvent approximer les fonctions continues. Par conséquent, les fonctions bornées telles que fonction sigmoïde [151] et tangente hyperbolique [152] sont les plus couramment appliquées dans les réseaux de neurones.

Une représentation graphique de la fonction sigmoïde est présentée dans la figure 3.15. Comme le montre le graphique, la fonction sigmoïde transforme les entrées en valeurs comprises entre 0 et 1. Cette fonction est utile pour résoudre les problèmes de classification, car elle permet de séparer les entrées en deux classes distinctes en utilisant un seuil. De plus, lorsqu'elle est utilisée pour un modèle de régression, la fonction sigmoïde est généralement combinée avec une fonction de coût pour évaluer les performances du modèle sur les données d'apprentissage et pour déterminer les poids optimaux des données en entrée.

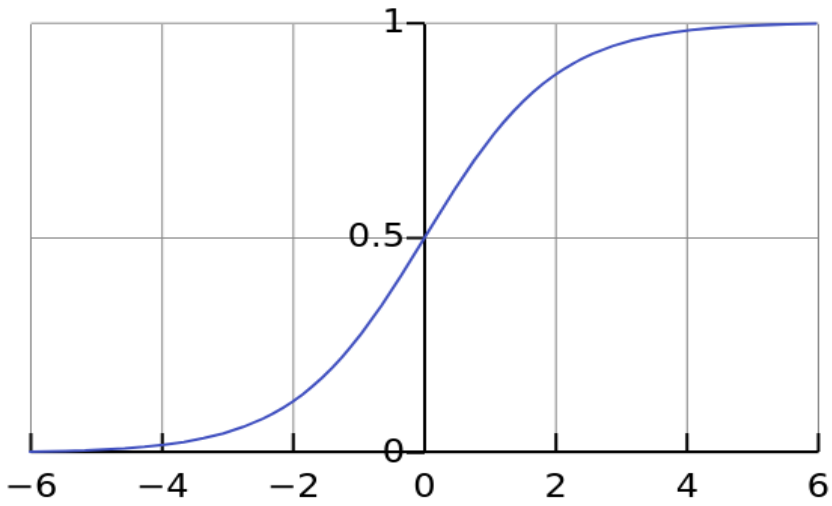


Figure 3.15: Représentation graphique de la fonction sigmoïde. Source https://commons.wikimedia.org/wiki/File:Logistic-curve.svg.

Plusieurs études ont effectivement prouvé comment une fonction d'activation bornée peut obtenir d'excellents résultats, en particulier dans les architectures de réseau peu profondes [153], [154]. Malheureusement, l'entraînement des réseaux équipés de ces fonctions souffre du problème de disparition du gradient [155] lors de l'application des méthodes d'apprentissage profond affectant l'entraînement des réseaux.

3.4.1.2 Fonction d'activation rectifiée

L'intérêt d'appliquer une fonction d'activation rectifiée consiste à éviter le problème de disparition du gradient [155] qui a été identifié comme l'une des principales difficultés des réseaux profonds. D'outre, de nombreuses nouvelles fonctions d'activation ont été développées ces dernières années. La majeure partie est inspirée du succès de *ReLU* [156] basée sur un format similaire avec des modifications mineures par rapport à la fonction origine [157], [158]. La fonction *ReLU* est définie comme

$$ReLU(x) = max(0, x) \tag{3.8}$$

où elle présente des caractéristiques importantes : elle est capable de démystifier le problème de disparition du gradient et de permettre par conséquent un codage parcimonieux [159]. Cependant, des défauts tels que le problème *ReLU* "mourant" [160]

ou la non-différenciabilité à zéro soulève des questions sur la fonctionnalité réelle de *ReLU*.

En conséquence, diverses variantes de *ReLU*, telles que Leaky ReLU (*LReLU*) [160], ont été développées récemment. Elle est définie comme

$$LReLU(x) = \begin{cases} x & \text{si } x < 0 \\ 0.01 & \text{sinon.} \end{cases} \tag{3.9}$$

Cette fonction résout les problèmes potentiels de *ReLU* où elle permet à l'unité de produire un petit gradient lorsque l'unité est saturée et inactive, ou lorsque $x \leq 0$.

Une autre fonction, Truncated rectified, a été introduite dans [161], pour la recherche d'alternatives à *ReLU* tout en se concentrant sur une forme spécifique de réseaux de neurones profonds (Autoencoders). L'un des problèmes que la Truncated rectified résout est celui des biais négatifs qui peuvent survenir lors de l'entraînement des auto-encodeurs. Ces biais peuvent se traduire par des distorsions dans les données encodées, entraînant une perte de précision et des résultats de mauvaise qualité.

La fonction d'activation, Truncated rectified (*TRec*) peut être définie comme :

$$TRec_t(x) = \begin{cases} x & \text{si } x > t \\ 0 & \text{sinon.} \end{cases} \tag{3.10}$$

Introduite en [162], la fonction softplus est considérée comme une approximation lisse de la fonction *ReLU*. Elle est définie comme

$$Softplus(x) = log(1 + exp(x)). \tag{3.11}$$

La forme plus lisse et l'absence de points de non-différenciation pourraient suggérer un meilleur comportement et un entraînement plus facile pour une fonction d'activation.

Une autre variante de fonction d'activation, ELU [163] qui sert à conserver l'identité pour les arguments positifs mais avec des valeurs non nulles pour les arguments négatifs. Elle est définie comme :

$$ELU(x) = \begin{cases} x & \text{si } x > 0 \\ \alpha \times (exp(x) - 1) & \text{sinon.} \end{cases} \tag{3.12}$$

De plus, de nombreuses autres fonctions ont été créées, notamment la fonction E-swish [164], Flatten-T Swish [165], et la fonction unité linéaire pondérée sigmoïde [166].

3.4.2 Fonctions d'activation entraînables

Le concept d'utilisation de fonctions d'activation entraînable n'est pas nouveau. Depuis les années 1990, de nombreuses études ont été publiées sur ce sujet [167]–[170]. Ces dernières années, l'intérêt croissant relatif aux réseaux de neurones a conduit les chercheurs à revoir la notion selon laquelle les fonctions d'activation entraînables peuvent améliorer les performances des réseaux de neurones.

Ce paragraphe donne un aperçu des principales stratégies proposées pour les fonctions d'activation. Trois familles peuvent être distinguées et ce en tenant compte de leurs principales caractéristiques.

3.4.2.1 Fonctions d'activation standard paramétrées

Le terme "fonctions d'activation standard paramétrées" désigne des fonctions qui ont une forme similaire à une fonction standard, mais ayant des paramètres ajustables qui peuvent être entraînés. L'ajout de ces paramètres implique des ajustements, même mineurs, dans le processus d'apprentissage du modèle.

Dans [171], la fonction proposée est une généralisation de sigmoïde classique $sig(x) = \frac{1}{1+exp(-x)}$ avec deux paramètres entraînables α et β pour ajuster la forme de la fonction, c'est-à-dire :

$$AGSig(x) = \frac{\alpha}{1 + exp(-\beta x)}. \tag{3.13}$$

Les deux paramètres sont appris à l'aide de la méthode de descente de gradient et l'algorithme de rétropropagation pour calculer la dérivée de la fonction de coût en fonction des paramètres du réseau.

Même procédure dans [167] où la fonction proposée généralise la tangente hyperbolique classique $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ en introduisant deux paramètres entraînables : α pour définir le niveau de saturation et β pour contrôler par la suite la pente :

$$AGTanh(x) = \frac{\alpha(1 - exp(-\beta x)}{(1 + exp(-\beta x)}. \tag{3.14}$$

Ces deux paramètres sont appris avec les poids du réseau en se basant sur l'algorithme classique de descente combiné à la rétropropagation avec toutes les valeurs initialisées d'une manière aléatoire.

Un travail récent dans [172] tente d'éviter la définition manuelle du paramètre α de l'unité *ELU* tout en fournissant une version alternative basée sur deux paramètres entraînables, c'est-à-dire

$$PELU(x) = \begin{cases} \frac{\beta}{\gamma}x & \text{si } x \geq 0 \\ \beta \times (exp(\frac{x}{\gamma}) - 1) & \text{sinon} \end{cases} \tag{3.15}$$

où β, γ sont des paramètres entraînables.

Une fonction d'activation plus récente, caractérisée par une certaine flexibilité de ReLU, a été proposée dans [173] :

$$frelu(x) = ReLU(x + \alpha) + \beta. \tag{3.16}$$

avec α, β appris par les données. Ceci est fait pour capturer les informations négatives qui sont perdues dans la fonction ReLU classique [163]. Étant donné que la valeur de x est la somme pondérée des entrées et du biais, le paramètre α peut être considéré comme faisant partie de l'entrée de la fonction, de sorte que les auteurs ont réduit la fonction à

$$frelu(x) = ReLU(x) + \beta. \tag{3.17}$$

La fonction d'activation introduite dans [174] est une autre fonction de type ReLU qui apprend partiellement sa forme à partir de l'ensemble d'apprentissage. En effet, elle peut modifier la partie négative des données via le paramètre α. Cette fonction s'appelle ReLU Paramétrique où elle peut être définie comme suit :

$$PReLU(x) = \begin{cases} x & \text{si } x > 0 \\ \alpha \times x & \text{sinon.} \end{cases} \tag{3.18}$$

où le paramètre α est appris conjointement avec l'ensemble du modèle en utilisant la méthode de descente de gradient.

Les auteurs dans [175] présentent une méthode de recherche visant à identifier les fonctions d'activation les plus performantes pour résoudre le problème de la disparition du gradient. Ils ont créé un ensemble de nouvelles fonctions candidates en combinant différentes fonctions à partir d'un ensemble prédéfini de fonctions d'activation de base. Ensuite, pour chaque fonction d'activation candidate, un réseau utilisant la fonction générée, est entraîné sur une tâche donnée afin d'évaluer les performances. Parmi toutes les fonctions testées, la meilleure est la fonction "Swish" définie comme suit :

$$Swish(x) = x \times sig(\alpha x). \tag{3.19}$$

où α est un paramètre entraînable.

3.4.2.2 Fonctions basées sur des méthodes d'ensemble

Le terme "méthode d'ensemble" fait référence à toute technique fusionnant différentes fonctions. Fondamentalement, chacune de ces techniques utilise :

- un ensemble de fonctions standards, qui peut contenir des fonctions à forme fixe ou des fonctions entraînables ou les deux ;
- un modèle de combinaison, qui définit la façon dont les fonctions de base sont combinées entre elles.

En tant qu'exemple de ce type de méthodes, les auteurs en [176] proposent une méthode pour étudier les fonctions d'activation construites comme des composites de diverses fonctions d'activation élémentaires.

Dans [177], une méthode similaire a été introduite, prévoyant un algorithme génétique composé de deux nouvelles fonctions d'activation. La première fonction proposée par les auteurs, Exponential Linear Sigmoid SquasHing (EliSH), est définie comme

$$EliSH(x) = \begin{cases} \frac{x}{1+\exp^{-x}} & \text{si } x \geq 0 \\ \frac{\exp^{x}-1}{1+\exp^{-}x} & \text{si } x < 0. \end{cases} \tag{3.20}$$

À partir de l'équation (3.20), il est clair que *ELiSH* partage les propriétés de *Swish* (Eq. 3.19), puisque sa partie négative est une multiplication de *ELU* et *Sigmoid*, tout en partageant la même partie positive avec *Swish*. En ce qui concerne la deuxième fonction, *HardELiSH*, elle est définie comme étant une multiplication de *HardSigmoid* et *ELU* dans la partie négative et de *HardSigmoid* et Linear dans la partie positive

$$HardELiSH(x) = \begin{cases} \max(0, \min(1, (x+1)/2)) \times x & \text{si } x \geq 0 \\ \max(0, \min(1, (x+1)/2)) \times (e^{x}-1) & \text{si } x < 0. \end{cases} \tag{3.21}$$

Dans les expériences décrites dans [178], la valeur de k a été sélectionnée à l'aide d'un processus de validation croisée. L'estimation des paramètres w_i et b_i ont quant à eux été régularisés avec une pénalité ℓ_2, dans le but d'empêcher l'optimiseur de converger vers des valeurs trop élevées pour ces paramètres, ce qui pourrait causer une instabilité numérique.

Dans [179], une autre fonction d'activation intéressante est proposée, appelée Mexican Hat Linear Unit (*MeLU*). Cette fonction d'activation résout les problèmes d'apprentissage instable liés aux paramètres entraînables. Les problèmes d'apprentissage instable entraînent une baisse de la précision et une augmentation de l'erreur de généralisation.

Ce phénomène se produit lorsque les performances d'un modèle varient en réponse aux changements légères caractérisant les données ou les paramètres, ce qui entraîne des modifications importantes dans ses prédictions. Les fonctions de type "chapeau mexicain" ont une courbe plus lisse que la *ReLU*, ce qui empêche la saturation et permet d'atteindre des performances optimales.

Pour définir la fonction *MeLU*, soit :

$$f_{\gamma,\lambda}(x) = max(\lambda - |x - \gamma|, 0). \tag{3.22}$$

où λ, γ sont des nombres réels. Cette fonction est nulle lorsque $|x - \gamma| > \lambda$. De plus, elle augmente avec une dérivée de 1 entre $\gamma - \lambda$ et γ, puis diminue en utilisant une dérivée de moins 1 entre γ et $\gamma + \lambda$. *MeLU* est définie pour chaque couche comme,

$$MeLU(x) = PReLU(x) + \sum_{j=1}^{k-1} c_j f_{\gamma_j,\lambda_j}(x). \tag{3.23}$$

où k représente le nombre de paramètres qui peut être appris dans chaque neurone, c'est-à-dire, les $k - 1$ coefficients des fonctions de chapeau mexicain plus un paramètre apprenable dans *PReLU*. Les paramètres c_j sont des réels apprenables, γ_j, λ_j sont des paramètres fixes.

Les fonctions d'activation développées dans [180] visent à adapter les réseaux de neurones artificiels aux problèmes d'apprentissage avec des ensembles de données difficiles du monde réel. Bien qu'il soit avantageux d'avoir des fonctions d'activation personnalisées, il est également important d'identifier des fonctions qui se généralisent à plusieurs ensembles de données du monde réel. Les auteurs ont évalué les trois fonctions d'activation suivantes :

$$TanhSoft - 1 : \mathcal{F}_1(x;\alpha) = \tanh(\alpha x)Softplus(x), \tag{3.24}$$

$$TanhSoft - 2 : \mathcal{F}_2(x;\beta,\gamma) = x\tanh(\beta.e^{\gamma x}), \tag{3.25}$$

$$TanhSoft - 3 : \mathcal{F}_3(x;\delta) = ln(1 + e^x \tanh(\delta x)), \tag{3.26}$$

où α, β et δ sont des paramètres fixes.

Dans [181], les auteurs ont introduit une nouvelle fonction appelée, Adaptive Blending Units (*ABU*). Il s'agit d'une combinaison linéaire entraînable d'un ensemble de fonctions d'activation. Cette fonction s'appuie sur la même idée qui a motivé l'introduction de "TanhSoft", à savoir que les fonctions d'activation courantes ne sont pas toujours les plus adaptées aux spécifications du réseau et aux tâches d'apprentissage. la fonction *ABU* est définie comme :

$$ABU(x) = \sum_{i=1}^{k} \alpha_i \times f_i(x). \tag{3.27}$$

où (α_1, α_2,..., α_k) sont des paramètres à apprendre et $f_1(.)$, $f_1(.)$, ..., $f_k(.)$ est un ensemble de fonctions d'activation composé de *tanh* (Tab. 3.2), *ELU* (Eq. (3.12)), *ReLU* (Eq. (3.8)), *id* (Tab. 3.2) et *Swish* (Eq. 3.19). Les paramètres α_i sont tous initialisés à la valeur $\frac{1}{k}$ et sont entraînés par la méthode de descente de gradient.

Les auteurs de [182] ont présenté différents mélanges des fonctions *ELU* et *ReLU* pour obtenir une fonction d'activation apprise à partir des données, dans un contexte probabiliste et hiérarchique, dans le but de résoudre le problème de disparition de gradient. Plus précisément, les auteurs proposent les fonctions d'activation suivantes :

- activation mixte (illustrée dans Figure 3.16[a]) :

$$f_{mix}(x) = p \times f_{lrelu}(x) + (1 - p) \times f_{elu}(x). \tag{3.28}$$

  avec $p \in [0, 1]$ appris à partir des données, $f_{lrelu}(x) = LReLU(x)$, et $f_{elu}(x) = ELU(x)$;

- activation contrôlée (figure 3.16[b]):

$$f_{gate}(x) = sig(\omega x) f_{lrelu}(x) + (1 - sig(\omega x)) f_{elu}(x). \tag{3.29}$$

- Les deux stratégies présentées ci-dessus se concentrent sur les manières de combiner des fonctions d'activation de base avec des paramètres prédéfinis. Une autre extension consiste à ajuster les paramètres des fonctions d'activation de base. Les auteurs proposent une troisième stratégie appelée activation hiérarchique (voir figure 3.16[c]). Cette stratégie hiérarchique comporte trois niveaux:

les nœuds de bas niveau, les nœuds de niveau intermédiaire et les nœuds de haut niveau. Chaque nœud de bas niveau est activé par une fonction d'activation de base $f_{low}^{n}(x)$, où n est l'indice du nœud. Les nœuds de niveau intermédiaire sont activés par la stratégie "controlée" qui combine les valeurs de deux nœuds de bas niveau enfants pour produire une valeur parente $f_{mid}(x)$ en utilisant un masque de déclenchement appris noté ω_m. Chaque nœud de niveau intermédiaire peut être vu comme une paire de nœuds de bas niveau $f_{mid}^{m,left}(x)$ et $f_{mid}^{m,right}(x)$, où m est l'indice du nœud de niveau intermédiaire. Enfin, les nœuds de haut niveau sont activés en sélectionnant la valeur maximale parmi un ensemble de nœuds de niveau intermédiaire. Cette opération d'activation globale est notée $f_{high}(x)$.

$$f(x) = \begin{cases} f_{low}^{n,left}(x) & \textit{si nœuds de bas niveau} \\ sig(\omega x) f_{mid}^{m,left} + (1 - sig(\omega x) \times f_{mid}^{m,right} & \textit{si nœuds de niveau intermédiaire} \\ max f_{mid}^{m}(x) & \textit{si nœuds de haut niveau} \end{cases} \tag{3.30}$$

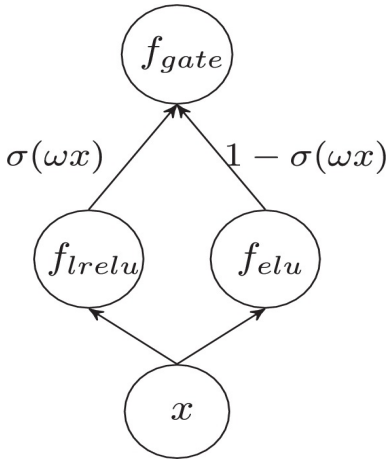
où $f_{mid}^{m,left}(x) = PReLU(x)$ et $f_{mid}^{m,right}(x) = PELU(x)$

Une autre approche récente présentée dans [183] consiste à exprimer les fonctions d'activation entraînables à l'aide de sous-réseaux à une seule couche cachée. En effet, un réseau de neurones à une couche cachée peut approximer avec précision toute cartographie fonctionnelle continue d'un espace de dimension finie à un autre, ce qui permet la création de formes variées. La fonction d'activation proposée est modélisée comme une fonction d'activation non linéaire, avec un neurone doté d'une fonction d'activation identité qui envoie sa sortie à un sous-réseau à une couche cachée avec un seul neurone de sortie ayant également une fonction d'activation identité.

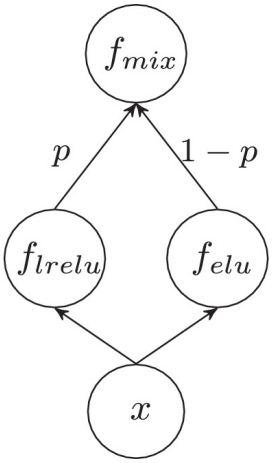
Le modèle proposé peut être formalisé comme

$$VAF(x) = \sum_{j=1}^{k} \beta_j g(\alpha_j x + \alpha_{0j}) + \beta_0. \tag{3.31}$$

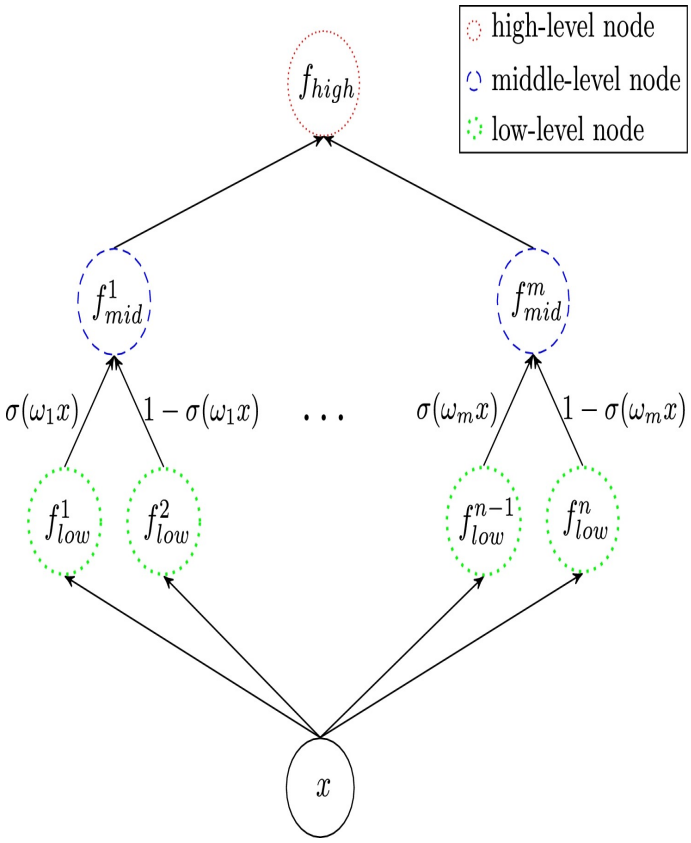
Dans cette équation, g est une fonction d'activation de forme fixe, k est un hyper-paramètre qui détermine le nombre de neurones dans la couche cachée, α_j, α_{0j}, β_j, β_0 sont des paramètres appris à partir des données lors du processus d'apprentissage, et j représente l'indice du neurone dans la couche cachée du sous-réseau. Un schéma VAF général est illustré dans la Figure 3.17.



(a) : Activation mixte



(b) : Activation controlée



(c) : Activation hiérarchique

Figure 3.16: Fonctions d'activation adaptative [182].

"Kafnets" est une autre approche proposée dans [184] pour améliorer les performances des réseaux de neurones. Elle propose d'utiliser des fonctions d'activation non paramétriques basées sur un noyau pour remplacer les fonctions d'activation classiques. Ces fonctions ont une flexibilité limitée, ce qui entraîne en moyenne des gains de

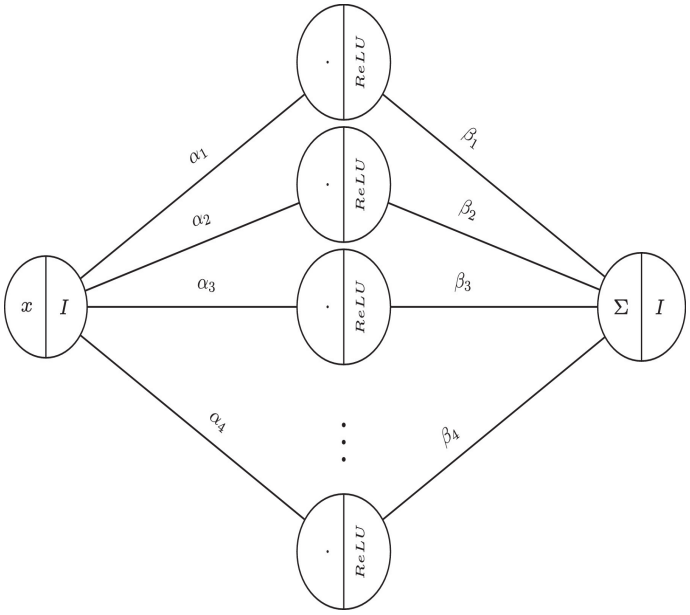


Figure 3.17: Un schéma VAF général [183].

performances mitigés. Les fonctions d'activation du noyau (kernel Activation Function - KAF) ont des propriétés, telles que : *i)* elles peuvent être calculées à moindre coût en utilisant des opérations vectorielles-matrices ; *ii)* elles sont linéaires par rapport aux paramètres entraînables ; et *iii)* les paramètres peuvent être régularisés en utilisant n'importe quelle approche classique, y compris la possibilité d'imposer la parcimonie par l'utilisation de normes ℓ_1. La fonction d'activation *KAF* est modélisée en termes d'expansion du noyau comme suit :

$$KAF(x) = \sum_{i=1}^{D} \alpha_i \times k(x, d_i). \tag{3.32}$$

où α_1, α_2,..., α_D sont des paramètres entraînables, d_1, d_2,..., d_D sont appelés les éléments d'un dictionnaire qui sont généralement sélectionnés à partir des données d'apprentissage, et k une fonction noyau $k : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ [185]. De plus, dans [186], les auteurs montrent deux méthodes différentes pour adapter la fonction d'activation *KAF* afin qu'elle soit également utilisée avec les réseaux de neurones à valeurs complexes [187].

3.4.2.3 Fonctions d'activation basées sur d'autres techniques

Nous introduisons un autre type de techniques de modélisation pour la fonction d'activation. Dans [188], une tentative de modélisation d'une fonction d'activation polynomiale à coefficients ajustables a été proposée. Pour un degré donné $k \in \mathbb{N}$, la fonction polynomiale relative est définie par

$$AP(x) = \sum_{i=0}^{k} \alpha_i \times x^i. \tag{3.33}$$

où (α_0, α_1, α_2,..., α_k) est un vecteur de paramètres à apprendre. Ces paramètres peuvent être appris avec les paramètres du réseau à l'aide de l'algorithme de rétropropagation. Toutefois, il convient de noter que les fonctions polynomiales ne sont pas des approximateurs universels [189].

D'autres catégories de méthodes de logique floue ont été largement utilisées pour modéliser des fonctions d'activation, telles que les méthodes de logique floue de type 2 présentées dans [190]–[192]. Cette forme avancée de logique floue permet aux systèmes d'inférence d'utiliser des variables à plusieurs niveaux. Elle est basée sur l'utilisation de fonctions d'appartenance à l'ensemble flou qui sont définies par des intervalles et non par des points précis. Ces intervalles sont utilisés pour représenter des valeurs incertaines et imprécises, ce qui implique une plus grande flexibilité et une plus grande précision dans la représentation des données [193]–[195].

Dans [196], une unité neuronale basée sur la logique floue de type 2 est proposée, IT2-FRU (Interval Type-2 Fuzzy Rectifying Unit), principalement pour atténuer le problème de disparition de gradient. Elle peut être exprimée comme une fonction simple ou sophistiquée en fonction de trois hyperparamètres représentant des pentes négatives qui peuvent être définies également comme positives. Ces hyperparamètres peuvent être ajustables et appris avec les autres paramètres du réseau.

La fonction d'activation floue proposée, IT2-FRU, repose sur les propriétés du système de logique floue IT2-FLS (Interval Type-2 Fuzzy Logic Systems). Ce système est constitué de trois règles floues de la forme

$$\text{Si } \sigma \text{ est } \tilde{A} \text{ Alors } \varphi_0 \text{ est } B_n, \quad n = 1, 2, 3. \tag{3.34}$$

où σ est la valeur d'entrée normalisée, φ_0 est la sortie et B_n sont les parties conséquentes de la règle floue définies par des fonctions d'appartenance. Les parties antécédentes sont définies par des IT2-FS (Interval Type-2 Fuzzy Sets). La fonction IT2-FRU est formulée comme

$$\varphi(\sigma) = P\sigma K(\sigma) \tag{3.35}$$

où P est un paramètre à définir manuellement, $K(\sigma)$ est défini comme

$$K(\sigma) = \frac{1}{2}\left(\frac{1}{\alpha + \sigma - \alpha\sigma} + \frac{-1+\alpha}{-1+\alpha\sigma}\right) \tag{3.36}$$

D'autres travaux ont proposé des fonctions d'activation dont la forme est calculée à l'aide de techniques d'interpolation, comme décrit dans [197]. Ces techniques exigent souvent des entrées supplémentaires. Pour pallier ce problème, les auteurs dans [168] ont introduit l'utilisation de fonctions d'activation basées sur les splines.

3.4.3 Discussion

L'apprentissage des fonctions d'activation est un sujet populaire dans le domaine de l'apprentissage automatique, car les performances des architectures d'apprentissage peuvent être améliorées grâce à des fonctions d'activation mieux adaptées. Plusieurs approches ont été explorées pour améliorer ces performances. La plupart des fonctions d'activation entraînables sont des variantes des fonctions d'activation standard, dont la forme est ajustée à l'aide de paramètres entraînables. Par exemple, la fonction AGSig (*AGTanh*) se traduit par la fonction sigmoïde (*Tanh*) avec un lissage et une amplitude réglés par les paramètres α et β. De mme, Swish peut être considérée comme une variation paramétrée de SiLU/ReLU, dont la forme finale est apprise pour trouver un bon compromis entre ces deux fonctions. Plusieurs travaux sur les fonctions d'activation entraînables signalent des améliorations de performances substantielles par rapport aux architectures de réseaux neuronaux équipées de fonctions d'activation fixes classiques telles que *ReLU* ou *sigmoïde*.

D'autres fonctions d'activation entraînables peuvent être exprimées comme des sous-réseaux imbriqués dans les architectures principales, ou celles qui sont basées sur des approches différentes de celles des fonctions d'activation classiques comme la

logique floue [192], [196] ou les interpolation de splines [197]. Bien que ces fonctions d'activation aient un potentiel important pour améliorer les performances des modèles de réseau de neurones, leur mise en œuvre peut être plus complexe et nécessite plus de temps et de ressources pour l'apprentissage.

Malgré les résultats encourageants, il est toujours difficile d'identifier une stratégie pour l'apprentissage automatique d'une fonction d'activation qui résoudrait les différents problèmes et améliorerait considérablement les performances. La plupart des fonctions d'activation entraînables utilisent la méthode de la descente de gradient pour l'estimation des hyperparamètres. Les principales limites de ces techniques résident dans le temps de calcul et la disparition du gradient. Ce processus empêche le réseau d'apprendre des caractéristiques en profondeur et peut même l'amener à utiliser une capacité de traitement excessive pendant l'entraînement.

En effet, les réseaux de neurones peuvent être piégés dans des minima locaux [198], ce qui peut nuire aux performances du modèle. Ce phénomène est en partie dû à la disparition du gradient qui se produit lorsque les dérivées diminuent à mesure que le modèle s'approfondit. Cette baisse de gradient rend plus difficile l'optimisation du modèle, ce qui peut entraîner une diminution de ses performances.

3.5 Conclusion

L'apprentissage profond est un sujet vaste et en constante évolution, utilisé dans de multiples domaines grâce à sa capacité à résoudre des problèmes complexes. Les méthodes d'apprentissage profond s'articulent autour du développement des modèles informatiques basés sur les structures de données complexes. Elles créent des modèles de calcul qui se composent de différentes couches de traitement dans le but de construire divers niveaux d'abstraction de données complexes.

Nous avons introduit dans ce chapitre les concepts de base des méthodes d'apprentissage, y compris les *CNN*. Ces méthodes sont basées sur l'extraction de caractéristiques importantes des images pour pouvoir prédire des résultats corrects. Ensuite, nous avons énuméré diverses méthodes et techniques d'optimisation, ainsi que plusieurs fonctions d'activation qui existent dans la littérature. Les performances des algorithmes

d'apprentissage profond dépendent fortement des algorithmes choisis. Cependant, malgré la variété des méthodes, il existe de nombreuses limitations qui peuvent réduire la précision du modèle d'apprentissage. Dans les chapitres suivants, nous proposons des modèles pour apporter des éléments de réponse à certains problèmes des méthodes d'apprentissage.

4

Prognostic d'images médicales à l'aide de réseaux de neurones récurrents et convolutifs

Sommaire

4.1 Introduction

En décembre 2019, le premier cas de coronavirus 2019 a été découvert en Chine. Le virus s'est propagé de manière agressive dans plusieurs pays [199]. Jusqu'à présent, COVID-19 a touché un grand nombre de personnes, près de 630 millions de cas et 6,5 millions de décès dans le monde [200]. La gravité du COVID-19 réside dans le fait qu'il peut engendrer des maladies des voies respiratoires, des maux de tête, des étourdissements, et surtout des pneumonies sévères dans certains cas extrêmes [201], [202]. La pneumonie est une infection provocant une inflammation principalement au niveau des alvéoles pulmonaires responsables des échanges d'oxygène [203], [204].

Les cas graves de COVID-19 peuvent être dus à deux faits générateurs à savoir la forte contagion et le degré de gravité [205]. L'impact sur les systèmes de santé est également important et crucial en raison du nombre de personnes qui ont besoin d'accéder aux unités de soins intensifs (USI) et aux ventilateurs mécaniques pendant de longues périodes [206]. La mise en évidence d'une infection au COVID-19 est généralement associée à des pathologies pulmonaires spécifiques. Le moyen le plus fiable pour évaluer l'existence d'ARN (Acide Ribonucléique) Covid-19 chez l'individu hôte est d'effectuer des tests PCR (Polymerase Chain Reaction) [207]. Les tests PCR bénéficient de taux de précision élevés. Cependant, la principale faiblesse d'une telle méthode moléculaire réside dans sa grande spécificité. Les techniques de PCR en temps réel ont désormais remplacé les PCR conventionnelles car elles sont plus sensibles, plus précises, plus reproductibles et mieux adaptées aux grandes séries.

Au tout début de la propagation de l'épidémie, la plupart des pays réservaient la réalisation de ces tests aux sujets souffrant de symptômes peu clairs et avancés, ou bien les sujets qui ont été en contact avec des personnes infectées.

En ce sens, la radiographie est une technique encore largement utilisée aujourd'hui pour identifier la maladie COVID-19. Des examens radiologiques peuvent être réalisés dans le but d'analyser les images radiographiques thoraciques, tout en identifiant les tissus pulmonaires potentiellement porteurs d'infections [208]. Cependant, l'analyse des données radiographiques nécessite un expert en radiologie, ce qui ralenti la procédure de détection.

Par conséquent, le développement d'un système d'analyse automatisé peut résoudre

ce problème. Dans ce contexte, les solutions basées sur l'intelligence artificielle peuvent fournir un diagnostic plus précis, plus fiable et moins coûteux, qu'il soit dédié au COVID-19 ou d'autres types de pneumonie [209]. De plus, il est nécessaire de vérifier la progression de la pathologie chez les personnes infectées, en particulier celles souffrant de pathologies pulmonaires sévères nécessitant le cas échéant des soins intensifs. Un diagnostic pathologique est également utile pour prioriser le nombre de patients à traiter, notamment lorsque le nombre d'unités de soins intensifs est limité [210], [211].

Nous présentons un modèle de pronostic pathologique pour l'analyse de séries longitudinales d'images radiographiques thoraciques. Le modèle proposé consiste en un réseau de neurones récurrent et convolutif, et permet de classer les patients en deux classes de sévérité : évolution positive ou négative.

4.2 État de l'art

Les données des séries chronologiques font référence à un flux cohérent d'un ensemble de données sur une période espacée dans le temps. L'analyse des séries temporelles est devenue un domaine d'intérêt récent en intelligence artificielle. Des prévisions précises deviennent de plus en plus vitales afin de prendre des décisions plus éclairées et précises. L'analyse des séries chronologiques est principalement utilisée pour : *i)* l'analyse descriptive, c'est-à-dire l'identification des tendances dans les données corrélées, *ii)* la prévision, pour prédire les événements à court terme et *iii)* l'analyse d'intervention pour étudier comment l'événement peut être évalué au cours de la série chronologique.

Cependant, l'analyse de séries temporelles présente généralement une limitation liée au manque de jeux de données annotés. Les annotations du temps T_n doivent généralement tenir compte des temps T_i avec $i < n$. Disposer de données comparables temporellement est donc un problème supplémentaire qui rend difficile la conception d'algorithmes génériques. En ce sens, ce problème a été abordé dans la littérature, notamment pour le clustering [212], la classification [213], la détection de changements [214], [215], et la prévision [216].

En imagerie médicale, les séries temporelles sont souvent utilisées pour l'imagerie fonctionnelle [217]–[219] et spectroscopique [220], ainsi que l'analyse du mouvement [221]. Dans ce contexte, les images radiographiques, et plus particulièrement la tomodensitométrie sont largement recommandées en raison de leurs hautes résolutions spatiales et temporelles. En ce qui concerne le diagnostic pulmonaire, les images radiographiques sont généralement utilisées pour suivre une pneumonie ou d'autres maladies spécifiques. Pour analyser des données tridimentionnelles et temporelles, il est important de concevoir des algorithmes sophistiqués, automatiques, rapides et précis.

Par ailleurs, l'apprentissage profond a démontré son efficacité dans divers domaines. Depuis 2012, plusieurs modèles de réseaux de neurones profonds à convolution [222] ont été proposés comme *AlexNet* [223], *VGG* [224], *GoogleNet* [225], *ResidualNet* [226], *DenseNet* [227] et *CapsNet* [228].

En lien avec le problème abordé ici, la classification des radiographies thoraciques est loi d'être une nouvelle application de l'apprentissage profond. De nombreux ensemble de données ont été publiés et notamment celles relatives à la classification du COVID-19. Des travaux récemment menés ont permis de concevoir des outils d'aide aux diagnostics basés sur les méthodes d'apprentissage profond afin de détecter les spécificités du Covid-19 dans les images radiographiques pulmonaires. La quasi-totalité de ces travaux optent pour des méthodes d'apprentissage par transfert, et certains d'entre-eux envisagent de nouvelles architectures *CNN* avec des performances comparables aux méthodes basées sur l'apprentissage par transfert [229].

Dans [230], un réseau de neurones modifié ResNet-50 a été proposé, sur lequel un réseau pyramidal de fonctionnalités est utilisé pour identifier et extraire automatiquement les lésions des images *TDM*. Grâce à cette approche, le modèle peut détecter et classer les images CT en trois classes possibles : sain, COVID-19 et pneumonie bactérienne. De même, des images radiographiques thoraciques ont été utilisées dans [231]. Les auteurs utilisent un *CNN* basé sur divers modèles pré-entraînés ImageNet[1] pour extraire les fonctionnalités de haut niveau (fonctionnalités profondes). Ces différentes fonctionnalités ont été introduites dans une machine à vecteurs de support (*SVM*) en tant que classificateur d'apprentissage automatique qui constitue

[1] http://www.image-net.org/

l'amélioration majeure de cette approche.

Une autre étude [232] adopte une architecture (*CNN*) profonde, appelée DeTraC [233] (Decompose, Transfer, and Compose) et orientée vers la classification d'images de radiographie pulmonaire COVID-19. Le modèle DeTraC se compose de trois phases. La première phase consiste à entraîner le *CNN* pré-entraîné pour extraire les caractéristiques profondes de chaque image. Ensuite, la méthode de décomposition de classes [234] est appliquée pour simplifier la structure locale de la distribution des données. Elle vise à partitionner chaque classe de l'ensemble de données d'image en plusieurs sous-classes. De nouvelles étiquettes sont ensuite attribuées au nouvel ensemble, où chaque sous-ensemble est considéré comme une classe indépendante. Lors de la deuxième phase, l'entraînement est réalisé en utilisant la méthode de descente de gradient. Pour la troisième phase, une méthode de composition de classe est exploitée pour assembler les sous-ensembles afin de produire la classification finale des images.

L'efficacité et la performance de la classification COVID-19 susmentionnées sont dues à l'utilisation de l'apprentissage par transfert. Ce dernier offre une solution efficace avec peu d'images annotées, en transférant les connaissances des *CNN* pré-entraînés vers des tâches d'imagerie médicale spécifiques. Malgré les performances obtenues par les différentes techniques décrites ci-dessus pour la détection du COVID-19, ces approches ne tiennent pas compte des corrélations temporelles dans la proportion où elle ne fournissent une décision que sur une seule image.

Le pronostic reste donc une question ouverte. En effet, le suivi de l'évolution de la maladie chez les patients déclarés infectés est une tâche essentielle notamment pour les patients hospitalisés. Le problème est qu'aucun hôpital n'est en mesure de traiter tous les cas positifs de COVID-19. Selon plusieurs experts en médecine pulmonaire dans le monde, le manque d'oxygène pourrait mettre les patients COVID-19 en danger de mort. Les personnes atteintes de maladies chroniques sont les plus vulnérables à ce risque mortel. Afin de gérer toute situation épidémique, il est important de rechercher d'autres moyens susceptibles de sauver le maximum de patients, tout en tenant compte bien évidemment de la gravité et la particularité de chaque situation.

Les réseaux de neurones récurrents (*RNN*) sont une famille de méthodes d'apprentissage profond conçues pour gérer les corrélations temporelles entre les images dans des

séries temporelles. Les *RNNs* ont des connexions récurrentes, c'est-à-dire que les sorties des étapes précédentes sont alimentées en entrée de l'étape courante, dans le sens où ils gardent des informations en mémoire : ces réseaux peuvent prendre en compte à l'instant T_n un certain nombre d'états passés T_i où $i < n$. Ils ont été utilisés entre autres en télédétection pour évaluer la détection des changements dans les images multispectrales et hyperspectrales [235].

Les *RNN* sont principalement appliqués pour gérer de longues séquences, mais ils sont capables de mémoriser des caractéristiques durant une durée limitée. La figure 4.1 représente à quoi ressemble un *RNN* typique :

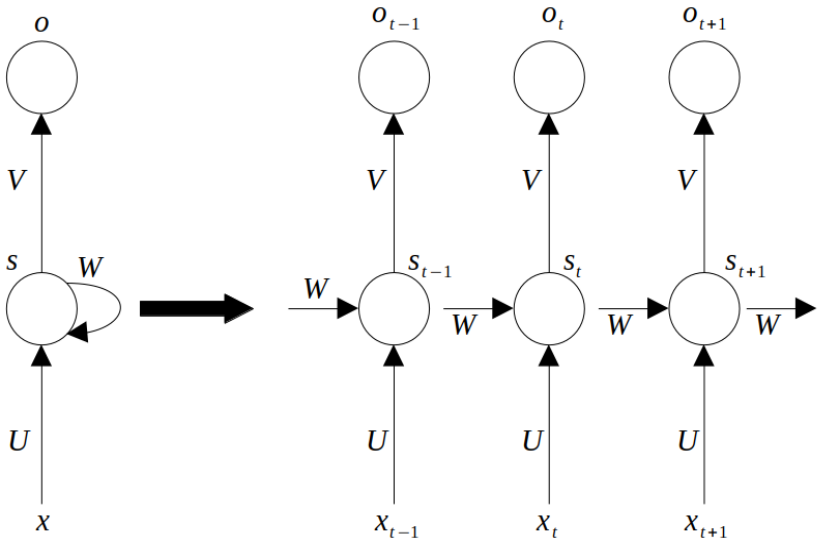


Figure 4.1: Cellule de réseau de neurones récurrent.

où :

- x_t représente un vecteur d'entrée à l'instant t.
- U, V, W sont des poids du modèle d'apprentissage.
- s_t représente un état caché à l'instant t. Il constitue la "mémoire" d'un *RNN* et est calculé en utilisant les vecteurs d'entrée et l'état cachés.
- o_t est le vecteur de sortie du réseau *RNN*.

L'état caché s_t ainsi que le vecteur de sortie o_t sont représentés dans les équations 4.1 et 4.2.

$$s_t = f(Ux_t + Ws_{t-1}). \tag{4.1}$$

où f désigne l'une des deux fonctions *ReLU* ou la tangente hyperbolique,

$$o_t = softmax(Vs_t). \tag{4.2}$$

Le problème majeur des *RNNs* est qu'ils commencent à "oublier" après quelques itérations, ce qui rend l'apprentissage compliqué pour de nombreuses applications. Les algorithmes utilisés pour mettre à jour le poids dans les *RNN* sont principalement basés sur le gradient ; ils présentent certains problèmes pratiques bien connus tels que l'explosion du gradient [236].

Depuis leur création en 1995, les réseaux de longue mémoire à court terme (Long short-term memory - *LSTM*) [237]–[239] ont montré des performances prometteuses pour pallier la limitation des *RNN*. Ces modèles capturent explicitement les corrélations temporelles récursives dans la mesure où ils ont déjà prouvé leur efficacité dans divers domaines, tels que la reconnaissance vocale [240], le traitement du langage naturel [241]–[244] et la complétion d'images [245]. Les *LSTM* ont récemment été utilisés en imagerie médicale comme dans [246] où les auteurs proposent une méthode avec multi-modalité et des contraintes d'adjacences pour la segmentation de l'imagerie cérébrale. Enfin, il convient de noter que les *LSTM* et réseaux de neurones convolutifs ont déjà été combinés dans un certain nombre de travaux [247]–[249].

4.3 Méthode proposée : classification d'images COVID-19

4.3.1 Aperçu général

La méthodologie proposée consiste à appliquer une architecture d'apprentissage profond dédiée à la classification multi-temporelle d'images radiographiques pour évaluer l'évolution du COVID-19.

La méthodologie proposée est basée sur la combinaison des architectures *CNN* et *RNN*.

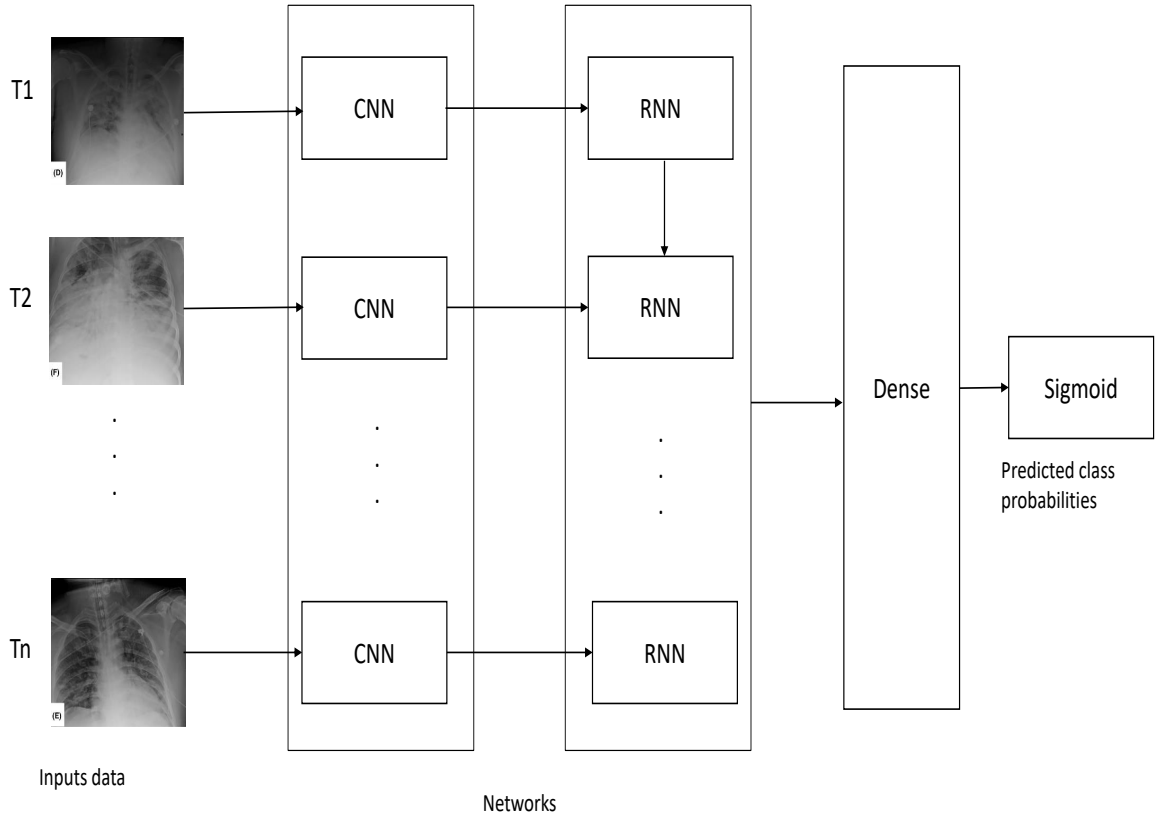


Figure 4.2: Aperçu de l'architecture ProgNet proposée.

Le *CNN* agit comme un détecteur de caractéristiques entraînable pour le signal spatial. Il apprend de puissantes caractéristiques convolutionnelles qui opèrent sur l'image d'entrée spatiale tandis que le *RNN* reçoit une séquence de ces caractéristiques pour étudier l'évolution des images dans le temps. L'architecture proposée implique donc une dualité au niveau de la classification des séries temporelles à savoir l'évolution positive ou négative. Une telle classification peut aider les médecins à deviner un pronostic opportun pour les patients dans des situations critiques.
Notre architecture est capable d'apprendre automatiquement les corrélations temporelles des images fournies en entrée (voir figure 4.2).

Pour une séquence $T = (T_1, T_2, \ldots, T_n)$ d'images à rayon X, chaque image T_i passe par un *CNN* afin d'extraire un vecteur caractéristique. Le vecteur obtenu est un ensemble d'informations générées suite à l'application de plusieurs convolutions suivies d'une couche de pooling. ResNet-50 est l'une des architectures potentielles à utiliser comme

CNN où il a d'ailleurs montré son efficacité dans différentes applications de classification d'images.

Un *RNN* est ensuite appliqué pour apprendre des corrélations temporelles entre les différents vecteurs caractéristiques liés à chaque image d'une séquence T. *LSTM* est utilisé en raison de ses bonnes performances démontrées dans la littérature [237]–[239]. Quatre couches fully-connected avec une fonction de décision *sigmoïde* sont ensuite appliquées pour effectuer une classification binaire.
Une description détaillée des réseaux adoptés est fournie dans le paragraphe 4.3.2.

4.3.2 Architecture utilisée

4.3.2.1 ResNet

Un réseau résiduel (ResNet) est un *CNN* largement utilisé dans diverses disciplines. Dans les réseaux profonds, les caractéristiques classificateurs de bas, intermédiaire et haut niveau, sont extraites à partir d'un ensemble de couches.

ResNet résout principalement deux problèmes clés généralement rencontrés lors de l'entraînement des modèles d'apprentissage profond : la disparition et l'explosion des gradients [236], [250]. L'idée centrale de ResNet est d'introduire la "connexion de raccourci d'identité" qui saute une ou plusieurs couches. Figure 4.3[gauche] illustre des couches simples où chaque convolution est connectée les unes aux autres, ce qui était le cas de toutes les architectures postérieures au ResNet, tandis que la figure 4.3 [droite] affiche le réseau résiduel (saut de connexion). Sauter des couches, permet d'éviter la disparition de gradient lors de l'étape de rétropropagation. En utilisant de telles connexions, cette opération donne lieu à des réseaux extrêmement profonds et permet donc de capturer des motifs complexes dans les données.

Comme illustré sur la figure 4.4, l'architecture ResNet-50 [226] utilisée dans cette étude est composée de 5 blocs, où chacun d'entre eux contient une série de couches de convolutions, max-pooling, suivi d'un saut de connexion.

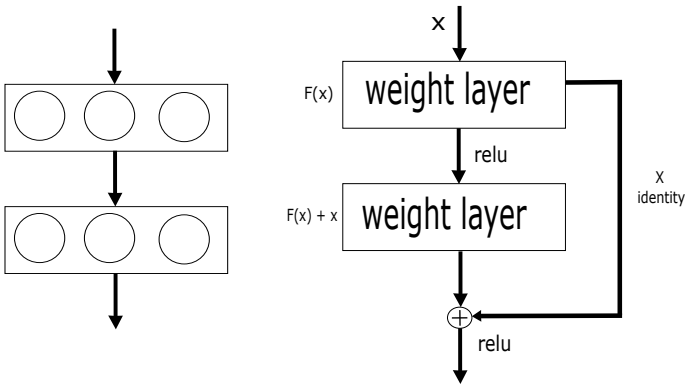


Figure 4.3: Couches simples (à gauche) et blocs résiduels avec connexion sautée (à droite) .

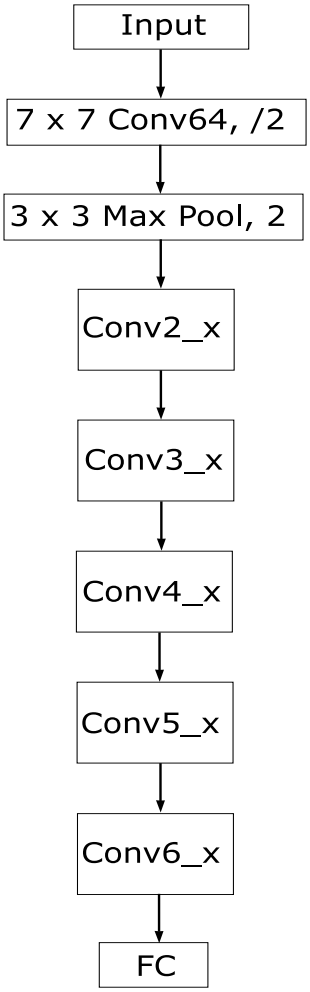


Figure 4.4: Aperçu de l'architecture ResNet-50.

4.3.2.2 LSTM

Les *LSTM* sont utilisés ici compte tenu de leur capacité à apprendre la dépendance à long terme des données de séries chronologiques et à résoudre le problème principal posé par la disparition du gradient sur l'axe du temps. Mélanger les dépendances à long et à court terme dans les images radiographiques du thorax semble être important pour les dépendances spatiales analysées par notre *CNN*. Les *LSTM* ont une dynamique qui leur permet de "mémoriser" facilement des informations pour un nombre étendu de pas de temps. La mémoire à long terme est stockée dans un vecteur de cellules mémoire {C_1, C_2,...C_n }.
Les propriétés de base de *LSTM* sont : décider à écraser la cellule mémoire, la récupérer ou la conserver pour le prochain pas de temps. Une cellule *LSTM* typique est illustrée dans la figure 4.5.

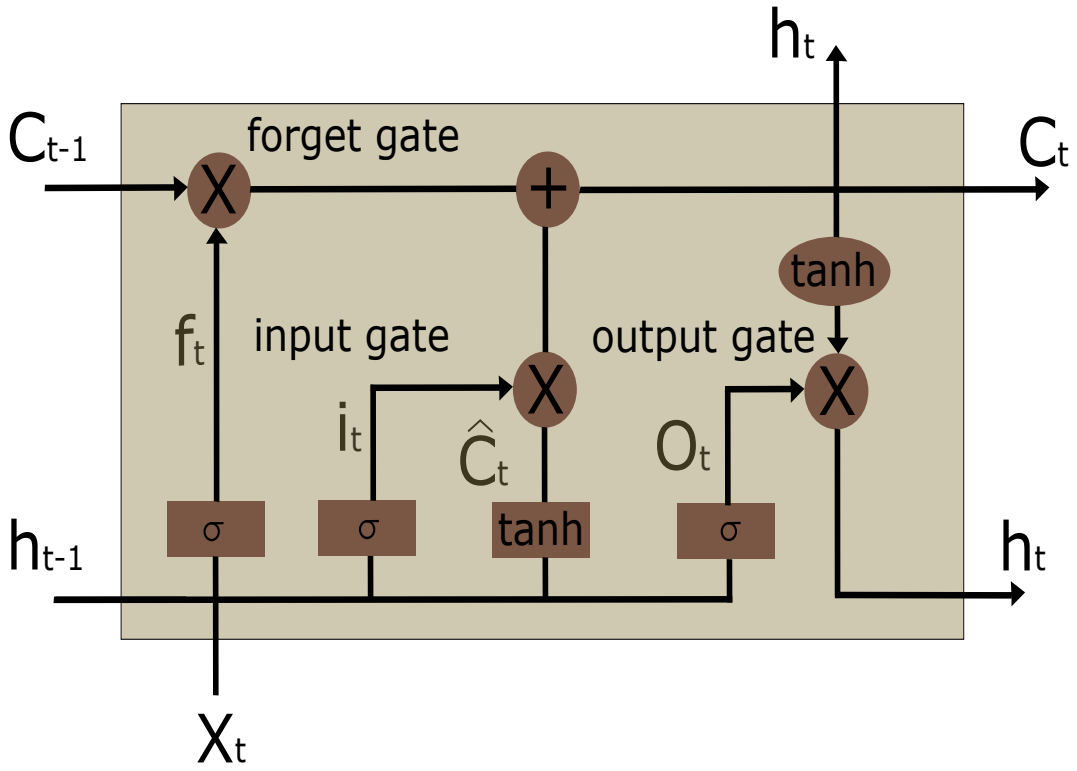


Figure 4.5: Architecture détaillée d'une cellule LSTM.

Dans la figure 4.5, l'entrée de la cellule est notée X_t, tandis que h_{t-1} représente l'état caché à l'étape précédente, et C_{t-1} correspond à la cellule mémoire précédente. Les sorties de la cellule sont le prochain état caché h_t et la cellule mémoire C_t.

L'architecture *LSTM* diffère des réseaux de neurones récurrents standards par deux

points majeurs au niveau des blocs qui les constituent. Premièrement, l'élément le plus crucial est l'état de la cellule qui contient l'information à mémoriser et à transmettre aux cellules subséquentes. Elle est divisée en deux parties, les états à long terme C_t et à court terme h_t. Deuxièmement, trois portes de contrôle le long du chemin d'état (forget, input, et output gates) sont ajoutées pour réguler et traiter les états de la cellule.

La première étape d'une cellule *LSTM* consiste à déterminer la quantité d'informations à supprimer depuis le vecteur d'entrée X_t et de la sortie précédente h_{t-1}. Cette décision est toujours prise par une couche *sigmoïde* ($\sigma(x) = \frac{1}{1+e^{-x}}$) appelée "forget gate layer (f_t)". Elle évalue les quantités h_{t-1} et X_t afin de produire pour chaque état de cellule C_{t-1} des valeurs comprises entre 0, c'est-à-dire ignorer la valeur, et 1 pour la conserver.

L'équation (4.3) montre comment contrôler la suppression des informations de l'état à long terme précédent C_{t-1} :

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f), \tag{4.3}$$

où σ est la fonction *sigmoïde*, W_f et b_f correspondent à la matrice de poids et au biais, respectivement.

L'étape suivante, input gate, consiste à générer les nouvelles informations qui doivent être mémorisées dans l'état de la cellule et à les transmettre à l'état suivant. Pour cela, il faut procéder par deux étapes. Premièrement, une couche *sigmoïde* appelée "input gate layer i_t" (Eq.(4.4)) qui décide des valeurs dont on a besoin pour les mettre à jour. Deuxièmement, une couche tangente hyperbolique ($tanh(x) = \frac{e^{-x}-e^{-x}}{e^{-x}+e^{-x}}$) vise à créer un vecteur de nouvelles valeurs candidates $\widehat{C}_t$ (Eq.(4.5)), qui seront ajoutées à l'état. Ces éléments sont ensuite combinés pour créer une mise à jour d'état :

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i), \tag{4.4}$$

$$\widehat{C}_t = \tanh(W_c[h_{t-1}, X_t] + b_c). \tag{4.5}$$

Comme dans l'équation (3.1), W_i et W_c représentent les matrices de poids, tandis que

b_i et b_c sont les termes de biais.

Dans une cellule *LSTM*, l'ancien état de la cellule C_{t-1} doit alors être mis à jour dans le nouvel état de la cellule C_t suivant l'équation (4.6) :

$$C_t = f_t.C_{t-1} + i_t.\widehat{C}_t, \tag{4.6}$$

où "." est la multiplication matricielle ponctuelle.
La dernière étape, output gate, est susceptible de prendre la décision de l'information finale produite à la sortie de la cellule. Le calcul de cette dernière porte o_t (Eq. (4.7)) dépend de l'état de la cellule, tandis que l'état caché est mis à jour selon l'équation (4.8) :

$$o_t = \sigma(W_o[h_{t-1}, X_t] + b_o). \tag{4.7}$$

$$h_t = o_t \tanh C_t. \tag{4.8}$$

Dans (4.7), W_o et b_o correspondent à la matrice de poids et au biais. La sortie du bloc est connectée de manière récurrente à l'entrée du bloc et à toutes les portes.

4.4 Validation Expérimentale

Des expériences sont menées sur une base de données ouverte de cas de COVID-19 avec des radiographies thoraciques [2] pour valider l'architecture proposée pour le pronostic COVID-19. Cet ensemble de données contenant des acquisitions temporelles avec une annotation de vérité terrain selon deux classes : décès ou survie pour chaque patient.

Le nombre limité de données disponibles, ainsi que le fait de ne considérer qu'une seule base de données, pourraient introduire des biais que nous ne sommes pas en mesure d'évaluer. Cette difficulté a été soulignée dans un certain nombre de travaux récents

[2] https://github.com/ieee8023/covid-chestxray-dataset/

[251], certains travaux sont également liés au diagnostic par rayons X du Covid-19 [252].

Pour la validation de l'architecture proposée, nous avons utilisé 51 séquences pour l'entraînement, comprenant chacune 3 images radiographiques d'un même patient, et 16 séquences pour le test.

Pour implémenter notre architecture ProgNet, nous avons mis en place quatre couches denses, FC-1024, 512, 128 et 64. Pour le codage, nous avons utilisé python avec les bibliothèques Keras et Tensorflow.

Pour évaluer les performances de la méthode ProgNet sur les données disponibles, nous avons effectué des comparaisons avec trois autres configurations possibles. Chaque configuration repose sur un *CNN* différent (voir figure 4.2) : AlexNet, VGG16 et VGG19. Ces réseaux ont déjà démontré des performances remarquables dans la littérature.

4.4.1 Comportement des courbes de perte et de précision

Nous avons utilisé une technique d'optimisation ADAM avec un taux d'apprentissage de 10^{-3}. En ce qui concerne la profondeur du *LSTM* utilisé, nous avons défini 64 unités cachées et ajouté une couche de dropout de $0,5$.

La figure 4.6 affiche les courbes de précision et de perte lors de la phase d'entraînement, obtenues pour le modèle proposé (Figure 4.6[(d)]) ainsi que les autres configurations utilisées : *AlexNet* (Figure 4.6[(a)s]), *VGG16* (Figure 4.6[(b)]), *VGG19* (Figure 4.6[(c)]). Bien que les courbes montrent des performances de convergence similaires entre les différentes architectures, notre approche présente une stabilité accrue, ce qui est illustré par la régularité de nos courbes par rapport à la fluctuation des courbes des autres architectures.

En ce qui concerne l'ensemble de test, la figure 4.7 affiche les courbes de précision et de perte caractérisant toutes les architectures. Lors de l'inspection visuelle de la figure 4.7[(d)], nous constatons que les courbes de perte et de précision s'améliorent plus rapidement pour l'architecture *ProgNet*. L'entraînement avec l'architecture *ProgNet* est

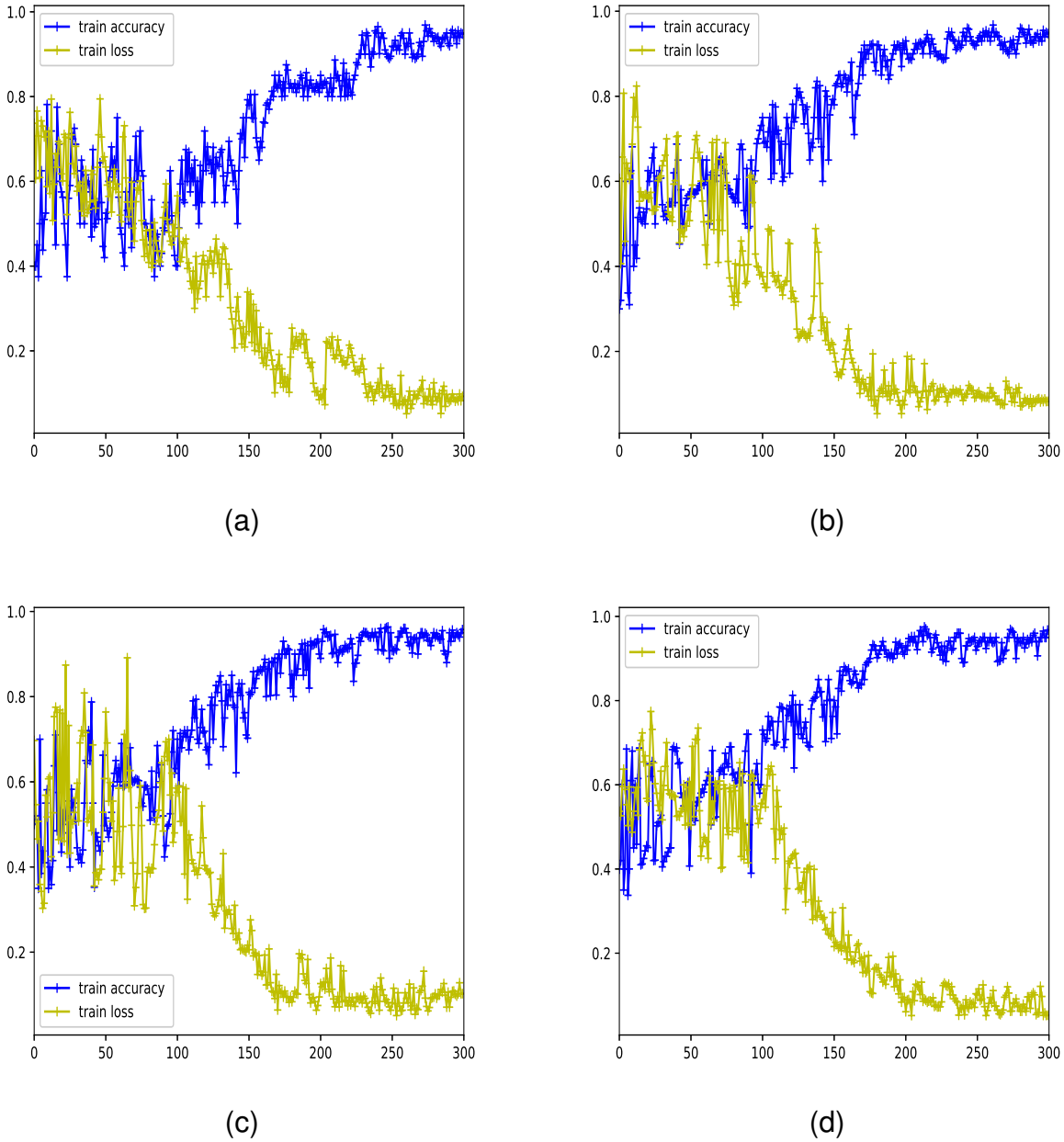


Figure 4.6: Précision/Perte des courbes d'entraînement obtenues avec (a) AlexNet, (b) VGG16, (c) VGG19 et (d) ProgNet.

donc plus rapide et plus efficace que les autres modèles.

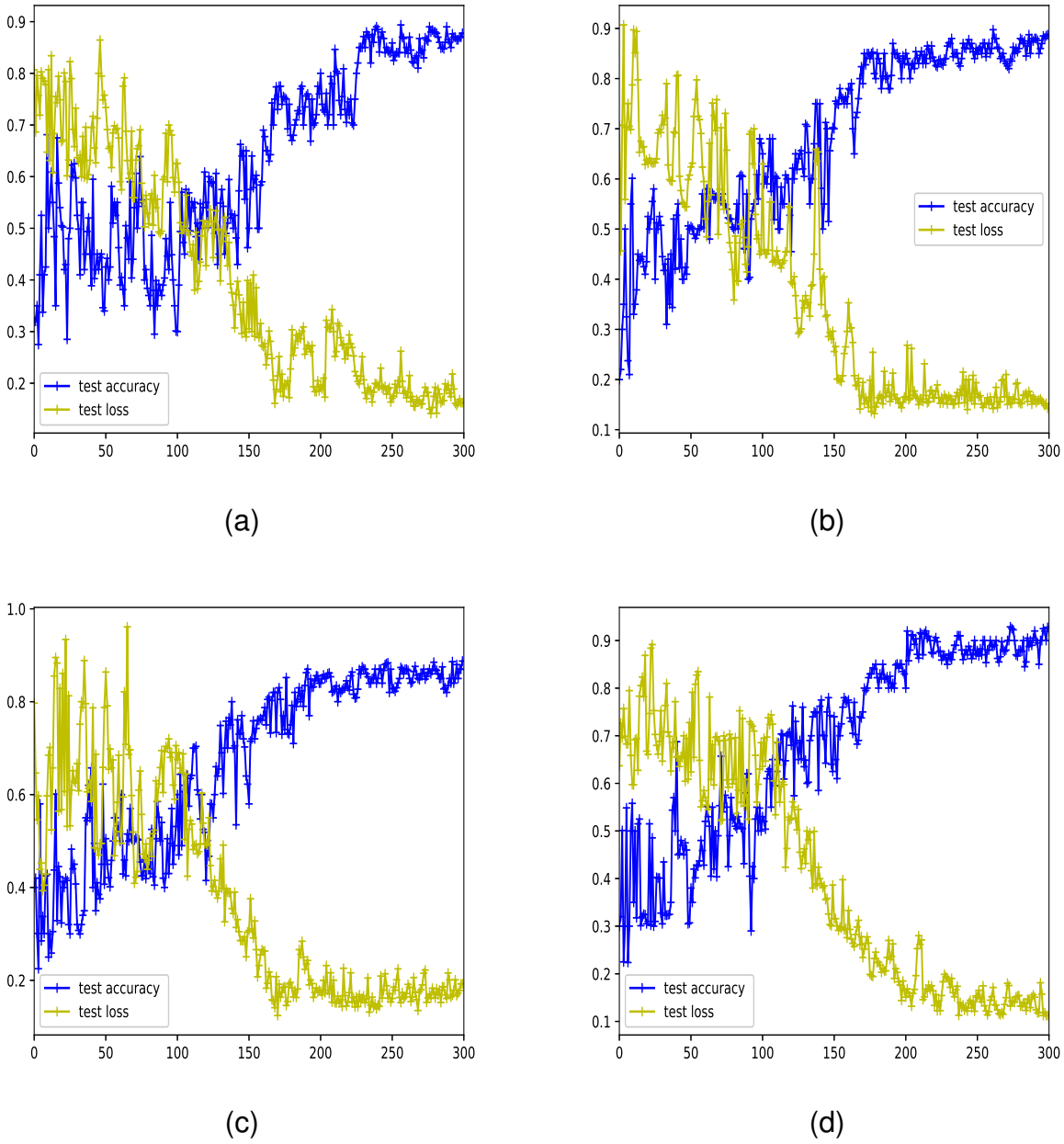


Figure 4.7: Précision/Perte des courbes de test obtenues avec (a) AlexNet, (b) VGG16, (c) VGG19 et (d) ProgNet.

Il convient de noter que les pics observés dans les courbes de précision et de perte peuvent être causés par un manque des données d'entraînement et de test ce qu'engendre donc une difficulté à apprendre certaines caractéristiques fines. Cela mérite plus d'investigation pour évaluer la capacité du modèle à être généraliser [253], [254].

4.4.2 Évaluation quantitative

Le tableau 4.1 rapporte les valeurs pour les taux de vrais positifs (VP), vrais négatifs (VN), faux positifs (FP) et faux négatifs (FN). Comme indiqué dans le tableau 4.1, tous les réseaux fonctionnent bien avec des taux de TP supérieurs à 86%. Cependant, l'architecture ProgNet proposée surpasse légèrement les autres avec les taux VP et VN les plus élevés, tandis que les taux FN et FP sont également les plus bas. En bénéficiant du FN le plus bas, l'architecture proposée apparaît très intéressante puisque pour un tel pronostic, il est crucial de ne pas rater les cas d'évolution négative.

Table 4.1: Valeurs VP, FN, FP et VN pour la méthode ProgNet proposée et les architectures AlexNet, VGG16 et VGG19.

	VP	**FN**	**FP**	**VN**
AlexNet	86.25%	10.93%	10.48%	86.13%
VGG16	88.11%	9.57%	10.02%	88.94%
VGG19	89.78%	8.44%	7.66%	89.32%
ProgNet	**92.51%**	**6.15%**	**6.09%**	**92.32%**

Pour évaluer davantage les performances quantitatives, l'accuracy, la précision et le rappel sont également fournis dans le tableau 4.2, en plus du score F_1 [255] et le temps de calcul. Le score F_1 tient compte à la fois de la précision et du rappel afin de valider l'exactitude. Il s'agit de la moyenne harmonique des deux mesures et peut-être calculée comme suit

$$F_1 = 2 \times \frac{\text{precision} \times \text{rappel}}{\text{precision} + \text{rappel}}. \tag{4.9}$$

Dans le tableau 4.2, les valeurs moyennes sur 10 exécutions sont fournies. Pour chaque exécution, une sous-partie de données d'entraînement est sélectionnée de manière aléatoire. Les écarts-types sur les 10 essais sont également fournis dans le tableau.

À travers les valeurs rapportées, on peut facilement remarquer que la méthode ProgNet proposée surpasse globalement les autres architectures concurrentes. Plus précisément, des valeurs de précision et de rappel plus élevées indiquent que *ProgNet* est

Table 4.2: Exactitude, Précision, Rappel, F_1-Score et le temps de calcul (en minutes) pour l'architecture ProgNet proposée ainsi que AlexNet, VGG16 et VGG19.

	AlexNet	**VGG16**	**VGG19**	**ProgNet**
Exactitude	0.86 ± 0.06	0.88 ± 0.03	0.89 ± 0.04	**0.92** ± 0.03
Précision	0.84 ± 0.08	0.88 ± 0.02	0.88 ± 0.05	**0.92** ± 0.04
Rappel	0.81 ± 0.08	0.86 ± 0.05	0.87 ± 0.07	**0.91** ± 0.05
F_1**-score**	0.82 ± 0.09	0.86 ± 0.05	0.88 ± 0.04	**0.91** ± 0.04
Temps (min)	83.2 ± 0.05	135.8 ± 0.04	147.6 ± 0.05	**218.2** ± 0.05

plus efficace pour identifier les cas d'infection ambigus. De plus, les faibles valeurs de variation standard rapportées montrent une meilleure stabilité pour le modèle proposé, ce qui mène aux meilleures propriétés de généralisation. Malgré les performances atteintes par notre modèle ProgNet, il est à noter que ce modèle nécessite un temps de convergence le plus long par rapport aux autres modèles.
Une inspection visuelle des séries chronologiques ambiguës est fournie dans le paragraphe 4.4.3.

4.4.3 Analyse qualitative

Dans ce paragraphe, nous illustrons quelques résultats représentatifs obtenus à partir d'un groupe de patients. Ces séries temporelles sont constituées de 3 images acquises aux instants T_1, T_2 et T_3.
La figure 4.8 affiche quatre séries chronologiques de patients Covid-19, deux en haut ont survécu, tandis que les deux en bas sont décédés. Pour chaque patient, le "score de survie" (SS) obtenu est fourni sur la première colonne. Ce score n'est rien d'autre que la probabilité prédite par la fonction sigmoïde.

Visuellement, il est clair que les séries chronologiques avec risque de décès montrent plus de zones blanches sur toutes les images. De manière générale, l'étendue de ces zones augmente avec le temps. Les valeurs SS rapportées indiquent dans quelle mesure le modèle ProgNet est confiant dans les résultats de la classification pour ces séries chronologiques.

Comme indiqué dans le paragraphe précédent, de meilleures valeurs de précision et de rappel confirment la capacité de notre architecture ProgNet à classer correctement les cas ambigus. En ce sens, la figure 4.9 montre deux exemples de séries chronologiques liées à deux patients présentant des problèmes de survie et de décès. Ces patients sont

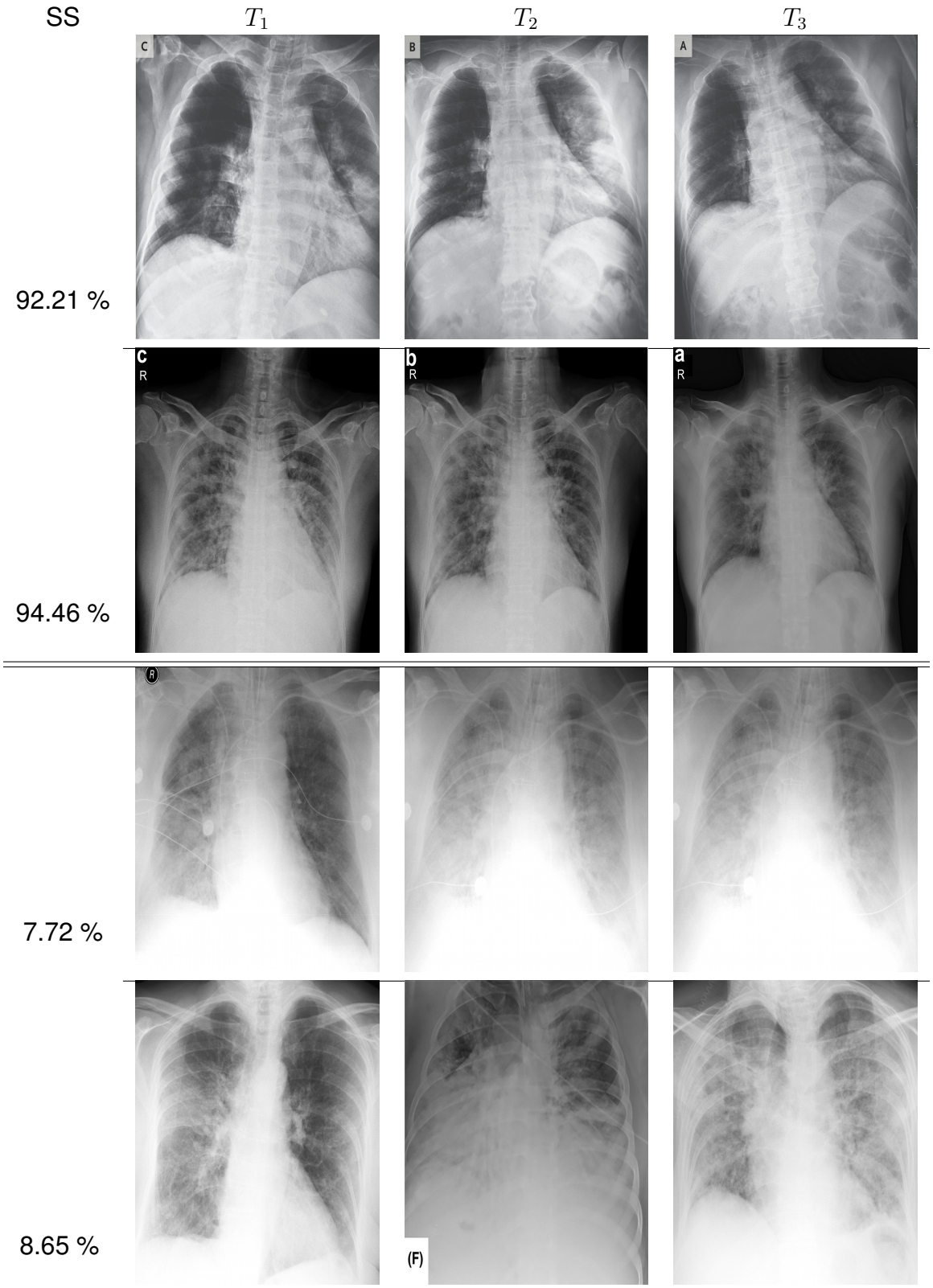


Figure 4.8: Visualisation de la classification COVID-19 à l'aide de l'architecture ProgNet proposée.

bien classés par l'architecture *ProgNet* proposée, contrairement aux autres modèles. Lors de l'inspection visuelle des images pour le problème de cas décédé, l'ambiguïté provient de zones blanches étendues dans toutes les images sans amélioration concrète au fil du temps. La même remarque vaut pour le cas de survie où les zones blanches persistent encore dans le temps.

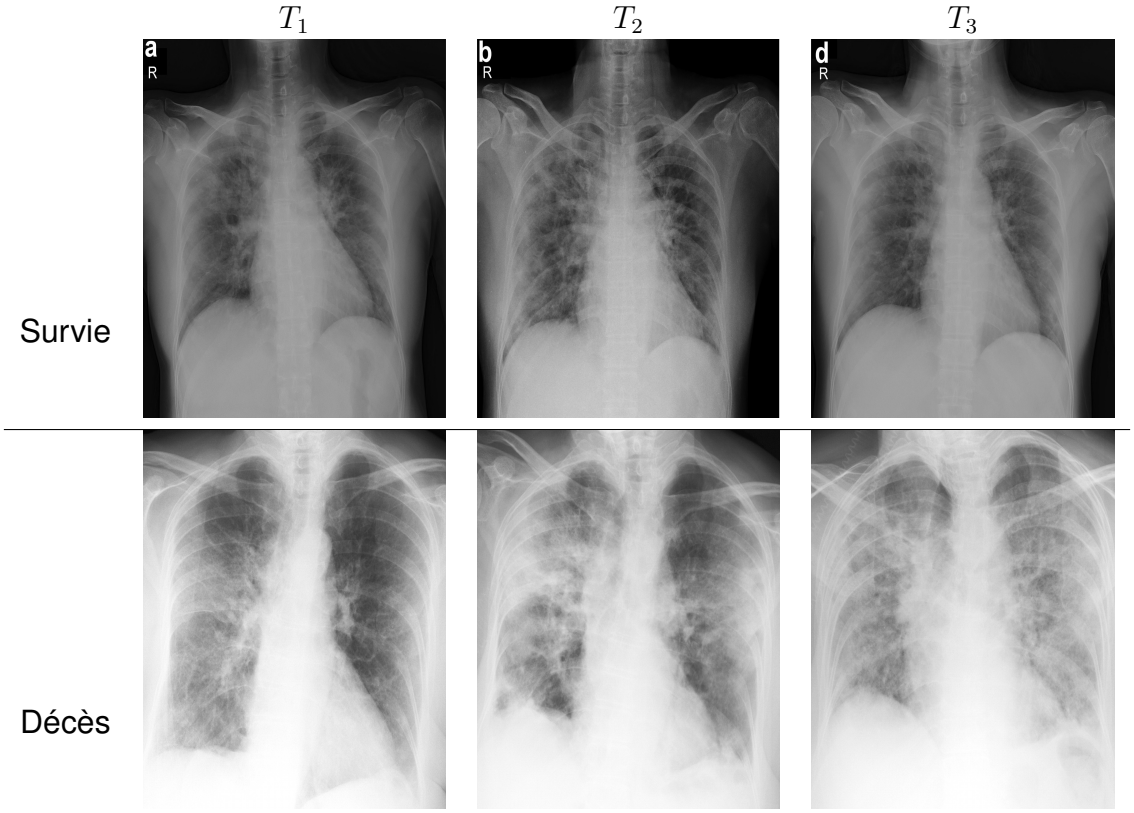


Figure 4.9: Exemples de séries chronologiques bien classées par ProgNet pour lesquelles les méthodes concurrentes échouent.

Pour étudier la raison pour laquelle l'architecture ProgNet proposée ne parvient pas à classer certains cas, la figure 4.10 montre des images d'une série chronologique liée à un patient mal classifié par ProgNet. En effet, on remarque dans certains cas un changement inattendu de la situation d'un patient lors de la dernière image de la séquence. Cela peut entraîner des problèmes d'apprentissage de toutes les fonctionnalités et de gestion de la corrélation temporelle entre elles. L'analyse de séquences plus longues peut certainement améliorer les performances du modèle.

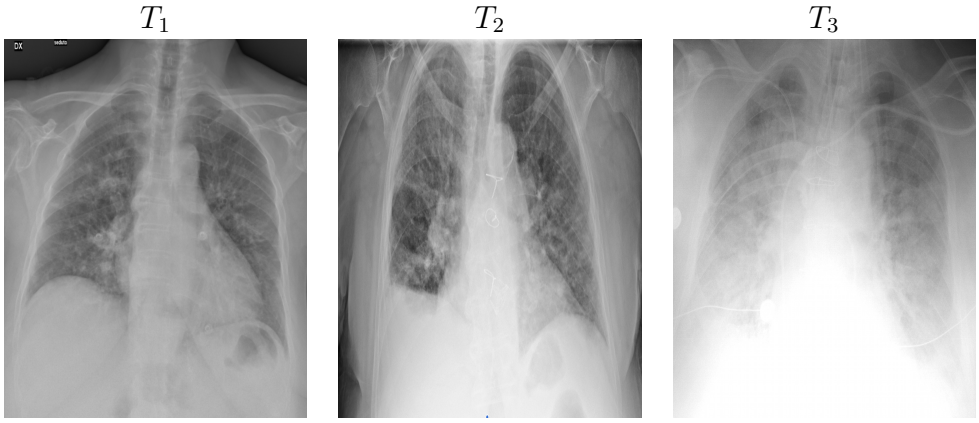


Figure 4.10: Images d'une série chronologique ambiguë mal classée par ProgNet.

4.4.4 Discussion

L'approche que nous avons proposée, qui examine l'aspect temporel dans l'apprentissage profond, est l'un des rares travaux dans le domaine médical. Les *LSTM* sont des modèles efficaces et évolutifs pour résoudre de nombreux problèmes d'apprentissage liés aux données séquentielles; ils et ont été utilisés avec succès dans de nombreux domaines. Dans notre étude, nous avons démontré comment combiner le *CNN* et *LSTM* pour la classification temporelle de l'évolution du COVID-19. Nous avons également comparé les performances de notre approche à celles des trois autres architectures, AlexNet, VGG16 et VGG19.

Les résultats des tableaux 4.1 et 4.2 ont clairement démontré que notre modèle proposé, ProgNet, atteint une précision globale élevée par rapport aux autres architectures concurrentes. Il est donc possible d'affirmer que ProgNet représente une approche d'apprentissage profond plus flexible, permettant d'obtenir une grande précision dans la dimension temporelle.

De plus, la figure 4.8 a montré que notre méthode de prédiction présente la qualité de pouvoir classer avec précision une séquence temporelle donnée en "Survie" et "Mort", ce qui est une tâche difficile à réaliser. Les résultats obtenus ont démontré la fiabilité de notre méthode et sa capacité à être utilisée dans des applications réelles, telles que la prédiction de la survie des patients atteints de maladies graves. Cependant, il est important de noter que Prognet n'est pas infaillible, dans la mesure où l'approche échoue lors de la classification de certaines séquences assez complexes.

Notre étude présente néanmoins une limitation qui doit être prise en compte. Le temps de calcul de notre approche ProgNet reste élevé, en raison du grand nombre de paramètres de notre architecture profonde et de notre processus d'optimisation basé sur la descente de gradient. Bien que cette méthode soit efficace pour minimiser la fonction de coût et améliorer les performances du modèle, elle peut prendre beaucoup de temps pour converger vers la solution optimale.

Dans les chapitres 5 et 6, nous proposons deux approches pour améliorer le traitement des séries temporelles d'images en apprentissage profond, en se concentrant sur l'optimisation et la fonction d'activation.

4.5 Conclusion

Nous avons proposé une architecture pour le pronostic Covid-19, basée sur une combinaison de réseaux *CNN* et *RNN*. L'architecture proposée analyse à la fois les dépendances spatiales et temporelles dans les séries chronologiques d'images radiographiques thoraciques.

Appliquée sur des données radiographiques, l'architecture ProgNet proposée a abouti à des résultats avantageux avec certaines efficiences caractérisant ses performances de classification, en particulier pour les cas ambigus.

5

Optimisation bayésienne non lisse pour les réseaux de neurones artificiels

Sommaire

5.1 Introduction

Les *CNN* ont été développés pour apprendre de manière automatique les hiérarchies d'entités spatiales en utilisant divers blocs de construction, notamment les couches de convolution, les couches de regroupement et les couches entièrement connectées, en utilisant la méthode de rétropropagation [64]. Cependant, entraîner un *CNN* est une tâche difficile, en particulier pour une architecture profonde contenant un grand nombre de paramètres (poids du modèle) à estimer. Ainsi, l'utilisation d'algorithmes sophistiqués d'optimisation est nécessaire.

Au fil des ans, de nombreux algorithmes d'optimisation ont été proposés, mais leur performance dépend largement de la convexité et de la différentiabilité de la fonction de perte cible. Par conséquent, choisir une stratégie d'optimisation qui cherche à trouver les optima globaux lors de l'apprentissage est généralement difficile, et une technique inappropriée peut conduire le réseau à un minimum local lors de la phase d'entraînement. De plus, accélérer le processus d'optimisation pour les grandes bases de données est un défi complexe, et le grand nombre de poids du réseau entraîne souvent un coût élevé en termes de mémoire et de ressources de calcul. Pour résoudre ce problème, plusieurs stratégies de réduction du nombre de poids du réseau (parcimonie des poids) ont été proposées, telles que les techniques basées sur l'optimisation parcimonieuse.

Dans ce chapitre, nous présentons une approche d'optimisation basée sur la chaîne de Markov Monte-Carlo (Markov Chain Monte-Carlo - *MCMC*) formulée dans le cadre bayésien. Plus précisément, nous présentons une approche basée sur la méthode hamiltonienne non lisse. Cette approche permet de surmonter les limitations des techniques d'optimisation classiques en permettant une exploration plus efficace de l'espace des paramètres du modèle.

5.2 Chaîne de Markov Monte-Carlo

5.2.1 Processus de la chaîne de Markov

Les algorithmes *MCMC* (Monte Carlo à chaîne de Markov) ont pour objectif de générer une chaîne d'échantillons dans l'espace des paramètres. Chaque point de la chaîne

est obtenu en transitionnant depuis le point précédent en suivant des règles spécifiques qui assurent la convergence de la chaîne vers la densité *a posteriori* souhaitée. Si la chaîne est suffisamment longue, les échantillons seront répartis selon la densité *a posteriori* recherchée. L'un des avantages clés des algorithmes *MCMC* est leur capacité à marginaliser naturellement chaque paramètre par rapport aux autres, même dans des espaces à plusieurs dimensions. Cette propriété permet de faire des estimations statistiques sur chaque paramètre individuellement en effectuant une intégration sur les distributions unidimensionnelles obtenues à partir de la chaîne d'échantillons.
Formellement, une chaîne de Markov est un processus stochastique X défini comme suit :

$$X = \{X_t : t \in T\} \tag{5.1}$$

$$X_t : \Omega \to S, \tag{5.2}$$

où t est un indice de temps et chaque X_t une variable aléatoire prenant ses valeurs dans un espace d'état S. T est généralement discret et se résume aux nombres naturels. Cependant S peut être discret ou continu. La chaîne est Markovienne si à n'importe quelle étape du processus, la valeur de l'échantillon suivant ne dépend pas des valeurs passées de la chaîne mais seulement de la valeur actuelle. Mathématiquement, la propriété de Markov peut être formalisée comme suit :

$$P(X_{t+1} \in A | X_0 = x_0, ..., X_t = x_t) = P(X_{t+1} \in A | X_t = x_t), \tag{5.3}$$

pour tout ensemble mesurable $A \subset S$. Si ces probabilités sont indépendantes du temps t, la chaîne est dite homogène. Ainsi une chaîne homogène aura un comportement constant dans le temps.

En pratique, les algorithmes *MCMC* utilisent diverses stratégies qui décrit les probabilités de transition d'un état à l'autre de la chaîne de Markov. La méthode *MCMC* la plus couramment utilisée est la méthode de Metropolis-Hastings.

5.2.2 Algorithme de Metropolis-Hastings

L'algorithme Metropolis-Hastings a été introduit pour la première fois par Metropolis, Rosenbluth et al. en 1953 [256] pour étudier l'équation d'état des molécules en interaction. L'algorithme a ensuite été généralisé par Hastings dans [257] en 1970 et renommé Metropolis-Hastings (*MH*).

Pour générer la chaîne de Markov, l'algorithme utilise une fonction appelée la fonction de proposition de transition $Q(.|X_t)$. Cette fonction de densité de probabilité propose un point dans l'espace des paramètres étant donné l'état actuel de la chaîne X_t. C'est à l'utilisateur de définir la forme de $Q(.|X_t)$. Dans tous les cas, proposer un point avec un tirage au sort conduit à un comportement de marche aléatoire de la chaîne.

L'algorithme suivant résume les détails de l'algorithme MH générique.

Algorithm 1: algorithme Métrololis-Hastings.

- Initialiser $X_0 \sim Q(.)$;
for $i = 1, 2, \ldots, N$ **do**
  - Proposer : $X_* \sim Q(.|X_t)$;
  - Probabilité d'acceptance : ;
    $\alpha(X_*|X_t) = min\Big\{1, \frac{Q(X_t|X_*)\pi(X_*)}{Q(X_*|X_t)\pi(X_t)}\Big\}$;
  - $u \sim uniform(u; 0, 1)$;
  if $u < \alpha$ **then**
    Accepter la proposition: $X_{t+1} \leftarrow X_*$;
  else
    Rejeter la proposition: $X_{t+1} \leftarrow X_t$;

Dans l'algorithme 1, N représente le nombre total d'itérations de l'algorithme, c'est-à-dire le nombre d'échantillons souhaité dans la chaîne de Markov.

Le nouveau candidat X_* est accepté à travers l'équation suivante,

$$\alpha(X_*|X_t) = min\Big\{1, \frac{Q(X_t|X_*)\pi(X_*)}{Q(X_*|X_t)\pi(X_t)}\Big\} \tag{5.4}$$

La première étape consiste à initialiser la valeur d'échantillon de chaque variable aléatoire. Cette valeur est souvent échantillonnée à partir de la distribution a priori

de la variable. La boucle principale de l'algorithme ci-dessus se compose de trois composants :

- Générer un échantillon X_* à partir de la distribution de proposition $Q(.|X_t)$.
- Calculer la probabilité d'acceptation avec la fonction d'acceptation $\alpha(X_*|X_t)$ basée sur la distribution de proposition et de la densité totale $\pi(.)$.
- Accepter ou rejeter l'échantillon candidat.

Les méthodes de chaîne de Markov Monte-Carlo que nous avons examinées peuvent être appliquées à une grande variété de problèmes, y compris l'optimisation parcimonieuse que nous allons maintenant aborder.

5.3 Optimisation parcimonieuse

Les réseaux de neurones profonds sont bien connus à être surparamétrés. Cependant, peu de poids du réseau sont généralement nécessaires pour apprendre avec précision les caractéristiques des données. Ils sont également connus pour avoir de nombreux paramètres redondants [258], [259]. En outre, le grand nombre de poids augmente considérablement le coût du stockage et les ressources informatiques. Plusieurs stratégies ont été proposées pour surmonter ces problèmes et limiter le nombre de paramètres du réseau (sparsification des poids) soit dans le modèle pré-entraîné, soit lors de l'étape d'apprentissage.

Pour favoriser la parcimonie des réseaux de neurones profonds (Deep Neural Network - *DNN*), trois grandes catégories de méthodes peuvent être identifiées: l'élagage ou "pruning", le dropout et les techniques basées sur l'optimisation parcimonieuse.

L'élagage consiste à supprimer les paramètres de poids qui sont indépendants de la performance, ce qui permet d'obtenir des réseaux plus compacts que les réseaux denses établis [260]. En outre, les travaux [261]–[263] ont mis en œuvre la fonction de perte hessienne pour supprimer les connexions du réseau. Dans un autre travail [264], les connexions ayant peu d'effet sur l'apprentissage sont supprimées pour obtenir des réseaux parcimonieux. Il existe également des méthodes sophistiquées qui utilisent divers critères pour déterminer les paramètres ou les connexions inutiles. [265]–[267]. Cependant, ces approches nécessitent une configuration manuelle de la sensibilité des

couches dans les critères d'élagage ainsi que des hypothèses heuristiques tout au long du processus d'élagage [268].

Le dropout réduit la taille des réseaux pendant l'entraînement en abandonnant au hasard des unités sur les connexions du *DNN* [130], [269], [270]. Le dropout biaisé et le dropout croisé [271] ont été proposés pour implémenter le dropout respectivement dans les unités cachées et les couches convolutives. Ces méthodes peuvent réduire efficacement le surapprentissage et améliorer les performances. Cependant, l'entraînement d'un réseau avec dropout prend généralement plus de temps que l'entraînement d'un réseau standard, même en appliquant la même architecture [130]. Aussi, le dropout ne peut que simplifier les réseaux pendant l'entraînement. L'étape de prédiction nécessite toujours un réseau de taille réelle.

Les techniques basées sur l'optimisation parcimonieuse introduisent une régularisation pour favoriser la parcimonie du réseau en mettant à zéro les paramètres redondants pendant le processus de l'entraînement. Par rapport à l'élagage, ce type d'approche ne repose pas sur une configuration manuelle. À l'inverse du dropout, les réseaux simplifiés obtenus par une stratégie d'optimisation parcimonieuse peuvent être utilisés à l'étape de la prédiction. De nombreuses méthodes compressent le réseau avec une perte de précision négligeable [272], mais les expériences indiquent que certaines approches basées sur l'optimisation parcimonieuse peuvent surpasser les performances du réseau d'origine [258], [273]–[275].

Bien que la parcimonie basée sur l'optimisation soit la classe la plus prometteuse, l'introduction de régularisateurs parcimonieux conduit généralement à des fonctions de coût non-différenciables. L'utilisation de techniques basées sur le gradient est donc sous optimale. De plus, les régularisations non convexes telles que la pseudo-norme ℓ_0 sont plus susceptibles de produire un modèle non biaisé avec des solutions plus parcimonieux. Cependant, à notre connaissance, il n'existe aucun travail antérieur proposant une technique flexible basée sur l'optimisation permettant de gérer des régularisateurs convexes et non convexes, avec des fonctions de coût non-différentiables.

Dans ce contexte, l'utilisation des techniques bayésiennes a fait d'énormes progrès dans diverses disciplines au fil des décennies conjugués par de nombreux avantages

pratiques. Par ailleurs, la flexibilité des modèles bayésiens permet d'introduire des contraintes sophistiquées, telles que celles liées à la parcimonie des réseaux. En ce qui concerne l'inférence, les techniques basées sur une approche*MCMC* peuvent également être adaptées aux grands problèmes de données [276].

Comme expliqué dans le paragraphe précédent, un cadre bayésien suppose que tous les paramètres sont des réalisations de variables aléatoires. Un estimateur des paramètres formulés par les distributions *a priori* et de vraisemblance est dérivé en utilisant un estimateur du maximum *a posteriori* (*MAP*). Cependant, la principale difficulté réside dans la dérivation d'expressions analytiques pour les estimateurs en raison de la complexité de la distribution *a posteriori* résultant de l'utilisation de *a priori* sophistiqués, notamment ceux qui favorisent la parcimonie. Dans ce cas, les techniques *MCMC* sont généralement utilisées pour échantillonner les coefficients de la cible *a posteriori* [277]. La principale limitation de ces techniques réside dans le niveau de complexité élevé, en particulier lorsque les données multidimensionnelles sont manipulées. Dans certains cas, des méthodes d'échantillonnage efficaces ont été proposées dans la littérature telle que les algorithmes de marche aléatoire Metropolis Hastings (random-walk Metropolis Hastings - *rw-MH*) [278] ou langevin ajusté en fonction de la métropole (Metropolis-adjusted Langevin algorithm - *MALA*) [279]. Récemment, l'échantillonnage utilisant la dynamique hamiltonienne [280] a été étudié en développant l'échantillonnage dit Monte Carlo hamiltonien (Hamiltonian Monte Carlo - *HMC*). Un algorithme plus sophistiqué a été proposé dans [281] appelé échantillonnage de Monte Carlo hamiltonien non-lisse (non-smooth Hamiltonian Monte Carlo - *ns-HMC*). Cette méthode résout le problème des schémas *HMC* qui ne peuvent pas être utilisés dans le cas de distributions exponentielles avec une fonction d'énergie non-différentiable.

Nous étudions l'exploitation de *ns-HMC* dans le processus d'entraînement des réseaux de neurones. Plus précisément, l'idée est d'implémenter une nouvelle méthode d'optimisation bayésienne pour minimiser la fonction de perte cible. La méthode proposée cible les schémas de régularisation favorisant les réseaux parcimonieux [282]. En effet, les méthodes du gradient dans ce cas ne sont pas très efficaces en raison de problèmes de dérivabilité et de convergence. Les performances d'apprentissage peuvent donc être altérées. Nous démontrons que l'utilisation de la méthode proposée conduit à des résultats de haute précision qui ne peuvent pas être atteints à l'aide d'optimiseurs concurrents.

L'apport de l'approche réside dans :

- La formulation bayésienne du problème d'optimisation pour les *DNN*. C'est une stratégie puissante pour trouver les extrema de fonctions objectives qui sont coûteuses à évaluer.
- La procédure d'optimisation flexible et efficace proposée, permet de résoudre le problème de non-différentiabilité et de gérer différents régularisateurs.
- La méthode proposée assure une convergence rapide vers le minimum global contrairement à d'autres techniques (par exemple celles basées sur le gradient).

5.4 Formulation du problème

Il est bien connu que l'optimisation des poids est l'une des étapes clés pour concevoir un modèle d'apprentissage efficace. Par exemple, si l'on considère un problème de classification, les poids, W, d'un *CNN* sont mis à jour en minimisant une erreur entre la vérité-terrain et les étiquettes estimées à l'aide du réseau. Une procédure itérative est généralement effectuée ainsi que des procédures d'optimisation basées sur le gradient sont utilisées. Dans un souci d'efficacité, une régularisation peut également être effectuée afin d'avoir une configuration des poids plus précise.

La méthode d'optimisation parcimonieuse peut être utilisée pour diverses tâches afin de produire des solutions parcimonieuses. La pénalité ℓ_1 ajoutée au coût de classification peut être interprétée comme une convexification de la pénalité ℓ_0. Dans [264], les poids avec la plus petite amplitude dans les réseaux pré-entraînés sont supprimés. La norme ℓ_0, qui compte le nombre d'éléments non nuls, est la forme la plus intuitive de régularisateurs parcimonieux. Il peut promouvoir la solution la plus parcimonieuse. Cependant, la minimisation du problème ℓ_0 est généralement NP-difficile [283]. La norme ℓ_1 est le substitut le plus couramment utilisé, qui peut être résolu facilement.

Lorsqu'il est appliqué dans les *DNN*, le régularisateur parcimonieux est censé mettre à zéro les poids redondants et supprimer par conséquent les connexions inutiles. Pour promouvoir des réseaux parcimonieux, des régularisations parcimonieuses doivent être mises en œuvre, ce qui rend l'utilisation d'algorithmes à base de gradient inefficace

puisque l'erreur à minimiser dans ce cas n'est plus différentiable.

Nous proposons une approche pour permettre l'optimisation des poids selon des régularisations non-lisses. Nous désignons par x une entrée à présenter au réseau de neurones artificiels. Le label (ou valeur numérique) estimé sera désigné par $\widehat{y}(x, W)$ comme une fonction non linéaire de l'entrée x et du vecteur de poids $W \in \mathbb{R}^N$, tandis que le label (ou valeur numérique) de vérité terrain sera noté y.

Le vecteur de poids peut être déterminé pendant la phase d'apprentissage en utilisant une méthode générique $\mathcal{L}$ (Euclidienne, Minkowski, etc.) basée sur l'erreur appliquée aux M données d'entrée :

$$\widehat{W} = \arg\min_{W} \sum_{m=1}^{M} \mathcal{L}(\widehat{y}(x^m; W) - y^m) + \lambda \|W\|_1, \tag{5.5}$$

où λ représente le paramètre de régularisation équilibrant une solution entre la fidélité des données et les termes de régularisation. Il convient de noter que d'autres termes de régularisation peuvent être utilisés dans l'équation (6.3) (ℓ_0, ℓ_p,...).

Puisque le problème d'optimisation dans l'équation (6.3) n'est pas différentiable, l'utilisation d'algorithmes basés sur le gradient avec rétropropagation n'est pas possible. Cela rend le processus d'apprentissage très complexe et coûteux.

Nous détaillons ensuite notre méthode pour estimer efficacement les poids sans augmenter la complexité d'apprentissage. Le problème d'optimisation dans l'équation (5.5) est formulé dans un cadre bayésien.

Cette formulation présente deux avantages principaux. Le premier est lié à la flexibilité de ces modèles pour gérer un large panel de termes de régularisation au travers une formulation exponentielle afin d'imiter la forme variationnelle dans l'équation (5.5). Le deuxième avantage est associé à la possibilité de concevoir des schémas entièrement automatiques sans intervention/configuration de l'utilisateur. En effet, ceci est très important, notamment pour les problèmes complexes où l'ajustement des paramètres est compliqué.

5.5 Optimisation bayésienne

Comme indiqué ci-dessus, le problème d'optimisation des poids est formulé dans un cadre bayésien. En ce sens, les paramètres et hyperparamètres du problème sont supposés suivre des distributions de probabilité. Plus précisément, une distribution de vraisemblance est définie pour modéliser le lien entre le vecteur des poids cibles et les données, tandis qu'une distribution a priori est définie pour modéliser les connaissances a priori sur les poids cibles.

5.5.1 Modèle bayésien hiérarchique

D'après le concept de minimisation de l'erreur entre le label (ou valeur numérique) de référence y et la valeur estimée $\widehat{y}$, et en supposant une erreur quadratique (premier terme de (6.3)), on définit la distribution de vraisemblance

$$f\left(\mathbf{y}, \mathbf{x}; W\right) \propto \prod_{m=1}^{M} \exp\left[-\mathcal{L}(\widehat{y}(x^m; W) - y^m)\right], \tag{5.6}$$

où $\mathbf{x} = \{x^1, \ldots, x^M\}$ et $\mathbf{y} = \{y^1, \ldots, y^M\}$.

En ce qui concerne l'information a priori sur les poids cibles, et pour favoriser la parcimonie du vecteur estimé (et donc la parcimonie du réseau profond), un choix courant est de recourir à une pénalisation ℓ_1. Dans un cadre bayésien, la distribution de Laplace peut être utilisée:

$$f(W; \lambda) \propto \exp\left(-\frac{\|W\|_1}{\lambda}\right), \tag{5.7}$$

où λ est un hyperparamètre à définir. Cet a priori nous permet d'introduire exactement la même information a priori que la norme ℓ_1 dans l'équation (6.3).
En adoptant une approche *MAP*, nous devons d'abord exprimer la distribution *a posteriori*. Sur la base de la vraisemblance et de l'*a priori* défini, cette a posteriori s'écrit:

$$
\begin{aligned}
f(W;\mathbf{y},\mathbf{x},\sigma,\lambda) &\propto f(\mathbf{y},\mathbf{x};W)f(W;\lambda) \\
&\propto \prod_{m=1}^{M} \exp\left(-\mathcal{L}(\widehat{y}(x^m;W)-y^m)\right) \times \exp\left(-\frac{\|W\|_1}{\lambda}\right).
\end{aligned} \tag{5.8}
$$

Il est clair que cette *a posteriori* n'est pas simple à manipuler pour obtenir une expression directe de l'estimation $\widehat{W}$. Par conséquent, nous utilisons une méthode d'échantillonnage stochastique pour approximer numériquement la loi *a posteriori*, ce qui permet de calculer un estimateur pour $\widehat{W}$. Le paragraphe suivant détaille la procédure d'échantillonnage adoptée.

5.5.2 Monte Carlo Hamiltonien

Le Monte-Carlo hamiltonien [284] est une classe d'algorithmes d'échantillonnage inspirés de la dynamique hamiltonienne. C'est une reformulation de la théorie de la mécanique classique qui vise à décrire le mouvement des objets, et donc à modéliser les systèmes physiques dynamiques [285]. Une particule dynamique de masse m peut être caractérisée essentiellement par ses positions W et sa quantité de mouvement $q = mv$, où $v = \frac{\partial W}{\partial t}$ qui représente la vitesse de la particule. Le système hamiltonien modélise la cinétique totale de cette particule, respectivement l'énergie potentielle $E(W)$ et l'énergie cinétique $K(v) = \frac{1}{2}mv^2$, qui peuvent être formulés par rapport à la quantité de mouvement $K(q) = \frac{1}{2m}q^2$. Ainsi, l'hamiltonien $H(W,q)$ peut être exprimé par :

$$
H(W,q) = E(W) + K(q), \tag{5.9}
$$

et la dynamique de la particule peut-être spécifiée par un ensemble d'équations différentielles couplées [286],

$$
\frac{dq}{dt} = -\frac{\partial H}{\partial W}. \tag{5.10}
$$

$$
\frac{dW}{dt} = \frac{\partial H}{\partial q}. \tag{5.11}
$$

À partir de tout intervalle de temps de durée s, ces équations définissent une application

T_s, de l'état à tout instant t à l'état à l'instant $t + s$.

L'échantillonnage du modèle est effectué en deux étapes. Le premier échantillonnage q selon la distribution gaussienne multivariée $N(0,I_N)$, où I_N représente la matrice d'identité $N \times N$. La deuxième étape met à jour à la fois la quantité de mouvement q et la position W en proposant deux candidats W^* et q^*. La dynamique hamiltonienne est simulée pour générer ces deux candidats. Cette dynamique est discrétisée à l'aide de certaines techniques de discrétisation, telles que la méthode leapfrog [280]. La discrétisation peut être effectuée en utilisant les pas L_f de la méthode leapfrog avec une taille de pas $\epsilon > 0$: L_f peut être soit fixée manuellement, soit réglée automatiquement [287].

L'algorithme *HMC* est particulièrement utile pour simuler des échantillons corrélés et distribués selon des lois à haute dimensionnalité, telles que celles rencontrées en physique statistique, en traitement du signal et en apprentissage automatique.

5.5.3 Monte Carlo hamiltonien non-lisse

Comme nous avons expliqué dans le paragraphe précédent, l'algorithme Hamiltonian Monte Carlo est une méthode d'échantillonnage qui utilise la dynamique hamiltonienne pour explorer l'espace des paramètres de la distribution cible. L'idée est de simuler une trajectoire en utilisant les équations de mouvement de la dynamique hamiltonienne, ce qui permet de mieux explorer l'espace des paramètres que les méthodes d'échantillonnage plus simples.

La méthode *HMC* est seulement appropriée aux distributions de probabilité ayant des fonctions d'énergie lisses dont le gradient peut être calculé. Cette contrainte représentait une véritable limitation dans les applications où la parcimonie est une propriété clé, car les distributions de probabilité favorisant la parcimonie ont souvent une fonction d'énergie non différentiable. Cela signifie que dans les cas où la distribution de probabilité est caractérisée par une fonction d'énergie non différentiable, la méthode HMC peut ne pas être adaptée.

Pour résoudre ce problème, un nouveau schéma d'échantillonnage appelé Monte Carlo hamiltonien non-lisse (*ns-HMC*) a été proposé dans [281] afin de permettre l'utilisation de la dynamique d'échantillonnage hamiltonienne, même pour les distributions cibles avec des fonctions d'énergie non-lisses. La technique d'échantillonnage repose sur certains résultats intéressants de l'optimisation convexe et des méthodes de Monte-Carlo hamiltoniennes. L'idée principale du schéma *ns-HMC* est de modifier le schéma

de discrétisation leapfrog.

Pour une distribution ayant comme fonction d'énergie E_θ, le schéma *ns-HMC* détaillé est donné par l'algorithme 2, où L_f représente le nombre de pas de la discrétisation leapfrog et ϵ est la taille du pas.

Algorithm 2: Algorithme de ns-HMC [281]

- Initialiser avec quelques $\boldsymbol{W}^{(0,0)}$, définir le numéro d'itération $r = 0$, L_f and ϵ;
for $r = 1, \ldots, S$ **do**
  - Échantillonner $q^{(r,0)} \sim \mathcal{N}(\mathbf{0}, I_N)$;
  - Calculer $q^{(r,\frac{1}{2}\epsilon)} = q^{(r,0)} - \frac{\epsilon}{2}\left[W^{(r-1,0)} - prox_{E_\theta}(W^{(r-1,0)})\right]$;
  - Calculer $W^{(r,\epsilon)} = W^{(r-1,0)} + \epsilon q^{(r,\frac{1}{2}\epsilon)}$;
  for $l_f = 1$ *to* $L_f - 1$ **do**
    * Calculer $q^{(r,(l_f+\frac{1}{2})\epsilon)} = q^{(r,l_f\epsilon)} - \frac{\epsilon}{2}\left[W^{(r,l_f\epsilon)} - prox_{E_\theta}(W^{(r,l_f\epsilon)})\right]$;
    * Calculer $W^{(r,(l_f+1)\epsilon)} = W^{(r,l_f\epsilon)} + \epsilon q^{(r,(l_f+\frac{1}{2})\epsilon)}$;
  end
  - Calculer $q^{(r,(L_f+\frac{1}{2})\epsilon)} = q^{(r,L_f\epsilon)} - \frac{\epsilon}{2}\left[W^{(r,L_f\epsilon)} - prox_{E_\theta}(W^{(r,L_f\epsilon)})\right]$;
  - Appliquer la règle d'acceptation standard MH en prenant pour $q^* = q^{(r,\epsilon L_f)}$ and $W^* = W^{(r,\epsilon L_f)}$;
end

5.5.4 Échantillonnage hamiltonien

Nous utilisons la distance euclidienne comme mesure générique dans nos expériences. Notons $\alpha = \dfrac{\lambda}{\sigma^2}$, $\theta = \{\sigma^2, \lambda\}$.

Pour un vecteur de poids W, nous définissons la fonction d'énergie suivante

$$E_\theta(W) = \frac{\alpha}{2}\sum_{m=1}^{M} \|\widehat{y}(x^m; W) - y^m\|_2^2 + \|W\|_1. \tag{5.12}$$

La loi *a posteriori* dans (5.8) peut être donc reformulée par

$$f(W; y, x, \theta) \propto \exp\left(-E_\theta(W)\right). \tag{5.13}$$

Pour échantillonner selon cette *a posteriori* exponentielle, et puisque l'échantillonnage direct n'est pas possible compte tenu de la forme de la fonction E_θ, l'échantillonnage hamiltonien est adopté. En effet, la stratégie de la dynamique hamiltonienne [280] a

été largement utilisée dans la littérature pour échantillonner des vecteurs de grandes dimensions.

En ce qui concerne le calcul de l'opérateur proximal, nous notons $\varphi(W) = \|W\|_1$. En suivant la définition standard de l'opérateur proximal, nous pouvons écrire pour tout vecteur z

$$\begin{aligned} & prox_{E_\theta}(z) = p \Leftrightarrow z - p \in \partial E_\theta(p) \\ & \Leftrightarrow z - p \in \partial\varphi(p) + \frac{\alpha}{2} G_{\mathcal{L}}(p). \end{aligned} \tag{5.14}$$

En faisant l'hypothèse simplificatrice que le gradient $G_{\mathcal{L}}$ est constant tout au long de la trajectoire, il peut être calculé une seule fois (ou bien mis à jour périodiquement) à l'initialisation. Ce calcul peut être réalisé à l'aide d'une étape de rétropropagation. Nous substituons $G_{\mathcal{L}}(p)$ par $C = G_{\mathcal{L}}(W^0)$ dans l'expression de l'opérateur proximal (5.14) :

$$\begin{aligned} & z - \frac{\alpha}{2} G_{\mathcal{L}}(W^0) - p \in \partial\varphi(p) \\ & \Leftrightarrow p = prox_{\varphi}\left(z - \frac{\alpha}{2} G_{\mathcal{L}}(W^0)\right). \end{aligned} \tag{5.15}$$

Puisque prox_{φ} n'est rien d'autre que l'opérateur de seuillage doux [288], l'opérateur proximal dans (5.15) peut être facilement calculé une fois qu'un seul pas de gradient est appliqué (rétropropagation) pour calculer $G_{\mathcal{L}}(W^0)$.

L'échantillonneur de Métropolis-Hastings est résumé dans l'algorithme 3.

Les candidats proposés sont donnés par $q^* = q^{(r,\epsilon L_f)}$ et $W^* = W^{(r,\epsilon L_f)}$ après les L_f étapes de leapfrog. Ces candidats sont ensuite acceptés sur la base de la règle MH standard, c'est-à-dire avec la probabilité suivante

$$\min\{1, \exp[H(W^{(r)}, q^{(r)}) - H(W^{(*)}, q^{(*)})]\} \tag{5.16}$$

où H est l'opérateur hamiltonien défini dans (5.9). Après la convergence, l'algorithme 3 fournit des chaînes de coefficients échantillonnées selon la distribution cible de W. Ces chaînes peuvent être utilisées pour calculer un estimateur *EQMM* (Erreur Quadratique Moyenne Minimale) après avoir écarté les échantillons correspondant à la période de

Algorithm 3: Échantillonneur MH utilisé pour W.

- Initialiser avec W^0, calculer $C = G_{\mathcal{L}}(W^{(0,0)})$, définir le numéro d'itération $r = 0$, L_f et ϵ ;

for $r = 1, \ldots, S$ **do**

- Échantillonner $q^{(r,0)} \sim \mathcal{N}(\mathbf{0}, I_N)$;
- $q^{(r,\frac{1}{2}\epsilon)} = q^{(r,0)} - \frac{\epsilon}{2}\left[W^{(r-1,0)} - prox_{\varphi}\left(W^{(r-1,0)} - \frac{\alpha}{2}C\right)\right]$;
- $W^{(r,\epsilon)} = W^{(r-1,0)} + \epsilon q^{(r,\frac{1}{2}\epsilon)}$;

for $l_f = 1$ *to* $L_f - 1$ **do**

* $q^{(r,(l_f+\frac{1}{2})\epsilon)} = q^{(r,l_f\epsilon)} - \frac{\epsilon}{2}\left[W^{(r,l_f\epsilon)} - prox_{\varphi}\left(W^{(r,l_f\epsilon)} - \frac{\alpha}{2}C)\right)\right]$;
* $W^{(r,(l_f+1)\epsilon)} = W^{(r,l_f\epsilon)} + \epsilon q^{(r,(l_f+\frac{1}{2})\epsilon)}$;

end

- $q^{(r,(L_f+\frac{1}{2})\epsilon)} = q^{(r,L_f\epsilon)} - \frac{\epsilon}{2}\left[W^{(r,L_f\epsilon)} - prox_{\varphi}\left(W^{(r,L_f\epsilon)} - \frac{\alpha}{2}C\right)\right]$;
- Appliquer la règle d'acceptation/rejet MH à (W^*, q^*) avec $q^* = q^{(r,\epsilon L_f)}$ et $W^* = W^{(r,\epsilon L_f)}$;

end

Table 5.1: Détails des ensembles d'images utilisés : ensemble d'entraînement, de test et les classes.

Base	Ensemble d'entra.	Ensemble de test	# Classes
Images TDM - class. simple	1210	430	2
Images TDM - class. difficile	566	180	2
Fashion-MNIST	48000	12000	10
CIFAR-10	50000	10000	10

chauffe (*burn-in*).

5.6 Validation expérimentale

5.6.1 Classification transversale

Afin de valider la méthode proposée, cinq expériences de classification d'images sont menées à l'aide de quatre ensembles de données : deux ensembles de données COVID-19 comprenant des images *TDM* pour une classification simple [289] et difficile [290], et deux ensembles de données standard, à savoir *Fashion-MNIST* [291], *CIFAR-10* [292]. Le tableau 6.1 illustre les détails des paramètres des différents ensembles de données.

Trois types d'optimiseurs sont utilisés pour la comparaison : *i)* les méthodes basées

Table 5.2: Paramétrage des algorithmes de benchmark.

Algorithme	Paramètres	Description	Valeurs
MH	σ	écart-type de la dist. normale de la proposition	1
rw-MH	σ	écart-type de la dist. normale de la proposition	1
Adam	lr	Taux d'apprentissage	10^{-3}
	β_1	Taux de décr. exp. d'estimation au 1er moment	0.9
	β_2	Taux de décr. exp. d'estimation au 2e moment	0.999
	ϵ	Constante de stabilité numérique	$1e-08$
SGD	lr	Taux d'apprentissage	10^{-3}
	momentum	Valeur d'accélération	0.8
	decay	décr. du taux d'apprentissage à chaque màj	$1e-6$
Adadelta	lr	Taux d'apprentissage	10^{-3}
	rho	Taux de décomposition	0.95
	ϵ	Constante de stabilité numérique	$1e-08$
Adamax	lr	Taux d'apprentissage	10^{-3}
	β_1	Taux de décr. exp. d'estimation au 1er moment	0.9
	β_2	Taux de décr. exp. d'estimation au 2e moment	0.999
	ϵ	Constante de stabilité numérique	$1e-08$
IWT	minv	Borne inférieure	-2
	maxv	Limite supérieure	2
	size	Nombre de particules	30
	p_s	Paramètre spirale	3
DA	minv	Borne inférieure	-2
	maxv	Limite supérieure	2
	size	Nombre de particules	15
SSA	minv	Borne inférieure	-2
	maxv	Limite supérieure	2
	size	Nombre de particules	30

sur MCMC, précisément l'algorithme standard de Metropolis-Hastings (MH) [293] et sa variante de marche aléatoire (*rw-MH*), *ii)* les techniques les plus populaires et les plus largement utilisées : *Adam*, *Adamax*, *SGD*, *Adadelta* et *iii)* trois algorithmes métaheuristiques : *IWT*, *DA* et *SSA*. Le paramétrage de tous ces algorithmes est détaillé dans le tableau 5.2.

5.6.1.1 Modèles ConvNet

Trois architectures *CNN* sont utilisées dans cette partie. Comme le modèle LeNet [294], la première comprend trois couches convolutifs et deux couches *FC*. La deuxième a neuf couches convolutives et trois couches *FC* qui sont organisées de la même manière que VGG-Net [295]. Ces architectures sont présentées dans le tableau 5.2. La troisième

est un *CNN* plus profond utilisé à des fins de comparaison comprenant 25 couches convolutives (voir paragraphe (5.6.1.7). Toutes impliquent des couches convolutives avec 3×3 la taille du filtre en plus de 2×2 max-pooling, et un *stride size* égale à 1. Toutes les couches dans les différentes configurations utilisaient la fonction *ReLU*.
Comme les réseaux de neurones profonds peuvent facilement sur-adapter lorsqu'ils sont entraînés avec de petits ensembles de données, les *CNN* utilisés sont étendus avec trois techniques de régularisation :

- Normalization par lots [296] : traite du changement de la distribution de l'espace des caractéristiques avec le modèle pendant l'entraînement. L'entrée de la couche est normalisée pour être une moyenne nulle avec une variance unitaire. Cette étape agit non seulement comme un régularisateur, mais permet également un entraînement plus rapide, des taux d'apprentissage plus élevés et moins de dépendance à l'initialisation des poids.
- Régularisation ℓ_1 [297]: La régularisation ℓ_1 est le choix préféré lorsqu'on a un grand nombre de fonctionnalités car elle fournit des solutions parcimonieuses. Cela permet d'obtenir l'avantage de calcul car les caractéristiques avec des coefficients nuls peuvent être évitées. Dans notre cas, le paramètre de régularisation utilisé a été défini sur $\lambda = 0.001$.
- Dropout [130] : désactivation aléatoire des neurones pendant l'entraînement avec probabilité (ou pourcentage) p. Dans notre étude, le taux de dropout est fixé par validation croisée à $p = 0.3$.

Nous avons choisi d'utiliser trois d'architectures *CNN* principalement pour évaluer la performance et le comportement de notre méthode à divers niveaux de profondeur. En ce sens, l'entraînement avec des données volumineuses peut s'avérer coûteux. Cela exige un matériel considérable pour les calculs complexes.

5.6.1.2 Analyse de la parcimonie et de la stabilité

Dans cette section, nous évaluons la parcimonie et la stabilité de l'estimation des poids avec différentes valeurs de λ et comparons à *Adam* en tant qu'optimiseur de pointe. L'architecture CNN_1 est appliquée avec la base d'images *TDM* Covid-19.
Le tableau 5.4 rapporte la précision, la sensibilité, la spécificité, le temps de calcul et la norme ℓ_1 des poids estimés en utilisant différentes valeurs du paramètre de régularisation λ sur 10 exécutions de Monte Carlo. Pour évaluer davantage le niveau de parcimonie, le tableau 5.4 prévoit également les valeurs de pseudo-norme ℓ_0 (nombre

Table 5.3: ConvNet avec des techniques de régularisation.

CNN_1	CNN_2
Conv3x3-32:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.2)	3 X Conv3x3-32:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.3)
Conv3x3-64:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.3)	3 X Conv3x3-64:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.3)
Conv3x3-128:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.4)	3 X Conv3x3-128:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.4)
Flatten	Flatten
FC-64 Dropout(0.3)	FC-128 Dropout(0.35) FC-64 Dropout(0.35)
FC-softmax	FC-softmax

de non-zéros). Les écarts-types sur les 10 exécutions sont fournis dans le tableau.

Les scores obtenus indiquent clairement que notre méthode fournit des estimations avec un niveau de parcimonie plus élevé par rapport à *Adam* : le nombre de poids non nuls est significativement inférieur à l'optimiseur d'*Adam*. La méthode proposée fournit des estimations avec un niveau de parcimonie supérieur de 14%. De plus, les faibles valeurs d'écart type rapportées dans le tableau 5.4 illustrent clairement de bonnes propriétés de stabilité de notre optimiseur comparé à l'échantillonnage aléatoire de la procédure *MCMC*, ce qui confirme les bonnes propriétés de convergence. Cette stabilité vaut pour la précision, la parcimonie, la sensibilité, la spécificité et le temps de calcul. De plus, les mêmes conclusions valent pour toutes les valeurs λ testées. En d'autres termes, les mêmes conclusions sont valables pour différents niveaux de

parcimonie du réseau.

Table 5.4: Précision, sensibilité, spécificité, temps de calcul (en minutes), normes ℓ_0 et ℓ_1 des poids estimés pour CNN_1 en utilisant Adam et la méthode proposée avec différentes valeurs de λ.

Optimiseurs	λ	$\|\cdot\|_0$	$\|\cdot\|_1$	Précision	Temps	Sensibilité	spécificité
ns-HMC	10^{-3}	149113	48988	89.68	37.28	88.21	86.95
		± 9.07	± 9.10	± 0.04	± 0.63	± 0.06	± 0.08
	10^{-2}	**142542**	**45476**	**90.02**	**38.79**	**89.02**	**88.57**
		$\pm$8.78	**$\pm$8.81**	**$\pm$0.02**	**$\pm$0.61**	**$\pm$0.4**	**$\pm$0.3**
	10^{-1}	152904	51232	89.42	38.47	87.81	86.11
		± 9.11	± 9.19	± 0.05	± 0.70	± 0.5	± 0.6
Adam	10^{-3}	180513	51727	86.91	52.12	84.36	81.25
		± 9.77	± 9.11	± 0.07	± 0.87	± 0.9	± 0.8
	10^{-2}	191229	67732	85.34	54.91	83.14	80.66
		± 10.28	± 10.63	± 0.12	± 1.08	± 1.12	± 1.09
	10^{-1}	189075	58823	85.49	53.85	84.09	82.49
		± 10.15	± 10.47	± 0.09	± 0.95	± 1.05	± 1.01

5.6.1.3 Expérience 1 : Classification COVID-19 à l'aide d'images TDM

Ce paragraphe étudie les performances de notre optimiseur pour classer les données TDM en cas normaux et Covid-19 avec l'ensemble public d'images *TDM* pour l'identification du SRAS-CoV-2. Il est constitué de 1252 images d'une taille de 230 $\times$ 230 qui sont positives et 1230 images négatives. Ces données ont été recueillies auprès de vrais patients dans des hôpitaux de São Paulo, Brésil [1].

Le tableau 5.5 présente la précision, la perte, la sensibilité, la spécificité et le temps de calcul, pour tous les optimiseurs avec CNN_1 et CNN_2. Les scores rapportés indiquent que notre *ns-HMC* surpasse les optimiseurs concurrents, y compris les méthodes métaheuristiques en matière de précision d'apprentissage, et donc des performances de classification. Les valeurs de précision révèlent une légère supériorité en faveur de la méthode proposée pour les deux *CNN*. La faible performance obtenue par les méthodes métaheuristiques (*IWT*, *DA* et *SSA*) est causée par la convergence précoce subie lors du processus de recherche de la valeur optimale. Ce phénomène est dû au manque de diversité de la population et est connu sous le nom de convergence prématurée [298]. La méthode proposée bénéficie de propriétés de convergence plus rapide tout en atteignant l'optimum global grâce à la formulation bayésienne avec des

[1] https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset

taux de perte plus faibles.

Table 5.5: Expérience 1 : Résultats de la classification TDM avec les architectures CNN_1 et CNN_2 (optimiseurs (Opt), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

	CNN_1					CNN_2				
Opt.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
ns-HMC	**37**	**0.90**	**0.10**	**0.89**	**0.87**	**58**	**0.92**	**0.09**	**0.90**	**0.89**
MH	70	0.84	0.18	0.81	0.77	102	0.86	0.16	0.85	0.80
rw-MH	64.8	0.85	0.17	0.84	0.79	95	0.87	0.14	0.86	0.82
Adam	52	0.87	0.12	0.86	0.83	85.2	0.88	0.11	0.87	0.85
SGD	53	0.88	0.13	0.85	0.80	87.1	0.86	0.15	0.84	0.81
Adadelta	56	0.86	0.12	0.84	0.81	90.6	0.87	0.11	0.85	0.83
Adamax	54	0.87	0.13	0.86	0.84	85.6	0.87	0.12	0.86	0.85
IWT	59	0.84	0.21	0.81	0.78	70.2	0.86	0.19	0.84	0.83
DA	61	0.83	0.25	0.82	0.79	73.2	0.85	0.22	0.83	0.81
SSA	57	0.86	0.18	0.85	0.83	61.2	0.88	0.17	0.87	0.86

Le comportement des algorithmes pendant l'apprentissage est affiché dans les figures 5.1 et 5.2 où les courbes montrent clairement une convergence avec un taux de précision élevé pour la plupart des optimiseurs. Une différence significative entre les courbes d'entraînement et de perte peut indiquer un surajustement potentiel obtenu avec certains optimiseurs. Cet écart est réduit à l'aide de l'optimiseur proposé. Fait intéressant, le même comportement est observé pour les deux modèles *CNN*. De plus, l'augmentation de la précision entre CNN_1 et CNN_2 est presque la même pour tous les optimiseurs (voir tableau 5.5).

Les performances supérieures de la méthode proposée peuvent s'expliquer par la bonne exploration de l'espace de recherche grâce à la formulation bayésienne et au schéma d'échantillonnage efficace, réduisant ainsi également le temps de calcul. En effet, l'échantillonnage *ns-HMC* intègre une information de gradient liée à la géométrie de la distribution cible, ce qui conduit finalement à une convergence plus rapide de l'échantillonneur utilisé.

Il convient de mentionner que le comportement irrégulier des courbes pour les techniques bayésiennes (méthode proposée, *MH* et *rw-MH*) est du à l'effet d'échantillonnage aléatoire.

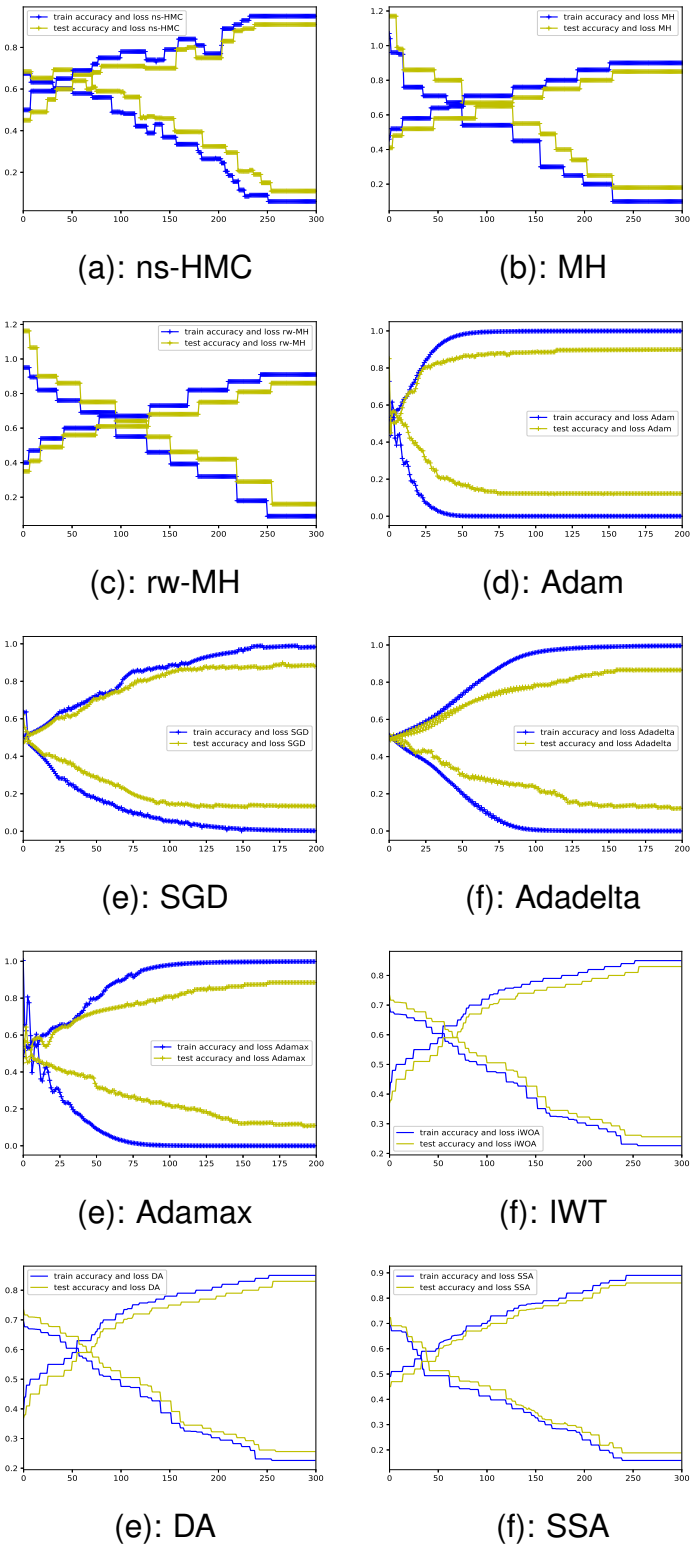


Figure 5.1: Expérience 1 : courbes d'entraînement et de test à l'aide de CNN_1.

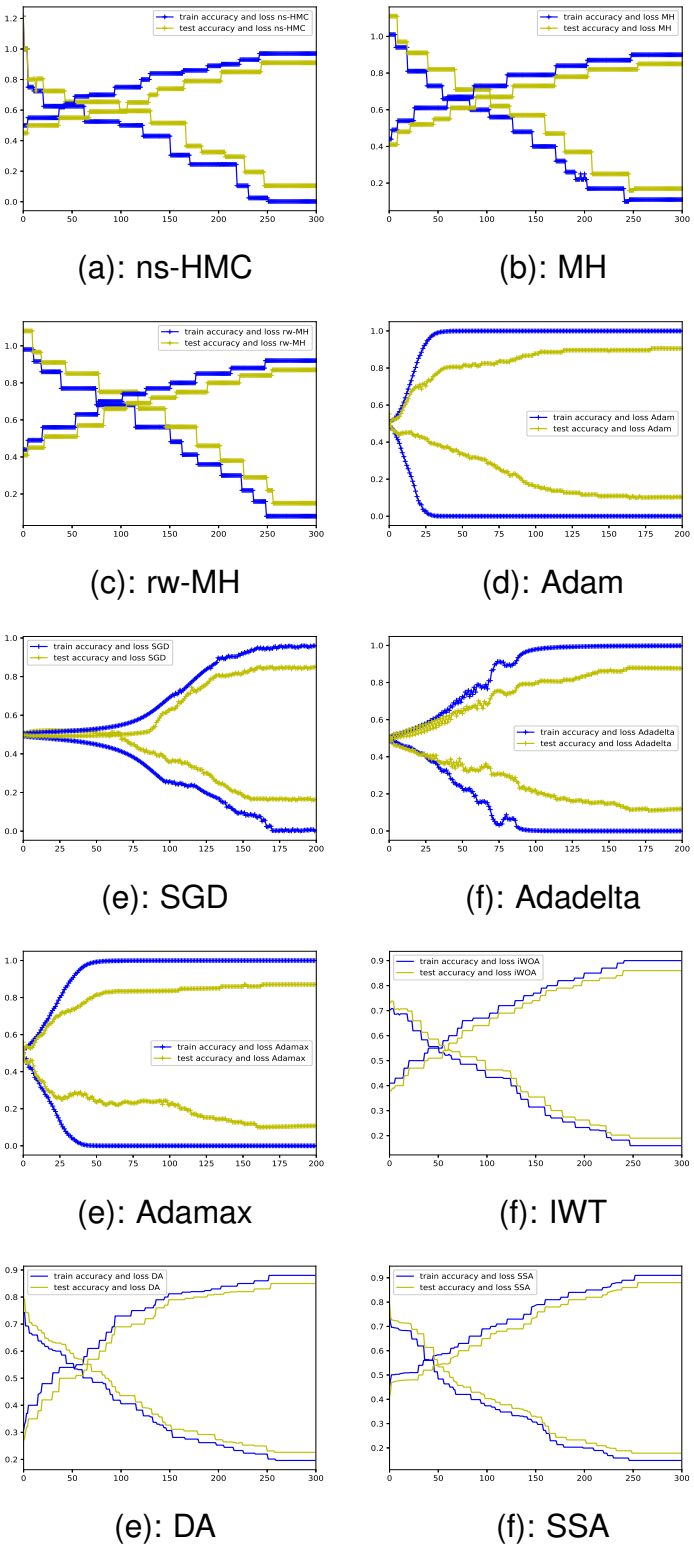


Figure 5.2: Expérience 1 : courbes d'entraînement et de test à l'aide de CNN_2.

5.6.1.4 Expérience 2: cas difficile

Un cas de classification plus difficile est abordé dans cette expérience. Les mêmes *CNN* sont utilisés afin d'identifier les infections à Covid-19 d'autres pneumonies pour les images TDM. Contrairement à l'expérience 6.5.1.3, cette tâche est difficile en raison du *riche contenu des images TDM* conjugué par la similitude entre l'infection par Covid-19 et d'autres pneumonies. L'ensemble de données COVID-TDM contient 349 images *TDM* positives pour COVID-19 appartenant à 216 patients et 397 images *TDM* négatives pour COVID-19. L'ensemble de données est ouvert au public [2]. Nous avons utilisé 566 images d'apprentissage et 180 de test avec une taille 230×230.

Les scores rapportés dans le tableau 5.6 indiquent que la méthode proposée surpasse clairement les optimiseurs concurrents dans les deux modèles pour résoudre ce problème de classification difficile. De plus, une forte diminution des performances est observée pour certains optimiseurs. *IWT*, *DA* et *SSA* ont atteint une précision légèrement meilleure que les méthodes basées sur le gradient et *MCMC*. L'algorithme *DA* est le plus performant par rapport à tous les algorithmes concurrents sur cet ensemble de données. Toutefois, il a une précision inférieure à celle de notre optimiseur *ns-HMC* d'environ 6%. Cela est principalement dû à cette classification difficile, qui conduit à un processus d'apprentissage plus complexe.

Table 5.6: Expérience 2 : Résultats de la classification TDM – cas difficile – avec les architectures CNN_1 et CNN_2 (optimiseurs (Opt), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

	CNN_1					CNN_2				
Opt.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
ns-HMC	**40**	**0.84**	**0.26**	**0.82**	**0.80**	**53**	**0.88**	**0.24**	**0.86**	**0.85**
MH	77.4	0.73	0.38	0.71	0.69	92.4	0.76	0.34	0.74	0.72
rw-MH	65	0.76	0.36	0.75	0.72	94.8	0.77	0.32	0.75	0.74
Adam	58	0.73	0.43	0.69	0.68	81	0.77	0.39	0.74	0.72
SGD	59	0.65	0.45	0.64	0.62	82.2	0.68	0.42	0.67	0.65
Adadelta	61.8	0.67	0.42	0.65	0.63	87.6	0.70	0.38	0.69	0.67
Adamax	60.6	0.69	0.41	0.67	0.66	88	0.74	0.36	0.72	0.71
IWT	54	0.75	0.38	0.74	0.72	89	0.78	0.35	0.77	0.75
DA	57	0.78	0.36	0.77	0.76	87	0.81	0.33	0.80	0.76
SSA	51	0.76	0.37	0.76	0.75	83	0.79	0.36	0.78	0.77

[2] https://www.kaggle.com/luisblanche/covidct

5.6.1.5 Expérience 3: Classification d'images Fashion-MNIST

Dans ce scénario, les performances d'apprentissage sont évaluées en utilisant une base standard *Fashion-MNIST*. Un ensemble d'entraînement de 60 000 images est adopté, tandis que l'ensemble de test est composé de 10 000 images. Chaque exemple est une image en niveaux de gris 28, associée à une étiquette de 10 classes, avec 7 000 images par classe. Pour l'entraînement du modèle, nous avons utilisé 48 000 images et 12 000 pour le test.

Le tableau 5.7 montre les résultats obtenus avec la base *Fashion-MNIST*. Tous les optimiseurs concurrents n'ont pas bien fonctionné sur la base de l'ensemble des données susmentionné. En outre, notre optimiseur était le plus performant avec une précision allant jusqu'à 93% pour les deux architectures, nettement mieux que tous les optimiseurs concurrents. En effet, comme rapporté dans le tableau 5.7, le temps de calcul pour les méthodes concurrentes est généralement d'environ 160 minutes pour le CNN_1, soit plus du double du temps nécessaire pour la méthode proposée. Même constat pour l'architecture profonde CNN_2.

Table 5.7: Expérience 3 : Résultats de la classification d'images Fashion-MNIST utilisant CNN_1 et CNN_2 (optimiseurs (Opt), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

	CNN_1					CNN_2				
Opt.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
ns-HMC	**90.5**	**0.92**	**0.22**	**0.90**	**0.88**	**308.4**	**0.93**	**0.19**	**0.91**	**0.89**
MH	172.2	0.86	0.35	0.85	0.82	505	0.87	0.33	0.84	0.82
rw-MH	191.6	0.88	0.33	0.86	0.83	519	0.88	0.31	0.85	0.84
Adam	145.4	0.90	0.38	0.85	0.82	421.3	0.91	0.32	0.88	0.87
SGD	164.4	0.88	0.71	0.73	0.70	452.4	0.89	0.56	0.84	0.83
Adadelta	169.8	0.70	1.20	0.66	0.64	439.8	0.78	0.96	0.71	0.70
Adamax	149	0.91	0.49	0.88	0.82	448.2	0.91	0.26	0.88	0.87
IWT	180.2	0.82	0.37	0.79	0.73	486	0.83	0.36	0.80	0.79
DA	174	0.79	0.40	0.76	0.74	469	0.82	0.35	0.78	0.78
SSA	165.7	0.84	0.33	0.81	0.75	453	0.86	0.24	0.86	0.85

5.6.1.6 Expérience 4: Classification d'images CIFAR-10

Dans cette section, nous testons les performances de notre approche avec la base standard *CIFAR-10*. Cet ensemble contient 60 000 images couleur de taille 32×32

réparties en 10 classes. De ce nombre, 50 000 seront utilisées pour l'entraînement des données et 10 000 pour les tests.

Les résultats de classification pour l'ensemble de données *CIFAR-10* sont présentés dans le tableau 5.8. La même conclusion peut être tirée que celle de l'ensemble de données *Fashion-MNIST*. L'optimiseur bayésien proposé a montré d'excellentes performances globales même si plus de classes sont prises en compte par rapport à tous les optimiseurs concurrents.

Table 5.8: Expérience 4 : Résultats de la classification d'images CIFAR-10 à l'aide de CNN_1 et CNN_2 (optimiseurs (Opt), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

	CNN_1					CNN_2				
Opt.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
ns-HMC	**100.7**	**0.90**	**0.25**	**0.89**	**0.87**	**331**	**0.92**	**0.21**	**0.90**	**0.87**
MH	190	0.83	0.41	0.80	0.78	512.2	0.84	0.36	0.83	0.81
rw-MH	197	0.85	0.36	0.84	0.81	525	0.86	0.35	0.84	0.83
Adam	161	0.87	0.42	0.83	0.81	429	0.90	0.36	0.87	0.86
SGD	169	0.86	0.75	0.65	0.65	459.7	0.86	0.53	0.83	0.80
Adadelta	174.7	0.75	0.92	0.68	0.65	453.3	0.79	0.81	0.72	0.70
Adamax	155	0.90	0.34	0.88	0.85	507.8	0.91	0.33	0.87	0.85
IWT	186	0.80	0.35	0.78	0.74	503	0.81	0.34	0.79	0.78
DA	179	0.82	0.35	0.77	0.75	489	0.84	0.32	0.78	0.76
SSA	172.7	0.83	0.31	0.81	0.77	487	0.85	0.29	0.83	0.80

5.6.1.7 Comparaison sur un réseau CNN profond :

Cette section étudie les performances de notre méthode textitns-HMC en utilisant un *CNN* profond sur le jeu de données standard *Fashion-MNIST*. Le *CNN* profond proposé est plus profond que CNN_1 et CNN_2. Il est composé de 5 X Conv3x3-32, 5 X Conv3x3-64, 5 X Conv3x3-128, 5 X Conv3x3-256, et 5 X Conv3x3-512, *FC*-512, *FC*-256, *FC*-128 et une couche FC-softmax. Toutes les couches convolutives utilisent des filtres de taille 3 et sont suivies d'une opération de max-pooling avec une fenêtre de 2, avec un stride size égal à 1.

L'utilisation d'un *CNN* profond valide la bonne performance de notre optimiseur proposée et ce en termes de précision, perte, temps de calcul, sensibilité et de spécificité, comme le montre dans le tableau 5.9. De plus, la plupart des optimiseurs concurrents indiquent clairement un effet de surajustement comme *SGD* et Adadelta, contrairement

à la méthode proposée.
Par conséquent, on peut facilement remarquer la précision globale élevée de notre optimiseur bayésien, abstraction faite de la profondeur des architectures.

Table 5.9: Résultats de la classification Fashion-MNIST avec une architecture profond (optimiseurs (Opt), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

Optimiseurs	Temps (min)	Précision	Perte	Sensibilité	spécificité
ns-HMC	**638**	**0.93**	**0.21**	**0.91**	**0.90**
MH	917	0.85	0.38	0.82	0.79
rw-MH	925	0.86	0.37	0.84	0.80
Adam	854	0.89	0.34	0.87	0.85
SGD	859	0.88	0.41	0.84	0.81
Adadelta	875	0.82	0.60	0.78	0.76
Adamax	857	0.92	0.32	0.90	0.88
IWT	810	0.87	0.34	0.82	0.79
DA	806	0.86	0.37	0.82	0.78
SSA	813	0.89	0.32	0.86	0.84

5.6.2 Classification longitudinale

Nous validons dans ce paragraphe notre méthode proposée sur la classification de séries temporelles d'images. Deux expériences sont utilisées à l'aide de deux ensembles de données : le premier ensemble se concentre sur l'évolution du COVID-19, même ensemble de données présenté dans le chapitre 4, tandis que le deuxième ensemble de données est dédié à l'évaluation de la maladie d'Alzheimer.

Pour analyser l'évolution du COVID-19 et de la maladie d'Alzheimer, l'application utilise une architecture d'apprentissage profond pour la catégorisation multi-temporelle des images radiographiques, ProgNet (voir chapitre 4). Elle est construit à l'aide d'un hybride d'architecture de réseau neuronal convolutif (*CNN*) et de réseau neuronal récurrent (*RNN*).

L'architecture utilisée est identique à celle exposée dans le chapitre précédent, ProgNet. Quant au *CNN*, nous avons opté pour le réseau ResNet-50.

Après cela, un *RNN* est utilisé pour apprendre les corrélations temporelles entre les multiples vecteurs caractéristiques connectés à chaque image d'une séquence T, le réseau *LSTM* étant utilisé en raison de ses bonnes performances dans la littérature (avec 64 unités cachées). [299]–[301].

Pour la comparaison, la technique d'optimisation la plus populaire utilisée en Deep Learning, Adam, est utilisée avec un taux d'apprentissage égal à 10^{-3}.

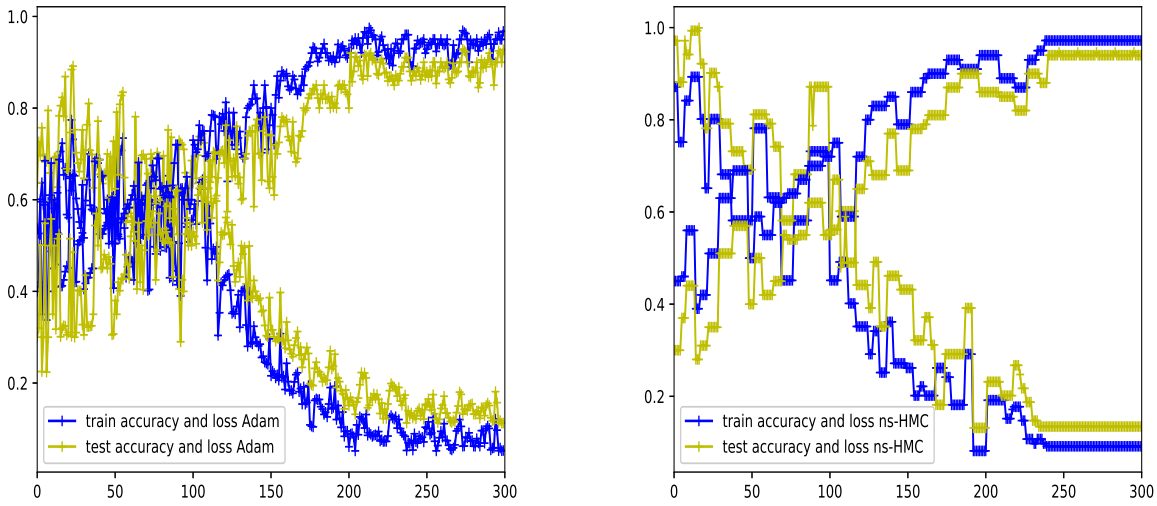
5.6.2.1 Expérience 1 : pronostic COVID-19

Comme décrit dans le chapitre précédent, nous avons utilisé 51 séquences pour l'entraînement, chacune comprenant 3 images radiographiques du même patient, et 16 séquences pour le test.

Pour une séquence $T = (T_1, T_2, T_3)$ d'images à rayon X, chaque image T_i passe par un *CNN* pour extraire un vecteur caractéristique en appliquant de nombreuses convolutions suivies de couches de max-pooling.

Évaluation de la perte et de la précision

La figure 5.3 présente les courbes de précision et de perte pour les phases d'entraînement et de test. Les courbes montrent la convergence de notre optimiseur *ns-HMC* et de l'optimiseur *Adam*, avec un taux de précision élevé et un faible taux de perte.



(a): Prognet-Adam (b): Prognet-ns-HMC

Figure 5.3: Courbes d'entraînement et de test : pronostoc COVID-19,

Évaluation quantitative

Le tableau 5.10 rapporte les valeurs obtenues pour les taux de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs. L'architecture *ProgNet* proposée avec

notre optimiseur Bayésien surpasse l'optimiseur *Adam* avec une différence de 3% pour les vrais positifs et vrais négatifs. Les taux de faux positifs et de faux négatifs sont également plus bas avec notre méthode, ce qui permet d'améliorer la performance et la précision de l'évolution du COVID-19.

Table 5.10: Valeurs VP, FN, FP et VN de l'optimiseur ns-HMC et Adam pour la base COVID-19.

	VP	**FN**	**FP**	**VN**
ProgNet-Adam	92.51%	6.15%	6.09%	92.32%
ProgNet-ns-HMC	**94.14%**	**4.34%**	**4.21%**	**94.47%**%

Pour mieux évaluer les performances de l'optimiseur proposé, le tableau 5.11 résume les valeurs obtenues pour les taux d'exactitude, précision rappel, F_1-score et le temps de calcul.

Dans le tableau 5.11, les moyennes ont été calculées à partir de 10 exécutions, avec une sélection aléatoire de sous-parties de données d'entraînement pour chaque exécution. Les écarts-types pour les 10 essais sont également fournis dans le tableau.

Des valeurs de précision et de rappel plus élevées indiquent que notre optimiseur ns-HMC avec l'architecture ProgNet est plus efficace pour récupérer les cas d'infection ambigus de la maladie COVID-19. Il est à noter que l'optimiseur proposé ns-HMC a un temps de calcul plus rapide que les optimiseurs de l'état de l'art dédiés à la classification des séries temporelles d'images, tels que Adam. En outre, les faibles valeurs d'écart-type rapportées indiquent une stabilité accrue pour le modèle proposé, ce qui conduit à de meilleures capacités de généralisation.

Table 5.11: Exactitude, précision, rappel, F_1-Score et le temps de calcul (en minutes) de l'optimiseur ns-HMC et Adam pour la base COVID-19.

	ProgNet-Adam	**ProgNet-ns-HMC**
Exactitude	0.92 ± 0.03	**0.94** ± 0.04
Precision	0.92 ± 0.04	**0.94** ± 0.03
Rappel	0.91 ± 0.05	**0.92** ± 0.04
F_1**-score**	0.91 ± 0.04	**0.92** ± 0.03
Temps (min)	218.2 ± 0.05	**138.9** ± 0.03

Analyse qualitative

Nous décrivons dans cette partie les mêmes séquences d'évolution du COVID-19 (décès ou survie) représentées au chapitre 4. La figure 5.4 illustre quatre séries

d'images de patients atteints de Covid-19. Les deux en haut, ont survécu, tandis que les deux en bas sont morts.

Les résultats obtenus indiquent que notre modèle a bien classé les séries chronologiques pour les deux cas de survie et de décès. Ces résultats confirment les valeurs présentées dans le tableau 5.11, qui montrent à quel point l'optimiseur *ns-HMC* est capable de bien classer les cas ambigus pour le problème de la mort par rapport à l'optimiseur *Adam*.

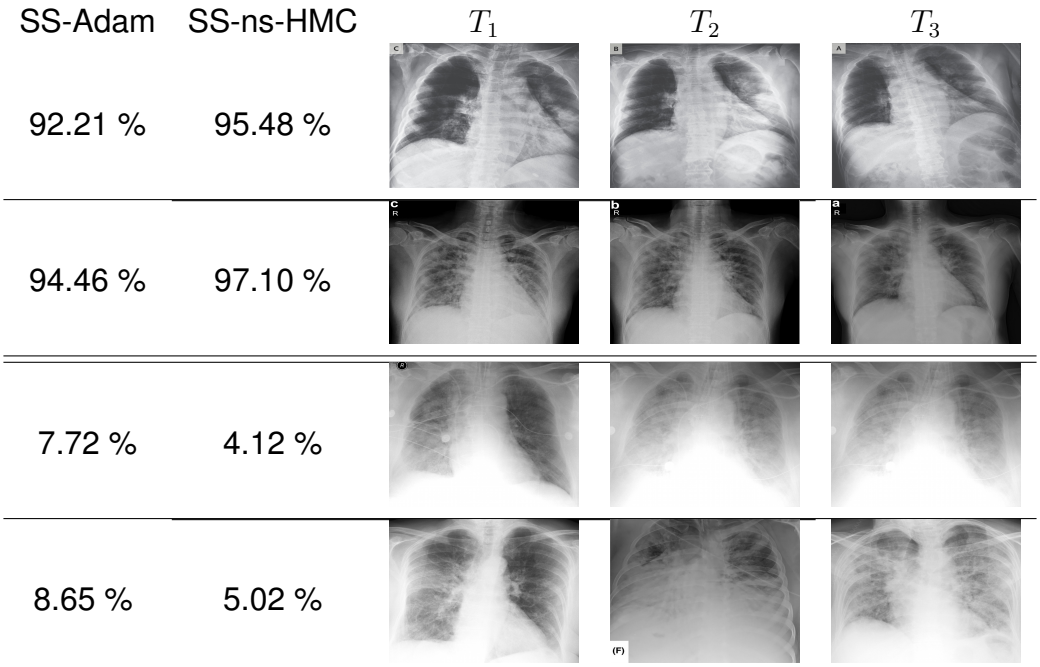


Figure 5.4: Visualisation de la classification COVID-19 à l'aide de l'optimiseur ns-HMC proposée et l'optimiseur concurrent Adam.

5.6.2.2 Expérience 2: pronostic Alzheimer

Cette section présente les performances de l'optimiseur proposé pour l'évolution de la maladie d'Alzheimer pour les personnes âgées. Des expériences sont réalisées sur une base de données ouverte de cas de maladie d'Alzheimer en utilisant des *IRM* [3] [302]. Deux classes sont considérées : personnes démentes et non démentes.

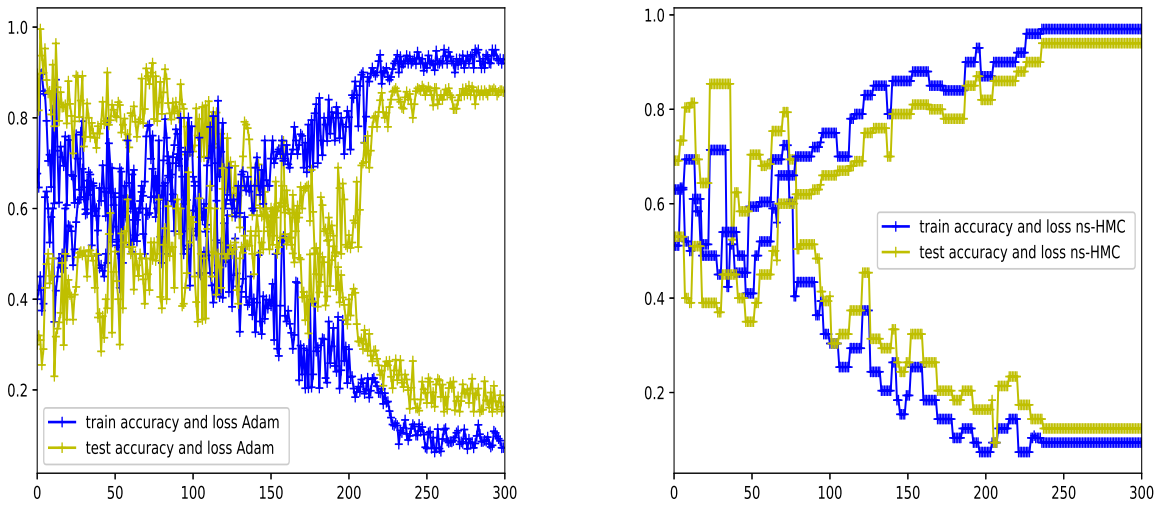
L'ensemble de données *IRM* est une série longitudinale constituée de 150 sujets âgés de 60 à 96 ans. Ils sont capturés avec le même scanner au moyen de séquences de

[3] https://www.oasis-brains.org/#data

4 images. Chaque sujet a fait l'objet d'au moins deux visites sur une période d'un an, pour 373 séances d'imagerie. Les sujets ont été caractérisés comme non déments ou atteints de la maladie d'Alzheimer très légère à légère. Au cours de l'étude, 72 des sujets étaient considérés comme non déments, et 64 d'entre eux ont été classés comme atteints de démence. En conséquence, les caractéristiques de l'échantillon étaient semblables entre les individus avec et sans la maladie d'Alzheimer, ce qui peut rendre cette tâche difficile.

Évaluation de la perte et de la précision

Compte tenu la régularisation utilisée, un comportement de surajustement peut être signalé en analysant les courbes affichées par la figure 5.5 de l'optimiseur concurrent *Adam*. Il est clair visuellement que l'optimiseur bayésien converge vers un taux de précision élevé et un faible taux de perte par rapport à *Adam*. Malgré la difficulté de cette base, les courbes d'entraînement et de test sont plus stables, ce qui indique les bonnes performances de l'optimiseur *ns-HMC*.



(a): Prognet-Adam (b): Prognet-ns-HMC

Figure 5.5: Courbes d'entraînement et de test : pronostic Alzheimer,

Évaluation quantitative

Les scores rapportés dans le tableau 5.12 prévoient que la méthode proposée surpasse clairement *Adam* d'environ 94% pour les vrais positifs et vrais négatifs. Les valeurs

obtenues de FN et FP, à peu près 5%, affirment le bon fonctionnement de l'optimiseur proposé.

Les résultats quantitatifs sont les mêmes dans le tableau 5.13 pour *ns-HMC* par rapport à *Adam* qui ne dépasse pas le taux de 86%. Notre optimiseur offre un gain de temps de calcul plus important que l'optimiseur *Adam* standard. De plus, les faibles valeurs de variance indiquent une bonne stabilité du modèle.

Table 5.12: Valeurs VP, FN, FP et VN de l'optimiseur ns-HMC et Adam pour la base Alzheimer.

	VP	**FN**	**FP**	**VN**
ProgNet-Adam	86.12%	9.42%	8.29%	86.71%
ProgNet-ns-HMC	**93.79%**	**5.81%**	**5.05%**	**93.54%**

Table 5.13: Exactitude, précision, rappel, F_1 Score et le temps de calcul (en minutes) de l'optimiseur ns-HMC et Adam pour la base Alzheimer.

	ProgNet-Adam	**ProgNet-ns-HMC**
Exactitude	0.86 ± 0.07	**0.93** ± 0.04
Precision	0.85 ± 0.11	**0.93** ± 0.06
Rappel	0.85 ± 0.10	**0.93** ± 0.05
F_1 **score**	0.85 ± 0.09	**0.93** ± 0.04
Temps (min)	327.4 ± 0.04	**210.9** ± 0.03

Analyse qualitative

Nous présentons quelques résultats représentatifs obtenus à partir d'un groupe de patients. Ces séries temporelles consistent en quatre images prises à T_1, T_2, T_3 et T_4. La figure 5.6 affiche quatre séries chronologiques de patients, deux d'entre eux (en haut) sont atteints de la maladie d'Alzheimer, tandis que les deux autres (en bas) ne sont pas atteints de démence.

Les séries chronologiques d'IRM du cerveau de personnes atteintes de la maladie d'Alzheimer montrent souvent une augmentation des zones noires, notamment dans la cavité cérébrale et les sillons. Cette augmentation des zones noires est généralement due à la perte progressive de neurones dans le cerveau, qui est une caractéristique de la maladie d'Alzheimer et d'autres maladies neurodégénératives. Au fur et à mesure que la maladie d'Alzheimer progresse, les zones de tissu cérébral endommagé peuvent s'étendre à d'autres régions du cerveau, ce qui peut entraîner une perte progressive des fonctions cognitives et un déclin fonctionnel chez les patients.

Les scores obtenus dans la figure 5.6 affirment la bonne précision obtenue par notre optimiseur comme démontrée dans les tableaux 5.12 et 5.13 pour les deux cas déments et non déments.

L'analyse de séquences plus longues comprenant quatre images a conduit à une amélioration de la corrélation temporelle entre ces dernières. En outre, pour les personnes atteintes de la maladie d'Alzheimer légère, l'extension des zones affectées par la maladie tend à augmenter progressivement au fil du temps. Les deux méthodes proposées, *ProgNet-ns* et *HMC*, ont permis de surmonter le problème de similarité des caractéristiques présentes au sein d'une même séquence d'images.

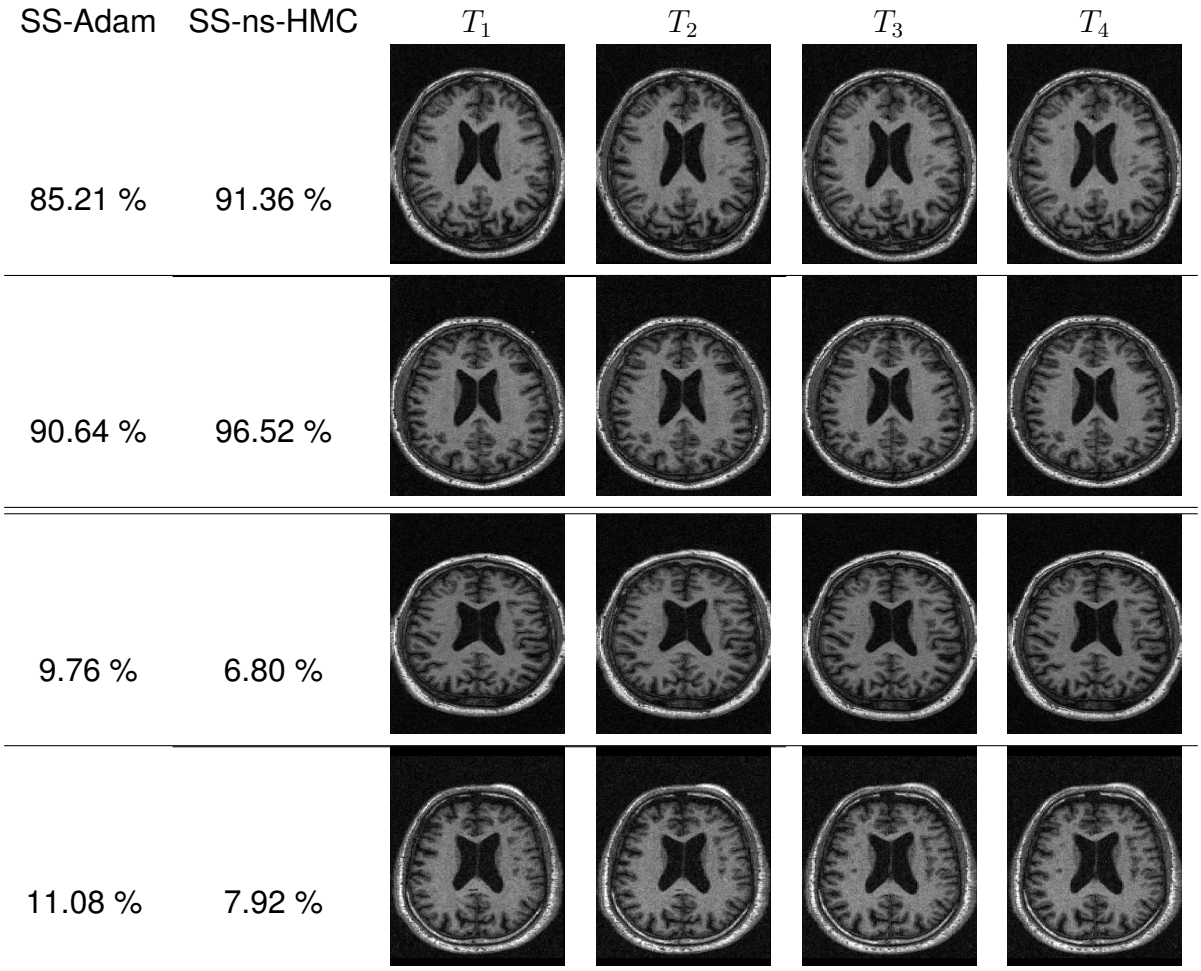


Figure 5.6: Visualisation de la classification d'Alzheimer à l'aide de l'optimiseur ns-HMC proposée et l'optimiseur concurrent Adam.

5.7 Discussion

Nous avons présenté un modèle d'optimisation basé sur le cadre bayésien en utilisant un schéma Monte-Carlo hamiltonien pour résoudre les problèmes d'optimisation avec des contraintes de parcimonie.

Notre optimiseur a été validé sur des tâches de classification transversale et longitudinale en utilisant quatre ensembles de données différents, ainsi qu'une validation de référence sur les bases de données Fashion-MNIST et CIFAR-10. La méthode a été comparée à d'autres optimiseurs de référence tels que *Adam*, *SGD* et *Adadelta*. Dans la partie longitudinale, deux ensembles de données concernant l'évolution du COVID-19 et de la maladie d'Alzheimer ont été utilisés, et la méthode a été comparée à l'optimiseur le plus adapté en apprentissage profond, *Adam*.

Ces différents ensembles de données ont été exploités afin d'évaluer : *i)* l'efficacité de notre méthode sur un cas de classification où des optimiseurs concurrents fournissent de bons résultats, *ii)* les propriétés de convergence rapide, et *iii)* la robustesse de notre méthode en fonction de la taille de l'échantillon.

Bien que les réseaux de neurones profonds aient une bonne capacité d'expression, leurs grands paramètres de modèle, qui pèsent lourdement sur le calcul, restent un problème à résoudre. Ce problème entrave le déploiement à grande échelle d'applications basées sur les réseaux de neurones profonds (*DNN*). Il est donc primordial de réduire les paramètres du modèle sans perdre en performances. Les réseaux parcimonieux sont un moyen de réduire efficacement la complexité, ce qui peut améliorer l'efficacité et la généralisabilité du modèle.

Des preuves empiriques montrent que les architectures profondes nécessitent souvent d'être sur-paramétrées (ayant plus de paramètres que d'exemples d'entraînement) pour qu'elles soient formées avec succès [303], [304]. En effet, l'utilisation de ces réseaux est utile pour extraire plus de caractéristiques implicites qui conduisent à une bonne précision du modèle susceptible de réduire l'effet de surapprentissage. Cependant, une fois que les relations entrées-sorties sont correctement représentées par un réseau complexe, un tel réseau peut constituer un point de départ pour trouver une architecture plus simple et plus parcimonieuse [303], [304].

Les expériences conduisent à conclure que le schéma d'échantillonnage hamiltonien non lisse proposé fournit une convergence plus rapide et plus précise avec des effets de surapprentissage plus faibles. Le gain obtenu n'est pas seulement dû à la formulation bayésienne, mais aussi à l'efficacité du schéma d'inférence.

La méthode proposée accélère le temps d'apprentissage pour les architectures utilisées. En effet, pour le problème de classification difficile à étudier (expérience 2 de la première partie par exemple), un réseau plus profond n'a pas résolu le problème de surajustement avec les optimiseurs standard, tandis que notre méthode surmonte cette limitation en fournissant une optimisation plus précise du critère cible, dans la mesure où il n'a besoin que d'une architecture de réseau de neurone simple pour produire une grande précision.

5.8 Conclusion

Nous avons mis en place un nouveau modèle d'optimisation bayésienne pour ajuster les poids des réseaux de neurones artificiels parcimonieux. La méthode proposée repose sur la dynamique hamiltonienne avec des régularisations non lisses. L'optimiseur *ns-HMC* proposé a montré des résultats prometteurs avec des performances de classification globales élevées et de bonnes propriétés de généralisation. De plus, il se caractérise par un temps de calcul relativement faible comparé aux autres algorithmes concurrents.

L'utilisation de jeux de données standard (*Fashion-MNIST* et *CIFAR-10*) ou des séries temporelles pour les bases COVID-19 et Alzheimer avec différentes séquences, confirme la stabilité de notre optimiseur en termes de précision. Par conséquent, les différentes expériences réalisées ont montré une bonne généralisation de *ns-HMC* et leur capacité de s'appliquer pour la classification transversale ou encore tout en utilisant des séquences temporelles.

6

Fonctions d'activation entraînables

Sommaire

6.1 Introduction

L'introduction de nouvelles fonctions d'activation a suscité un regain d'intérêt de la part des chercheurs pour les réseaux de neurones. Par exemple, l'utilisation de *ReLU* [153], de la *ReLU paramétrique* [174], ainsi que d'autres fonctions d'activation similaires, s'est avérée considérablement utile pour améliorer les performances du réseau, grâce à des propriétés telles que la non-saturation qui permet d'éviter des problèmes comme la disparition du gradient [305]. Étant donné que *ReLU* est très efficace et très simple, de nombreux chercheurs en apprentissage profond ont concentré leurs efforts sur la recherche de fonctions d'activation similaires à *ReLU*, mais avec des propriétés légèrement différentes.

Afin d'améliorer l'efficacité de notre modèle *ns-HMC* et de surmonter les difficultés associées aux fonctions d'activation décrites dans le chapitre 3, nous proposons une nouvelle forme de fonction d'activation entraînable. L'estimation des paramètres de cette fonction est intégrée dans le modèle bayésien, et l'inférence MCMC permet d'estimer ces paramètres directement à partir des données, tout comme les poids du modèle. Cette contribution étend l'optimiseur proposé *ns-HMC* en adoptant des méthodes hamiltoniennes non-lisses pour ajuster les poids des réseaux de neurones artificiels parcimonieux.

6.2 Formulation du problème

Nous avons examiné différentes fonctions d'activation présentées dans l'état de l'art du chapitre 3. Elles peuvent prendre diverses formes, telles que les fonctions d'activation standard (*sigmoïde* [151], *tangente hyperbolique* [152], *ReLU* [153]). D'autres types de fonctions largement utilisées sont les "fonctions entraînables", dont les paramètres sont ajustés à l'aide de la méthode de descente de gradient, comme *FReLU* [173] et *MeLU* [179]. Les fonctions entraînables sont utilisées pour améliorer la capacité du modèle à apprendre rapidement et efficacement pendant l'entraînement. Elles ont démontré leur capacité à surpasser les fonctions classiques, mais ces progrès n'ont pas réussi à remplacer *ReLU*, qui reste la fonction d'activation la plus utilisée dans la communauté de l'apprentissage profond.

Cependant, il est important de noter que ces approches présentent des limites, telles que le temps de calcul et la disparition du gradient, qui peuvent conduire à des modèles

sensibles à la présence de minima locaux [198] et à une diminution des performances. La flexibilité des fonctions et l'estimation des paramètres sont encore des problèmes ouverts.

Dans ce chapitre, nous introduisons une modification de la fonction d'activation *MeLU* [306], et nous intégrons l'estimation de ses paramètres dans un cadre global d'optimisation bayésienne. Le choix de la fonction se justifie principalement par ses performances prometteuses, ainsi que par sa forme qui permet de favoriser la non-linéarité et la parcimonie. Cependant, les limites de la fonction *MeLU* sont principalement liées aux besoins en mémoire.

La fonction d'activation *Modified Mexican ReLU (MMeLU)* est proposée pour résoudre le problème de complexité et améliorer les performances du modèle. La spécificité de cette nouvelle fonction d'activation est, d'une part, qu'elle nécessite moins de paramètres à estimer que *MeLU* et, d'autre part, qu'elle permet l'ajustement de tous les paramètres du réseau de neurones artificiels dans un cadre bayésien, plutôt que d'utiliser une procédure d'optimisation standard comme l'optimiseur *ADAM*. De plus, notre méthode peut être appliquée aux problèmes de régression qui cherchent à trouver des relations entre des variables dépendantes et indépendantes ainsi qu'à l'estimation de variables continues.

Pour définir *MMeLU*, soit

$$f_{\gamma,b}(x) = max(b - |x - \gamma|, 0) \tag{6.1}$$

où x est l'entrée d'une couche de réseau de neurones et γ, b sont des nombres réels.

En utilisant $f_{\gamma,b}(x)$, la fonction d'activation proposée *MMeLU* peut être définie comme

$$\textit{MMeLU}(x) = c\, f_{\gamma,b}(x) + (1 - c)\textit{ReLU}(x) \tag{6.2}$$

où c est un nombre réel appartenant à l'intervalle $[0, 1]$. c, b et γ sont les paramètres à estimer.

L'allure de la fonction *MMeLU* proposée, ainsi que celui des fonctions concurrentes *ReLU*, *FReLU*, *PReLU* et *MeLU*, est illustrée à la figure 6.1. La fonction d'activation proposée dans ce chapitre est formée d'un mélange de la fonction *ReLU* et de la fonction chapeau mexicain. Les résultats des courbes (a-d) montrent clairement la flexibilité et la non-linéarité de notre fonction *MMeLU* avec différentes configurations des paramètres γ, b et c.

La fonction mexicaine est souvent utilisée comme fonction d'activation dans les réseaux de neurones en raison de ses avantages. Les fonctions mexicaines sont continues, ce qui signifie que les changements dans les entrées produisent des changements continus dans les sorties. Cela est important dans les réseaux de neurones car il permet une mise à jour graduelle des poids. De plus, elles ont une capacité de représentation élevée, ce qui signifie qu'elles peuvent modéliser des fonctions complexes avec précision. Cela permet aux réseaux de neurones d'apprendre des relations non linéaires entre les entrées et les sorties.

La fonction chapeau mexicain ressemble à une cloche mais avec un pic au centre qui la rend plus prononcée pour les petites valeurs, comme le montrent les courbes de *MMeLU* pour les figures 6.1[a] et [b]. Cela signifie que pour les petites valeurs d'entrée, la fonction chapeau mexicain aura une réponse plus forte que d'autres fonctions d'activation, telles que *ReLU* ou *FReLU*. Ce comportement peut être utile dans certaines situations, par exemple lorsque les données d'entrée ont une plage de valeurs restreinte ou lorsque la réponse du réseau de neurones doit être plus sensible aux petites variations des données en entrée.

La fonction d'activation *MMELU*, illustrée dans la figure 6.2, est appliquée après chaque couche de convolution dans l'architecture *CNN*, en remplacement de la fonction *ReLU*. Cette stratégie a le potentiel d'augmenter la non-linéarité du modèle et d'améliorer la capacité de représentation des caractéristiques des images.

Le label (ou valeur numérique) estimé est obtenu par l'application de la fonction d'activation proposée *MMeLU*(x, W), qui est une fonction non linéaire de l'entrée x et du vecteur de poids $W \in \mathbb{R}^N$. Le vecteur de poids peut être déterminé pendant la phase d'apprentissage en utilisant une distance générique $\mathcal{L}$ (Euclidienne, Minkowski, etc.) basée sur l'erreur appliquée aux M données d'entrée :

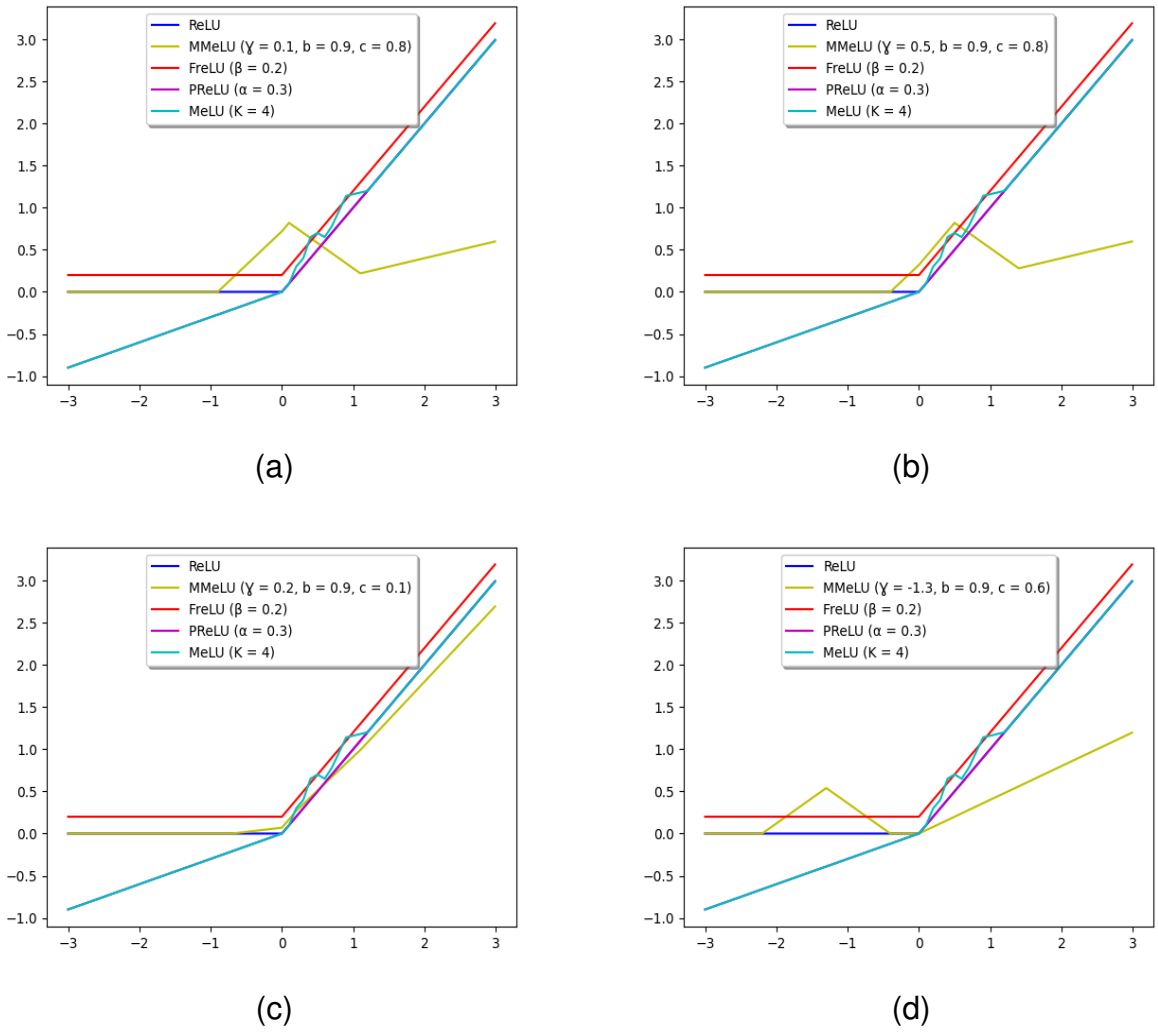


Figure 6.1: Comparaison des courbes ReLU, FreLU, PReLU, MeLU et MMeLU avec différentes configurations.

$$\widehat{W} = \arg\min_{W} \sum_{m=1}^{M} \mathcal{L}(MMeLU(x^m; W) - y^m) + \sum_{l=1}^{L} \lambda_l \|W^l\|_1, \tag{6.3}$$

où y représente la vérité terrain, L est le nombre de couches dans le réseau et λ_l est un paramètre de régularisation à estimer d'une couche l qui équilibre la solution entre le terme d'attache aux données et les termes de régularisation ℓ_1.

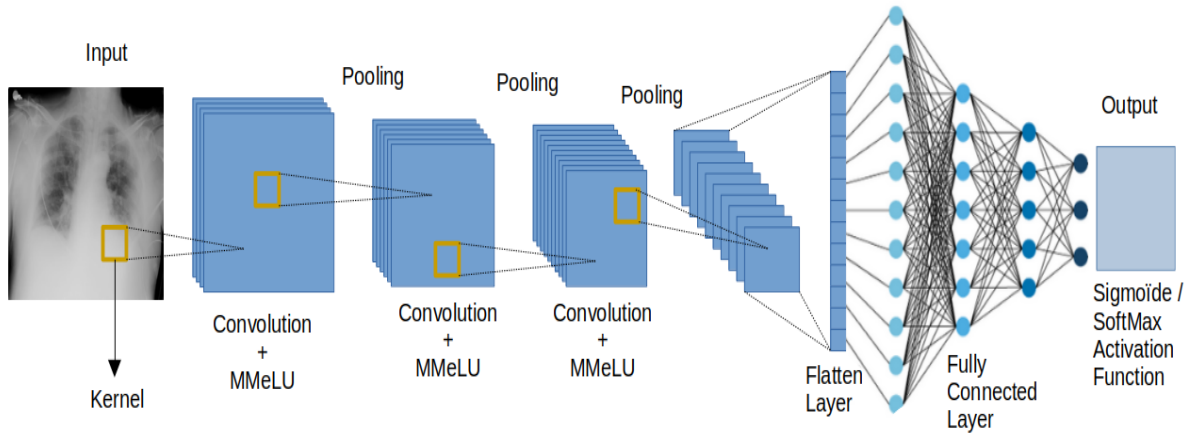


Figure 6.2: Un schéma général de MMeLU.

6.3 Modèle bayésien hiérarchique

Le problème d'estimation des paramètres de la fonction d'activation *MMeLU* est formulé dans un cadre bayésien. En ce sens, tous les paramètres et hyperparamètres sont censés suivre des lois de probabilité. Une vraisemblance est définie pour modéliser le lien entre le vecteur des poids cibles, les paramètres de la fonction d'activation et les données. Une loi *a priori* est définie pour modéliser les connaissances *a priori* sur les poids cibles, ainsi tous les paramètres de la fonction d'activation.

6.3.1 Vraisemblance

Selon le principe de minimisation de l'erreur entre le vecteur de référence y (label ou valeurs continues) et son estimation $\widehat{y}$, nous définissons la loi de vraisemblance comme

$$f(\mathbf{y}, \mathbf{x}; W, c, \gamma, b) \propto \times \prod_{m=1}^{M} \exp\left[-\mathcal{L}\left(MMeLU(x^m; W, c, \gamma, b) - y^m\right)\right]. \tag{6.4}$$

où $\mathbf{x} = \{x^1, \ldots, x^M\}$ et $\mathbf{y} = \{y^1, \ldots, y^M\}$.

6.3.2 Lois *a priori*

Dans notre modèle, les paramètres inconnus sont regroupés dans le vecteur inconnu $\theta = \{W, c, \gamma, b, \lambda\}$, avec $\lambda = \{\lambda_1, \ldots, \lambda_L\}$.

Loi *a priori* de W :
Concernant la connaissance *a priori* du vecteur de poids W, nous proposons d'utiliser une distribution de Laplace afin de favoriser la parcimonie du réseau de neurones :

$$f(W;\lambda) \propto \prod_{l=1}^{L}\prod_{k=1}^{K_l}\left[\frac{1}{\lambda_l}\exp\left(-\frac{|W_k^l|}{\lambda_l}\right)\right], \tag{6.5}$$

où K_l est le nombre de poids de la couche l du réseau, λ_l est un paramètre à estimer.

Loi *a priori* de λ_l :
Étant donné que les valeurs de λ_l sont des réels positifs, nous avons choisi d'utiliser une distribution inverse gamma (IG):

$$f(\lambda_l;\delta,\mu) = IG(\lambda_l;\delta,\mu) \propto (\lambda_l)^{-1-\delta}\exp\left(-\frac{\mu}{\lambda_l}\right). \tag{6.6}$$

où δ et μ sont des paramètres positifs qui ont été fixés à 10^{-3} pour avoir une loi non-informative.

Loi *a priori* de c :
En ce qui concerne le paramètre c, on considère une loi uniforme a sur l'intervalle $[0,1]$ notée

$$c \sim U_{[0,1]}(c). \tag{6.7}$$

Loi *a priori* de γ :
Comme γ est un réel, une loi gaussienne est utilisée comme suit

$$f(\gamma;\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\gamma^2}{2\sigma^2}\right). \tag{6.8}$$

où σ^2 est un hyperparamètre à estimer.

Loi *a priori* de b :
Comme b est un réel positif, une loi exponentielle est utilisée comme suit

$$f(b;\lambda_b) \propto \begin{cases} \frac{1}{\lambda_b}\exp\left(-\frac{b}{\lambda_b}\right) \text{ si } b \geq 0 \\ 0 \text{ sinon.} \end{cases} \tag{6.9}$$

où λ_b est un hyperparamètre à estimer.

6.3.3 Lois hyper-a priori

Étant donné que λ_b et σ^2 sont des réels positifs, une distribution inverse gamma (IG) a été utilisée comme hyper-*a priori* :

hyper-*a priori* pour λ_b :

$$f(\lambda_b;\delta,\mu) = IG(\lambda_b;\delta,\mu) \propto (\lambda_b)^{-1-\delta}\exp\left(-\frac{\mu}{\lambda_b}\right). \tag{6.10}$$

hyper-*a priori* pour σ^2 :

$$f(\sigma^2;\delta,\mu) = IG(\sigma^2;\delta,\mu) \propto (\sigma^2)^{-1-\delta}\exp\left(-\frac{\mu}{\sigma^2}\right). \tag{6.11}$$

où δ et μ sont des paramètres positifs qui ont été fixés à 10^{-3}.

6.4 Schéma d'inférence

En adoptant une approche *MAP*, nous devons d'abord exprimer la distribution *posteriori*. Notons Φ_e comme les hyperparamètres à estimer, représentés par Φ_e = {σ^2, λ_b}, et

Φ_m comme les hyperparamètres à fixer, $\Phi_m = \{\delta, \mu\}$. En utilisant les informations de vraisemblance, les lois a priori et les hyperpriors définis, on peut écrire la distribution *a posteriori* comme :

$$f(\theta, \Phi_e; y, x, \Phi_m) \propto f(y, x; \theta) f(\theta; \Phi_e) f(\Phi_e; \Phi_m) \tag{6.12}$$

qui peut être reformulé dans une version détaillée comme

$$\begin{aligned} &f(\theta, \Phi_e; y, x, \Phi_m) \propto \\ &\sum_{m=1}^{M} \exp\left[-\mathcal{L}\left(MMeLU(x^m; W, c, \gamma, b) - y^m\right)\right] \times \prod_{l=1}^{L} \frac{1}{\lambda_l^{K_l}} \prod_{k=1}^{K_l} \left[\exp\left(-\frac{|W_k^l|}{\lambda_l}\right)\right] \times \\ &(\lambda_l)^{-1-\delta} \exp\left(-\frac{\mu}{\lambda_l}\right) \times \exp\left(-\frac{\gamma^2}{2\sigma^2}\right) \times \frac{1}{\lambda_b} \exp\left(-\frac{b}{\lambda_b}\right) 1_{\mathbb{R}_+}(b) \times \\ &1_{[0,1]}(c) \times (\lambda_b)^{-1-\delta} \exp\left(-\frac{\mu}{\lambda_b}\right) \times (\sigma^2)^{-1-\delta} \exp\left(-\frac{\mu}{\sigma^2}\right). \end{aligned} \tag{6.13}$$

Afin de calculer la distribution conditionnelle liée à chaque paramètre du modèle, il faut intégrer la distribution *posteriori* conjointe dans 6.13 par rapport à tous les autres paramètres.

En ce qui concerne le paramètre W, les calculs basés sur (6.13) conduisent à la forme suivante

$$f(W; c, \gamma, b, \lambda) \propto \exp\left[-\sum_{m=1}^{M} \left(\mathcal{L}\left(MMeLU(x^m; W, c, \gamma, b) - y^m\right)\right) - \sum_{l=1}^{L} \sum_{k=1}^{K} \frac{|W_k^l|}{\lambda_l}\right]. \tag{6.14}$$

La loi conditionnelle pour le paramètre c s'écrit comme

$$f(c; W, b, \gamma) \propto \exp\left[-\sum_{m=1}^{M} \left(\mathcal{L}\left(MMeLU(x^m; W, c, \gamma, b) - y^m\right)\right)\right] \times 1_{[0,1]}(c). \tag{6.15}$$

La loi conditionnelle pour le paramètre b s'écrit comme

$$f(b; W, c, \gamma, \lambda_b) \propto \exp\left[-\sum_{m=1}^{M}\left(\mathcal{L}\left(MMeLU(x^m; W, c, \gamma, b) - y^m\right) - \frac{b}{\lambda_b}\right)\right]. \quad (6.16)$$

La loi conditionnelle pour le paramètre γ s'écrit comme

$$f(\gamma; W, b, c, \sigma^2) \propto \exp\left[-\sum_{m=1}^{M}\left(\mathcal{L}\left(MMeLU(x^m; W, c, \gamma, b) - y^m\right) - \frac{\gamma^2}{2\sigma^2}\right)\right]. \quad (6.17)$$

La loi conditionnelle pour le paramètre λ_l s'écrit comme

$$f(\lambda_l; \delta, \mu) \propto \frac{1}{\lambda_l^{K_l}} \times \lambda_l^{-1-\delta} \exp\left(-\frac{\mu}{\lambda_l}\right) \propto \lambda_l^{-1-(\delta+K_l)} \exp\left(-\frac{\mu}{\lambda_l}\right) \propto IG(\delta + K_l, \mu). \quad (6.18)$$

Pour le vecteur d'hyperparamètres Φ_e, il faut calculer les lois conditionnelles à partir desquelles il est possible d'échantillonner en fonction de la vraisemblance et des *a priori* adoptés.

La loi conditionnelle pour l'hyperparamètre λ_b s'écrit comme

$$f(\lambda_b; , b, \mu, \delta) \propto \lambda_b^{-2-\delta} \exp\left(-\frac{b+\mu}{\lambda_b}\right) \propto IG(\delta + 1, b + \mu) \quad (6.19)$$

La loi conditionnelle pour l'hyperparamètre σ^2 s'écrit comme

$$f(\sigma^2; \mu, \gamma, \delta) \propto (\sigma^2)^{-1-\delta} \exp\left(-\frac{\gamma^2 + 2\mu}{2\sigma^2}\right) \propto IG(\delta, \gamma + 2\mu). \quad (6.20)$$

Nous présentons ici un schéma d'échantillonnage qui permet d'échantillonner les poids du modèle W ainsi que les paramètres de la fonction *MMeLU* proposée. Nous utilisons une distance euclidienne comme mesure générique $\mathcal{L}$ dans nos expériences. De plus, l'échantillonnage de W sera effectué en utilisant la méthode de Monte-Carlo Hamiltonien non-lisse (*ns-HMC*) que nous avons présentée dans le chapitre précédent.

L'échantillonneur de Gibbs résultant est résumé dans l'algorithme 4.

Algorithm 4: Algorithme *MMeLU* proposé.

- Fixer les hyperparamètres Φ_m ;
for $r = 1, \ldots, S$ **do**
    * Échantillonner c selon $f(c; W, b, \gamma)$;
    * Échantillonner γ selon $f(\gamma; W, b, c, \sigma^2)$;
    * Échantillonner b selon $f(b; W, c, \gamma, \lambda_b)$;
    * Échantillonner σ^2 selon $f(\sigma^2; \mu, \gamma, \delta)$;
    * Échantillonner λ_b selon $f(\lambda_b; b, \mu, \delta)$;
    * Échantillonner λ_l selon $f(\lambda_l; \delta, \mu)\ \forall\ l \in \{1, ..., L\}$;
    * Échantillonner W comme décrit dans le chapitre 5 ;
end

Table 6.1: Ensembles de données utilisés.

Base de données	Base d'apprentissage	Base de test	Classes
Classification des images *TDM*	566	180	2
Fashion-MNIST	48000	12000	10
CIFAR-10	50000	10000	10

6.5 Validation expérimentale

6.5.1 Classification transversale

Trois (3) expériences de classification d'images sont menées à l'aide de trois ensembles de données : un ensemble de données COVID-19 comprenant des images *TDM* [290], et deux ensembles de données standard, à savoir *Fashion-MNIST* [291] et *CIFAR-10* [292]. Le tableau 6.1 illustre les détails des paramètres des différents ensembles de données en terme de nombre d'images utilisées pour l'apprentissage et pour le test.

Pour des fins de comparaison, huit des fonctions d'activation les plus connues sont utilisées avec l'optimiseur Adam [81] (taux d'apprentissage égal à 10^{-3}) : *ReLU* [153], *LReLU* [160], *ELU* [163], *PReLU* [174], *SELU* [307], [308], *swish* [175], *FreLU* [173] et *MeLU* [179].

6.5.1.1 Modèles ConvNet

En s'appuyant sur les mêmes architectures décrites dans le chapitre précédent, nous utilisons trois architectures *CNN*. Le premier réseau a trois couches convolutives (Conv-32, Conv-64 et Conv-128). La deuxième architecture est composée de neuf couches (3

Table 6.2: Architectures *CNN* utilisées.

CNN_1	CNN_2
Conv3x3-32:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.2)	3 X Conv3x3-32:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.3)
Conv3x3-64:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.3)	3 X Conv3x3-64:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.3)
Conv3x3-128:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.4)	3 X Conv3x3-128:stride=1 BatchNormalization MaxPool 2x2 Dropout(0.4)
Flatten	Flatten
FC-64 Dropout(0.3)	FC-128 Dropout(0.35) FC-64 Dropout(0.35)
FC-softmax	FC-softmax

$\times$ Conv-32, 3 $\times$ Conv-64 et 3 $\times$ Conv-128). Ces architectures sont présentées dans le tableau 6.2.
La troisième est un *CNN* plus profond utilisé à des fins de comparaison comprenant 25 couches convolutives (voir section 6.5.1.6). Chaque architecture utilise des couches convolutives avec un stride size de 1 et des filtres de dimension 3 $\times$ 3, en plus de max-pooling de dimension 2 $\times$ 2. Trois stratégies de régularisation ont été utilisées : normalisation par lots, régularisation ℓ_1 et le dropout.

6.5.1.2 Résultats d'échantillonnage

Après avoir utilisé la méthode d'optimisation bayésienne proposée pour entraîner le *CNN* détaillé ci-dessus dans la classification d'images Covid-19 aux rayons X, nous avons analysé le comportement de convergence. La figure 6.3 présente les chaînes

d'échantillonnage pour les paramètres γ, b et c de la fonction *MMeLU* proposée (a-c), ainsi que les histogrammes des échantillons correspondants (d-f). Les chaînes d'échantillonnage et les histogrammes des coefficients échantillonnés confirment les bonnes propriétés de convergence de l'échantillonneur de Gibbs utilisé.

6.5.1.3 Expérience 1: Classification COVID-19

La présente section examine l'efficacité de notre méthode pour classer les infections provoquées par la maladie causées par le Covid-19 et d'autres pneumonies à partir des données *TDM* (voir section 5.6.1.4 du chapitre 5).

Le tableau 6.3 montre que la méthode bayésienne proposée est nettement supérieure à toutes les fonctions d'activation concurrentes pour les deux architectures CNN_1 et CNN_2. Les résultats indiquent également que le temps de calcul est plus court avec le modèle proposé. De plus, même en adoptant diverses techniques de régularisation, des baisses importantes de performance sont constatées pour toutes les fonctions concurrentes. Cela est principalement dû à la difficulté de classifier les images de TDM en raison de la richesse de leur contenu, ainsi que de la similitude entre les images d'infections par le Covid-19 et celles d'autres types de pneumonies. En les comparant avec l'optimiseur proposé *ns-HMC*, les résultats de précision et de pertes obtenus sur ce jeu de données ont révélé une nette supériorité de *MMeLU*, qui a atteint un taux de précision allant jusqu'à 90% pour le CNN_1 avec un temps de calcul d'environ 46 minutes, tandis que *ns-HMC* n'a atteint que 84% en un temps légèrement réduit de 40 minutes (Tableau 5.6.1.4 du chapitre 5). Les mêmes niveaux de performance ont également été atteints avec le CNN_2.

Les figures 6.4 et 6.5 illustrent le comportement des algorithmes en montrant une amélioration significative de la précision, observée avec la plupart des fonctions d'activation. En comparaison avec l'approche bayésienne proposée *MMeLU*, il existe une différence marquée entre les courbes de précision et de perte pour toutes les méthodes concurrentes. Par exemple, la fonction *LReLU* introduit un biais négatif qui supprime les activations excessives, mais une valeur de biais inappropriée peut conduire à un sous-apprentissage. La fonction *ELU* permet des activations négatives, mais sa forme exponentielle peut entraîner une explosion de la valeur d'activation pour les grandes valeurs de x. La fonction Swish accélère la convergence de l'apprentissage, mais elle peut conduire à un sur-apprentissage en étant plus sensible aux valeurs aberrantes. Enfin, la fonction *FReLU* peut capturer des motifs de données complexes, mais elle

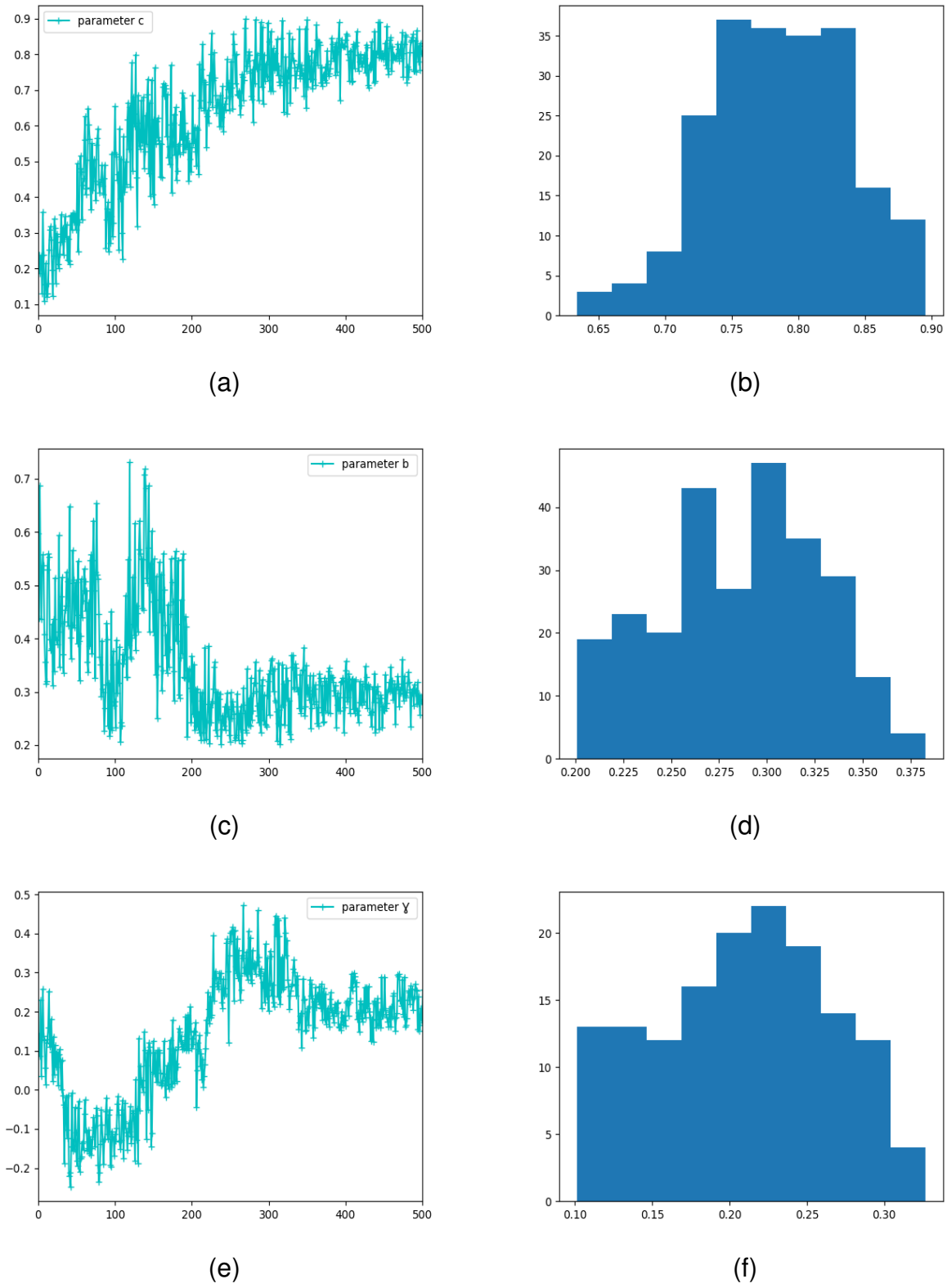


Figure 6.3: Échantillonnage des paramètres c (a,b), b (c,d) et γ (e,f) : chaînes et histogrammes.

Table 6.3: Expérience 1 : résultats de la classification CT avec CNN_1 et CNN_2 (fonctions d'activation (Fcts Act), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

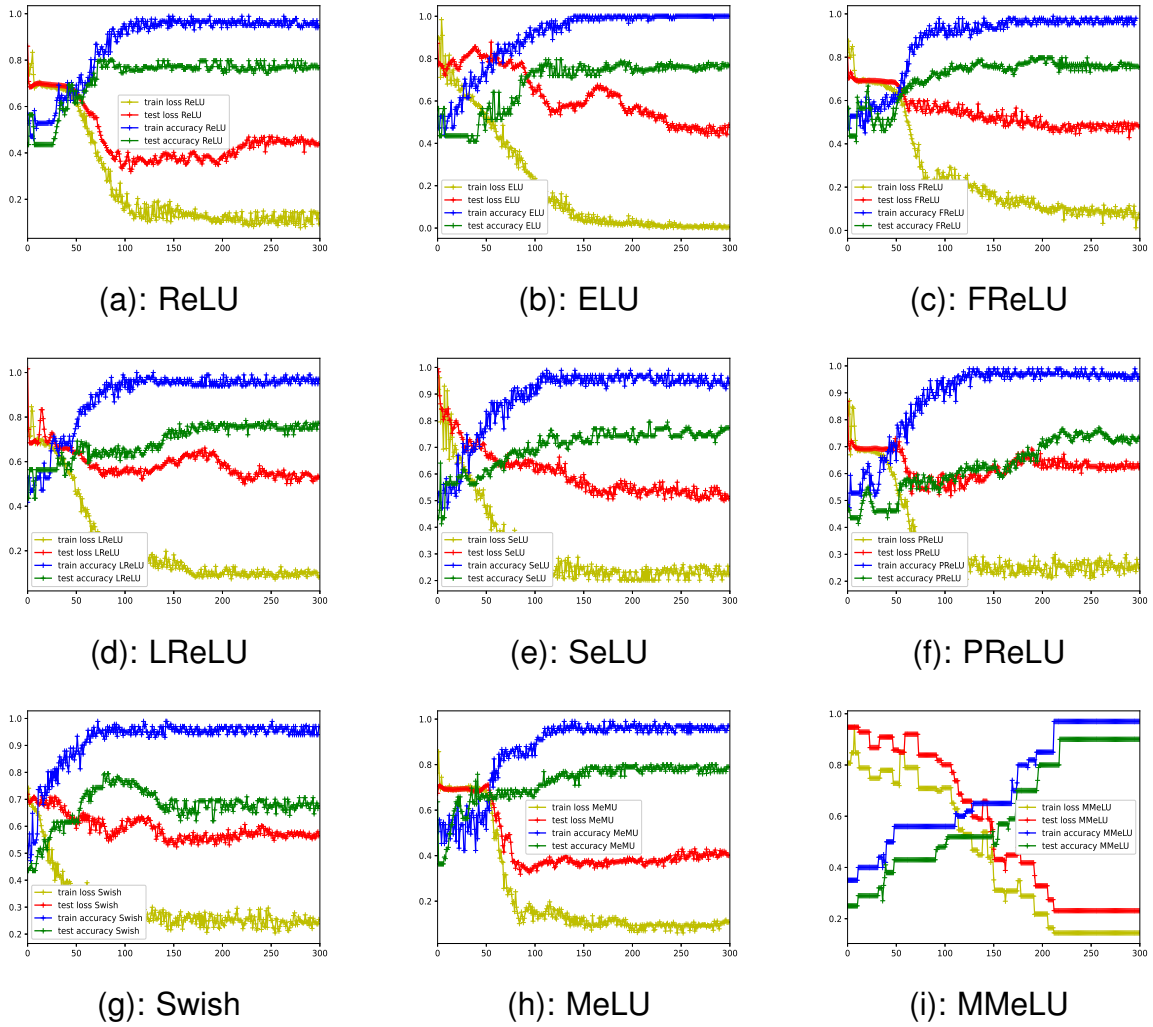
	CNN_1					CNN_2				
Fcts Act.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
MMeLU	**46.21**	**0.90**	**0.23**	**0.87**	**0.86**	**61.92**	**0.91**	**0.21**	**0.87**	**0.87**
ReLU	58	0.73	0.43	0.69	0.68	81	0.77	0.39	0.74	0.72
LReLU	65.18	0.73	0.52	0.71	0.69	105	0.78	0.44	0.76	0.75
ELU	63	0.75	0.47	0.75	0.74	97	0.76	0.46	0.75	0.75
PReLU	71.28	0.68	0.72	0.64	0.62	119	0.70	0.76	0.68	0.67
SeLU	64.75	0.77	0.78	0.74	0.72	107	0.76	0.69	0.74	0.73
Swish	83.41	0.68	0.58	0.65	0.62	132	0.73	0.55	0.71	0.70
FReLU	77.8	0.76	0.59	0.76	0.75	123	0.77	0.52	0.76	0.75
MeLU	95.89	0.77	0.43	0.77	0.76	146	0.80	0.38	0.80	0.80

peut également souffrir de surapprentissage si les paramètres ne sont pas bien choisis. Ces différences remarquables confirment la puissance et l'efficacité de notre fonction *MMeLU*. De plus, les résultats obtenus mettent en évidence un comportement similaire des deux architectures de *CNN* étudiées.

6.5.1.4 Expérience 2: Classification d'images Fashion-MNIST

Les performances d'apprentissage des algorithmes d'optimisation concurrents sont évaluées dans ce scénario avec la base standard *Fashion-MNIST* (plus de détails dans la section 5.6.1.5 du chapitre 5).

Le tableau 6.4 aborde les valeurs obtenues pour l'ensemble de données *Fashion-MNIST*. Toutes les fonctions d'activation concurrentes ont mal performé sur cet ensemble de données, probablement en raison de sa petite taille. Notre méthode proposée a surpassé toutes les autres fonctions d'activation concurrentes, avec une précision remarquable allant jusqu'à 92% pour les deux *CNN*. De plus, étant envisagé par le tableau 6.4, le temps de traitement des méthodes concurrentes est généralement d'environ 140 minutes pour CNN_1, alors que seulement 100 minutes sont nécessaires pour notre fonction *MMeLU*. De même, l'architecture profonde CNN_2 a abouti à la même conclusion. Cela démontre clairement que notre approche permet d'obtenir un excellent résultat en termes de précision et une réduction considérable du temps de traitement. De même, elle a également montré une légère supériorité par rapport à l'optimiseur *ns-HMC* (voir tableau 5.6.1.5, chapitre 5) pour les deux architectures *CNN* utilisées.



(a): ReLU (b): ELU (c): FReLU

(d): LReLU (e): SeLU (f): PReLU

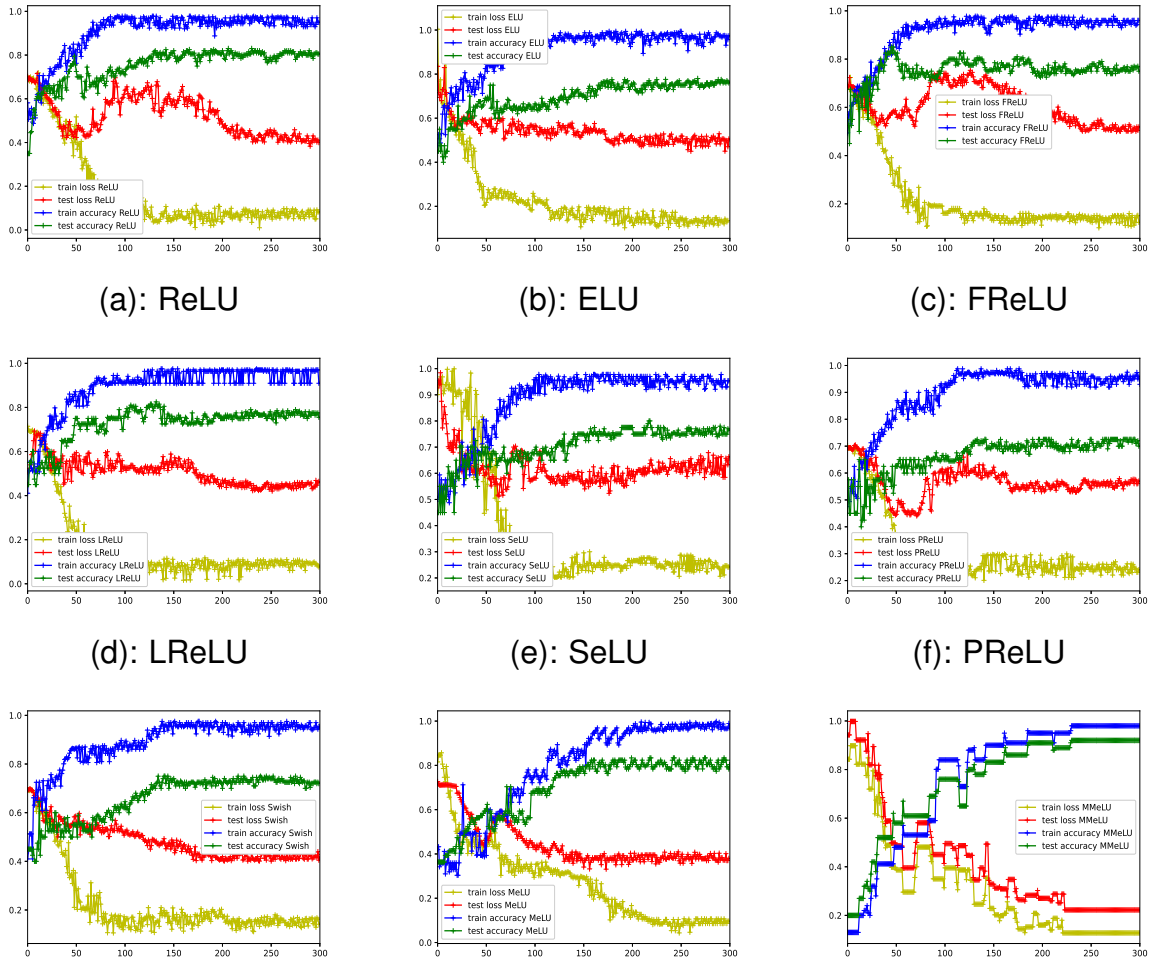
(g): Swish (h): MeLU (i): MMeLU

Figure 6.4: Expérience 1 : Courbes d'entraînement et de test à l'aide de CNN_1.

6.5.1.5 Expérience 3: Classification d'images CIFAR-10

Dans ce paragraphe, les performances d'apprentissage de la méthode proposée sont évaluées en utilisant la base standard *CIFAR-10* (section 5.6.1.6 du chapitre 5).

Les résultats de classification pour l'ensemble de données *CIFAR-10* sont présentés dans le tableau 6.5. Le modèle bayésien suggéré s'est globalement bien comportée même lorsque plusieurs classes sont utilisées. Toutes les fonctions concurrentes d'activation ont une précision similaire, environ 88%, avec un taux de perte presque double que celui du *MMeLU* proposé pour les deux architectures. De plus, la fonction d'activation *MMeLU* prend significativement moins de temps pour l'entraînement que les autres approches qui sont souvent complexes et prennent beaucoup de temps



(a): ReLU (b): ELU (c): FReLU

(d): LReLU (e): SeLU (f): PReLU

(g): Swish (h): MeLU (i): MMeLU

Figure 6.5: Expérience 1 : Courbes d'entraînement et de test à l'aide de CNN_2.

avant la convergence. Les résultats prouvent que la fonction *MMeLU* proposée offre de meilleures performances d'apprentissage sur des bases de données standards telles que *CIFAR-10*. Les mêmes conclusions peuvent être tirées en examinant les résultats de l'optimiseur *ns-HMC* sur cette base de données (tableau 5.6.1.6 du chapitre 5).

6.5.1.6 Expérience 4: Comparaison sur un CNN profond

Ce paragraphe étudie les performances de notre algorithme en utilisant la base *Fashion-MNIST* avec un *CNN* profond. Ce *CNN* est plus profond que CNN_1 et CNN_2. Il comporte 5 $\times$ Conv3x3-32, 5 $\times$ Conv3x3-64, 5 $\times$ Conv3x3-128, 5 $\times$ Conv3x3-256 et 5 $\times$ Conv3x3-512. Toutes les couches convolutives utilisent des filtres de taille 3 et sont

Table 6.4: Expérience 2 : Résultats de la classification Fashion-MNIST avec CNN_1 et CNN_1 (fonctions d'activation (Fcts Act), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

	CNN_1					CNN_2				
Fcts Act.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
MMeLU	**111.6**	**0.93**	**0.20**	**0.90**	**0.89**	**319.8**	**0.94**	**0.19**	**0.92**	**0.91**
ReLU	145.4	0.90	0.38	0.85	0.82	421.3	0.91	0.32	0.88	0.87
LReLU	157.2	0.86	0.36	0.85	0.83	409.7	0.87	0.34	0.85	0.84
ELU	154.5	0.87	0.33	0.86	0.84	389	0.88	0.32	0.86	0.85
PReLU	158.7	0.87	0.32	0.86	0.85	400.3	0.85	0.33	0.83	0.82
SeLU	150	0.85	0.34	0.84	0.83	383.8	0.87	0.33	0.86	0.85
Swish	148.5	0.86	0.36	0.83	0.81	423	0.87	0.35	0.84	0.82
FReLU	149.8	0.87	0.35	0.85	0.83	395.7	0.86	0.34	0.85	0.84
MeLU	169	0.89	0.33	0.86	0.84	452.5	0.89	0.31	0.88	0.87

Table 6.5: Expérience 3 : Résultats de la classification CIFAR-10 avec CNN_1 et CNN_2 (fonctions d'activation (Fcts Act), temps de calcul en minutes, précision (Pres), perte, sensibilité (Sens) et spécificité (Spec)).

	CNN_1					CNN_2				
Fcts Act.	Temps	Pres.	Perte	Sens.	Spec.	Temps	Pres.	Perte	Sens.	Spec.
MMeLU	**120.7**	**0.91**	**0.21**	**0.89**	**0.88**	**332.5**	**0.93**	**0.20**	**0.90**	**0.88**
ReLU	161	0.87	0.42	0.83	0.81	429	0.90	0.36	0.87	0.86
LReLU	170	0.87	0.52	0.85	0.84	437.9	0.85	0.55	0.83	0.81
ELU	165	0.88	0.41	0.86	0.85	438	0.88	0.39	0.86	0.85
PReLU	198.2	0.82	0.48	0.84	0.82	466.8	0.84	0.47	0.81	0.79
SeLU	173.6	0.84	0.43	0.85	0.84	454.3	0.86	0.39	0.87	0.86
Swish	209.7	0.83	0.49	0.82	0.79	478.3	0.85	0.46	0.83	0.81
FReLU	172.3	0.84	0.41	0.82	0.81	431.7	0.86	0.38	0.83	0.81
MeLU	219.9	0.86	0.40	0.83	0.83	485.6	0.90	0.34	0.88	0.87

suivies d'une opération de max-pooling avec une fenêtre de 2, avec un stride size égal à 1.

Les résultats montrent que notre méthode *MMeLU* maintient de bonnes performances sur cette architecture profonde, comme le montre le tableau 6.6. Les mêmes conclusions peuvent être tirées que des expériences précédentes. Il convient également de noter que l'utilisation de fonctions d'activation concurrentes telles que *LReLU*, *PReLU* et *SELU* conduit à un surapprentissage de cette architecture, contrairement à notre méthode. Cela confirme la performance globale élevée de *MMeLU* pour différents niveaux de profondeur.

Table 6.6: Résultats de la classification Fashion-MNIST avec un CNN profond (fonctions d'activation (Fcts Act), temps de calcul en minutes, précision, perte, sensibilité et spécificité).

Fcts Act.	Temps (min)	Prec.	Perte	Sens.	Spec.
MMeLU	**651.4**	**0.94**	**0.19**	**0.92**	**0.92**
ReLU	854	0.89	0.34	0.87	0.85
LReLU	847.9	0.86	0.38	0.84	0.82
ELU	870	0.87	0.35	0.85	0.84
PReLU	855.6	0.86	0.36	0.83	0.83
SeLU	867.3	0.88	0.33	0.87	0.86
Swish	880	0.87	0.34	0.86	0.85
FReLU	873.4	0.85	0.36	0.84	0.82
MeLU	890	0.90	0.32	0.86	0.85

6.5.2 Classification longitudinale

Dans ce paragraphe, nous évaluons notre méthode proposée en classifiant des séries temporelles d'images pour déterminer l'évolution du COVID-19 (survie ou décès) et de la maladie d'Alzheimer (démence ou non démence).

Nous adoptons la même architecture ProgNet précédemment détaillée dans les chapitres 4 et 5 pour l'analyse longitudinale d'images, avec l'utilisation de notre fonction d'activation proposée *MMeLU*.

À titre de comparaison, le modèle d'optimisation *ns-HMC* et la technique d'optimisation la plus populaire utilisée en Deep Learning, *Adam* (avec un taux d'apprentissage égal à 10^{-3}), sont adoptés.

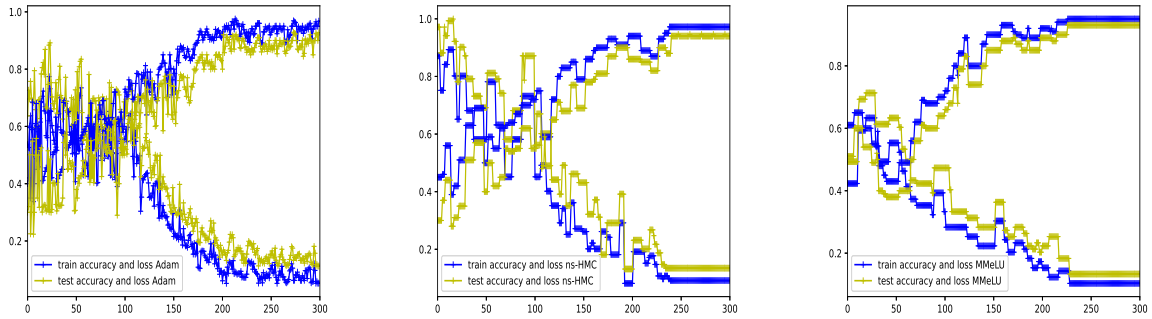
6.5.2.1 Expérience 1 : pronostic COVID-19

Comme expliqué dans les chapitres précédents, la base temporelle COVID-19 contient 51 séquences d'entraînement et 16 séquences de test, chacune avec une séquence de 3 images radiographiques $T = (T_1, T_2, T_3)$.

Évaluation de la perte et de la précision

En se référant à la figure 6.6, on peut observer que les courbes d'entraînement et de test des trois méthodes, à savoir *Adam*, *ns-HMC* et *MMeLU*, présentent des comportements similaires sur cet ensemble d'images.

Évaluation quantitative



(a): Prognet-Adam (b): Prognet-ns-HMC (c): Prognet-MMeLU

Figure 6.6: Courbes d'entraînement et de test : pronostoc COVID-19,

Les valeurs présentées dans le tableau 6.7 suggèrent une légère supériorité de la méthode proposée *MMeLU*. L'avantage du taux minimal de faux négatifs et faux positifs, estimée à environ 4%, est particulièrement remarquable et rend *MMeLU* très intéressante pour traiter efficacement les séquences d'évolutions les plus complexes associés au COVID-19.

Comme dans le chapitre précédent, les moyennes présentées dans le tableau 6.8 ont été calculées à partir de 10 exécutions distinctes, chacune impliquant une sélection aléatoire de sous-ensembles de données d'entraînement à partir de la base pronostic COVID-19.

En outre, les résultats obtenus dans le tableau 6.8 sont très convaincants et démontrent que la *MMeLU* surpasse les autres algorithmes en termes de stabilité, comme en témoignent les faibles valeurs de variance obtenues. En effet, cette méthode parvient d'améliorer considérablement la précision globale tout en ayant un temps de calcul encore plus court que l'optimiseur *Adam*.

Table 6.7: Valeurs VP, FN, FP et VN pour la méthode MMeLU proposée, et les deux optimiseurs ns-HMC et Adam pour la base COVID-19.

	VP	**FN**	**FP**	**VN**
ProgNet-Adam	92.51%	6.15%	6.09%	92.32%
ProgNet-ns-HMC	94.14%	4.34%	4.21%	94.47%
ProgNet-MMeLU	**96.53%**	**4.06%**	**3.68%**	**95.83%**

Analyse qualitative

Il est clairement démontré par les résultats présentés dans la figure 6.7 que la fonction

Table 6.8: Exactitude, précision, rappel, temps de calcul (en minutes) et F_1-score de la méthode proposée MMeLU ainsi que ns-HMC et Adam pour la base COVID-19.

	ProgNet-Adam	**ProgNet-ns-HMC**	**ProgNet-MMELU**
Exactitude	0.92 ± 0.03	0.94 ± 0.04	**0.96** ± 0.03
Precision	0.92 ± 0.04	0.94 ± 0.03	**0.96** ± 0.05
Rappel	0.91 ± 0.05	0.92 ± 0.04	**0.94** ± 0.04
F_1-score	0.91 ± 0.04	0.92 ± 0.03	**0.94** ± 0.05
Temps (min)	218.2 ± 0.05	138.9 ± 0.03	**147** ± 0.04

d'activation *MMeLU* est performante, atteignant une précision remarquable de 98%. En ce qui concerne la détection des cas de décès, *MMeLU* a obtenu de meilleurs résultats que les autres algorithmes. Cela est particulièrement remarquable et ce compte tenu de la grande quantité de régions blanches présentes dans les images, ce qui rend généralement difficile l'identification des cas de décès. Malgré cela, la méthode proposée a réussi à faire la distinction avec une grande précision en utilisant une fonction d'activation configurable.

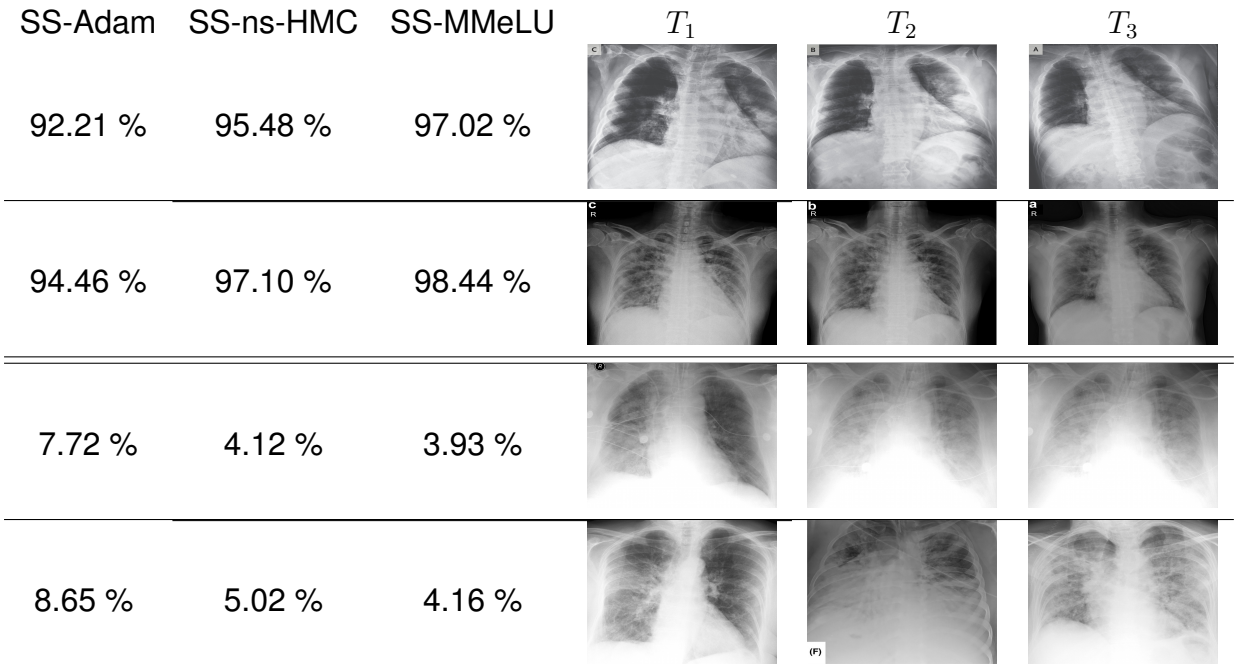


Figure 6.7: Visualisation de la classification COVID-19 à l'aide de la fonction d'activation entraînable proposée, l'optimiseur ns-HMC et Adam.

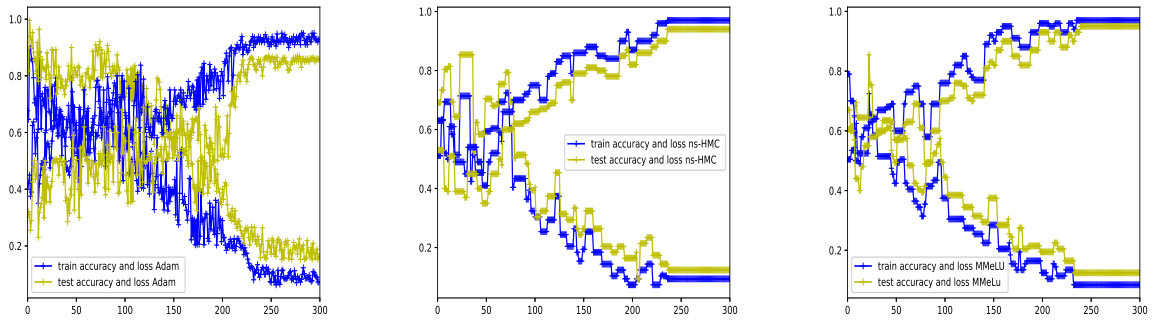
6.5.2.2 Expérience 2: pronostic Alzheimer

Comme indiqué dans la section 5.6.2.2 du chapitre 5, nous présentons les performances de la méthode proposée à l'aide d'une base de données liée la maladie d'Alzheimer. Deux catégories principales sont traitées : démence et non démence.

Évaluation de la perte et de la précision

Les informations présentées sur les courbes de précision et de perte de la figure 6.8 révèlent clairement que les méthodes proposées *ns-HMC* et *MMeLU* sont plus performantes que l'optimiseur *Adam*. À mesure que le taux d'apprentissage augmente, on observe une amélioration du score obtenu avec ces deux méthodes, ce qui indique une capacité accrue à généraliser les modèles entraînés.

Ces résultats démontrent que *ns-HMC* et *MMeLU* peuvent être utilisées pour améliorer les performances des modèles d'apprentissage dans des applications complexes. De plus, leur capacité à s'adapter à des données complexes et à atteindre un niveau d'apprentissage optimal en un minimum de temps est remarquable.



(a): Prognet-Adam (b): Prognet-ns-HMC (c): Prognet-MMeLU

Figure 6.8: Courbes d'entraînement et de test : pronostoc Alzheimer,

Évaluation quantitative

Le tableau 6.10 fournit les moyennes des valeurs sur 10 exécutions ainsi que les écarts-types correspondants. L'analyse quantitative des données fournit une preuve solide de l'excellence du modèle proposé *MMeLU*, qui est clairement visible à travers les scores présentés dans les tableaux 6.9 et 6.10. En particulier, les valeurs de précision produites par *MMeLU* sont élevées, même pour les cas difficiles qui ont été mal interprétés par les approches concurrentes.

Table 6.9: Valeurs VP, FN, FP et VN pour la méthode MMeLU proposée.

	VP	**FN**	**FP**	**VN**
ProgNet-Adam	86.12%	9.42%	8.29%	86.71%
ProgNet-ns-HMC	93.79%	5.81%	5.05%	93.54%
ProgNet-MMeLU	**96.19%**	**4.22%**	**3.94%**	**95.94%**%

Table 6.10: Exactitude, précision, rappel, temps de calcul (en minutes) et F_1-score de la méthode proposée MMeLU ainsi que ns-HMC et Adam pour la base d'Alzheimer.

	ProgNet-Adam	**ProgNet-ns-HMC**	**ProgNet-MMeLU**
Exactitude	0.86 ± 0.07	0.93 ± 0.04	**0.95** ± 0.05
Precision	0.85 ± 0.11	0.93 ± 0.06	**0.95** ± 0.04
Rappel	0.85 ± 0.10	0.93 ± 0.05	**0.95** ± 0.03
F_1**-score**	0.85 ± 0.09	0.93 ± 0.04	**0.95** ± 0.04
Temps (min)	327.4 ± 0.04	210.9 ± 0.03	**221.4** ± 0.05

Analyse qualitative

Nous présentons quelques résultats représentatifs de séries chronologiques de patients souffrant ou non de démence à partir de quatre images. La figure 6.9 affiche quatre séries temporelles de patients atteints de la maladie d'Alzheimer.

Pour chaque patient, trois scores sont fournis pour *Adam*, *ns-HMC* et le modèle *MMeLU*. Une analyse visuelle des résultats obtenus désigne la précision élevée de la fonction d'activation entraînable à l'aide de l'optimiseur proposé *ns-HMC*.

6.6 Discussion

Nous avons proposé une nouvelle fonction d'activation entraînable. Nous avons mené des expériences en utilisant deux types de classification : transversale et longitudinale. Dans la première partie, nous avons utilisé trois bases d'images : COVID-19, *Fashion-MNIST* et *CIFAR-10*. Pour la classification longitudinale, nous avons recouru à deux bases temporelles pour l'évolution du COVID-19 et de la maladie d'Alzheimer. Nous avons également comparé notre fonction d'activation avec neuf autres fonctions d'activation les plus connues dans l'état de l'art. Pour l'analyse des séries temporelles, nous avons comparé la performance de notre fonction d'activation proposée avec deux optimiseurs : *ns-HMC* et *Adam*.

Les résultats ont montré que notre fonction d'activation proposée, associée à l'optimiseur

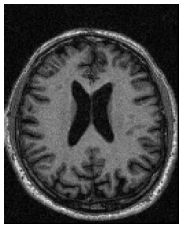
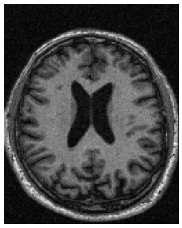
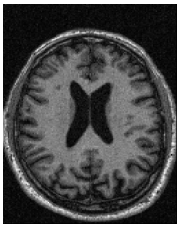
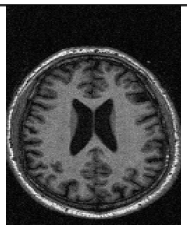
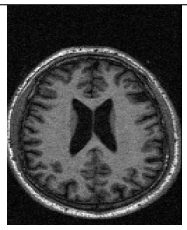
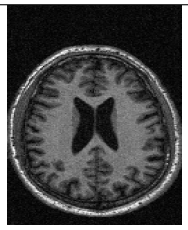
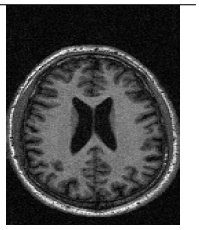
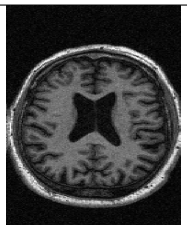
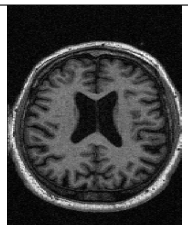
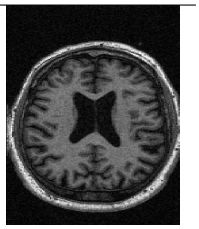
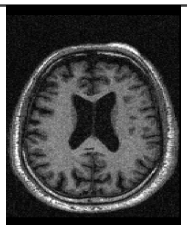
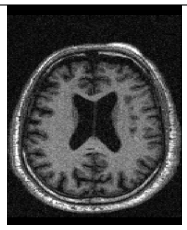
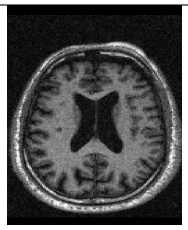
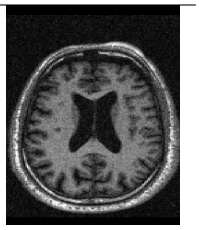
SS-Adam	SS-ns-HMC	SS-MMeLU	T_1	T_2	T_3	T_4
85.21 %	91.36 %	94.76 %				
90.64 %	96.52 %	98.45 %				
9.76 %	6.80 %	4.35 %				
11.08 %	7.92 %	5.09 %				

Figure 6.9: Visualisation de la classification d'Alzheimer à l'aide de la fonction d'activation proposée, ns-HMC et Adam.

bayésien *ns-HMC*, permet une convergence plus rapide et plus précise tout en minimisant le surajustement. De plus, elle est efficace pour traiter des données massives avec des garanties de convergence robustes. Nous avons également montré que l'utilisation de notre fonction d'activation *MMeLU* conjuguée avec l'optimiseur bayésien *ns-HMC* entraîne une précision globale plus élevée par rapport à toutes les fonctions concurrentes.

Nous avons constaté que pour le problème difficile de classification, étudié dans l'expérience 1 de la classification transversale, un réseau plus profond n'arrivait pas à résoudre le problème de surajustement avec les optimiseurs standard. Cependant, notre méthode contourne cette limitation en offrant une optimisation plus précise du

critère cible et en ne nécessitant qu'une architecture de réseau neuronal simple pour produire une haute précision. En outre, nous avons prouvé que pour les méthodes concurrentes, il n'y a pas d'activation qui soit systématiquement meilleure que les autres. De plus, nous avons comparé les performances de notre fonction d'activation proposée avec deux optimiseurs (*ns-HMC* et *Adam*) pour l'analyse des séries temporelles, où les résultats ont montré une performance globale élevée par rapport à *Adam*. Ces résultats suggèrent que notre fonction d'activation proposée peut être utilisée pour résoudre efficacement des problèmes liés aux séries temporelles.

En conclusion, notre fonction d'activation proposée offre une alternative prometteuse pour l'entraînement des réseaux de neurones profonds, avec des garanties de convergence robustes et des performances améliorées. Les résultats obtenus suggèrent l'application de cette méthode sur d'autres tâches d'apprentissage automatique avec des résultats similaires.

6.7 Conclusion

Dans ce chapitre, nous avons présenté une approche bayésienne pour un réseau parcimonieux avec des fonctions d'activation entraînables pour ajuster les poids et tous les paramètres de la fonction *MMeLU* proposée. Par rapport à tous les autres algorithmes concurrents, y compris les fonctions d'activation fréquemment utilisées, la technique suggérée a donné des résultats prometteurs avec une meilleure efficacité du modèle de classification ; des propriétés de généralisation élevées, et un temps de calcul réduit.

7

Conclusion

Sommaire

7.1 Introduction

L'imagerie médicale permet de visualiser les organes du corps humain dans le but de diagnostiquer, suivre et traiter des problèmes médicaux, tels que la détection et le suivi de la progression des pathologies. Il est essentiel de détecter rapidement les maladies pour améliorer les chances de survie des patients. L'imagerie médicale est considérée comme une révolution pour les médecins, car elle permet de détecter des maladies telles que les tumeurs à un stade précoce.

Plusieurs techniques d'acquisition d'imagerie médicale sont disponibles, notamment l'imagerie par résonance magnétique, la tomodensitométrie et la tomographie par émission de positrons. Ces techniques sont moins invasives et moins coûteuses pour le patient.

Au cours de la dernière décennie, la quantité de données d'images longitudinales accessibles a considérablement augmentée. Les données longitudinales sont collectées à partir d'au moins deux scans. Elles sont courantes dans les milieux biomédicaux et cliniques, ainsi que dans d'autres domaines de recherche. Ces ensembles de données comportent des séries temporelles pour de multiples utilisations, comme le suivi de l'évolution de la pathologie pour les analyses médicales et l'évaluation d'un objet pour l'analyse d'images satellitaires. L'appréciation de l'utilité de l'imagerie longitudinale est en constante augmentation en raison de son importance pour fournir des résultats plus fiables et précis.

L'analyse d'images longitudinales peut être difficile en raison du coût élevé et de la complexité de la réalisation d'études longitudinales. Les erreurs de corrélation entre les mesures répétées peuvent conduire à une variabilité d'échantillon mal estimée et à des conclusions scientifiques trompeuses. Par conséquent, le but de cette présente thèse a été de résoudre ces problèmes liés à la classification d'images pour des séries temporelles.

Dans ce cadre, notre première contribution a consisté à proposer une méthode pour l'analyse temporelle d'images au travers une approche d'apprentissage profond (chapitre 4 du manuscrit). Elle se base principalement sur une association d'un réseau *CNN* et *RNN* pour l'extraction et la mémorisation des caractéristiques les plus importantes entre les séquences.

Les méthodes développées par la suite visent à améliorer le temps de traitement, à augmenter le taux de précision des modèles et à assurer une bonne gestion de la corrélation entre les séquences d'images. Pour ce faire, nous nous sommes concentrés principalement sur deux étapes essentielles de l'architecture d'apprentissage proposée : l'optimisation et la fonction d'activation. Il s'agit de deux éléments primordiaux dans un réseau de neurones artificiels qui ont une incidence directe sur les résultats. Elles ont fait l'objet de discussions approfondies dans le chapitre 2 en exposant les principales méthodes examinées dans l'état de l'art, ainsi que les avantages et les inconvénients de chacune. Les conclusions tirées à la fin de ce chapitre nous ont permis de proposer un schéma d'optimisation efficace (chapitre 5 de la thèse) formulé dans un cadre bayésien pour ajuster les poids des réseaux. Dans le chapitre 6, nous avons étendu la méthode d'optimisation au travers une fonction d'activation entraînable en utilisant la méthode MCMC pour l'estimation des paramètres.

Les méthodes proposées ont été validées à l'aide de deux ensembles de données transversales standards, deux bases de données pour la classification du COVID-19, et deux bases de données de suivi de la progression du COVID-19 et de la maladie d'Alzheimer. Les résultats expérimentaux ont montré que les approches proposée sont très prometteuses pour la classification d'images dans les séries temporelles. La méthode d'optimisation et la fonction d'activation proposés ont amélioré les performances des modèles et ont permis de mieux gérer la corrélation entre les images d'une séquence donnée.

Pour résumer, trois contributions majeures ont été proposées dans cette thèse.

Contribution 1 :

Nous avons présenté une méthode de prédiction de l'évolution de la pathologie pulmonaire due au Covid-19 en utilisant des architectures d'apprentissage profond. Notre approche combine un réseau de neurones convolutifs et récurrents pour classifier des images radiographiques thoraciques multi-temporelles. Le CNN agit comme un détecteur de caractéristiques entraînables en apprenant des fonctions convolutionnelles puissantes pour agir sur les images spatiales d'entrée, tandis que le RNN reçoit une séquence de ces représentations de haut niveau et évalue l'évolution temporelle des images. L'architecture proposée permet de classer les séries chronologiques en deux catégories : évolution positive (survie) et évolution négative (décès). une telle classifica-

tion aide les médecins à prédire les pronostics importants pour les patients dans des situations graves.

Contribution 2 :

Plusieurs études ont été menées pour améliorer les méthodes d'optimisation existantes dans les approches d'apprentissage telles que les méthodes basées sur les gradients. L'un des plus grands défis lors de l'utilisation d'une base de données d'apprentissage annotée, est l'optimisation des poids du modèle. La régularisation est couramment utilisée pour des raisons d'efficacité.

L'utilisation d'une régularisation non lisse pour favoriser des réseaux parcimonieux, telle que la norme ℓ_1, peut rendre l'optimisation difficile en raison de la non-différentiabilité du critère cible. Nous avons proposé un schéma d'optimisation formulé dans un cadre bayésien, basé sur des méthodes *MCMC* et une dynamique hamiltonienne, pour résoudre des problèmes d'optimisation parcimonieux. L'objectif principal est de minimiser la fonction de coût du modèle d'apprentissage en ajustant les hyperparamètres. La méthode hamiltonienne non lisse peut créer des schémas d'échantillonnage rapides et efficaces tout en traitant des fonctions d'énergie non différentiables.

Contribution 3 :

Au cours des dernières années, l'intérêt pour l'exploration des fonctions d'activation dans les réseaux de neurones profonds a augmenté en raison de leur potentiel pour améliorer les performances du réseau. Un modèle bayésien a été proposé pour estimer les poids et les paramètres d'une fonction d'activation entraînable pour des réseaux de neurones convolutifs. Cette approche étend une méthode d'optimisation qui utilise la méthode hamiltonienne non lisse (*ns-HMC*) pour ajuster des réseaux de neurones artificiels parcimonieux. La fonction d'activation proposée est une version modifiée de la fonction *MeLU* avec peu de paramètres à estimer. Cette méthode peut aider à éviter les problèmes de surapprentissage et à améliorer le temps de convergence, ce qui permet d'améliorer les performances globales du réseau.

7.2 Perspective

En ce qui concerne les perspectives, cette thèse propose plusieurs possibilités à explorer, tant au niveau de la méthodologie que des applications.

Exploration des réseaux parcimonieux pour l'apprentissage supervisé et faiblement supervisé :

Les mécanismes d'attention et les réseaux génératifs sont des approches prometteuses pour l'apprentissage faiblement supervisé, car ils permettent une exploration plus efficace de l'espace latent des données, offrent une meilleure interprétabilité des modèles appris et peuvent être utilisés pour générer de nouvelles données. Les réseaux parcimonieux peuvent être utilisés pour améliorer davantage ces approches, en réduisant la complexité des modèles et en permettant une utilisation plus efficace des ressources informatiques. À l'avenir, le développement de réseaux parcimonieux pour les approches utilisant les mécanismes d'attention et les réseaux génératifs pour l'apprentissage faiblement supervisé pourrait jouer un rôle important dans la résolution de tâches complexes en traitement du langage naturel, vision par ordinateur et autres domaines. Cette perspective représente une opportunité passionnante pour la recherche en intelligence artificielle, offrant la possibilité d'améliorer considérablement les performances des modèles d'apprentissage automatique tout en réduisant leur complexité.

Segmentation des lobes pulmonaires pour les images 3D :

Actuellement, la segmentation lobaire des images *CT* est réalisée manuellement par un radiologiste pulmonaire expérimenté. Cependant, cette tâche est extrêmement laborieuse, nécessitant généralement plusieurs heures par patient. En conséquence, la segmentation manuelle du lobe est rarement utilisée dans la pratique clinique, les cliniciens s'appuyant plutôt sur une inspection visuelle subjective. L'idée est de proposer un système pour une segmentation automatique des lobes pulmonaires et le calcul de leur volume. Des coupes longitudinales des lobes sont ensuite réalisées pour chaque patient afin de suivre l'évolution de la maladie pulmonaire dans le temps, telle que les problèmes des voies respiratoires ou le cancer du poumon. Pour ce problème difficile, il est important d'utiliser des algorithmes d'apprentissage profond performants et flexibles.

Nous allons donc appliquer nos techniques pour résoudre ce problème.

Implémentation GPU :

Nous nous concentrerons prochainement sur l'étude de la mise en œuvre parallèle des méthodes *ns-HMC* et *MMeLU* proposées afin de réduire davantage le temps de calcul.

A

Bibliographie

Sommaire

Références

[1] A. A. Bui and R. K. Taira, *Medical imaging informatics*. Springer Science & Business Media, 2009.

[2] M. Chappell, *Principles of Medical Imaging for Engineers*. Springer, 2019.

[3] E. Picano, "Sustainability of medical imaging," *Bmj*, vol. 328, no. 7439, pp. 578–580, 2004.

[4] J. Holmes, L. Sacchi, R. Bellazzi, *et al.*, "Artificial intelligence in medicine," *Ann R Coll Surg Engl*, vol. 86, pp. 334–8, 2004.

[5] S. Dick, "Artificial intelligence," 2019.

[6] J. C. Gore, *Artificial intelligence in medical imaging*, 2020.

[7] E. R. Ranschaert, S. Morozov, and P. R. Algra, *Artificial intelligence in medical imaging: opportunities, applications and risks*. Springer, 2019.

[8] C. B. Begg, "Advances in statistical methodology for diagnostic medecine ni the 1980s," *Statistics in medicine*, vol. 10, no. 12, pp. 1887–1895, 1991.

[9] A. Jankowski and G. Ferretti, "Tomodensitometrie volumique: principe, parametres," *Revue des maladies respiratoires*, vol. 27, no. 8, pp. 964–969, 2010.

[10] G. Fitzmaurice, N. Laird, and J. Ware, "Applied longitudinal analysis. john wiley & sons," *Inc., Haboken, NJ*, 2004.

[11] C.-C. Chang, "A library for support vector machines," *http://www. csie. ntu. edu. twcjlin/libsvm*, 2001.

[12] B. J. Erickson, P. Korfiatis, Z. Akkus, and T. L. Kline, "Machine learning for medical imaging," *Radiographics*, vol. 37, no. 2, p. 505, 2017.

[13] D. Shen, G. Wu, D. Zhang, K. Suzuki, F. Wang, and P. Yan, "Machine learning in medical imaging.," *Comput. Medical Imaging Graph.*, vol. 41, pp. 1–2, 2015.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] D. Ravi, C. Wong, F. Deligianni, *et al.*, “Deep learning for health informatics,” *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.

[16] P. P. Shinde and S. Shah, “A review of machine learning and deep learning applications,” in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, IEEE, 2018, pp. 1–6.

[17] Y. Li, Z. Hao, and H. Lei, “Survey of convolutional neural network,” *Journal of Computer Applications*, vol. 36, no. 9, p. 2508, 2016.

[18] K. OShea and R. Nash, “An introduction to convolutional neural networks,” *arXiv preprint arXiv:1511.08458*, 2015.

[19] H. H. Aghdam and E. J. Heravi, “Guide to convolutional neural networks,” *New York, NY: Springer*, vol. 10, no. 978-973, p. 51, 2017.

[20] L. R. Medsker and L. Jain, “Recurrent neural networks,” *Design and Applications*, vol. 5, pp. 64–67, 2001.

[21] D. Mandic and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, 2001.

[22] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, *et al.*, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*, vol. 89, 2015, pp. 89–94.

[23] W. Jin, Z. J. Li, L. S. Wei, and H. Zhen, “The improvements of bp neural network learning algorithm,” in *WCC 2000-ICSP 2000. 2000 5th international conference on signal processing proceedings. 16th world computer congress 2000*, IEEE, vol. 3, 2000, pp. 1647–1649.

[24] A. J. MacDonald, C. A. Greig, and V. Baracos, “The advantages and limitations of cross-sectional body composition analysis,” *Current opinion in supportive and palliative care*, vol. 5, no. 4, pp. 342–349, 2011.

[25] M. Fakhfakh, B. Bouaziz, F. Gargouri, and L. Chaari, “Bayesian optimization using hamiltonian dynamics for sparse artificial neural networks,” in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, IEEE, 2022, pp. 1–4.

[26] O. Stark, E. Atkins, O. Wolff, and J. Douglas, “Longitudinal study of obesity in the national survey of health and development.,” *Br Med J (Clin Res Ed)*, vol. 283, no. 6283, pp. 13–17, 1981.

[27] R. S. Liu, L. Lemieux, G. S. Bell, *et al.*, “A longitudinal study of brain morphometrics using quantitative magnetic resonance imaging and difference image analysis,” *Neuroimage*, vol. 20, no. 1, pp. 22–33, 2003.

[28] S. Overgaauw, A. C. van Duijvenvoorde, B. Gunther Moor, and E. A. Crone, “A longitudinal analysis of neural regions involved in reading the mind in the eyes,” *Social Cognitive and Affective Neuroscience*, vol. 10, no. 5, pp. 619–627, 2015.

[29] F. M. Bengel, P. Ueberfuhr, S. I. Ziegler, S. Nekolla, B. Reichart, and M. Schwaiger, “Serial assessment of sympathetic reinnervation after orthotopic heart transplantation: a longitudinal study using pet and c-11 hydroxyephedrine,” *Circulation*, vol. 99, no. 14, pp. 1866–1871, 1999.

[30] I. Y. Gong, G. Ong, C. Brezden-Masley, *et al.*, “Early diastolic strain rate measurements by cardiac mri in breast cancer patients treated with trastuzumab: a longitudinal study,” *The International Journal of Cardiovascular Imaging*, vol. 35, no. 4, pp. 653–662, 2019.

[31] S. J. Walsh, R. E. Bilsborrow, S. J. McGregor, *et al.*, “Integration of longitudinal surveys, remote sensing time series, and spatial analyses,” in *People and the Environment*, Springer, 2004, pp. 91–130.

[32] T. Werner, A. Bebbington, and G. Gregory, “Assessing impacts of mining: recent contributions from gis and remote sensing,” *The Extractive Industries and Society*, vol. 6, no. 3, pp. 993–1012, 2019.

[33] J. Fox, *People and the environment: Approaches for linking household and community surveys to remote sensing and GIS*. Springer Science & Business Media, 1999, vol. 2.

[34] G. M. Fitzmaurice, N. M. Laird, and J. H. Ware, *Applied longitudinal analysis*. John Wiley & Sons, 2012, vol. 998.

[35] G. McKhann, D. Drachman, M. Folstein, R. Katzman, D. Price, and E. M. Stadlan, “Clinical diagnosis of alzheimers disease: report of the nincds-adrda work group* under the auspices of department of health and human services task force on alzheimers disease,” *Neurology*, vol. 34, no. 7, pp. 939–939, 1984.

[36] A. Varma, J. Snowden, J. Lloyd, P. Talbot, D. Mann, and D. Neary, “Evaluation of the nincds-adrda criteria in the differentiation of alzheimer’s disease and frontotemporal dementia,” *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 66, no. 2, pp. 184–188, 1999.

[37] G. Wollny, “Analysis of changes in temporal series of medical images,” Ph.D. dissertation, Max Planck Institute for Human Cognitive and Brain Sciences Leipzig, 2004.

[38] L. Li, Z. Yang, Z. Dang, *et al.*, “Propagation analysis and prediction of the covid-19,” *Infectious Disease Modelling*, vol. 5, pp. 282–292, 2020.

[39] R. BENTATA, “Segmentation d’images tomographiques par émission de positons,” Ph.D. dissertation, Université d’Oran1-Ahmed Ben Bella, 2012.

[40] S. Webb, “The contribution, history, impact and future of physics in medicine,” *Acta Oncologica*, vol. 48, no. 2, pp. 169–177, 2009.

[41] J. Giron and F. Joffre, *Bases physiques et evolution de limagerie radiologique*. Masson, 1993.

[42] G. Katti, S. A. Ara, and A. Shireen, “Magnetic resonance imaging (mri)–a review,” *International journal of dental clinics*, vol. 3, no. 1, pp. 65–70, 2011.

[43] B. E. Warren, *X-ray Diffraction*. Courier Corporation, 1990.

[44] J. Selb, “Source virtuelle acousto-optique pour l’imagerie des milieux diffusants,” *UNIVERSITE PARIS XI*, 2002.

[45] E. Blondiaux, A. Cochet, E. Durand, S. Kremer, M. Montaudon, C. Parlier-Cuau, *et al.*, *Imagerie medicale: Les fondamentaux: radioanatomie, biophysique, techniques et semeiologie en radiologie et medecine nucleaire*. Elsevier Health Sciences, 2017.

[46] J. J. Wild and J. M. Reid, “Application of echo-ranging techniques to the determination of structure of biological tissues,” *Science*, vol. 115, no. 2983, pp. 226–230, 1952.

[47] C. J. Harvey, J. M. Pilcher, R. J. Eckersley, M. J. Blomley, and D. O. Cosgrove, “Advances in ultrasound,” *Clinical radiology*, vol. 57, no. 3, pp. 157–177, 2002.

[48] G. K. Matsopoulos and S. Marshall, “Use of morphological image processing techniques for the measurement of a fetal head from ultrasound images,” *Pattern Recognition*, vol. 27, no. 10, pp. 1317–1324, 1994.

[49] G. N. Hounsfield, “Computerized transverse axial scanning (tomography): part 1. description of system,” *The British journal of radiology*, vol. 46, no. 552, pp. 1016–1022, 1973.

[50] A. Sarma, M. E. Heilbrun, K. E. Conner, S. M. Stevens, S. C. Woller, and C. G. Elliott, “Radiation and chest ct scan examinations: what do we know?” *Chest*, vol. 142, no. 3, pp. 750–760, 2012.

[51] S. Aubry, “Modelisation tridimensionnelle des vertebres a but didactique en radio-anatomie et radiologie interventionnelle sous guidage tomodensitometrique,” Ph.D. dissertation, 2007.

[52] S. A. Bobman, S. J. Riederer, J. N. Lee, *et al.*, “Cerebral magnetic resonance image synthesis.,” *American Journal of Neuroradiology*, vol. 6, no. 2, pp. 265–269, 1985.

[53] I. M. Vavasour, *Magnetic resonance of human and bovine brain*. Citeseer, 1998.

[54] C. Guinet and J. Grellet, *Introduction a IIRM: de la theorie a la pratique*. Masson, 1992.

[55] M. BELADGHAM, "Construction d'une technique d'aide audiagnostic en imagerie medicale. application à la compression d'images," Ph.D. dissertation, Thèse de doctorat en électronique, Université Abou-Beker Belkaid-Tlemcen, 2012.

[56] A. Sundin, U. Garske, and H. Orlefors, "Nuclear imaging of neuroendocrine tumours," *Best Practice & Research Clinical Endocrinology & Metabolism*, vol. 21, no. 1, pp. 69–85, 2007.

[57] M. Laruelle, A. Abi-Dargham, C. H. Van Dyck, *et al.*, "Single photon emission computerized tomography imaging of amphetamine-induced dopamine release in drug-free schizophrenic subjects.," *Proceedings of the National Academy of Sciences*, vol. 93, no. 17, pp. 9235–9240, 1996.

[58] G. Muehllehner and J. S. Karp, "Positron emission tomography," *Physics in Medicine & Biology*, vol. 51, no. 13, R117, 2006.

[59] G. Dreyfus, "Les réseaux de neurones pour la modelisation des procedes industriels: du ruban adhesif au soudage par point," *Signal processing and Machine learning laboratory (SIGMA lab), Ecole Superieure de Physique et de Chimie Industrielle (ESPCI), Paris*, 2002.

[60] J.-W. Lin, "Artificial neural network related to biological neuron network: a review," *Advanced Studies in Medical Sciences*, vol. 5, no. 1, pp. 55–62, 2017.

[61] F. Rosenblatt, "Perceptron simulation experiments," *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, 1960.

[62] M. Gabrie, A. Manoel, C. Luneau, N. Macris, F. Krzakala, L. Zdeborova, *et al.*, "Entropy and mutual information in models of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[63] B. Bouzy, "Descente de gradient," 2005.

[64] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[65] O. John, "Mise en oeuvre d'un systeme de traduction du couple de langue franccais-fongbe,"

[66] H. Ramchoun, M. J. Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: architecture optimization and training with mixed activation functions," in *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, 2017, pp. 1–6.

[67] J. BORDAS and F. TSCHIRHART, "Ecole superieure de genie informatique," 2009.

[68] Y. LeCun, Y. Bengio, *et al.*, *The handbook of brain theory and neural networks*, 1998.

[69] A. Drewek-Ossowicka, M. Pietrolaj, and J. Ruminski, "A survey of neural networks usage for intrusion detection systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 497–514, 2021.

[70] T. K. Sajja and H. K. Kalluri, "Image classification using regularized convolutional neural network design with dimensionality reduction modules: rcnn–drm," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2021.

[71] K. Ostad-Ali-Askari and M. Shayan, "Subsurface drain spacing in the unsteady conditions by hydrus-3d and artificial neural networks," *Arabian Journal of Geosciences*, vol. 14, no. 18, pp. 1–14, 2021.

[72] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proceedings of the 4th ACM workshop on information hiding and multimedia security*, 2016, pp. 5–10.

[73] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[74] R. Zaheer and H. Shaziya, "A study of the optimization algorithms in deep learning," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, IEEE, 2019, pp. 536–539.

[75] C. Xie and F. Zhang, "A new sequence optimization algorithm based on particle swarm for machine learning," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2021.

[76] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[77] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, PMLR, 2013, pp. 1139–1147.

[78] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems*, vol. 26, 2013.

[79] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.

[80] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.,” *Journal of machine learning research*, vol. 12, no. 7, 2011.

[81] D. Kingma and J. Ba, “Adam: a method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[82] C. Darken and J. Moody, “Note on learning rate schedules for stochastic optimization,” *Advances in neural information processing systems*, vol. 3, 1990.

[83] M. C. Mukkamala and M. Hein, “Variants of rmsprop and adagrad with logarithmic regret bounds,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 2545–2553.

[84] D. F. Shanno, “Conditioning of quasi-newton methods for function minimization,” *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

[85] J. Pajarinen, H. L. Thai, R. Akrour, J. Peters, and G. Neumann, “Compatible natural gradient policy search,” *Machine Learning*, vol. 108, no. 8, pp. 1443–1466, 2019.

[86] J. Nocedal and S. J. Wright, “Quadratic programming,” *Numerical optimization*, pp. 448–492, 2006.

[87] Z. Wei, W. Chen, C.-W. Qiu, and X. Chen, “Conjugate gradient method for phase retrieval based on the wirtinger derivative,” *JOSA A*, vol. 34, no. 5, pp. 708–712, 2017.

[88] L. Shi, T. Lee, L. Zhang, J. K. Nielsen, and M. G. Christensen, “A fast reduced-rank sound zone control algorithm using the conjugate gradient method,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 436–440.

[89] E. Stiefel, “Methods of conjugate gradients for solving linear systems,” *J. Res. Nat. Bur. Standards*, vol. 49, pp. 409–435, 1952.

[90] J. Nocedal and S. J. Wright, “Sequential quadratic programming,” *Numerical optimization*, pp. 529–562, 2006.

[91] R. Bollapragada, R. H. Byrd, and J. Nocedal, “Exact and inexact subsampled newton methods for optimization,” *IMA Journal of Numerical Analysis*, vol. 39, no. 2, pp. 545–578, 2019.

[92] P. T. Harker and J.-S. Pang, “A damped-newton method for the linear complementarity problem,” *Lectures in Applied Mathematics*, vol. 26, pp. 265–284, 1990.

[93] P. Y. Ayala and H. B. Schlegel, “A combined method for determining reaction paths, minima, and transition state geometries,” *The Journal of chemical physics*, vol. 107, no. 2, pp. 375–384, 1997.

[94] J. Nocedal and S. J. Wright, “Quasi-newton methods,” *Numerical optimization*, pp. 135–163, 2006.

[95] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-newton method for large-scale optimization,” *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[96] P. Moritz, R. Nishihara, and M. Jordan, “A linearly-convergent stochastic l-bfgs algorithm,” in *Artificial Intelligence and Statistics*, PMLR, 2016, pp. 249–258.

[97] L. Bottou and O. Bousquet, “The tradeoffs of large scale learning,” *Advances in neural information processing systems*, vol. 20, 2007.

[98] N. N. Schraudolph, J. Yu, and S. Gunter, “A stochastic quasi-newton method for online convex optimization,” in *Artificial intelligence and statistics*, PMLR, 2007, pp. 436–443.

[99] J. Martens *et al.*, “Deep learning via hessian-free optimization.,” in *ICML*, vol. 27, 2010, pp. 735–742.

[100] R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal, “On the use of stochastic hessian information in optimization methods for machine learning,” *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 977–995, 2011.

[101] S.-I. Amari, “Natural gradient works efficiently in learning,” *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[102] J. Martens, “New insights and perspectives on the natural gradient method,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5776–5851, 2020.

[103] W. Sun and Y.-X. Yuan, *Optimization theory and methods: nonlinear programming*. Springer Science & Business Media, 2006, vol. 1.

[104] R. H. Byrd, J. C. Gilbert, and J. Nocedal, “A trust region method based on interior point techniques for nonlinear programming,” *Mathematical programming*, vol. 89, no. 1, pp. 149–185, 2000.

[105] L. Hei, “Practical techniques for nonlinear optimization,” Ph.D. dissertation, Northwestern University, 2007.

[106] A. S. Berahas, R. H. Byrd, and J. Nocedal, “Derivative-free optimization of noisy functions via quasi-newton methods,” *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 965–993, 2019.

[107] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. Society for Industrial and Applied Mathematics (SIAM), 2009.

[108] C. Audet and M. Kokkolaras, *Blackbox and derivative-free optimization: theory, algorithms and applications*, 2016.

[109] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[110] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[111] M. Dorigo and T. Stutzle, "Ant colony optimization: overview and recent advances," *Handbook of metaheuristics*, pp. 311–351, 2019.

[112] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[113] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.

[114] I. Boussaid, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information sciences*, vol. 237, pp. 82–117, 2013.

[115] Y. Shi, "Particle swarm optimization," *IEEE connections*, vol. 2, no. 1, pp. 8–13, 2004.

[116] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: optimizing connections and connectivity," *Parallel computing*, vol. 14, no. 3, pp. 347–361, 1990.

[117] M. Khishe and M. Mosavi, "Improved whale trainer for sonar datasets classification using neural network," *Applied Acoustics*, vol. 154, pp. 176–192, 2019.

[118] H. Jia, K. Sun, W. Zhang, and X. Leng, "An enhanced chimp optimization algorithm for continuous optimization domains," *Complex & Intelligent Systems*, pp. 1–18, 2021.

[119] M. Khishe and H. Mohammadi, "Passive sonar target classification using multi-layer perceptron trained by salp swarm algorithm," *Ocean Engineering*, vol. 181, pp. 98–108, 2019.

[120] M. R. Mosavi, M. Khishe, M. J. Naseri, G. R. Parvizi, and M. Ayat, "Multi-layer perceptron neural network utilizing adaptive best-mass gravitational search algorithm to classify sonar dataset," *Archives of Acoustics*, vol. 44, 2019.

[121] M. Khishe and A. Safari, "Classification of sonar targets using an mlp neural network trained by dragonfly algorithm," *Wireless Personal Communications*, vol. 108, no. 4, pp. 2241–2260, 2019.

[122] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, “The arithmetic optimization algorithm,” *Computer methods in applied mechanics and engineering*, vol. 376, p. 113 609, 2021.

[123] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, “Towards federated learning at scale: system design,” *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[124] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[125] D. Polap, “Fuzzy consensus with federated learning method in medical systems,” *IEEE Access*, vol. 9, 2021.

[126] D. P. Bertsekas, “Nonlinear programming,” *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[127] I. Loshchilov, M. Schoenauer, and M. Sebag, “Adaptive coordinate descent,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 885–892.

[128] H.-S. Chang, E. Learned-Miller, and A. McCallum, “Active bias: training more accurate neural networks by emphasizing high variance samples,” *arXiv preprint arXiv:1704.07433*, 2017.

[129] D. M. Hawkins, “The problem of overfitting,” *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[130] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[131] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[132] J. Schmidhuber, “Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-hook,” Ph.D. dissertation, Technische Universitat Munchen, 1987.

[133] T. Schaul and J. Schmidhuber, “Metalearning,” *Scholarpedia*, vol. 5, no. 6, p. 4650, 2010.

[134] C. Adjiman, I. Androulakis, C. Maranas, and C. Floudas, “A global optimization method, α bb, for process design,” *Computers & Chemical Engineering*, vol. 20, S419–S424, 1996.

[135] C. Adjiman, C. Schweiger, and C. Floudas, “Mixed-integer nonlinear optimization in process synthesis,” in *Handbook of combinatorial optimization*, 1998, pp. 1–76.

[136] L. Xu and D. Schuurmans, “Unsupervised and semi-supervised multi-class support vector machines,” in *AAAI*, vol. 40, 2005, p. 50.

[137] S. Diamond, R. Takapoui, and S. Boyd, “A general system for heuristic minimization of convex functions over non-convex sets,” *Optimization Methods and Software*, vol. 33, no. 1, pp. 165–193, 2018.

[138] D. Park, A. Kyrillidis, S. Bhojanapalli, C. Caramanis, and S. Sanghavi, “Provable non-convex projected gradient descent for a class of constrained matrix optimization problems,” *stat*, vol. 1050, p. 4, 2016.

[139] S. Balakrishnan, M. J. Wainwright, B. Yu, *et al.*, “Statistical guarantees for the em algorithm: from population to sample-based analysis,” *Annals of Statistics*, vol. 45, no. 1, pp. 77–120, 2017.

[140] Z. Wang, Q. Gu, Y. Ning, and H. Liu, “High dimensional expectation-maximization algorithm: statistical optimization and asymptotic normality,” *arXiv preprint arXiv:-1412.8729*, 2014.

[141] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1-212.5701*, 2012.

[142] M. Avriel, *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

[143] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[144] A. Apicella, F. Donnarumma, F. Isgro, and R. Prevete, “A survey on modern trainable activation functions,” *Neural Networks*, vol. 138, pp. 14–32, 2021.

[145] G.-B. Huang, “What are extreme learning machines? filling the gap between frank rosenblatt’s dream and john von neumann’s puzzle,” *Cognitive Computation*, vol. 7, pp. 263–278, 2015.

[146] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini, “Kafnets: kernel-based non-parametric activation functions for neural networks,” *Neural Networks*, vol. 110, pp. 19–32, 2019.

[147] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: comparison of trends in practice and research for deep learning,” *arXiv preprint arXiv:1811.03378*, 2018.

[148] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[149] L. Vecci, F. Piazza, and A. Uncini, “Learning and approximation capabilities of adaptive spline activation function neural networks,” *Neural Networks*, vol. 11, no. 2, pp. 259–270, 1998.

[150] B. Widrow and M. A. Lehr, “30 years of adaptive neural networks: perceptron, madaline, and backpropagation,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[151] G. Cybenko, *Continuous valued networks with two hidden layers are sufficient: technical report*, 1988.

[152] F.-C. Chen, “Back-propagation neural networks for nonlinear self-tuning adaptive control,” *IEEE control systems Magazine*, vol. 10, no. 3, pp. 44–48, 1990.

[153] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[154] D. Pedamonti, “Comparison of non-linear activation functions for deep neural networks on mnist classification task,” *arXiv preprint arXiv:1804.02763*, 2018.

[155] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[156] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Icml*, 2010.

[157] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, “Noisy activation functions,” in *International conference on machine learning*, PMLR, 2016, pp. 3059–3068.

[158] B. Xu, R. Huang, and M. Li, “Revise saturated activation functions,” *arXiv preprint arXiv:1602.05980*, 2016.

[159] A. Montalto, G. Tessitore, and R. Prevete, “A linear approach for sparse coding by a two-layer neural network,” *Neurocomputing*, vol. 149, pp. 1315–1323, 2015.

[160] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. icml*, Citeseer, vol. 30, 2013, p. 3.

[161] K. Konda, R. Memisevic, and D. Krueger, “Zero-bias autoencoders and the benefits of co-adapting features,” *arXiv preprint arXiv:1402.3337*, 2014.

[162] C. Dugas, Y. Bengio, F. Belisle, C. Nadeau, and R. Garcia, “Incorporating second-order functional knowledge for better option pricing,” *Advances in neural information processing systems*, vol. 13, 2000.

[163] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.

[164] E. Alcaide, “E-swish: adjusting activations to different network depths,” *arXiv preprint arXiv:1801.07145*, 2018.

[165] H. H. Chieng, N. Wahid, P. Ong, and S. R. K. Perla, “Flatten-t swish: a thresholded relu-swish-like activation function for deep learning,” *arXiv preprint arXiv:1812.062-47*, 2018.

[166] S. Elfwing, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” *Neural Networks*, vol. 107, pp. 3–11, 2018.

[167] C.-T. Chen and W.-D. Chang, “A feedforward neural network with function shape autotuning,” *Neural networks*, vol. 9, no. 4, pp. 627–641, 1996.

[168] S. Guarnieri, “Multilayer neural networks with adaptive spline-based activation functions,” in *Proceedings of Word Congress on Neural Networks WCNN95, Washington, DC, July*, 1995, pp. 17–21.

[169] F. Piazza, A. Uncini, and M. Zenobi, “Artificial neural networks with adaptive polynomial activation function,” 1992.

[170] F. Piazza, A. Uncini, and M. Zenobi, “Neural networks with digital lut activation functions,” in *Proceedings of 1993 international conference on neural networks (IJCNN-93-Nagoya, Japan)*, IEEE, vol. 2, 1993, pp. 1401–1404.

[171] Z. Hu, “The study of neural network adaptive control systems,” *Control and Decision*, vol. 7, pp. 361–366, 1992.

[172] L. Trottier, P. Giguere, and B. Chaib-Draa, “Parametric exponential linear unit for deep convolutional neural networks,” in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2017, pp. 207–214.

[173] S. Qiu, X. Xu, and B. Cai, “Frelu: flexible rectified linear units for improving convolutional neural networks,” in *2018 24th international conference on pattern recognition (icpr)*, IEEE, 2018, pp. 1223–1228.

[174] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[175] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.

[176] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.

[177] M. Basirat and P. M. Roth, “The quest for the golden activation function,” *arXiv preprint arXiv:1808.00783*, 2018.

[178] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, “Learning activation functions to improve deep neural networks,” *arXiv preprint arXiv:1412.6830*, 2014.

[179] G. Maguolo, L. Nanni, and S. Ghidoni, “Ensemble of convolutional neural networks trained with different activation functions,” *Expert Systems with Applications*, vol. 166, p. 114 048, 2021.

[180] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey, “Tanhsoft—dynamic trainable activation functions for faster learning and better performance,” *IEEE Access*, vol. 9, pp. 120 613–120 623, 2021.

[181] L. R. Sutfeld, F. Brieger, H. Finger, S. Fullhase, and G. Pipa, “Adaptive blending units: trainable activation functions for deep neural networks,” in *Science and Information Conference*, Springer, 2020, pp. 37–50.

[182] S. Qian, H. Liu, C. Liu, S. Wu, and H. San Wong, “Adaptive activation functions in convolutional neural networks,” *Neurocomputing*, vol. 272, pp. 204–212, 2018.

[183] A. Apicella, F. Isgro, and R. Prevete, “A simple and efficient architecture for trainable activation functions,” *Neurocomputing*, vol. 370, pp. 1–15, 2019.

[184] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini, “Kafnets: kernel-based non-parametric activation functions for neural networks,” *Neural Networks*, vol. 110, pp. 19–32, 2019.

[185] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” 2008.

[186] S. Scardapane, S. Van Vaerenbergh, A. Hussain, and A. Uncini, “Complex-valued neural networks with nonparametric activation functions,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 140–150, 2018.

[187] T. Nitta, “Local minima in hierarchical structures of complex-valued neural networks,” *Neural Networks*, vol. 43, pp. 1–7, 2013.

[188] F. Piazza, A. Uncini, and M. Zenobi, “Artificial neural networks with adaptive polynomial activation function,” 1992.

[189] M. Stinchcombe and H. White, “Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights,” in *1990 IJCNN International Joint Conference on Neural Networks*, IEEE, 1990, pp. 7–16.

[190] C.-C. Jou, “Fuzzy activation functions,” in *Proceedings 1991 IEEE International Joint Conference on Neural Networks*, IEEE, 1991, pp. 128–133.

[191] M. Karakose and E. Akin, “Type-2 fuzzy activation function for multilayer feedforward neural networks,” in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, IEEE, vol. 4, 2004, pp. 3762–3767.

[192] A. Nguyen and A. M. Korikov, “Models of neural networks with fuzzy activation functions,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 177, 2017, p. 012 031.

[193] J. M. Mendel, “Type-2 fuzzy sets and systems: an overview,” *IEEE computational intelligence magazine*, vol. 2, no. 1, pp. 20–29, 2007.

[194] J. M. Mendel and R. B. John, “Type-2 fuzzy sets made simple,” *IEEE Transactions on fuzzy systems*, vol. 10, no. 2, pp. 117–127, 2002.

[195] A. K. De, D. Chakraborty, and A. Biswas, “Literature review on type-2 fuzzy set theory,” *Soft Computing*, vol. 26, no. 18, pp. 9049–9068, 2022.

[196] A. Beke and T. Kumbasar, “Interval type-2 fuzzy systems as deep neural network activation functions,” in *11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)*, Atlantis Press, 2019, pp. 267–273.

[197] S. Scardapane, M. Scarpiniti, D. Comminiello, and A. Uncini, “Learning activation functions from data using cubic spline interpolation,” in *Italian Workshop on Neural Nets*, Springer, 2017, pp. 73–83.

[198] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen, “Reltanh: an activation function with vanishing gradient resistance for sae-based dnns and its application to rotating machinery fault diagnosis,” *Neurocomputing*, vol. 363, pp. 88–98, 2019.

[199] L. Chaari and O. Golubnitschaja, “Covid-19 pandemic by the “real-time” monitoring: the tunisian case and lessons for global epidemics in the context 3pm strategies,” *EPMA Journal*, no. 10223, 2020, DOI:10.1007/ s13167-020-00207-0.

[200] C. Graphs, *Worldwide Cases and Deaths - Worldometer*, https://www.worldometers.info/coronavirus/worldwide-graphs/#total-cases20, 2022.

[201] C. Huang, Y. Wang, X. Li, *et al.*, “Clinical features of patients infected with 2019 novel coronavirus in wuhan, china,” *The Lancet*, vol. 395, no. 10223, pp. 497–506, 2020.

[202] P. Gautret, J.-C. Lagier, P. Parola, *et al.*, “Clinical and microbiological effect of a combination of hydroxychloroquine and azithromycin in 80 COVID-19 patients with at least a six-day follow up: an observational study,” *https://www.mediterranee-infection.com/wp-content/uploads/2020/03/COVID-IHU-2-1.pdf*, pp. 1–28, 2020.

[203] C. Sohrabi, Z. Alsafi, N. O’Neill, *et al.*, “World health organization declares global emergency: a review of the 2019 novel coronavirus (covid-19),” *International Journal of Surgery*, 2020.

[204] H. Li, S.-M. Liu, X.-H. Yu, S.-L. Tang, and C.-K. Tang, “Coronavirus disease 2019 (covid-19): current status and future perspective,” *International journal of antimicrobial agents*, p. 105 951, 2020.

[205] L. Chih-Cheng, S. Tzu-Ping, K. Wen-Chien, T. Hung-Jen, and H. Po-Re, "Severe acute respiratory syndrome coronavirus 2 (sars-cov-2) and coronavirus disease-2019 (covid-19): the epidemic and the challenges," *International J. of Antimicrobial Agents*, vol. 55, no. 3, p. 105 924, 2020.

[206] S. J. S, S. Kim, D. H. Shin, and M. S. Kim, "Inhibition of sars-cov 3cl protease by flavonoids," *J. Enzyme Inhib. Med. Chem.*, vol. 35, no. 3, pp. 145–151, 2020.

[207] V. M. Corman, O. Landt, M. Kaiser, *et al.*, "Detection of 2019 novel coronavirus (2019-ncov) by real-time rt-pcr," *Euro Surveill*, vol. 25, no. 3, pp. 1–8, 2020.

[208] S. A. Zhou and A. Brahme, "Development of phase-contrast x-ray imaging techniques and potential medical applications," *Physica Medica*, vol. 24, no. 3, pp. 129–148, 2008.

[209] X. Xiaowei, J. Xiangao, M. Chunlian, *et al.*, "Deep learning system to screen coronavirus disease 2019 pneumonia," *https://arxiv.org/ftp/arxiv/papers/2002/2002.09334.pdf*, 2020.

[210] K. O. Leslie, S. Trahan, and J. Gruden, "Pulmonary pathology of the rheumatic diseases," in *Seminars in respiratory and critical care medicine*, New York: Thieme Medical Publishers, c1994, vol. 28, 2007, pp. 369–378.

[211] H. A. Rothan and S. N. Byrareddy, "The epidemiology and pathogenesis of coronavirus disease (covid-19) outbreak," *Journal of autoimmunity*, p. 102 433, 2020.

[212] B. Chandra, M. Gupta, and M. P. Gupta, "A multivariate time series clustering approach for crime trends prediction," in *2008 IEEE International Conference on Systems, Man and Cybernetics*, 2008, pp. 892–896.

[213] E. Woźniak, W. Kofman, S. Aleksandrowicz, M. Rybicki, and S. Lewiński, "Multitemporal indices derived from time series of sentinel-1 images as a phenological description of plants growing for crop classification," in *2019 10th International Workshop on the Analysis of Multitemporal Remote Sensing Images (MultiTemp)*, 2019, pp. 1–4.

[214] Y. Lin, L. Zhang, and N. Wang, "A new time series change detection method for landsat land use and land cover change," in *10th International Workshop on the Analysis of Multitemporal Remote Sensing Images (MultiTemp)*, 2019, pp. 1–4.

[215] W. Gharbi, L. Chaari, and A. Benazza-Benyahia, "Unsupervised bayesian change detection for remotely sensed images," *Signal, Image and Video Processing*, 2020, accepted.

[216] R. Chuentawat and Y. Kan-ngan, "The comparison of pm2.5 forecasting methods in the form of multivariate and univariate time series based on support vector

machine and genetic algorithm," in *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2018, pp. 572–575.

[217] L. Chaari, T. Vincent, F. Forbes, M. Dojat, and P. Ciuciu, "Fast joint detection-estimation of evoked brain activity in event-related fmri using a variational approach," *IEEE Transactions on Medical Imaging*, vol. 32, no. 5, pp. 821–837, May 2013.

[218] M. Albughdadi, L. Chaari, J. Y. Tourneret, F. Forbes, and P. Ciuciu, "A bayesian non-parametric hidden markov random model for hemodynamic brain parcellation," *Signal processing*, vol. 135, no. 10223, pp. 132–146, 2017.

[219] B. Bouaziz, L. Chaari, H. Batatia, and A. Quintero-Rincon., "Epileptic seizure detection using a convolutional neural network," in *In International Conference on Digital Health Technologies (ICDHT)*, Sfax, Tunisia, 2018.

[220] A. Laruelo, L. Chaari, H. Batatia, *et al.*, "Hybrid sparse regularization for magnetic resonance spectroscopy," in *IEEE International Conference of Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, 2013, pp. 6768–6771.

[221] M. Prummer, J. Hornegger, G. Lauritsch, L. Wigstrom, E. Girard-Hughes, and R. Fahrig, "Cardiac c-arm ct: a unified framework for motion estimation and dynamic CT," *IEEE Transactions on Medical Imaging*, vol. 28, no. 11, pp. 1836–1849, 2009.

[222] T. N. Sainath, M. Abdel-rahman, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 8614–8618.

[223] K. Alex, S. Ilya, and H. G. E, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[224] S. Karen and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[225] T. Fang, "A novel computer-aided lung cancer detection method based on transfer learning from googlenet and median intensity projections," in *IEEE International Conference on Computer and Communication Engineering Technology (CCET)*, 2018, pp. 286–290.

[226] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[227] Z. Wu, J. Hai, L. Zhang, J. Chen, G. Cheng, and B. Yan, “Cascaded fully convolutional densenet for automatic kidney segmentation in ultrasound images,” in *2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2019, pp. 384–388.

[228] P. V. Arun, K. M. Buddhiraju, and A. Porwal, “Analysis of capsulenets towards hyperspectral classification,” in *9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2018, pp. 1–5.

[229] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and M. Chowdhury, “A review of deep learning-based detection methods for covid-19,” *Computers in Biology and Medicine*, p. 105 233, 2022.

[230] S. Ying, Z. Shuangjia, L. Liang, *et al.*, “Deep learning enables accurate diagnosis of novel coronavirus (covid-19) with ct images,” *medRxiv*, 2020.

[231] S. P. Kumar and B. S. Kumari, “Detection of coronavirus disease (covid-19) based on deep features,” *Preprints*, vol. 2020030300, p. 2020, 2020.

[232] A. Asmaa, A. M. M, and G. M. Medhat, “Classification of covid-19 in chest x-ray images using detrac deep convolutional neural network,” *arXiv preprint arXiv:2003.13815*, 2020.

[233] A. Abbas, M. M. Abdelsamea, and M. M. Gaber, “Detrac: transfer learning of class decomposed medical images in convolutional neural networks,” *IEEE Access*, pp. 1–1, 2020.

[234] R. Vilalta, M.-K. Achari, and C. F. Eick, “Class decomposition via clustering: a new framework for low-variance classifiers,” in *Third IEEE International Conference on Data Mining*, IEEE, 2003, pp. 673–676.

[235] M. Russwurm and M. Korner, “Multi-temporal land cover classification with sequential recurrent encoders,” *ISPRS International Journal of Geo-Information*, vol. 7, no. 4, p. 129, 2018.

[236] S. Kanai, Y. Fujiwara, and S. Iwamura, “Preventing gradient explosions in gated recurrent units,” *Advances in neural information processing systems*, vol. 30, 2017.

[237] H. Sepp and S. Jurgen, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[238] S. Kamilya and J. A. Pappachen, “A survey on lstm memristive neural network architectures and applications,” *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[239] “Lstm: a search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[240] K. Dutta and K. K. Sarma, “Multiple feature extraction for rnn-based assamese speech recognition for speech to text conversion application,” in *International Conference on Communications, Devices and Intelligent Systems (CODIS)*, 2012, pp. 600–603.

[241] W. Kahuttanaseth, A. Dressler, and C. Netramai, “Commanding mobile robot movement based on natural language processing with rnn encoder-decoder,” in *5th International Conference on Business and Industrial Research (ICBIR)*, 2018, pp. 161–166.

[242] S. Merity, N. S. Keskar, and R. Socher, “Regularizing and optimizing lstm language models,” *arXiv preprint arXiv:1708.02182*, 2017.

[243] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, “Breaking the softmax bottleneck: a high-rank rnn language model,” *arXiv preprint arXiv:1711.03953*, 2017.

[244] H. Inan, K. Khosravi, and R. Socher, “Tying word vectors and word classifiers: a loss framework for language modeling,” *arXiv preprint arXiv:1611.01462*, 2016.

[245] Q. Wang, H. Fan, G. Sun, W. Ren, and Y. Tang, “Recurrent generative adversarial network for face completion,” *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[246] X. Kai and W. Ying, “Lstm-ma: a lstm method with multi-modality and adjacency constraint for brain image segmentation,” in *IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 240–244.

[247] M. Vittorio, K. Aleem, and C. Marcello, “Improvement in land cover and crop classification based on temporal features learning from sentinel-2 data using recurrent-convolutional neural network (r-cnn),” *Applied Sciences*, vol. 10, no. 1, p. 238, 2020.

[248] P. Hu, J. Tong, J. Wang, Y. Yang, and L. d. Oliveira Turci, “A hybrid model based on cnn and bi-lstm for urban water demand prediction,” in *IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 1088–1094.

[249] Y. Heryadi and H. L. H. S. Warnars, “Learning temporal representation of transaction amount for fraudulent transaction recognition using cnn, stacked lstm, and cnn-lstm,” in *IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, 2017, pp. 84–89.

[250] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[251] J. P. Cohen, M. Hashir, R. Brooks, and H. Bertrand, *On the limits of cross-domain generalization in automated x-ray prediction*, 2020. arXiv: 2002.02497 [eess.IV].

[252] G. Maguolo and L. Nanni, *A critic evaluation of methods for covid-19 automatic detection from x-ray images*, 2020. arXiv: 2004.12823 [eess.IV].

[253] J. Gareth, W. Daniela, H. Trevor, and T. Robert, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[254] G. Ian, B. Yoshua, and C. Aaron, *Deep learning*. MIT press, 2016.

[255] Y. Sasak, "The truth of the f-measure," School of Computer Science, University of Manchester, Tech. Rep., Oct. 2007.

[256] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[257] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," 1970.

[258] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, 2017.

[259] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2857–2865.

[260] S. Hanson and L. Pratt, *A comparison of different biases for minimal network construction with back-propagation, advances in neural information processing, d. touretzsky*, 1989.

[261] B. Hassibi and D. Stork, "Second order derivatives for network pruning: optimal brain surgeon," *Advances in neural information processing systems*, vol. 5, 1992.

[262] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989.

[263] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *IEEE international conference on neural networks*, IEEE, 1993, pp. 293–299.

[264] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.

[265] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, "Exploring sparsity in recurrent neural networks," *arXiv preprint arXiv:1704.05119*, 2017.

[266] S. Anwar, K. Hwang, and W. Sung, “Structured pruning of deep convolutional neural networks,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 1–18, 2017.

[267] M. Zhao, J. Peng, S. Yu, L. Liu, and N. Wu, “Exploring structural sparsity in cnn via selective penalty,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1658–1666, 2021.

[268] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2017.

[269] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.

[270] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regularization of neural networks using dropconnect,” in *International conference on machine learning*, PMLR, 2013, pp. 1058–1066.

[271] A. Poernomo and D.-K. Kang, “Biased dropout and crossmap dropout: learning towards effective dropout regularization in convolutional neural network,” *Neural networks*, vol. 104, pp. 60–67, 2018.

[272] J. M. Alvarez and M. Salzmann, “Learning the number of neurons in deep networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2270–2278.

[273] J. Yoon and S. J. Hwang, “Combined group and exclusive sparsity for deep neural networks,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 3958–3966.

[274] Y. Fan, J. Yu, Y. Mei, *et al.*, “Neural sparse representation for image restoration,” *ArXiv*, vol. abs/2006.04357, 2020.

[275] L. Han, H. Lin, and L. Jun, “Remote sensing image classification based on convolutional neural networks with two-fold sparse regularization,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 992–995.

[276] M. Quiroz, M. Villani, and R. Kohn, “Slable mcmc for large data problems using data subsampling and the difference estimator,” *Riksbank Research Paper Series*, no. 160, pp. 1–32, 2016.

[277] M. Fakhfakh, L. Chaari, and N. Fakhfakh, “Bayesian curved lane estimation for autonomous driving,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–11, 2020.

[278] C.-H. Lee, X. Xu, and D. Y. Eun, “Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling,” *ACM SIGMETRICS Performance evaluation review*, vol. 40, no. 1, pp. 319–330, 2012.

[279] G. Roberts and R. Tweedie, “Exponential convergence of langevin distributions and their discrete approximations,” *Bernoulli*, vol. 2, no. 4, pp. 341–363, 1996.

[280] K. Hanson, “Markov chain monte carlo posterior sampling with the hamiltonian method,” in *Medical Imaging 2001: Image Processing*, International Society for Optics and Photonics, vol. 4322, 2001, pp. 456–467.

[281] L. Chaari, J. Tourneret, C. Chaux, and H. Batatia, “A hamiltonian monte carlo method for non-smooth energy sampling,” *IEEE Trans. on Signal Process.*, vol. 64, no. 21, pp. 5585–5594, 2016.

[282] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, “Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science,” *Nature Communications*, vol. 9, pp. 1–12, 2018.

[283] B. K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM journal on computing*, vol. 24, no. 2, pp. 227–234, 1995.

[284] T.-M. Li, J. Lehtinen, R. Ramamoorthi, W. Jakob, and F. Durand, “Anisotropic gaussian mutations for metropolis light transport through hessian-hamiltonian dynamics,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–13, 2015.

[285] B. J. Alder and T. E. Wainwright, “Studies in molecular dynamics. i. general method,” *The Journal of Chemical Physics*, vol. 31, no. 2, pp. 459–466, 1959.

[286] R. M. Neal *et al.*, “Mcmc using hamiltonian dynamics,” *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[287] Z. Wang, S. Mohamed, and N. Freitas, “Adaptive hamiltonian and riemann manifold monte carlo,” in *International conference on machine learning*, PMLR, 2013, pp. 1462–1470.

[288] C. Chaux, P. Combettes, J. Pesquet, and V. Wajs, “A variational formulation for frame-based inverse problems,” *Inverse Problems*, vol. 23, no. 4, p. 1495, 2007.

[289] P. Angelov and E. Almeida Soares, “Sars-cov-2 ct-scan dataset: a large dataset of real patients ct scans for sars-cov-2 identification,” *medRxiv*, 2020.

[290] X. Yang, X. He, J. Zhao, Y. Zhang, S. Zhang, and P. Xie, “Covid-ct-dataset: a ct image dataset about covid-19,” *arXiv preprint arxiv:2003.13865*, vol. 3, 2020.

[291] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.

[292] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, “Do cifar-10 classifiers generalize to cifar-10?” *arXiv preprint arXiv:1806.00451*, 2018.

[293] S. Chib and E. Greenberg, “Understanding the metropolis-hastings algorithm,” *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[294] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[295] U. Muhammad, W. Wang, S. P. Chattha, and S. Ali, “Pre-trained vggnet architecture for remote-sensing image scene classification,” in *24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1622–1627.

[296] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[297] Z. Xu, H. Zhang, Y. Wang, X. Chang, and Y. Liang, “L 1/2 regularization,” *Science China Information Sciences*, vol. 53, no. 6, pp. 1159–1169, 2010.

[298] H. M. Pandey, A. Chaudhary, and D. Mehrotra, “A comparative review of approaches to prevent premature convergence in ga,” *Applied Soft Computing*, vol. 24, pp. 1047–1077, 2014.

[299] H. Sepp and S. Jürgen, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[300] S. Kamilya and J. A. Pappachen, “A survey on lstm memristive neural network architectures and applications,” *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[301] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “Lstm: a search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[302] D. S. Marcus, A. F. Fotenos, J. G. Csernansky, J. C. Morris, and R. L. Buckner, “Open access series of imaging studies: longitudinal mri data in nondemented and demented older adults,” *Journal of cognitive neuroscience*, vol. 22, no. 12, pp. 2677–2684, 2010.

[303] A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz, “Sgd learns over-parameterized networks that provably generalize on linearly separable data,” *arXiv preprint arXiv:1710.10174*, 2017.

[304] H. N. Mhaskar and T. Poggio, “Deep vs. shallow networks: an approximation theory perspective,” *Analysis and Applications*, vol. 14, no. 06, pp. 829–848, 2016.

[305] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[306] G. Maguolo, L. Nanni, and S. Ghidoni, “Ensemble of convolutional neural networks trained with different activation functions,” *Expert Systems with Applications*, vol. 166, p. 114 048, 2021.

[307] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” *Advances in neural information processing systems*, vol. 30, 2017.

[308] F. Belisle, Y. Bengio, C. Dugas, R. Garcia, and C. Nadeau, “Incorporating second-order functional knowledge for better option pricing,” 2002.

Liste des Travaux

Journaux internationaux

[1] M. Fakhfakh, B. Bouaziz, F. Gargouri, and L. Chaari, “Prognet: covid-19 prognosis using recurrent and convolutional neural networks,” *The Open Medical Imaging Journal*, vol. 12, no. 1, 2020.

[2] M. Fakhfakh, L. Chaâri, B. Bouaziz, and F. Gargouri, “Non-smooth bayesian learning for artificial neural networks,” *Journal of Ambient Intelligence and Humanized Computing*, 2022.

Conférences internationales à comité de lecture

[3] M. Fakhfakh, B. Bouaziz, H. Batatia, and L. Chaari, “Bayesian optimization for sparse artificial neural networks: application to change detection in remote sensing,” in *Proceedings of International Conference on Information Technology and Applications*, Springer, 2022, pp. 39–49.

[4] M. Fakhfakh, B. Bouaziz, F. Gargouri, and L. Chaari, “Bayesian optimization using hamiltonian dynamics for sparse artificial neural networks,” in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, IEEE, 2022, pp. 1–4.

[5] M. Fakhfakh, B. Bouaziz, L. Chaari, and F. Gargouri, “Efficient bayesian learning of sparse deep artificial neural networks,” in *International Symposium on Intelligent Data Analysis*, Springer, 2022, pp. 78–88.

[6] M. Fakhfakh, B. Bouaziz, F. Gargouri, and L. Chaari, “Bayesian optimization for artificial neural networks: application to covid-19 image classification,” in *International Conference on Digital health Technologies (ICDHT)*, 2022, pp. 1–4.

[7] M. Fakhfakh and L. Chaari, “Fully automatic bayesian method for trainable activation function and deep neural networks,” in *International Conference on Image Processing - ICIP*, 2023 (Soumis).

[8] M. Fakhfakh and L. Chaari, “An automated bayesian technique for optimizing activation functions,” in *European Signal Processing Conference - EUSIPCO*, 2023 (Soumis).

Résumé :
L'imagerie médicale ne cesse de profiter des progrès technologiques et scientifiques. Elle permet d'explorer le corps humain sans examens intrusifs et d'opérer avec grande précision. La mise en avant des innovations technologies durant ces dernières années a favorisé l'émergence de nouvelles techniques pour l'aide au diagnostic. Pour des fins de précisions, le diagnostic peut être réalisé aujourd'hui sur des séries longitudinales d'images. Durant ces années de thèse, trois contributions ont été proposées : Nous avons présenté une méthode de pronostic Covid-19 basée sur des architectures d'apprentissage en profondeur. La méthode proposée est basée sur la combinaison d'un réseau de neurones convolutifs et récurrents pour classifier des images radiographiques thoraciques multi-temporelles et prédire l'évolution de la pathologie pulmonaire observée. L'un des principaux défis dans les méthodes d'apprentissage est l'optimisation des poids du réseau. Dans ce contexte, nous avons développé une nouvelle méthode d'optimisation bayésienne permettant d'ajuster les poids des réseaux de neurones artificiels parcimonieux. La méthode proposée repose sur la dynamique hamiltonienne avec des régularisations non lisses. Par la suite, nous étendons dans la troisième contribution le schéma d'optimisation en proposant une fonction d'activation entraînable à l'aide des Chaîne de Markov Monte Carlo.

Keywords: Optimization, apprentissage profond, MCMC, ns-HMC

Abstract:
Medical imaging continues to benefit from technological and scientific progress. It allows to explore the human body without intrusive examinations and to operate with high precision. The highlighting of technological innovations in recent years has favored the emergence of new techniques for diagnostic assistance. To be more precise, a diagnosis can be made today on longitudinal series of images. During these years of the thesis, three contributions have been proposed: We have proposed a Covid-19 prognosis method based on deep learning architectures. The proposed method is based on the combination of a convolutional and recurrent neural network to classify multi-temporal chest X-ray images and predict the evolution of the observed lung pathology. One of the main challenges in learning methods is the optimization of network weights. In this context, we have developed a new Bayesian optimization method to adjust the weights of sparse artificial neural networks. The proposed method is based on Hamiltonian dynamics with non-smooth regularizations. Then, in the third contribution, we extend the optimization scheme by proposing a trainable activation function using Markov Monte Carlo chains.

Keywords: Optimization, Deep Learning, MCMC, ns-HMC.

Univ. de Toulouse, France - Univ. de Sfax, Tunisie