Détection des défauts dans un environnement de maintenance prédictive

par

Alexandre CHARTRAND

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE ÉLECTRIQUE
M. Sc. A.

MONTRÉAL, LE 22 DÉCEMBRE 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Tony Wong, directeur de mémoire
Département de génie des systèmes à l'École de technologie supérieure

M. Chakib Tadj, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Mohamad Saad, membre du jury
École de génie à l'Université du Québec en Abitibi-Témiscamingue

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 24 NOVEMBRE 2021

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Détection des défauts dans un environnement de maintenance prédictive

Alexandre CHARTRAND

RÉSUMÉ

L'anticipation des défaillances est une fonction importante en maintenance prédictive. Ce mémoire propose une méthodologie capable de prédire un nombre de défauts en utilisant des signaux captés des moteurs à courant continu. Ces moteurs électriques sont très présents dans le monde industriel et en particulier dans des chaînes de production. Avec l'avènement de l'Internet industriel des objets, l'instrumentation des moteurs et la collecte des signaux générés sont grandement simplifiées. Désormais, l'utilisation des signaux enregistrés dans la prédiction des défauts est devenu un sujet de recherche très prisé dans le domaine de la maintenance industrielle.

Pour concevoir la fonction de la prédiction des défauts, des signaux temporels sont générés par simulation à l'aide de MATLAB et de Simulink. Ces signaux représentent différentes régimes de fonctionnement des moteurs soumis à des défauts mécaniques et électriques. Dans ce projet, cinq (5) défauts sont analysés sur trois (3) moteurs différents. Ces défauts sont : problème de roulement, court-circuit de l'alimentation, défaut des capteurs de vitesse, de courant et de tension. De plus, des consignes de vitesse sont appliquées aux moteurs afin de simuler des conditions d'utilisation tels le convoyage, le transfert et le levage. Au total 9000 signaux d'une durée de 5 secondes chacun sont générés pour ce projet.

Quatre (4) modèles prédictifs tirés des techniques d'apprentissage profond sont évalués à l'aide des signaux générés. Ces modèles sont : le FCN, le ResNet, l'Encodeur et le LSTM. La performance prédictive de ces modèles sont quantifiée par des métriques de classification telles l'exactitude, la précision, le rappel et le F-score. Le modèle prédictif Encodeur présente les meilleurs résultats lors de son entraînement obtenant 89.6% pour l'exactitude, 90.47% en précision, 90.08% pour le rappel et 90.28% pour le F-score. Ce même modèle appliqué à des nouvelles données produit une exactitude 88.53% et un rappel de 88.67%. Plus intéressant encore, le modèle Encodeur est en mesure de prédire correctement l'absence des défauts à un taux de 83.14% alors que le score est de 50.4% pour le FCN, 14.4% pour le ResNet et 54.4% pour le LSTM.

Enfin, le code source des expérimentations est disponible sur GitHub à : https://github.com /alexchartrand/Fault-detection-in-IIoT et les fichiers de simulation MATLAB-Simulink sont entreposés dans le dossier https://github.com/alexchartrand/DCMotor-fault-simulation.

Mots-clés: Maintenance prédictive, modèles prédictifs, apprentissage profond, simulation, séquences temporelles

Fault detection in a predictive maintenance environment

Alexandre CHARTRAND

ABSTRACT

Failure forecasting is an important function in predictive maintenance. This thesis proposes a methodology capable of predicting a number of faults using signals collected from permanent magnet DC motors. These electric motors are often used in the industrial world, especially in production lines. With the advent of the Industrial Internet of Things, condition monitoring and signal acquisition are now greatly simplified. That is why, fault prediction has become a prevalent research topic in the field of industrial maintenance.

In this work, motor signals are generated by simulation using MATLAB and Simulink. These signals represent different motor operating regimes subjected to mechanical and electrical faults. Five (5) types of faults are analyzed using three (3) different engines. They are bearing problems, power supply short circuit, speed, current and voltage sensor problems. In addition, speed setpoints are used as input reference simulating different motor operating conditions such as conveying, transfer and lifting. A total of 9000 signals each having a 5-second length are generated for this project.

Four (4) predictive models, FCN, ResNet, Encoder and LSTM, from deep learning techniques are trained and evaluated using the generated signals. Their performances are quantified using classification metrics such as accuracy, precision, recall and F-score. Experimental results indicate that the Encoder is the best model during training with an accuracy of 89.6%, a precision of 90.47%, a recall 90.08% and a F-score of 90.28%. This trained model when applied to new data was able to produce an accuracy of 88.53% and a recall of 88.67%. More interesting still, the Encoder model is able to correctly classify the absence of faults at a rate of 83.14% while the score is 50.4% for FCN, 14.4% for ResNet and 54.4% for LSTM.

Finally, source code for this project is available at : https://github.com/alexchartrand/Fault-detection-in-IIoT and MATLAB-Simulink signal generation files are stored in the folder https://github.com/alexchartrand/DCMotor-fault-simulation.

Keywords: Predictive maintenance, predictive models, deep learning, simulation, time series

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ALSTM-FCN	Attention LSTM-FCN
BN	Batch Normalisation
BRBs	Broken Rotor Bars
CART	Classification And Regression Trees
CNN	Convolutional Neural Network
FCN	Fully Convolutional Neural Network
GAP	Global Average Pooling
HRA	Hierarchical framework of the Recognition Algorithm
IIoT	Industrial Internet of Things
IoT	Internet of Things
ITSC	Inter-Turn Short Circuit
k-NN	k-Nearest Neighbors
Lr	Learning rate
LSTM	LongShort-Term Memory
LSTM-FCN	Long Short Term Memory Fully Convolutional Network
MALSTM-FCN	Multivariate Attention LSTM-FCN
MCDCNN	Multi Channel Deep Convolutional Neural Network
MCNN	Multi-scale Convolutional Neural Network
MLP	Multi layer perceptron
MLSTM-FCN	Multivariate LSTM-FCN
NLL	Negative Log Likelihood
OC	Objets Connectés
OSA-CBM	Open System Architecture for Condition-Based Monitoring

PDM	Product Data Management
PID	Proportionel, Intégral, Dérivé
PLM	Production Lifecycle Management
PMDC	Permanent Magnet Direct Currant
PReLU	Paremetric Rectified Linear Unit
PWM	Pulse Width Modulation
ReLU	Rectified Linear Units
ResNet	Residual Network
RNN	Recurent Neural Network
SGD	Stochastic Gradient Descent
t-LeNet	Time Le-Net
Time-CNN	Time Convolutional Neural Network
TWIESN	Time Warping Invariant Echo State Network

INTRODUCTION

0.1 Problématique

Les moteurs électriques font partie des actionneurs les plus utilisés dans l'industrie manufacturière québécoise et canadienne. En effet, dans bien des pays industrialisés, on estime que près de 70% de l'électricité consommée par le secteur manufacturier sont utilisées dans l'alimentation des entraînements électriques (Gómez *et al.*, 2020). En 2020, le Canada a importé pour près de 1,38 milliard USD de moteurs et générateurs électriques et 25% de ces importations concernent les moteurs à courant continu (CC) (TrendEconomy). Ces moteurs sont robustes, offrent un couple élevé au démarrage et ils sont bien adaptés pour des applications tels le convoyage et le levage dans les procédés de fabrication.

Dans un contexte industriel, les arrêts intempestifs causés par des pannes de moteurs réduisent la cadence des opérations et entravent la productivité. Ces temps morts non planifiés peuvent coûter cher aux entreprises. En effet, plus le délai de production est long, plus le coût de la main-d'œuvre directe de production est élevé (Auger, 2018). Advenant une panne, le personnel technique doit alors réparer sur-le-champ l'équipement afin de redémarrer la production au plus vite. Cela implique une disposition des ressources (pièces de rechange et personnel qualifié) en tout temps et augmente par le fait même les frais d'exploitation. Pour bien des entreprises, le stockage des pièces de rechange et le maintien du personnel technique pour la réparation des pannes représentent un fardeau financier supplémentaire. C'est pourquoi la maintenance réactive qui consiste à effectuer les réparations lorsque les bris surviennent n'est pas vraiment une solution efficace.

Une avenue plus intéressante pour les entreprises manufacturières est de planifier et effectuer la maintenance des moteurs selon des critères définis préalablement c'est ce qu'on nomme la maintenance préventive. L'objectif de la maintenance préventive est de réduire la probabilité de la

défaillance ou de la dégradation de l'équipement (Ministère de l'Économie et de l'Innovation du Québec, 2019). À intervalles réguliers et selon leur taux d'utilisation, les machines de production sont arrêtées pour fin de maintenance. La nature planifiée des opérations peut réduire l'impact des temps d'arrêt sur la cadence de la production. Il devient possible de minimiser l'inventaire des pièces de rechange stockées et de contracter les opérations de maintenance à l'externe. Or, pour être efficace, l'intervalle des opérations de maintenance doit être estimé avec précision. On veut que le potentiel de défaillance future soit identifié et éliminé lors de la maintenance préventive. Pour les entreprises qui opèrent en mode de production flexible et personnalisée, la détermination de l'intervalle des opérations de maintenance est plus complexe. Dans ce mode de production, les produits sont fabriqués sur demande et la quantité des produits à fabriquer peut varier (en anglais, c'est le *make-to-order manufacturing*) (Ministère de l'Économie et de l'Innovation du Québec, 2021), (ISA, 2016). Ainsi, le taux d'utilisation et le type d'équipement utilisé varieront selon les besoins de la clientèle. L'usure des actionneurs est alors une fonction des commandes à recevoir. En pratique, cela se traduit par des projections comportant une grande part d'incertitude. Pour cette raison, la maintenance à intervalles réguliers n'est pas aussi attrayante pour les entreprises œuvrant dans ce mode de production. Ce qui convient mieux à un environnement de production flexible est évidemment une maintenance qui est également flexible. C'est-à-dire, il est préférable de réaliser les opérations de maintenance au moment jugé propice.

La maintenance prédictive est une discipline naissante qui tente d'offrir cette flexibilité par le biais des technologies tirées de télécommunication, des techniques statistiques et de l'apprentissage automatique. L'idée de base est de réaliser la maintenance avant la défaillance des moteurs. Plus précisément, le schéma de la maintenance prédictive comprend l'analyse continue de l'état des moteurs et le suivi des tendances afin de reconnaître les dégradations ou défauts qui peuvent mener vers une défaillance (Mobley, 2002). On entrevoit, pour réaliser un tel schéma, la nécessité d'une infrastructure de communication et de stockage pour l'enregistrement continu

des données. Le moment propice pour enclencher les opérations de maintenance est alors estimé par des techniques statistiques jumelées à la capacité prédictive de l'apprentissage automatique. La Figure 0.1 résume les scénarios de maintenance mentionnés.

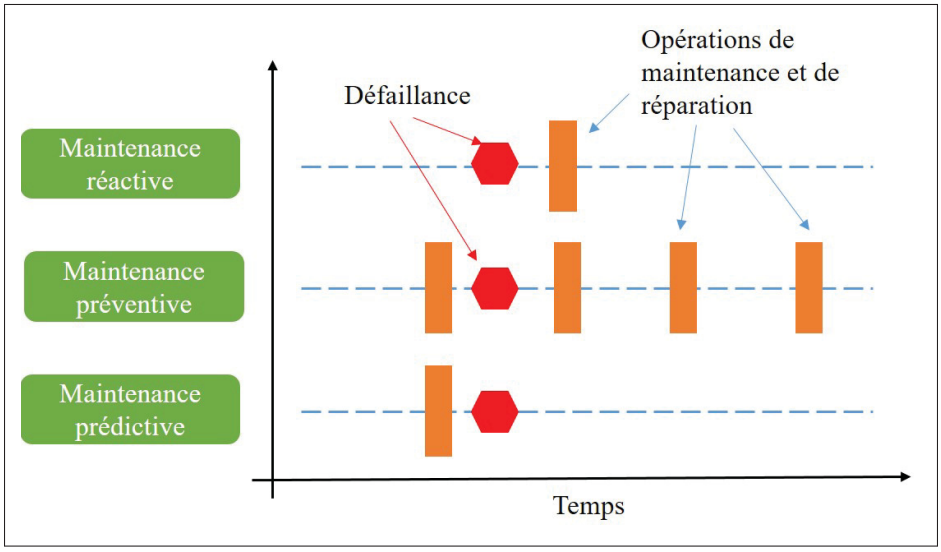


Figure 0.1 Différents scénarios de maintenance

Le scénario « Maintenance réactive » est celui dans lequel des réparations sont effectuées après avoir rencontré une défaillance. Il représente le scénario le plus coûteux sur le plan de la perte de production des trois scénarios de la Figure 0.1. Le scénario « Maintenance préventive » est une amélioration notable, car il impose une régularité dans les opérations de maintenance. La probabilité de la défaillance est réduite, mais en mode de production flexible ce type de maintenance peut devenir difficile à appliquer. Dans le scénario « Maintenance prédictive » les opérations de maintenance ne sont plus effectuées périodiquement. Elles seront réalisées lorsqu'il y aura indication probante de défaut. Ce scénario convient à bien des entreprises qui doivent reconfigurer fréquemment les équipements de production pour répondre aux commandes reçues et les commandes à venir.

D'après le scénario « Maintenance prédictive » de la Figure 0.1, le moment où les opérations de maintenance sont enclenchées est estimé à partir des statistiques de bris passés et l'usure des

moteurs. Cela suppose, au préalable, l'enregistrement des données d'opération pour fin d'analyse. L'enregistrement de ces données est grandement simplifié si les moteurs sont en mesure de fournir eux-mêmes les données pertinentes à leur maintenance. Ce fait est connu et la plupart des moteurs modernes sont dotés de fonctions de diagnostic. Si ces données sont communiquées à une unité de traitement, ce dernier pourra les transformer en informations utiles et le cas échéant produire un bilan d'intervention destiné au personnel technique. En automatisant le processus, le travail de la maintenance pourra être planifié et exécuté au moment opportun. Dans le cas où l'ensemble des systèmes de production d'un manufacturier serait doté d'un tel mécanisme, alors les données générées pourront former un portrait plus global et encore plus utile pour les décideurs. Il devient possible d'identifier l'impact sur la production des moteurs en arrêt de maintenance et éventuellement d'effectuer la prédiction des pannes. Dans le cas de la maintenance des moteurs, la maintenance automatique n'est pas encore possible puisque les machines de production n'ont pas la capacité de s'automaintenir. En revanche, rien n'empêche de guider les actions correctives en identifiant automatiquement les défauts pouvant mener à des pannes de fonctionnement. En agissant ainsi, l'identification des défauts peut alors aider à prévenir des pannes et à réduire le temps d'arrêt de la production.

0.2 Cadre de référence OSA-CBM

La prédiction des défauts dans un contexte de maintenance nécessite une infrastructure de soutien adéquate. Des références architecturales ont été proposées afin de standardiser les modules matériels et interfaces logicielles appropriées pour la maintenance prédictive. Dans ce projet de recherche, le cadre de référence *Open System Architecture for Condition-Based Monitoring* (OSA-CBM) est utilisé comme modèle conceptuel. L'OSA-CBM a l'avantage d'être à la fois ouvert et normé par l'organisation internationale de normalisation ISO. Ainsi, l'architecture ouverte de OSA-CBM assure l'interopérabilité des systèmes tandis que le standard ISO 13374

assure la pérennité et la reconnaissance internationale de cette architecture de référence en matière de monitoring et diagnostic des systèmes.

Ce cadre de référence comprend six blocs fonctionnels et sont montrés dans la Figure 0.2 (OSA-CBM). En plus des fonctionnalités associées, l'OSA-CBM suggère également les méthodes d'interfaçage entre les blocs et les structures de données à utiliser.

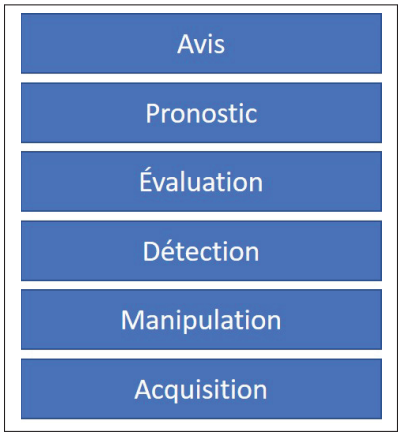


Figure 0.2 Six blocs fonctionnels du cadre de référence OSA-CBM

Les blocs « Acquisition » et « Manipulation » représentent les accès aux capteurs, la collecte des données, les transformations appliquées aux données et l'extraction des caractéristiques utiles pour le monitoring des systèmes. Le bloc « Détection » sert à identifier l'état des systèmes sous surveillance. En comparant les valeurs actuelles des caractéristiques aux profils historiques, on peut détecter dans quel état se trouvent les systèmes. Sachant l'état actuel des systèmes, le bloc « Évaluation » peut alors déterminer leur régime de fonctionnement (régime normal, en détérioration, en défaut, etc.). C'est la responsabilité du bloc « Pronostic » de produire une prévision sur l'évolution temporelle de l'état opérationnel des machines. Cette évolution est ensuite interprétée par le bloc « Avis » afin d'émettre des recommandations reliées aux activités de maintenance à effectuer.

Le présent projet de recherche propose l'implantation des blocs « Acquisition », « Manipulation », « Détection » et « Évaluation » par des techniques de l'apprentissage profond. L'acquisition des signaux par des capteurs sera remplacée par la simulation. Cette façon de faire est beaucoup plus économique puisqu'il n'est plus nécessaire de se doter d'un banc de tests. Plus encore, les conditions de tests seront facilement enregistrées et aisément reproduites. Quant aux blocs de « Manipulation », « Détection » et « Évaluation », ces fonctionnalités font partie du flux de travail de l'apprentissage automatique (Hapke & Nelson, 2020). Enfin, les fonctionnalités de pronostic ainsi que l'émission des recommandations sur les opérations de maintenance sont souvent liées à la politique et le fonctionnement interne des entreprises. Pour cette raison, ces fonctionnalités ne seront pas abordées dans ce mémoire.

0.3 Données et apprentissage profond

L'accumulation des données par l'Internet des objets industriels (IIoT) offre l'opportunité d'analyse par des méthodes modernes de l'apprentissage profond. Auparavant, de telles prédictions de pannes étaient généralement réalisées à l'aide de systèmes recueillant des données et capables d'alerter un opérateur sur la base de règles qui sont explicitement programmées pour imiter le processus décisionnel d'un expert humain (Mobley, 2002).

Avec l'essor de l'apprentissage profond, la maintenance prédictive s'est progressivement tournée vers de nouveaux algorithmes pour sa mise en œuvre. Ceux-ci nécessitent que nous leur fournissions des données, qu'il faut préalablement choisir et collecter. Ces données peuvent par exemple être des mesures de vibrations, de niveaux sonores, de courants ou de tensions. Autrement dit, les données les plus communes en matière de maintenance prédictive sont en fait des signaux temporels. Ces signaux sont prétraités et mis dans un format utilisable par des algorithmes de l'apprentissage profond. Dans ce mémoire, les signaux traités et formatés pour l'apprentissage seront désignés par le terme « séquences temporelles ». Cette désignation des

données renforce le fait que les signaux à l'état brut sont transformés en segments d'observation utilisables par des algorithmes de prédiction.

Comme mentionné dans la section 0.1, la collecte de données est simplifiée par l'essor récent des technologies d'objets connectés ainsi que par le développement de nouveaux outils de gestion des données. Ces données collectées sont utilisées, généralement lors d'une étape d'apprentissage dit supervisé, pour entraîner un modèle prédictif. Ce modèle entraîné permet de prédire les défauts pouvant mener à des pannes.

Les algorithmes d'apprentissage pouvant être utilisés pour la maintenance prédictive sont généralement classés en deux catégories :

- les algorithmes de régression, dont la prédiction produite est par exemple un nombre indicatif de la durée de vie utile restante d'une pièce ou d'un système. C'est-à-dire la durée dans laquelle la pièce ou le système demeure opérationnel jusqu'à la prochaine panne,
- les algorithmes de classification, qui peuvent notamment être utilisés pour prédire la cause la plus probable d'une panne. Ces algorithmes seront l'objet d'études dans ce projet de recherche.

Le choix des modèles sera basé sur le type de prédiction à intégrer dans le système de maintenance prédictive. Pour les algorithmes de classification, des informations sur les défauts des machines sont nécessaires. Ainsi, cinq (5) types de défauts seront générés sur trois (3) moteurs à courant continu. Pour bien représenter le contexte industriel, trois profils de vitesse seront appliqués à l'entrée de ces moteurs. Ces profils de vitesse correspondent aux opérations de convoyage, de levage et de transfert dans une chaîne de production. Les données résultant de ces diverses manipulations seront partitionnées d'une façon aléatoire en jeux de données. Ils seront utilisés

dans l'évaluation des modèles prédictifs. Enfin, les détails techniques de ces différents sujets seront présentés dans les prochains chapitres de ce mémoire.

0.4 Contenu du mémoire

Le contenu de ce mémoire est organisé de la façon suivante. Les objectifs de recherche et la méthodologie appliquée sont présentés dans le chapitre 1. Les éléments des modèles prédictifs profonds, les méthodes de réduction de la dimension ainsi que différentes approches récentes pour la classification des défauts sont détaillés dans le chapitre 2. Dans le chapitre 3, une procédure de génération par simulation est utilisée afin de produire des données convenables pour les expérimentations à entreprendre. Différents moteurs, consignes d'entrée et types de défaut sont ainsi générés et formatés en jeux de données pour l'entraînement et l'évaluation des modèles prédictifs. Des expériences, utilisant l'apprentissage supervisé, sont exécutées sur quatre (4) modèles prédictifs profonds et leur performance en classification est présentée dans le chapitre 4. C'est aussi dans ce dernier chapitre que la sélection du meilleur modèle prédictif est présentée.

CHAPITRE 1

OBJECTIFS ET MÉTHODOLOGIE DE RECHERCHE

1.1 Introduction

Une question importante découlant de la référence OSA-CBM concerne justement l'identification des défauts. Comment peut-on réaliser cette identification à partir des données générées lors du fonctionnement des machines ? La démarche moderne propose l'utilisant des techniques prédictives du domaine de l'apprentissage profond. Or, le répertoire des techniques proposées est très garni et l'on ne peut pas appliquer ces modèles prédictifs profonds sans une analyse fine de leurs caractéristiques et limitation. Ce projet de recherche propose donc une étude sur les modèles prédictifs profonds propices à la classification des défauts dans un contexte de maintenance prédictive. Établir, dans la mesure du possible, les caractéristiques souhaitées et examiner la performance des modèles prédictifs profonds dans l'identification des défauts des moteurs CC.

1.2 Objectifs

La maintenance industrielle est le fait d'entretenir un équipement de production dans un état où il peut assurer la fonction prévue. On installe un système de maintenance afin de prévenir les problèmes et diminuer les pertes de productivité. Dans ce projet de recherche, on suppose que le système de maintenance permet de mesurer en continu l'état de fonctionnement de l'équipement. Dans cet environnement industriel, la classification des défauts est une tâche qui vient minimiser les pannes futures en planifiant des opérations de maintenance uniquement lorsque la situation l'impose. Ainsi, les principaux objectifs de ce travail de recherche sont :

1. Proposer une approche permettant la classification des défauts des moteurs CC.
2. Déterminer s'il est possible de classifier les défauts sans l'extraction explicite des caractéristiques (*features extraction*).

3. Examiner la performance des modèles prédictifs profonds appliqués à cette tâche.

Le premier objectif vise à démontrer l'identification des défauts à l'aide des modèles prédictifs profonds. On connaît les capacités intrinsèques de ces modèles dans la reconnaissance d'images et dans la traduction des langages naturels. Il est à se demander s'ils peuvent offrir les mêmes capacités dans l'identification des défauts d'un système à partir des séquences temporelles. Le déploiement du système de classification sera grandement simplifié s'il est possible de prédire directement les défauts sans la nécessité d'extraire explicitement les caractéristiques propices à la classification. Normalement, l'extraction des caractéristiques est une étape nécessaire en apprentissage automatique. L'omission de cette étape peut-elle impacter négativement l'identification des défauts ? Le deuxième objectif a été formulé pour élucider cette question. À noter que l'atteinte des objectifs 1 et 2 ne peut garantir l'efficacité des opérations de maintenance. On peut très bien reconnaître les défauts sans la nécessité de l'extraction des caractéristiques. Or, si le taux de reconnaissance est faible alors le système résultant sera inutile. C'est pourquoi le troisième objectif est consacré à la performance des modèles prédictifs appliqués à la tâche. De plus, l'atteinte de cet objectif permettra la sélection des modèles utiles à la classification des défauts. Dans ce travail de recherche un taux de classification de 80% et plus sera considéré comme un taux de reconnaissance acceptable. Nous avons fixé nous même ce taux comme objectif de départ minimal à atteindre. Les modèles prédictifs présentant une performance supérieure à ce taux seront retenus pour fin de comparaison.

1.3 Méthodologie de recherche

L'approche préconisée pour ce projet de recherche est une adaptation du modèle en spiral conçu à l'origine pour le développement logiciel (Boehm, 2000). Ce modèle a l'avantage d'inclure explicitement les risques technologiques associés à un projet d'ingénierie. Le modèle en spiral considère la réalisation d'un projet comme un processus itératif comportant des tâches et des raffinements continus. Ainsi, la fin d'une tâche et le début de la tâche suivante ne sont pas

délimités d'une façon nette et précise. Ce qui reflète bien la réalité de bien des projets de recherche.

La Figure 1.1 présente les quatre (4) grandes activités du modèle adaptées pour le présent projet. Le cycle de démarrage consiste à déterminer les objectifs et les contraintes initiaux, clarifier les inconnues du projet en effectuant une revue de la littérature puis réaliser une conception préliminaire de la solution. Il s'agit de dégager l'approche générale de solution, d'identifier les modèles prédictifs convenables dans la classification des séquences temporelles et de formuler les résultats attendus. Dès lors, on peut entamer la planification du protocole d'expérimentation qui sera utilisée dans l'évaluation des modèles prédictifs. Dans cette activité, on doit déterminer les paramètres contrôlant les expériences et les jeux de données à appliquer. Le livrable de ce cycle est une maquette conceptuelle présentant le protocole d'expérimentation, les modèles prédictifs retenus pour fin de comparaison et les métriques de performance à utiliser.

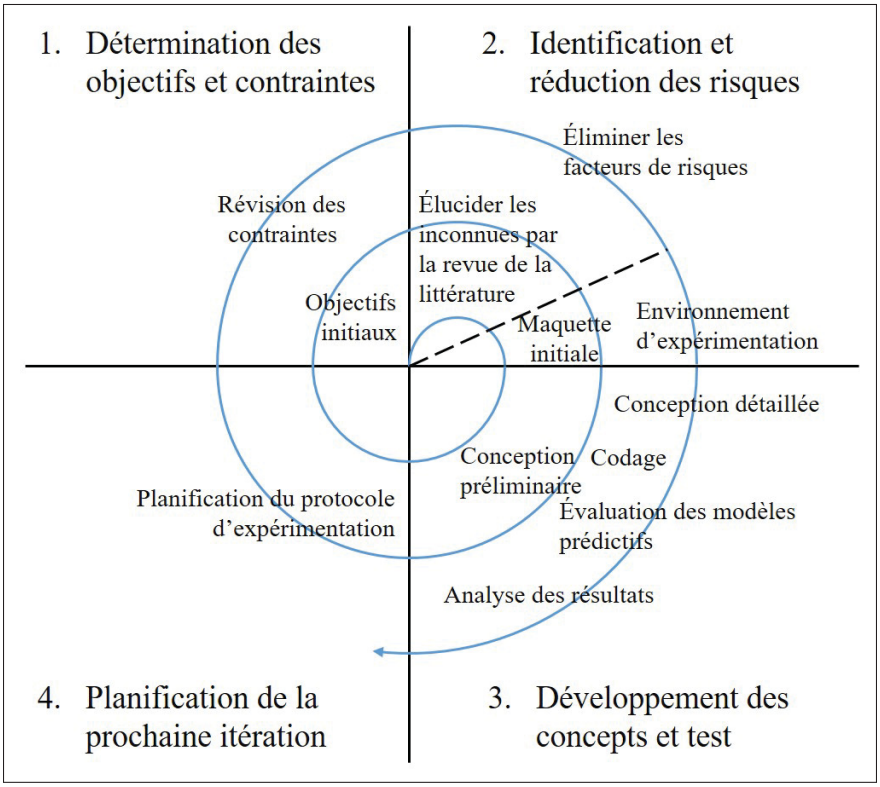


Figure 1.1 Modèle en spiral utilisé dans ce projet de recherche

Au cycle suivant, les contraintes du projet sont révisées et au besoin, des ajustements sont apportés. Quant à la gestion des risques, il s'agit d'éliminer les facteurs de risque ou d'atténuer leurs impacts sur le projet. C'est ainsi que l'on a constaté que le temps d'entraînement est devenu un goulot d'étranglement important dans l'étude des modèles prédictifs. Pour contourner ce problème, les calculs numériques ont été exécutés à l'aide d'une carte graphique Nvidia Tesla V100 afin d'atténuer l'impact de ce facteur de risque sur l'achèvement du projet. Cette carte graphique possède 640 cœurs de tenseur et a une performance théorique de 125 téraflops. Toujours dans ce cycle, la conception détaillée et le codage de l'environnement d'expérimentation sont entamés. Dès lors que l'environnement est devenu stable, on procède à l'évaluation des modèles prédictifs selon le protocole d'expérimentation déjà établi. Le livrable de ce cycle est l'ensemble des constatations et recommandations dégagées par l'analyse des résultats.

1.4 Contributions

Outre que la proposition d'une approche permettant la classification des défauts des moteurs CC, ce projet de recherche vise également à déterminer la capacité des modèles prédictifs profonds à traiter des formes d'ondes multidimensionnelles. L'application primaire de ces modèles se trouve dans le domaine de reconnaissance d'images. Conséquemment, les données multidimensionnelles habituellement traitées par ces modèles sont des pixels organisés en 2D et les canaux de couleur. Les résultats de cette recherche viendront confirmer empiriquement que ces modèles prédictifs sont en mesure de reconnaître des signaux temporels et ce, sans le besoin d'un prétraitement spécifique. De plus, cette recherche identifie d'une façon objective le ou les modèles prédictifs capables de prédire correctement le type de défauts présents dans les signaux, contribuant ainsi à l'avancement des systèmes de maintenance prédictive.

1.5 Conclusion

Les entreprises œuvrant dans un environnement de production flexible peuvent bénéficier des avantages d'une maintenance dite prédictive. Cette dernière est basée sur l'analyse et le suivi de l'état des machines. Elle propose d'exécuter les opérations de maintenance par anticipation.

Pour réaliser ce scénario de maintenance, l'apport de l'Internet des objets est indispensable pour la collecte continue des données en temps réel. Ces données temporelles, constituant l'état d'opération des machines, sont analysées automatiquement par des techniques statistiques et de l'apprentissage profond. L'objectif est de prédire le moment où se produira la panne ou encore prédire le type de défaut qui surviendra.

Dans cette section, nous avons présenté nos objectifs de recherche qui consiste à proposer une approche de classification des défauts pour moteurs CC, à valider la possibilité de prédire les défauts sans l'extraction de caractéristiques et d'analyser la performance des modèles. Nous avons aussi défini notre méthodologie de recherche qui est une adaptation du modèle en spirale.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 Introduction

La revue de l'état de l'art présentée dans ce chapitre concerne la classification des défauts des moteurs à courant continu (CC) à l'aide de l'apprentissage profond. Le choix des moteurs CC comme sujet d'étude est motivé par le fait que l'industrie manufacturière canadienne est en grande partie propulsée par le gaz naturel et l'électricité. Selon Statistiques Canada, le secteur manufacturier canadien a consommé 2 196 pétajoules d'énergie. De cette quantité d'énergie, 30.7% sont de sources gazières et la part de l'électricité s'élève à 29.3% (Statistique Canada).

Les moteurs CC possèdent une riche historique et ont été longuement étudiés. La maîtrise de ces moteurs nous permet de cataloguer les défauts usuels pouvant se produire d'une façon générale :

- défauts au stator résultant d'une ouverture ou d'un court-circuit ;
- connexion anormale des enroulements stator ;
- court-circuit au rotor cassé ;
- bris dans les roulements.

Le comité IEEE *Motor reliabilty working group* a publié en 1985 la répartition des différents défauts pouvant se produire sur une machine électrique (IEEE Motor reliabilty working group, 1985) . Le sommaire de cette répartition est :

- 41% dans les roulements ;
- 37% stator ;
- 10% rotor ;
- 12% autres.

Des proportions similaires ont été rapportées dans une autre étude publiée dix ans plus tard (Thorsen & Dalva, 1995). Tout porte à croire qu'elles sont demeurées sensiblement les mêmes encore aujourd'hui. La détection de ces défauts est intéressante puisqu'il est possible d'identifier l'emplacement du défaut et de déterminer les composants fautifs (Buckley & Fernandez, 2012). Les travaux de recherche actuelle tentent donc de dépasser le cadre de l'analyse à posteriori pour aboutir vers la prédiction des défaut à l'aide de l'apprentissage fond.

2.2 Éléments des modèles prédictifs profonds

Les modèles prédictifs étudiés dans cette recherche sont, pour la plupart, constitués de deux grandes fonctions communes : la détection des caractéristiques et la classification des données. Ces fonctions communes sont réalisées par des éléments de traitement agencés en cascade sous forme de couches. C'est l'organisation et le choix des éléments de traitement qui procurent la spécificité distinctive aux différents modèles prédictifs. La Figure 2.1 est une illustration montrant une architecture générique des modèles étudiés.

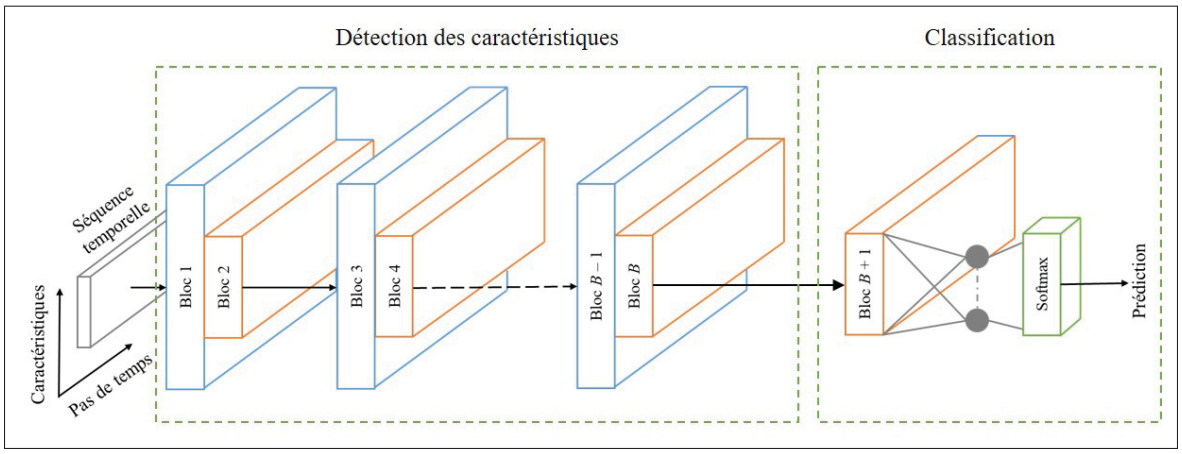


Figure 2.1 Fonctions communes des modèles prédictifs

L'entrée du modèle est un tableau de séquences de données dont l'organisation précise sera présentée plus loin dans la section 4.1.1. L'extraction des caractéristiques est réalisée par des éléments de traitement. Dans la Figure 2.1, ces éléments de traitement sont représentés par des blocs connectés en cascade. Dans un tel modèle, une couche est composée d'un ou plusieurs

éléments de traitement. La dernière couche de la fonction de détection des caractéristiques constitue la sortie de cette fonction commune. Cette sortie est amenée à l'entrée de la classification où d'autres éléments de traitement sont aussi connectés en cascade. Notons que l'opération de classification proprement dite est généralement réalisée par un réseau de neurones pleinement connectés.

Il existe un grand nombre d'éléments de traitement associé aux modèles prédictifs modernes. Voici une description succincte des éléments de traitement que l'on retrouve dans la plupart des modèles présentés dans les sections subséquentes de ce chapitre et dans le chapitre 4 de ce mémoire.

2.3 Convolution et fonction d'activation

Une couche de convolution sert à détecter les « formes » les plus utiles pour la classification. Dans notre cas, ce sont les formes d'ondes contenues dans les séquences temporelles qu'il faut reconnaître. En général, nous avons pour une couche de convolution

$$z_{m,n} = \sum_{i=1}^{I} \sum_{j=1}^{J} w_{i,j} x_{m-i,n-j} \tag{2.1}$$

où $z_{m,n}$ est la valeur à la position (m, n) de la matrice résultante $\mathbf{Z}$, $x_{a,b}$ est la valeur à la position (a, b) de la matrice de données en entrée $\mathbf{X}$ et $w_{i,j}$ est la valeur à la position (i, j) d'une matrice $\mathbf{W}$ de dimension $I \times J$ appelée « filtre ». Par analogie avec un réseau de neurones classique, on peut considérer les valeurs $w_{i,j}$ comme des pondérations associées à des connexions neuronales. À noter que la taille du filtre $\mathbf{W}$ est plus petite que celle de la matrice des données d'entrée $\mathbf{X}$. La dimension habituelle d'un filtre varie de 3×3 à 11×11 (Ahmed & Karim, 2020). On doit donc déplacer $\mathbf{W}$ par rapport aux éléments de $\mathbf{X}$ (ou vice versa) à chaque application des produits scalaires.

Le résultat de l'équation (2.1), $\boldsymbol{Z} = [z_{m,n}]$, est connu sous le nom de carte de caractéristiques (*feature map*) (Alom *et al.*, 2019). Le contenu de $\boldsymbol{Z}$ est censé représenter certaines « formes »

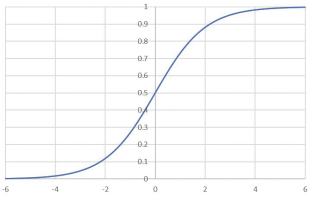
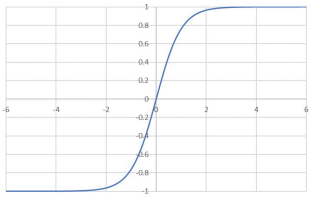
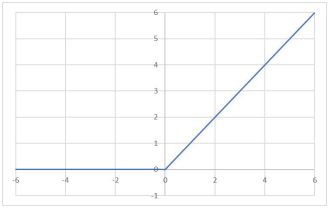
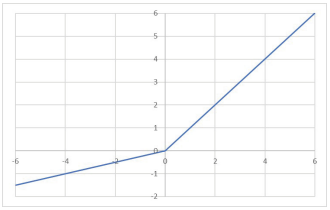
remarquables détectées par la couche de convolution. Or, un seul filtre ne peut capturer l'ensemble des formes remarquables des séquences temporelles d'entraînement. C'est pourquoi on retrouve dans une couche de convolution $n > 1$ filtres et $n \in \{32, 64, 128\}$ sont des valeurs usuelles en pratique. Les valeurs initiales de ces n filtres sont réglées aléatoirement et elles sont ajustées durant l'entraînement du modèle prédictif. Le calcul de l'équation (2.1) est techniquement une corrélation croisée, mais cette différence n'a pas d'impact néfaste puisque le résultat $\mathbf{Z}$ n'est pas utilisé directement dans la classification. En effet, la carte de caractéristiques $\mathbf{Z}$ est transformée non-linéairement par une fonction appelée fonction d'activation (Xu *et al.*, 2015). Puisque la convolution est une opération linéaire, elle ne peut approximer adéquatement les formes non-linéaires des séquences temporelles. L'ajout d'une fonction d'activation vient combler cette lacune et ce, sans alourdir outre mesure les calculs. Ainsi, on peut augmenter l'équation (2.1) en

$$a_{m,n} = \Phi(z_{m,n}) \tag{2.2}$$

où Φ est une fonction d'activation et $\mathbf{A} = [a_{m,n}]$ est la carte de caractéristiques transformées. Une telle fonction doit posséder trois (3) caractéristiques : i) d'être non-linéaire ; ii) dérivable ; iii) facilement calculée. On doit pouvoir dériver la fonction Φ pour le bon fonctionnement de la procédure d'optimisation utilisée dans l'entraînement. Elle doit être simple à calculer puisqu'elle est appliquée sur chacun des éléments de la carte de caractéristiques $\mathbf{Z}$. Plus Φ est complexe plus le temps nécessaire pour obtenir la carte de caractéristiques transformées $\mathbf{A}$ est long. Le Tableau 2.1 donne les fonctions d'activation qui sont utilisées dans les modèles prédictifs présentés dans ce mémoire. Notons que la convolution et la transformation non-linéaire Φ sont souvent positionnés dans les blocs impairs d'un modèle prédictif (voir Figure 2.1).

À noter que la fonction d'activation ReLU n'est pas dérivable au point $x = 0$. Pour contourner ce problème, il suffit de poser la valeur $\Phi(x) = 0$ lorsque $|x| \leq \epsilon$ où ϵ est une petite valeur positive près de zéro (Xu *et al.*, 2015). Quant à la fonction d'activation PReLU, la valeur a est un paramètre à ajuster durant l'entraînement (He *et al.*, 2016). Donc, l'utilisation de PReLU exige une modification dans le processus d'apprentissage pour tenir compte de ce nouveau paramètre.

Tableau 2.1 Fonctions d'activation

Nom	Fonction	Représentation
Sigmoïde	$\Phi(x) = \frac{1}{1-e^{-x}}$	
Tangente hyperbolique	$\Phi(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
Rectified Linear Unit (ReLU)	$\Phi(x) = \max(0, x)$	
Parametric Rectified Linear Unit (PReLU)	$\Phi(x) = \begin{cases} x, & \text{Si } x > 0, \\ ax, & \text{Si } x \leq 0 \end{cases}$	

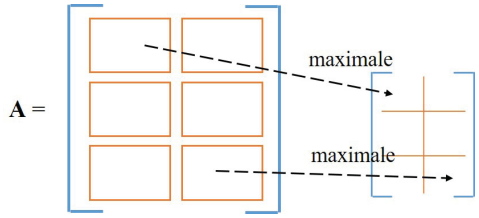
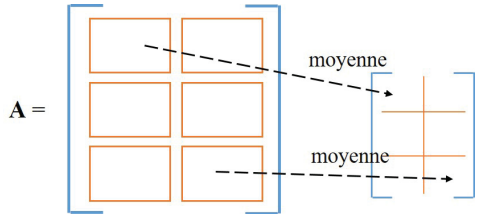
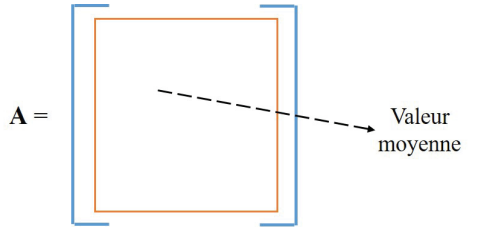
Heureusement, le nombre de calculs ajoutés au processus est tout à fait minime puisqu'il n'y aura qu'un paramètre à ajuster par bloc de PReLU.

2.4 Réduction de la dimension

De base, chaque bloc de convolution et de l'activation est associé à une carte de caractéristiques transformées **A**. Or, comme il peut y avoir $n \in \{32, 64, 128\}$ filtres dans un bloc, il y aura à sa sortie n cartes transformées. Toutes ces cartes sont ensuite traitées par le bloc de convolution et de l'activation en aval et ainsi de suite. Clairement, une telle quantité de données exigerait un nombre très élevé d'opérations arithmétiques. Le *pooling* est une méthode de réduction de dimension tirée du domaine de traitement d'images. Elle consiste à partitionner la carte

de caractéristiques en sous-matrices puis représenter chacune de ces sous-matrices par une seule valeur. La valeur maximale (*max pooling*) et la valeur moyenne (*average pooling*) sont souvent utilisées comme valeur représentative dans le pooling. On peut étendre cette méthode de réduction de la dimensionnalité à la carte de caractéristiques dans son ensemble. Le *Global Average Pooling* (GAP) est une méthode qui représente une carte de caractéristiques par sa valeur moyenne (Lin *et al.*, 2013). Le GAP sert surtout dans la fonction de classification des modèles prédictifs. Le Tableau 2.2 résume ces méthodes de réduction de dimensionnalité.

Tableau 2.2 Méthodes de réduction de dimensionnalité de la carte de caractéristiques

Méthode	**Représentation**
Max pooling	
Average pooling	
Global Average Pooling (GAP)	

À noter que les méthodes Max, Average pooling et le GAP sont souvent implantées dans les blocs pairs de la Figure 2.1 tandis que la méthode GAP est aussi utilisée à la sortie des modèles prédictifs profonds.

2.5 Méthodes de régularisation

Ce sont des méthodes développées pour combattre le surapprentissage où le modèle prédictif est fortement couplé aux données d'entraînement et est incapable de bien performer pour de nouvelles données d'entrée. Les méthodes de régularisation utilisées dans ce projet de recherche sont celles appliquées à des blocs constituant le modèle prédictif. La méthode de normalisation par lot (*Batch Normalization*, BN), proposée par Ioffe & Szegedy (2015), est une méthode très souvent rencontrée. Elle consiste à standardiser un lot de séquences temporelles en les ramenant à une moyenne nulle et un écart-type unitaire. Selon Dauphin & Cubuk (2021), la méthode BN peut réguler la croissance des valeurs $w_{i,j}$ des filtres durant l'entraînement. En effet, plus les valeurs des filtres sont grandes, plus le modèle prédictif est sensible à la variation des données en entrées (Hinton *et al.*, 2012). Le fait de pouvoir réguler l'amplitude des $w_{i,j}$ peut donc aider à contrer le surapprentissage. On peut aussi appliquer la normalisation sur chacune des séquences temporelles. La méthode de normalisation par instance (*Instance Normalization*, IN) ramène chacune des séquences d'entrée à une moyenne nulle et un écart-type unitaire. Ces deux types de normalisation BN et IN agissant sur les données en entrées afin de réguler la croissance des valeurs des filtres.

Une autre méthode couramment utilisée pour réduire le risque de surapprentissage est la méthode de Dropout. Cette dernière agit directement sur les valeurs $a_{m,n}$ des cartes de caractéristiques transformées en les excluant temporairement dans les calculs. À chaque époque de l'entraînement, chacune des valeurs des cartes transformées a une probabilité non nulle d'être ignorée. Cette probabilité peut être ajustée par le processus d'apprentissage ou réglée comme un hyperparamètre du modèle prédictif. L'exclusion temporaire des $a_{m,n}$ dans les calculs est analogue à une reconfiguration des blocs de convolution. Donc, à chaque époque de l'entraînement, on fait l'apprentissage d'un modèle prédictif qui est quelque peu différent de l'époque précédente. C'est par la diversité, même minime, que la méthode Dropout tente de combattre le surapprentissage. Évidemment, on peut utiliser les méthodes BN, IN et Dropout conjointement pour combattre le surapprentissage.

2.6 Sortie du modèle prédictif

Dans la plupart des modèles, la prédiction de l'appartenance des données à une classe est réalisée à l'aide de plusieurs éléments de traitement. D'abord, il y a le pooling et plus précisément le GAP qui transforme chacune des cartes des caractéristiques en une valeur scalaire. Ces valeurs scalaires sont ensuite acheminées à l'entrée d'un réseau de neurones. C'est ce dernier qui produira des valeurs de sorties indiquant l'appartenance des données à une classe. Dans le cas d'une classification à K classes, le réseau de neurones aura K sorties et la règle de décision habituelle est celle de la fonction *Softmax*. Cette fonction redistribue proportionnellement les valeurs d'entrée de sorte que la somme des valeurs à la sortie est égale à 1. L'une des façons réalisant une telle fonction est

$$Q(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}, \quad i = 1, 2, ..., K \tag{2.3}$$

où y_i est l'une des K sorties du réseau de neurones. L'appartenance de la classe est indiquée par $\max(Q(y_i))$, c'est-à-dire, la plus grande valeur parmi les K valeurs de l'équation (2.3). Aussi, le choix de la fonction Softmax comme bloc de sortie simplifie grandement la formulation de la fonction d'objectif du processus d'apprentissage. L'intégration de la fonction Softmax dans l'apprentissage sera présentée brièvement dans la section 4.2.2 du chapitre 4.

2.7 Méthodologie pour la classification des défauts

De façon générale, les tâches nécessaires à la classification des défauts sont (Cinar *et al.*, 2020) :

- cueillette des données ;
- préparation des données ;
- sélection des modèles prédictifs ;
- entraînement des modèles prédictifs ;
- évaluation des modèles prédictifs ;

- ajustement des hyperparamètres des modèles prédictifs ;
- exécution des prédictions.

Une application concrète de ces étapes a été réalisée par Maitre *et al.* (2018) pour le problème de la classification des défauts d'un moteur à induction. Plus spécifiquement, ils s'intéressent aux moteurs triphasés à cage d'écureuil. Leur objectif est de bien classer deux types de défauts : les courts-circuits d'une phase du stator (*Inter-Turn Short Circuit*) et les bris des barres du rotor (*Broken Rotor Bars*) utilisant des données générées par une série de simulations. Ces simulations leur permettent d'obtenir des séquences temporelles contenant la vitesse du moteur, la tension et le courant électrique. De plus, deux types d'alimentation, sinusoïdale et triphasée *Pulse-Width Modulation* (PWM) sont appliqués au moteur. Dans chacune de leur simulation et pour chaque type d'alimentation, ils changent les paramètres de simulation de façon aléatoire. Ils font varier le couple, la vitesse du moteur et l'algorithme de contrôle du couple. Chacune des simulations est échantillonnée à 10 kHz et a une durée de 5 secondes. Avec ces conditions, Maitre *et al.* (2018) créent trois jeux de données. Le premier jeu de données D1 contient 20 simulations d'une durée de 5 secondes pour chaque type de défaut de machine. Ce jeu de données est utilisé pour l'apprentissage des modèles avec 70% pour l'entraînement et 30% pour la validation. Dans un deuxième temps, deux jeux de données D2 et D3 sont générés pour évaluer la performance des modèles. Dans ces deux jeux de données, un bruit blanc est ajouté.

L'objectif est de classifier les différents défauts produits par la simulation et d'identifier la source des défauts dans le moteur. Pour réussir la classification des défauts, Maitre *et al.* (2018) ont utilisé un classifieur hiérarchique (Hierarchical Framework of the Recognition Algorithm, HRA). Ce type de classifieur permet d'améliorer la performance du système lorsqu'il y a un grand nombre de classes à classifier. Ce qui est le cas ici avec 129 classes à reconnaître.

La Figure 2.2 montre l'architecture de ce classifieur. Chacun des nœuds (points noirs) représente un classifieur. Chaque classifieur a une tâche simple et précise : reconnaître s'il y a défaut, déterminer la phase où il y a défaut, déterminer le type de défaut des rotor.

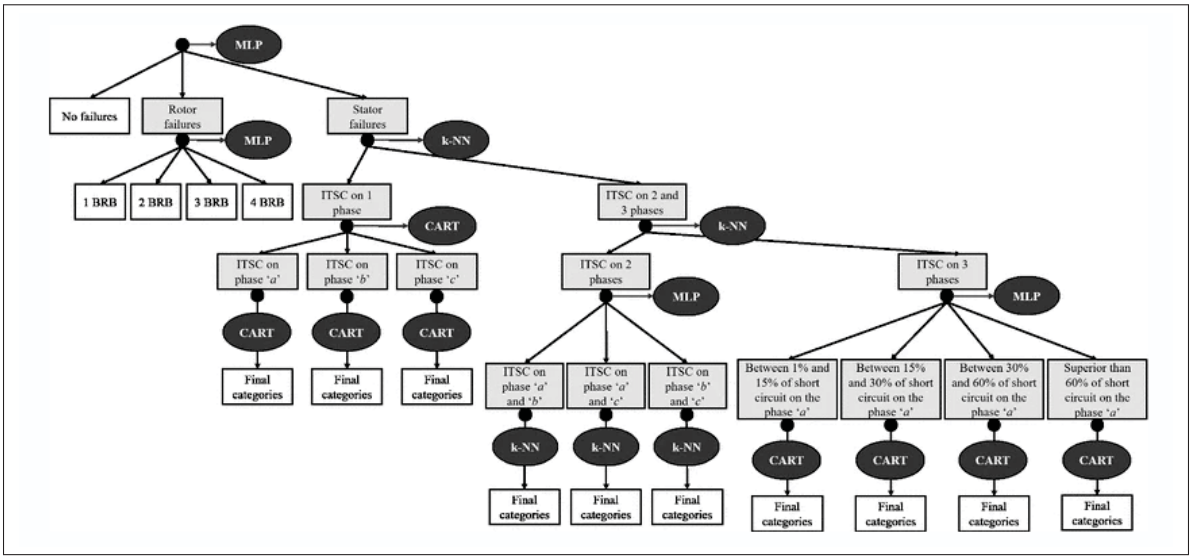


Figure 2.2 Classifieur hiérarchique
Tirée de Maitre *et al.* (2018)

Pour chacun des nœuds, trois types d'algorithmes de classification ont été implantés :

1. Classification and Regression Trees (CART) ;
2. k-Nearest Neighbors (k-NN) ;
3. Multilayer Perceptron (MLP).

Pour analyser la performance de leur classifieur hiérarchique, l'équipe de Maitre *et al.* (2018) ont comparé les résultats de HRA avec ceux d'un simple k-NN. Dans le cas du jeu de données D1, la différence en ce qui concerne le taux de classification n'est pas remarquable entre leur classifieur et le k-NN soit 93.50% pour le k-NN et 92.29% pour le HRA. Cependant, pour le jeu de données D2 et D3, leur classifieur surpasse le k-NN et donne de très bons résultats. Le taux de classification pour le HRA est de 81.26% pour D2 et de 81.57% pour D3 en comparaison avec le k-NN qui a un taux de classification de 70.89% et 71.45% pour les deux jeux de données.

2.8 Classification des séquences temporelles

À l'heure actuelle, l'apprentissage profond connaît un grand succès dans le traitement d'image, de vidéos et des langues. Il est donc naturel de l'étendre dans le domaine des signaux industriels. Un nombre de modèles prédictifs profonds ont été conçus pour la classification des séquences temporelles (Buckley & Fernandez, 2012). Cette abondance relative peut être expliquée par l'engouement des géants numériques du WEB pour le traitement automatique des langues où chaque phrase est considérée comme une séquence temporelle.

C'est dans ce contexte que Ismail Fawaz *et al.* (2019) ont identifié neuf (9) réseaux neuronaux profonds ou modèles prédictifs capables de réaliser la classification des séquences temporelles. Ces modèles sont :

1. Multi Layer Perceptron (MLP) ;
2. Fully Convolutional Neural Network (FCN) ;
3. Residual Network (ResNet) ;
4. Encodeur ;
5. Multi-scale Convolutional Neural Network (MCNN) ;
6. Time Le-Net (t-LeNet) ;
7. Multi Channel Deep Convolutional Neural Network (MCDCNN) ;
8. Time Convolutional Neural Network (Time-CNN) ;
9. Time Warping Invariant Echo State Network (TWIESN).

Pour évaluer la performance de ces architectures neuronales, ils ont entraîné ces neufs (9) modèles prédictifs à l'aide de 97 jeux de données contenant chacun dix (10) séquences temporelles donnant un total 8730 apprentissages. Parmi les 97 jeux de données, 85 sont des séquences monovariables et seulement 12 contenant des séquences temporelles multivariables. L'exactitude des classifications est la métrique de performance choisie dans les expériences effectuées. Le

classement relatif des modèles prédictifs est montré visuellement par le diagramme de différence critique. Le diagramme de différence critique est une méthode proposée par Demšar (2006) pour réaliser la comparaison entre différents modèles prédictifs à l'aide de plusieurs jeux de données. L'axe des abscisses représente la valeur du rang arithmétique moyen pour un modèle j donné. Cette valeur moyenne est calculée par

$$\bar{r_j} = \frac{\sum_{i=1}^{N} r_{i,j}}{N_D} \tag{2.4}$$

où i est l'indice du jeu de données, j est l'indice du modèle prédictif, N_D est le nombre total de jeux de données et $r_{i,j}$ est le rang obtenu par le modèle j sur le jeu de données i. Donc, plus le rang moyen d'un modèle est bas, mieux il performe en moyenne en comparaison avec les autres modèles. Les résultats obtenus par Ismail Fawaz *et al.* (2019) sont présentés dans les Figures 2.3 et 2.4.

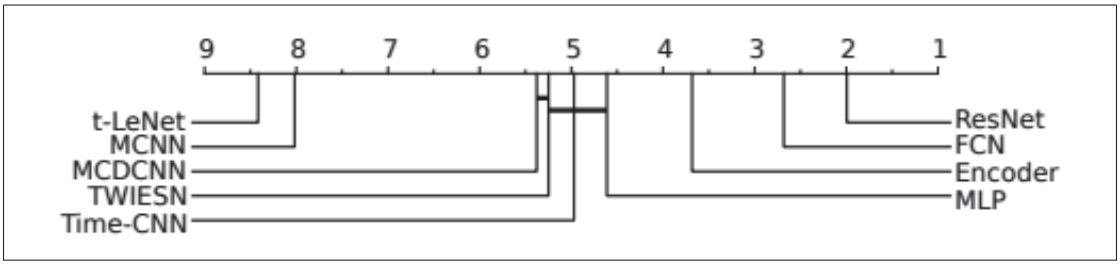


Figure 2.3 Diagramme de différence critique pour les neuf (9) modèles sur des séquences temporelles monovariables
Tirée de Ismail Fawaz *et al.* (2019)

D'après les résultats de la Figure 2.3, les modèles ResNet, FCN et Encoder performent mieux pour des jeux de données monovariable. Il en est de même pour le cas des jeux de données multivariables. Les résultats de la Figure 2.4 montrent que ces trois (3) mêmes modèles offrent de meilleures performances. Les expériences réalisées par Ismail Fawaz *et al.* (2019) nous permettent de cerner les architectures neuronales qui sont efficaces dans la classification des séquences temporelles.

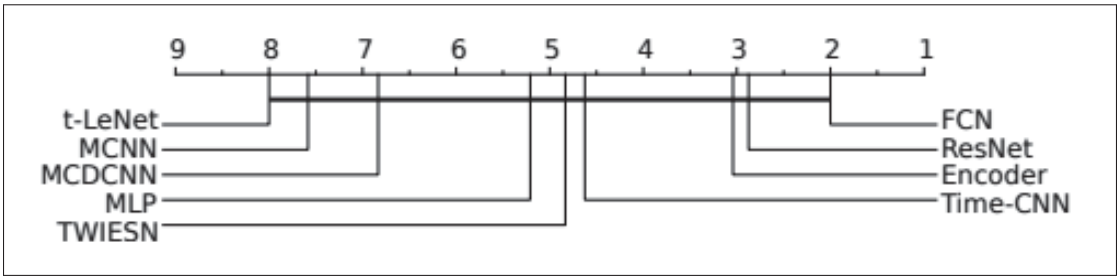


Figure 2.4 Diagramme de différence critique pour les neuf (9) modèles sur des séquences temporelles multivariables
Tirée de Ismail Fawaz *et al.* (2019)

2.9 Classification des défauts

Karim *et al.* (2019) proposent deux modèles d'apprentissage profond pour réaliser la classification de séries temporelles multivariables. Leurs travaux utilisent deux modèles utiles à la classification de séquences temporelles monovariable : le Long Short Term Memory Fully Convolutional Network (LSTM-FCN) et le Attention LSTM-FCN (ALSTM-FCN). Puis, ils adaptent ces modèles afin qu'ils puissent traiter des séquences temporelles multivariables. Ces deux nouveaux modèles proposés sont baptisés : Multivariate LSTM-FCN (MLSTM-FCN) et Multivariate Attention LSTM-FCN (MALSTM-FCN). Leurs deux nouveaux modèles utilisent en entrée des séries temporelles multivariables qui sont des tenseurs de dimension N, Q, M où N est le nombre d'échantillons, Q est le nombre maximal d'incréments à travers toutes les variables et M est le nombre de variables. La Figure 2.5 montre l'architecture du MLSTM-FCN, dans le cas du MALSTM-FCN, les cellules LSTM sont remplacées par des cellules d'attentions. La section de convolutions contient trois blocs de convolutions temporelles et est utilisée pour extraire les caractéristiques de la série temporelle. Chaque bloc de convolutions est composé d'une convolution temporelle, d'une normalisation par lot BN et d'une fonction d'activation de type ReLU. En plus, un bloc squeeze-and-excite est ajouté aux deux premiers blocs de convolutions. C'est cette addition qui différencie le MLSTM-FCN de la version LSTM-FCN originale. Finalement, le dernier bloc de convolutions est suivi par un GAP.

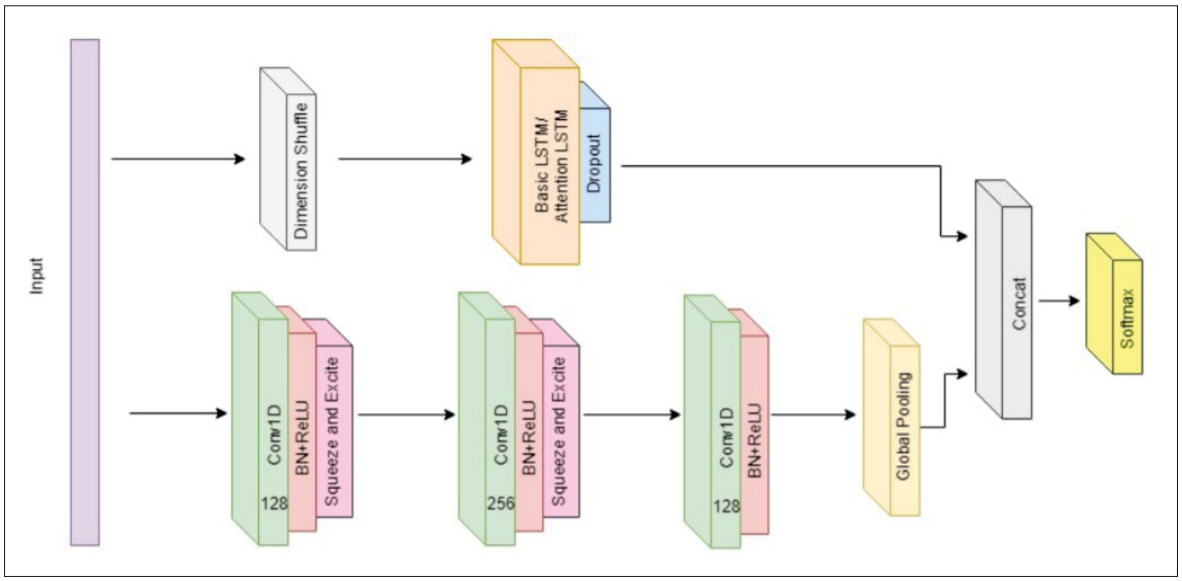


Figure 2.5 Architecture du MLSTM-FCN
Tirée de Karim *et al.* (2019)

La Figure 2.6 montre le fonctionnement du bloc «squeeze-and-excite». Ce bloc permet de calibrer à nouveau les cartes de caractéristiques des convolutions. Selon Karim *et al.* (2019), ce bloc est essentiel pour obtenir une bonne performance de prédiction des séquences temporelles multivariables.

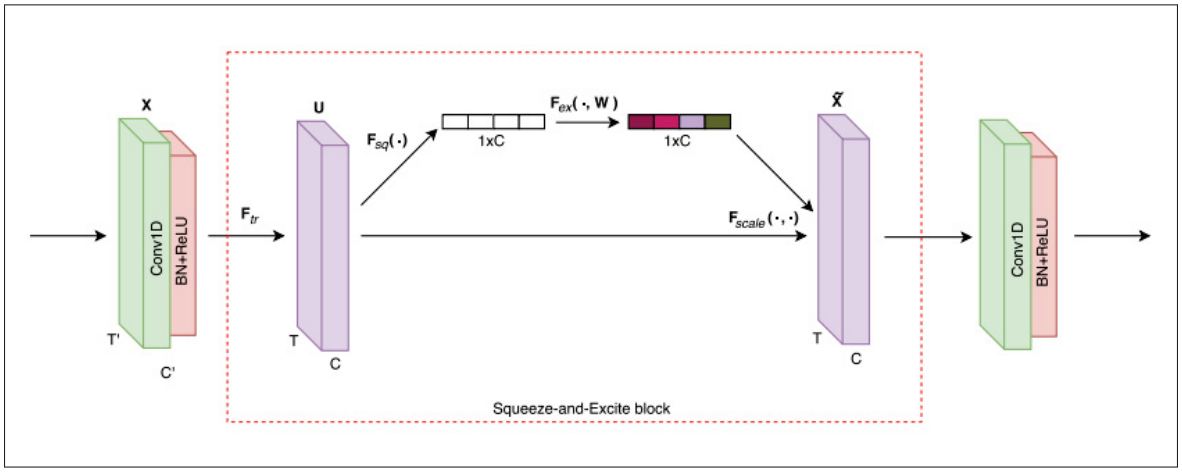


Figure 2.6 Bloc squeeze-and-excite
Tirée de Karim *et al.* (2019)

En parallèle, l'entrée de la série temporelle multivariable passe dans un modificateur de dimension suivi par un bloc de LSTM qui est identique à celui du LSTM-FCN original. Le modificateur de dimension sert à accélérer la vitesse d'entraînement et d'inférence du LSTM en transposant la dimension Q et M lorsque Q est plus grand que M. Pour le MALSTM-FCN, c'est un bloc d'attention qui est utilisé et il est identique à celui du ALSTM-FCN original. Karim *et al.* (2019) ont utilisé 35 jeux de données temporelles multivariables provenant des domaines de la santé, la reconnaissance de la parole et la reconnaissance de mouvements. Tout d'abord, ils ont obtenu un résultat de base pour chacune des jeux de données en testant des modèles simples : *Dynamic Time Warp*, *Random Forest*, *Support Vector Machine* avec noyau linéaire et avec noyau polynomial. Ils ont conservé le modèle qui performe le mieux pour obtenir leurs valeurs de référence sur chacun des jeux de données. Ensuite, ils ont entraîné leurs deux modèles sur ces jeux de données et ils ont comparé leurs performances avec ceux des modèles de base (LSTM-FCN et ALSTM-FCN). De plus, ils ont comparé leurs résultats avec différents modèles qui représentent la référence dans le domaine (HULM, HCRF, NL, FKL, ARKernel, LPS, mv-ARF, SMTS, WEASEL+MUSE, and dUFS).

Dans la majorité des cas, le MLSTM-FCN et le MALSTM-FCN sont beaucoup plus performants que les modèles de référence (28 sur 35 pour le MLSTM-FCN et 27 sur 35 pour le MALSTM-FCN). La figure 2.7 montre le diagramme de différences critiques entre le MLSTM-FCN, le MALSTM-FCN, LSTM-FCN le ALSTM-FCN et trois autres modèles de la littérature qui ont eu de bonnes performances.

Nous pouvons observer que le MLSTM-FCN et le MALSTM-FCN performent mieux en moyenne que tous les autres modèles testés. Donc, la nouvelle approche proposée par Karim *et al.* (2019) est une amélioration dans la classification de séries temporelles multivariables. Ces deux (2) modèles ont atteint des résultats surpassant la référence dans la majorité des cas.

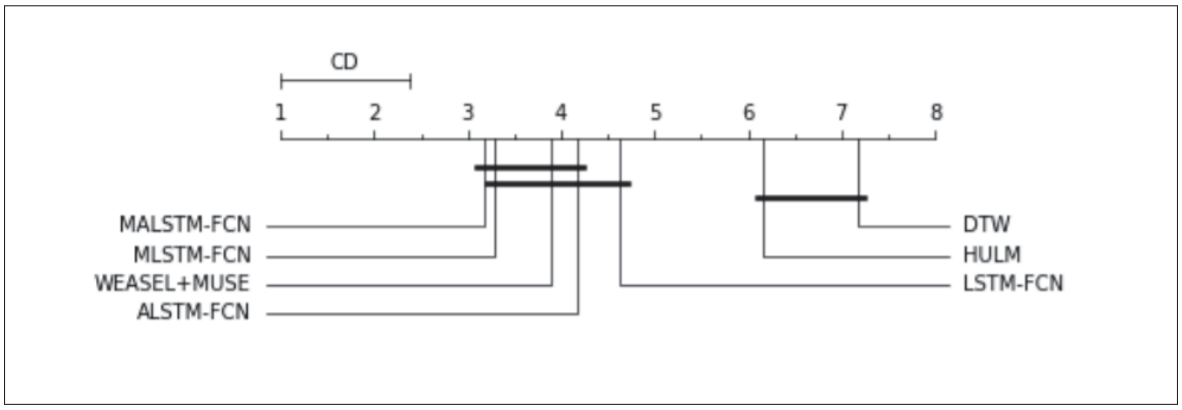


Figure 2.7 Diagramme de différence critique sur les 35 jeux de données
Tirée de Karim *et al.* (2019)

2.10 Conclusion

Les moteurs CC sont les objets d'études dans ce travail de recherche. D'une part, ils font partie des machines électriques les plus répandues dans l'industrie manufacturière. D'autre part, les défauts usuels de fonctionnement de ces moteurs sont bien connus.

Dans ce chapitre, nous avons présenté un survol des concepts en apprentissage automatique et des éléments de l'apprentissage profond. En particulier,

1. Les données d'entrées et de sorties ;
2. Les convolutions ;
3. Les fonctions d'activations ;
4. La réduction de la dimension ;
5. Les méthodes de régularisation.

Également, nous avons étudié la méthodologie employée par l'équipe de Maitre *et al.* (2018). C'est-à-dire entraîner et évaluer les modèles prédictifs à l'aide de signaux générés par la simulation. Cependant, nous apporterons une simplification qui consiste à utiliser uniquement

l'apprentissage profond pour la prédiction des défauts. De plus, au lieu d'avoir un seul moteur à l'essai, nous proposerons une plus grande diversité en utilisant plusieurs moteurs.

Enfin, nous avons passé en revue différents modèles d'apprentissage profond pour la classification de séquences temporelles. Nous allons utiliser trois des modèles qui performaient le mieux de Ismail Fawaz *et al.* (2019) : (1) FCN, (2) ResNet et (3) encodeur et les testés sur notre base de données. De plus, les travaux de Karim *et al.* (2019) nous ont démontré la faisabilité et la performance de certains modèles d'apprentissage profond dans la classification des séquences temporelles multivariables qui est la tâche que nous voulons effectuer.

CHAPITRE 3

GÉNÉRATION DES DONNÉES PAR SIMULATION

3.1 Introduction

Les données sont au centre de la méthodologie de l'apprentissage profond. À l'instar des travaux de Maitre *et al.* (2018), la génération des données par simulation s'avère une approche qui est à la fois efficace et économique. Différents types de défauts peuvent être introduits de façon contrôlée. La variabilité des paramètres et le moment de l'injection des défauts peuvent être simulés avec précision. De plus, certains de ces défauts peuvent entraîner la destruction physique du système à l'étude (voir section 2.1). Cette conséquence est inacceptable puisqu'il est nécessaire d'avoir suffisamment de données pour un nombre de défauts afin de valider la performance des algorithmes de l'apprentissage profond.

Dans ce chapitre, des moteurs CC à aimant permanent (PMDC) sont soumis à des défauts allant de simple bris jusqu'au court-circuit de l'alimentation. Pour mieux illustrer leur utilisation en milieu industriel, ces moteurs sont intégrés dans un système de commande en boucle fermée. De plus, des modulations sont introduites aux paramètres des moteurs afin de simuler la variabilité des procédés de fabrication de ces moteurs.

3.2 Modélisation du moteur à courant continu

Les moteurs électriques à courant continu peuvent être approximés par un système d'équations de premier ordre. La modélisation se décompose en deux parties : électrique et mécanique. Le modèle que nous utiliserons est celui décrit dans Shahgholian & Shafaghi (2010). La Figure 3.1 montre le moteur PMDC utilisé dans la modélisation. Pour simplifier le modèle, nous posons deux suppositions :

1. Le moteur a une efficacité de 100% ;

2. Le champs magnétique généré est constant.

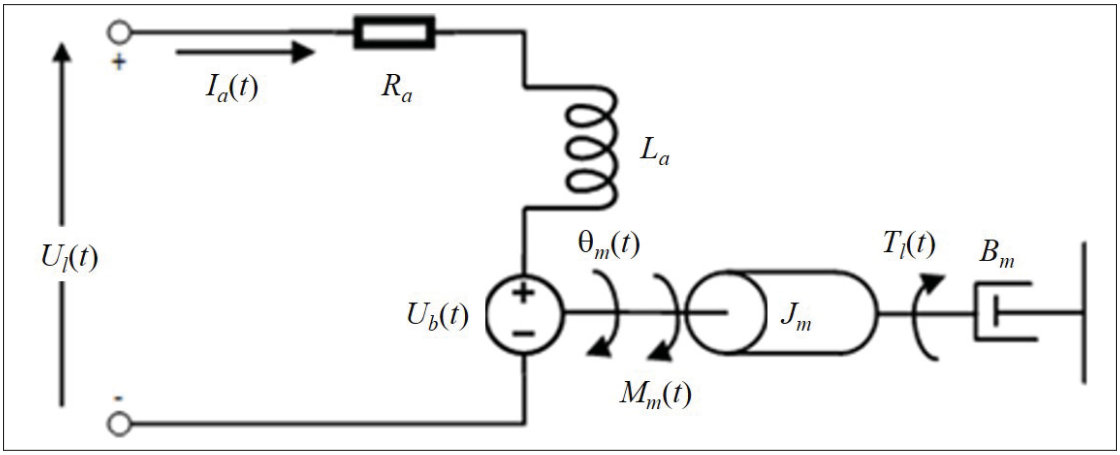


Figure 3.1 Modèle du moteur PMDC

La première supposition nous permet de poser que $K_e = K_t$ où K_e est la constante de force contre-électromotrice et K_t est la constante du couple moteur. Ainsi, dans la suite de ce mémoire nous utiliserons seulement le terme K_t afin de simplifier l'écriture. La deuxième supposition nous permet de considérer la tension $U_b(t)$ comme proportionnelle à la vitesse angulaire ω_m et que le couple du moteur $M_m(t)$ est proportionnel au courant $i_a(t)$. Les équations dynamiques du moteur PMDC sont les suivantes (Shahgholian & Shafaghi, 2010) :

$$\frac{di_a(t)}{dt} = -\frac{R_a}{L_a} i_a(t) - \frac{K_t}{L_a}\omega_m(t) + \frac{1}{L_a} U_t(t) \tag{3.1}$$

$$\frac{d\omega_m(t)}{dt} = \frac{K_t}{J_m} i_a(t) - \frac{B_m}{J_m}\omega_m(t) - \frac{1}{J_m} T_l(t) \tag{3.2}$$

$$\frac{d\theta_m(t)}{dt} = \omega_m(t) \tag{3.3}$$

où i_a est le courant du moteur, R_a est la résistance équivalente interne, L_a est l'inductance équivalente interne, $\omega_m(t)$ est la vitesse de rotation, $U_t(t)$ est la tension appliquée aux bornes du moteur, J_m est l'inertie du rotor, B_m est le coefficient de friction visqueux, $T_l(t)$ est le couple produit par la charge et $\theta_m(t)$ est la position angulaire.

Dans le but d'évaluer la capacité de généralisation des modèles d'apprentissage profond, trois (3) moteurs PMDC sont implantés. Ainsi nous obtenons une plus grande diversité des données lors du processus d'entraînement et de test. Ces ensembles de valeurs sont tirés de Shahgholian & Shafaghi (2010) et sont présentés dans le Tableau 3.1.

Tableau 3.1 Paramètres des moteurs étudiés

Paramètre	**Moteur 1**	**Moteur 2**	**Moteur 3**	**Tolérance**	**Unité**
B_m	$4.32\text{x}10^{-4}$	$1\text{x}10^{-5}$	$3\text{x}10^{-3}$	-	N.M.s/rad
L_a	8.05	5.35	162.73	5%	mH
R_a	1.4	3.2	7.72	7.5%	Ω
K_t	0.095	0.3	1.25	10%	N.m
J_m	$7.49\text{x}10^{-4}$	$5\text{x}10^{-4}$	$2.36\text{x}10^{-2}$	-	Kg.m^2

La colonne «Tolérance» du Tableau 3.1 indique l'étalement du paramètre correspondant. Cette dispersion sert à représenter les imprécisions de fabrication du bobinage des moteurs. Lors des simulations, les paramètres du Tableau 3.1 varieront selon une distribution normale dont la moyenne est la valeur des paramètres et l'écart-type est la valeur de la tolérance. La Figure 3.2 et la Figure 3.3 montrent la vitesse et le courant de chacun des moteurs sans charge. Ces figures montrent bien que ces moteurs ont des comportements temporels bien distincts.

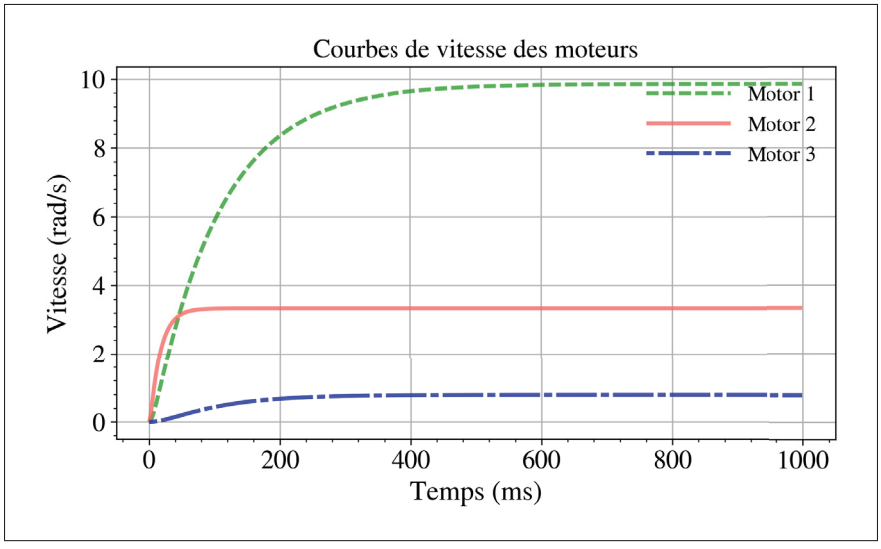


Figure 3.2 Courbes de vitesse des moteurs avec échelon unitaire

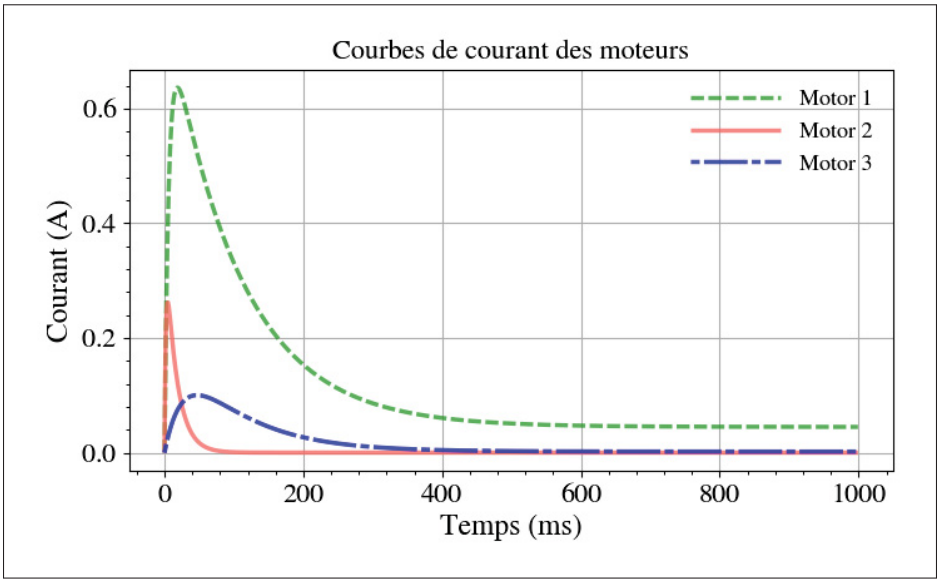


Figure 3.3 Courbes de courant des moteurs avec échelon unitaire

Le modèle de ces moteurs électriques est réalisé dans Simulink en utilisant la représentation d'état. Cette technique de modélisation nous permet de faire varier les variables d'état interne et d'y injecter des défauts programmés au modèle. La forme générale d'un modèle par représentation d'état linéaire temps invariant est donnée par l'équation suivante :

$$x(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t), \quad y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t) \tag{3.4}$$

où $x(t) \in \mathbb{R}^n$ est le vecteur des variables d'état, $u(t) \in \mathbb{R}^m$ est le vecteur des variables de contrôle, $y(t) \in \mathbb{R}^p$ est le vecteur des variables de sortie, $\boldsymbol{A} \in \mathbb{R}^{nxn}$ est la matrice d'état, $\boldsymbol{B} \in \mathbb{R}^{nxm}$ est la matrice de contrôle, $\boldsymbol{C} \in \mathbb{R}^{pxn}$ est la matrice d'observation et $\boldsymbol{D} \in \mathbb{R}^{pxm}$ est la matrice d'action directe.

Ainsi, nous pouvons réécrire les équations 3.1 à 3.3 en représentation d'état. Le modèle comporte ainsi deux (2) variables d'entrées soient, la tension U_t et le couple T_l ainsi que deux (2) variables de sorties soient le courant i_a et la vitesse angulaire ω_m. Le modèle est représenté par les

équations 3.5.

$$\boldsymbol{x} = \begin{bmatrix} i_a & \omega_n \end{bmatrix}^\intercal \tag{3.5a}$$

$$\boldsymbol{y} = \begin{bmatrix} i_a & \omega_n \end{bmatrix}^\intercal \tag{3.5b}$$

$$\boldsymbol{u} = \begin{bmatrix} U_t & T_l \end{bmatrix}^\intercal \tag{3.5c}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_t}{L_a} \\ \frac{K_t}{J_m} & -\frac{B_m}{J_m} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J_m} \end{bmatrix} \boldsymbol{u} \tag{3.5d}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{u} \tag{3.5e}$$

À partir de ces équations, nous obtenons facilement les matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ et $\boldsymbol{D}$ caractérisant un moteur PMDC.

$$\begin{gathered} \boldsymbol{A} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_t}{L_a} \\ \frac{K_t}{J_m} & -\frac{B_m}{J_m} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J_m} \end{bmatrix}, \\ \boldsymbol{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{gathered} \tag{3.6}$$

La stabilité des moteurs du Tableau 3.1 peut être déterminée à partir du modèle par représentation d'état. En effet, le positionnement des pôles de ces moteurs est obtenu par les valeurs propres de $\boldsymbol{A}$. Pour qu'un système linéaire invariant soit stable, la partie réelle de tous les pôles doit être négative (Akhrif, 2019). L'équation (3.7) montre que les valeurs propres des moteurs sont toutes négatives. Ainsi, les trois (3) systèmes sont asymptotiquement stables en boucle ouverte selon le critère « entrée bornée/sortie bornée ».

$$\lambda[\boldsymbol{A}_{m1}] = \begin{bmatrix} -164.7984 \\ -9.6914 \end{bmatrix}, \ \lambda[\boldsymbol{A}_{m2}] = \begin{bmatrix} -535.2730 \\ -62.8779 \end{bmatrix}, \ \lambda[\boldsymbol{A}_{m3}] = \begin{bmatrix} -36.1444 \\ -11.4232 \end{bmatrix}. \tag{3.7}$$

3.3 Système de commande

Dans les applications de convoyage, de transfert et de levage, les moteurs sont rarement utilisés en boucle ouverte. C'est pourquoi il est opportun d'appliquer un système de commande aux moteurs modélisés. La Figure 3.4 montre un système muni d'un compensateur Proportionnel, Intégral, Dérivé (PID) classique, une consigne d'entrée et une rétroaction.

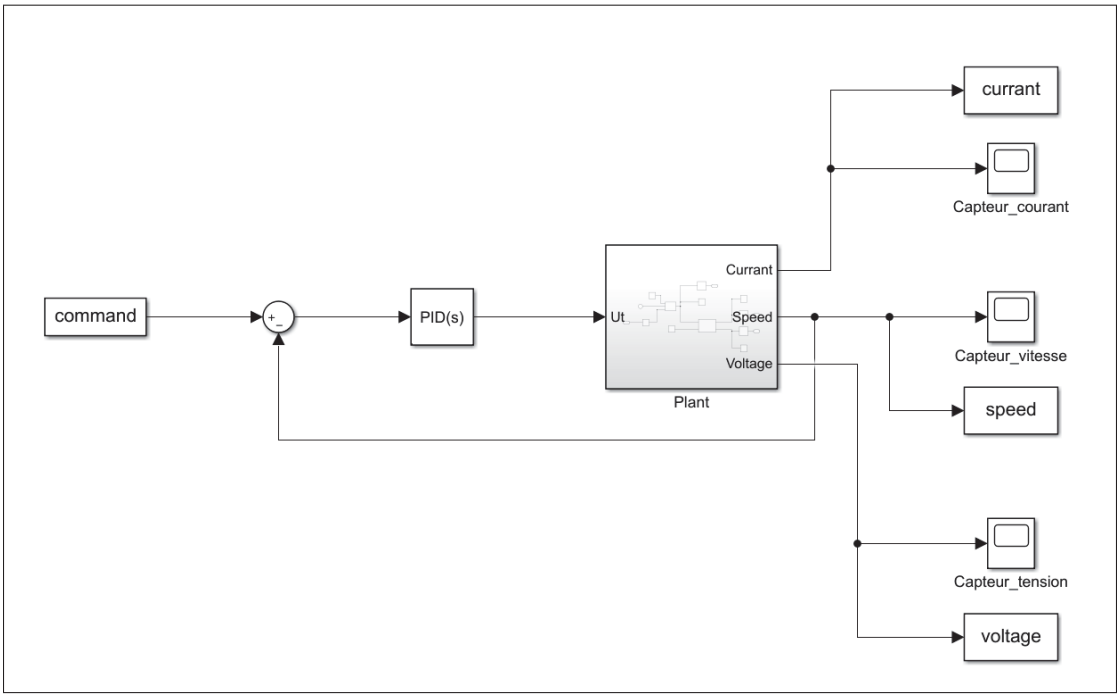
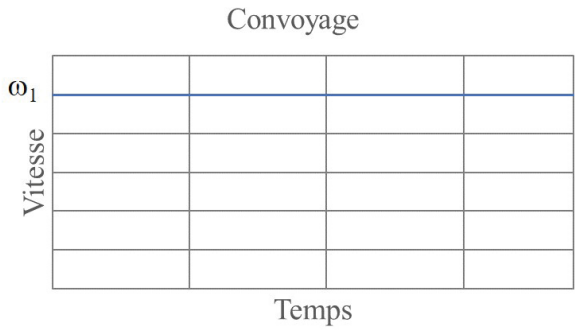
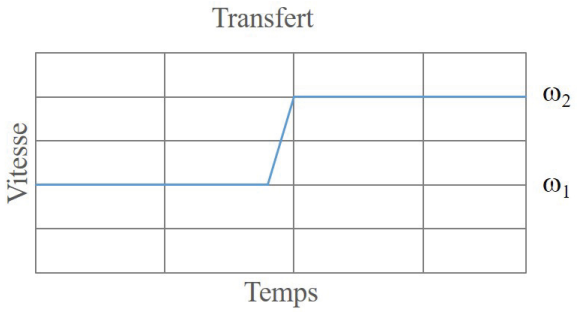
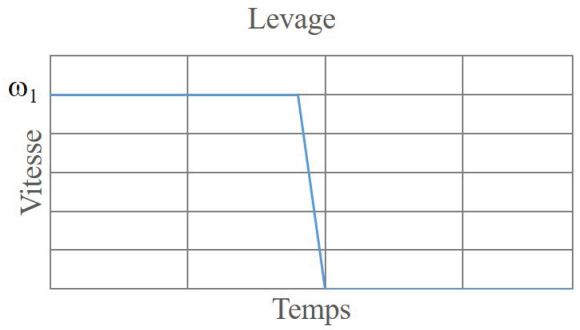


Figure 3.4 Système de commande et capteurs

Dans ce système, la rétroaction provient du capteur de vitesse installé à la sortie du moteur. La consigne de vitesse sert à simuler différentes applications des moteurs dans une chaîne de production. Pour ce faire, trois (3) profils de vitesses sont appliqués. Ces consignes, montrées dans le Tableau 3.2, représentent le convoyage, le transfert d'un convoyeur à l'autre et le levage des pièces d'assemblage.

La valeur des vitesses ω_1 et ω_2 ainsi que le moment d'enclenchement des changements de vitesse sont appliquées de façon aléatoire selon une distribution uniforme lors de la génération des données.

Tableau 3.2 Profils de vitesse appliquée aux moteurs

Profil	Utilisation	Représentation
1	Convoyage : Maintien du moteur à une vitesse ω_1 pour le fonctionnement continu.	
2	Transfert : Démarrage du moteur à une vitesse ω_1 puis on l'amène à une vitesse ω_2.	
3	Levage : Démarrage du moteur à une vitesse ω_1 puis on arrête le moteur.	

Le schéma bloc de la Figure 3.5 montre la composition d'un compensateur PID. Sa fonction de transfert est donnée par l'équation (3.8) où $e(t)$ est l'erreur calculée, $u(t)$ est le signal de sortie, K_p, K_d et K_i sont les coefficients des gains proportionnel, dérivé et intégral respectivement (en.wikibooks.org, 2019),

$$u(t) = K_p e(t) + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} + K_i \int_0^t e(t)\,\mathrm{d}t \tag{3.8}$$

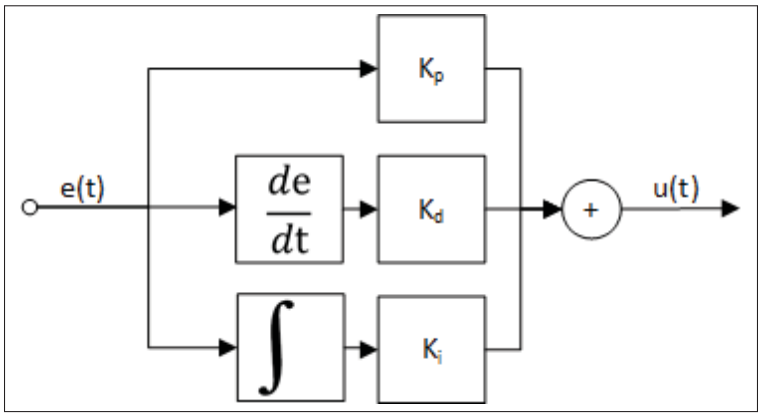


Figure 3.5 Schéma bloc d'un contrôlleur PID

Notons que la fonction `pidtune` de MATLAB est utilisée pour le calcul des gains du compensateur PID. Enfin, pour réaliser l'acquisition des signaux, nous avons utilisé plusieurs capteurs : un capteur de courant (A), un capteur de vitesse (rad/s) et un capteur de tension (V) chacun avec un gain unitaire. L'emplacement de ces capteurs est montré dans la Figure 3.4.

3.4 Simulation des défauts

Comme mentionné dans la section 2.1 du chapitre 2, plusieurs types de défauts sont couramment rencontrés et intrinsèques aux moteurs électriques. Dans un environnement de production industrielle, les moteurs ainsi que le système de commande sont instrumentés afin d'y effectuer l'analyse continue. De ce fait, les instruments de mesure peuvent aussi être sujets à des défauts. Ainsi, la prédiction des défauts doit aussi tenir compte de la possibilité de bris des capteurs (Santos *et al.*, 2018). Le Tableau 3.3 identifie les défauts appliqués et leur conséquence sur le système.

Les Figures 3.6 à 3.10 illustrent ces 5 défauts pour le moteur 1 du Tableau 3.3. La consigne de vitesse à l'entrée de ce moteur est un échelon unitaire et la charge à sa sortie est nulle. Aussi, les simulations sont faites en boucle ouverte. Les défauts sont enclenchés au temps $T_{bris} = 1$ seconde.

Tableau 3.3 Types de défauts et les paramètres associés

Défaut	Type	Paramètre affecté
1	Problème de roulement	Augmentation B_m
2	Défaut du capteur de vitesse	Lecture $\omega_m = 0$
3	Défaut du capteur de courant	Lecture $i_a = 0$
4	Défaut du capteur de tension	Lecture $U_t = 0$
5	Déconnexion/court-circuit dans l'alimentation	$U_t = 0$

3.5 Génération des défauts

Dans tous les cas, le bris arrive subitement durant le fonctionnement des moteurs. Le temps auquel survient le bris est défini par T_{bris}. Cette valeur est générée de façon aléatoire dans l'intervalle $[0, T_{sim} - 1]$ où T_{sim} est le temps total de la simulation. Donc, le bris peut se produire au tout début de la simulation et au plus tard à une seconde de la fin de la simulation. Cette dernière contrainte permet aux signaux de se stabiliser et d'avoir des formes reconnaissables.

Le bris du roulement du rotor (défaut 1 du Tableau 3.3) peut être causé par une usure du moteur, une surchauffe du roulement ou un coup reçu sur l'arbre de rotation. Ce type de bris augmente la valeur du coefficient de friction B_m ce qui réduit l'efficacité du moteur. Donc, pour simuler cette condition, nous augmenterons de 25% la valeur de B_m. L'équation (3.9) définit la fonction non-linéaire utilisée pour simuler ce bris :

$$B_m(t) = \begin{cases} B_m(t) & \text{Si } t \leq T_{bris} \\ B_m(t) * 1.25 & \text{Si } t > T_{bris} \end{cases} \tag{3.9}$$

Dans le cas des trois capteurs (défauts 2 à 4 du Tableau 3.3), les bris occasionnés sont : une défaillance complète du capteur ou un problème de la transmission du signal (communication coupée). Dans les deux cas, le système reçoit la valeur 0 du capteur. L'équation 3.10 montre la fonction non-linéaire utilisée dans la simulation pour la défaillance des capteurs où $c(t)$ est la

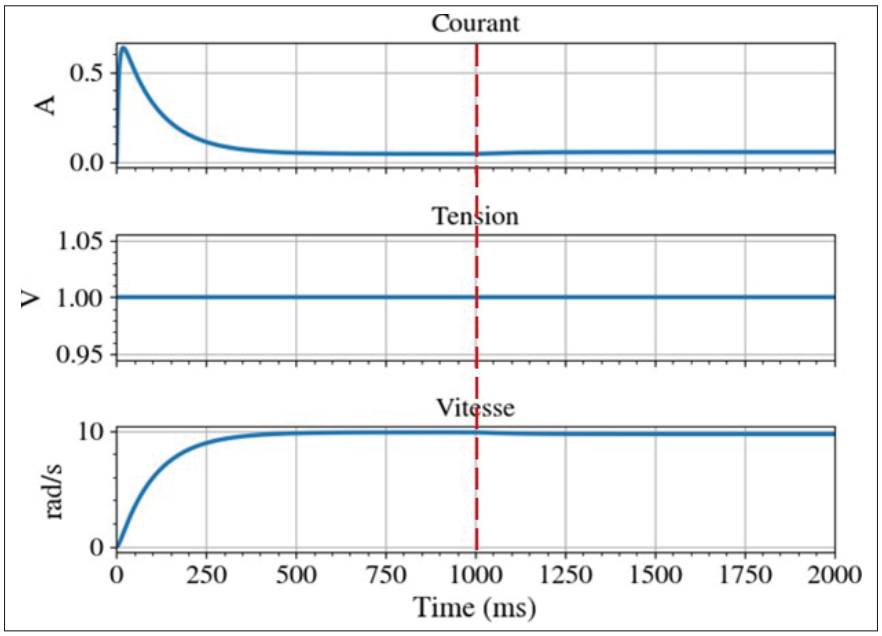


Figure 3.6 Lecture des capteurs du moteur 1 avec problème de roulement

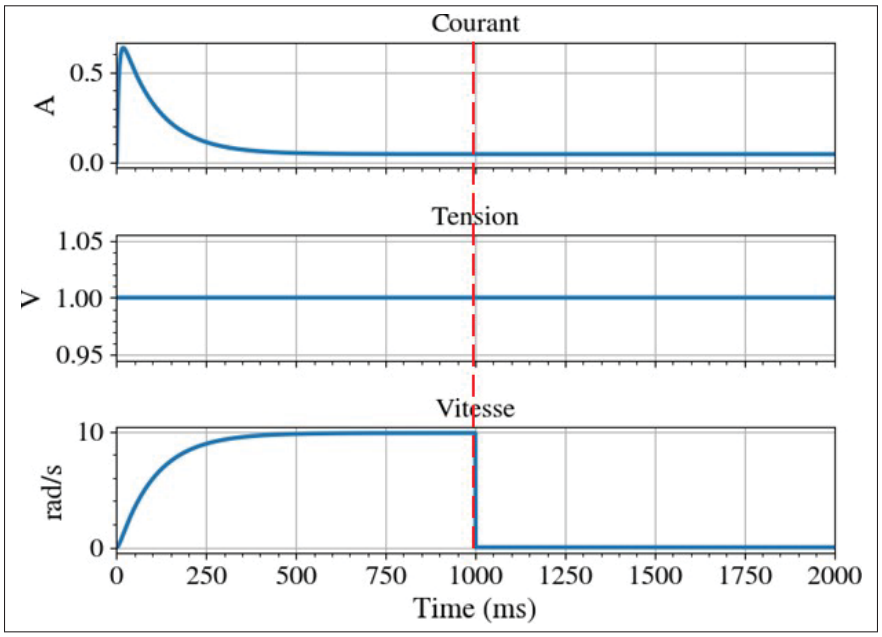


Figure 3.7 Lecture des capteurs du moteur 1 avec défaut du capteur de vitesse

valeur lue du capteur,

$$c(t) = \begin{cases} c(t) & \text{Si } t \leq T_{bris} \\ 0 & \text{Si } t > T_{bris} \end{cases} \tag{3.10}$$

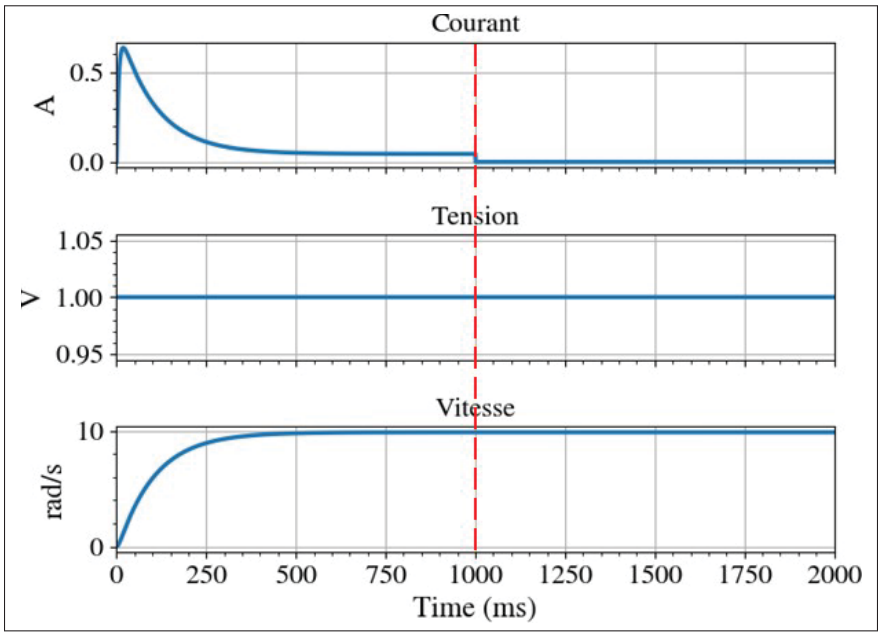


Figure 3.8 Lecture des capteurs du moteur 1 avec défaut du capteur de courant

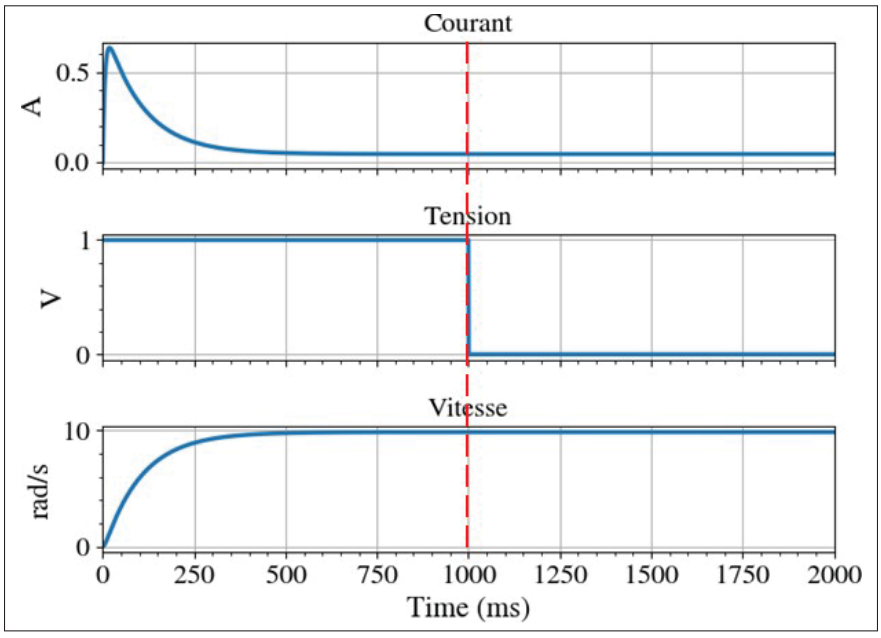


Figure 3.9 Lecture des capteurs du moteur 1 avec défaut du capteur de tension

Pour le court-circuit de l'alimentation, ce défaut peut être engendrée par un bris du bloc d'alimentation ou un court-circuit entre le rotor et le stator du moteur. Pour simuler ce défaut,

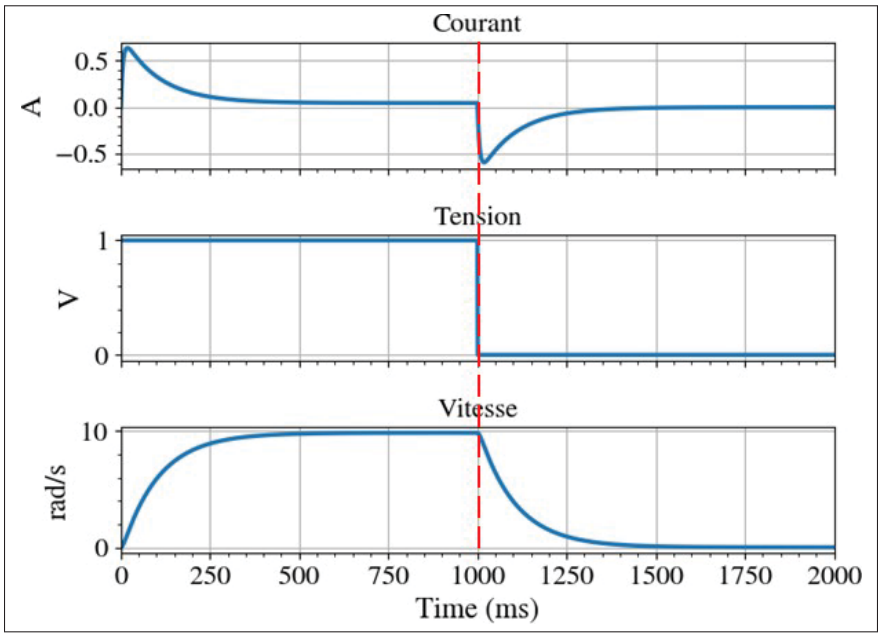


Figure 3.10 Lecture des capteurs du moteur 1 avec court-circuit de l'alimentation

nous assignerons la valeur 0 à U_t (la tension de sortie du bloc d'alimentation). L'équation (3.11) définit la fonction non-linéaire utilisée pour simuler ce bris :

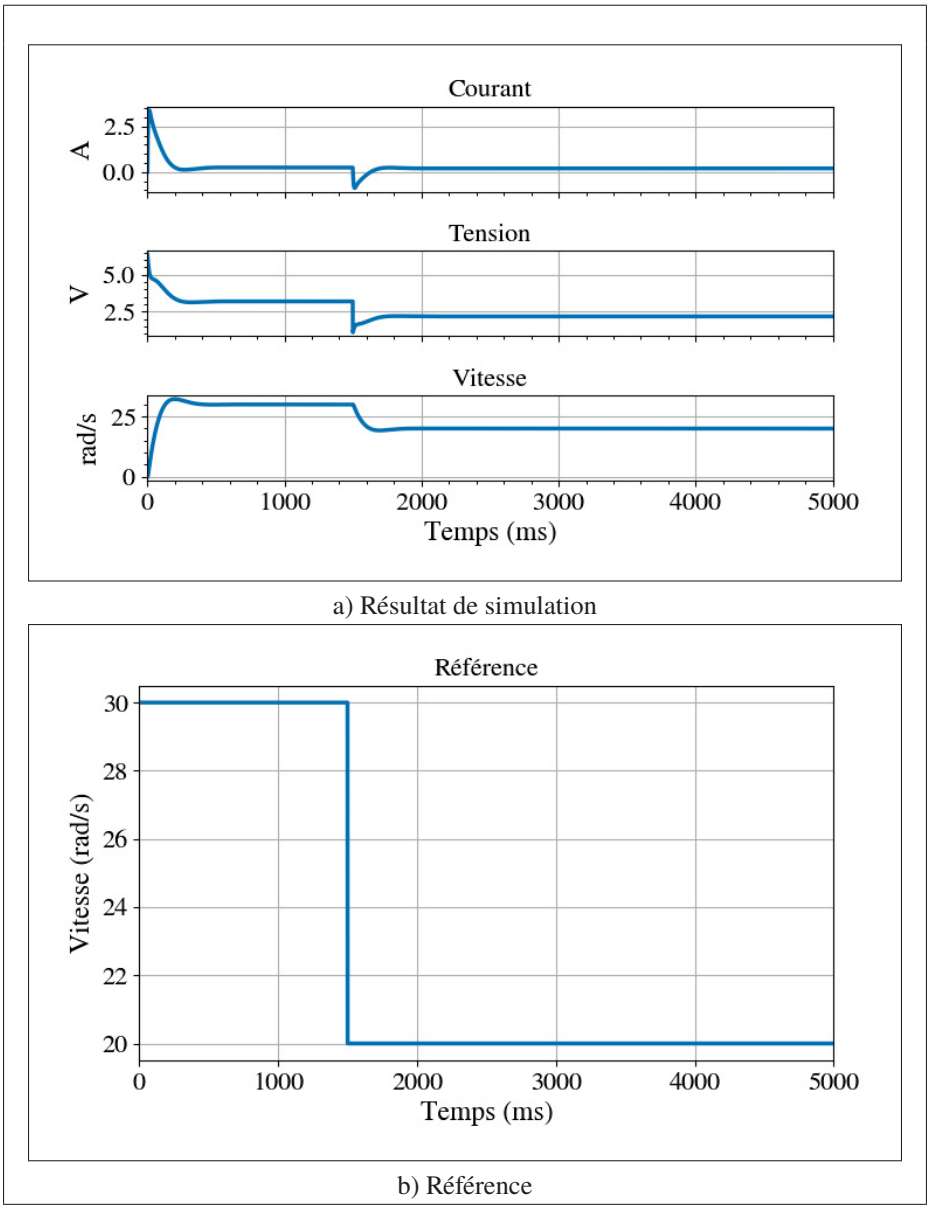
$$U_t(t) = \begin{cases} U_t(t) & \text{Si } t \leq T_{bris} \\ 0 & \text{Si } t > T_{bris} \end{cases} \tag{3.11}$$

3.6 Génération des signaux temporels

Le modèle de simulation d'un moteur PMDC et la boucle de commande ont été réalisés dans Simulink de MATLAB. Un script MATLAB détermine le type de défaut et assigne une valeur aléatoire au temps de déclenchement du défaut T_{bris}. Le script MATLAB exécute aussi la simulation avec les paramètres déterminés et enregistre les données reçues des différents capteurs. La fréquence d'échantillonnage utilisée est de 1 kHz et les signaux sont échantillonnés durant 5 secondes pour former des séquences temporelles.

Les Figures 3.11 et 3.12 présentent un exemple de simulation comprenant la boucle de commande et les capteurs. La première figure montre le moteur 1 simulé sans défaut et la deuxième figure montre le moteur 2 simulé avec défaut du capteur de vitesse qui se produit à $T_{bris} = 3$ secondes. Chacune des figures montre aussi la commande d'entrée. Dans les deux cas, nous pouvons remarquer que le régulateur PID est efficace puisque les deux moteurs suivent bien la consigne de vitesse. Aussi, dans la Figure 3.12 nous pouvons voir le défaut du capteur de vitesse et l'effet sur les signaux.

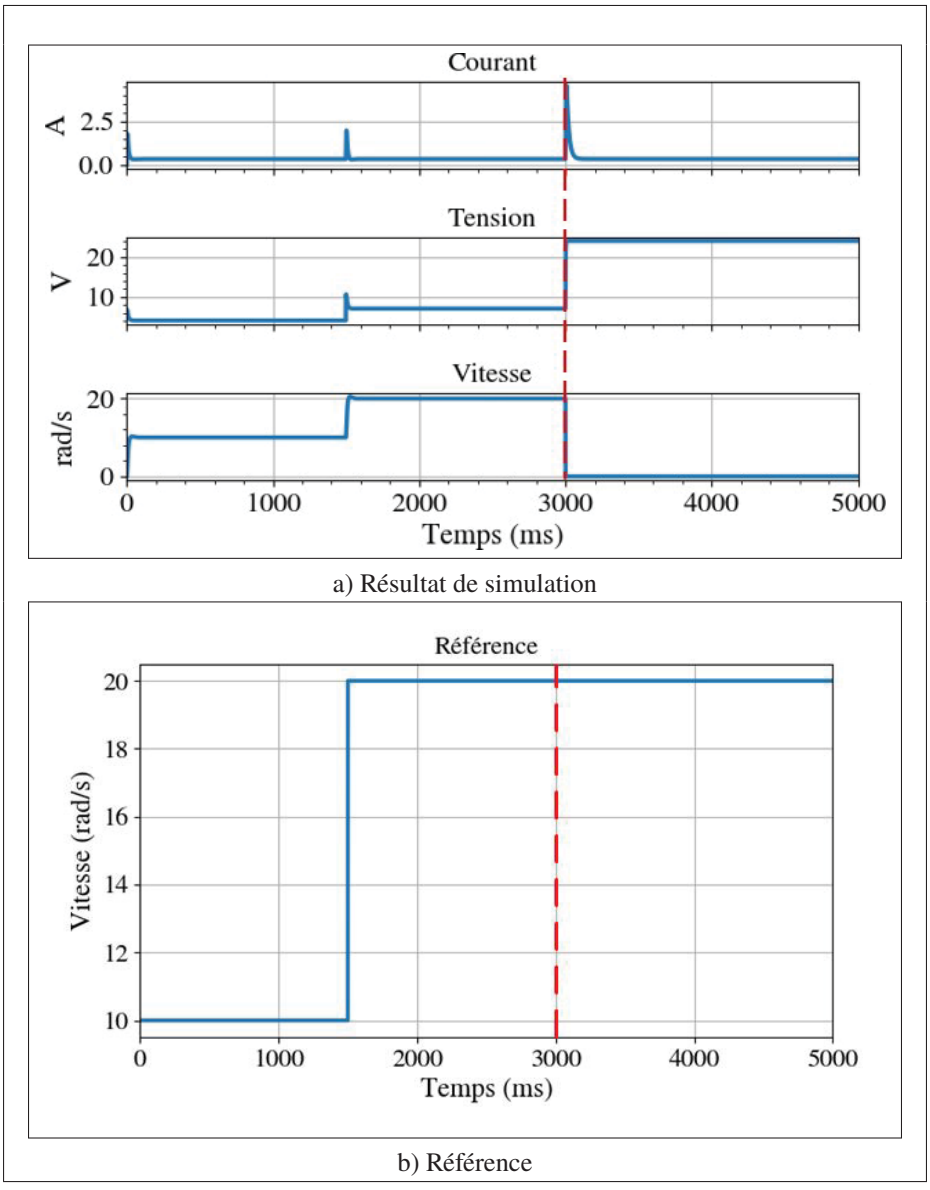
C'est ainsi qu'un grand nombre de séquences temporelles ont été générées afin de former les jeux de données nécessaires pour l'entraînement et le test des modèles prédictifs. Le contenu exact de ces séquences temporelles, le processus d'entraînement et d'évaluation seront présentés en détail dans le prochain chapitre.



a) Résultat de simulation

b) Référence

Figure 3.11 Simulation et référence du moteur 1 aucun défaut



a) Résultat de simulation

b) Référence

Figure 3.12 Simulation et référence du moteur 2 avec défaut au capteur de vitesse

3.7 Conclusion

Les données nécessaires à la prédiction des défauts ont été générées à l'aide d'un modèle numérique Simulink du moteur PMDC intégré dans une boucle de commande PID. Trois profils de vitesse sont appliqués au système suivant une distribution uniforme afin de simuler des applications courantes tels le convoyage, le transfert et le levage. Aussi, trois (3) moteurs PMDC différents sont utilisés. L'inclusion de ces moteurs sert à procurer une plus grande diversité parmi les données générées. Cinq types de défauts sont ensuite appliqués séparément — un défaut par séquence temporelle. Le moment du défaut ainsi que son type sont sélectionnés aléatoirement. L'ensemble de ces facteurs, à savoir les profils de vitesse, l'étalement des paramètres des moteurs, le moment et le type de défaut fait en sorte que la reconnaissance des défauts ne sera pas une tâche triviale. Enfin, le code pour le code MATLAB et le modèle Simulink sont disponibles sur GitHub à l'adresse suivante : https://github.com/alexchartrand/DCMotor-fault-simulation.

CHAPITRE 4

CLASSIFICATION DES DÉFAUTS

4.1 Introduction

Les trois (3) moteurs PMDC modélisés au Chapitre 3 sont soumis à des défauts programmés. Les signaux générés sont enregistrés et formatés en séquences temporelles. Ces séquences sont utilisées directement dans la prédiction des défauts sans la phase de l'extraction des caractéristiques.

Dans ce chapitre, la classification est une opération qui consiste à attribuer des étiquettes $\mathbf{y}$ à des données $\mathbf{X}$ avec l'aide d'un modèle f et de ses paramètres θ. On peut exprimer la classification à l'aide d'une relation d'application mathématique

$$\boldsymbol{y} = f(\theta, \mathbf{X}) \tag{4.1}$$

où la tâche est de déterminer f et θ afin de minimiser l'erreur d'étiquetage. En d'autres mots, le modèle f reçoit en entrée des données $\mathbf{X}$ appelées caractéristiques et doit produire une sortie identifiant l'appartenance de $\mathbf{X}$ à l'une des valeurs $y \in \mathbf{y}$. Il existe plusieurs techniques et modèles prédictifs pour résoudre le problème de classification.

Nous analyserons la performance des modèles d'apprentissage profond FCN, ResNet, Encodeur et LSTM appliqués au problème de la classification des défauts. Le choix de ces modèles est motivé par les travaux de Ismail Fawaz *et al.* (2019) et de Karim *et al.* (2019) présentés dans le Chapitre 2.

4.1.1 Représentation des données temporelles

Une séquence temporelle peut être unidimensionnelle ou multidimensionnelle. Dans le cas d'une séquence unidimensionnelle, elle comporte une seule valeur à chaque pas de temps,

$\mathbf{x} = [x_1, x_2, ..., x_N]^\intercal$ où $\mathbf{x} \in \mathbb{R}^N$ est une séquence de N valeurs. Dans le cas d'une séquence multidimensionnelle, on observe m valeurs à chaque pas de temps et l'on aura $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]^\intercal$ où $\mathbf{X} \in \mathbb{R}^{m \times N}$. La Figure 4.1 est la conceptualisation des séquences temporelles comportant plusieurs dimensions.

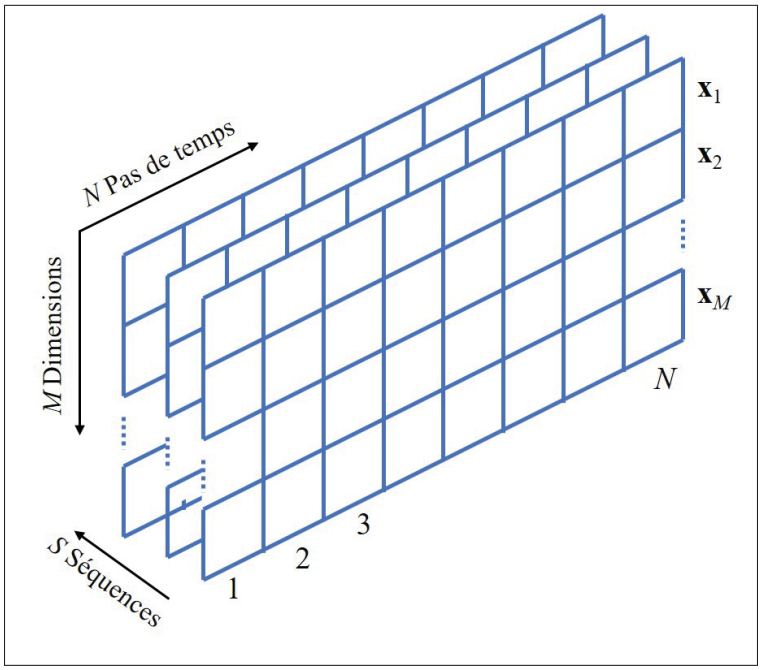


Figure 4.1 Représentation des séquences temporelles multidimensionnelles

L'organisation des valeurs de la Figure 4.1 est un tableau 3D dans lequel chaque colonne est un pas de temps et chaque ligne est une caractéristique. Les séquences temporelles sont empilées selon la 3e dimension du tableau. L'ensemble des caractéristiques d'une colonne forme une observation au pas de temps associé. À l'aide de cette représentation, un jeu de données D peut être défini comme une collection de couples $D = \{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \cdots, \mathbf{X}_N, y_N)\}$ où $\mathbf{X}_i$ est la i^e séquences du tableau 3D et y_i est l'étiquette associée. Une étiquette peut prendre l'une des K valeurs possibles $y_i \in [0, 1, ..., k-1]$. Dans la littérature, on dit que le problème de classification résultante est un problème à K classes.

4.1.2 Apprentissage supervisé

L'apprentissage supervisé est le type d'apprentissage le plus commun en apprentissage automatique (LeCun *et al.*, 2015/05/27). Contrairement à l'apprentissage non supervisé, où les données n'ont pas d'étiquette, au moment de l'entraînement du modèle, toutes les données sont étiquetées. Ce type d'apprentissage consiste à optimiser les paramètres θ du modèle prédictif f afin d'obtenir $\theta_{\min}$ qui sont les paramètres optimisés en fonction des étiquettes de sortie $\boldsymbol{y}$. Succinctement,

$$\theta_{\min} = \arg\min_{\theta} \mathcal{L}\left(f(\theta, \mathbf{X}, \mathbf{y})\right) \tag{4.2}$$

où $\mathcal{L}$ est la fonction de coût, $\mathbf{X}$ et $\mathbf{y}$ sont les données en entrées et les étiquettes de l'ensemble d'entraînement.

La méthode d'optimisation la plus commune en apprentissage profond est l'algorithme du gradient stochastique (*Stochastic Gradient Descent*, SGD). Il s'agit d'un algorithme simple à appliquer, rapide et qui produit de bons résultats (Goodfellow *et al.*, 2016). Le principe de fonctionnement de SGD consiste à prendre un lot (*batch*) aléatoire de données d'entrée $(\boldsymbol{X}, \boldsymbol{y}) \in D$ et de réaliser l'étiquetage de $\boldsymbol{X}$ par le modèle f. Les erreurs sont calculées selon la fonction de coût $\mathcal{L}$ en utilisant les vraies étiquettes $\mathbf{y}$. Puis, les pondérations du modèle sont ajustées par l'application de l'algorithme du gradient. La quantité de changement dans la direction opposée du gradient est appelée le taux d'apprentissage (*Learning rate*, Lr). Ce processus est répété tant qu'il y aura réduction de la valeur de la fonction de coût $\mathcal{L}$. Le nombre de fois où l'on passe à travers notre jeu de données est appelé le nombre d'époque. Une époque correspond donc à une traversée complète du jeu de données.

4.2 Protocole d'expérimentation

Suivant la démarche proposée par Maitre *et al.* (2018) (Chapitre 2, section 2.2), les données nécessaires pour l'apprentissage ont été générées par simulation. Chaque caractéristique (ligne) du jeu de données représente un paramètre mesuré d'un moteur. Le Tableau 4.1 présente les caractéristiques du jeu de données et les données de l'étiquette associée.

Tableau 4.1 Contenu des données échantillonnées et l'étiquetage associé

Caractéristique	**Capteur**	**Unité**
1	Temps	sec
2	Courant	A
3	Vitesse	rad/sec
4	Tension	V
Étiquetage		
Type de défaut du moteur CC		

En considérant un taux d'échantillonnage de 1 ms et une durée de séquence de 5 secondes (voir section 3.6) on aura, d'après le Tableau 4.1, $M = 4$ valeurs enregistrées à chaque pas de temps et ce durant $N = 5000$ pas. Chacune des séquences temporelles est organisée dans une matrice $\mathbf{X}$ de dimension $M \times N$ donnant un total de 20 000 valeurs. Une étiquette y est associée à chacune des séquences $\mathbf{X}$ et elle représente le type de défaut injecté aux moteurs PMDC. Pour les besoins d'implantation informatique, le numéro de moteur et le temps de défaut T_{bris} sont aussi enregistrés pour faciliter l'évaluation de l'apprentissage des modèles prédictifs.

4.2.1 Validation de l'apprentissage

La validation par des jeux de données indépendants (*hold-out validation*) (Yadav & Shukla, 2016) est utilisée pour confirmer le bon fonctionnement du processus d'apprentissage. Cette technique est largement utilisée pour atténuer le phénomène de surapprentissage.

Cette technique de validation divise le jeu de données en trois (3) jeux distincts et indépendants : un jeu d'entraînement, un jeu de validation et un jeu de test. Chacun de ces jeux joue un rôle bien déterminé. Le jeu d'entraînement comporte les seules données que le modèle peut utiliser pour s'entraîner. Le modèle prédictif est évalué tout au long de l'entraînement à l'aide du jeu de validation pour s'assurer que le modèle n'entre pas en régime de surapprentissage et ainsi promouvoir la généralisation. De plus, ce jeu de données permet de sélectionner le modèle le plus performant. Finalement, le jeu de test permet d'évaluer les modèles prédictifs après leur entraînement et ainsi obtenir un aperçu de leur performance. De façon générale, la séparation

des jeux d'entraînement et de validation s'effectue de façon aléatoire en s'assurant que chacune des classes est représentée proportionnellement dans ces deux (2) jeux de données. Dans le cas du jeu de test, les séquences temporelles seront générées de façon indépendante pour qu'elles ne soient pas impliquées dans l'apprentissage des modèles prédictifs. L'isolement du jeu de test est nécessaire pour s'assurer d'une plus grande impartialité lors de l'évaluation de la performance des modèles. Dans ce travail de recherche, les six classes possibles sont (Chapitre 3, Section 3.4) :

1. Sans défaut ;
2. Problème de roulement ;
3. Défaut du capteur de vitesse ;
4. Défaut du capteur de courant ;
5. Défaut du capteur de tension ;
6. Déconnexion de l'alimentation.

Six (6) classes sont réparties uniformément dans les trois jeux de données. Cette répartition des données est importante, car elle permettra aux modèles prédictifs d'apprendre et d'être évalués à l'aide de différents types de défaut sur différents moteurs ayant différents profils de vitesse. Le Tableau 4.2 montre la distribution des données parmi les jeux de données. On note une plus grande proportion des données assignées au jeu d'entraînement. Le but est de donner une plus grande diversité dans les données et de favoriser la capacité de généralisation des modèles.

Tableau 4.2 Jeux de données pour la validation hold-out

Jeu	**Nombre de séquences**	**Proportion**
Entraînement	6375	77%
Validation	1125	14%
Test	750	9%

Rappelons encore une fois que le problème de classification résultant comporte au total trois moteurs, trois profils de vitesse et six classes à traiter.

4.2.2 Fonction de coût

La fonction de coût utilisée dans l'entraînement des modèles est la fonction de vraisemblance logarithmique négative (*Negative Log Likelihood*, NLL). Cette fonction de coût provient de la technique de l'estimation du maximum vraisemblance logarithmique. Cette technique a été puisée du domaine de l'identification des systèmes où l'on tente d'estimer les paramètres d'un système sous étude à partir de son modèle et de ses sorties. Ainsi, la vraisemblance est grande lorsque les paramètres estimés du modèle produisent des sorties qui sont proches des sorties du système sous étude. La Figure 4.2 illustre le comportement de la fonction NLL en fonction de la vraisemblance. Ici, une grande valeur de vraisemblance signifie une grande probabilité que la séquence temporelle soit bien classée. C'est pourquoi la fonction NLL convient dans la mesure indirecte de la performance de l'apprentissage.

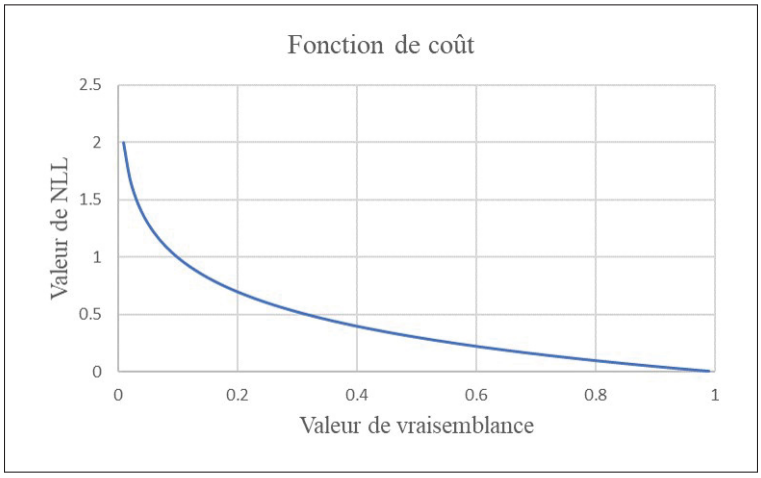


Figure 4.2 Fonction de coût utilisée dans l'optimisation des modèles prédictifs

Pour l'apprentissage supervisé, les paramètres du modèle prédictif sont estimés à partir de ses prédictions. Par convention, l'apprentissage implique une procédure de minimisation (voir Eq. 4.2), c'est pourquoi la fonction de vraisemblance logarithmique possède le signe négatif. Cette fonction de coût se définit comme suit (Goodfellow *et al.*, 2016) :

$$J(\theta) = -\mathbb{E}\left[\ln\left(p_{\text{model}}(\mathbf{Y}|\mathbf{X})\right)\right] \tag{4.3}$$

où $\mathbb{E}$ est l'espérance mathématique de l'expression entre crochets, θ et p_{model} sont respectivement les paramètres et la distribution des probabilités de classification du modèle prédictif, $\mathbf{X}$ est la matrice des données en entrée, $\mathbf{Y}$ est la matrice des étiquettes de données. En pratique, une classification à classe multiple, $\boldsymbol{y}$ est un vecteur contenant l'encodage de type «one-hot». Dans un problème de classification avec K classes où la classe correspondante est dénotée par k, l'encodage « one-hot » est défini par

$$\mathbf{y} \in \{0,1\}^K, \quad y_i = \begin{cases} 1, & \text{si } i = k, \\ 0, & \text{autrement.} \end{cases} \tag{4.4}$$

À l'aide de cet encodage binaire, le calcul de l'espérance mathématique devient la moyenne des prédictions. Donc, on peut réécrire l'équation (4.3) en

$$J(\theta) = -\frac{1}{s} \sum_{\mathbf{X},\boldsymbol{y} \in D} \ln(p_{\text{model}}(\mathbf{y}|\mathbf{x})_k). \tag{4.5}$$

où s est le nombre de séquences temporelles dans le lot de données d'entraînement. Pour pouvoir calculer p_{model}, il faut connaître la nature de la sortie du modèle prédictif. Dans le cas de l'apprentissage profond, la couche de sortie des réseaux de neurones contribue directement au calcul de p_{model}. Dans ce projet de recherche, la fonction d'activation Softmax est celle qui sera utilisée pour ce calcul. Le choix de cette fonction est motivé par le fait qu'elle a été conçue pour des problèmes multi-classe en distribuant les probabilités de classification parmi les K classes. En combinant l'équation du Softmax (2.3) et l'équation (4.5) nous obtenons la forme finale de la fonction de coût

$$\begin{aligned} J(\theta) &= -\frac{1}{s} \sum_{\mathbf{X},\boldsymbol{y} \in D} \ln\left(Q(p_{\text{model}}(\mathbf{y}|\mathbf{x}))_k\right), \\ &= -\frac{1}{s} \sum_{\mathbf{X},\boldsymbol{y} \in D} \left[\ln\left(\frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_j}} \right) \right], \\ &= -\frac{1}{s} \sum_{\mathbf{X},\boldsymbol{y} \in D} \left[x_k - \ln\left(\sum_{j=1}^{K} e^{x_j} \right) \right]. \end{aligned} \tag{4.6}$$

4.2.3 Métrique de performance

Dans cette analyse, nous utiliserons quatre (4) métriques de performance : l'exactitude moyenne (*accuracy*), la précision, le rappel (*recall*) et le F-score. Ces métriques de performance sont couramment utilisées en apprentissage automatique pour évaluer la performance des classifieurs. Pour simplifier l'interprétation de ces métriques, désignons les classes d'un problème de classification par les termes génériques de « classe A », « classe B » et « classe C » et ranger les différents résultats de prédiction sous forme d'un tableau 2D :

Tableau 4.3 Type de prédiction

		Prédiction		
		Classe A	**Classe B**	**Classe C**
Réalité	Classe A	VPA	FNA/FPB	FNA/FPC
	Classe B	FPA/FNB	VPB	FPC/FNB
	Classe C	FPA/FNC	FPB/FNC	VPC

Dans ce tableau de résultats, connu sous le nom de matrice de confusion, VPA, VPB et VPC signifient le nombre de vrais positifs « classe A », « classe B » et «classe C ». De même, FNA, FNB et FNC signifient le nombre de faux négatifs « classe A », « classe B » et « classe C » respectivement. Finalement, FPA, FPB et FPC signifient le nombre de faux positifs « classe A », « classe B » et « classe C ».

L'exactitude est une métrique qui définit à quel point les résultats du modèle sont près de la réalité. Donc, elle mesure le pourcentage des classes correctement prédit par un modèle. Pour un problème multiclasse, il s'agit de l'exactitude moyenne puisque le classifieur doit pouvoir prédire l'appartenance de plusieurs classes. L'équation (4.7) donne la définition de l'exactitude moyenne,

$$\bar{A} = \frac{\text{VPA} + \text{VPB} + \text{VPC}}{\text{VPA} + \text{VPB} + \text{VPC} + \text{FPA} + \text{FPB} + \text{FPC}} \times 100\% \tag{4.7}$$

Le dénominateur de l'équation (4.7) est tout simplement la somme totale des prédictions effectuées.

La précision quant à elle, mesure la proportion des prédictions correctes d'une classe donnée. Ainsi, la précision P pour la classe A est calculée par

$$P_A = \frac{\text{VPA}}{\text{VPA} + \text{FPA}} \times 100\% \tag{4.8}$$

où FPA représente le nombre de prédictions de la classe A qui sont en réalité fausses. On peut calculer, de la même façon, la précision des autres classes du problème.

La métrique rappel sert à déterminer la capacité du modèle prédictif à identifier correctement une classe contenue dans le jeu de données. Le rappel R pour la classe A est calculé par

$$R_A = \frac{\text{VPA}}{\text{VPA} + \text{FNA}} \times 100\% \tag{4.9}$$

Finalement, on peut calculer la valeur moyenne entre la précision P et le rappel R aussi appelé le F-score. C'est-à-dire,

$$F = \frac{2 \times P \times R}{P + R} \tag{4.10}$$

Il s'agit d'une moyenne harmonique, car P et R sont des ratios.

4.3 Modèle de base

Quatre (4) modèles pour la classification de séquences temporelles sont implantés et testés. Ces modèles représentent l'application des concepts récents du domaine de l'apprentissage automatique. Les quatre modèles étudiés dans cette section sont : le *Fully Convolutional Network* (FCN), le *Residual Network* (ResNet), l'Encodeur et le *Recurent Neural Network* (RNN).

L'objectif de cette section est de mettre à l'essai ces modèles tirés de la littérature qui ont montré une bonne performance sur des séquences temporelles multidimensionnelles. À noter que les valeurs des hyperparamètres contrôlant ces modèles sont celles proposées dans la littérature. Nous ferons l'ajustement des hyperparamètres dans la prochaine section.

4.3.1 FCN

Le FCN est un modèle proposé par Wang *et al.* (2016). Ce modèle utilise des convolutions pour traiter les séquences temporelles. Ce type de modèle donne de très bons résultats en classification d'images. Le FCN proposé est légèrement adapté pour la tâche de classification de séquence temporelle. La particularité de ce modèle est qu'il n'utilise aucune couche de pooling, mais seulement des convolutions identiques qui conservent la dimension des données d'entrées dans toutes les couches de neurones.

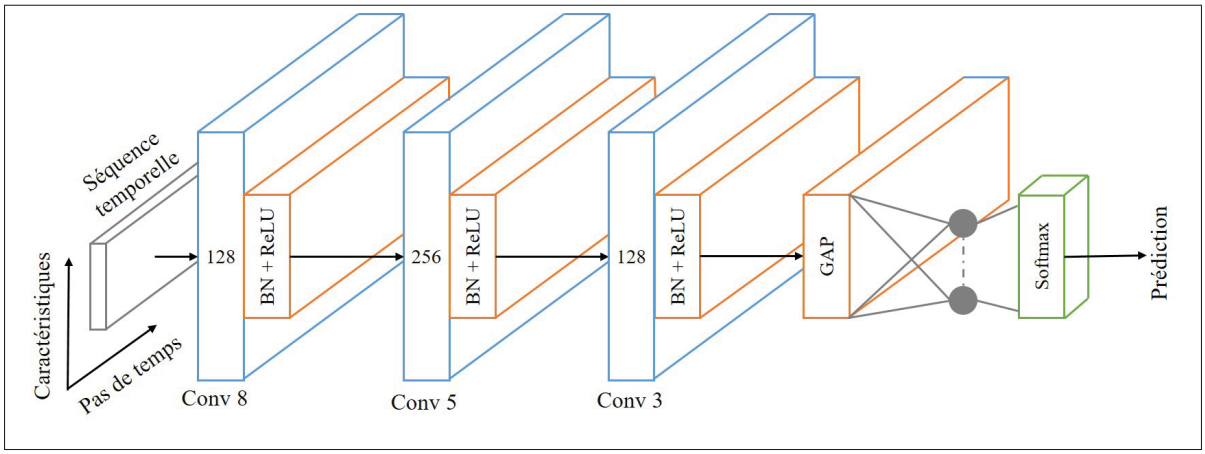


Figure 4.3 Architecture du FCN

On voit dans la Figure 4.3, le FCN est composé de trois blocs de convolution, d'un bloc GAP et d'une couche Softmax complètement connectée. Comme indiqué dans la section 2.2.2, le pooling est une opération qui sert à réduire la dimension d'une matrice par le calcul d'une valeur représentative pour différents sous-blocs de la matrice. Cette valeur représentative peut être la valeur maximale ou encore la moyenne des sous-blocs constituants. Ici, le GAP réduit la dimension $M \times N$ d'une matrice d'entrée en un vecteur de $1 \times M$ où M est le nombre de caractéristiques et N est le nombre de pas de temps. Donc, en introduisant une couche GAP nous parvenons à réduire le surapprentissage tout en diminuant le nombre de paramètres à entraîner. La Figure 4.3 montre la composition des couches de ce modèle. Les blocs de traitement sont composés d'une convolution, d'une normalisation par lot BN et d'un ReLU. Les étapes de calcul

d'un bloc de traitement sont :

$$\begin{aligned} h &= \boldsymbol{W} \circledast \boldsymbol{x} + \boldsymbol{b} \\ s &= BN(h) \\ y &= ReLU(s) \end{aligned} \tag{4.11}$$

où $\boldsymbol{W}$ est la matrice des paramètres de la couche h, $\circledast$ est l'opérateur de convolution et $\boldsymbol{b}$ est le biais ou l'ordonnée à l'origine. La normalisation par lot (bloc BN) est un mécanisme de régularisation qui aide à rendre l'espace de fouille plus lisse lors de l'optimisation. Aussi, la normalisation par lot augmente la vitesse de convergence du modèle lors de l'apprentissage. Pour une couche du modèle à d dimensions avec $\boldsymbol{x} = (x_1, ..., x_d)$ chacune des dimensions sera normalisée par la standardisation selon l'équation (4.12)

$$\hat{x}_k = \frac{x_k - \mathbb{E}\left[x_k\right]}{\sqrt{\text{Var}\left[x_k\right]}}, \quad k = 1, 2, ..., d \tag{4.12}$$

où $\mathbb{E}$ est l'espérance mathématique des données d'entraînement et Var est la variance.

Les trois blocs de convolutions sont composés de 128, 256 et 128 filtres et ont un noyau de dimension 8, 5 et 3 respectivement. Le modèle est entraîné sur le jeu de données d'entraînement en utilisant l'algorithme d'optimisation Adam (Kingma & Ba, 2014) et les hyperparamètres présentés dans le Tableau 4.4. Dans ce tableau, Lr est le taux d'apprentissage qui est l'incrément utilisé lors de l'optimisation. β_1, β_2 et ϵ sont des hyperparamètres reliés à l'algorithme d'optimisation Adam. Le nombre d'époques représente le nombre de fois que le jeu de données est parcouru au complet et la dimension des lots est le nombre de séquences temporelles traitées par le modèle avant d'appliquer l'optimisation des pondérations.

4.3.2 ResNet

Le deuxième modèle testé est le ResNet. Ce modèle est aussi proposé par Wang *et al.* (2016). Le ResNet, tout comme le FCN, est un modèle prédictif à convolution. Cependant, il possède un nombre plus important de couches cachées par rapport au modèle FCN. Pour aider le gradient à

Tableau 4.4 Paramètres d'entraînement du FCN

Paramètre	**Valeur**
Lr	0.001
β_1	0.9
β_2	0.999
ϵ	1×10^{-8}
Époques	30
Dimension des lots	16

se propager à la couche la plus profonde, le ResNet propose des connexions résiduelles qui aident le gradient à se propager lors de l'apprentissage (He *et al.*, 2016). Ces connexions résiduelles sont aussi appelées connexions raccourcies et leur développement constituait une avancée importante permettant aux modèles prédictifs d'avoir un grand nombre de couches cachées.

Pour définir ces connexions résiduelles, posons $H(\boldsymbol{x})$ comme étant quelques couches du modèle comprenant des fonctions non-linéaires où $\boldsymbol{x}$ est le vecteur d'entrée de la première couche. En supposant que la dimension de $H(\boldsymbol{x})$ et $\boldsymbol{x}$ est la même, alors la fonction résiduelle est définie par $F(\boldsymbol{x}) = H(\boldsymbol{x}) + \boldsymbol{x}$. Dans le cas où la dimension de $H(\boldsymbol{x})$) serait différente de celle de $\boldsymbol{x}$, He *et al.* (2016) propose une technique appelée application identité (*identity mapping*). Cette technique consiste à ajouter une projection linéaire à $\boldsymbol{x}$ afin de changer sa dimension. La fonction résiduelle devient alors $F(\boldsymbol{x}) = H(\boldsymbol{x}) + \boldsymbol{W}_s\boldsymbol{x}$ où $\boldsymbol{W}_s$ est la matrice de projection.

L'architecture du réseau utilisée est montrée dans la Figure 4.4. Pour construire le modèle, le bloc de convolution du FCN (Éq. 4.11) est réutilisé et est renommé en $Bloc_{f-k}(x)$ où f dénote le nombre de filtres et k la dimension des filtres. Le nombre de filtres pour chacun des blocs résiduels est 64, 128 et 128 respectivement. Le ResNet est composé de trois blocs résiduels qui

sont présentés dans l'équation (4.13).

$$\begin{aligned} h_1 &= Bloc_{f-8}(\boldsymbol{x}) \\ h_2 &= Bloc_{f-5}(h_1) \\ h_3 &= Bloc_{f-3}(h_2) \\ f &= h_3 + \boldsymbol{x} \\ y &= ReLU(f) \end{aligned} \tag{4.13}$$

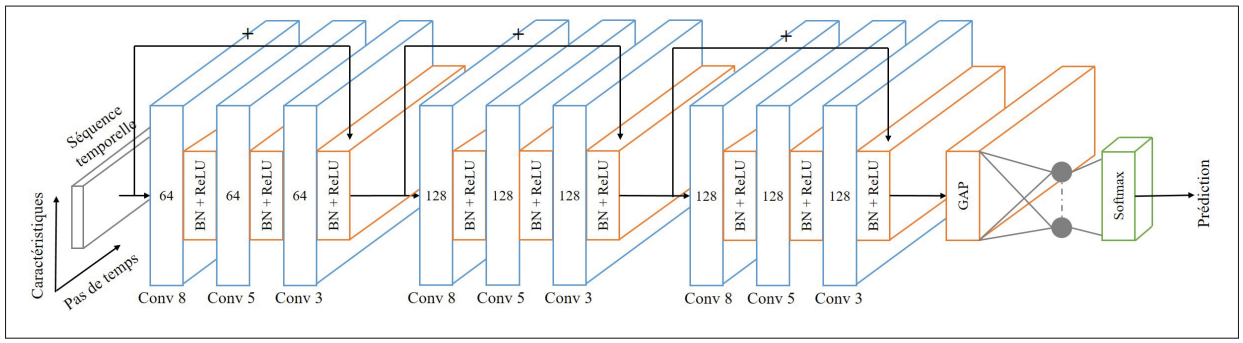


Figure 4.4 Architecture du ResNet

Dans la Figure 4.4, le symbole "+" représente l'addition de l'entrée du bloc avec la sortie du bloc qui est le principe de fonctionnement du ResNet. Cette particularité du ResNet est indiquée dans l'équation du bloc résiduel par $f = h_3 + \boldsymbol{x}$. Enfin, le résultat calculé est acheminé dans une couche GAP puis une couche Softmax complètement connectée à la sortie du réseau. Ce modèle est entraîné avec l'algorithme de descente de gradient Adam et utilise les paramètres du Tableau 4.5.

Tableau 4.5 Paramètres d'entraînement du ResNet

Paramètre	**Valeur**
Lr	0.001
β_1	0.9
β_2	0.999
ϵ	1×10^{-8}
Époques	30
Dimension des lots	16

4.3.3 Encodeur

Le prochain modèle testé est un Encodeur pour les séquences temporelles. L'Encodeur est proposé par Serrà *et al.* (2018)). Ce modèle est similaire au modèle FCN puisqu'il utilise des blocs de convolution, mais plutôt que d'utiliser une couche GAP, c'est un mécanisme appelé « Attention » (Vaswani *et al.*, 2017)) qui est utilisé. Aussi, les blocs de convolution comportent certaines différences que nous allons expliciter dans cette section.

Tout d'abord, l'Encodeur est composé de trois blocs de convolution définis par l'équation (4.14) et est schématisé dans la Figure 4.5.

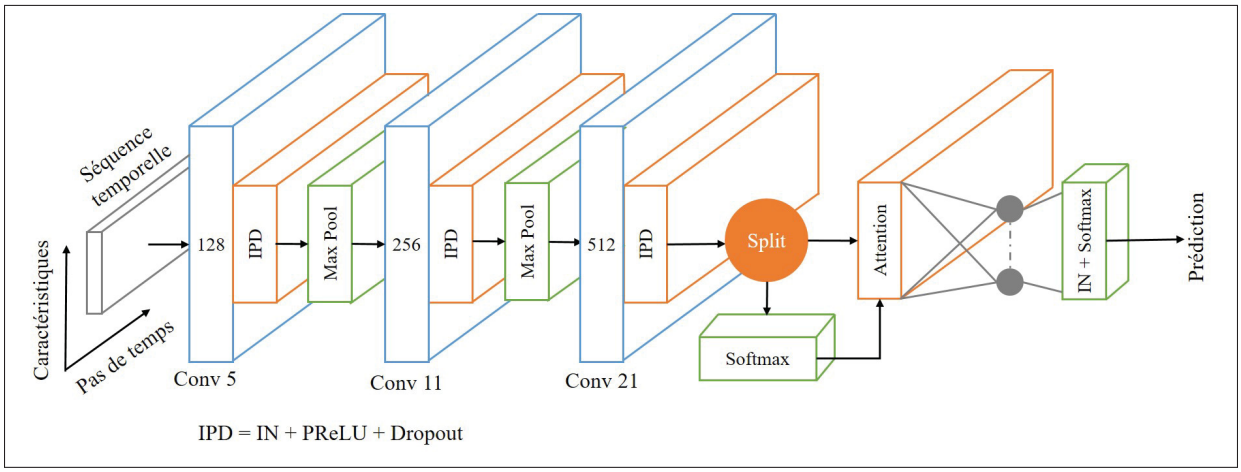


Figure 4.5 Architecture de l'encodeur

$$
\begin{aligned}
h_1 &= \boldsymbol{W} \circledast \boldsymbol{x} + \boldsymbol{b} \\
s_1 &= IN(h_1) \\
s_2 &= PReLU(s_1) \\
y &= Dropout(s_2)
\end{aligned}
\tag{4.14}
$$

Ce qui différencie le bloc de convolution avec celui du FCN est qu'il comprend un bloc de normalisation d'instance IN avec l'application affine (Ulyanov *et al.*, 2016), d'une fonction

d'activation PReLU et un bloc Dropout. Les trois blocs de convolutions sont composés de 128, 256 et 512 filtres et ont des noyaux de dimensions 5, 11 et 21 respectivement.

À la suite des deux premiers blocs de convolution, il y a une couche de type Max pooling (Goodfellow *et al.*, 2016) avec une dimension de filtre de 2 et, à la sortie du troisième bloc, il y a le mécanisme d'attention. Le mécanisme d'attention a été proposé la première fois par Vaswani *et al.* (2017). C'est une technique qui a été développée pour créer un lien entre les éléments d'une séquence, même quand cette séquence est très longue. Ce mécanisme sert à pallier le problème des RNN qui n'arrive pas à trouver des dépendances dans les longues séquences. Dans le cas de l'Encodeur, le mécanisme d'attention diffère un peu de celui proposé dans le papier original. Après le troisième bloc de convolution, les 512 filtres sont séparés en deux. Une première moitié dénotée $\boldsymbol{A}$ passe dans un Softmax pour la dimension du temps et servira de mécanisme d'attention pour la deuxième moitié dénotée H. Ensuite, pour chaque filtre, on applique le produit scalaire

$$h_i = \boldsymbol{h}_i \cdot \boldsymbol{a}_i \tag{4.15}$$

où $\boldsymbol{h}_i$ est un filtre de l'ensemble H et $\boldsymbol{a}_i$ est une ligne de la matrice$\boldsymbol{A}$. Finalement, ce résultat est passé dans une couche complètement connectée, un IN et un Softmax pour faire la classification finale.

L'algorithme utilisé pour entraîner ce modèle est le SGD. Le Tableau 4.6 résume les hyperparamètres utilisés. En plus, pour entraîner l'Encodeur, Serrà *et al.* (2018)) ont utilisé un ordonnanceur pour le taux d'apprentissage. L'ordonnanceur divise la valeur du taux d'apprentissage par 3 à toutes les 10 époques sans amélioration du modèle sur le jeu de validation. Cependant, la valeur minimum du Lr est bloquée à 10^{-4}.

Tableau 4.6 Paramètres d'entraînement de l'encodeur

Paramètre	**Valeur**
Lr	0.005
Dropout	0.2
Époques	30
Dimension des lots	12

4.3.4 RNN

Le dernier modèle testé est le Recurent Neural Network (RNN). Le RNN est un modèle qui a été développé pour la prédiction et la classification de séquences ou de listes. Il est très utilisé notamment dans la traduction, la classification de séquences vidéo et dans le traitement automatique du langage. Le RNN est similaire à un réseau de neurones classique. Par contre, il y a un ajout supplémentaire d'une boucle pour que les données du passé puissent contribuer à la prédiction. La Figure 4.6 montre le fonctionnement du RNN à 3 pas de temps. Donc, comme un réseau de neurones classique, le RNN possède une matrice **W** qui multiplie l'entrée x_t et aussi une matrice supplémentaire **U** qui multiplie le résultat obtenu au temps $t-1$.

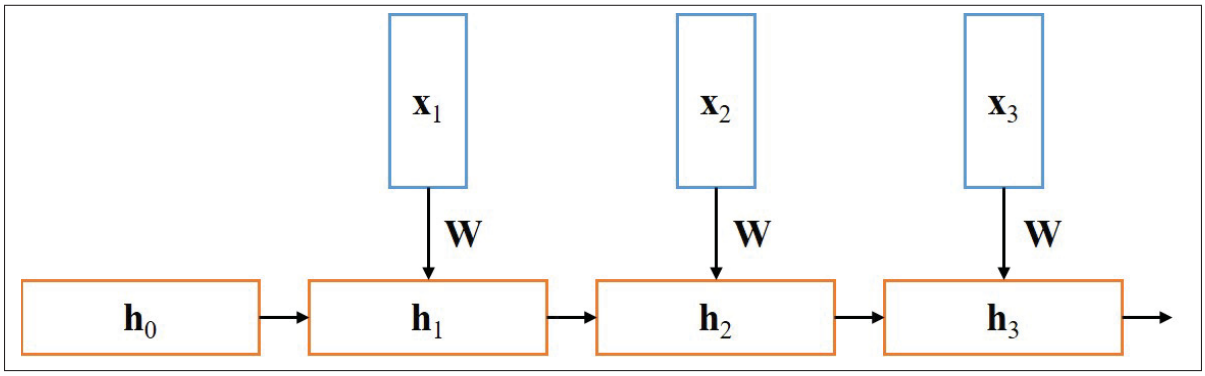


Figure 4.6 RNN de base

Un RNN se décrit de la façon suivante :

$$\boldsymbol{h}_t = \Phi(\boldsymbol{b} + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{W}\boldsymbol{x}_t) \tag{4.16}$$

où $\boldsymbol{h}_t$ est le résultat du RNN à l'instant t, $\boldsymbol{b}$ est le biais, Φ est la fonction d'activation tangente hyperbolique et **U** et **W** sont des matrices des pondérations à ajuster durant l'entraînement.

Un des problèmes de ce modèle est qu'il est souvent difficile à entraîner lorsque les séquences deviennent longues. Ce problème est connu comme étant le *exploding gradient* ou le *vanishing gradient*. De plus, le RNN a de la difficulté à tenir compte des dépendances à long terme (Bengio *et al.*, 1994). Dans notre cas, les séquences à analyser sont plutôt longues et le RNN n'est

pas le modèle convenable. Pour contrer ces problèmes, un modèle amélioré a été proposé par Hochreiter & Schmidhuber (1997). Ce modèle s'appelle le *Long Short-Term Memory* (LSTM). Il utilise le concept du RNN, mais y ajoute le concept de « porte » pour arriver à apprendre les dépendances dans de longues séquences. La Figure 4.7 montre le fonctionnement interne d'une cellule LSTM. Une cellule LSTM est composée de trois (3) portes : entrée, oubli et sortie. Une composante importante du LSTM est l'ajout d'un « état de cellule » qui est représenté par c_t. L'état de cellule permet au LSTM d'apprendre quelles données du passé sont à retenir. Aussi, c'est grâce à ce mécanisme que le gradient peut se propager lors de l'apprentissage. Les Équations 4.17, 4.18 et 4.19 décrivent le fonctionnement interne d'une cellule LSTM. Dans les équations, $\boldsymbol{b}_{[x]}$ sont les biais, $\boldsymbol{U}_{[x]}$ et $\boldsymbol{W}_{[x]}$ sont les matrices avec les paramètres à apprendre et Φ est la fonction d'activation sigmoïde.

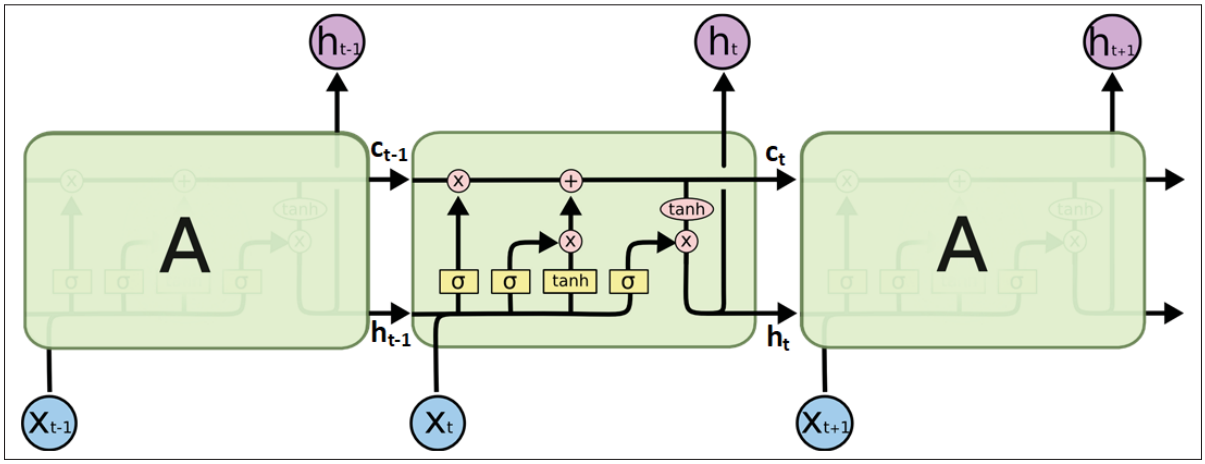


Figure 4.7 Cellule LSTM
Tirée de Olah (2015, 27 août)

Portes :

$$\begin{aligned} \boldsymbol{i}_t &= \sigma(\boldsymbol{b}_{[i]} + \boldsymbol{U}_{[i]} h_{t-1} + \boldsymbol{W}_{[i]} x_t) \\ \boldsymbol{f}_t &= \sigma(\boldsymbol{b}_{[f]} + \boldsymbol{U}_{[f]} h_{t-1} + \boldsymbol{W}_{[f]} x_t) \\ \boldsymbol{o}_t &= \sigma(\boldsymbol{b}_{[o]} + \boldsymbol{U}_{[o]} h_{t-1} + \boldsymbol{W}_{[o]} x_t) \end{aligned} \tag{4.17}$$

État :

$$\begin{aligned} \widetilde{\boldsymbol{c}_t} &= \tanh(\boldsymbol{b}_{[c]} + \boldsymbol{U}_{[c]} h_{t-1} + \boldsymbol{W}_{[c]} x_t) \\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \widetilde{\boldsymbol{c}_t} \end{aligned} \tag{4.18}$$

Couche cachée :

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{4.19}$$

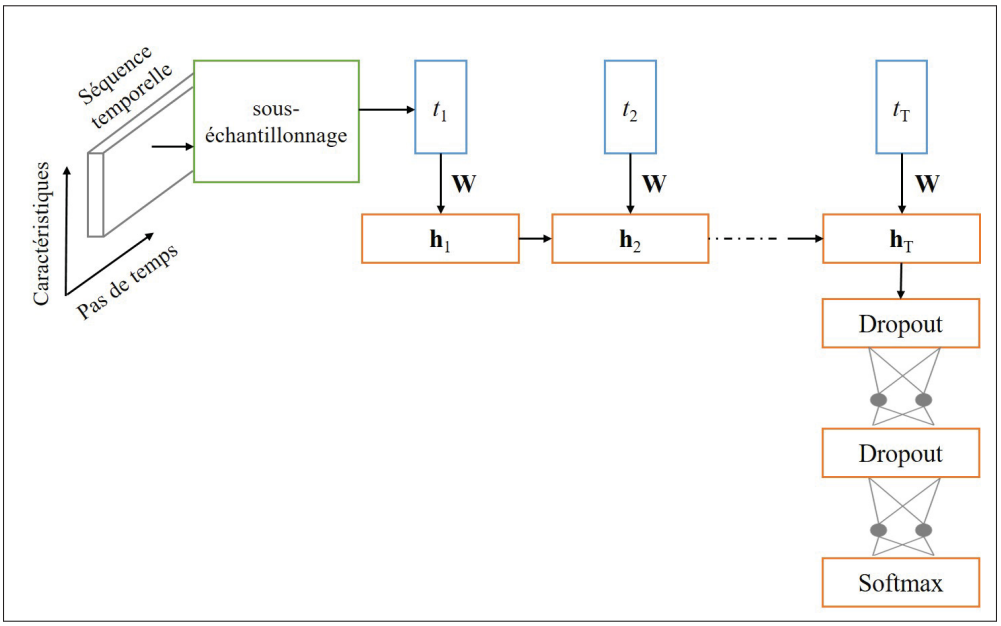


Figure 4.8 Architecture du LSTM

La Figure 4.8 montre l'architecture utilisée dans ce travail. Tout d'abord, puisque nos données temporelles sont très longues (5000 pas de temps), un sous-échantillonnage est appliqué pour accélérer l'apprentissage du LSTM. Le sous-échantillonnage réduit notre dimension d'entrée par 10 en faisant une moyenne de 10 pas de temps. Ensuite, la séquence de 500 échantillons est passée dans un LSTM bidirectionnel. Finalement, le résultat de la dernière cellule est passé dans un perceptron multicouche (MLP) de deux couches puis la classification est obtenue grâce à un softmax. Dans chacune des couches du LSTM et du MLP il y a un dropout. Tous les paramètres du LSTM sont décrits dans le Tableau 4.7.

Tableau 4.7 Paramètres d'entraînement du LSTM

Paramètre	**Valeur**
Lr	0.1
Momentum	0.9
Nombre de couche du LSTM	3
Dimension de cellule du LSTM	512
Dimension MLP couche 1	1024
Dimension MLP couche 2	512
Dropout	0.1
Époques	30
Dimension des lots	16

4.4 Analyse des résultats

Dans cette section, nous analysons la performance des modèles prédictifs présentés dans la section 4.3. Chacun des modèles a été entraîné durant 30 époques à l'aide d'une carte graphique NVidia Tesla V100. Lors de l'entraînement des modèles, à chaque époque, la valeur de la fonction de coût NLL et de la métrique exactitude sont calculées sur les données d'entraînement et de validation. Lorsque l'entraînement est terminé, le modèle le plus performant en validation est conservé. Les Figures 4.9, à 4.12 comparent respectivement les résultats sur le jeu d'entraînement et de validation des quatre modèles. Ces graphiques nous informent sur la progression de l'entraînement des modèles. Dans le cas du ResNet (Figure 4.10) et du RNN (Figure 4.12), il n'y a pas de tendances observables à la sortie de la fonction de coût NLL et sur l'exactitude des classifications. Une raison possible de ce comportement est le manque de capacité de ces modèles prédictifs. Le nombre de couches cachées n'est probablement pas suffisant pour le problème en main. Pour le FCN (Figure 4.9) et l'Encodeur (Figure 4.11), on observe une tendance à la hausse pour l'exactitude des classifications et une tendance à la baisse pour la fonction de coût NLL. Ces tendances sont de bon augure puisqu'elles indiquent une bonne capacité d'apprentissage de ces modèles prédictifs.

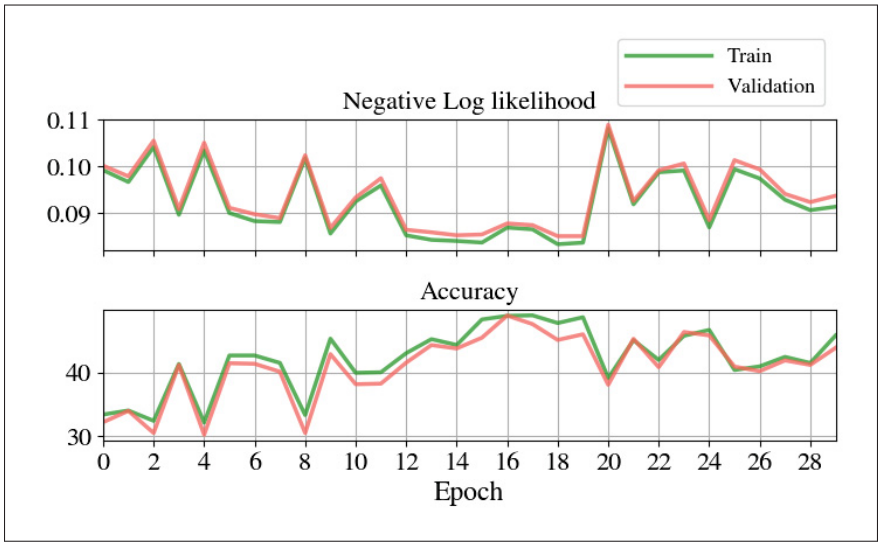


Figure 4.9 Courbe d'entraînement du FCN

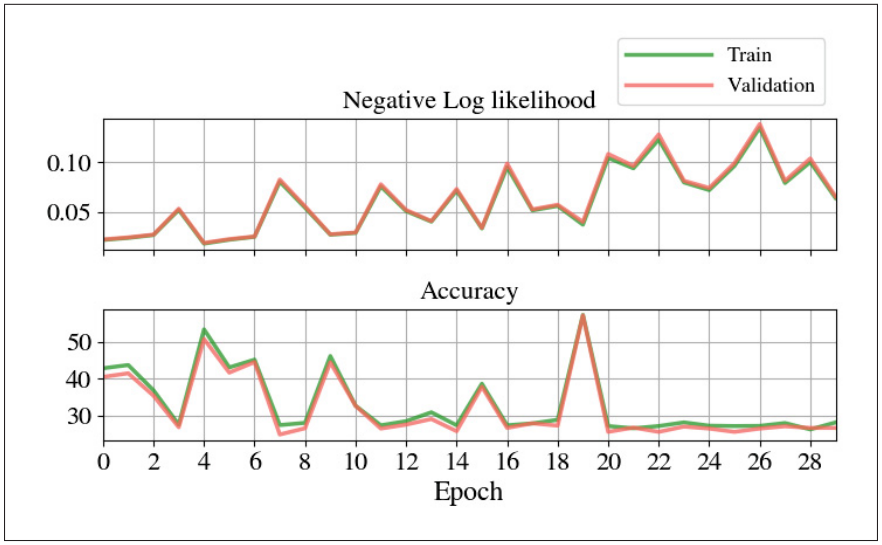


Figure 4.10 Courbe d'entraînement du ResNet

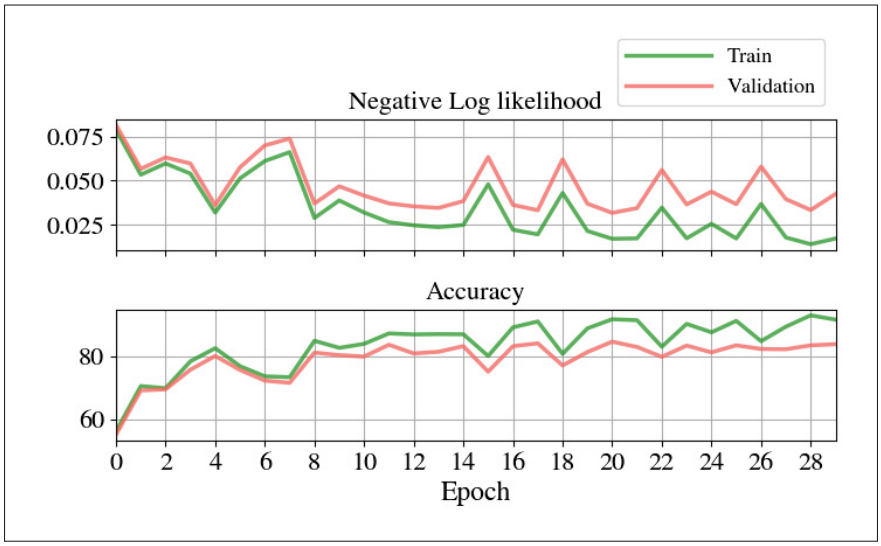


Figure 4.11 Courbe d'entraînement de l'encodeur

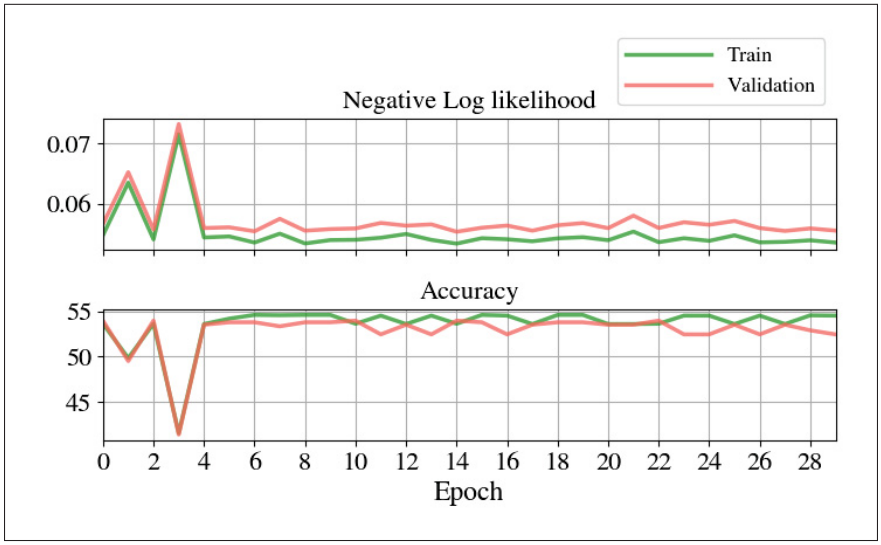


Figure 4.12 Courbe d'entraînement du RNN

4.4.1 Validation des modèles

La performance en classification de ces modèles est évaluée à l'aide du jeu de données de validation contenant 1125 séquences temporelles. Les Figures 4.13 à 4.16 représentent les matrices de confusion pour le FCN, le ResNet l'Encodeur et le RNN respectivement. La matrice de confusion permet aussi de visualiser les classes « faciles » et « difficiles » pour un modèle à prédire.

Pour le modèle FCN, la matrice de confusion résultante est donnée dans la Figure 4.13. Sa performance en classification est résumée dans le Tableau 4.8. Le FCN est capable de très bien reconnaître les défauts de type 2 (défaut du capteur de vitesse) et 3 (défaut du capteur de courant), mais est incapable de reconnaître convenablement les autres défauts. Dans cette expérience, le FCN ne reconnaît aucun défaut de type 4 (défaut du capteur de tension).

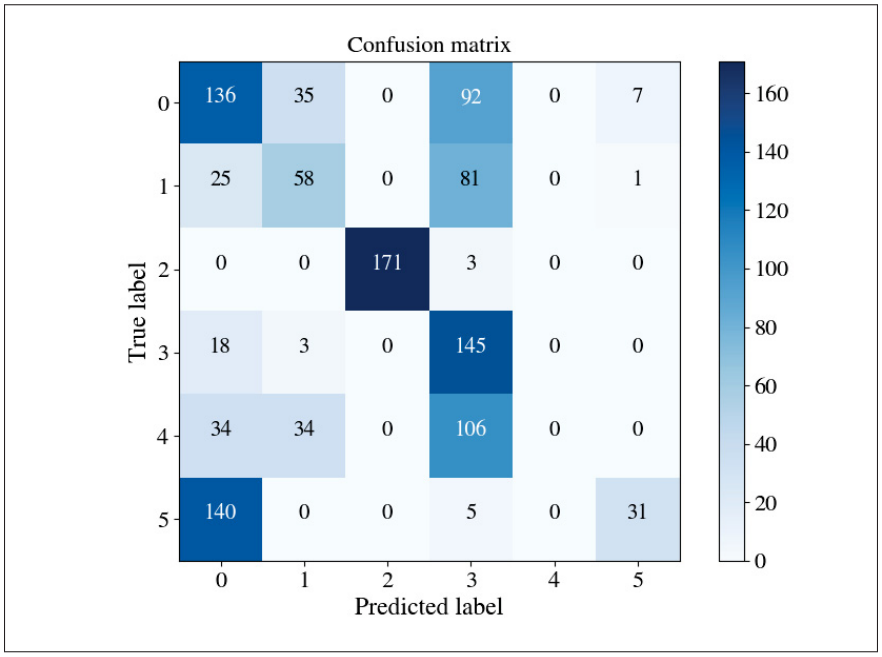


Figure 4.13 Matrice de confusion du FCN

Pour le modèle ResNet, la matrice de confusion résultante est donnée dans la Figure 4.14. Sa performance en classification est résumée dans le Tableau 4.9. Le ResNet est capable de très bien reconnaître les défauts de type 2 (défaut du capteur de vitesse) , 4 (défaut du capteur de tension)

Tableau 4.8 Performance du modèle FCN

Exactitude moyenne, Ā	
541/1125 (48.09%)	
Classe	**Rappel, R**
0	136/270 (50.4%)
1	58/165 (35.2%)
2	171/174 (98.3%)
3	145/166 (87.3%)
4	0/174 (40%)
5	31/176 (17.6%)

et 5 (déconnexion de l'alimentation) obtenant un score parfait pour ces trois types de défauts. Or, ce modèle est incapable de reconnaître convenablement les autres défauts. Dans cette expérience, le ResNet présente une mauvaise performance pour les défauts de type 3 (défaut du capteur de courant).

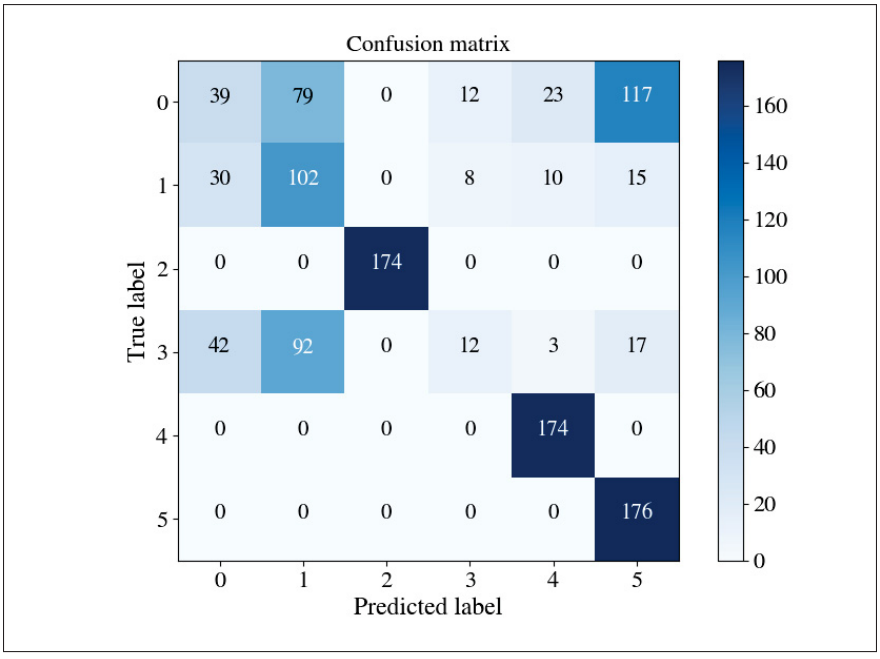


Figure 4.14 Matrice de confusion du ResNet

Pour le modèle Encodeur, la matrice de confusion résultante est donnée dans la Figure 4.15. Sa performance en classification est résumée dans le Tableau 4.10. Le modèle Encodeur est

Tableau 4.9 Performance du modèle ResNet

Exactitude moyenne, Ā	
677/1125 (60.18%)	
Classe	**Rappel, R**
0	39/270 (14.4%)
1	102/165 (61.8%)
2	174/174 (100%)
3	12/166 (7.22%)
4	174/174 (100%)
5	176/176 (100%)

capable de très bien reconnaître tous les types de défauts à l'exception du type 1 (problème de roulement).

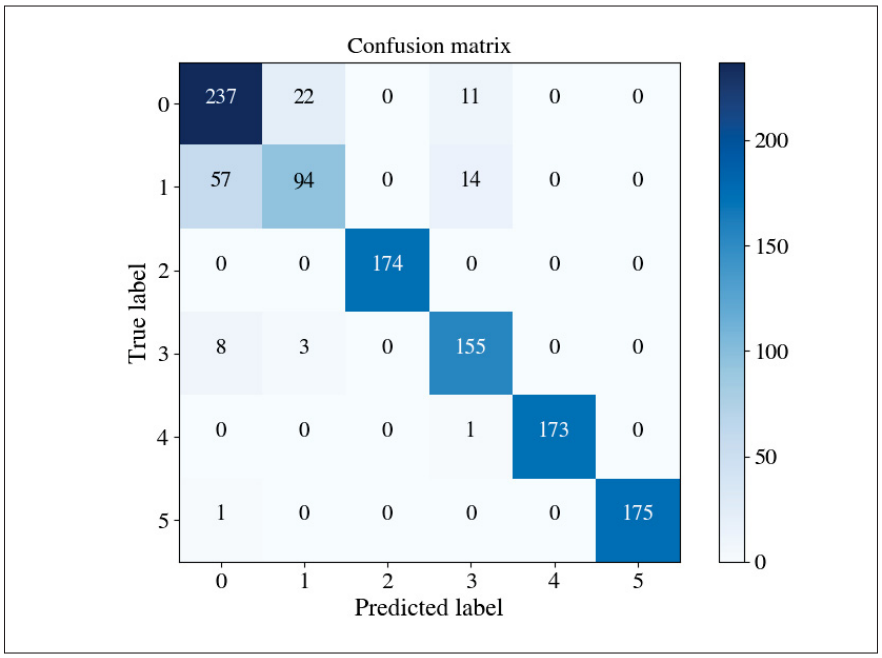


Figure 4.15 Matrice de confusion de l'encodeur

Finalement, pour le modèle LSTM, la matrice de confusion résultante est donnée dans la Figure 4.16. Sa performance en classification est résumée dans le Tableau 4.11. Le LSTM est capable de très bien reconnaître les défauts de type 2 (défaut du capteur de vitesse) , 3 (défaut du capteur de courant) et 4 (défaut du capteur de tension) obtenant un score parfait les défauts de type

Tableau 4.10 Performance du modèle Encodeur

Exactitude moyenne, Ā	
1008/1125 (89.6%)	
Classe	**Rappel, R**
0	237/270 (87.8%)
1	94/165 (56.97%)
2	174/174 (100%)
3	155/166 (93.4%)
4	173/174 (99.4%)
5	175/176 (99.4%)

2. Or, ce modèle est incapable de reconnaître convenablement les autres défauts. Dans cette expérience, le LSTM présente une mauvaise performance pour les défauts de type 1 (problème de roulement) et de type 5 (déconnexion de l'alimentation).

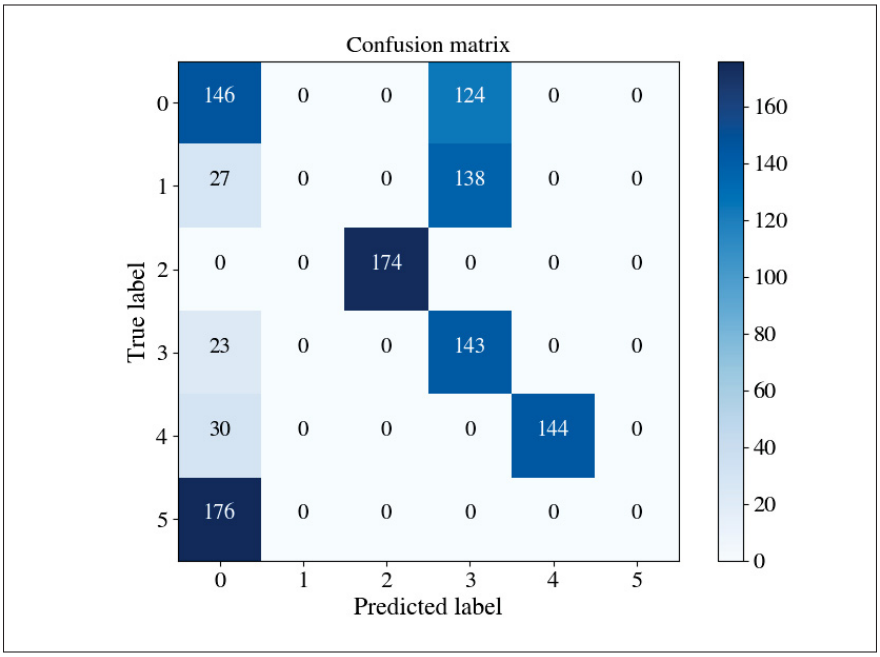


Figure 4.16 Matrice de confusion du LSTM

Tableau 4.11 Performance du modèle LSTM

Exactitude moyenne, Ā	
607/1125 (53.96%)	
Classe	**Rappel, R**
0	146/270 (54.1%)
1	0/165 (0%)
2	174/174 (100%)
3	143/166 (86.1%)
4	144/174 (82.8%)
5	0/176 (0%)

4.5 Sélection d'un modèle final

Deux critères sont utilisés dans la comparaison des modèles prédictifs. Le premier critère est la valeur moyenne des métriques de performance. Le Tableau 4.12 résume la performance moyenne de chacun des modèles sur le jeu de validation. Nous constatons que le modèle Encodeur est de loin le modèle le plus performant avec les plus grandes valeurs moyennes pour les métriques exactitude, précision, rappel et F-score. De plus, l'Encodeur présente une moyenne supérieure à 80% pour toutes ces métriques.

Tableau 4.12 Performance moyenne des modèles de base

Modèle	**Exactitude**	**Précision**	**Recall**	**F-score**
FCN	48.09%	49.37%	58.23%	53.43%
ResNet	60.18%	57.83%	63.92%	60.72%
Encodeur	**89.6%**	**90.47%**	**90.08%**	**90.28%**
RNN	53.96%	45.27%	53.83%	49.18%

D'après les résultats obtenus, ces modèles prédictifs sont capables de prédire les défauts. On remarque aussi que certains défauts sont plus « faciles » à prédire que d'autres. Que peut-on constater pour le cas où il n'y a pas de défaut ? Prédire un défaut alors qu'il n'y en a pas peut entraîner des coûts de maintenance supplémentaire et annuler les avantages de la maintenance prédictive. Il est donc important pour un modèle prédictif de classer convenablement les séquences temporelles où il n'y a pas de défaut. Le deuxième critère de comparaison est donc

d'identifier le modèle qui présente le plus haut taux de rappel dans le cas où il y aurait absence de défaut.

Le Tableau 4.13 donne la performance des modèles prédictifs sur la classe 0 (cas sans défaut). Encore une fois, le modèle Encodeur est en mesure de prédire 87.8% des cas sans défaut. Les autres modèles font piètre figure dans cette comparaison.

Tableau 4.13 Performance des modèles prédictifs

Modèle	**Rappel, R pour la classe 0**
FCN	50.4%
ResNet	14.4%
Encodeur	**87.8%**
LSTM	54.1%

4.6 Ajustement du modèle final

La sélection du modèle Encodeur comme modèle final nous permet de procéder à l'ajustement de ses hyperparamètres afin d'augmenter sa performance en classification. Rappelons que ces hyperparamètres sont des paramètres de contrôle du processus de l'entraînement. Ce sont des valeurs prédéterminées manuellement avant l'entraînement du modèle. Une fois le réglage des hyperparamètres réalisé, le modèle Encodeur subira une dernière évaluation à l'aide du jeu de test.

Afin de minimiser le temps de fouille, seuls quatre (4) hyperparamètres de l'Encodeur sont sujets à des variations discrètes (fouille par valeurs discrètes). Ces hyperparamètres à ajuster sont : i) la valeur du dropout ; ii) le nombre de couches du modèle ; iii) le nombre de filtres dans les couches ; iv) la dimension des filtres de convolution. Le Tableau 4.14 montre les valeurs initiales et les nouvelles valeurs identifiées de ces hyperparamètres. Ces nouvelles valeurs ont été identifiées par des essais empiriques et qui ont produit une exactitude moyenne supérieure à celle du Tableau 4.10 pour le modèle Encodeur.

Tableau 4.14 Valeurs des hyperparamètres

Hyper-paramètre	Valeur initiale	Nouvelle valeur
Dropout	0.2	0.35
Dimension du modèle	128, 256, 512	64, 128, 256, 256, 512
Noyaux de convolutions	5, 11, 21	3, 5, 7

La Figure 4.17 montre l'évolution de l'exactitude moyenne sur le jeu de validation de l'Encodeur avec les nouvelles valeurs d'hyperparamètres. Cette figure nous permet de visualiser la contribution des hyperparamètres à la performance de classification durant l'entraînement. Nous pouvons remarquer que l'exactitude moyenne de l'Encodeur est grandement influencée par le changement de dimension du modèle (nombre de couches et nombre de filtres). Aussi, la tendance à la hausse des courbes de la Figure 4.17 laisse croire qu'il est possible d'améliorer sa performance en augmentant le nombre d'époques d'entraînement. Puisque l'exactitude moyenne est calculée sur le jeu de validation, nous pouvons conclure que le modèle n'est pas encore en régime de surapprentissage.

Les nouvelles valeurs du Tableau 4.14 sont appliquées dans l'entraînement du modèle Encodeur durant 200 époques. La Figure 4.18 est la courbe d'entraînement du modèle. Encore une fois, la tendance à la baisse de la fonction de perte NLL et la tendance à la hausse de l'exactitude moyenne laissent présager que le modèle est en régime d'apprentissage et non pas en régime de sous-apprentissage.

Nous avons sélectionné le modèle qui performe le mieux sur la base de validation et nous avons testé ce modèle sur notre base de test. La figure 4.19 montre la matrice de confusion de l'encodeur final sur la base de test. Nous pouvons remarquer que pour la majorité des classes, le modèle fait des prédictions excellentes. Le seul endroit où le modèle a un peu plus de difficulté est pour différencier les classes 0 et 1, soit le moteur en fonctionnement normal et le problème au niveau du roulement à bille. Au total, notre modèle performe avec une exactitude de 88.53% sur la base de test, ce qui démontre une excellente efficacité.

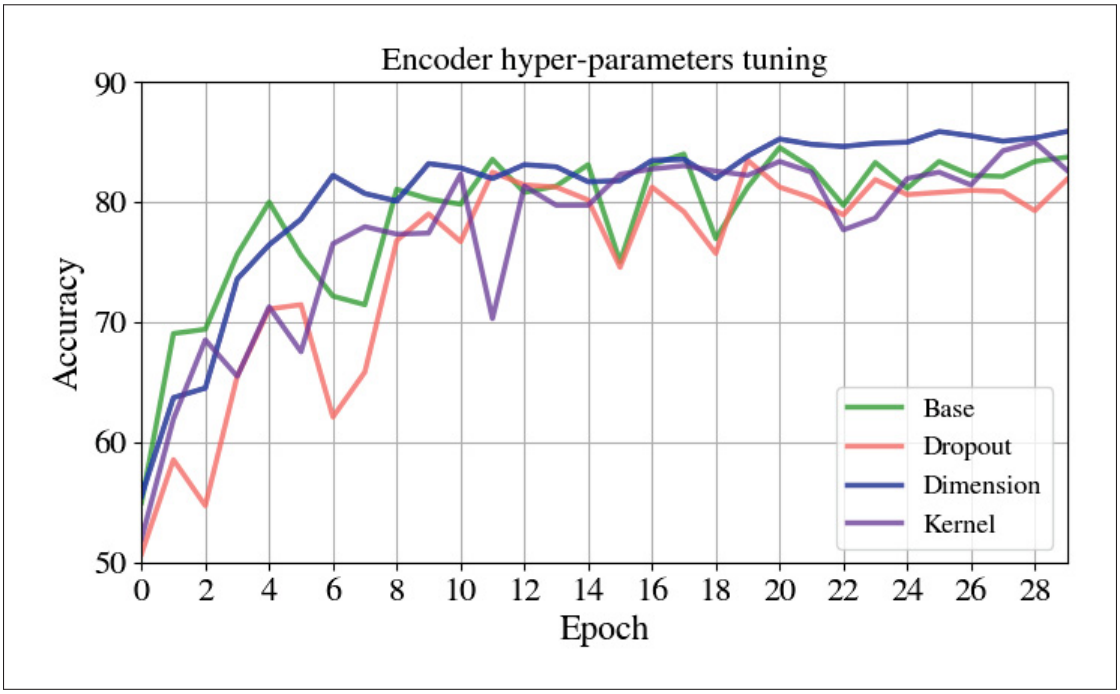


Figure 4.17 Impact des hyperparamètres de l'Encodeur sur l'exactitude moyenne

L'évaluation de l'Encodeur ajusté et entraîné s'effectue par le biais du jeu de test. Il s'agit d'une évaluation équitable et sans biais puisque le jeu de test n'est pas impliqué dans l'entraînement de l'Encodeur. La Figure 4.19 montre la matrice de confusion de l'Encodeur final sur le jeu de test.

Tableau 4.15 Performance du modèle Encodeur final sur le jeu de test

Exactitude moyenne, $\bar{A}$	
664/750 (88.53%)	
Classe	**Rappel, R**
0	143/270 (83.14%)
1	66/165 (61.68%)
2	116/116 (100%)
3	109/125 (87.2%)
4	121/121 (100%)
5	109/109 (100%)

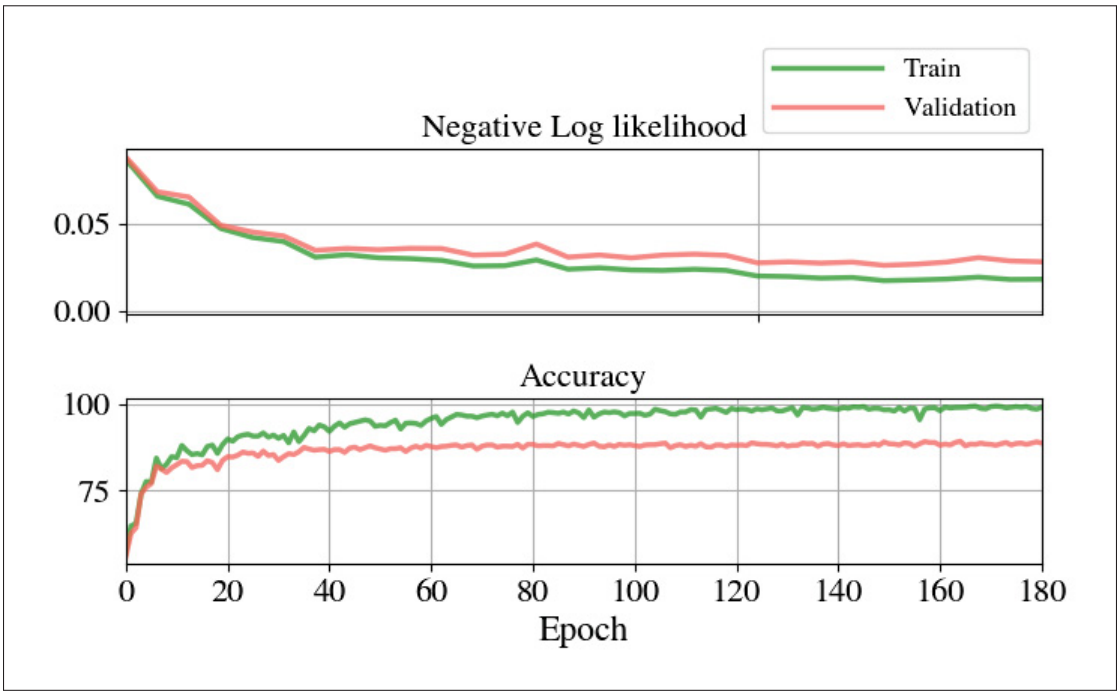


Figure 4.18 Courbe d'entraînement de l'encodeur final

Pour la majorité des classes, le modèle final produit des prédictions qui sont excellentes. La seule classe où le modèle a produit un rappel inférieur à 80% est la classe 1 représentant un défaut du roulement à billes. En moyenne, le modèle final connaît une exactitude moyenne de 88.53% sur le jeu de test, ce qui démontre une excellente efficacité.

En résumé, l'analyse des résultats montre que le modèle Encodeur présente la meilleure performance sur le jeu de validation avec des scores de 89.6% pour l'exactitude moyenne, 90.08% pour le rappel, 90.28% pour le F-score. L'Encodeur est aussi très performant en ce qui a trait à la détection des cas sans défaut dans le jeu de validation avec un rappel de la classe 0 à 87.8%. Sa performance sur le jeu de test est aussi très satisfaisante avec une exactitude moyenne de 88.53%, un rappel 88.67%. Sa capacité à détecter les cas sans défaut dans le jeu de test est de 83.14% ce qui est également très bon.

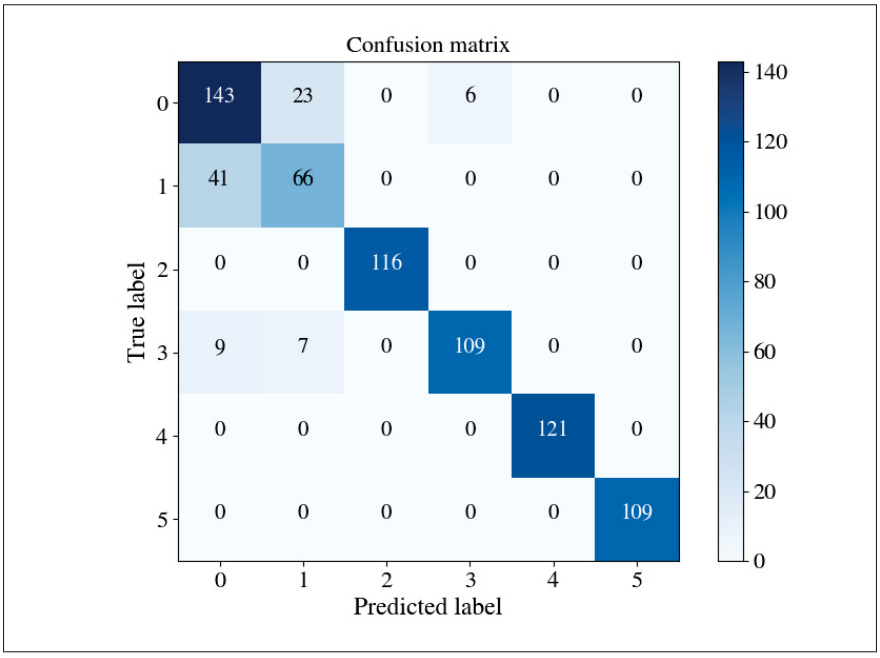


Figure 4.19 Matrice de confusion de l'Encodeur final sur le jeu de test

4.7 Conclusion

Ce chapitre est axé sur la méthodologie, le développement et les tests de différents modèles d'apprentissage profond. Chacun des modèles a pour rôle de classifier des défauts de moteurs PMDC.

Nous avons sélectionné quatre différents modèles prédictifs pour réaliser des essais. Ils sont : le FCN, le ResNet, l'Encodeur et le RNN. Pour chacun des modèles, nous avons décrit leur fonctionnement et la méthode d'entraînement utilisée.

Nous avons analysé la performance de ces modèles afin de sélectionner le modèle convenable. Dans cette étude, l'Encodeur s'avère le modèle le plus performant en termes de l'exactitude, la précision, le rappel et le F-score. Des ajustements manuels ont été appliqués aux hyperparamètres de l'Encodeur et la performance globale de ce dernier a pu atteindre une exactitude de 88.53% et un rappel de 88.67% sur le jeu de test. Plus encore, le modèle Encodeur était en mesure de reconnaître 83.14% des cas où il n'y a pas de défaut dans les séquences temporelle. Enfin,

le code pour l'entraînement et les tests des modèles sont disponibles à l'adresse suivante : https://github.com/alexchartrand/Fault-detection-in-IIoT.

CONCLUSION ET RECOMMANDATIONS

Ce projet de recherche est axé sur la valorisation des données dans un environnement industriel. Plus précisément, nous avons proposé une méthodologie propice à la détection des défauts servant à la maintenance prédictive. Dans cette méthodologie, quatre (4) grandes tâches doivent être réalisées. Elle sont : la collecte des données par simulation, la sélection des modèles prédictifs, la classification des défauts et l'évaluation de la performance des modèles prédictifs. La détection des défauts est une fonction importante de la maintenance prédictive. C'est dans ce contexte que nous avons formulé les objectifs de recherche suivants :

1. Proposer une approche permettant la classification des défauts des moteurs CC ;
2. Déterminer s'il est possible de classifier les défauts sans l'extraction explicite des caractéristiques ;
3. Examiner la performance des modèles prédictifs profonds appliqués à cette tâche.

Pour ce faire, nous avons développé un programme de simulation sous MATLAB et Simulink afin de produire les jeux de données nécessaires à l'entraînement des modèles prédictifs. La simulation sert à générer les signaux des moteurs PMDC intégrés dans une boucle de commande PID. Le temps d'enclenchement et le type des défauts sont des paramètres aléatoires du programme de simulation. Cinq (5) types de défaut sont programmés :

1. Problème de roulement ;
2. Défaut du capteur de vitesse ;
3. Défaut du capteur de courant ;
4. Défaut du capteur de tension ;
5. Déconnexion/court-circuit dans l'alimentation.

Au total, nous avons généré 9000 séries temporelles distinctes de 5 secondes chacune.

Nous avons entraîné et évalué quatre (4) modèles prédictifs. Ils sont : le FCN, le ResNet, l'Encodeur et le LSTM. Ces modèles ont été identifiés par une revue de la littérature portant sur la prédiction des séquences temporelles en apprentissage profond. L'évaluation de ces modèles est basée sur quatre (4) métriques de performance - l'exactitude moyenne, la précision, le rappel et le F-score. Nos résultats ont montré que le modèle Encodeur offre une performance supérieure aux autres modèles. L'ajustement des hyperparamètres du modèle Encodeur a permis d'obtenir une exactitude moyenne de 88.53%, ce qui est au delà de nos attentes.

5.1 Recommandations

La génération des données par la simulation est une avenue intéressante. Cette technique peut grandement simplifier la collecte des données impliquant des défauts ou défaillances des systèmes électromécaniques. Il est possible d'étendre son application à une chaîne de production plus complet comprenant des convoyeurs, des capteurs de proximité, des actionneurs et des équipements de contrôle. L'interaction entre les défauts et le comportement des équipements pourront alors être analysés d'une façon plus approfondie.

L'entraînement des modèles prédictifs est une tâche très exigeante en terme de calculs numériques. À l'heure actuelle et dans un avenir rapproché, la phase d'entraînement des modèles en apprentissage profond demeura sans doute dans l'arène des grappes d'ordinateurs et des cartes graphiques de haute performance. Quant à l'inférence produite, c'est-à-dire la prédiction des défauts par les modèles entraînés, une nouvelle approche se dessine à l'horizon. Il s'agit de l'informatique en périphérie (*Edge computing*). La «périphérie», dans cette approche, est située entre le réseau de terrain et le réseau de gestion d'une entreprise de production. Le caractère distinctif de cette approche est l'utilisation des dispositifs embarqués (*edge devices*) dans le traitement des données collectées localement. Ainsi, les modèles prédictifs entraînés peuvent

être exécutés à l'aide des dispositifs edge et ainsi produire des inférences sans avoir recours à une infrastructure infonuagique.

Les dispositifs edge sont des ordinateurs monocartes dotés de processeurs et ports d'entrée-sortie. Certains de ces dispositifs sont aussi munis de processeurs dédiés à des applications en apprentissage automatique. L'utilisation de ces dispositifs edge est convenable en milieu industriel puisqu'ils sont peu coûteux et ne nécessitent pas de manutention particulière. Or, pour pouvoir profiter de ces avantages, les modèles prédictifs doivent être modifiés en tenant compte des ressources limitées de ces dispositifs. L'impact de ces modifications sur la performance des modèles n'est pas encore élucidé.

LISTE DE RÉFÉRENCES

Ahmed, W. & Karim, A. (2020, 04). The Impact of Filter Size and Number of Filters on Classification Accuracy in CNN. pp. 88-93. doi : 10.1109/CSASE48920.2020.9142089.

Akhrif, O. (2019). SYS810 : Techniques de simulation. École de technologie supérieure.

Alom, M. Z., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M., Hasan, M., Essen, B., Awwal, A. & Asari, V. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, 8, 292. doi : 10.3390/electronics8030292.

Atzori, L., Iera, A. & Morabito, G. (2010). The Internet of Things : A survey. *Computer Networks*, 54(15), 2787 - 2805. doi : https://doi.org/10.1016/j.comnet.2010.05.010.

Auger, M. (2018). Coût de production et coût de revient : Un outil de prise de décision et d'amélioration de la rentabilité de votre entreprise. Repéré à https://www.mapaq.gouv.qc.ca/SiteCollectionDocuments/Regions/Outaouais/MartinAuger_Coutdeproductionetcoutderevient.pdf.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157-166.

Boehm, B. (2000). *Spiral Development : Experience, Principles, and Refinements*. Special Report. Software Engineering Institute. doi : CMU/SEI-2000-SR-008.

Buckley, I. & Fernandez, E. B. (2012, July 23-27). Failure Patterns : A New Way to Analyze Failures. *International Symposium on Software Architecture and Patterns*.

Chebudie, A. B., Minerva, R. & Rotondi, D. (2014). *Towards a definition of the Internet of Things (IoT)*. (Thèse de doctorat).

Cinar, Z. M., Abdussalam Nuhu, A., Zeeshan, Q., Korhan, O., Asmael, M. & Safaei, B. (2020). Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0. *Sustainability*, 12(19). doi : 10.3390/su12198211.

Dauphin, Y. & Cubuk, E. D. (2021). Deconstructing the Regularization of BatchNorm. *International Conference on Learning Representations*. Repéré à https://openreview.net/forum?id=d-XzF81Wg1.

Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.*, 7, 1–30.

en.wikibooks.org. (2019). *Control Systems*. MIT Press. Repéré à https://upload.wikimedia.org/wikipedia/commons/e/e4/Control_Systems.pdf.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press. Repéré à http://www.deeplearningbook.org.

Gómez, J. R., Quispe, E. C., Castrillón, R. d. P. & Viego, P. R. (2020). Identification of Technoeconomic Opportunities with the Use of Premium Efficiency Motors as Alternative for Developing Countries. *Energies*, 13(20). doi : 10.3390/en13205411.

Hapke, H. & Nelson, C. (2020). *Building Machine Learning Pipelines*. O'Reilly Media, Inc.

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026-1034. doi : 10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778. doi : 10.1109/CVPR.2016.90.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580. Repéré à http://arxiv.org/abs/1207.0580.

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, 9, 1735-80. doi : 10.1162/neco.1997.9.8.1735.

IEEE Motor reliabilty working group. (1985). Report of Large Motor Reliability Survey of Industrial and Commercial Installations, Part I. *IEEE Transactions on Industry Applications*, IA-21(4), 853-864. doi : 10.1109/TIA.1985.349532.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift.

ISA. (2016). Digitization improves manufacturing responsiveness, quality, and efficiency. Repéré à https://www.isa.org/intech-home/2016/may-june/features/industry-4-0-intelligent-and-flexible-production.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P.-A. (2019). Deep learning for time series classification : a review. *Data Mining and Knowledge Discovery*, 33(4), 917–963. doi : 10.1007/s10618-019-00619-1.

Jeschke, S., Brecher, C., Meisen, T., Özdemir, D. & Eschert, T. (2017). Industrial Internet of Things and Cyber Manufacturing Systems. Dans Jeschke, S., Brecher, C., Song, H. & Rawat, D. B. (Éds.), *Industrial Internet of Things : Cybermanufacturing Systems* (pp. 3–19). Cham : Springer International Publishing. doi : 10.1007/978-3-319-42559-7_1.

Johnston, S. J., Scott, M. & Cox, S. J. (2016, Dec). Recommendations for securing Internet of Things devices using commodity hardware. *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 307-310. doi : 10.1109/WF-IoT.2016.7845410.

Karim, F., Majumdar, S., Darabi, H. & Harford, S. (2019). Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116, 237–245. doi : 10.1016/j.neunet.2019.04.014.

Kingma, D. P. & Ba, J. (2014). Adam : A Method for Stochastic Optimization.

LeCun, Y., Bengio, Y. & Hinton, G. (2015/05/27). Deep learning. *Nature*, 521(7553), 436 - 44. Repéré à http://dx.doi.org/10.1038/nature14539.

Lin, M., Chen, Q. & Yan, S. (2013). Network In Network.

Maitre, J., Bouzouane, A. & Gaboury, S. (2018). A Hierarchical Approach for the Recognition of Induction Machine Failures. *Journal of Control, Automation and Electrical Systems*, 29(1), 44 - 61. Repéré à http://dx.doi.org/10.1007/s40313-017-0353-8.

Maitre, J. (2017). *Reconnaissance des défauts de la machine asynchrone : application des modèles d'intelligence artificielle*. (Thèse de doctorat, Université du Québec à Chicoutimi, 555 Boulevard de l'Université, Chicoutimi, QC).

Mattern, F. & Floerkemeier, C. (2010). From the Internet of Computers to the Internet of Things. Dans Sachs, K., Petrov, I. & Guerrero, P. (Éds.), *From Active Data Management to Event-Based Systems and More : Papers in Honor of Alejandro Buchmann on the Occasion of His 60th Birthday* (pp. 242–259). Berlin, Heidelberg : Springer Berlin Heidelberg. doi : 10.1007/978-3-642-17226-7_15.

Ministère de l'Économie et de l'Innovation du Québec. (2019). Guide et outils. Repéré à https://www.economie.gouv.qc.ca/bibliotheques/outils/gestion-dune-entreprise/production/maintenance/#c53709.

Ministère de l'Économie et de l'Innovation du Québec. (2021). Plan d'action en économie numérique : Feuille de route industrie 4.0. Repéré à https://www.economie.gouv.qc.ca/bibliotheques/outils/gestion-dune-entreprise/industrie-40/feuille-de-route-industrie-40/.

Mobley, R. K. (2002). Contents. Dans *An Introduction to Predictive Maintenance (Second Edition)* (éd. Second Edition, pp. v-xii). Burlington : Butterworth-Heinemann. doi : https://doi.org/10.1016/B978-075067531-4/50000-2.

Olah, C. (2015, 27 août). Understanding LSTM Networks [Billet de blogue]. Repéré à http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

OSA-CBM. MIMOSA - Open Standards for Physical Asset Management. Repéré à http://www.mimosa.org/mimosa-osa-cbm/.

Santos, L., Palhares, R., D'Angelo, M., Mendes, J., Veloso, R. & Ekel, P. (2018). A New Scheme for Fault Detection and Classification Applied to DC Motor. *TEMA (São Carlos)*, 19, 327 - 345. Repéré à http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2179-84512018000200327&nrm=iso.

Serrà, J., Pascual, S. & Karatzoglou, A. (2018). Towards a universal neural network encoder for time series. *CoRR*, abs/1805.03908. Repéré à http://arxiv.org/abs/1805.03908.

Shahgholian, G. & Shafaghi, P. (2010, May). State space modeling and eigenvalue analysis of the permanent magnet DC motor drive system. *2010 2nd International Conference on Electronic Computer Technology*, pp. 63-67. doi : 10.1109/ICECTECH.2010.5479987.

Statistique Canada. (2020). La consommation d'énergie du secteur de la fabrication a légèrement augmenté en 2019. Repéré à https://www150.statcan.gc.ca/n1/daily-quotidien/201125/dq201125f-fra.htm.

Thorsen, O. & Dalva, M. (1995). A survey of faults on induction motors in offshore oil industry, petrochemical industry, gas terminals, and oil refineries. *IEEE Transactions on Industry Applications*, 31(5), 1186-1196. doi : 10.1109/28.464536.

TrendEconomy. (2020). Annual International Trade Statistics by Country. Repéré à https://trendeconomy.com/data/h2/Canada/8501.

Ulyanov, D., Vedaldi, A. & Lempitsky, V. S. (2016). Instance Normalization : The Missing Ingredient for Fast Stylization. *CoRR*, abs/1607.08022. Repéré à http://arxiv.org/abs/1607.08022.

Union, I. T. (2015, Juillet, 20). Internet of Things Global Standards Initiative [Format]. Repéré à https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (NIPS'17), 6000–6010.

Wang, Z., Yan, W. & Oates, T. (2016). Time Series Classification from Scratch with Deep Neural Networks : A Strong Baseline. *CoRR*, abs/1611.06455. Repéré à http://arxiv.org/abs/1611.06455.

Xu, B., Wang, N., Chen, T. & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network.

Yadav, S. & Shukla, S. (2016). Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pp. 78-83. doi : 10.1109/IACC.2016.25.