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

N° Réf :……

Centre Universitaire
Abd Elhafid Boussouf Mila

Institut des Sciences et Technologie **Département de Mathématiques et Informatique**

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Une approche d'optimisation pour une meilleure efficacité d'un modèle d'estimation de temps restant utile du moteur à double flux à base de deep learning

Préparé par : **Hassani Nihal**
Zabat Asma

Soutenue devant le jury

Encadré par	**Abderrezak Samira**	**Grade MAA**
Président	**Guemri Oualid**	**Grade MAB**
Examinateur	**Khalfi Souheila**	**Grade MAA**

Année Universitaire : 2021/2022

الشكر والعرفان

قال تعالى في محكم تنزيلهْ وَسَيَجْزِي اللَه الشَاْكِرِينَ وكذلك مصدقا لقوله آولئِنْ شَكرتُم لأزيدنّكمْآ نحمد الله عز وجل حمدا يليق بعضمة شأنه ونشكره أن وفقنا على إنجاز هذا البحث، وإتمامه بعونه. ونصلي ونسلم على من بلغ الرسالة وأدى الأمانة ونصح الأمة نبي الله محمد ■ صلى الله عليه وسلم ■

كلمة شكر وتقدير لصاحب الفضل في تفوفينا ونجاحنا الاستاذة آ سميرة عبد الرزاقْ فقد كانت خير رفيق لنا في هذا المشوار التعليمي والتي ساعدتنا بشكل كبير في تحديد موضوع ملنا بعناية ومكنتنا من خلال نصائحها وأفكرها المبتكرة في تحسين جودة العمل. مهما قدمنا من كلمات شكر لا نستطيع أن نوافيك حقك. خالص شكرنا لأعضاء لجنة التحكيم قمري وليد رئيسا وخالفي سهيلة مناقشا على مشاركتهم وموافقتهم على تقييم عملنا

إلى أهلنا الأعزاء وخاصة والدينا وإخواننا وأخواتنا وعائلاتنا الذين لطالما دعمت على طول تشكيلتنا. لجميع أصدقائنا ، على دعمهم المعنوي ومساعدتهم الثمينة شكرا لكم جميعا

اهداء

الى من قال فيهما الرحمان : **آوَاخْفِضْ لَهُمَا جَنَاحَ الذُّلِّ مِنَ الرَّحْمَةِ وَقُلْ رَبِّ ارْحَمْهُمَا كَمَا رَبَّيَانِي صَغِيرًا آ** اهدي هذا العمل إلى أحلي وأجمل وأرق لفظ دعا الله إلى طاعته ■ إلى سمتي في الحياة وأملي ومرجعي في الدنيا نبع الحنان ■ **أمي الغالية** آ. إلى من كللت أنامله ليقدم لي لحظة سعادة ■ إلى صاحب القلب الكبير■ إلى نور دربي وضياء حياتي حبيب قلبي **ءبي الغاليْ** اسأل الله أن يمده بالصحة والعافية. إلى البعيد عن العين والقريب من القلب، إلى سندي ونور عيني **ءخي الغالي عمارْ** حفضك الله ورعاك. إلى من قاسمت معهم الأيام إلى إخوتي آ هيثم ، ناريمان، لينة، سلسبيل، رقية، نجود، أمينة، شيماء، جمانة، زياد، أحمد، أرامْ. الكتابة لا تكفي لأصف كم أحبكم، والعمر قصير لأكتب حبكم، أراكم بسمتي وأرى جمال الأيام أنتم. إلى كل عائلتي الكريمة **آجداتي، عمي مصطفى وعمي عماد وزوجاتهم، وعمتي، خالتيْ** دمتم عونا لي ودام حبكم يسري في قلبي حفضكم الله وأطال عمركم. إلى الأرواح الطيبة التي غادرتنا **آجدي، عمي عبد الحميد، خالتيْ** أسكناهم الله فسيح الجنان. إلى نجم بيتنا ومن يمنحه بريق البراءة والمدح **ءلى ملاكي الصغير أيوبْ** حفظه الله. إلى من تشاركت معها هذا العمل **ءسماء زعباط آ** حفضها الله ورعاها فاللهم رزقها فرحة تدمع لها العين. صديقات الطفولة والدراسة زهرات الصبا **آ أسماء، رميسة، مها، كنزة، رغدة، سميرة، ملاك، ريان، أميمة، ماجدة، ناريمانْ** في أفلاك صداقتكم تدور محبتي، وعلى عتبات نبلكم يقف وفائي. إلى كل من عرفته بقلب سليم. إلى كل من حملهم قلبي ونسيهم قلمي. إلى من أمد لي يد العون والمساعدة من أساتذة. وطلاب والأسرة الجامعية ككل.

اهداء

يشرفني أن أهدي هذا العمل المتواضع الذي تم بعون الله تعالى لقد كرست هذا العمل المهم في حياتي لوالداي ، اللذين لا أستطيع أن أشكرهما بما فيه الكفاية والدتي عقيلة التي قادت بمهارة خطواتي الأولى في هذا العالم. هذا العمل هو ثمرة تضحياتك التي قدمتها من أجل تعليمي وتدريبي.

والدي الغالي أحمد ، أنت تمثل لي رمزًا للخير بامتياز ، ومصدر رقة.

إلى إخوتي الأعزاء وأخواتي الكريمات رياض، بوعلام، رشيد، عماد ، عزالدين، ياسمينة ،أمال، عائشة وأولاد أخي وأختي جواد، جاسر، فؤاد، نسيم ، كنزة ،اسلام إلى كل عائلتي وأصدقائي من قريب أوبعيد إلى شريكتي العزيزة نهال أخيرًا إلى جميع صديقاتي مها، روميسة، ملاك، فاطمة ،رغدة، زينب، مريم ، ايمان وأخيراً إلى جميع زملائي من عام ٢٠٢١. لكم جميعا ، شكرا لكم

الملخص

يعتبر تقدير العمر الإنتاجي المتبقي إجراءً تنبؤياً فعالاً يتنبأ بصحة الماكينة بناءً على نمذجة التدهور ومراقبة الحالة. يعد تطبيق توقع الحياة المفيدة المتبقية مهمًا للغاية من حيث تحسين الطاقة وفعالية التكلفة وتخفيف المخاطر. خوارزميات التنبؤ غوى الحالية هي في الأساس أطر عمل التعلم العميق. في هذه الأطروحة ، قمنا بتنفيذ نموذج بصةپ وإضافة طبقة أخرى من التحسين باستخدام الخوارزميات الجينية ، والتي تهدف إلى تحسين تناسق التنبؤات. المعلمات الفائقة ـ تم تحسين معدل التعلم وحجم الدُفعة وعدد الوحدات وحجم النافذة وعدد الحقبة بما يتجاوز القدرة اليدوية. تم اختبار هذا النموذج على مجموعة بيانات اصا ةربفن چت ينگن. يمكن للبنية المحسّنة أن تتنبأ بالمعلمات المفرطة المعينة بشكل مستقل وتوفر نتائج متفوقة. يوضح اختبار وتقييم النموذج المذكور فوائد استخدام هذه الخوارزميات ومدى فعاليتها في حل مثل هذه المشكلات.

الكلمات الرئيسية

تعلم عميق؛ الخوارزمية الجينية ذاكرة طويلة المدى الصيانة القائمة على الحالة والتنبؤات والإدارة الصحية ؛ متبقي العمر الإنتاجي؛ تهيئة؛ الاستدلال الفوقي ، مجموعة بيانات المحرك المروحي.

Abstract

Remaining Useful Life (RUL) estimation is an effective prognostic measure that predicts machine health based on degradation modeling and condition monitoring. The application of RUL prediction is very important in terms of energy optimization, cost effectiveness and risk mitigation. Existing RUL prediction algorithms are mainly deep learning frameworks. In this thesis, we implement the LSTM model and add another layer of optimization using genetic algorithm, wher objective is to improve the consistency of predictions. Hyper-parameters - learning rate, batch size, unit count, window size and epoch counts are optimized beyond manual capability. This model is tested on the NASA Turbofan Jet Engine dataset. The optimized architecture can predict the given hyper-parameters autonomously and provide superior results. Testing and evaluation of the mentioned model demonstrates the benefits of using such algorithm and their effectiveness in solving such problems.

Keywords

deep learning ; genetic algorithm ; long short-term memory ; Condition Based maintenance, prognostics and health management ; remaining useful life ; Optimisation ; Meta-heuristics, turbofan engine dataset (CMAPSS).

Résumé

L'estimation de la durée de vie utile restante (RUL) est une mesure pronostique efficace qui prévoit l'état de santé de la machine en fonction de la modélisation de la dégradation et de la surveillance de l'état. L'application de la prédiction de la durée de vie utile restante (RUL) est très importante en termes d'optimisation énergétique, de rentabilité et d'atténuation des risques. Les algorithmes de prédiction RUL existants constituent principalement des cadres d'apprentissage en profondeur. Dans ce mémoire, nous implémentons le modèle LSTM et ajoutons une autre couche d'optimisation utilisant des algorithmes génétiques, dont l'objectif est améliorer la cohérence des prévisions. Les hyper-paramètres - taux d'apprentissage, taille de lot, nombre d'unités, taille de fenêtre et nombres d'époques sont optimisé au-delà de la capacité manuelle. Ce modèle est testé sur le jeu de données de la NASA Turbofan Jet Engine. L'architecture optimisée peut prédire les hyper-paramètres donnés de manière autonome et fournir des résultats supérieurs. Les tests et l'évaluation du modèle mentionné démontrent les avantages de l'utilisation de tels algorithmes et leur efficacité dans la résolution de tels problèmes.

Mots clés
La Maintenance conditionnelle (CBM) ; PHM ; La durée de vie restante utile (RUL) ; LSTM (Long-Short Term Memory) ; Optimisation, Méta-heuristiques, L'algorithme Génétique ; l'ensemble de données C-MAPSS.

Table des matières

Table des figures

Liste des tableaux

Glossaire **ANN :** Artificial Neural Network.
ARIMA : Autoregressive Integrated Moving Average.
ARMA : Autoregressive moving average.
ARMAX : Autoregressive moving average model with exogenous inputs model.
Bidirectional LSTM : bidirectional Long Short-Term Memory.
CBM : condition based maintenance.
CNN_Bidirectional : Convolutional Neural Network et Bidirectional lstm.
CNN_LSTM : Convolutional Neural Network et Long Short-Term Memory.
CNN : Convolutional Neural Network.
C-MAPSS : Commercial Modular Aero-Propulsion System Simulation.
DEAP : Distributed Evolutionary Algorithm Python. **DNN :** Deep Neural Networks.
Encoder-Decoder LSTM : Encoder-Decoder Long Short-Term Memory.
FD : Fragment Data.
FD X60 000 : la norme AFNOR.
FFNN : Feed-Forward Neural Network.
GRU : Gated Recurrent Unit.
HMM : Hidden Markov Model.
HSMM : hidden semi-Markov model.
IA : Artificial Intelligence.
IDE : Integrated Development Environment.
JANET : Just Another NETwork.
LSTM : Long Short-Term Memory.
LSTM générative : Generative Long Short-Term Memory.
MAE : mean absolute error.
MLP : Multi Layer Perceptron.
MSE : mean square error.
NASA : National Aeronautics and Space Administration.
Open CV : Open Computer Vision.
PHM : prognostics and health management .
R^2 **:** ¨ regression score.
RàPC : raisonnement à partir de cas.
RBD : réseau bayésien dynamique .
RELU : Rectified Linear Unit.
RNA : Réseaux de Neurones Artificiels.
RMSE : Root mean square error.

RNN : Recurrent Neural Network .
RMSProp : Root Mean Square Propagation.
RUL : remaining useful life.
Seq2Seq : Sequence To Sequence
Stacked LSTM : Stacked Long Short-Term Memory.
Vanilla LSTM : Vanilla Long Short-Term Memory.
WS : Windows Size.

Introduction Générale

La maintenance est une étape cruciale dans tout domaine de l'industrie. Les stratégies traditionnelles de la maintenance telles que la maintenance corrective en cas de panne et la maintenance préventive systématique programmée n'ont pas montré leurs fiabilité et efficacité dans le domaine. Les technologies intelligentes de gestion des pronostics et de la santé (PHM), également appelées maintenance conditionnelle (CBM), montrent des capacités prometteuses dans les industries. elle contribue à la maximisation de la disponibilité opérationnelle, la réduction des coûts de maintenance et l'amélioration de la fiabilité et de la sécurité du système en surveillant les conditions de l'installation.

L'objectif de la maintenance conditionnelle est l'estimation de la durée de vie utile restante (en anglais : Remaining Useful Life, RUL) de la machine pour l'évaluation de la dégradation de ses performances.
La durée de vie utile restante (RUL) peut être estimée en fonction des données de trajectoire de l'historique, ce qui est très important pour améliorer les calendriers de maintenance afin d'éviter les pannes catastrophiques d'ingénierie et d'économiser les coûts résultants. L'estimation RUL a un rôle important dans différents domaines, y compris les industries aéronautiques, les équipements médicaux et les centrales électriques, ce qui a inspiré les chercheurs à développer une variété d'approches de prédiction RUL.

En général, les approches existantes pour l'estimation de RUL peuvent être regroupées en trois catégories principales, les approches basées sur les modèles, les approches basées sur les données et les approches hybrides.
Les approches basées sur les modèles ont tendance à être plus précises si la dégradation du système complexe est modélisée avec précision , elles nécessitent des connaissances préalables approfondies sur les systèmes physiques qui ne sont généralement pas disponibles dans la pratique .
Les approches basées sur les données sont capables de modéliser les caractéristiques de dégradation basées sur les données historiques des capteurs. Les corrélations et causalités sous-jacentes dans les données de capteur collectées peuvent être révélées, et les informations du système de correspondances telles que RUL peuvent être déduites. Les approches basées sur les données nécessitent généralement des données historiques suffisantes pour la formation des modèles. Ces dernières années, de nombreux algorithmes basés sur

les données ont été proposés, de bons résultats pronostiques ont été obtenus, notamment : les modèles d'apprentissage comme les réseaux de neurones, la machine à vecteurs de support (SVM), les modèles de Markov cachés, etc.

Récemment, un outil puissant émergée appelée Deep learning où des abstractions de haut niveau de données peuvent être bien modélisées à l'aide de structures profondes complexes, conduisant à une extraction de caractéristiques plus efficace par rapport aux réseaux peu profonds. Les méthodes d'apprentissage en profondeur ont suscité un grand intérêt et obtenu des résultats significatifs dans de nombreux domaines, notamment la reconnaissance d'images et la reconnaissance vocale. Étant donné que les données brutes obtenues à partir de la surveillance de la santé des machines partagent une dimensionnalité élevée similaire à celles des recherches sur le traitement d'images, l'architecture d'apprentissage en profondeur a un grand potentiel dans la surveillance de la santé du pronostic (PHM) et l'estimation RUL.

Le réseau neuronal récurrent, une classe d'architectures de deep learning, est un modèle plus intuitif pour les données de séries chronologiques, mais il convient à la prédiction de la valeur future des séries chronologiques. Nous traitons le problème d'estimation RUL comme une régression de séries chronologiques multivariées. GRU et LSTM des types de RNN utilisant la mémoire à long court terme. Ils conviennet à la classification, au traitement et à la prédiction de séries temporelles avec des délais inconnus de temps et de taille entre des événements importants. Ces modèles utilisent des hyperparamètres, qui doivent être ajuster pour conduire à de meilleurs résultats. Pour cela on utilise les algorithmes génétiques considérés comme l'une des techniques importantes dans la recherche du choix optimal parmi un ensemble de solutions disponibles pour une conception particulière, et adoptent le principe de sélection de Darwin, où ce traitement génétique transmet les avantages optimaux à travers des processus de sélection successifs et renforce ces traits. Ces derniers ont la plus grande capacité à entrer dans le processus Reproduction, la production d'une progéniture optimale et en répétant le cycle génétique, la qualité de la progéniture s'améliore progressivement.

Notre objectif s'inscrit dans le cadre de l'estimation de RUL en utilisant le deep learning et en cherchant la meilleure combinaison d'hyperparamètres à l'aide d'une méthode d'optimisation aboutissant à une meilleure performance du modèle. Pour atteindre cet objectif, nous allons utiliser LSTM et GRU comme modèle de deep learning et l'algorithme génétique comme méthode

d'optimisation des hyperparamètres. Nous faisons ensuite une comparaison des résultats de l'algorithme génétique avec ceux appliqués avec Grid Search. Enfin, une comparaison entre ces modèles et ceux de la littérature sera accomplie.
Le modèle proposé sera entraîner, tester et valider sur l'ensemble de données CMAPSS de la NASA.

Organisation de mémoire

Ce mémoire est structuré de la manière suivante :
Chapitre 1 : La durée de vie utile restante :
Dans ce chapitre, nous aborderons dans une première section la maintenance et ses différents types et dans une seconde section la prédiction de la durée de vie résiduelle (RUL) ainsi que la classification des approches prédictives.
Chapitre 2 : L'apprentissage profond (DEEP LEARNING)
Dans ce chapitre, nous parlerons des différents types d'apprentissage automatique, passerons par les réseaux de neurones. Ensuite nous nous intéresserons au deep learning et ses différents types. Nous définisserons plus en détail le LSTM qu'on va utiliser dans notre approche.
Chapitre 3 : L'optimisation
Dans ce chapitre, nous parlerons de l'optimisation, de l'optimisation combinatoire et de méthodes d'optimisation (Exactes et approchées). Nous allons abordé en détail l'approche des algorithmes génétiques.
Chapitre 4 : Contribution
Nous allons décrire dans ce chapitre l'architecture du modèle LSTM utilisé, les étapes suivi commun à tout problème de prédiction ainsi que le dataset utilisé pour évaluer le modèle. Ensuite et en se basant sur les travaux connexes décrits au chapitre 3, nous tenterons dans ce chapitre d'améliorer les résultats obtenues par le modèle LSTM en choisissant d'autre combinaison d'hyperparamètres. Nous présenterons en détail ces derniers et les différentes étapes d'application de l'algorithme génétique. Nous présenterons à la fin les résultats obtenues, et nous allons les comparer avec ceux des travaux connexes de la littérature.
Chapitre 5 : Implémentation
Dans ce chapitre nous allons présenter les outils utilisés et les pseudo codes implémentés pour la réalisation du modèle proposé.

A la fin nous allons terminer par une conclusion générale qui résume notre travail et quelques perspectives possibles du travail.

Chapitre 1

La durée de vie utile restante

1.1 Introduction

La maintenance est définie comme un ensemble de toutes les actions techniques, administratives et organisationnelles durant le cycle de vie d'un bien dans le but de le maintenir ou de le restaurer [1]. Les tâches de maintenance varient en fonction de l'environnement de travail, qui comprend, par exemple, l'inspection visuelle, les tests, la mesure, le changement des consommables (lubrification, filtres à huile), le réglage, la réparation, la maintenance, le remplacement des pièces, la maintenance, l'échantillonnage de l'huile, la lubrification, Et vissage, nettoyage, recherche de panne, diagnostic de panne, etc. La maintenance est divisée en deux parties, pré-panne et post-panne. La maintenance de la machine avant la panne lance une prévision de la durée de vie utile restante (RUL) basée sur le changement de certaines conditions qui affectent le fonctionnement de la machine. La prédiction de la durée de vie utile résiduelle (RUL), également connue sous le nom de pronostic RUL, est un sujet étudié dans la ressource PHM sur l'automatisation et la mécanique. Il peut être défini comme une estimation du temps restant entre le moment actuel et le moment de la panne (ou la condition maximale acceptable) ; Cette espérance est généralement donnée en termes de probabilité. La durée de vie utile résiduelle (RUL) d'un actif ou d'un système est définie comme le temps restant entre le moment actuel et la fin de sa vie utile. Dans ce chapitre, nous aborderons dans une première section la maintenance de ses différents types et dans une seconde section la prédiction de la durée de vie et de la durée de vie résiduelle (RUL) et la classification des approches prédictives.

1.2 Partie 01 : La maintenance

1.2.1 Définition

Selon la (norme FD X60-000) La maintenance est un ensemble de toutes les actions techniques, administratives et de management durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise[2]. La maintenance est définie comme la combinaison de tous les aspects techniques, administratifs et actions au cours du cycle de vie d'un élément destinées à le conserver ou à le restaurer dans un état dans lequel il peut exécuter la fonction requise. Les tâches de maintenance varient selon les environnements de travail, qui comprennent par exemple inspection visuelle, test, mesure, changement de consommables (graissage, lubrification, filtres à huile), réglage, réparation, entretien, remplacement de pièces, entretien, échantillonnage d'huile, 1lubrification, resserrage des boulons, nettoyage, détection de panne, diagnostic de panne, etc [3].

1.2.2 Les types de la maintenance

Il existe deux types de maintenance (figure :1.1) la maintenance corrective et la maintenance préventive. La différence entre elles réside au moment d'intervention vis-à-vis de la panne. Le premier type de maintenance est appliqué après l'occurrence de la panne, alors que le deuxième type s'applique avant cette dernière[4].

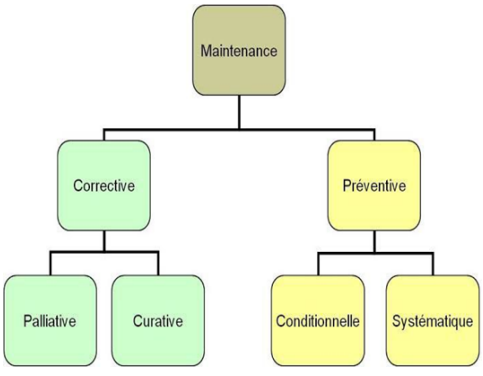

FIGURE 1.1 – Les différents types de maintenance

1.2.2.1 Maintenance corrective

Selon la (norme FD X60-000) La maintenance corrective exécutée après détection d'une panne et destinée à remettre un bien dans un état dans lequel il peut accomplir une fonction requise [2]. La maintenance corrective peut être :

1.2.2.1.1 Maintenance curative

Selon la (norme FD X60-000) c'est une action de maintenance corrective ayant pour objet de rétablir un bien dans un état spécifié pour lui permettre d'accomplir une fonction requise. Le résultat des actions réalisées doit présenter un caractère permanent. Des modifications et améliorations peuvent être apportées, afin de réduire l'occurrence d'apparition de la défaillance ou d'en limiter l'incidence (figure 1.2 [2]).

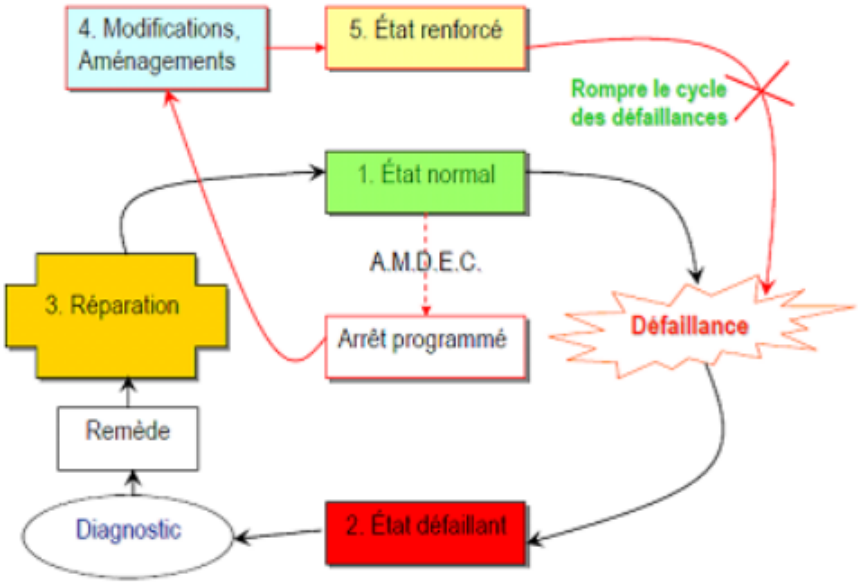

FIGURE 1.2 – Maintenance curative ou réparation .

1.2.2.1.2 Maintenance palliative

Selon la (norme FD X60-000) c'est une action de maintenance corrective destinée à permettre à un bien d'accomplir provisoirement tout ou partie d'une fonction requise Appelée couramment (dépannage), la maintenance palliative

est principalement constituée d'actions à caractère provisoire qui doivent être suivies d'actions curatives (figure 1.3)[2].

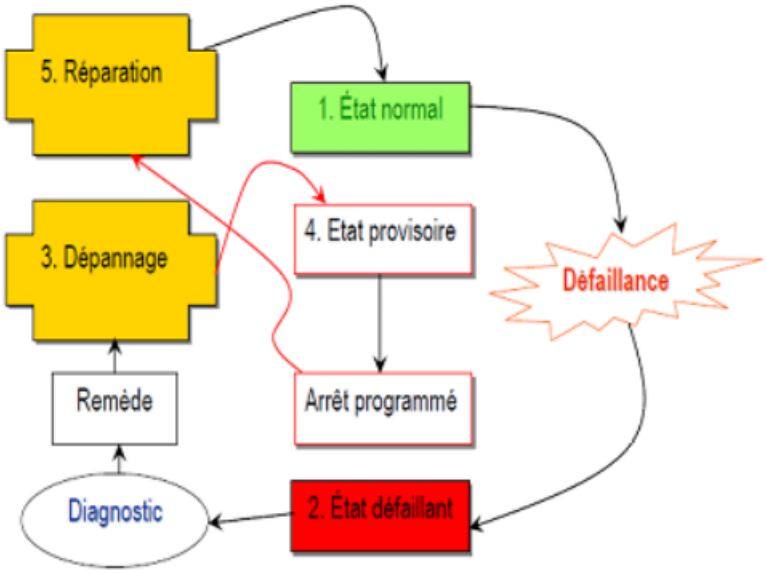

FIGURE 1.3 – Maintenance palliative ou dépannage.

Cette maintenance est utilisée lorsque l'indisponibilité du système n'a pas de conséquences majeures ou quand les contraintes de sécurité sont faibles.

1.2.2.2 Maintenance préventive

La maintenance préventive est réalisée avant la survenue d'une défaillance, elle se définit comme (la maintenance exécutée à des intervalles prédéterminés ou selon des critères prescrits et destinés à réduire la probabilité de défaillance ou la dégradation du fonctionnement d'un bien). Elle agit donc en complément de la maintenance corrective afin de diminuer le risque de défaillance et tendre vers le (zéro)panne [2]. La maintenance préventive est pratiquée selon la panne (les figures : 1.4 et 1.5) :

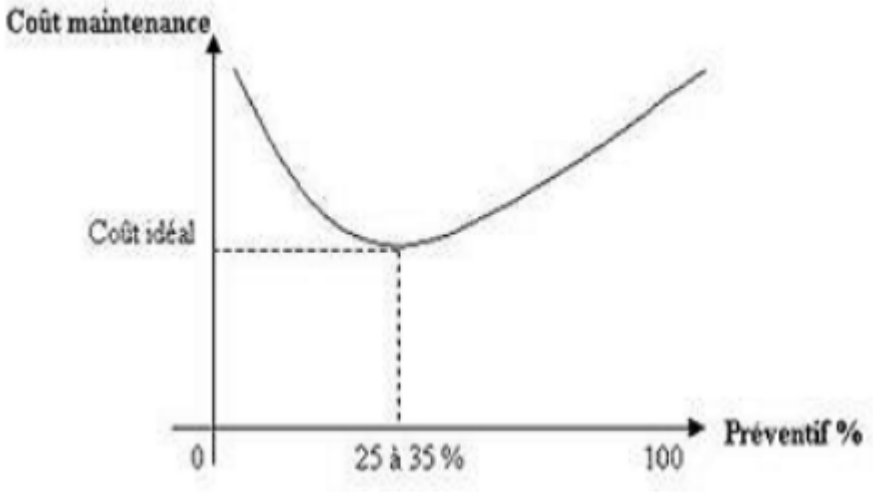

FIGURE 1.4 – Equilibre maintenance corrective (curative)/préventive(Valeurs indicatives pour la mécanique)

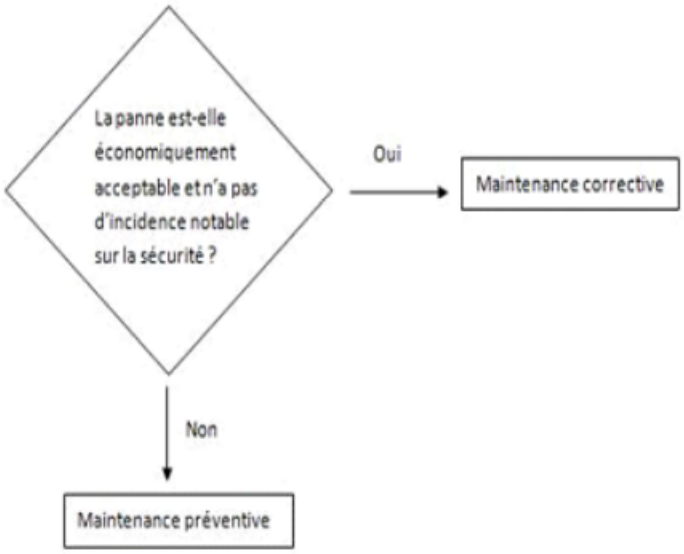

FIGURE 1.5 – Equilibre maintenance corrective (curative)/préventive(Valeurs indicatives pour la mécanique).

1.2.2.2.1 Maintenance systématique

Lorsque la maintenance préventive est réalisée à des intervalles prédéterminés, on parle de maintenance systématique. L'opération de maintenance est effectuée conformément à un échéancier, un calendrier déterminé a priori. Aucune intervention n'a lieu avant l'échéance prédéterminée. L'optimisation d'une maintenance préventive systématique consiste à déterminer au mieux la périodicité des opérations de maintenance sur la base du temps, du nombre de cycles de fonctionnement, du nombre de pièces produites, etc (figure1.6)[5].

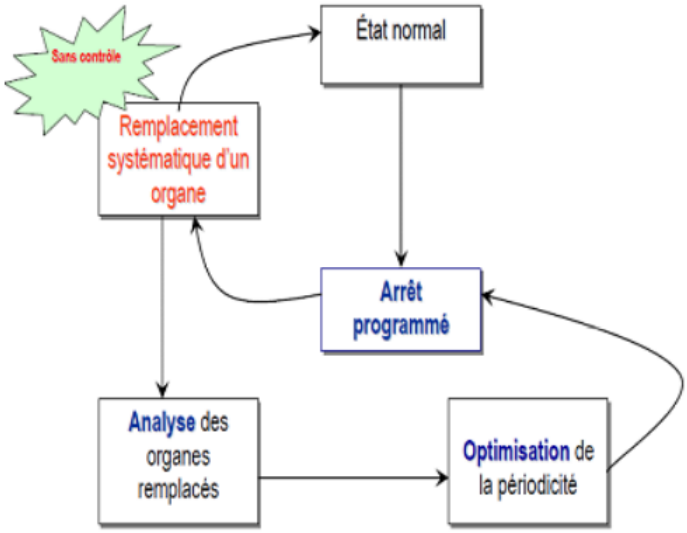

FIGURE 1.6 – Maintenance Systématique.

1.2.2.2.2 Maintenance conditionnelle

Lorsque l'opération de maintenance préventive est subordonnée à l'analyse de l'évolution surveillée de paramètres significatifs de la dégradation ou de la baisse de performance d'une entité, on parle de maintenance conditionnelle. Les paramètres significatifs de la dégradation peuvent être soit des mesures de caractéristiques physiques du système (épaisseur d'un matériau, degré d'érosion, température, pression, ...), soit des informations sur **la durée de vie restant utile : Remaining useful life (RUL)** (on parle alors de **maintenance prédictive**). La planification des interventions repose sur l'existence

et la détermination de seuils critiques pour ces paramètres de dégradation. On parle alors de seuils de décision [5].

1.3 Partie02 :La durée de vie restant utile RUL

1.3.1 Définition de la durée de vie utile RUL

La durée de vie utile restante (RUL) est un élément métrique crucial utilisée dans de nombreux systèmes industriels et définie comme le temps entre l'instant courant après la détection de la dégradation et le moment où la dégradation atteint le seuil de défaillance. Le sens de cette vie utile et sa gestion varie selon le domaine étudié et peut généralement être définie par le concepteur, ingénieurs et utilisateurs de l'actif. Sa prédiction précise permet de planifier la prochaine décision d'entretien à l'avance ce qui diminue les coûts et le temps de maintenance en annulant les maintenances inutiles. Pour illustrer le concept général de RUL, la figure 1.9 montre l'évolution de la détérioration d'un système donné. Dans ce cas, le RUL est le temps entre l'heure actuelle, qui correspond à la condition A, et le temps dans lequel une condition maximale acceptable de détérioration B est atteinte. Ce dernier peut correspondre au moment de la panne (c'est-à-dire pour 100 des la détérioration) ou à un seuil prédéfini dans une valeur inférieure. Par conséquent, on peut définir une durée de vie utile depuis le 'début de la vie' jusqu'à la défaillance ou jusqu'au moment de la condition B. En général, une prédiction RUL appropriée dans A est utile pour effectuer des activités de maintenance de A à B, économiser des ressources et améliorer les processus [6].

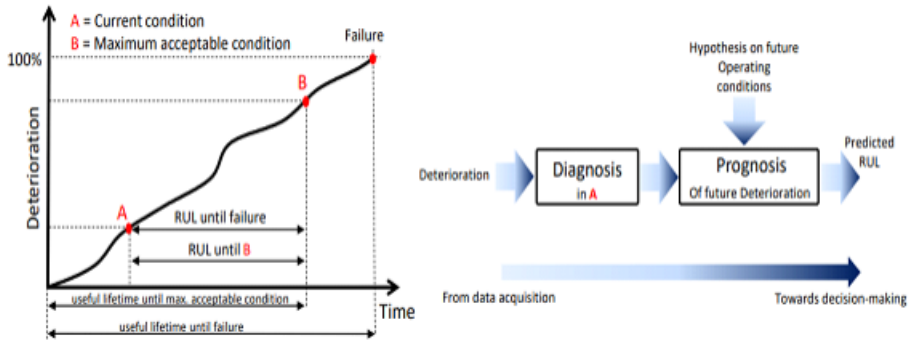

FIGURE 1.7 – Durée de vie utile restante(RUL), son concept et son utilisation

1.3.2 Le pronostic

C'est le processus d'estimation du temps restant(en englais remaining useful life RUL) pour un système ou un composant avant échec. Les pronostics sont utilisés par l'industrie pour gérer les risques résultant de panne d'équipement. Jusqu'à présent, il est toujours basé sur l'expérience des ingénieurs de maintenance. Cependant, la prise de décision humaine n'est pas toujours suffisamment fiable lorsqu'il s'agit des équipements complexes. Par conséquent, ces dernières années, un nombre important de recherches été entrepris pour développer des modèles pouvant être utilisés pour réduire la dépendance de l'industrie sur les individus. Cela peut être fait en effectuant une évaluation de la santé et une estimation de RUL pour les composants surveillés afin de planifier à l'avance les actions de maintenance requises [3].

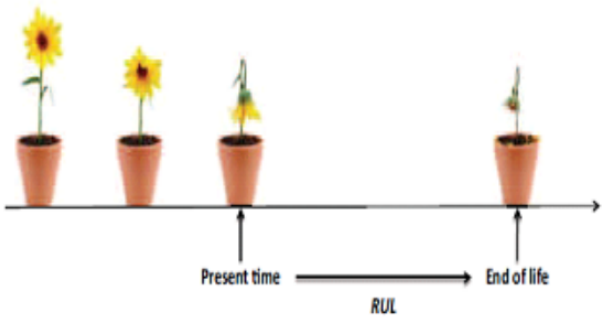

FIGURE 1.8 – Illustration de pronostic.

1.3.3 Classification des approches de pronostics

La classification la plus consensuelle dans la communauté CBM est de considérer les trois approches suivantes : les méthodes guidées par les données, les méthodes basées sur des modèles physiques et les méthodes hybrides, comme le montre la figure 1.9. Toutefois, on définira dans l'approche orientée données, 3 types de démarches : une statisticienne, une liée aux outils d'intelligence artificielle et plus spécifiquement à l'apprentissage automatique et une orientée connaissance [7].

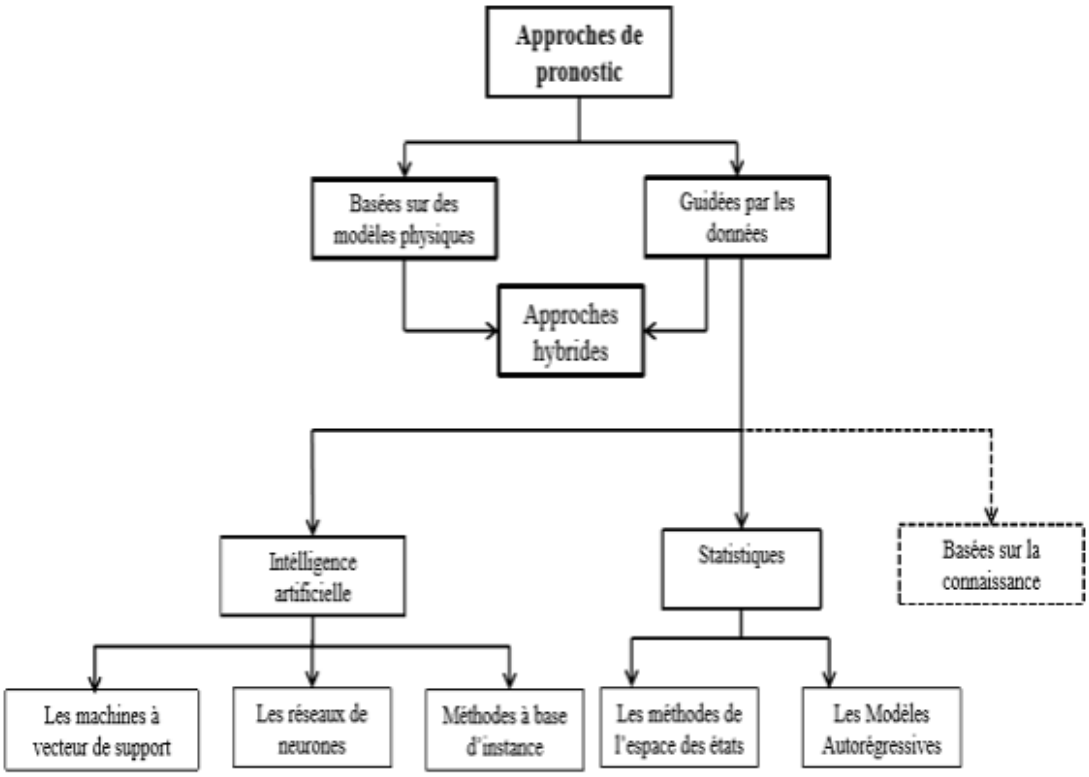

FIGURE 1.9 – classification des approches de pronostic selon la communauté CBM.

1.3.3.1 Les approches basées sur les modèles physiques

Les approches basées sur des modèles physiques ou basées sur la physique de défaillance construisent des modèles analytiques qui sont directement liés aux processus physiques influençant la santé des composants. Ces approches nécessitent des connaissances sur les lois physiques (mécanique, chimie, électricité, hydraulique etc). Le modèle physique peut être décrit par des systèmes dynamiques tels que les équations non-linéaires, les équations différentielles, la représentation d'état, etc. Le principe du pronostic basé sur un modèle physique est résumé à (figure 1.10). Le comportement du système est représenté par un modèle analytique. Ensuite, un test de cohérence est effectué en comparant les données capteurs issues du système réel et les sorties du modèle analytique. Les résidus générés sont évalués. En présence d'un dysfonctionnement, les résidus dépassent un seuil de détection de défauts [7]. L'estimation du temps restant avant défaillance est basée sur la projection de comportement du système et de sa dégradation dans le future. Les modèles physiques traitent par exemple des problèmes d'usures de matériau, de croissance de fissure, de cassure par fatigue et corrosion[8].

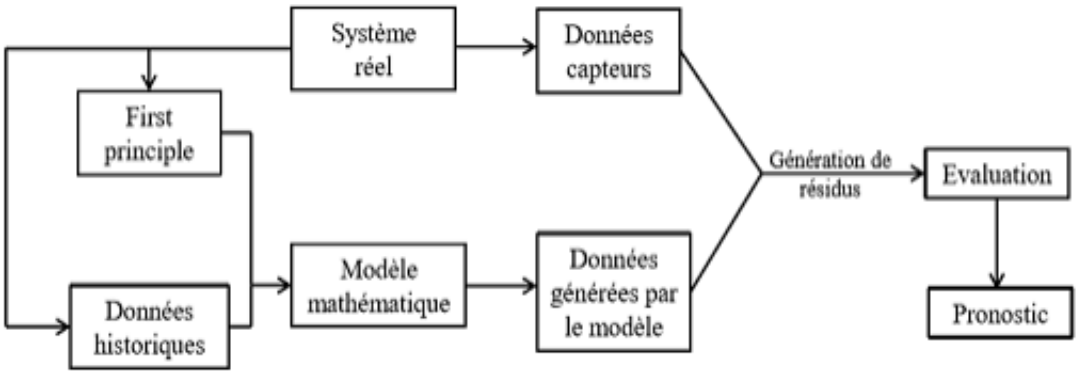

FIGURE 1.10 – organigramme des approches basées sur le modèle physique.

1.3.3.2 Les approches guidées par les données

Les approches guidées par les données cherchent à extraire à partir d'un historique de données de surveillance des modèles d'évolution du fonctionnement du système surveillé allant jusqu'à sa dégradation. Ces approches

comportent deux phases. Une première hors ligne est dédiée à la compréhension et l'apprentissage du comportement de la dégradation. Puis, une deuxième phase en ligne estime l'état de santé courant du système et prédit sa durée de fonctionnement avant défaillance. Elles peuvent être classées en deux catégories : les approches statistiques et celles issues de l'intelligence artificielle [7].

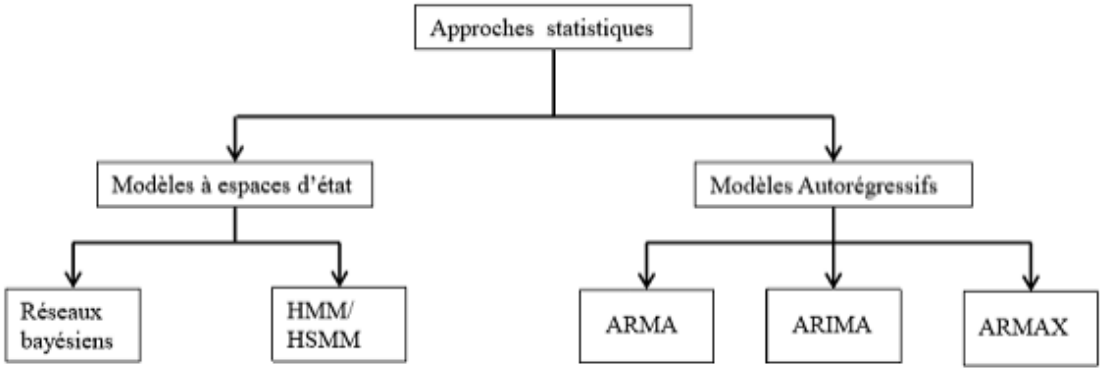

FIGURE 1.11 – La classification des approches statistiques.

1.3.3.2.1 Les statistiques multidimensionnelles

Nous passons en revue les méthodes les plus largement utilisées en pronostic, qui sont également illustrées à la figure 1.11

a- Les Modèles Auto-régressifs : Les modèles ARMA (modèles auto-régressifs et moyenne mobile), ARIMA (modèles moyenne mobile auto-régressif intégré) et ARMAX sont les méthodes de référence dans la modélisation des séries temporelles . Les modèles ARMA et ARMAX ne sont utilisés que pour des données stationnaires. Ils sont améliorés par l'application d'une opération d'intégration dans le modèle ARIMA. Les modèles ARMA ont été utilisés afin d'ajuster les données de vibration des roulements des boites de transmission des hélicoptères, pour prédire les tendances des turbines à flux, afin d'estimer la durée de vie utile restante avant la défaillance d'un système de mouvement de porte d'ascenseur. Les modèles ARIMA sont utilisé pour prédire les caractéristiques vibratoires des machines rotatives [8]. Les modèles régressifs ne nécessitent pas un historique de dégradation et doivent être évalués de manière récursive jusqu'à atteindre un certain seuil, ce qui

engendre une accumulation d'erreur systématique et détériore la performance du prédicateur[7].

b- Les modèles à espace d'état :

— **Les modèles de Markov cachés :** ou HMM, ont été appliqués en pronostic mais ne sont pas adaptés à représenter une structure temporelle. Ils ont été remplacés par les HSMM qui comptent une composante temporelle dans la structure de HMM , une approche basée sur les HSMM est développée afin de prédire le RUL des actionneurs entraînés (motorisés) par un moteur asynchrone. Une méthode a été proposé de combiner les HSMM avec une méthode Monte Carlo séquentielle. Ces outils calculent les probabilités de transition entre les états de santé et leur durée alors que la méthode Monte Carlo définit la relation probabiliste entre les états de santé et les observations de l'équipement surveillé. Les HSMM ont aussi été utilisés pour le pronostic des défaillances d'hélicoptères. Un autre modèle HSMM a été proposé pour le diagnostic et pronostic de UH-60Blackhawk (un hélicoptère utilitaire moyen de l'armée américaine). Le HSMM formé peut être utilisé pour diagnostiquer et estimer l'état courant de santé de l'équipement, et sera à la base de l'estimation du RUL à l'aide d'une équation récursive composée de la durée dans les états et les paramètres du modèle estimé. Aussi développée l'applicabilité des HSMM pour prédire le RUL des arbres à cames utilisés dans les hélicoptères[8].

— **Les réseaux bayésiens :** sont des modèles graphiques directs, ils sont issus de la fusion de la théorie des graphes et des probabilités. Le modèle RB a été étendu à un modèle réseau bayésien dynamique appliqué au pronostic grâce à la modélisation des changements de système au fil du temps. Il est en mesure de surveiller, mettre à jour et prédire les états futurs du système étudié. Une approche bayésienne hiérarchique a été proposée afin de construire un modèle probabiliste d'estimation du RUL. D'autre part, une approche bayésienne a été appliqué dans le but de mettre à jour la distribution des paramètres stochastiques du modèle exponentiel décrivant le processus de dégradation des paliers d'essai. La méthode de pronostic basée sur les RBD à été utilsé et étudier afin de prédire la durée de vie utile restante des trépans d'une machine de forage vertical. Le RUL est considéré comme un état caché et est déduit des données invisibles en utilisant le modèle de prédiction ainsi que des règles d'inférence[8].

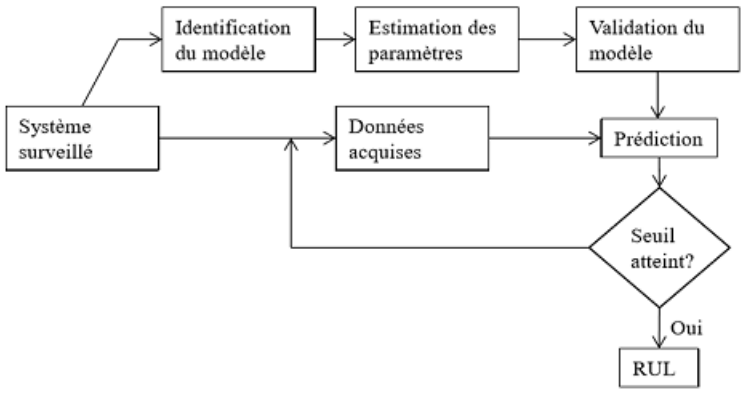

FIGURE 1.12 – Organigramme des approches auto-régressives pour la prédiction du RUL

1.3.3.2.2 L'intelligence artificielle ou apprentissage automatique

Nous recensons les trois types de méthodes les plus utilisées en pronostic : les réseaux de neurones artificiels (RNA), les méthodes de régression à vecteurs de support (SVR), et les méthodes à base d'instances (IBL)[7].

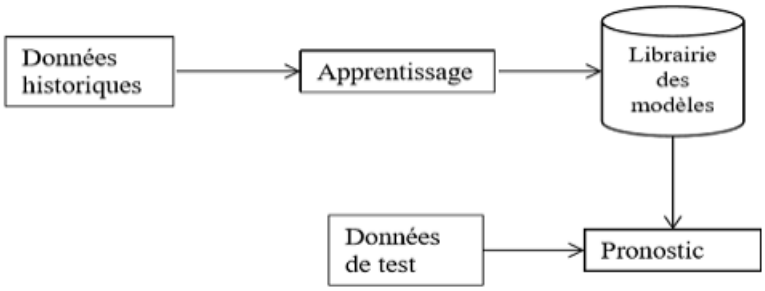

FIGURE 1.13 – Approche d'apprentissage automatique.

— **Les Réseaux de neurones** Les méthodes d'apprentissage automatique ont été étudiées pour la prédiction RUL, les réseaux de neurones (NN) recevant beaucoup d'attention compte tenu de leur capacité à approximer des fonctions directement à partir de données brutes [9]. En fait, les RNA sont connus pour leur capacité à modéliser des systèmes complexes, multidimensionnels et non-linéaires sans la nécessité d'une compréhension physique du comportement du système. Un

RNA est une fonction d'approximation non linéaire à multiples entrées et sorties et peut être utilisé à différentes fins parmi les quelles le pronostic. Selon les entrées du modèle RNA, le dernier a la souplesse nécessaire pour exécuter différentes tâches, comme la prédiction du temps restant avant la défaillance par exemple [7]. Récemment, des méthodes Deep NN ont été proposées pour pronostiquer des problèmes contenant de grandes quantités de données d'entrée temporelles [10].

— **Les machines a vecteurs de support :** les machines à vecteurs de support (SVM) ont été développées par Vapnick et ses collègues. Les SVM sont à la base d'un système d'apprentissage qui représente un espace de caractéristiques d'entrée dans un espace de plus grande dimension. Cela est appelé l'astuce de noyaux (kernel trick) et donne aux SVM la capacité à traiter des données non-linéaire. Ces méthodes ont une bonne généralisation et ont deux types d'application : la classification à vecteurs de support, utilisée pour résoudre le problème de diagnostic, et la régression à vecteurs de support, utilisée pour résoudre le problème de pronostic[7].

— **Méthodes à base d'instances**
les approches à base d'instances prennent appui sur un ensemble de données, sur lequel des techniques d'apprentissage sont appliquées. Elles se basent sur le principe : les expériences acquises lors de la résolution d'un problème peuvent être utilisées pour résoudre des cas similaires". Un algorithme souvent utilisé pour ce type d'approches est les k plus proches voisins. Une instance est généralement représentée par un vecteur dans un espace euclidien de dimension n et l'objectif est de trouver les instances similaires. L'avantage de ces méthodes est l'apprentissage incrémental. Quand un problème est résolu avec succès, l'expérience est conservée afin de résoudre de nouveaux problèmes similaires. Les méthodes à base d'instances font partie du raisonnement à partir de cas (RàPC) qui est un paradigme de raisonnement par analogie. Les nouveaux problèmes sont résolus en se basant sur la connaissance spécifique acquise lors de la résolution d'anciens problèmes [7].

1.3.3.3 Les approches hybrides

En combinant des approches basées sur des modèles et basées sur des données, les approches hybrides visent à utiliser les avantages des deux ap-

proches et à éviter les désavantages [11] Par exemple, l'estimateur bayésien récursif est une approche importante pour intégrer des modèles du système avec les données de capteurs et activer les modèles dits hybrides. Les estimateurs bayésiens récursifs ont deux procédures communes à chaque itération : prédire et actualiser. Deux variantes importantes de l'estimateur bayésien sont le filtre de Kalman et le filtre particulaire. Le filtre de Kalman est approprié pour les modèles espace-état linéaires avec un bruit gaussien tandis que le filtre particulaire peut être utilisé pour les modèles non linéaires avec du bruit non gaussien[7].

1.4 Conclusion

Dans ce chapitre, nous avons abordé dans la première section l'essentiel des concepts actuels attachés à la maintenance, en précisant les principaux types associés et en s'intéressant à la maintenance conditionnelle. Dans la deuxième section, on a parlé de la durée de vie résiduelle (RUL) ainsi que les approches effectuant l'estimation. Dans le chapitre suivant nous parlerons du deep learning, l'une des méthodes de l'IA qui est une des approches guidées par les données.

Chapitre 2

L'apprentissage profond (Deep Learning)

2.1 Introduction

L'objectif de la recherche sur intelligence artificielle (IA) est de fournir un système informatique doté de capacités de réflexion similaires à celles des humains. Il y a donc un défi à comprendre la pensée humaine, mais sur tout à la modéliser et à la reproduire [12]. L'intelligence artificielle est devenue un sujet brûlant dans les médias et les revues scientifiques en raison de nombreuses réalisations, dont beaucoup sont le résultat d'avancées dans le domaine de l'apprentissage automatique. L'IA est basée sur une démarche d'apprentissage afin de reproduire une partie de l'intelligence humaine à travers une application, un système ou un processus. La reconnaissance faciale, la perception visuelle et autre sont des exemples de systèmes d'intelligence artificielle. Le machine Learning (ML) est un sous-domaine de l'IA qui utilise les réseaux neuronaux artificiels (ANN) pour imiter la façon dont les êtres humains prennent des décisions. Le machine Learning permet aux ordinateurs de développer des modèles d'apprentissage par eux-mêmes, sans aucune programmation, à partir de gros ensembles de données. La couche immédiatement inférieure est occupée par le Deep Learning (DL), est l'une des nombreuses approches du machine Learning qui connue un grand succès dans ces dernières années, Pour illustrer la relation entre ces termes, nous pouvons utiliser des cercles concentriques : • Intelligence artificielle IA (intelligence artificielle) : le cercle plus large est l'idée qui a émergé en premier

dans ce domaine • Apprentissage automatique (machine Learning) : au milieu, il a prospéré plus tard après l'IA • Apprentissage approfondie (Deep Learning) : le plus petit cercle est une expansion de l'IA actuellement la figure 1 représente la relation entre MLet IA (figure2.1) [13] Dans ce chapitre,

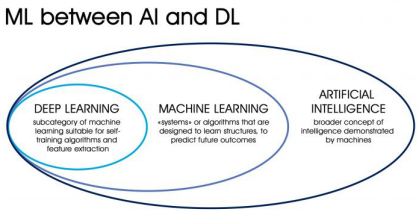

FIGURE 2.1 – La relation Entre L'IA et ML et le Deep Learning

nous allons présenter l'apprentissage profond, ses différents algorithmes ainsi que ses applications dans différents domaines .

2.2 Définition de l'apprentissage automatique

L'apprentissage automatique (machine learning) est le développement et l'analyse de méthodes, puis leur mise en œuvre, ce qui permet à la machine de développer ses tâches par l'apprentissage, car elle atteint l'exécution de tâches difficiles ou impossibles à réaliser par des moyens algorithmiques traditionnels[14]. L'apprentissage automatique fait référence à la capacité d'un ordinateur à concevoir des programmes qui s'améliorent avec des expériences passées. Les algorithmes utilisés se basent sur un ensemble de données contenues dans des connaissances afin de pouvoir ensuite adapter leurs analyses en réponse. Cela peut aider une machine à prendre des décisions face à des situations inconnues. De plus, grâce à ce processus d'apprentissage, un robot pourra alors réaliser des tâches jusqu'à présent difficiles voir impossibles à réaliser par des algorithmes plus classiques[14].

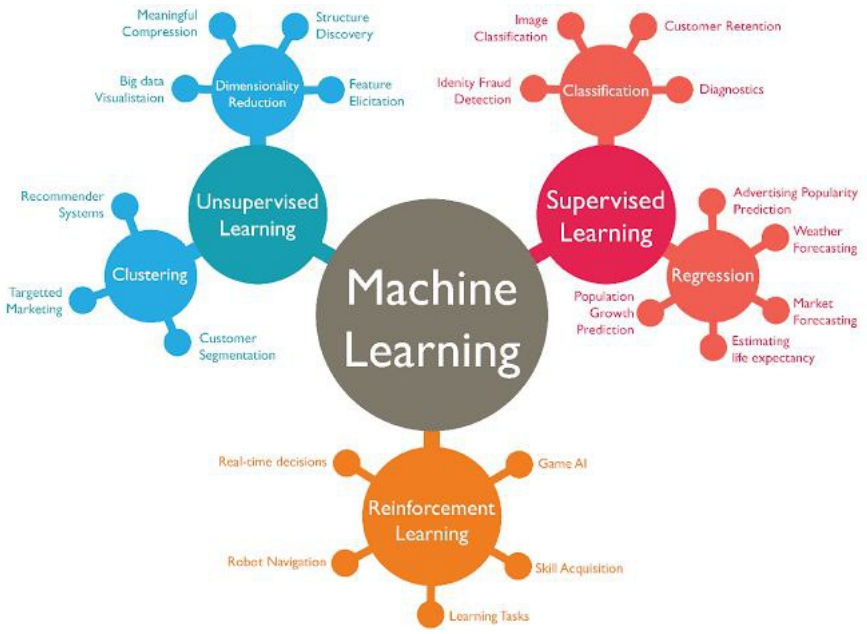

FIGURE 2.2 – Apprentissage automatique

2.3 Les types d'apprentissage

L'apprentissage peut être supervisé, semi-supervisé, non supervisé et renforcé.

2.3.1 Apprentissage Supervisé

Le processus se déroule en deux phases. La première est appelée la phase d'apprentissage durant laquelle l'oracle fournit une étiquette à chaque exemple disponible dans l'environnement en les associant à une classe : $x_1, x_2, ..., x_m, ...etx_n, x_O, ...$ dans le cas présenté ici. Ensuite il classifie les exemples en élaborant un échantillon de données étiquetées noté : $S_m = (x_1, U_1), (x_2, U_2), \ldots, (x_m, U_m)$. Il guide ainsi l'apprenant en lui fournissant des exemples sur ce qu'il doit apprendre à savoir faire. La seconde étape est nommée phase de test. Elle consiste à tenter de prédire l'étiquette d'un nouvel exemple en utilisant une fonction apprise à partir de S_m notée h. Avec cette hypothèse, l'apprenant peut associer $Y_n = h(x_n)$ à chaque x_n, $Y_o = h(x_o)$ pour chaque xo, et etc... Plus ces valeurs seront proches de celles que l'oracle aurait fournies à sa place, meilleur sera l'apprentissage. L'apprend apprend donc par lui-même à classer

ou à décider d'une action comme le fait l'oracle[14].

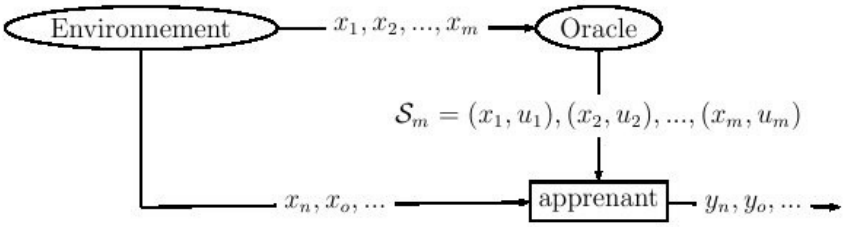

FIGURE 2.3 – schéma de l'apprentissage supervisé.

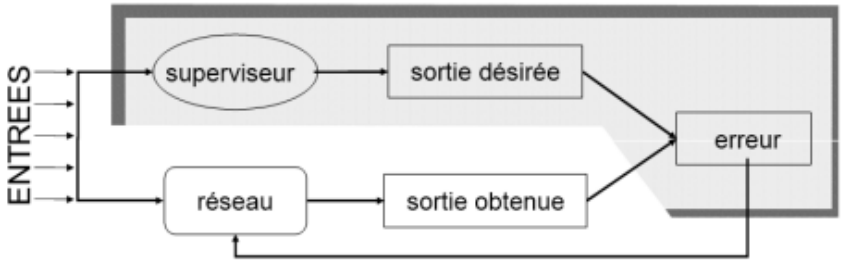

FIGURE 2.4 – Schéma d'un modèle supervisé.

2.3.2 Apprentissage Non-Supervisé

L'apprentissage non supervisé est le contraire de l'apprentissage supervisé. Dans ce cas,il n'y a pas d'oracle qui donne des étiquettes, le système contient seulement des exemples. Le nombre de classes est alors inconnu, et c'est à l'algorithme de déterminer la structure des données. Ce dernier est autodidacte puisque qu'il vise à tenter de faire une extraction de connaissances organisées à partir de données fournies sans indication supplémentaire. A partir d'un groupe hétérogène de données, l'apprentissage supervisé procède par regroupement en identifiant les données qui sont similaires puis il les regroupe dans un sous-groupe. Prenons comme exemple, un ensemble de points dans un espace quelconque, on peut se donner comme objectif de les regrouper en "paquets", en appliquant une méthode de classification de données appelée "clustering" (ou "classification non supervisée")[14].

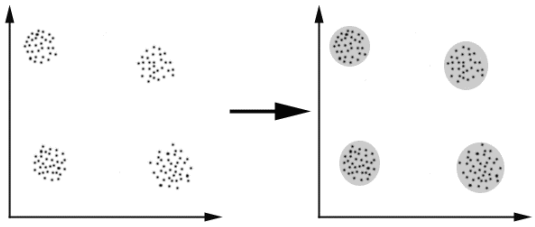

FIGURE 2.5 – Exemple de clustering.

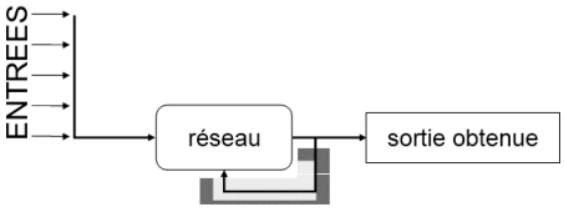

FIGURE 2.6 – Schéma d'un modèle non supervisé.

2.3.3 Apprentissage par renforcement

Les données utilisées dans l'apprentissage supervisé sont les mêmes que les données d'entrée dans l'apprentissage par renforcement, bien que l'apprentissage soit dirigé à travers l'environnement sous la forme de récompenses ou de punitions données en fonction de l'erreur survenue lors de l'apprentissage[15].

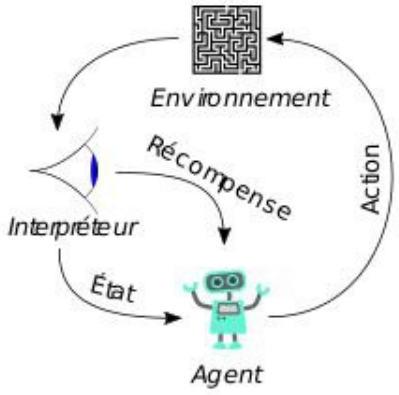

FIGURE 2.7 – Schéma d'un modèle par renforcement.

2.4 Principe des réseaux de neurones

Un réseau de neurones est un outil puissant pour modéliser les relations complexes entre les données d'entrée et de sortie. Cette technologie a été développée dans le but d'atteindre un système artificiel intelligent capable d'effectuer des tâches similaires aux tâches effectuées par le cerveau humain. Connaissances par apprentissage, et ses connaissances sont stockées dans des connexions interneurones connues sous le nom (poids synaptiques) [16].

2.5 Le fonctionnement des réseaux de neurones

Le fonctionnement des réseaux de neurones consiste à répartir les valeurs des variables dans des automates (les neurones).La responsabilité de ces unités est de combiner leurs informations entre elles afin de déterminer la valeur du coefficient de discrimination. Cela montre la capacité de la RN à distinguer de la connectivité de ces unités (les unités chargées d'intégrer l'information). Chaque neurone reçoit des informations numériques de ses cellules voisines, qui sont associées à un poids qui représente la force de la connexion. Les calculs sont effectués dans chaque cellule, et le résultat de ce processus est envoyé aux neurones en aval[17]. Dans la littérature, on parle de perceptron simple et de perceptron multicouche.

2.6 Perceptron simple

Il s'agit d'un neurone binaire, c'est-à-dire dont la sortie vaut 0 ou 1. Pour calculer cette sortie, le neurone effectue une somme pondérée de ses entrées (chaque entrée possède un poids) :$Y = (W_1X_1 + W_2X_2)$, puis applique une fonction d'activation à seuil : si la somme pondérée dépasse une certaine valeur, la sortie du neurone est 1, sinon elle vaut 0. Il ne peut donc faire que de la classification, puis de la prédiction[18].

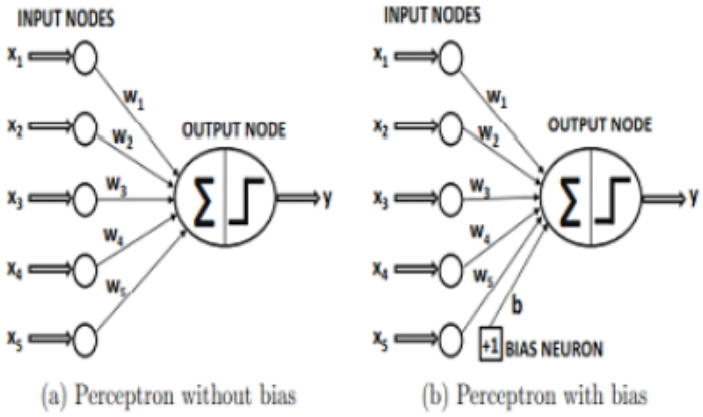

FIGURE 2.8 – perceptron simple avec et sans le bias.

2.7 Perceptron multicouches (PMC)

La famille de réseau de neurones majoritairement employée est le perceptron multicouches. À lui seul ce type de réseau recouvre plus de 95 des applications scientifiques et industrielles. Il comporte quelques dizaines à quelques centaines de neurones dans les cas usuels, voir plusieurs milliers pour les applications graphiques. Le PMC est un modèle de réseau à propagation par couche (Figure 2.9). Les neurones y sont organisés en couches successives : une couche d'entrée, une couche de sortie et entre les deux une ou plusieurs couches intermédiaires, appelées aussi couches cachées. Il n'existe pas de connexion entre les neurones d'une même couche, en revanche tout neurone d'une couche est connecté à tous les neurones de la couche suivante. La couche d'entrée n'est pas une réelle couche de neurones car elle se contente de coder les variables d'observation. La couche de sortie code la variable de discrimination. Les valeurs d'activité des neurones sont propagées dans le réseau, de l'entrée vers la sortie, sans retour arrière. La présence d'une couche cachée permet de modéliser des relations non linéaires entre les entrées et la sortie. En théorie une seule couche cachée suffit, mais le fait de disposer d'une seconde couche cachée permet de modéliser plus facilement une fonction de discrimination non continue. En pratique, la plupart des problèmes sont résolus avec un ou deux niveaux, trois au maximum[17]. La figure 2.9 illustre l'estimation de l'âge au décès à partir de l'observation de critères osseux de la surface sacro-pelvienne iliaque. Les entrées correspondent à l'observation

de critères morphologiques sur la surface sacro.

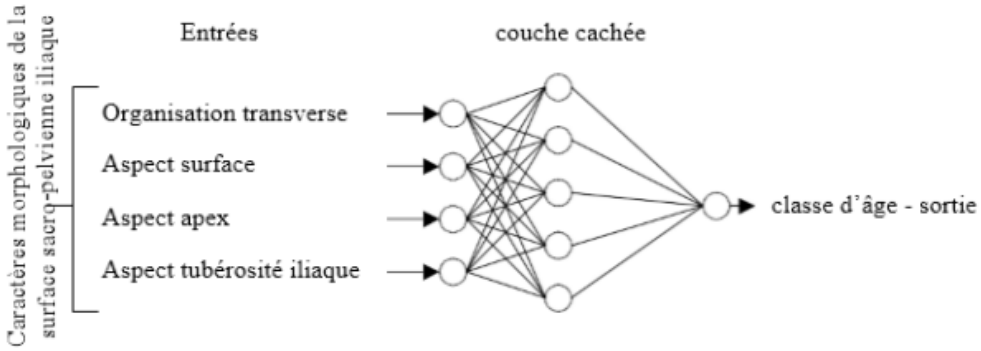

FIGURE 2.9 – Schéma d'un perceptron multicouches.

2.8 La fonction d'activation

La fonction d'activation est la fonction de transfert qui relie la sommation pondéré au signal de sortie. Il y'a plusieurs types de fonction d'activation, parmi lesquelles nous trouvons [19] :

2.8.1 La fonction tangente hyperbolique

tanh peut être considéré comme un candidat approprié pour la protection de la vie privée dans les modèles d'apprentissage en profondeur. D'une part, tanh limite et désactive les valeurs d'entrée maximales, ce qui aurait des effets significatifs sur la mémoire système, ce qui pourrait entraîner une mémoire supplémentaire involontaire de ces valeurs. D'autre part, l'amplitude relative entre les valeurs d'entrée est maintenue de sorte qu'on peut s'attendre à moins de dommages aux installations. Le maintien d'un signe (positif ou négatif) pour l'entrée est une caractéristique essentielle de tanh qui empêche la dégradation des performances[20].

2.8.2 La fonction sigmoïde

Cette fonction est l'une des plus couramment utilisées. Il est borné entre 0 et 1, et il peut être interprété stochastiquement comme la probabilité que

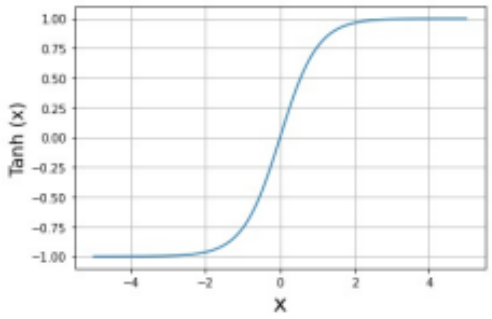

FIGURE 2.10 – Fonction tangente hyperbolique.

le neurone s'active, et il est généralement appelé la fonction logistique ou le sigmoïde logistique[21].

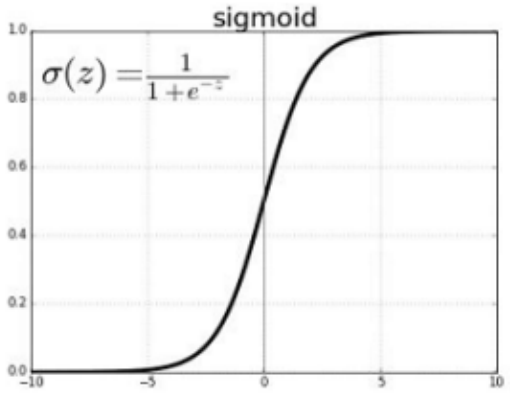

FIGURE 2.11 – représentation graphique de la fonction sigmoïde.

2.8.3 La fonction RELU

La fonction RELU est probablement la plus proche de sa correspondante biologique. Cette fonction est récemment devenue le choix de nombreuses tâches (notamment en computer vision). Comme dans la formule ci-dessus, cette fonction renvoie 0 si l'entrée z est inférieure à 0 et retourne z lui-même si il est plus grande que 0[21].

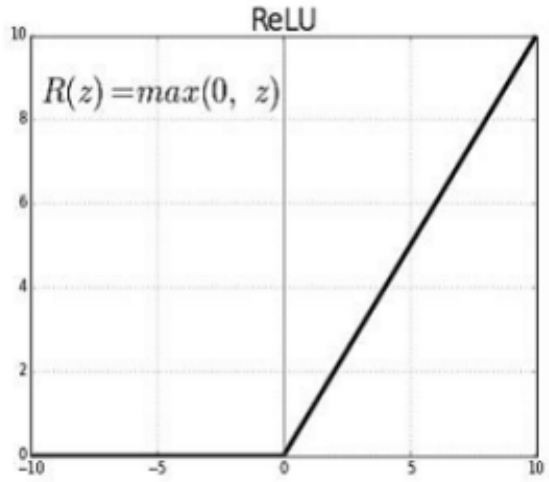

FIGURE 2.12 – représentation graphique de la fonction Relu.

2.8.4 La fonction linear

Comme on l'a vu, la fonction est une ligne ou linéaire. Par conséquent, la sortie des fonctions ne sera confinée entre aucune plage [22].

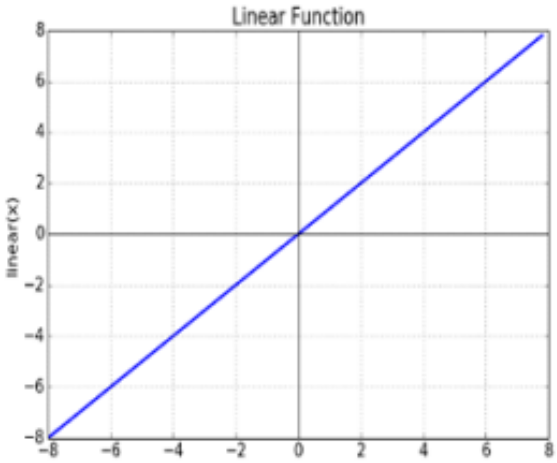

FIGURE 2.13 – représentation graphique de la fonction linear.

2.9 Définition du deep learning

Les réseaux de neurones dits "profonds" (Deep Neural Networks en anglais) sont des MLP avec un nombre de couches supérieur à trois. Pendant longtemps les méthodes d'apprentissage pour ce type de réseaux de neurones acycliques ne permettaient pas de converger vers un réseau de neurones performant. Des avancées majeures sur les méthodes d'entraînement et le choix de la fonction de transfert Rectified Linear Unit (ReLU) qui minimise l'impact de la dilution du gradient dans les couches basses du réseaux ont permis d'utiliser des réseaux de neurones de plus en plus gros [22]. Il exist deux types de réseaux de neurones profonds :

— **Les réseaux de neurones Feed forward.**
— **Les réseaux de neurones récurrents** utilisés pour les données séquentielles telles que le texte ou séries chronologiques(time series)[23].

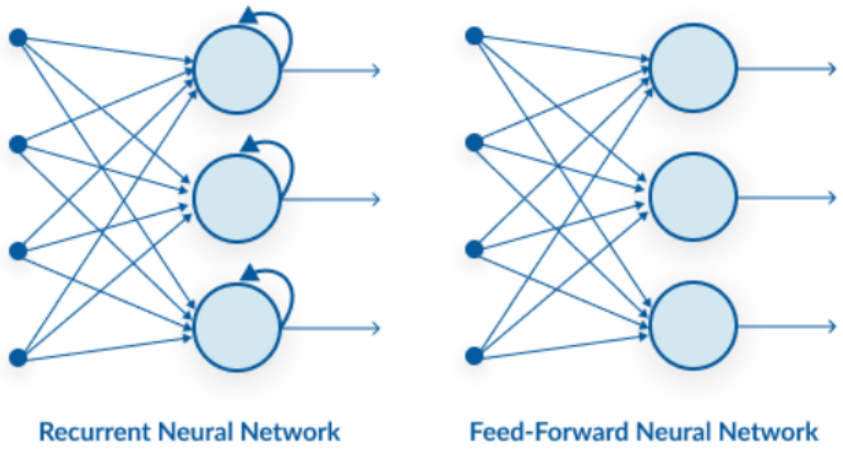

FIGURE 2.14 – transforme une couche cachée de ANN en RNN.

2.9.1 Réseaux de neurone FeedForward

Ce réseau de neurones est l'une des formes les plus simples d'ANN, où les données ou l'entrée se déplacent dans une direction. Les données passent par les nœuds d'entrée et sort sur les nœuds de sortie. Ce réseau neuronal peut avoir ou non les couches cachées. Tous simplement, il a une onde propagée

en façade et aucune rétropropagation en utilisant généralement une fonction d'activation [24]. Résaux de neurones convolutifs CNN

Les réseaux de neurones convolutifs sont similaires aux réseaux de neurones feed forward, où les neurones ont des poids apprenants et biases. Son application a été dans le signal et traitement d'image qui reprend Open CV dans le domaine de la vision par ordinateur. ConvNet sont appliqués dans des techniques comme le traitement du signal et techniques de classification d'images [25]. Les CNN sont très efficaces pour le traitement d'image, car une caractéristique peut se produire n'importe où dans l'image. Au fur et à mesure qu'une couche alimente sa sortie dans la couche suivante, les entités extraites peuvent devenir hiérarchiquement et progressivement plus complexes. Le processus d'optimisation des paramètres tels que les noyaux est appelé apprentissage, qui est effectué de manière à minimiser la différence entre les sorties et les ground truth labels grâce à un algorithme d'optimisation appelé rétropropagation(back propagation) et descente de gradient, entre autres [25].

2.9.1.1 Les couches CNN

CNN est une construction mathématique généralement composée de trois types de couches (ou blocs de construction) :la convolution, le pooling et les couches fully connected [25].

- **Fully connected** Couche classique de perceptron et dernière couche d'un réseau profond qui opère la discrimination finale entre par exemple des images à reconnaître. Les couches précédentes construisant, extrayant, des caractéristiques (features) de celles-ci.
- **Convolution** opère une convolution sur le signal d'entrée en associant une réduction de dimension.
- **Pooling** réduction de dimension en remplaçant un sous-ensemble des entrées (sous-image) par une valeur, généralement le max[26].

2.9.1.2 L'algorithme de rétropropagation

L'algorithme d'apprentissage de rétropropagation consiste dans un premier temps à circuler vers l'avant les données d'entrées jusqu'à l'obtention d'une entrée calculée par le réseau, puis la seconde étape est de comparer la sortie calculée à la sortie réelle connue (Rumelhart et al 1986). Les poids sont modifiés de telle sorte qu'à la prochaine itération, l'erreur commise entre

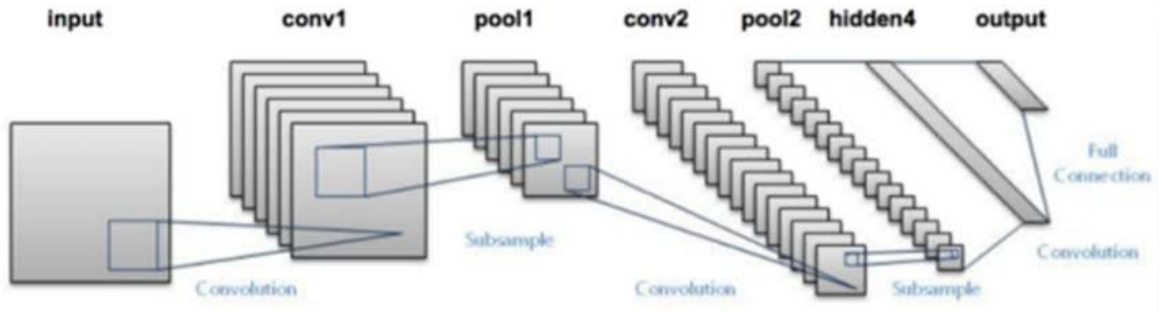

FIGURE 2.15 – Le schéma de CNN.

la sortie calculée est minimisée, en prenant en considération la présence des couches cachées, l'erreur est rétro-propagé vers l'arrière jusqu'à la couche d'entrée tout en modifiant la pondération. Le processus est répété sur tous les exemples jusqu'au temps où l'on obtienne une erreur de sortie considérée comme négligeable [27].

2.9.2 Les réseaux de neurones récurrents (RNN ou Recurrent Neural Networks)

Les humains ne commencent pas leur réflexion à partir de zéro chaque seconde. En lisant cet essai, vous comprenez chaque mot en fonction de votre compréhension des mots précédents. Vous ne jetez pas tout et recommencez à penser à partir de zéro. Vos pensées ont de la persévérance. Les réseaux de neurones traditionnels ne peuvent pas le faire, et cela semble être une lacune majeure. Par exemple, imaginez que vous souhaitiez classer le type d'événement qui se produit à chaque étape d'un film. On ne sait pas comment un réseau de neurones traditionnel pourrait utiliser son raisonnement sur des événements antérieurs dans le film pour les informer plus tard.

Les réseaux de neurones récurrents résolvent ce problème. Ce sont des réseaux avec des boucles qui permettent à l'information de persister. Dans la figure 2.17 ci-dessus, un segment de réseau neuronal ; "A" regarde une entrée X_t et fournit une valeur h_t. Une boucle permet de passer des informations d'une étape du réseau à l'autre. Ces boucles rendent les réseaux neuronaux récurrents un peu mystérieux. Cependant, si vous pensez un peu plus, il s'avère qu'ils ne sont pas tous différents d'un réseau de neurones normal. Un réseau de neurones récurrent peut être considéré comme des copies multiples

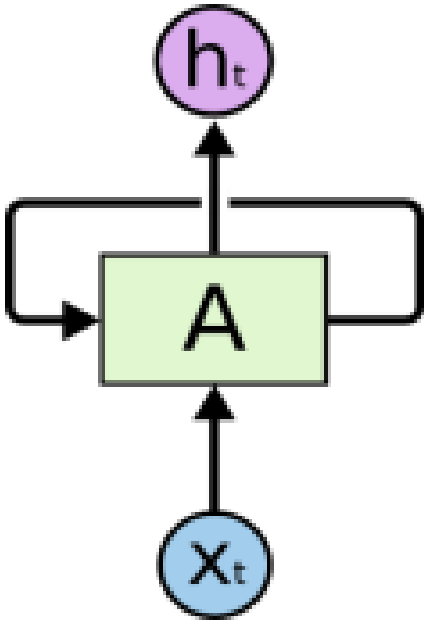

FIGURE 2.16 – Les réseaux neuronaux récurrents ont des boucles.

du même réseau, chacune transmettant un message à un successeur comme le montre la figure . Considérez ce qui se passe si nous déroulons la boucle : Cette

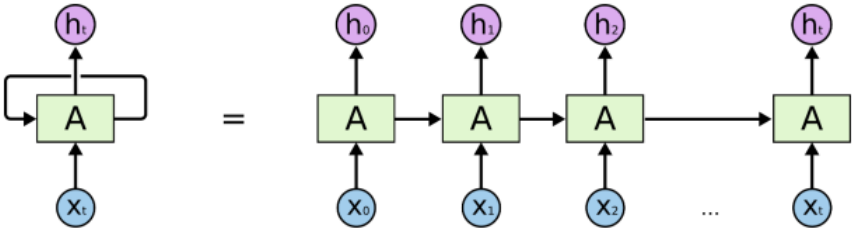

FIGURE 2.17 – Un réseau neuronal récurrent déroulé.

nature en chaîne révèle que les réseaux neuronaux récurrents sont intimement liés aux séquences et aux listes. Ils sont l'architecture naturelle du réseau de neurones à utiliser pour de telles données. Au cours des dernières années, il y a eu un succès incroyable en appliquant les RNN à une variété de problèmes : la reconnaissance de la parole, la modélisation du langage, la traduction, le sous-titrage des images ...etc[28].

2.10 Les types des réseaux RNN

Comme pour les FFNN, de nombreux types de RNN ont été développés au cours des 30 dernières années tels que les réseaux d'Elman , les réseaux de Jordan et les Echo State Networks . Au cours des dernières années, un type de RNN est devenu la norme grâce à ses excellentes performances sur des tâches aussi nombreuses que variées : les réseaux de neurones à base de cellules Long Short-Term Memory (LSTM). Parmi ces types de RNN nous citons quelques uns :

2.10.1 Les réseaux de Hopfield

Ce type de réseaux utilise un apprentissage non-supervisé, il est particulièrement utilisé dans la résolution de problèmes d'optimisation. Les réseaux de Hopfield sont considérés comme des réseaux totalement connectés et il n'y a aucune différenciation entre les neurones d'entrée et de sortie . Ce genre de réseaux opère comme une mémoire associative nonlinéaire qui a la capacité de discerner un objet stocké dans un espace de données) [29].

2.10.2 LSTM

Les cellules de mémoire à long terme court (LSTM) ont été introduites par Hochreiter et Schmidhuber (1997) et ont été créées afin de pouvoir apprendre les dépendances à long terme [30].

LSTM(de l'anglais : Long Short Term Memory) est l'une des topologies récurrentes de réseaux de neurones artificiels. Comme dans le cas des réseaux récurrents, un LSTM est composé de couches de neurones avec une connexion récurrente de sorte que l'état précédent du neurone dans un pas de temps précédent soit utilisé comme contexte pour formuler une sortie. Les unités de calcul de LSTM sont appelées cellules ou blocs de mémoire. Contrairement à d'autres RNNs, LSTM est une architecture spécialement conçue pour résoudre le problème du gradient. Pour un neurone, le gradient est l'erreur relative à ce neurone. Il peut en quelque sorte être vu comme la contribution du neurone à l'erreur globale.Le principal défi auquel sont confrontés les RNNs est de savoir comment les former efficacement.

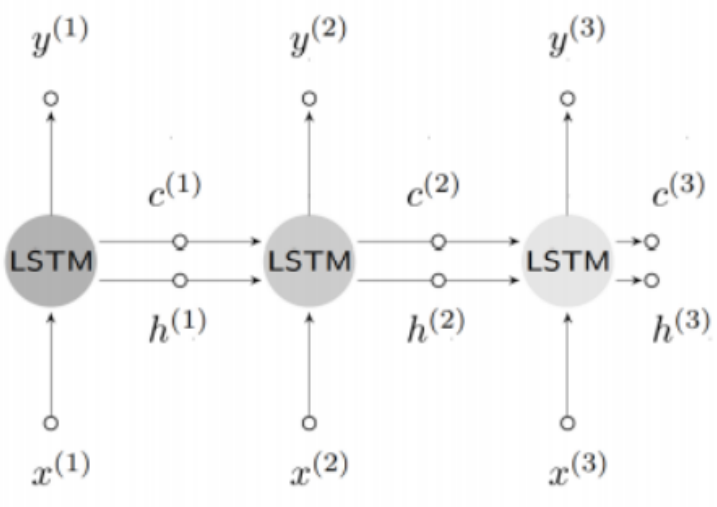

FIGURE 2.18 – Structure de LSTM.

2.10.3 GRU

Le réseau GRU appartient à l'algorithme RNN. La couche d'entrée est constituée de serval neurones qui est déterminée par la longueur de l'espace des caractéristiques. La couche de sortie est également déterminée par la taille de l'espace de sortie. La couche cachée est la partie clé de l'algorithme GRU-RNN qui comprend les cellules de mémoire avec les fonctions principales de GRU. La structure d'une cellule de mémoire GRU est représentée sur la figure2.19 . La sortie d'une cellule de mémoire dépend de la porte de réinitialisation tr et de la porte de mise à jour t_z.

La porte de réinitialisation tr et la porte de mise à jour t_z sont liées à t_x et t_1 h . $t - x$ est la séquence d'entrée respective et t_1 h est la sortie de la cellule de mémoire à l'instant précédent. Ces deux portes différentes jouent des rôles différents dans le réseau. La porte de réinitialisation contrôle le niveau d'ignorance de l'information en t 1 h . L'information est davantage ignorée lorsque la valeur de la porte de réinitialisation est plus petite. La porte de mise à jour est conçue pour contrôler l'impact des informations précédentes sur l'état actuel. Plus la valeur de mise à jour est grande, plus les informations précédentes sont utilisées pour déterminer l'état actuel. La propagation vers l'avant peut être représentée sur la base des équations suivantes :

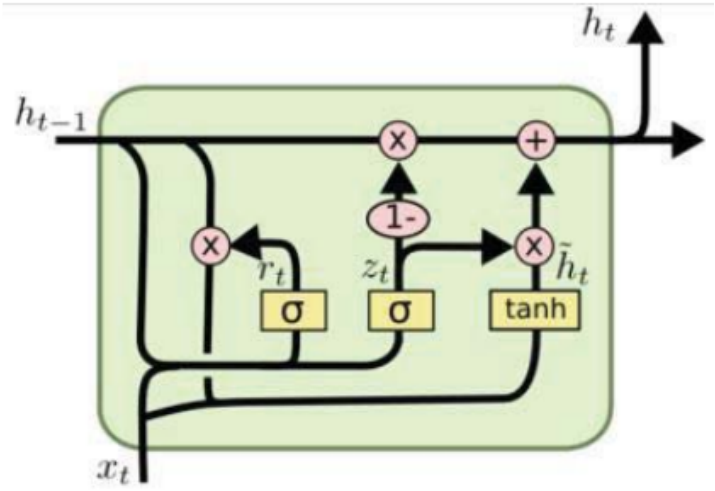

FIGURE 2.19 – La structure d'une cellule mémoire GRU.

Reset gate

$$r_t = \sigma(W_r.[h_{t-1}, x_t])$$

Update gate :

$$z_t = \sigma(W_z.[h_{t-1}, x_t])$$

Output :

$$\widetilde{h_t} = \tanh\left(W_{\widetilde{h}}.[r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t}$$

$$y_t = \sigma(W_0.h_t)$$

2.10.4 JANET

Ce type de réseaux semble répondre aux questions de savoir si toutes les portes d'un LSTM sont strictement nécessaire, fait déjà réfuté avec la proposition du Gated Recurrent Unit (GRU). De cette manière, un nouveau réseau appelé JANET qui ne garde que la porte de l'oubli et la cellule mémoire obtenant un réseau qui nécessite moins de puissance de calcul et est un modèle plus général [31]. L'unité récurrente fermée (GRU) peut être considérée comme une simplification du LSTM, qui n'utilise pas d'états de

cellule explicites. Une autre différence est que le LSTM contrôle directement la quantité d'informations modifiées dans l'état caché en utilisant des portes d'oubli et de sortie séparées. D'autre part, un GRU utilise une seule porte de réinitialisation pour atteindre le même objectif [32].

2.10.5 La machine de Boltzmann profonde (DBN ou Deep Belief Network) :

DBN est un réseau de neurones constitué de plus d'une machine Boltzmann restreinte (RBM). RBM et DBN ont été conçus par Geoffrey Hinton[31]. RBM est un algorithme utile pour la réduction des dimensions, la régression, le filtrage collaboratif, classification, apprentissage de fonctionnalités et modélisation de thèmes. Il C'est un réseau de neurones à deux couches peu profond composé d'une couche d'entrée visible et une couche de sortie cachée. Ils sont les blocs de construction de DBN. Pour chaque RBM la couche visible représente les inputs et la couche cachée reçoit les inputs de la couche visible, fait les calculs ensuite elle envoie les résultats à la couche visible. Dans la DBN chaque couche cachée du RBM représente la couche visible de la RBM suivante comme le montre la figure 2.20 [33].

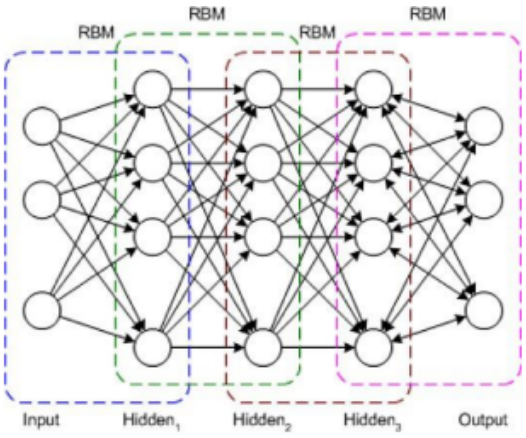

FIGURE 2.20 – DBN.

2.11 Long short term memory

Définition Une couche LSTM consiste en un ensemble de blocs connectés de manière récurrente, appelés blocs mémoire. Ces blocs peuvent être considérés comme une version différenciée des puces de mémoire dans un ordinateur numérique. Chacun contient une ou plusieurs cellules mémoire connectées de manière récurrente et trois unités multiplicatives - les portes d'entrée, de sortie et d'oubli - qui fournissent des analogues continus des opérations d'écriture, de lecture et de réinitialisation pour les cellules [34].

2.11.1 poids de LSTM (weights)

Une cellule de mémoire a des paramètres de poids pour l'entrée, la sortie, ainsi qu'un état interne qui est construit par l'exposition aux pas de temps d'entrée.

- **Poids d'entrée** Utilisé pour pondérer l'entrée pour le pas de temps actuel.
- **Poids de sortie** Utilisé pour pondérer la sortie du dernier pas de temps.
- **État interne** État interne utilisé dans le calcul de la sortie pour ce pas de temps. [34].

2.11.2 portes de LSTM (Gates)

également des fonctions pondérées qui régissent davantage le flux d'informations dans la cellule. Il y a trois portes :

- **Forget Gate** décide des informations à supprimer de la cellule.
- **Input Gate** décide des valeurs de l'entrée pour mettre à jour l'état de la mémoire.
- **Output Gate** Décide ce qu'il faut sortir en fonction de l'entrée et de la mémoire de la cellule[34].

2.11.3 Structure du modèle LSTM

Le modèle consiste séquentiellement en une couche d'entrée, suivie d'une couche LSTM, d'une couche dropout et de la couche finale, la couche dense[31].

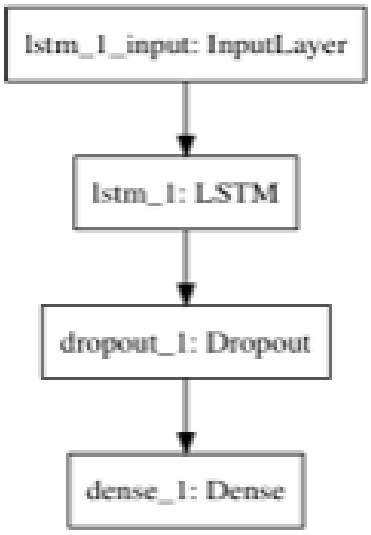

FIGURE 2.21 – La structure du modèle LSTM.

— **Couche lstm** Les couches Keras LSTM attendent une entrée sous la forme d'un tableau numérique de 3 dimensions (samples, time steps, features) où les samples sont le nombre de séquences d'entraînement, les time steps est la fenêtre de retour en arrière ou la longueur de la séquence et les features est le nombre d'entités de chaque séquence à chaque pas de temps.
— **Dropout** Généralement, la formation des réseaux de neurones prend plus de temps si bien que le sur ajustement (overfitting) devient un problème sérieux avec l'augmentation du nombre de couches réseau. Le sur ajustement (overfitting) donne au modèle de prédiction d'excellentes performances dans les données d'entraînement, mais pas dans les données de test. Pour surmonter ce problème, dropout est adopté dans le LSTM pour empêcher la capture répétée des mêmes fonctionnalités (features)[35].
— **Dense** Une couche entièrement connectée qui suit souvent les couches LSTM et est utilisée pour générer une prédiction est appelée Dense ().

2.11.4 Les modèles du LSTM

Les différentes architectures de LSTM sont :

2.11.4.1 Vanilla LSTM

Une configuration LSTM simple est le Vanilla LSTM. qui a une seule couche cachée d'unités LSTM. l'architecture qui donnera de bons résultats sur la plupart des petits problèmes de prédiction de séquence. Le Vanilla LSTM est défini comme :

- **Couche d'entrée.**
- **Couche cachée LSTM entièrement connectée.**
- **Couche de sortie entièrement connectée. [34]**

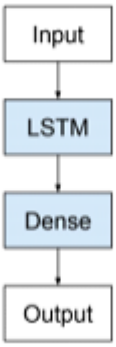

FIGURE 2.22 – Structure de Vanilla LSTM.

2.11.4.2 Stacked LSTM

est un modèle qui a plusieurs couches LSTM cachées où chaque couche contient plusieurs cellules de mémoire. Nous l'appellerons ici un LSTM empilé pour le différencier du LSTM non empilé (Vanilla LSTM) et d'une variété d'autres extensions du modèle LSTM de base[34].

2.11.4.3 CNN LSTM

L'architecture CNN LSTM implique l'utilisation de couches CNN (Convolutional Neural Network) pour l'extraction de caractéristiques sur les données d'entrée combinées avec des LSTM pour prendre en charge la prédiction de

FIGURE 2.23 – Structure de stacked LSTM .

séquence. Les LSTM CNN ont été développés pour les problèmes de prédiction de séries chronologiques visuelles et l'application de la génération de descriptions textuelles à partir de séquences d'images (par exemple des vidéos)[34].

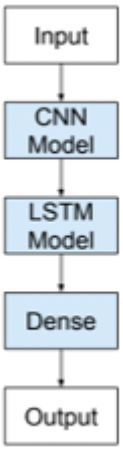

FIGURE 2.24 – Structure de CNN LSTM.

— Structure CNN LSTM : Le modèle cnn-lstm consiste séquentiellement des couches convolutionnelle 1D chaque couche suivie d'une couche de max pooling, la sortie est ensuite aplatie pour alimenter les couches LSTM (où la

couche Flatten relier entre cnn et lstm). Les couches LSTM cachées suivies d'une couche dense pour fournir la sortie [34].

2.11.4.4 Encodeur-décodeur

Une approche des problèmes de prédiction seq2seq qui s'est avérée très efficace est appelée Encoder-Decoder LSTM. Cette architecture est composée de deux modèles : un pour lire la séquence d'entrée et la coder dans un vecteur de longueur fixe, et un second pour décoder le vecteur de longueur fixe et sortir la séquence prédite. L'utilisation des modèles de concert donne à l'architecture son nom d'encodeur-décodeur LSTM conçu spécifiquement pour les problèmes seq2seq[34].

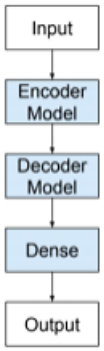

FIGURE 2.25 – Structure de encoder-decoder .

2.11.4.5 LSTM bidirectionnel

Nous avons vu l'intérêt d'inverser l'ordre des séquences d'entrée pour les LSTM discuté dans l'introduction des LSTM Encoder-Decoder. Les LSTM bidirectionnels se concentrent sur le problème de tirer le meilleur parti de la séquence d'entrée en parcourant les pas de temps d'entrée dans les directions avant et arrière. En pratique, cette architecture consiste à dupliquer la première couche récurrente dans le réseau afin qu'il y ait maintenant deux couches côte à côte, puis à fournir la séquence d'entrée telle quelle en entrée de la première couche et à fournir une copie inversée de la séquence d'entrée. à la seconde. Cette approche a été développée il y a quelque temps comme approche générale pour améliorer les performances des réseaux neuronaux récurrents (RNN) [34].

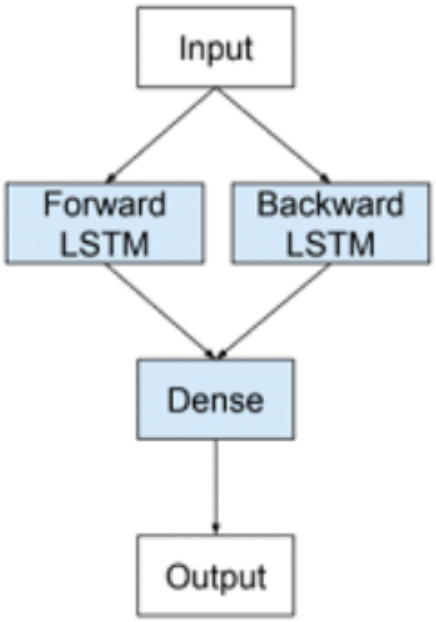

FIGURE 2.26 – Structure de LSTM bidirectionnel .

2.11.4.6 LSTM générative

Un LSTM génératif n'est pas vraiment une architecture, c'est plutôt un changement de perspective sur ce qu'un modèle prédictif LSTM apprend et comment le modèle est utilisé. On pourrait imaginer utiliser n'importe quelle architecture LSTM comme modèle génératif. Dans ce cas, nous utiliserons un simple LSTM Vanilla[34].

2.12 Conclusion

Dans ce chapitre, nous avons parlé de différents types d'apprentissage automatique, de réseaux de neurones et d'apprentissage en profondeur sous ses différentes formes et modèles. Dans le chapitre suivant, nous discuterons de l'optimisation des combinaisons.

Chapitre 3

OPTIMISATION

3.1 Introduction

L'optimisation est une branche des mathématiques qui cherche à modéliser, analyser et résoudre des problèmes analytiques ou numériques qui consistent à réduire ou à augmenter une fonction dans un ensemble. Dans ce chapitre, nous parlerons d'optimisation, d'optimisation compinatoire, de méthodes d'optimisation (Exact et approche), et nous avons aborderons en détail l'approche des algorithmes génétiques.

3.2 Un problème

Un problème dans le point de vue informatique (Computational Problem) veut dire comment faire relier un ensemble de données par un ensemble de résultats, où les données vont subir un ensemble d'opérations dont on les appelle le traitement. Donc il est conçu comme une relation $\pi \subseteq 1xS$ entre les entrées ou instances, et les sorties ou solutions. Pour qu'une instance d'un problème soit compréhensible par un ordinateur numérique, elle doit être décrite comme une séquence finie de symboles d'un ensemble fini arbitraire appelé alphabet. De même, la solution d'une instance d'un tel problème est émise dans ce format. Généralement, l est infini alors que S peut être fini [36].

3.3 Théorie de la Complexité des Algorithmes

La théorie de la complexité des algorithmes a donné un sens précis au terme d'algorithme efficace et de problème difficile, Alors évaluer la complexité d'un algorithme consiste à trouver en fonction de la taille des données, le nombre d'opérations élémentaires nécessite par cet algorithme. Alors on dit qu'un algorithme est efficace si le nombre des opérations nécessaires pour résoudre un problème est donné par une fonction polynomiale d'un paramètre caractérisant la taille du problème considéré [37].

3.4 Les problèmes peuvent être classés selon les propriétés de l'ensemble des solutions

Les problèmes peuvent être classés selon les propriétés de l'ensemble des solutions :

• **Un problème de décision**c'est un problème dont la réponse est tout simplement OUI ou NON.

• **Un problème polynomial réductible** on a L1 et L2 deux problèmes de décision, on dit L1 est polynomial réductible à L2 s'il existe un algorithme polynomial qui convertit chaque instance d'entrée de L1 à une autre instance de L2.

• **Un problème de la classe P** un problème est dit polynomial s'il existe un algorithme de complexité polynomiale permettant de répondre à la question posée dans ce problème, quel que soit la donnée de celui-ci. La classe P est l'ensemble de tous les problèmes de reconnaissances polynomiaux.

• **Un problème de la classe NP** un problème de décision est dans la classe NP si, pour toute instance de ce problème, on peut vérifier, en un temps polynomial par rapport à la taille de l'instance, qu'une solution proposée ou devinée permet d'affirmer que la présence est oui pour cette instance[38].

3.5 Un problème combinatoire

Un problème combinatoire est toute situation dont on cherche d'avoir une solution tout en respectant la présence d'un ensemble de contraintes. La solution c'est un résultat de faire combiner ensemble des éléments de problème d'une manière de maximise ou minimise un fonction objectif. Par

exemple, le schéma suivant présente une situation de problème combinatoire : où on veut acheter une voiture dans la mode et en même temps avec un prix raisonnable qui ne peut pas dépasser certaine limite. Si on maximise la première contrainte (une bonne voiture) on va avoir un prix maximale, dans le contraire on va aboutir à une mauvaise voiture mais avec un prix minimale dans les limites ; on constate dans cet exemple que c'est difficile d'arranger ces deux contraintes dans nos besoins.

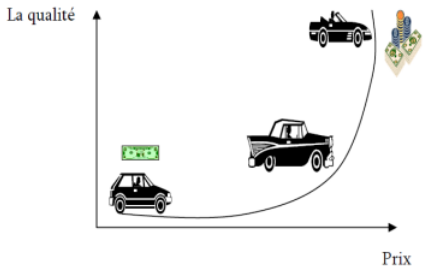

FIGURE 3.1 – illustrant un problème combinatoire.

3.6 Un problème d'optimisation

3.6.1 Définition

Les problèmes d'optimisation ont été définis pour représenter les problèmes décrits dans le paragraphe précédent. Ces problèmes occupent actuellement une place de choix dans la communauté scientifique. Non pas qu'ils aient été un jour considérés comme secondaires mais l'évolution des techniques informatiques a permis de dynamiser les recherches dans ce domaine. Le monde réel offre un ensemble très divers de problèmes d'optimisation :

- Problème combinatoire ou à variables continues.
- Problèmes à un ou plusieurs objectifs.
- Problèmes statistiques ou dynamiques.
- Problème dans l'incertain[39].

Un tel problème est caractérisé par :

- Un problème d'optimisation est défini par **un espace d'état**, **une ou plusieurs fonction(s) objective(s)** et **un ensemble de contraintes.**

— **L'espace d'état** est défini par l'ensemble de domaines de définition des variables du problème.
— **Les variables** du problème peuvent être de nature diverse (réelle, entière, booléenne, etc.) et sont exprimées de données qualitatives ou quantitatives.
— **Une fonction objective** représente le but à atteindre pour le décideur (minimisation de coût, de durée, d'erreur, ...). Elle définit un espace de solutions potentielles au problème.
— **L'ensemble de contraintes** définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité et permettent en général de limiter l'espace de recherche.

3.6.2 Fonction objective

C'est le nom donné à la fonction f (on l'appelle encore fonction de coût ou critère d'optimisation). C'est cette fonction que l'algorithme d'optimisation va devoir (optimiser) (trouver un optimum).

3.6.3 Variables de décision :

Elles sont regroupées dans le vecteur $\vec{x}$. C'est en faisant varier ce vecteur que l'on cherche un optimum de la fonction f.

3.6.4 Minimum global :

Un (point) $\vec{x}^*$ est un minimum global de la fonction f si on a : $f(\vec{x}^*) < f(\vec{x})$ Quel que soit $\vec{x}^* \neq \vec{x}$
Exemple : M3 dans la figure 3 [40].

3.7 Classification des problèmes d'optimisation

On peut classer les différents problèmes d'optimisation que l'on rencontre dans la vie courante en fonction de leurs caractéristiques :

— **Le domaine des variables de décision** soit Continu et on parle alors de **problème continu**, soit discret et on parle donc de **problème combinatoire** ;

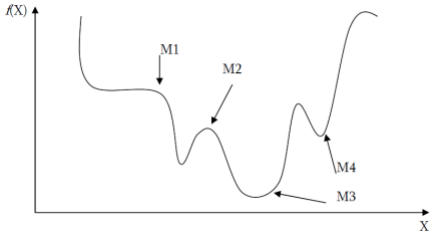

FIGURE 3.2 – différents Minima.

— **La nature de la fonction objective** soit linéaire et on parle alors de **problème linéaire**, soit non linéaire et on parle donc de **problème non linéaire** ;
— **Le nombre de fonctions objectifs à optimiser**soit une fonction scalaire et on parle alors de **problème mono-objectif**, soit une fonction vectorielle et on parle donc de **problème multiobjectif** ;
— **La présence ou non des contraintes** on parle de **problème sans contrainte** ou **avec contrainte**
— **L'environnement problème dynamique** (la fonction objectif change dans le temps).

3.8 Les problèmes d'optimisation mono-objectifs

Lorsqu'un seul objectif (critère) est donné, le problème d'optimisation est monoobjectif. Dans ce cas la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal, maximal). De manière formelle, à chaque instance d'un tel problème est associé un ensemble W des solutions potentielles respectant certaines contraintes et une fonction d'objectif f : $W \rightarrow Y$qui associe à chaque solution admissible $s \in W$ une valeur $f(s)$. Résoudre l'instance (W, f) du problème d'optimisation consiste à trouver la solution optimale $s^* \in X$ qui optimise (minimise ou maximise) la valeur de la fonction objectif f.

3.9 Méthodes optimisation Combinatoire

3.9.1 Définition

L'optimisation combinatoire recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction avec ou sans contraintes. En théorie, un problème d'optimisation combinatoire est défini par un ensemble d'instances. A chaque instance du problème est associé un ensemble discret de solutions S, un sous ensemble X de S représentant les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution $s \in X$ le nombre réel (ou entier) $f(s)$. Résoudre un tel problème (plus précisément une telle instance du problème) consiste à trouver une solution $s^* \in X$ optimisant la valeur de la fonction de coût f. Une telle solution S^* s'appelle une solution optimale ou un optimum global[41].

3.9.2 Classification des méthodes d'optimisation combinatoire

La résolution d'un problème d'optimisation combinatoire est réalisée à l'aide des méthodes illustrées dans la figure suivante :

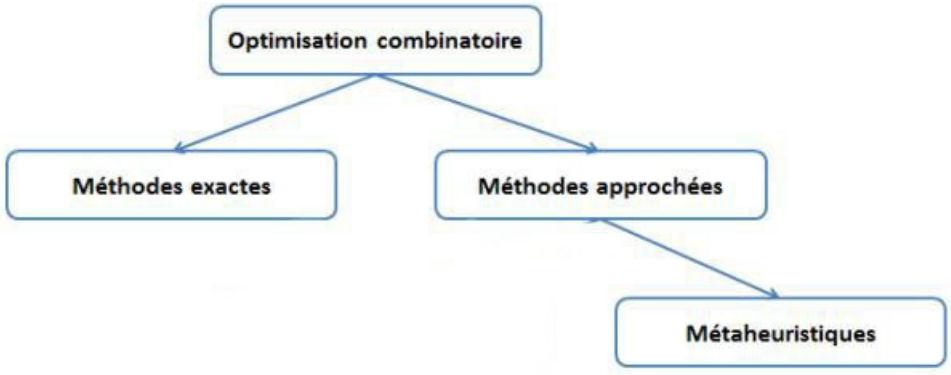

FIGURE 3.3 – Classification des méthodes d'optimisation.

3.9.2.1 Méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles les techniques de séparation et évaluation (Branch and Bound) ou la programmation dynamique. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable[42].

3.9.2.2 Méthodes approchées

Lorsque l'on dispose d'un temps de calcul limité ou lorsqu'on est confronté à des problèmes difficiles ou de taille importante, on peut avoir recours aux méthodes approchées, en se contentant de rechercher une solution de bonne qualité. Dans ce cas le choix est parfois possible entre une heuristique spécialisée et une métaheuristique [38]

- **Heuristique** Le mot heuristique vient du grec eurisko qui signifie je trouve d'où la célèbre Eureka d'Archimède. Une heuristique, ou méthode approximative, est un algorithme qui fournit rapidement(en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation difficile. Une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre.
- **Métaheuristique** Le mot métaheuristique est dérivé de la composition de deux mots grecs : méta signifiant au-delà et heuristique. En effet, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changement profond dans l'algorithme employé.

3.10 Métaheuristiques

3.10.1 Définition

Contrairement aux méthodes exactes, les métaheuristiques permettent de s'attaquer aux instances problématiques de grande taille en fournissant des solutions satisfaisantes dans un délai raisonnable. Il n'y a aucune garantie de trouver des solutions globales optimales ou même des solutions limitées. Les métaheuristiques ont reçu de plus en plus de popularité au cours des 20 dernières années. Leur utilisation dans de nombreuses applications montre leur efficacité pour résoudre des problèmes vastes et complexes. Parmi ces domaines, on peut citer les suivants : [43]

- Conception technique, optimisation de la topologie et optimisation structurelle en électronique et VLSI, aérodynamique, dynamique des fluides, télécommunications, automobile, et la robotique.
- Apprentissage automatique et fouille de données en bioinformatique et biologie computationnelle, et les finances.
- Simulation et identification en chimie, physique et biologie, traitement du signal et de l'image...etc.
- Planification des problèmes de routage, planification des robots, problèmes d'ordonnancement et de production, logistique et transport, gestion de la chaîne d'approvisionnement...etc.

3.10.2 Concepts fondamentaux des métaheuristiques

Les métaheuristiques ne nécessitent pas une connaissance particulière sur les problèmes d'optimisation à résoudre. Il suffit d'associer une ou plusieurs variables à une ou plusieurs solutions (optimum). Il existe deux points critiques pour toute métaheuristique :[44]

3.10.2.1 Diversification

Le principe de diversification d'une méthode d'optimisation donnée correspond à sa capacité de parcourir aisément l'espace de recherche pour obtenir des solutions très différentes les unes des autres. Autrement dit, c'est une mécanisme pour une exploration assez large de l'espace de recherche.

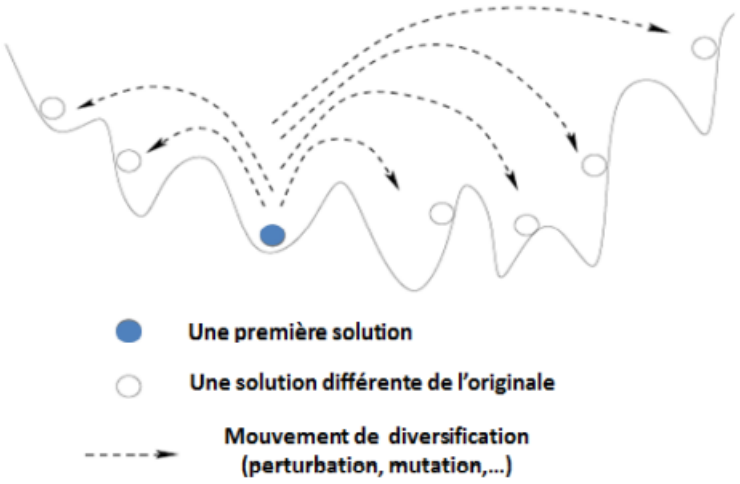

FIGURE 3.4 – Processus de diversification d'une solution.

3.10.2.2 Intensification

Autant le principe de diversification essaye de déplacer les solutions dans d'autres zones de l'espace de recherche, autant le processus d'intensification vise à forcer une solution donnée à tendre vers l'optimum local de la zone à laquelle elle est attachée. En effet, Elle permet une exploitation de l'information accumulée durant la recherche.

3.10.3 Classification des métaheuristiques :

Une manière de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois(méthodes de trajectoire). Voici une figure qui montre cette classification avec des exemples typiques pour chaque catégorie :

3.10.3.1 Métaheuristiques à solution unique

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [45].

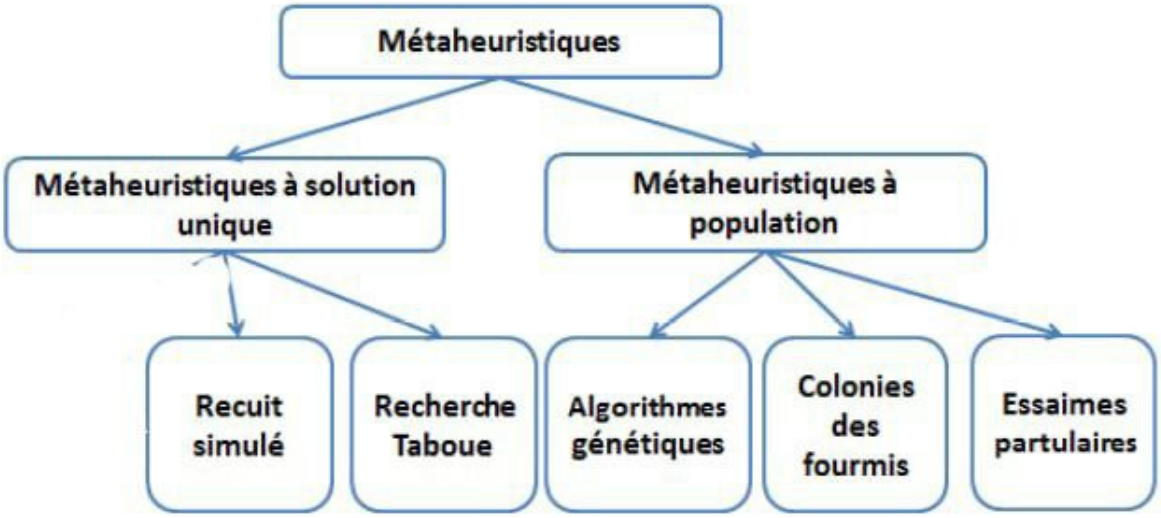

FIGURE 3.5 – Classification des métaheuristiques .

Il existe plusieurs méthodes pour cette classification, les plus connues sont :

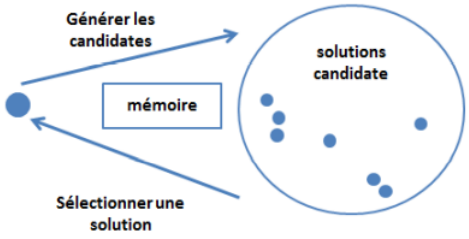

FIGURE 3.6 – Principe des métaheuristiques à solution unique .

3.10.3.1.1 Le recuit simulé

Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralisé à laquelle on introduit un paramètre de température qui sera ajusté pendant la recherche. Elle s'inspire des méthodes de simulation de Metropolis (années 50) en mécanique statistique. L'analogie historique s'inspire

du recuit des métaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global.[46] .

3.10.3.1.2 La recherche tabou

La recherche Tabou a été introduite par F. Glover et a montré sa performance sur de nombreux problèmes d'optimisation. Le principe de l'algorithme est le suivant : à chaque itération, le voisinage (complet ou sous ¬ensemble de voisinage) de la solution courante est examiné et la meilleure solution est sélectionnée. En appliquant ce principe, la méthode autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut-être un meilleur voisinage.[45] .

3.10.3.2 Métaheuristique à population

Dans cette classe les métaheuristiques utilisent la notion de population : elles manipulent toutes un échantillonnage de la fonction objectif, via des processus communs. Autrement dit, les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

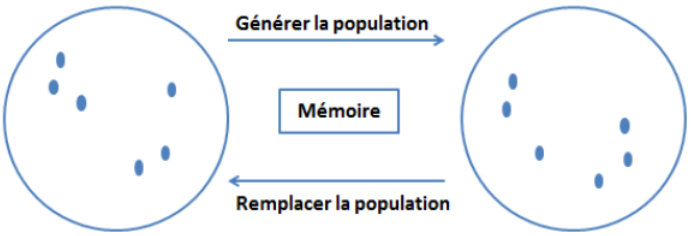

FIGURE 3.7 – Principe des métaheuristiques à population .

Parmi les algorithmes inclus dans cette classification on peut citer les suivants :

3.10.3.2.1 Les colonies des fourmis

L'algorithme de colonies de fourmis a été à l'origine principalement utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. On observe, depuis ses débuts, que son emploi se généralise à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification, ou encore le traitement d'image[47] .

3.10.3.2.2 Les algorithmes à essaim de particules

Les algorithmes d'optimisation par essaim de particules (PSO) ont été introduit en 1995 par Kennedy et Eberhart comme une alternative aux algorithmes génétiques standards. Ces algorithmes sont inspirés des essaims d'insectes (ou des bancs de poissons ou des nuées d'oiseaux) et de leurs mouvements coordonnés. En effet, tout comme ces animaux se déplacent en groupe pour trouver de la nourriture ou éviter les prédateurs, les algorithmes à essaim de particules recherchent des solutions pour un problème d'optimisation. Les individus de l'algorithme sont appelés particules et la population est appelée essaim[45]

3.10.3.2.3 Les algorithme génétiques

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, sélection, etc. Introduits par J.H. Holland au début des années 1970. Ils sont appliqués dans diverses domaines : l'économie, l'optimisation de fonctions, la finance, en théorie du contrôle optimal, théorie des jeux répétés et différentiels.

3.10.3.2.4 Les composants d'algorithme génétique

3.10.3.2.5 Codage de données

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée).

3.10.3.2.6 Génération de la population initiale

C'est lors de l'initialisation de l'algorithme génétique que les individus originaux sont générés. Un chromosome est une suite de gêne, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position. Chaque individu est représenté par un seul chromosomes, et une population est un ensemble d'individus [46].

3.10.3.2.7 sélection

La sélection permet d'identifier les individus susceptibles d'être croisés dans une population. Il existe plusieurs techniques de sélection. Nous présentons ici les quatre les plus utilisées parmi elles[46] :

- **Sélection par rang** consiste à ranger les individus de la population dans un ordre croissant ou décroissant, selon l'objectif (fonction fitness).
- **Sélection par roulette** elle consiste à créer une roue de loterie biaisée pour laquelle chaque individu de la population occupe une sélection de la roue proportionnelle à sa valeur d'évaluation.
- **sélection aléatoire**cette sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité uniforme d'être sélectionné, où est le nombre total d'individus dans la population.
- **sélection par tournoi** le tournoi le plus simple consiste à choisir aléatoirement un nombre k d'individus dans la population et à sélectionner celui qui a la meilleure performance. Les individus qui participent à un tournoi sont remis ou sont retirés de la population, selon le choix de l'utilisateur. Avec le tournoi binaire, sur deux individus en compétition, le meilleur gagne avec une probabilité ,

3.10.3.2.8 Croisement

L'opérateur de croisement combine les caractéristiques d'un ensemble d'individus parents (généralement deux) préalablement sélectionnés, et génère de nouveaux individus enfants. Là encore, il existe de nombreux opérateurs de croisement [47].

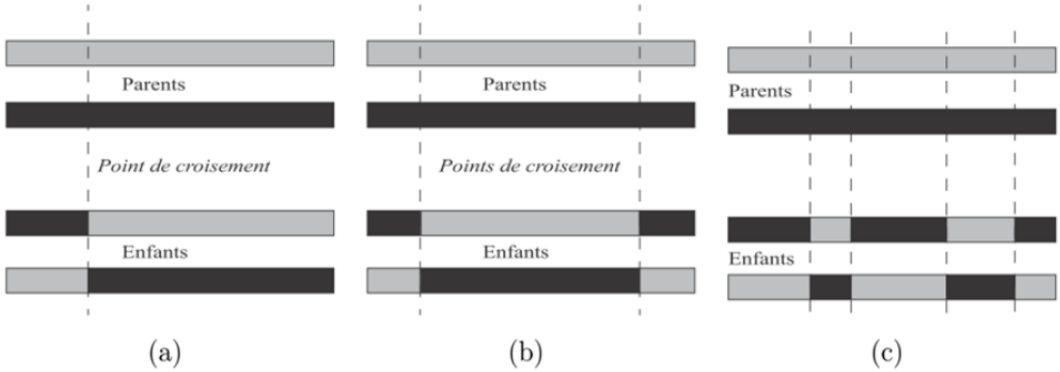

FIGURE 3.8 – Croisement : (a) croisement simple en un point, (b) croisement en deux points, (c) croisement uniforme.

3.10.3.2.9 Mutation

De façon aléatoire, un gène peut, au sein d'un chromosome être substitué à un autre. De la même manière que pour les enjambements, on définit ici un taux de mutation lors des changements de population qui est généralement compris entre 0 et 1 . Il est nécessaire de choisir pour ce taux une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de sélection et d'évolution. La mutation sert à éviter une convergence prématurée de l'algorithme. Par exemple lors d'une recherche d'extremum la mutation sert à éviter la convergence vers un extremum local.

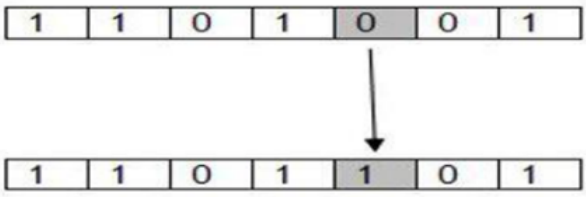

FIGURE 3.9 – une mutation dans le cas d'un codage binaire.

L'efficacité des algorithmes génétiques dépend fortement du réglage des différents paramètres caractérisant ces algorithmes, et qui sont parfois difficiles à déterminer. Des paramètres comme la taille de la population, le nombre maximal des générations, la probabilité de mutation pm, et la probabilité de croisement pc. Les deux premiers paramètres dépendent directement de la

nature du problème et de sa complexité, et leurs choix doit représenter un compromis entre la qualité des solutions et le temps d'exécution. La probabilité de croisement pc est liée à la forme de la fonction d'évaluation. Son choix est en général heuristique. Plus sa valeur est élevée, plus la population subit des changements importants. La probabilité de mutation pm est généralement faible puisqu'un taux élevé risque de conduire vers un recherche aléatoire [48].

3.10.3.2.10 Algorithme

Algorithme : Algorithme génétique
1. P = Créer une population initiale () ;
2. Tantque le critère d'arrêt n'est pas satisfait faire
3. P' = Selection(P) ;
4. E = Croisement(P') ;
5. E = Mutation(E) ;
6. P = Remplacement(E, P) ;
7. FinTantque
8. Retourner la meilleure solution trouvée ;

TABLE 3.1 – Algorithme génétique

3.11 Revue de quelques travaux existants de RUL à base de l'optimisation et le deep learning

Dans toute application RUL, la précision est une priorité absolue pour éviter les accidents et les pertes énormes. Même si les algorithmes et les modèles composites peuvent fournir des résultats précis, la cohérence est importante. Dans n'importe quel cycle de temps unitaire particulier, il peut y avoir divers résultats possibles en fonction même d'un léger changement dans les paramètres opérationnels et d'autres facteurs. La recherche sur l'optimisation s'est accrue pour couvrir ces nécessité. Toutes les architectures Deep Learning utilisent un certain optimiseur pour leur apprentissage, mais ceux-ci

seuls peuvent ne pas suffire. La flexibilité et la compatibilité des méthodes de métaheuristique évolutionnaires et plus particulièrement des algorithmes génétiques avec des cadres d'apprentissage en profondeur, ont incité d'énormes quantités de recherche dans le domaine de prédiction de RUL.Leur souplesse d'application ouvre de nombreuses possibilités d'optimisation. Ils peuvent être utilisés en apprentissage profond pour optimiser les hyper-paramètres, les fonctions d'activation, les architectures de modèles, etc. [49], Les DBN sont combinés pour établir un modèle d'ensemble pour l'estimation RUL. Une méthode d'apprentissage d'ensemble évolutif multiobjectif pour faire évoluer plusieurs DBN simultanément est proposée. La précision et la diversité sont utilisés comme deux objectifs contradictoires. Les pondérations combinées sont optimisées via l'ED à objectif unique en utilisant l'erreur d'entraînement moyenne comme objectif. La méthode proposée a été évaluée sur l'ensemble de données de moteurs d'avions C-MAPSS de la NASA. Les chercheurs ont apprécié l'efficacité du LSTM dans la prédiction à long terme. Un GA optimise un modèle LSTM à deux couches basé sur la machine Boltzmann restreint a été proposé dans [50]. L'algorithme d'optimisation du taux d'apprentissage adaptatif Adam avec le modèle Vanilla LSTM à une seule couche a été appliqué [51]. De même, l'algorithme d'optimisation du taux d'apprentissage adaptatif Adam a été appliqué pour optimiser le modèle LSTM dans Wu et al. [52]. D'autres approches d'apprentissage en profondeur étaient l'unité récurrente fermée par auto-encodeur [53] et les réseaux de neurones à convolution profonde [9]. Dans [54], les poids pour combiner RNN et LSTM sont optimisés par l'algorithme non-dominated sorting genetic algorithm II (NSGA-II). [55] a utilisé les algorithmes génétiques pour optimiser le taux d'apprentissage et le batch size d'un modèle à base de LSTM pour un turboréacteur (turbofan engine).On s'est inspiré de ce dernier dans notre travail, en ajoutant d'autres hyperparamètres que nous voyons essentiels dans la précision de la performance d'un modèle LSTM.

3.12 Conclusion

A travers ce chapitre, nous avons abordé les définitions essentielles à la compréhension de travail du point de vue optimisation. Nous avons pu voir qu'un grand nombre de méthodes d'optimisation existe pour résoudre un problème combinatoire. Ces méthodes peuvent être exactes ou bien approchées où on distingue deux grandes familles : les heuristiques et les métaheu-

ristiques. Cette dernière catégorie aussi résout deux types de problèmes, à solution unique et à base de population de solution. Enfin, nous avons présenté en détail les algorithmes génétiques. Cette méthode sera utilisée dans ce travail pour optimiser les valeurs des hyper paramètres du modèle LSTM dans le but d'augmenter la performance . Le chapitre suivant détaille notre contribution.

Chapitre 4

CONTRIBUTION

4.1 Introduction

Dans ce chapitre, nous utiliserons l'apprentissage en profondeur pour résoudre le problème de l'estimation de RUL pour la maintenance basée sur l'état. En se basant sur les travaux connexes décrits au chapitre 3, nous tenterons dans ce chapitre d'améliorer la performance du modèle deep utilisé en se servant de l'algorithme génétique. En effet, dans un premier temps, nous décrirons la problématique et les motivations de notre contribution. Nous présenterons ensuite l'architecture du modèle proposé et les différentes étapes communes à l'utilisation des algorithmes de deep learning. Nous décrivons aussi les hyperparamètres à optimiser ainsi que les paramètres de l'algorithme génétique et les différentes étapes suivies. Ainsi nous présenterons l'algorithme de Gridsearch avec laquelle nous allons faire des comparaisons. À la fin, une analyse des résultats obtenus et une comparaison avec la méthode implémentée avec le grid search et les autres approches similaires dans la littérature sont effectuées.

4.2 Motivation

L'estimation de la durée de vie utile restante (RUL) et l'évaluation de la dégradation des performances des machines ont un rôle crucial dans la maintenance conditionnelle (CBM). Elles aident à réduire les coûts de maintenance, à améliorer la fiabilité et ainsi elles influencent la prise de décision dans le système. Le deep learning est un outil puissant qui montre des capa-

cités plus fortes dans le domaine de l'estimation de RUL, néanmoins dans la capture des relations non-linéaires complexes entre les entrées et les sorties. Les données décrivant les paramètres de la CBM sont de type séries chronologiques (time series). Un type puissant de réseau neuronal conçu pour gérer la dépendance de séquence à long terme est le LSTM et GRU. Le LSTM et GRU sont choisi dans le modèle proposé aussi pour leurs possibilités d'apprentissage des propriétés temporelles des données pour la prédiction de séries temporelles. Nous avons entamé l'amélioration des hyperparamètres d'un réseau de neurone, car le processus d'obtention d'une solution optimale est un sujet difficile, en raison de la grande taille de l'espace de solutions. Cela peut prendre des heures et peut-être des jours pour les rechercher manuellement sans parvenir à une solution appropriée. Et comme déjà mentionné, le domaine de solutions est grand, nous avons choisi une des méthodes approchées au lieu de méthodes exactes. Ces dernières sont valables dans le cas de petites solutions, qui appliquant dans le cas contraire peuvent être long et non efficaces. Le choix est effectué aussi sur une méthode métaheuristique au lieu d'utiliser une heuristique, car les métaheuristiques sont des algorithmes de haut niveau, capable de guider et d'orienter le processus de recherche dans un espace de solution (souvent très grand) afin de déterminer des solutions (presque) optimales (généralement de très bonne qualité). Le rapport entre le temps d'exécution et la qualité de la solution d'une métaheuristique est très intéressant par rapport aux différents types d'approches de résolution.

En fait, le problème d'optimisation d'hyperparamètres d'un modèle d'apprentissage automatique est, en général, un problème d'optimisation avec des variables mixtes discrètes/continues (les hyperparamètres peuvent prendre des valeurs entières, d'autres des valeurs réelles, et certains sont des chaînes). De plus, les gradients de performance du modèle (degré d'exactitude ou de précision, par exemple) par rapport aux hyperparamètres ne peuvent pas être facilement calculés. Enfin, comme nous l'avons vu plus haut, l'espace de recherche peut être très grand et complexe, si le nombre d'hyperparamètres est suffisamment élevé. Sur la base de ce raisonnement, il s'agit d'une implémentation idéale des algorithmes d'optimisation stochastiques tels que les algorithmes génétiques.

En effet, un certain nombre de paramètres définissant un modèle, appelés hyperparamètres, ne sont pas optimisés lors de la formation. Par exemple : taille de fenêtre, le nombre d'unités, taille des lots, taux d'apprentissage, etc. Le choix des valeurs des hyperparamètres affecte la qualité du modèle final, parfois fortement. En revanche, choisir manuellement les meilleures valeurs

d'hyperparamètres peut s'avérer compliqué, même pour les "data scientists" les plus expérimentés. Heureusement, ce processus peut être automatisé, au moins partiellement.

4.3 Architecture de Deep Learning proposée

4.3.1 Le modèle LSTM

Dans notre travail, nous utilisons le modèle LSTM avec une couche en entrée, une couche cachée et une couche de sortie :

- **La couche LSTM :** Nous ajoutons deux couches LSTM avec un nombre amélioré d'unités cachées pour chaque couche. La première couche LSTM renvoie des séquences à la deuxième couche LSTM qui, à son tour, renvoie une valeur unique pour chaque cellule.
- **La couche Dropout :** Afin d'éviter le problème de overfitting, nous avons eu recours à l'ajout d'une couche Dropout après chaque couche LSTM. Elles sont utilisées avec une probabilité de 0.2 chacune.
- **La couche Dense :** Nous ajoutons à la fin une couche entièrement connectée, qui mappe la sortie de LSTM à la taille de sortie désirée. Cette dernière est égale à 1 et prévue par la fonction d'activation tanh.

4.3.2 Le modèle GRU

Nous utilisons le modèle GRU avec une architecture similaire à celle de LSTM. Il a 3 couches - 2 couches GRU composées chacune d'un nombre de cellules GRU parallèles amélioré et suivies d'une seule couche de sortie Dense à l'échelle d'un neurone et prévue par la fonction d'activation tanh. La première couche GRU renvoie des séquences à la deuxième, qui, à son tour, renvoie une valeur unique pour chaque cellule. Une couche Dropout est utilisée pour éviter le problème d'overfitting. Elles sont utilisées avec une probabilité de 0.2 chacune.

Nous avons utilisé aussi l'optimizer adam lors de la compilation des deux modèles. Il est connu par son efficacité dans les pluparts des travaux du domaine utilisant le deep learning.

4.4 Les hyperparamètres à optimiser

— **La taille de fenêtre :** Lors de l'utilisation de modèles d'apprentissage en profondeur pour la prédiction RUL, une approche de fenêtre temporelle (Time window) est largement utilisée pour convertir le flux de données de surveillance des conditions en échantillons d'entrée pour l'apprentissage supervisé. Plus précisément, l'approche de la fenêtre temporelle prend les caractéristiques des pas de temps précédents comme variables d'entrée, tandis que RUL étiquette le pas de temps suivant comme variable de sortie [56]. Le paramètre fenêtre temporelle échelonnée est essentiel car il au régresseur de profiter non seulement de l'information précédente, mais aussi pour contrôler le rapport auquel l'algorithme est alimenté par de nouvelles informations.

— **Taille du lot (Batch size) :** Est un hyperparamètre qui définit le nombre d'échantillons à traiter avant de mettre à jour les paramètres du modèle interne.

— **Le nombre d'époques :** Définit le nombre de fois que l'algorithme d'apprentissage fonctionnera sur l'ensemble de données d'apprentissage.

— **Le taux d'apprentissage :** "Le learning rate" est un hyperparamètre important, il contrôle la quantité de mise à jour des poids en réponse au gradient estimé à la fin de chaque lot. Cela peut avoir un impact important sur le compromis entre la rapidité ou la qualité d'apprentissage du problème par le modèle.

— **Le nombre d'unités :** Le nombre d'unités cachées est l'un des hyperparamètres les plus mystérieux parmi tous. Les réseaux de neurones sont des approximateurs de fonctions universels et pour que le réseau apprenne à se rapprocher d'une fonction (ou d'une tâche de prédiction), il doit avoir suffisamment de "capacité" pour apprendre la fonction. Pour une fonction simple, il faudra peut-être moins d'unités cachées. Plus la fonction est complexe, plus le modèle aura besoin de capacité d'apprentissage. Un nombre légèrement supérieur d'unités au nombre optimal n'est pas un problème, mais un nombre beaucoup plus grand conduira à un surajustement du modèle, c'est-à-dire que si on fournit un modèle avec trop de capacité, il peut avoir tendance à sur-ajuster et simplement essayer de mémoriser l'ensemble de données[57].

4.5 Les étapes de travail

Dans notre travail, nous élaborons le modèle proposé basé sur le réseau de neurones LSTM et GRU afin de prédire la durée de vie utile restante (RUL) des moteurs d'avion. Le réseau utilise des valeurs de capteur d'avion simulées pour prédire quand un moteur d'avion tombera en panne dans le futur afin que la maintenance puisse être planifiée à l'avance. Nous pouvons définir le problème comme un problème de régression, car il est donné un nombre en sortie (le nombre de cycles d'un moteur en service durera avant de tomber en panne). Nous utiliserons les données C MAPSS dataset de la NASA, c'est une série chronologique multivariée. décrit dans la section suivante :

4.5.1 Dataset utilisé

Les données NASA C-MAPSS est un ensemble de données de référence largement utilisé, généré à l'aide des logiciels de programme de simulation basé sur un modèle propriétaire nommé C-MAPSS. Ce dernier est un logiciel qui simule les effets des défauts et de la détérioration à différentes conditions de fonctionnement dans les cinq principaux composants rotatifs (ventilateur, compresseur basse pression, Compresseur haute pression (HPC), turbine haute pression et basse Turbine à pression) trouvée dans un gros turboréacteur commercial. L'ensemble de données est divisé en quatre sous-ensembles de données (étiquetés à partir de FD001 à FD004) avec un nombre différent de conditions de fonctionnement et de modes de défaut[58]. Chaque sous-ensemble de données est ensuite divisé en sous-ensembles d'apprentissage et de test : Les ensembles de données sont organisés dans une matrice N par 26, où N correspond au nombre de points de données dans chaque ensemble de données. Chaque ligne est un instantané des données prises au cours d'un cycle de temps de fonctionnement unique, qui comprend 26 colonnes et chaque colonne représente une variable différente. Les 26 colonnes de données se composent de deux valeurs d'index représentant le numéro du moteur et le numéro du cycle opérationnel actuel, trois paramètres opérationnels qui ont un effet substantiel sur les performances du moteur, ainsi que 21 valeurs de capteur. Chaque trajectoire dans les ensembles de données simule la durée de vie d'un moteur. Alors que chaque moteur est simulé avec des conditions initiales différentes, l'état de fonctionnement de chaque moteur est sain au début et commence à se dégrader au fil du temps jusqu'à ce qu'une panne se produise. Pour chaque trajectoire du moteur dans

les ensembles d'entraînement, la dernière entrée de données correspond au moment où le moteur est déclaré malsain. Bien que les trajectoires dans les ensembles de test se terminent à un certain moment avant la défaillance et l'objectif est de prédire le nombre de cycles jusqu'à la fin de la durée de vie du produit pour chaque moteur communément appelé RUL. La valeur RUL réelle des trajectoires d'essai pour l'ensemble de données C-MAPSS a été rendue publique[58].

4.5.2 Conditions du fonctionnement et modes de défaut

Les points de données sont classés en différents clusters distincts en utilisant deux facteurs, à savoir les conditions de fonctionnement et les modes de défaut associés. Quand aux conditions de fonctionnement, FD001 et FD003 sont simulées en un seul point (niveau de la mer), tandis que FD002 et FD004 sont simulées à six conditions de fonctionnement différentes. Pour les modes de défaut, FD001 et FD002 sont simulés avec uniquement une dégradation HPC, tandis que FD003 et FD004 sont simulés avec deux dégradation HPC et dégradation du ventilateur, ce qui entraîne des prédictions RUL plus complexes et difficiles [58].

4.5.3 Phase de Prétraitement

Premièrement, on charge l'ensemble de données en tant que dataframe Pandas, qui est est une série chronologique multivariée. Chaque entrée (ligne) dans l'ensemble de données reflète un cycle de fonctionnement d'un moteur spécifique identifié par l'ID du moteur et la durée du cycle. Il y a plusieurs entrées par moteur pour représenter différentes heures de rapport. Les autres colonnes représentent différentes fonctionnalités, 3 paramètres opérationnels et 21 capteurs : L'ensemble de données CMAPSS est divisé en 4 ensembles chacun pour l'entraînement, le test et le RUL. Chaque sous-ensemble représente une condition opérationnelle différente et se compose d'un nombre différent de moteurs. Tous les moteurs sont supposés être du même type de modèle et fonctionnant normalement au début de chaque série. Au cours de sa série, il développe une faille. Le cycle est un nombre entier croissant de manière monotone et pour des raisons de modèle, il est supposé être équidistant et relatif pour chaque moteur. Les données sont ensuite divisées en un ensemble d'apprentissage et de test, chacun nécessitant une interprétation subtile lors de son traitement. Le dernier id, entrée de cycle, correspond

au moment où le moteur est déclaré défectueux. Par exemple, si le premier moteur a 192 événements de séries chronologiques distincts, le cycle ira de 1 à 192, tandis que le RUL commencera par 192 et descendra à 1. Pendant la préparation des données, on ajoute une colonne d'étiquette appelée 'rul'.

4.5.4 Phase de normalisation de données

Les données temporelles multi-variées de l'ensemble de données C-MAPSS contiennent des mesures d'unité moteur de 21 capteurs . Cependant, certaines lectures de capteur ont des sorties constantes pendant la durée de vie du moteur et elles ne fournissent pas d'informations précieuses pour l'estimation de la RUL. Pour chacun des 4 sous-ensembles de données dans C-MAPSS, les données de mesure collectées de chaque capteur sont normalisées pour être dans la plage de [-1, 1] en utilisant la méthode de normalisation MinMaxScaler de la bibliothèque scikit-learn [59].

$$x_{norm}^{i,j} = \frac{2 \times x^{i,j} - x_{min}^{i,j}}{x_{max}^{j} - x_{min}^{j}} - 1, \forall i, j \tag{4.1}$$

où $x^{j,j}$ désigne le i^{me} point de données d'origine du j^{me} capteur, et $x_{norm}^{j,j}$ est la valeur normalisée de $x^{j,j}$. x_{max}^{j} et x_{min}^{j} désignent respectivement les valeurs maximale et minimale des données de mesure originales du j^{me} capteur [59]. Cette normalisation garantira une contribution égale de toutes les fonctionnalités dans toutes les conditions de fonctionnement [58].

4.5.5 Les fonctions de métriques utilisées

Pour évaluer les performances de la méthode pronostique proposée, on utilise les fonctions de métriques suivantes : mean squared error (MSE),mean absolute error(MAE) et root mean squared error(RMSE).

4.6 Mean Squared Error (MSE)

Pour les tâches de prédiction numérique, la (Mean Squared Error)(MSE) est une méthode courante. MSE quantifie la différence entre un estimateur et la valeur ciblée. Le MSE d'un ensemble de données est la moyenne de la

somme de toutes les erreurs carrées de chaque modèle. Étant donné n modèles de l'ensemble de données, pour le ième exemple, soit pi la valeur prédite et ai la valeur réelle. Le MSE de l'ensemble de données testé est [56] :

$$MSE = \frac{\sum_{i=0}^{n}(a_i - p_i)^2}{n}$$

4.7 Root Mean Squared Error (RMSE)

Le RMSE est couramment utilisé comme mesure de la performance car il donne des poids égaux pour les prévisions précoces et tardives. La formulation du RMSE est donnée par[58] :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} h_i^2}$$

4.8 Mean Absolute Error (MAE)

La fonction mean-absolute-error calcule l'erreur absolue moyenne, une mesure de risque correspondant à la valeur attendue de la perte d'erreur absolue ou de la perte normale[60]. Si est la valeur prédite du -ème échantillon, et est la valeur vraie correspondante, alors l'erreur absolue moyenne (MAE) estimée sur est définie comme :

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples-1}} |y_i - \hat{y}_i|$$

4.9 L'algorithme génétique pour LSTM/GRU

L'algorithme génétique est utilisée pour trouver la taille de fenêtre, la taille de lots, le taux d'apprentissage et le nombre d'unités optimaux dans un réseau de neurones basé sur la mémoire à court terme (LSTM) et GRU dont le but est d'estimer le temps restant utile. Deux conditions préalables doivent être remplies pour utiliser un AG :

1. une représentation de la solution ou la définition d'un chromosome.

2. une fonction de fitness pour évaluer les solutions produites. Dans notre cas, un tableau binaire est une représentation génétique d'une solution (voir la figure 4.1). Il sera initialisé aléatoirement en utilisant **la distribution de Bernoulli**. Et **l'erreur racine moyenne carrée (RMSE)** du modèle sur l'ensemble de validation agira comme une valeur de fitness. La taille de chromosome (suite des bits) est basée sur la zone de couverture de la solution. Nous avons choisi 6 bits pour couvrir la taille de fenêtre (ws), 7 bits pour le nombre d'unités de la première couche,7 bits pour le nombre d'unités de la deuxième couche, 6 bits pour le nombres d'épochs, 8 bits pour le batch size et 7 bits pour le taux d'apprentissage. Le choix est justifier par la couverture large de ces hyperparamètres. Par ailleurs, les trois opérations de base qui constituent un AG, sont les suivantes :
   Sélection : La selection à roulette, la méthode la plus utilisée est choisie dans notre cas. Son principe est de sélectionner k individus parmi les individus d'entrée en utilisant k tours de roulette. La sélection se fait en ne regardant que le premier objectif de chaque individu. La liste retournée contient des références aux individus d'entrée.
   Croisement(Crossover) : "ordered crossover" Exécute un croisement ordonné (OX) sur les individus d'entrée. Les deux individus sont modifiés en place. Ce croisement attend des individus de séquence d'indices, le résultat pour tout autre type d'individus est imprévisible.
   De plus, ce croisement génère des trous dans les individus d'entrée. Un trou est créé lorsqu'un attribut d'un individu se trouve entre les deux points de croisement de l'autre individu. Ensuite, il fait pivoter l'élément de sorte que tous les trous se trouvent entre les points de croisement et les remplit avec les éléments supprimés dans l'ordre.
   Mutation : Afin d'introduire de la diversité et de la nouveauté dans le pool de solutions en échangeant ou en désactivant de manière aléatoire des bits de solution.
   on a choisi la "shuffle mutation" pour mélanger les attributs de l'individu d'entrée et renvoyer le mutant. On s'attend à ce que l'individu soit une séquence. L'argument indpb est la probabilité que chaque attribut soit déplacé. D'habitude cette mutation est appliquée sur le vecteur des indices.

   Les solutions initiales (populations) sont générées aléatoirement. En-

suite, elles sont évaluées en fonction de la fonction de fitness et de la sélection, puis un croisement et une mutation sont effectués. Ce processus est répété pour un nombre spécifié d'itérations. Au final, la solution avec le score de fitness le plus élevé est choisie comme la meilleure solution. Utiliser une technologie appelée « élitisme », qui préserve certaines des meilleures solutions dans la population et les transmet à la génération suivante.

FIGURE 4.1 – représentation génétique d'une solution .

Le tableau ci-dessous 4.1 décrit l'algorithme :

Algorithme : AG pour LSTM/GRU optimisé
1. Diviser les données en données d'entraînement et données de test ; 2. Les données de training sont utilisées pour évaluer le LSTM et/ou GRU. 3. Initialiser la longueur du chromosome (type binaire), la taille de la population (4) et le nombre de générations (4) ; 4. Définir le RMSE comme fonction de fitness ; 5. tant que it == nombre de générations 6. Utiliser la taille de fenêtre d'entrée générée pour le prétraitement de données de train ; 7. Utiliser le nombre d'unités cachées, le nombre d'époques , le batchSize et le learning rate générés pour dérouler LSTM/GRU 8. Croisement de chromosomes avec probabilité 0,7 ; 9. Mutation d'un nouveau chromosome avec probabilité 0,05 ; 10. Évaluer l'aptitude du chromosome nouvellement généré ; 11. Fin tant que 12. Sélectionner le meilleur chromosome individuel, qui est la taille de la fenêtre d'entrée optimisée, le nombre d'unités cachées dans chaque couche LSTM/GRU, le nombre d'époques, le nombre de batchSize et le nombre de taux d'apprentissage ; 13. Utiliser la taille de fenêtre d'entrée optimale pour le prétraitement de données,le nombre optimal d'unités cachées, le nombre d'époques optimale,optimal batchSize et le learning rate optimal pour prédire les données invisibles/données de test ;

TABLE 4.1 – Algorithme génétique pour LSTM/GRU optimisé

Les solutions initiales (populations) sont générées aléatoirement. Ensuite, elles sont évaluées en fonction de la fonction de fitness et de la sélection, puis un croisement et une mutation sont effectués. Ce processus est répété pendant un nombre spécifié d'itérations (appelées générations dans la terminologie GA). Au final, la solution avec le score de fitness le plus élevé est choisie comme l'optimal et sera utilisé pour exécuter une autre fois les deux modèles afin d'obtenir les meilleurs résultats .

4.10 Grid Search

La méthode traditionnelle d'optimisation des hyperparamètres est une recherche par grille, qui fait simplement une recherche complète sur un sous-ensemble donné de l'espace des hyperparamètres de l'algorithme d'apprentissage. Étant donné que le paramètre d'algorithme d'apprentissage automatique espace peut inclure des espaces avec des valeurs réelles ou illimitées pour certains paramètres, il est possible que nous devons spécifier une limite pour appliquer une recherche par grille. La recherche de grille souffre des espaces dimensionnels, mais peuvent souvent être facilement parallélisés, puisque les valeurs d'hyperparamètres avec lesquels l'algorithme fonctionne sont généralement indépendants les uns des autres[61].
Les hyperparamètres dans la méthode de grid search sont appliqués avec les valeurs suivants :

Hyperparamètres	Valeurs
Learning_rate	[0.01,0.001,0.0001,0.00001]
Nombre d'unités 1	[50,75,100,150]
Nombre d'unités 2	[30,40,50,100]
Taille de fenêtre	[40,50,60,70]
Batch_size	[100,150,200]
Nombre d'époques	[10,20,30]

TABLE 4.2 – Les valeurs des hyperparamètres utilisées

Le tableau ci-dessous4.3 décrit l'algorithme :

Algorithme : GridSearch pour LSTM optimisé
1. Diviser les données en données d'entraînement et de test ; 2. Les données de training sont utilisées pour évaluer le modèle LSTM 3. Définir le RMSE comme fonction de fitness ; 4. Définir une grille pour chaque hyperparamètre suivant : Le nombre d'unités cachées pour les deux couches du LSTM, le nombre d'époques, le batchSize et le learning rate. 5. Pour toutes les combinaisons possible des valeurs de la grille : 6. Utiliser la taille de fenêtre pour le prétraitement de données 7. Utiliser les autres hyperparamètres de la grille pour dérouler le modèle LSTM 8. Évaluer le score d'erreur pour chaque combinaison dans la grille d'hyperparamètres. 9. Sélectionner la combinaison d'hyperparamètres avec la meilleure métrique d'erreur. 10. Fin de la boucle. 11. Utiliser la taille de fenêtre d'entrée optimale pour le prétraitement de données, le nombre optimal d'unités cachées, le nombre d'époques optimal,optimal batchSize et le learning rate optimal pour prédire les données de test ; 12. Identifier les hyperparamètres du modèle à optimiser, puis sélectionner les valeurs d'hyperparamètres à tester.

TABLE 4.3 – Algorithme de gridSearch pour LSTM optimisé

4.11 Les résultats finals

Les résultats finaux ont été obtenus après l'évaluation du modèle LSTM simple proposé sur l'ensemble de données (FD001,FD002,FD003,FD004) après un nombre d'époques à optimiser avec l'utilisation des paramètres que nous avons choisi au dessus ce qui rend le modèle plus performant. Nous avons testé le modèle avec trois fonctions de métrique MSE, RMSE et MAE, Afin de comparer et savoir quelle métrique correspond à quel modèle. Pendant la recherche, nous avons trouvé que RMSE est la métrique la plus utilisée pour les tâches de régression et correspond à la racine carrée de la différence quadratique moyenne entre la valeur cible et la valeur prédite par le modèle. Il est préférable dans certains cas car les erreurs sont d'abord mises au carré avant le calcul de la moyenne, ce qui entraîne une forte pénalité sur les erreurs importantes. Cela implique que RMSE est utile lorsque des erreurs importantes ne sont pas souhaitées[62]. Et MAE est plus robuste aux valeurs aberrantes et ne pénalise pas les erreurs aussi fortement que MSE. Et que MSE est l'une des mesures les plus préférées pour les tâches de régression. En corrigeant les différences, cela pénalise même une petite erreur qui conduit à une surestimation de la gravité du modèle. Il est préférable aux autres métriques car il est différent et peut donc être mieux optimisé[62]. et nos résultats Prouvent cette affirmation en utilisant l'algorithme génétique et le Grid Search dans le tableau .

	metriques	FD001	FD002	FD003	FD004
Lstm avec GA	RMSE	0.06	0.10	0.04	0.09
	MAE	0.05	0.08	0.03	0.07
	MSE	0.004	0.011	0.002	0.008
Lstm avec Grid Search	RMSE	0.08	0.07	0.07	0.08
	MAE	0.06	0.05	0.06	0.07
	MSE	0.008	0.006	0.007	0.009

TABLE 4.4 – Comparaison entre les résultats de l'algorithme génétique et le Grid Search

Les "Best parameters" trouvés avec l'application de grid search sont : [1e-05, 150, 100, 70, 200, 30] Les "Best parameters" dans l'ensemble de don-

nées avec l'utilisation de l'algorithme génétique se trouvent dans le tableau suivant :

	FD001	FD002	FD003	FD004	FULLData
Window Size	45	57	7	45	63
Num of Units1	14	10	80	87	102
Num of Units2	103	59	48	10	26
Num of epoch	29	19	16	17	28
learning_rate	0.001	0.015	0.008	0.002	0.010
Batch_size	52	57	16	60	27

TABLE 4.5 – Les valeurs de best paramètres avec GA

Une autre comparaison entre le modèle Lstm et GRU dans l'ensemble de données (FD001,FD002,F003,FD004) avec l'utilisation de l'algorithme génétique et le résultat se trouve dans le tableau suivant :

	metriques	FD001	FD002	FD003	FD004
Optimized LSTM proposé	RMSE	0.06	0.10	0.04	0.09
	MAE	0.05	0.08	0.03	0.07
	MSE	0.004	0.011	0.002	0.008
Optimized Gru proposé	RMSE	0.08	0.11	0.04	0.09
	MAE	0.07	0.09	0.03	0.08
	MSE	0.007	0.013	0.002	0.009

TABLE 4.6 – Comparaison entre les résultats de modèle LSTM et GRU

4.12 Les plots des résultats finals

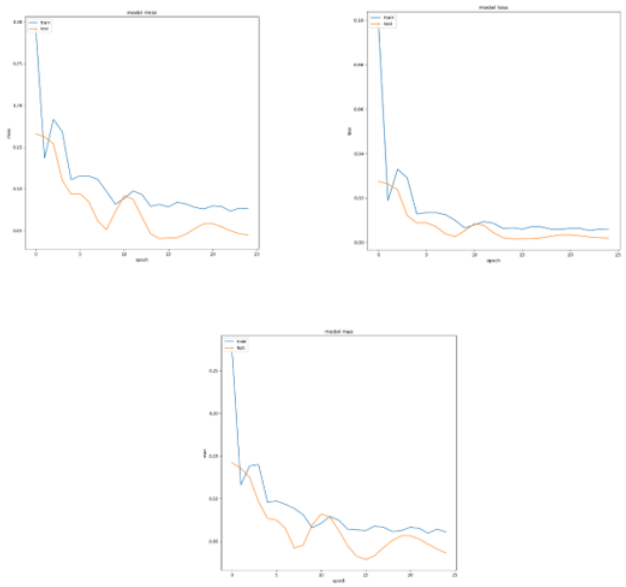

FIGURE 4.2 – Les plots des métriques MSE, MAE, RMSE avec le modèle(LSTM) pour FD001 de CMAPSS.

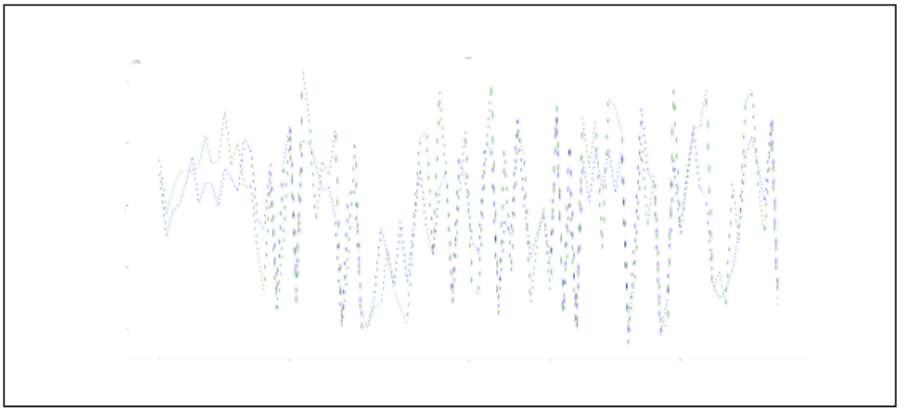

FIGURE 4.3 – Résultat de prédiction de RUL avec le modèle (LSTM) basée sur la sous données FD001 de C-MAPSS.

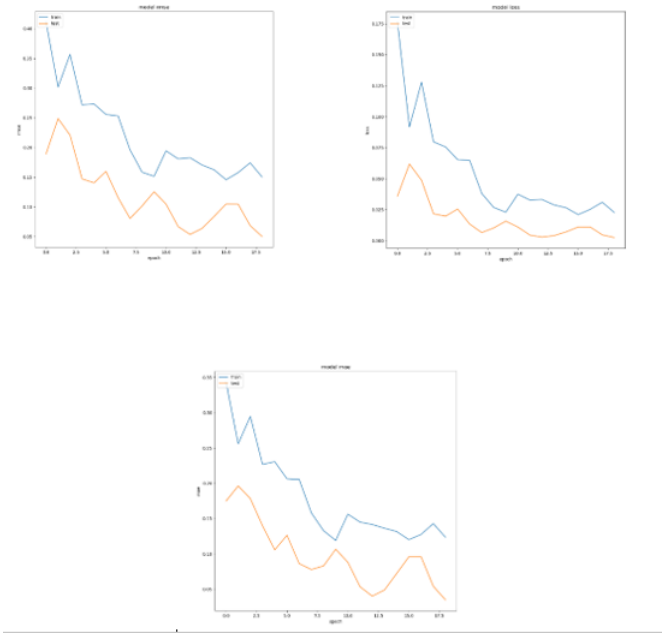

FIGURE 4.4 – Les plots des métriques MSE, MAE, RMSE avec le modèle(Gru) pour la sous donnée FD001 de CMAPSS.

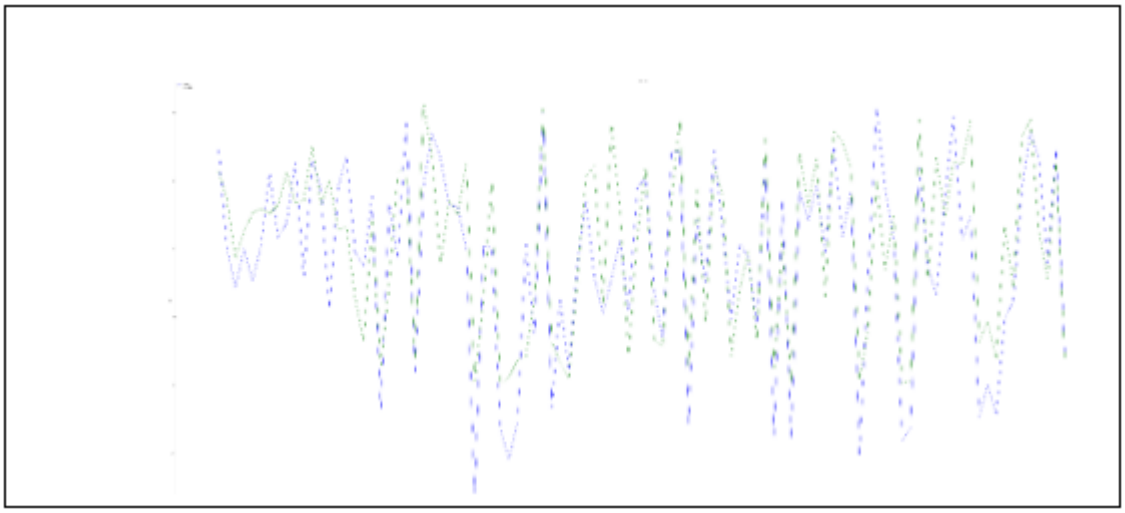

FIGURE 4.5 – Résultat de prédiction de RUL avec le modèle (Gru) basée sur la sous données FD001 de C-MAPSS.

4.13 Comparaison entre nos résultats et quelques travaux de la littérature

Dans le tableau suivant, nous effectuons une comparaison des résultats obtenus et ceux des travaux connexes dans la littérature utilisant le même sous ensemble de donnée. Nous mentionnons que :
LSTM + A : Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation
Ensemble model : Ensemble recurrent neural network-based residual useful life prognostics of aircraft engines

metriques	RMSE	MAE	MSE
Optimized LSTM proposé	0.09	0.07	0.008
Optimized LSTM [63]	0.0842	0.0581	0.0071
LSTM + A [64]	0.1091	-	0.0119
Ensemble model[65]	0.1505	-	0.0226

TABLE 4.7 – comparatif entre nos résultats et ceux des travaux de la litérature de test

Les résultats montrent que la méthode proposée donne de meilleurs résultats par rapport au résultats du travails[64][65] et une résultat proche au résultat du travail [63]. Ce dernier a appliqué la même méthode d'optimisation mais avec moins d'hyperparamètres, ce qui montre l'efficacité des hyperparamètres pris par notre travail et aussi plus en prendre en considération d'autres hyperparamètres plus l'efficacité sera augmentée.

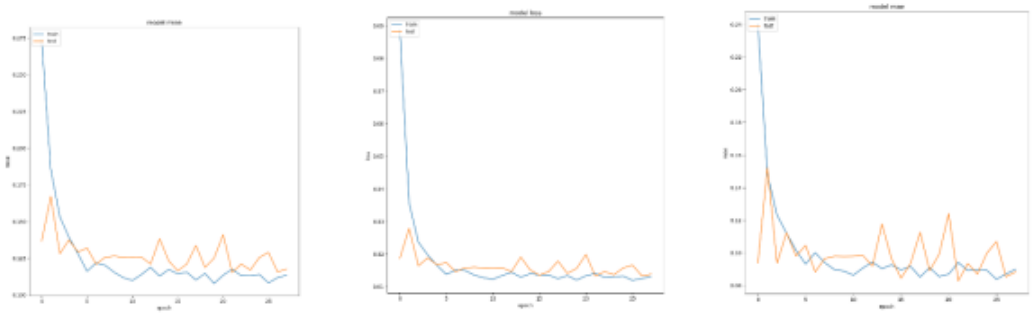

FIGURE 4.6 – Les plots des métriques MSE, MAE et RMSE avec le modèle (LSTM) pour les 4 sousdataset fusionnés de CMAPSS.

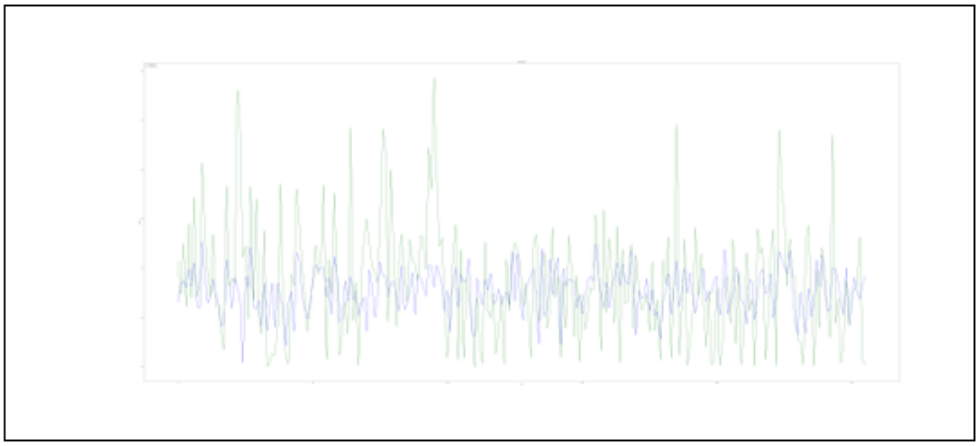

FIGURE 4.7 – Résultat de prédiction de RUL avec le modèle (LSTM) basé sur les 4 sousdataset fusionnés de CMAPSS

4.14 Conclusion

Nous avons proposé dans ce chapitre un modèle Lstm et GRU pour la prédiction de RUL ainsi que l'algorithme génétique pour l'optimisation des hyper-paramètres. A la fin, nous avons comparé le modèle avec les autres approches similaires dans la littérature. L'implémentation de ce modèle sera présenté dans le prochain chapitre.

Chapitre 5

IMPLÉMENTATION

5.1 Introduction

Dans ce chapitre nous avons présenté les outils utilisés et les pseudo codes implémentés pour la réalisation du modèle proposé.

5.2 Les outils de développement

Dans notre travail nous avons utilisé Python comme langage de programmation, Anaconda comme environnement de programmation et Spyder comme outil de programmation.

5.2.1 Le langage Python

Le langage Python est un langage de programmation open source multi-plateformes et orienté objet. Grâce à des bibliothèques spécialisées, Python s'utilise pour de nombreuses situations comme le développement logiciel, l'analyse de données, ou la gestion d'infrastructures. Il n'est donc pas, comme le langage HTML par exemple, uniquement dédié à la programmation web. Python est principalement utilisé pour le scripting et l'automatisation de tâches simples mais fastidieuses, c'est-à-dire l'interaction avec les navigateurs web. Mais Python est aussi utilisé pour : programmer des applications, générer du code, créer des services web, faire de la métaprogrammation. Langage principalement utilisé pour la machine Learning et la data science, Python

a fortement contribué à l'essor du big data. Grâce à ses nombreuses bibliothèques telles Panda, Bokeh, Numpy, Scipy, Scrapy, Matpotlib, Scikit-Learn ou encore TensorFlow, Python offre une grande flexibilité dans les tâches à effectuer et une grande compatibilité quelle que soit la plateforme utilisée [66]

5.2.2 Environnement Anaconda

Anaconda est une distribution open source pour Python et R. Il est utilisé pour la science des données, l'apprentissage automatique, l'apprentissage en profondeur, etc. Avec la disponibilité de plus de 300 bibliothèques pour la science des données, il devient assez optimal pour tout programmeur de travailler sur Anaconda pour science des données. Anaconda aide à simplifier la gestion et le déploiement des packages. Anaconda est livré avec une grande variété d'outils pour collecter facilement des données à partir de diverses sources à l'aide de divers algorithmes d'apprentissage automatique et d'IA [67].

5.2.3 L'outil Spyder

Ceci est un environnement de développement interactif puissant pour le langage Python. J'ai des fonctionnalités d'édition avancées, des tests interactifs, le débogage et l'introspection, ainsi qu'un environnement de calcul numérique. Grâce au support d'IPython (interpréteur Python interactif amélioré) et les bibliothèques Python populaires telles que NumPy, SciPy ou matplotlib (Tracé interactif 2D / 3D). Spyder peut également être utilisé comme bibliothèque qui fournit des widgets puissants liés à la console pour nos applications basées sur PyQt. Il peut être utilisé pour intégrer une console de débogage directement dans la conception de votre interface utilisateur graphique[68].

5.2.4 Les bibliothèques Tensorflow et Keras

TensorFlow est une bibliothèque open source, permettant d'exécuter des applications de machine learning et de deep learning. Cet outil dédié à l'apprentissage automatique a été développé par Google, et est fortement utilisé dans le domaine de l'intelligence artificielle (IA). Ainsi, des professionnels comme des novices peuvent créer des modèles de machine learning ou de deep learning pour optimiser les capacités de leur matérie[69]. Keras est un

framework de deep learning open source. Cet outil peut fonctionner au-dessus de TensorFlow, Theano, Microsoft Cognitive Toolkit et PlaidML. L'USP de Keras est sa vitesse – il est livré avec un support intégré pour le parallélisme des données, et donc, il peut traiter des volumes massifs de données tout en accélérant le temps de formation pour les modèles. Comme il est écrit en Python, il est incroyablement facile à utiliser et extensible. Si Keras se comporte brillamment pour les calculs de haut niveau, les calculs de bas niveau ne sont pas son point fort. Pour les calculs de bas niveau, Keras utilise une autre bibliothèque appelée backend [70]

FIGURE 5.1 – Les bibliothèques Tensorflow et Kera

5.2.5 La bibliothèque DEAP

Nous donnons une évaluation critique de la bibliothèque open source DEAP (Distributed Evolutionary Algorithm in Python) et la recommandons vivement aux débutants comme aux experts. DEAP prend en charge une gamme d'algorithmes évolutifs, y compris la programmation génétique fortement et faiblement typée, l'algorithme génétique et les algorithmes évolutifs multi-objectifs tels que NSGA-II et SPEA2. Il contient la plupart des fonctions de base requises par le calcul évolutif, de sorte que ses utilisateurs peuvent facilement construire diverses variantes d'algorithmes évolutifs à un ou plusieurs objectifs et les exécuter à l'aide de plusieurs processeurs. Il est idéal pour le prototypage rapide et peut être utilisé avec une multitude d'autres bibliothèques Python pour le traitement des données ainsi que d'autres techniques d'apprentissage automatique[71].

5.3 Quelques Fragments de code

```
30    # read training data
31    train_df = pd.read_csv('C:/Users/User/Desktop/projet/Dataset/train_FD001.txt', sep=" ", header=None)
32    train_df.drop(train_df.columns[[26, 27]], axis=1, inplace=True)
33    train_df.columns = ['id', 'cycle', 'setting1', 'setting2', 'setting3', 's1', 's2', 's3',
34                        's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14',
35                        's15', 's16', 's17', 's18', 's19', 's20', 's21']
36
37    train_df = train_df.sort_values(['id','cycle'])
38
39    # read test data
40    test_df = pd.read_csv('C:/Users/User/Desktop/projet/Dataset/test_FD001.txt', sep=" ", header=None)
41    test_df.drop(test_df.columns[[26, 27]], axis=1, inplace=True)
42    test_df.columns = ['id', 'cycle', 'setting1', 'setting2', 'setting3', 's1', 's2', 's3',
43                        's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14',
44                        's15', 's16', 's17', 's18', 's19', 's20', 's21']
45
46    # read ground truth data .
47    truth_df = pd.read_csv('C:/Users/User/Desktop/projet/Dataset/RUL_FD001.txt', sep=" ", header=None)
48    truth_df.drop(truth_df.columns[[1]], axis=1, inplace=True)
49
```

FIGURE 5.2 – Lecture des données

```
# TRAIN
#######
# Data Labeling - generate column RUL(Remaining Usefull Life or Time to Failure)
rul = pd.DataFrame(train_df.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id', 'max']
train_df = train_df.merge(rul, on=['id'], how='left')
train_df['RUL'] = train_df['max'] - train_df['cycle']
train_df.drop('max', axis=1, inplace=True)

# MinMax normalization (from 0 to 1)
train_df['cycle_norm'] = train_df['cycle']
cols_normalize = train_df.columns.difference(['id','cycle'])
min_max_scaler = preprocessing.MinMaxScaler()
norm_train_df = pd.DataFrame(min_max_scaler.fit_transform(train_df[cols_normalize]),
                             columns=cols_normalize,
                             index=train_df.index)
join_df = train_df[train_df.columns.difference(cols_normalize)].join(norm_train_df)
train_df = join_df.reindex(columns = train_df.columns)
```

FIGURE 5.3 – Normalisation des données de train

```
# TEST
######
# We use the ground truth dataset to generate labels for the test data.
# generate column max for test data
rul = pd.DataFrame(test_df.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id', 'max']
truth_df.columns = ['more']
truth_df['id'] = truth_df.index + 1
truth_df['max'] = rul['max'] + truth_df['more']
truth_df.drop('more', axis=1, inplace=True)

# generate RUL for test data
test_df = test_df.merge(truth_df, on=['id'], how='left')
test_df['RUL'] = test_df['max'] - test_df['cycle']
test_df.drop('max', axis=1, inplace=True)

# MinMax normalization (from 0 to 1)
test_df['cycle_norm'] = test_df['cycle']
cols_normalize = test_df.columns.difference(['id','cycle'])
min_max_scaler = preprocessing.MinMaxScaler()
norm_test_df = pd.DataFrame(min_max_scaler.fit_transform(test_df[cols_normalize]),
                            columns=cols_normalize,
                            index=test_df.index)
test_join_df = test_df[test_df.columns.difference(cols_normalize)].join(norm_test_df)
test_df = test_join_df.reindex(columns = test_df.columns)
test_df = test_df.reset_index(drop=True)
```

FIGURE 5.4 – Normalisation des données de test

```
def create_modele(ws, nb_f,nb_o,num_u1,num_u2):
    model = Sequential()
    model.add(LSTM(input_shape=(ws, nb_f),units=num_u1,return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(units=num_u2,return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(units=nb_o, activation='tanh'))
    model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mae','mse',rmse])
    return model
```

FIGURE 5.5 – Le modèle LSTM

```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_48 (LSTM)              (None, 63, 127)           77724

 dropout_48 (Dropout)        (None, 63, 127)           0

 lstm_49 (LSTM)              (None, 127)               129540

 dropout_49 (Dropout)        (None, 127)               0

 dense_24 (Dense)            (None, 1)                 128

=================================================================
Total params: 207,392
Trainable params: 207,392
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 5.6 – La structure du Modèle LSTM

```
history =model.fit(seq_array_test, label_array_test, epochs=10, batch_size=best_batch_size, validation_split=0.05, verbose=2,
      callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=0, mode='min'),
                   keras.callbacks.ModelCheckpoint(model_path,monitor='val_loss', save_best_only=True, mode='min', verbose=0)]
    )
```

```
Epoch 1/10
3/3 - 5s - loss: 0.0396 - mae: 0.1617 - mse: 0.0396 - rmse: 0.2162 - val_loss: 0.0296 - val_mae: 0.1407 - val_mse: 0.0296 -
val_rmse: 0.1722 - 5s/epoch - 2s/step
Epoch 2/10
3/3 - 0s - loss: 0.0222 - mae: 0.1236 - mse: 0.0222 - rmse: 0.1460 - val_loss: 0.0327 - val_mae: 0.1529 - val_mse: 0.0327 -
val_rmse: 0.1808 - 125ms/epoch - 42ms/step
Epoch 3/10
3/3 - 1s - loss: 0.0317 - mae: 0.1464 - mse: 0.0317 - rmse: 0.1694 - val_loss: 0.0089 - val_mae: 0.0746 - val_mse: 0.0089 -
val_rmse: 0.0941 - 912ms/epoch - 304ms/step
Epoch 4/10
3/3 - 0s - loss: 0.0113 - mae: 0.0897 - mse: 0.0113 - rmse: 0.1096 - val_loss: 0.0216 - val_mae: 0.1218 - val_mse: 0.0216 -
val_rmse: 0.1471 - 138ms/epoch - 46ms/step
Epoch 5/10
3/3 - 0s - loss: 0.0138 - mae: 0.0980 - mse: 0.0138 - rmse: 0.1065 - val_loss: 0.0126 - val_mae: 0.0966 - val_mse: 0.0126 -
val_rmse: 0.1123 - 106ms/epoch - 35ms/step
Epoch 6/10
3/3 - 1s - loss: 0.0090 - mae: 0.0761 - mse: 0.0090 - rmse: 0.0863 - val_loss: 0.0028 - val_mae: 0.0416 - val_mse: 0.0028 -
val_rmse: 0.0533 - 816ms/epoch - 272ms/step
Epoch 7/10
3/3 - 1s - loss: 0.0105 - mae: 0.0784 - mse: 0.0105 - rmse: 0.1012 - val_loss: 0.0023 - val_mae: 0.0339 - val_mse: 0.0023 -
val_rmse: 0.0484 - 606ms/epoch - 202ms/step
Epoch 8/10
3/3 - 0s - loss: 0.0090 - mae: 0.0720 - mse: 0.0090 - rmse: 0.0885 - val_loss: 0.0040 - val_mae: 0.0600 - val_mse: 0.0040 -
val_rmse: 0.0632 - 102ms/epoch - 34ms/step
Epoch 9/10
```

FIGURE 5.7 – l'exécution du Modèle LSTM

```
scores_test = model.evaluate(seq_array_test, label_array_test, verbose=1,batch_size=best_batch_size)
print('\nTest MAE: {}'.format(scores_test[1]))
print('\nTest MSE: {}'.format(scores_test[2]))
y_pred_test = model.predict(seq_array_test)
y_true_test = label_array_test
RMSE = np.sqrt(mean_squared_error(label_array_test, y_pred_test))
print('Test RMSE: ', RMSE)
```

```
Test MAE: 0.0557369627058506

Test MSE: 0.0049482788890600204
Test RMSE:  0.070344
```

FIGURE 5.8 – L'exécution de testing du Modèle LSTM

5.4 Conclusion

Dans ce chapitre nous avons présenté les outils de développement utilisés pour la réalisation du modèle proposé ainsi que quelques fragments de code associés.

Conclusion générale

Notre projet de fin d'études porte sur l'estimation du temps utile restant (RUL) pour la maintenance conditionnelle. L'objectif principal était dans un premier lieu, de proposer une architecture à base de deep learning. Ensuite dans une deuxième étape d'utiliser une méthode métaheuristique pour optimiser les hyparparamètres de ce modèle afin d'aboutir à des meilleurs résultats dans un temps résaunablle. Pour mener à bien notre travail, nous avons utilisé LSTM, méthode qui a prouvée son efficacité dans les problèmes de prédiction de séries chronologiques (time series) qui sont un type difficile de problèmes de modélisation prédictive ainsi que ses capacités dans l'extraction des caractéristiques. Et l'algorithme génétique pour l'optimisation. Une comparaison avec les autres modèles similaires de la littérature est effectuée afin d' étudier l'efficacité de la méthode par rapport au problème et par rapport aux autres méthodes proposées dans le domaine.
Ce projet s'est avéré bénéfique sur plusieurs plans. Il nous a permis d'acquérir de nouvelles connaissances et de renforcer d'autres. Il nous a permis de découvrir notamment le domaine de la maintenance, la maintenance conditionnelle, le RUL et le deep learning qui nous en font nécessiter une recherche bibliographique profonde. et de renforcer nos connaissances dans les algorithmes génétiques car nous avions une formation auparavant générale et non exhaustive. Il nous a permis aussi d'apprendre un nouveau langage de programmation qui est Python et ses différentes technologies.
Lors de la réalisation de ce travail, nous avons rencontré des difficultés, nous mentionnons :

1. Accès difficile aux documents et ressources qui ne sont pas toujours gratuits.
2. Dans la majorité des ressources disponibles, il y a des incohérences dans la sélection des modèles de comparaison et des critères d'évaluation, ce qui rend difficile la validation des résultats.

3. Plus important encore, le manque d'équipements physiques suffisamment robustes pour faciliter l'apprentissage et l'essai des méthodes appliquées. Ce problème fait également du temps d'exécution sur les autres groupes (FD002 et FD004) la base de données CMAPSS la plus longue (plus de 24 heures pour s'exécuter).

Au final, on peut dire que notre travail a atteint ses objectifs initialement fixés. Nous avons pu optimiser les hyperparamètres voyons essentiels de LSTM, et comparer les résultats obtenus avec celles des travaux similaires de la littérature. Les résultats de notre proposition ont justifié l'efficacité du modèle LSTM, de l'algorithme génétique ainsi que ses paramètres choisis. Mais comme tout travail scientifique, il reste ouvert à de nouvelles améliorations. Comme points de vue, on peut citer les suivants :

— Tester notre modèle sur d'autres jeux de données de RUL.
— Ajouter d'autres hyperparamètres comme le nombre de couches du réseau de neurone et autre pour améliorer la performance du modèle.
— Essayez d'appliquer une autre méthode d'optimisation à base de population comme le PSO ou l'Algorithme à évolution différentielle, et une méthode à base de solution unique comme la recherche Tabou ou autre et comparer les résultats.
— Faire une hybidation entre l'AG appliquée et une autre méthode d'optimisation.
— Utiliser un autre type de deep learning comme le CNN, l'autoencoder, Gru ou une hybridation.
— Essayez une autre comparaison et/ou intersection avec d'autres méthodes de régression.
— Les techniques d'apprentissage d'ensemble (ensemble learning) pourraient être utilisé sur la génération finale pour obtenir des valeurs moyennes.

Bibliographie

[1] Mana BERTRAND, Lucie GARET, Brice NORD, and Adan RIAZ. La qualité des dispositifs médicaux en exploitation : la norme iso 13485 adaptable aux services biomédicaux. *université de technologie de Compiègne, Master Technologies et Territoires de Santé, Mémoire d'Intelligence Méthodologique du projet d'intégration*, 2013.

[2] Messaoud Amira, Aissam Boulechefar, and Rachid Encadreur Belhadef. *Optimisation et amélioration de la maintenance par la fiabilité : Cas d'une turbine d'une centrale électrique.* PhD thesis, Université de Jijel, 2019.

[3] Ahmed Mosallam. *Remaining useful life estimation of critical components based on Bayesian Approaches.* PhD thesis, Université de Franche-Comté, 2014.

[4] Ouali Ilyes Belkasmi fahem. Memoire fin d'étude. `https://fr.scribd.com/document/566828949/memoire-fin-d-etude`, 2017-2018.

[5] Estelle Deloux. *Politiques de maintenance conditionnelle pour un système à dégradation continue soumis à un environnement stressant.* PhD thesis, Université de Nantes, 2008.

[6] Diego Jair Rodriguez Obando. *From Deterioration Modeling to Remaining Useful Life Control : a comprehensive framework for post-prognosis decision-making applied to friction drive systems.* PhD thesis, Université Grenoble Alpes, 2018.

[7] Racha Khelif. *Estimation du RUL par des approches basées sur l'expérience : de la donnée vers la connaissance.* PhD thesis, Université de Franche-Comté, 2015.

[8] Ali Al-Dulaimi, Soheil Zabihi, Amir Asif, and Arash Mohammadi. Hybrid deep neural network model for remaining useful life estimation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3872–3876. IEEE, 2019.

[9] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172 :1–11, 2018.

[10] Paulo Roberto de Oliveira da Costa, Alp Akçay, Yingqian Zhang, and Uzay Kaymak. Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering & System Safety*, 195 :106682, 2020.

[11] Piero Baraldi, Michele Compare, Sergio Sauco, and Enrico Zio. Ensemble neural network-based particle filtering for prognostics. *Mechanical Systems and Signal Processing*, 41(1-2) :288–300, 2013.

[12] Cédric Beaulac. *Intelligence artificielle avec apprentissage automatique pour l'estimation de la position d'un agent mobile en utilisant les modèles de Markov cachés*. PhD thesis, Université du Québec à Montréal, 2015.

[13] Pramila P Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pages 1–6. IEEE, 2018.

[14] Asma AMRAOUI and Badr BENMAMMAR. Optimisation des réseaux à l'aide des techniques de l'intelligence artificielle. *Gestion et contrôle intelligents des réseaux : Sécurité intelligente, optimisation multicritères, Cloud Computing, Internet of Vehicles, radio intelligente*, page 71, 2020.

[15] Mokhtar TAFFAR. Initiation al'apprentissage.

[16] Ahmed Nait-Chabane, Benoit Zerr, and Gilles Le Chenadec. Segmentation des images sonar latéral assurant l'invariance en rasance. *Traitement du signal*, 30(3-5) :119–148, 2013.

[17] Marc Parizeau. Réseaux de neurones. *GIF-21140 et GIF-64326*, 124, 2004.

[18] Bezza Khouloud. *Reconnaissance du visage, application aux dessins animés.* PhD thesis, Université Larbi Tébessi Tébessa, 2021.

[19] Mohammed Msaaf and Fouad Belmajdoub. L'application des réseaux de neurone de type «feedforward» dans le diagnostic statique. In *Xème Conférence Internationale : Conception et Production Intégrées*, 2015.

[20] Mehdi Amian. *Differential Privacy in Deep Learning Models.* PhD thesis, Maîtrise en télécommunications, 2022.

[21] Nithin Buduma, Nikhil Buduma, and Joe Papa. *Fundamentals of deep learning.* " O'Reilly Media, Inc.", 2022.

[22] Gregory Gelly. *Reseaux de neurones recurrents pour le traitement automatique de la parole.* PhD thesis, Université Paris Saclay (COmUE), 2017.

[23] Eithar Alfatih Abdalrahman Elbasheer. *Speech Activity Detection Implementation in Hearing Aids Using Deep Belief Network.* PhD thesis, University of Khartoum, 2017.

[24] Mohaiminul Islam, Guorong Chen, and Shangzhu Jin. An overview of neural network. *American Journal of Neural Networks and Applications*, 5(1) :7–11, 2019.

[25] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks : an overview and application in radiology. *Insights into imaging*, 9(4) :611–629, 2018.

[26] Wikistat. Classification non-supervisée — wikistat, 2016. [En ligne ; Page disponible le 21-janvier-2016].

[27] El Mahdi Brakni. *Réseaux de neurones artificiels appliqués à la méthode électromagnétique transitoire InfiniTEM.* Universite du Quebec en Abitibi-Temiscamingue (Canada), 2011.

[28] Christopher Olah. Understanding lstm networks–colah's blog. *Colah. github. io*, 2015.

[29] Tawfik Beghriche et al. *Prédiction des maladies les plus fréquentes par les techniques de deep learning.* PhD thesis, Univ M'sila, 2020.

[30] Ludovic Arnold, Sébastien Rebecchi, Sylvain Chevallier, and Hélène Paugam-Moisy. An introduction to deep learning. In *European Symposium on Artificial Neural Networks (ESANN)*, 2011.

[31] Nicolás Oyharcabal Astorga. Convolutional recurrent neural networks for remaining useful life prediction in mechanical systems. 2018.

[32] C Charu. Aggarwal neural networks and deep learning, 2018.

[33] Eithar Alfatih Abdalrahman Elbasheer. *Speech Activity Detection Implementation in Hearing Aids Using Deep Belief Network*. PhD thesis, University of Khartoum, 2017.

[34] Jason Brownlee. *Long short-term memory networks with python : develop sequence prediction models with deep learning*. Machine Learning Mastery, 2017.

[35] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172 :1–11, 2018.

[36] Rémy-Robert Joseph. *Systèmes interactifs d'aide à l'élaboration de plannings de travail de personnel : Contraintes, aide à la décision, représentation combinatoire des préférences, équité et résolution par décomposition arborescente et par consistance*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2003.

[37] Christophe Rapine. denis trystram, 2002, théorie de la complexité. *Notes de cours, ENSGI–INP Grenoble*.

[38] El-Ghazali Talbi. *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons, 2009.

[39] Jin-Kao Hao, Philippe Galinier, and Michel Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 13(2) :283–324, 1999.

[40] Laurent Noé. *Recherche de similarités dans les séquences d'ADN : modèles et algorithmes pour la conception de graines efficaces*. PhD thesis, Université Henri Poincaré-Nancy 1, 2005.

[41] Tufféry Stéphane. *Data mining et statistique décisionnelle : l'intelligence des données*. Editions Technip, 2012.

[42] Ilhem Boussaid. *Perfectionnement de métaheuristiques pour l'optimisation continue*. PhD thesis, Paris Est, 2013.

[43] Jean-Charles Boisson, Laetitia Jourdan, El-Ghazali Talbi, and Dragos Horvath. Single-and multi-objective cooperation for the flexible docking problem. *Journal of Mathematical Modelling and Algorithms*, 9(2) :195–208, 2010.

[44] Laetitia Jourdan. *Métaheuristiques pour l'extraction de connaissances : application à la génomique*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2003.

[45] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598) :671–680, 1983.

[46] Djamila Arbia. *Métaheuristiques appliquées à la classification non supervisée de données*. PhD thesis, UNIVERSITE MOHAMED BOUDIAF-M'SILA FACULTE DES MATHEMATIQUES ET DE L ..., 2019.

[47] Walid Tfaili. *Conception d'un algorithme de fourmis pour l'optimisation continue dynamique*. PhD thesis, Paris 12, 2007.

[48] Ilhem Boussaid. *Perfectionnement de métaheuristiques pour l'optimisation continue*. PhD thesis, Paris Est, 2013.

[49] Chong Zhang, Pin Lim, A Kai Qin, and Kay Chen Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, 28(10) :2306–2318, 2016.

[50] André Listou Ellefsen, Emil Bjørlykhaug, Vilmar Æsøy, Sergey Ushakov, and Houxiang Zhang. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183 :240–251, 2019.

[51] Yuting Wu, Mei Yuan, Shaopeng Dong, Li Lin, and Yingqi Liu. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275 :167–179, 2018.

[52] Jun Wu, Kui Hu, Yiwei Cheng, Haiping Zhu, Xinyu Shao, and Yuanhang Wang. Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network. *ISA transactions*, 97 :241–250, 2020.

[53] Yi-Wei Lu, Chia-Yu Hsu, and Kuang-Chieh Huang. An autoencoder gated recurrent unit for remaining useful life prediction. *Processes*, 8(9) :1155, 2020.

[54] Kwok Tai Chui, Brij B Gupta, and Pandian Vasant. A genetic algorithm optimized rnn-lstm model for remaining useful life prediction of turbofan engine. *Electronics*, 10(3) :285, 2021.

[55] Shaashwat Agrawal, Sagnik Sarkar, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. Genetically optimized prediction of remaining useful life. *Sustainable Computing : Informatics and Systems*, 31 :100565, 2021.

[56] Tangbin Xia, Ya Song, Yu Zheng, Ershun Pan, and Lifeng Xi. An ensemble framework based on convolutional bi-directional lstm with multiple time windows for remaining useful life estimation. *Computers in Industry*, 115 :103182, 2020.

[57] Samarth Agrawal. Hyperparameters in deep learning, 2019.

[58] Ali Al-Dulaimi, Soheil Zabihi, Amir Asif, and Arash Mohammadi. Hybrid deep neural network model for remaining useful life estimation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3872–3876. IEEE, 2019.

[59] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172 :1–11, 2018.

[60] Ralf C Staudemeyer and Christian W Omlin. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In *Proceedings of the South African institute for computer scientists and information technologists conference*, pages 218–224, 2013.

[61] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm : a big comparison for nas. *arXiv preprint arXiv :1912.06059*, 2019.

[62] Divyanshu Mishra. Regression : An explanation of regression metrics and what can go wrong. *Towards Data Science*, 2019.

[63] Shaashwat Agrawal, Sagnik Sarkar, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. Genetically optimized prediction of remaining useful life. *Sustainable Computing : Informatics and Systems*, 31 :100565, 2021.

[64] Paulo Roberto De Oliveira Da Costa, Alp Akcay, Yingqian Zhang, and Uzay Kaymak. Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation. *International Journal of Prognostics and Health Management*, 10(4), 2019.

[65] Jun Wu, Kui Hu, Yiwei Cheng, Ji Wang, Chao Deng, and Yuanhan Wang. Ensemble recurrent neural network-based residual useful life prognostics of aircraft engines. *Structural Durability & Health Monitoring*, 13(3) :317, 2019.

[66] la rédaction de Futura. Définition : Python : Futura tech.

[67] Mohammad Waseem ·, Mohammad Waseem, Oct. 22, and Like(14)Comment. Anaconda python tutorial : Everything you need to know - dzone big data, Oct 2019.

[68] Damián A. Spyder, un puissant environnement de développement interactif pour python, Dec 2017.

[69] Pramila P Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pages 1–6. IEEE, 2018.

[70] Quels sont les frameworks utilisés en deep learning ?, Aug 2021.

[71] Jinhan Kim and Shin Yoo. Software review : Deap (distributed evolutionary algorithm in python) library. *Genetic Programming and Evolvable Machines*, 20(1) :139–142, 2019.