UNIVERSITÉ DU QUÉBEC À MONTRÉAL

DÉVELOPPEMENT DE MODÈLES DE CATÉGORISATION ET DE PRÉDICTION DE TACHES DE PILOTAGE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

GABRIELLE JOYCE TATO NANA

SEPTEMBRE 2023

REMERCIEMENTS

Mes sincères remerciements s'adressent premièrement à tous ceux qui ont participé de près ou de loin à l'élaboration de ce mémoire.

Un remerciement spécial a mon directeur de recherche Roger Nkambou qui m'a permis de travailler sur un projet d'une aussi grande envergure et a accepté de me suivre tout au long de celui-ci. Je lui suis également reconnaissante pour son orientation de marque.

Je remercie l'équipe dynamique du GDAC, tous les membres du projet Pilote-IA, qui m'ont toujours soutenu et accordé de leurs temps lorsque j'en avais besoin.

Je remercie les experts industriels qui m'ont accompagnés tout au long du projet : Sylvain Therien (Bombardier) et Jocelyn Archambault (CAE).

Un grand merci à toute la famille TATO, particulièrement ma grande sœur Ange Tato qui a été une mentore de qualité et d'un soutien inestimable.

Un grand merci à tous mes collègues et amis pour m'avoir supporté pendant la réalisation de ce projet : Tchogna Junior Stéphane, Marc-Antoine Courtemanche, Massimo Pietracupa, Silatsa Carine et Guy Carlos.

TABLE DES MATIÈRES

LISTE DES TABLEAUX vi
LISTE DES FIGURES vii
RÉSUMÉ x
CHAPITRE I INTRODUCTION 1
1.1 Contexte 1
1.2 Problématique 3
1.3 Objectifs 5
1.4 Méthodologie 6
1.5 Organisation du mémoire 8
CHAPITRE II REVUE DE LA LITTÉRATURE 9
2.1 Notions de profil comportemental de pilotage 9
2.1.1 Gain du pilote 9
2.1.2 Mesure du gain de pilote 12
2.2 L'apprentissage automatique dans l'aéronautique 15
2.2.1 Cas d'application des modèles de regroupement 15
2.2.2 Les modèles appris 17
2.3 Les séries temporelles et les algorithmes d'apprentissage machine . . 18
2.3.1 Définition 18
2.3.2 Regroupement des séries temporelles 19
2.3.3 Réseaux de neurones récurrents : cas des LSTM 23
2.3.4 Les transformeurs 26
2.4 Conclusion 27
CHAPITRE III UNE APPROCHE POUR LE PROFILAGE COMPORTEMENTAL DE PILOTAGE 29

3.1 Les données 29
3.1.1 Sélection des variables 30
3.1.2 Normalisation 30
3.2 Expérience 1 : Regroupement des données globales de décollages . . . 31
3.2.1 Approche de réduction de dimensions 31
3.2.2 Autoencodeur LSTM 33
3.2.3 K-moyennes pour les séries chronologiques 39
3.2.4 Analyse des résultats 41
3.3 Expérience 2 : Regroupement de pilotes sur les séquences de décollages 42
3.3.1 Séquençage de données 42
3.3.2 Regroupement des séries chronologiques 45
3.4 Conclusion 49
CHAPITRE IV MODÈLES APPRIS DES COMPORTEMENTS DE PILOTAGE 51
4.1 Implémentation 51
4.1.1 Segment5 : Procédure du *pitch stick* pour contrer l'effet du réglage de la poussée du décollage (procédure spécifique à Airbus) 52
4.1.2 Segment 6 : Rotation 56
4.1.3 Segment 7 : Décollage 59
4.2 Résultats et discussions 63
4.2.1 Segment5 63
4.2.2 Segment6 64
4.2.3 Segment7 66
4.3 Conclusion 68
CHAPITRE V VERS UN CADRE D'EXÉCUTION DU MODÈLE APPRIS 69
5.1 Interfaçage avec X-Plane 70
5.1.1 Définition 70

5.1.2 API de communication avec X-Plane 70
5.2 X-Plane Runner 71
5.2.1 Architecture 71
5.3 Exécution du modèle appris 77
5.3.1 Mise en correspondance des paramètres des différents modèles 77
5.3.2 Expérience 1 : tests indépendants 80
5.3.3 Expérience 2 : test complet du Runner 82
5.3.4 Résultats 86
5.4 Conclusion 86
CHAPITRE VI CONCLUSION 88
6.1 Limitations 89
6.2 Travaux futurs 90
APPENDICE A RÉDUCTION DE DIMENSION 91
APPENDICE B TABLEAU DE RÉFÉRENCE 92
APPENDICE C AUTRES EXEMPLES DE PRÉDICTION 93
C.1 Model 5 93
C.2 Model 7 94
APPENDICE D ONTOLOGIE DE TACHE 97
APPENDICE E FICHIER DE MAPPING 98
APPENDICE F PROCÉDURE DE MISE EN MARCHE POUR LE DÉCOLLAGE 105
APPENDICE G PLUGIN ULTIME AIRBUS A320 107
RÉFÉRENCES 109

LISTE DES TABLEAUX

Tableau Page

3.1 Scores silhouette 37

3.2 Scores silhouette pour les séries chronologiques. 39

5.1 Entrées du modèle appris du segment 5. 77

5.2 Sortie du modèle 5 78

5.3 Entrées du modèle appris du segment 6 78

5.4 Sortie du modèle appris du segment 6 79

5.5 Entrées du modèle appris du segment 7 79

5.6 Sorties du modèle appris du segment 7 80

5.7 Sorties du modèle appris du segment 7 80

LISTE DES FIGURES

Figure Page

2.1 Exemple de trace PSD, source : (Niewind, 2012) 14

2.2 Combinaison d'algorithmes de *clustering*, source :(Mangortey *et al.*, 2020) 16

2.3 Regroupement des données temporelles, source :(Liao, 2005) . . . 19

2.4 Mesure de similarité avec DTW, source : (https://paperswithcode.com/method/dtw) 21

2.5 Fonctionnement d'un LSTM, source : (https://france.devoteam.com/paroles-dexperts/aller-plus-loin-en-deep-learning-avec-les-reseaux-de-neurones-recurrents-rnns/). 24

2.6 Autoencodeur, source : (https://jafwin.com/2020/07/04/les-auto-encodeurs-expliques-en-detail/) 25

2.7 Architecture d'un Autoencodeur LSTM, souce : (https://machinelearningmastery.com/lstm autoencoders/) 26

3.1 Temps des décollage de pilotes. 32

3.2 Exemple de sous-échantillonnage de la valeur du paramètre vent arrière. 33

3.3 Architecture de l'autoencodeur LSTM poposée. 34

3.4 Courbe de perte. 35

3.5 Tracé des distorsions 36

3.6 Groupe 1 obtenue de la méthode de réduction de dimensions (LSTM) suivie du K-moyennes 37

3.7 Groupe 2 obtenue de la méthode de réduction de dimensions (LSTM) suivie du K-moyennes 38

3.8 Superposition des 2 groupes obtenus de la méthode de réduction de dimensions (LSTM) suivie du K-moyennes 38

3.9 Groupe 1 obtenue par la méthode des K-moyennes pour les séries chronologiques 40

3.10 Groupe 2 obtenue par la méthode des K-moyennes pour les séries chronologiques 40

3.11 Superposition des 2 clusters 41

3.12 Les groupes extraits du segment 5 46

3.13 Les groupes extraits du segment 6 47

3.14 Segment 7 : groupe 1. 48

3.15 Segment 7 : groupe 2 49

3.16 Superposition des centroïdes des groupes 1 et 2 du segment 7 . . . 49

4.1 Durée mise par les pilotes pour effectuer le segment 5. 53

4.2 Illustration du modèle appris sur le segment 5. 54

4.3 Architecture du modèle appris du segment 5. 55

4.4 Durée mise par les pilotes pour effectuer le segment 6 57

4.5 Illustration du modèle appris sur le segment 6. 58

4.6 Architecture du modèle appris sur le segment 6. 58

4.7 Durée mise par les pilotes pour effectuer le segment 7 60

4.8 Illustration du modèle appris sur le segment 7 61

4.9 Architecture du modèle appris sur le segment 7 62

4.10 Exemple de prédiction sur le segment 5. 63

4.11 Exemple 1 de prédiction sur le segment 6. 65

4.12 Exemple 2 de prédiction sur le segment 6. 65

4.13 Exemple de prédiction sur le segment 7. 67

5.1 Architecture du runner. 72

5.2 Représentation simplifiée de l'ontologie du domaine. 73

5.3 Architecture finale du modèle appris. 76
5.4 Initialisation de la classe 83
5.5 Fonction de prédiction de comportement 83
5.6 Connexion du runner 84
5.7 Chargement des ontologies 85
5.8 Boucle de simulation 85
5.9 Intégration de modèles appris 86
5.10 Environement de simulation 87

RÉSUMÉ

L'aéronautique est une industrie dynamique et performante ayant pour objectif l'évolution et la maintenance des appareils volants. Cette industrie est classée première dans l'économie européenne, car elle mobilise une grande quantité et diversité de talents. L'aéronautique regroupe les activités de conception, de fabrication et de commercialisation des aéronefs (avions, hélicoptères, drones, etc.) et des équipements spécifiques associés (propulsion, systèmes de bord, etc.), mais aussi le transport commercial, civil et militaire. Une dimension importante de la sécurité du transport aérien est l'enregistrement des données de vol. Ces données peuvent être de plusieurs types : activités et comportements du pilote (pression des freins, direction de l'avion, etc.), comportement de l'avion (vitesse indiquée, angle de chute), communications avec la tour de contrôle, etc. Certaines données varient tout au long du vol alors que d'autres restent constantes. Il peut également exister des paramètres environnementaux, c'est-à-dire extérieurs au comportement de l'avion ou des pilotes tels que la vitesse et l'orientation du vent qui peut être de travers ou en arrière, etc. L'analyse de ces données permet de décrire comment le pilote agit sur les manettes pendant le vol, ce qu'on appellera gain du pilote.

Une perspective importante de l'apprentissage automatique est l'automatisation du développement de modèles analytiques et comportementaux basés sur des algorithmes d'optimisation. Les réseaux neuronaux, une sous-branche de l'apprentissage automatique, ont été largement utilisés pour les problèmes de généralisation. Ils peuvent apprendre des modèles cachés/latents à partir des données. Il est donc possible de les utiliser pour apprendre des comportements humains à partir des données représentant de nombreux comportements, tels que ceux liés au pilotage d'un avion.

Cette recherche traite tout d'abord de l'apprentissage automatique pour l'analyse des comportements de pilotage afin de mettre en évidence des profils comportementaux de pilotage (gain du pilote) : "gain élevé / gain faible" et plus s'il y en a. Ensuite, le développement d'un modèle d'apprentissage des actions de pilotage plus précisément, les actions effectuées par les pilotes au décollage d'un Airbus 320. Ce modèle utilise les réseaux de neurones récurrents. Enfin, cette recherche vise le développement d'un pilote synthétique capable d'exécuter le modèle appris des actions de pilotage dans le simulateur X-Plane. Cette exécution constitue un cadre de validation des modèles développés.

Mots clés : apprentissage automatique, réseaux de neurones, gain du pilote, pilote synthétique, modèle appris, aéronautique.

CHAPITRE I

INTRODUCTION

Ce travail est réalisé dans le cadre d'un projet de recherche au sein du laboratoire GDAC de l' Université du Québec à Montréal et une équipe de recherche à l'Université de Montréal, ce en partenariat avec CAE, Bombardier et BMU(Beam Me Up). Ce projet est financé par le Consortium de recherche et d'innovation en aérospatiale au Québec (CRIAQ) et le conseil de recherches en sciences naturelles et en génie du Canada (CRSNG).

1.1 Contexte

L'aéronautique est un ensemble de techniques et de sciences utilisées dans la conception, la construction et l'exploitation des appareils capables de voler. Il s'agit d'un domaine strict dans ses normes et réglementations en matière de sécurité, maintenance et exploitation des aéronefs, un secteur économique important et un domaine de la science et de la technologie qui a connu de nombreuses avancées au fil des siècles, notamment depuis les ballons dirigeables de Henri Giffard en 1852. Cependant, c'est au cours du 19e siècle qu'il a connu un véritable développement avec l'apparition de nombreux pionniers du vol tels que les frères Wright qui sont les inventeurs de l'avion motorisé et ont réalisé le premier vol en avion en 1903 (Cregger *et al.*, 2022). Tout comme l'aéronautique, les avions ont eux aussi

connu de nombreux développements au cours de l'histoire. On peut entre autres citer :

- les avions à ailes fixes qui ont été les premiers types d'avions développés ne pouvant pas être inclinés pour effectuer des virages ;
- les avions à ailes volantes qui ont été développés au début du 20e siècle et ont permis des performances aériennes nettement améliorées grâce à leur capacité à incliner leurs ailes pour effectuer des virages. Ils sont devenus populaires pour l'aviation militaire et de loisirs (Brunt *et al.*, 2019) ;
- les avions à réaction permettant des vitesses beaucoup plus évoluées grâce à l'utilisation des moteurs à réaction.
- les avions à décollage et atterrissage verticaux (VTOL) conçus pour des décollages et atterrissages verticaux permettant de s'adapter à des terrains étroits ;
- les avions à décollage et atterrissage courts (STOL) conçus pour des décollages et atterrissages verticaux permettant de s'adapter aux aéroports de petite taille ;
- avions à fuselage incliné permettant de décoller et d'atterrir à des angles plus rapides que les avions conventionnels.

Les acteurs du secteur de l'aéronautique peuvent être regroupés en quatre grandes catégories : les constructeurs aéronautiques (ex : Airbus, Boeing, Bombardier, etc.), les fournisseurs de pièces et de composants (ex : Pratt & Whitney, Rolls-Royce), les formateurs sur simulateur (ex : CAE), et les compagnies aériennes (ex : Air Canada). En termes économiques, l'aéronautique est un secteur qui génère des revenus considérables et contribue à la croissance économique de nombreux pays. Il a généré un chiffre d'affaires de 833 milliards de dollars en 2018. Il est également un employeur important, avec environ 64 millions d'emplois directement liés au secteur dans le monde. L'aéronautique est un domaine en constante évolution qui fait face à de nombreux défis notamment :

— L'environnement : Selon Une etude [1], l'aviation représente environ 2,5 pourcents des émissions de gaz à effet de serre dans le monde, et ce chiffre devrait doubler d'ici 2050 si des mesures ne sont pas prises pour réduire les émissions.

— La sécurité : il y a de nombreux enjeux de sécurité auxquels l'industrie doit faire face, tels que les accidents d'avion, les actes de sabotage et les menaces terroristes (FDA [2]).

— La fiabilité : c'est un autre défi important pour l'industrie aéronautique. Les avions doivent être fiables pour assurer la sécurité des passagers et pour minimiser les retards et les annulations de vols, qui peuvent avoir un impact économique important sur les compagnies aériennes et leurs clients.

— La formation des pilotes : l'un des principaux défis est la pénurie des pilotes qualifiés. Selon l'organisation de l'aviation civile internationale (OACI), il y aura besoin de 790 000 nouveaux pilotes d'ici 2037 pour répondre à la croissance de la demande de voyage aérien. Cette pénurie de pilotes est due à un certain nombre de facteurs, notamment le vieillissement de la population de pilotes, la réduction du nombre de personnes intéressées par une carrière de pilote et les exigences croissantes en matière de formation et de qualifications.

1.2 Problématique

Des ingénieurs d'essais en vol expérimentés vous diront qu'en ce qui concerne les pilotes d'essai, il en existe deux types : à faible gain (*low gain*) et à haut gain (*high gain*). En terme technique, le gain est ce que les ingénieurs appellent le

1. https ://ourworldindata.org/co2-emissions-from-aviation

2. Fédéral Aviation Administration, 2018

rapport entre la réponse et l'erreur. Le gain du pilote décrit le niveau d'agressivité dans l'activité de contrôle de l'avion. Cela dépend de plusieurs facteurs dont la dynamique de l'avion, la tâche à accomplir, le niveau de stress, la personnalité, etc (Hess, 2006). En règle générale, le pilote à faible gain effectue des entrées/actions considérées comme plus lisses, et généralement à des amplitudes plus petites que le pilote à gain élevé. Chaque professionnel chevronné des essais en vol est capable d'identifier les pilotes d'essai qui sont considérés être des pilotes à gain élevé ou à gain faible (Mitchell *et al.*, 1990).

Les pilotes humains sont formés pour gérer les incertitudes de vol ou les situations d'urgence telles que des conditions météorologiques extrêmes ou une défaillance du système. En revanche, les systèmes de contrôle de vol automatique ou pilote automatique, sont très limités, capables d'effectuer des tâches de pilotage minimales dans des conditions simples (non urgentes) (Baomar et Bentley, 2016).

Le problème actuel des systèmes de contrôle automatique de vol est lié à la sécurité. Bien que ces systèmes aient considérablement amélioré la sécurité de l'aviation, ils sont dans l'incapacité de gérer des conditions météorologiques extrêmes par exemple.

En ce qui concerne la formation des pilotes, en plus du problème de pénurie des pilotes qualifiés (cette situation qui est exacerbée par le départ à la retraite des pilotes expérimentés et la difficulté de recruter de nouveaux candidats), il y a aussi la difficulté de la formation en vol réel. Elle est due à la rareté d'une assistance efficace et constante des élèves pilotes.

Il également aussi très important de pouvoir différencier les profils *high gain* des profils *low gain* car il s'agit des attitudes caractérisant individuellement le pilote. Selon une étude publiée dans le Journal of Aviation/Aerospace Education &

Research [3], les programmes de formation qui sont adaptés aux besoins et aux compétences individuels des pilotes (en particulier les différences culturelles) peuvent être plus efficaces. Ces différences comportementales peuvent s'étendre au gain de pilotage. Par exemple, les pilotes à faible gain pourraient bénéficier de programmes de formation qui mettent l'accent sur la pratique et la répétition pour renforcer leur confiance et leurs compétences, tandis que les pilotes à gain élevé pourraient bénéficier de programmes de formation qui leur offrent de nouvelles expériences de vol et des défis pour continuer à développer leurs compétences. Selon les ingénieurs d'essai en vol que nous avons rencontrés ainsi que certaines études (Niewind, 2011), cette différenciation de gain est un des aspects les plus importants dans la planification des essais en vol. Dès lors, comment établir les différentes catégories de gain sachant que la littérature ne se limite pas au gain élevé et faible mais peut parfois inclure le gain standard comme c'est le cas dans (Gilbreath, 2001) ? Comment pouvons-nous identifier de façon automatique le gain d'un pilote étant donné son pilotage ? Le niveau de gain d'un pilote est-il absolu ou pourrait-il varier selon l'activité en cours ? Comment pouvons-nous extraire la connaissance des pilotes qualifiés afin d'assister efficacement les élèves pilotes ou d'en dégager un modèle de pilotage synthétique (modèle appris) ? Dans quel cadre et comment pourront s'exécuter ces modèles ? Ce mémoire traite de ces questions.

1.3 Objectifs

Cette recherche vise d'une part à identifier de façon automatique les classes de profil de pilotage et d'autre part à extraire des comportements de pilotage à partir de données réelles de vol. L'idée est de pouvoir produire des modèles qui simulent le pilotage d'un avion, potentiellement utiles pour la formation, l'assistance aux

3. https ://commons.erau.edu/jaaer/vol27/iss2/1/

pilotes et à la conception de poste de pilotage. Pour y arriver, nous devons atteindre les objectifs suivants :

1. identifier de façon automatique le gain d'un pilote en prenant appui sur la littérature et sur les experts du domaine ;
2. développer un modèle d'apprentissage automatique dit modèle appris (qui apprend des comportements de pilotage à partir des données réelles de vols) capable de prédire les actions de pilotes au fil du temps et selon différents segments de décollage ;
3. effectuer les multiples validations des modèles obtenues avec les experts ;
4. situer le modèle appris dans un cadre itératif (cadre d'exécution des modèles dans le simulateur de vol X-Plane) afin de tester et de valider celui-ci.

1.4 Méthodologie

L'étude se concentre sur la tâche de décollage, un choix suggéré par les experts du domaine vu le caractère critique de cette phase dans un vol, l'ampleur des données disponibles et le temps imparti pour la réalisation du projet. Selon les données de la Federal Aviation Administration (FAA) des États-Unis, les accidents liés au décollage représentent environ 14% de tous les accidents d'aviation commerciale. L'idée est bien sûr d'aboutir à terme à un modèle qui couvre un vol complet. Nous utilisons dans ce travail les données de décollages en séries chronologiques, recueillis à partir des simulateurs de vol complets de niveau D de CAE sur des avions de type Airbus 320 (A320).

Afin d'atteindre nos objectifs, nous avons premièrement fait le choix de nos architectures. Une comparaison minutieuse a été faite entre les différents algorithmes d'apprentissage automatique pour les séries temporelles d'abord pour le regroupement des données pour des fins de profilage de pilotes, ensuite pour la prédiction

des comportements de pilotage (modèle appris), et enfin pour l'exécution des modèles développés à travers un simulateur.

Par ailleurs, les données collectées furent nettoyées et structurées en vue de l'étape suivante du traitement des données. C'est une tâche difficile et compliquée. Parfois, le prétraitement des données lui-même peut prendre plus de la moitié du temps total consacré à la résolution du problème de l'exploration de données (Bienkowski *et al.*, 2012). Le prétraitement implique la sélection des attributs, le nettoyage, la normalisation des données, etc. Il est suivi de la construction des différents modèles : 1) Le modèle de regroupement (*clustering*) des comportements de pilotages en vue d'extraire des profils comportementaux de pilotages ; 2) les modèles de prédiction des actions de pilotage dans le temps ou encore le modèle appris (qui regroupe plusieurs modèles de prédiction), grâce aux comportements appris des données.

La dernière étape de notre méthodologie concerne la validation des modèles développés en deux phases : en premier lieu, une validation faite avec l'aide des experts du domaine pour l'étiquetage des différents profils comportementaux obtenus par regroupement, suivi d'une seconde étape pour discuter et critiquer les différentes prédictions (présentées sous forme de graphes) faites par les modèles. Bien qu'une validation sera faite avec les experts et également en utilisant les méthodes d'évaluation de modèles telles que le L'erreur quadratique moyen(RMSE) ou l'erreur absolue moyenne(MAE), une troisième étape de la validation consiste à créer un cadre itératif d'exécution des modèles en utilisant un modèle de référence (sous forme d'ontologie) et un simulateur de vol comme cadre d'exécution (X-Plane dans notre cas).

1.5 Organisation du mémoire

Ce mémoire comporte 6 chapitres dont le premier est l'introduction. Étant donné la problématique et les objectifs fixés plus haut, l'organisation des autres chapitres est la suivante :

Le chapitre 2 représente le cadre théorique de notre mémoire. Ce chapitre fait tout d'abord un état de l'art sur les différents profils de pilotages tel que considéré dans le domaine de l'aéronautique. Nous nous intéressons ensuite aux travaux en apprentissage automatique dans l'aéronautique, particulièrement les regroupements et prédictions d'actions de pilotage. Enfin nous explorons les algorithmes d'ap-apprentissage automatique pour les séries chronologiques.

Le chapitre 3 présente les différentes expériences faites, les analyses et les conclusions qui en découlent après concertation des experts en matière de profils de pilotage.

Le chapitre 4 présente les résultats des différentes expériences sur la prédiction dans le temps des comportements de pilotages par les modèles appris développés sur les différents segments de décollages de différents pilotes dans un cadre non itératif.

Le chapitre 5 présente le cadre d'exécution (boucle de simulation) du modèle appris dans le simulateur X-Plane en utilisant une API développée pour l'occasion permettant la connexion entre les modèles et le simulateur avec quelques tests et validations effectués.

Pour conclure, le chapitre 6 fait le bilan général du travail effectué et expose les perspectives et les travaux futurs que nous visons pour le projet.

CHAPITRE II

REVUE DE LA LITTÉRATURE

2.1 Notions de profil comportemental de pilotage

En raison de l'augmentation de la complexité des systèmes de vol et de l'importance croissante de la sécurité en aviation, un accent a été mis sur la dynamique du pilote. Les avions modernes sont devenus plus sophistiqués et plus automatisés, ce qui a entraîné des exigences accrues en matière de compétences et de formation pour les pilotes. Les systèmes de navigation et de communication ont également évolué rapidement, ce qui a rendu nécessaire une meilleure compréhension de la dynamique de vol pour les pilotes.

2.1.1 Gain du pilote

Définition

Le terme *gain du pilote* décrit la manière dont le pilote agit sur les manettes pendant le vol. Il n'y a pas de définition universellement acceptée du gain du pilote. Souvent, d'autres termes sont utilisés dans la littérature pour décrire le phénomène (Niewind, 2011). Selon une étude menée par Niewind (Niewind, 2011), dans le milieu des essais en vol, le gain d'un pilote décrit en général son niveau d'agressivité dans l'exercice de sa fonction. Le gain est dit faible pour un pilote

moins agressif et élevé pour un pilote plus agressif. «Bien que ce soit un moyen de combiner le gain du pilote avec l'agressivité, la plupart du temps, l'agressivité est associée à la violence et donc évaluée comme un mauvais comportement alors que le gain du pilote élevé ou faible a la même valeur et n'est pas lié à une bonne ou une mauvaise performance ».(Simm, 2012)

Une étude sur les manœuvres du pilote (Mitchell *et al.*, 1990) montre plusieurs tendances entre différents profils de pilotes :

- un pilote H dit à « *low gain* » avait généralement des fréquences de croisement plus faibles et les erreurs de suivi de l'axe de roulis et de l'axe de tangage les plus élevés que la majorité des pilotes.
- un pilote B dit à « *high gain* » avait généralement les fréquences de croisement les plus élevées et les erreurs de suivi de l'axe de roulis et l'axe de tangage les plus faibles.
- un pilote V faisait généralement partie du groupe «*low gain*» pour le suivi de l'axe de tangage, mais du groupe «*high gain*» pour le suivi de l'axe de roulis. Difficile alors de déterminer si un pilote est réellement un «*high gain*» ou un «*low gain*» absolu.

Une autre étude (Klyde *et al.*, 2001) a montré qu'un pilote peut changer de profil dépendamment de la dynamique de l'avion. En effet, la mesure du gain de pilote est un processus complexe qui dépend des multiples facteurs tels que le contexte, l'environnement, l'état de l'avion, l'urgence d'une situation par exemple qui peuvent considérablement varier d'un lieu à un autre ou d'un moment à un autre ; ce qui rend difficile l'obtention d'une mesure fiable et précise du gain du pilote. « La mesure numérique du gain du pilote a toujours été une sorte d'art noir » (Gray, 2007). Les recherches telles que (Niewind, 2011) et (Niewind, 2012) ont tenté d'enquêter sur la notion ou la définition du gain de pilote. Il en ressort que le gain du pilote inclut la charge de travail. La charge de travail des pilotes

peut avoir un impact sur leur performance et leur capacité à gérer les tâches de pilotage. Si la charge de travail est trop élevée, les pilotes peuvent avoir des difficultés à gérer les tâches de vol complexes. Il est aussi à noter que les traits de personnalité humaine jouent un rôle dans la détermination des différences de gain de pilote. Dès lors, un gain élevé ne signifie pas forcément que la charge de travail est élevée. Certains pilotes peuvent être de gain élevé sans nécessairement subir une grande charge de travail.

Il est important de souligner dans la mesure du gain qu'il n'y a fort probablement pas une variable dans le cerveau d'un pilote qui pourrait directement se traduire par l'amplitude de « gain » (Gray, 2009). En effet, la performance d'un pilote dépend de plusieurs facteurs tels que les compétences de pilotage, la capacité à gérer la charge de travail, la capacité à prendre des décisions en situation de crise, et bien d'autres. Ces facteurs sont interconnectés et interdépendants.

Les oscillations induites par le pilote (PIO) sont des « mouvements d'attitude et de trajectoire de vol involontaires et indésirables de l'aéronef qui proviennent d'interactions anormales entre l'aéronef et le pilote »(Van der Weerd, 2000). Ces oscillations peuvent être causées par des erreurs de pilotage telles que des réactions excessivement brusques aux commandes de vol ou des incohérences dans les commandes de vol (Hansen *et al.*, 2004). Cette notion est donc étroitement liée à la notion de gain du pilote. Le pilote peut arrêter son PIO en réduisant son gain (Moormann, 2000).

Gain élevé et gain faible

Le gain du pilote est une question de tâche, de formation, de dynamique de l'avion et de niveau de stress actuel, mais c'est aussi une question de disposition individuelle. Dans presque toutes les organisations d'essais en vol, une distinction est

faite entre un pilote à faible gain et un pilote à haut gain (Mitchell et Klyde, 2019). Les deux dispositions ont leurs avantages et leurs inconvénients et aucune n'est généralement supérieure à l'autre. Plus le gain du pilote est faible, plus le système pilote-véhicule ressemble à la dynamique stable de l'avion ; plus le gain du pilote est élevé, moins le système pilote-véhicule est stable. C'est pourquoi les pilotes à gain élevé ont tendance à trouver une dynamique d'aéronef défavorable qu'un pilote à faible gain ne peut rencontrer qu'en situation d'urgence. Il existe plusieurs tâches à gain élevé qui sont généralement mentionnées pour évoquer les PIO : ravitaillement aérien, vol en formation, suivi de précision, approches de précision et atterrissages ponctuels (par exemple : approche de transporteur), suivi du terrain, transitions exigeantes ou inattendues (McRuer, 1995). Un contrôle précis, des réactions rapides et la recherche d'une faible tolérance aux erreurs sont caractéristiques. Les taches typiques à faible gain quant à elles sont à basse fréquence, douces, et demandent peu de corrections. Les approches ILS (système d'atterrissage aux instruments) sont un bon exemple.(Bauschat *et al.*, 2004)

2.1.2 Mesure du gain de pilote

On retrouve trois classes de mesures potentielles de gain de pilote : basées sur le domaine temporel, le domaine fréquentiel et sur le modèle pilote.

1. Mesures basées sur le domaine temporel

   L'enquête menée auprès des pilotes dans l'étude (Niewind, 2011) révèle que les pilotes préfèrent les mesures basées sur le domaine temporel comme la déviation du manche, la vitesse du manche, l'accélération du manche et le nombre d'entrées du manche. Ces paramètres sont vécus dans la vraie vie et sont donc plus tangibles. La déviation est directement déduite du simulateur tant dit que les deux autres sont calculés. Les paramètres recommandés dans (Niewind, 2013) pour la mesure du gain de pilote sur le

domaine temporel sont :

— la vitesse moyenne du manche (ou '*Mean Stick Speed*' en anglais) est une mesure de la vitesse à laquelle un pilote bouge le manche de commande de l'avion. Une vitesse moyenne élevée du manche peut indiquer une réaction excessivement brusque aux commandes de vol, ce qui peut entraîner des oscillations de vol. En revanche, une vitesse moyenne faible du manche peut indiquer un manque de réactivité ou de précision dans les commandes de vol.

— la vitesse du manche RMS (*Root Mean Square*) : Cette mesure utilise la racine carrée de la moyenne des carrés de la vitesse du manche pour donner une valeur qui reflète la variabilité de la vitesse du manche, ce qui est une mesure plus précise que la vitesse moyenne.

— pourcentage de vitesses de manche élevées : C'est une mesure qui indique la fréquence à laquelle un pilote utilise des mouvements brusques et rapides du manche de commande de l'avion.

— accélération moyenne du manche : c'est une mesure de la vitesse à laquelle un pilote change la position du manche de commande de l'avion.

— Pourcentage d'accélérations élevées du manche

2. Mesures basées sur le domaine fréquentiel : PSD

   L'étude faite par Belyavin (Belyavin *et al.*, 2005) énumère plusieurs possibilités générales de mesure de gain pilote, dont la densité spectrale de puissance (PSD) qui est généralement utilisée pour identifier les fréquences dominantes dans les signaux de commande de vol, ce qui permet de caractériser la bande passante de ces signaux. Les tracés de PSD sont généralement représentés sous la forme d'un graphique à barres ou d'un graphique en ligne. Leur axe horizontal représente la fréquence, et l'axe vertical représente la puissance du signal à cette fréquence. La figure 2.1 montre un

exemple de trace PSD. Les pics de PSD entre 0 Hz et de la tache la plus élevée (HTF) 0,7 Hz sont nets et définis. Les pics de PSD dans la zone au-delà du HTF (0,7 Hz) sont plus dispersés et répartis sur une plage de fréquences. Ils sont généralement le résultat soit d'un gain élevé du pilote, soit de défauts dans la dynamique de l'avion (ou les deux).

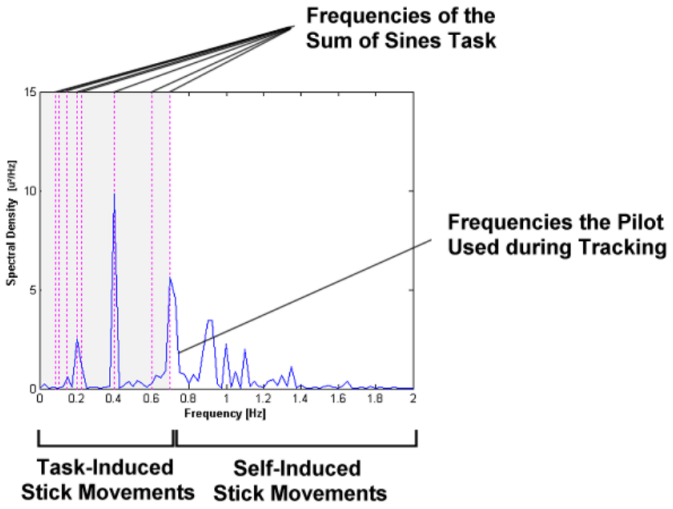


FIGURE 2.1 Exemple de trace PSD, source : (Niewind, 2012)

Lorsque la dynamique de l'avion est maintenue constante, le gain du pilote est le principal facteur contribuant aux différences dans ce domaine.

3. Mesures basées sur le modèle pilote

   Les recherches de Niewind en 2012 (Niewind, 2012) ont évalué deux modèles pilotes différents, mais un seul modèle a créé des données pouvant être utilisées comme mesure de gain pilote. Ce modèle est identique au modèle utilisé dans le critère de (Neal et Smith, 1971). Il tient compte de la compensation d'avance et de retard appliquée par le pilote.

2.2 L'apprentissage automatique dans l'aéronautique

2.2.1 Cas d'application des modèles de regroupement

Plusieurs recherches ont mis en exergue le regroupement ou *clustering* en aéronautique pour détecter les anomalies de vols. Dans ce sens, l'étude (Li *et al.*, 2015) présente une nouvelle méthode (ClusterAD-Flight) de détection d'anomalies basée sur des grappes pour détecter les vols anormaux qui peut aider les experts du domaine à détecter les anomalies et les risques associés liés aux opérations de routine des compagnies. Les tests ont été menés à l'aide de deux ensembles de données opérationnelles comprenant 365 vols de Boeing 777 et 25 519 vols d'Airbus 320.(Sheridan *et al.*, 2020) a fait de même en utilisant *Density-Based Clustering Spatial* (DBCAN) qui s'appuie sur la densité estimée des clusters pour détecter les vols anormaux.

En plus de la détection d'anomalies de vols, les recherches dans (Mangortey *et al.*, 2020) rajoutent à cela le développement d'une méthodologie robuste pour identifier les paramètres qui sont révélateurs de ces anomalies. Les expériences ont été faites sur des données constituées de 5000 vols. Après le prétraitement et la réduction des dimensions de données (à l'aide de l'outil de visualisation des données de grande dimension t-SNE), les chercheurs ont testé plusieurs algorithmes de *clustering* (figure 2.2) dans le but d'obtenir les meilleures conclusions.

Chaque métrique avait un classement différent. Pour le meilleur score de connectivité, l'algorithme de *clustering* hiérarchique avec deux clusters était optimal. Pour le meilleur indice de Dunn, l'algorithme de clustering hiérarchique avec huit clusters était optimal. Pour le meilleur score de Silhouette, l'algorithme de regroupement K-moyennes avec 8 groupes était optimal. Les tableaux de bord de prise de décision ont été utilisés pour décider quels résultats analyser en détail. Ils ont

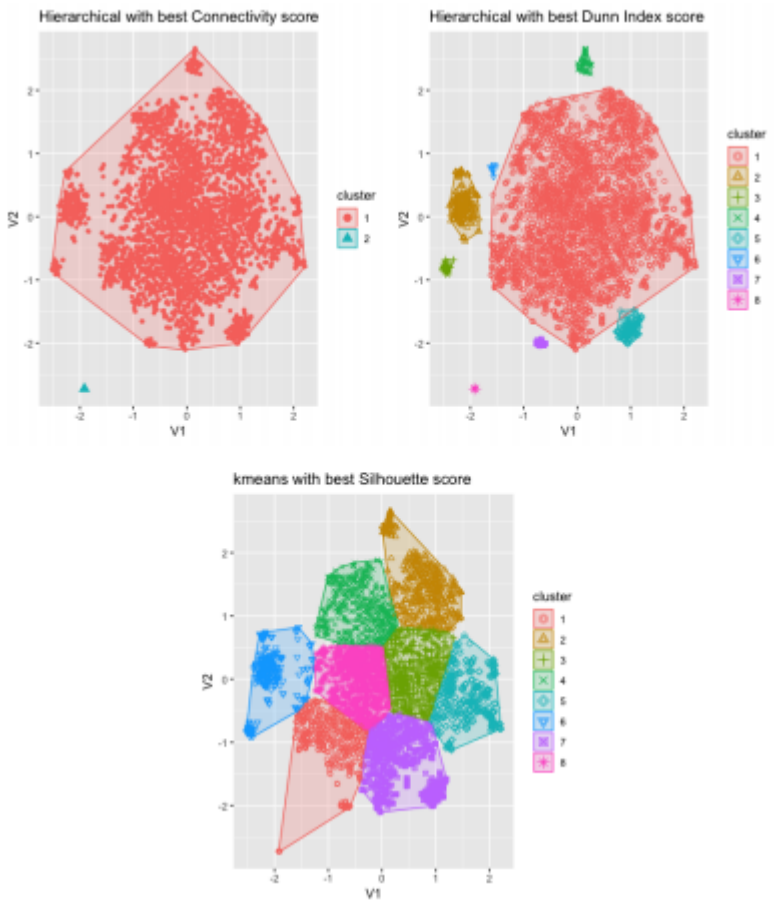


FIGURE 2.2 Combinaison d'algorithmes de *clustering*, source :(Mangortey *et al.*, 2020)

retenu après analyse que l'algorithme de clustering hiérarchique avec huit clusters était le meilleur. D'autres recherches ont tenté de regrouper (*K-means Clustering*) des trajectoires de décollage d'avions similaires et comprendre leurs dépendances avec les attributs de vol et de piste afin de conseiller les pilotes de façon appropriée (Mathews,). Il s'est avéré une dépendance entre la consommation de carburant, la distance parcourue par l'aéronef et le degré total de virages effectués par l'aéronef dans sa trajectoire de décollage.

Par contre, très peu d'études se sont concentrées sur l'extraction du niveau de compétence des pilotes ou encore sur la catégorisation de profil comportementale. Dans cette perspective, (Nittala *et al.*, 2018) a proposé un système pour prédire le niveau de compétence et la charge de travail des lots à l'aide d'algorithmes d'apprentissage automatique tels que la machine à vecteurs de support (SVM), la régression logistique avec la régularisation LASSO qui consiste à ajouter la valeur absolue de l'amplitude du coefficient comme terme de pénalité à la fonction de perte, etc.

2.2.2 Les modèles appris

Les pilotes humains sont formés pour faire face aux incertitudes ou aux urgences pouvant survenir lors d'un vol, telles que des conditions météorologiques extrêmes ou une défaillance du système. En revanche, les systèmes de contrôle automatique de vol (AFCS/autopilote) sont très limités, capables d'effectuer des tâches de pilotage minimales (Baomar et Bentley, 2016). Dans de nombreux domaines tels que les sciences de données et l'apprentissage automatique, il est souvent impossible d'exprimer une relation directement sous la forme d'une fonction déterministe et explicite. Dans cette perspective, la relation est plutôt exprimée par un modèle implicite à variation dynamique qui a appris des données ou des informations disponibles. Au fur et à mesure que les données ou les informations changent, le modèle doit être mis à jour par la formation ou l'apprentissage pour refléter les nouvelles informations. De tels modèles sont appelés modèles appris(Yang, 2016). Le principal défi dans la conception de ces modèles dans le pilotage d'avion est l'aspect dynamique et la complexité des manœuvres de pilotes. Néanmoins, plusieurs recherches ont tenté de modéliser le comportement de pilotage. La thèse (Bodin, 2020) a réussi à identifier quelques manœuvres (taches réalisées) de vol à l'aide des techniques d'apprentissage automatique telles que : la régression lo-

gistique, les machines à vecteurs de support (SVM) et les réseaux de neurones. (Lowe et How, 2015) a tenté de prédire le comportement de pilotes en développant un modèle génératif basé sur les trajectoires réelles qui utilise un mélange gaussien pour prévoir les trajectoires d'avions et déduire les durées de vol. Une autre recherche s'est penchée sur la question réelle du comportement de pilotage (Baomar et Bentley, 2016). Elle tente d'étendre les capacités du pilote automatique moderne afin qu'il soit capable de gérer plusieurs scénarios (incertitudes, urgences) de vols sans intervention humaine en développant un modèle qui observe et imite le pilote humain. Cette approche consiste à collecter les données de pilotes à l'aide d'un simulateur (X-Plane) et d'entraîner des modèles d'apprentissage machine tels que les réseaux de neurones, à partir de ces données collectées. Les réseaux de neurones sont utilisés pour prédire le comportement du pilote dans les mêmes conditions. Dans cet article, les auteurs se sont penchés uniquement sur les conditions environnementales (météorologiques) comme impact de comportement de pilote. Dans le même contexte, (Matsumoto *et al.*, 2015) a proposé une approche d'apprentissage similaire qui repose sur l'apprentissage à partir de la démonstration (LFD) pour capturer les compétences du pilote humain et les appliquer dans un système aérien sans pilote (UAS) autonome.

2.3 Les séries temporelles et les algorithmes d'apprentissage machine

2.3.1 Définition

Une série temporelle ou chronologique (souvent appelée *timeserie*) est une mesure historique d'une variable observable dans une période de temps à intervalles régulier. Elles sont étudiées à plusieurs fins telles que la prévision de l'avenir basé sur le passé (phénomènes météorologiques, prévision des valeurs boursières, etc)(Bontempi *et al.*, 2013). Les séries chronologiques jouent un rôle crucial dans

presque tous les domaines : télécommunications, finance, informatiques, économie, etc(Palit et Popovic, 2006). Une série temporelle observée peut être décomposée en trois composantes : la tendance (direction à long terme), la saisonnalité (mouvement systématique lié au calendrier) et l'irrégularité (fluctuations non systématiques à court terme).

2.3.2 Regroupement des séries temporelles

Le *clustering* est une méthode de partitionnement de données en groupes ayant des caractéristiques similaires. Le regroupement des données temporelles peut être abordé sur 3 phases (Liao, 2005) : représentation des données, mesures de distance et algorithmes (voir figure 2.3) :

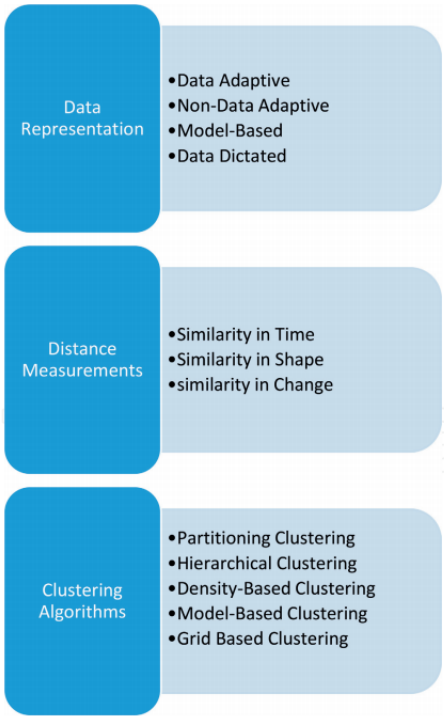


FIGURE 2.3 Regroupement des données temporelles, source :(Liao, 2005)

Représentation des données

Les séries temporelles sont en général très volumineuses à cause de leurs multi-dimensionnalités (Lin *et al.*, 2003) ce qui rend la représentation de ces données plus complexe (temps de traitement croissant...). Le volume de ces données rend ainsi la mesure de la similarité plus difficile. La représentation des données peut être examinée sur quatre axes : données adaptatives, données non adaptatives, données basées sur un modèle et données dictées (Aghabozorgi *et al.*, 2015).

- Méthodes adaptatives de données : techniques utilisées pour traiter et analyser les données de séries chronologiques en utilisant des algorithmes qui peuvent s'adapter aux caractéristiques changeantes de ces données. Il en existe plusieurs : la décomposition en valeurs singulières (SVD) (Baker, 2005), langage naturel symbolique (Portet *et al.*, 2009) etc.
- Méthodes adaptatives non basées sur les données : cette technique utilise les caractéristiques de taille fixe pour les séries temporelles représentatives.
- Méthodes basées sur les modèles : sont utilisées pour décrire et comprendre la structure sous-jacente des séries chronologiques. Ces méthodes impliquent généralement l'utilisation de modèles mathématiques pour décrire les propriétés statistiques des séries chronologiques, comme la tendance, la saisonnalité et la variabilité aléatoire. On peut avoir comme exemple les HMM (*Hidden Markov model-based*) (Kayte *et al.*, 2015).
- Méthodes dictées par les données : il est question ici de la déduction automatique des taux de réduction de dimensions par les utilisateurs.(Liao, 2005)

Mesures de distances

Les mesures ou calculs de distances des séries temporelles sont des éléments importants du regroupement. Une méthode permettant de calculer efficacement les distances entre les séries temporelles est la déformation temporelle dynamique (Yadav et Alam, 2018). DTW (*Dynamic Time Warping*) est une fonction qui prend deux séries temporelles (T, U) en entrée et calcule leur distance d. Elle permet de mesurer la similarité entre 2 séquences temporelles qui ne s'alignent pas exactement dans le temps, la vitesse ou la longueur (voir figure 2.4).

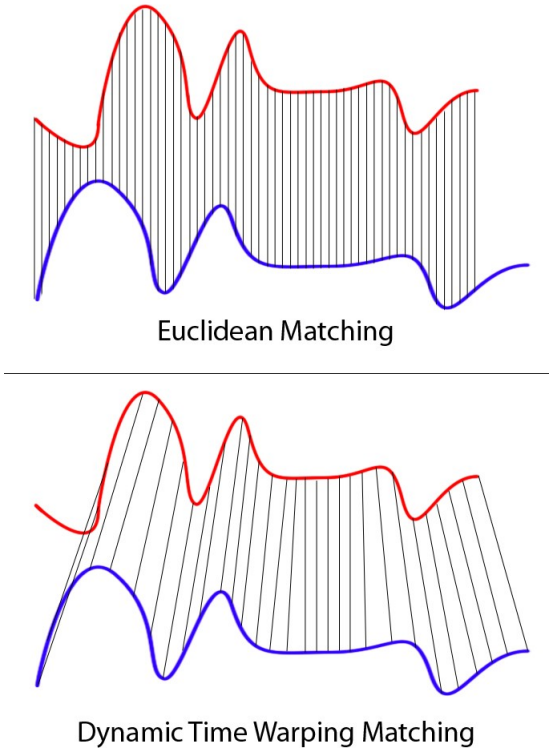


FIGURE 2.4 Mesure de similarité avec DTW, source : (https://paperswithcode.com/method/dtw)

Algorithme de regroupement

Il existe plusieurs classes d'algorithmes de regroupement. Certains algorithmes utilisent directement les séries temporelles pour le regroupement, mais d'autres ont besoin d'un prétraitement des séries temporelles. Ce prétraitement consiste en la réduction de dimensionnalité (Wang et Megalooikonomou, 2008). On peut entre autres citer :

1. *clustering* de partitionnement

   Le *clustering* de partitionnement est une méthode de regroupement de données qui consiste à diviser un ensemble de données en groupes (appelés *clusters*) de manière à minimiser une mesure de dissimilarité entre les éléments d'un même groupe. Elle a pour avantage la simplicité et la rapidité. Un exemple de *clustering* de partitionnement assez populaire est l'algorithme des K-Moyennes(Sinaga et Yang, 2020) .

2. Clustering hiérarchique

   Contrairement au clustering de partitionnement, le regroupement hiérarchique ne nécessite pas de déterminer un nombre de classes au préalable. En effet, en modifiant la profondeur de l'arbre, on peut voir toutes les possibilités et choisir le nombre de classes qui nous convient le mieux. Il a pour avantage une représentation graphique des données, mais a une complexité de calcul élevée.

3. Clustering basé sur la densité

   Cette technique trouve des échantillons de base de haute densité et développe des groupes à partir d'eux. Elle détermine les zones denses de l'espace objet. Elle est intéressante pour les données qui contiennent des clusters de densité similaire. DBCAN (Khan *et al.*, 2014) est un exemple d'algorithme de *clustering* basé sur la densité.

2.3.3 Réseaux de neurones récurrents : cas des LSTM

Un réseau de neurones est un ensemble d'algorithmes inspiré du fonctionnement du cerveau humain. Les deux types de réseaux de neurones les plus utilisés sont : les CNN (réseaux de neurones convulsifs) et les RNN (réseaux de neurones récurrents). Les RNN ont la capacité de traiter les informations temporelles. Ils traitent dans une séquence d'entrée un élément à la fois, en maintenant dans leurs unités cachées un «vecteur d'état» qui contient implicitement des informations sur l'historique de tous les éléments passés de la séquence (LeCun *et al.*, 2015) alors que les CNN sont incapables d'interpréter efficacement les informations temporelles.

> "Actuellement, la plupart des ensembles de données de séries chronologiques du monde réel sont multivariés et riches en informations dynamiques. De tels ensembles de données attirent beaucoup d'attention ; par conséquent, le besoin d'une modélisation précise de ces ensembles de données de grande dimension augmente. Récemment, l'architecture profonde du réseau de neurones (RNN) et sa variante de mémoire à long court terme (LSTM) se sont avérées plus précises que les méthodes statistiques traditionnelles dans la modélisation des données chronologiques"(Lee *et al.*, 2014).

Appréciation

Un réseau de neurones récurrent est un modèle neuronal qui atteint des performances de pointe sur les tâches importantes telles que la modélisation du langage, la reconnaissance vocale, la traduction automatique, etc. (Sherstinsky, 2020). Ils sont un type puissant et robuste de réseaux neuronaux et appartiennent aux algorithmes prometteurs du moment. En raison de leur mémoire interne, les RNN ont la capacité de se souvenir des informations importantes concernant les don-

nées qu'ils ont reçues, d'où la précision dans la prédiction suivante. Ils performent beaucoup mieux sur les données séquentielles que d'autres algorithmes.

Fonctionnement

Les LSTM (*Long Short Term Memory*) sont un cas particulier ou une variante pour les réseaux de neurones récurrents (RNN) (Sagheer et Kotb, 2019) qui étend leur mémoire. Les LSTM permettent aux RNN de se souvenir des entrées sur une longue période de temps. Les LSTM conservent leurs informations dans une mémoire et peuvent effectuer des modifications sur ces informations là. La mémoire du LSTM est vue comme une cellule *gated*, où *gated* signifie que la cellule décide dépendamment de l'importance qu'elle attribue à l'information, de la sauvegarder ou de la supprimer. L'attribution de l'importance se fait à travers des poids, qui sont également appris par l'algorithme. La Figure 2.5 illustre les trois portes d'un LSTM : entrée, oubliée et sortie.

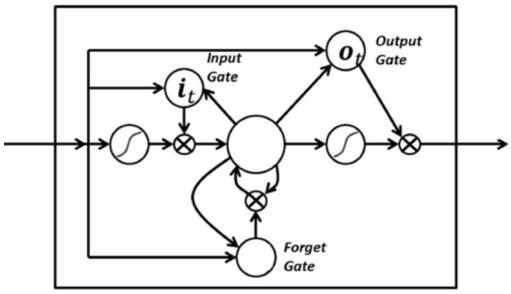


FIGURE 2.5 Fonctionnement d'un LSTM, source : (https://france.devoteam.com/paroles-dexperts/aller-plus-loin-en-deep-learning-avec-les-reseaux-de-neurones-recurrents-rnns/).

LSTM Autoencodeurs

Les auto-encodeurs sont des algorithmes d'apprentissage automatique non supervisé à base de réseau de neurones, qui permettent de construire une nouvelle représentation d'un jeu de données. Généralement, celle-ci est plus compacte, ce qui permet de réduire la dimensionnalité du jeu de données. L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur (voir figure 2.6). L'encodeur est constitué d'un ensemble de couches de neurones, qui met sur pieds une nouvelle représentation d'informations dites "encodées". Les couches de neurones du décodeur reçoivent à leur tour ces représentations et tentent de reconstruire l'information initiale. Les différences entre les données initiales et les données décodées permettent de mesurer l'erreur commise par l'auto-encodeur. L'entraînement consiste à jouer avec les paramètres de l'auto-encodeur afin de réduire l'erreur de décodage mesurée sur les différents exemples du jeu de données.

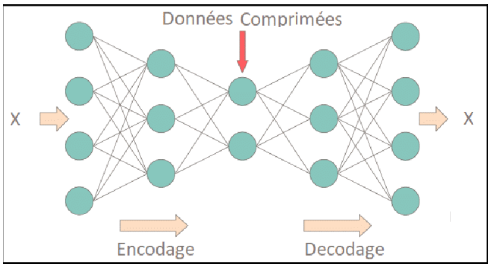


FIGURE 2.6 Autoencodeur, source : (https://jafwin.com/2020/07/04/les-auto-encodeurs-expliques-en-detail/)

Un auto-encodeur LSTM est une implémentation d'un auto-encodeur pour les données séquentielles. Dans cette architecture (figure2.7), un modèle de codeur LSTM lit pas à pas la séquence d'entrée. Après avoir lu toute la séquence d'entrée, l'état caché ou la sortie de ce modèle devient une représentation apprise interne de l'ensemble de la séquence d'entrée sous la forme d'un vecteur de longueur fixe. Ce vecteur est ensuite fourni comme entrée au modèle au décodeur qui l'interprète au

fur et à mesure que chaque étape de la séquence de sortie est générée. (Srivastava *et al.*, 2015)

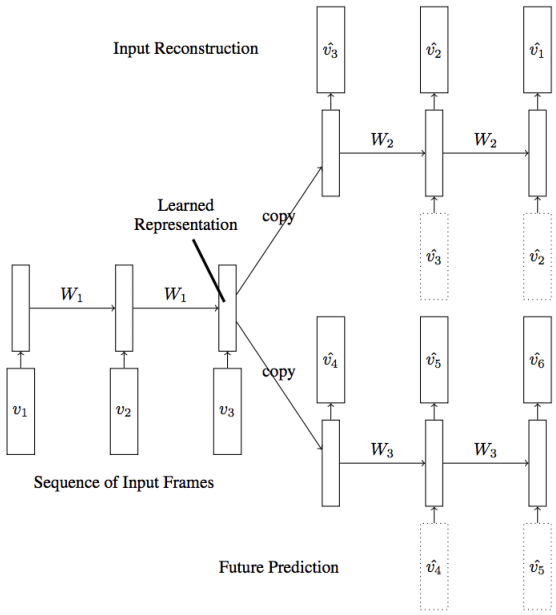


FIGURE 2.7 Architecture d'un Autoencodeur LSTM, souce : (https://machinelearningmastery.com/lstm-autoencoders/)

2.3.4 Les transformeurs

Définition

Un transformeur est un type d'architecture de réseau de neurones qui utilise des encodeurs et des décodeurs empilés pour traiter une séquence de mots en parallèle plutôt qu'en mémorisant des séquences entières, comme c'est le cas avec des réseaux de neurones récurrents (Vaswani *et al.*, 2017). Les architectures de transformeur ont révolutionné différents domaines de l'apprentissage en profondeur, du traitement du langage naturel. Depuis l'avènement des premiers modèles de transformateurs basés sur l'attention, il y a eu des tentatives intéressantes pour

les adapter à l'univers des séries chronologiques. Les transformeurs permettent au modèle l'accès à n'importe quelle partie d'une série, quelle que soit la distance, ce qui la rend potentiellement plus adaptée pour comprendre les schémas récurrents avec dépendances à long terme.

GPT

Le modèle GPT *Generative Pre-Trained Transformer* a été développé par OpenAI et publié en 2018. Depuis sa création, il est devenu de plus en plus populaire. Il est un modèle d'apprentissage automatique pré-entraîné sur une grande quantité de textes grâce à un apprentissage non supervisé pour générer des réponses de langage écrit de type humain (Chan, 2022). Il est basé sur l'architecture du transformeur, il apprend les relations contextuelles entre les mots dans un texte. C'est très pratique, car cela permet d'accélérer massivement la formation des modèles grâce au traitement parallèle.

En bref, les transformeurs sont des modèles d'apprentissage conçus pour gérer des données séquentielles en utilisant des mécanismes d'attention. Ils sont très puissants et très utilisés pour les tâches complexes de traitement de langage naturel (NLP), bien qu'ils nécessitent en général beaucoup de données et de temps de calcul.

2.4 Conclusion

Le présent chapitre présente la revue de la littérature portant premièrement sur la notion de profil comportemental de pilotage. Nous avons pu constater que, un pilote peut changer de profil (gain élevé /gain faible) dépendamment des conditions dans lesquelles il se trouve tel que la dynamique de l'avion, la charge de travail, etc. La mesure de ce profil ou gain de pilote se mesure selon 3 classes

potentielles basées sur : le domaine temporel, le domaine fréquentiel et le domaine pilote. Ensuite, nous avons recensé les travaux en apprentissage automatique dans l'aéronautique, particulièrement les regroupements et prédictions. Plusieurs travaux ont été mis en exergue pour la détection des anomalies de vol, mais très peu pour profil des pilotes. D'un autre côté, plusieurs recherches ont tenté de modéliser le comportement de pilotages. De ce qui est vu dans la littérature, aucune étude ne se base sur les segments de vols (segments de décollage en particulier) ni pour apprendre du comportement de pilotage ni pour prédire ceux-ci pourtant, un pilote peut exhiber différents types de comportements durant un vol. Définir les segments de décollage ouvre donc la porte à la possibilité d'aller identifier ces différents comportements. Ce qui n'est pas toujours le cas si on considère le vol ou un décollage en entier sans les segments lors de l'extraction ou la prédiction de comportements de pilotage. Enfin, nous avons parcouru les algorithmes d'apprentissage automatique pour les données de vols (séries temporelles) tels que le regroupement des séries temporelles, les transformeurs et les LSTM. Malgré que les transformeurs performent en général bien, nous nous penchons sur les LSTM car, nous n'avons pas assez de données ni d'architecture matérielle pour ceux-ci.

CHAPITRE III

UNE APPROCHE POUR LE PROFILAGE COMPORTEMENTAL DE PILOTAGE

Dans ce chapitre, nous d'identifions à travers des algorithmes d'apprentissage non supervisés des catégories de profils de pilotes émergents des données, entre autres les profils à gain élevé et à gain faible. Tout d'abord nous opterons pour les LSTM auto-encodeurs et l'algorithme des K-moyennes. Dans un second temps, nous explorerons les solutions de regroupement de séries temporelles.

3.1 Les données

Ce travail se concentre sur la tâche de décollage d'un Airbus 320. Les données (télémétrie) utilisées ici sont recueillies à partir de simulateurs de vol complets de niveau D de CAE et comprennent 90 décollages effectués par 90 pilotes. Nous avons recensé parmi ces décollages 67 qui se sont effectués dans des conditions normales (sans pannes). Ces données comprennent l'orientation de l'avion, la vitesse, les commandes du pilote, des données contextuelles, etc. Les données de décollages, qui sont séquentielles et multidimensionnelles, sont associées à un indice temporel qui est le dixième de seconde, représentant la variation des paramètres dans le temps. Plusieurs prétraitements ont été appliqués aux données avant leurs utilisations dans les différents algorithmes.

3.1.1 Sélection des variables

Il est crucial de comprendre l'importance de la sélection des caractéristiques lors de la création d'un modèle d'apprentissage automatique. L'ajout de variables redondantes réduit la capacité de généralisation du modèle et peut également réduire la précision globale d'un modèle. De plus, ajouter plus de variables à un modèle augmente la complexité globale du modèle. Les données reçues pour chaque pilote comprennent 195 variables.

- La première étape consiste à la suppression de tous les paramètres à faible variance ou variance nulle afin de rendre fiable le modèle final. Ceci a été fait à l'aide du «*VarianceTreshold*» de la librairie *scikit-learn* de Python. La sélection a permis de garder 125 paramètres sur les 195 de départ.
- La seconde étape s'est faite avec les experts du domaine afin de sélectionner les variables les plus pertinentes pour un décollage (entrées du pilote et sortie de l'avion). Sur les 125 variables 2 ont été sélectionnés.

3.1.2 Normalisation

La remise à l'échelle des données à partir de la plage d'origine permet que toutes les valeurs soient comprises dans le même intervalle. Nous utilisons ici deux techniques dépendamment des cas de figure.

- *Min-Max Scaler* : c'est la technique que nous appliquons dans le cas des données à 2 dimensions. À l'issue de cette transformation, les valeurs des paramètres seront comprises dans un intervalle fixe $[0, 1]$.
- *TimeSeriesScalerMinMax* : c'est la technique utilisée pour les données à 3 dimensions. Elle met à l'échelle les séries chronologiques de sorte que leur étendue dans chaque dimension soit comprise entre un minimum et un maximum.

Le but d'avoir un tel intervalle restreint est de réduire l'espace de variation des valeurs d'un paramètre et par conséquent réduire l'effet des valeurs aberrantes (c'est une valeur ou une observation qui est distante des autres).

3.2 Expérience 1 : Regroupement des données globales de décollages

Nous allons dans cette expérience identifier les catégories de profils de pilotage sur les données globales de pilotage d'une part à l'aide de l'algorithme des K-moyennes qui est une méthode nécessitant des données de dimension deux ou une structure de données tabulaires composées de colonnes et de lignes (similaire à une feuille de calcul d'un tableur Excel). Nous commençons par réduire la dimension de nos données (LSTM Autoencodeurs) et par la suite nous appliquons le regroupement sur ces données.D'une autre part, nous effectuons le regroupement de séries temporelles.

3.2.1 Approche de réduction de dimensions

Comme dit précédemment, les données à notre disposition sont des données chronologiques à trois dimensions : la première dimension représente les pilotes ; la seconde dimension représente la chronologie pour chaque décollage et la dernière dimension représente l'ensemble de variables pour chaque décollage. La première étape de cette approche consiste à passer les données de la dimension 3 à la dimension 2. On pourra alors appliquer simplement l'algorithme des K-moyennes pour le regroupement.

Comme le montre la figure 3.1, les pilotes effectuent leurs décollages avec des durées différentes. Nous allons premièrement échantillonner les données afin d'avoir un temps égal de décollage pour notre LSTM.

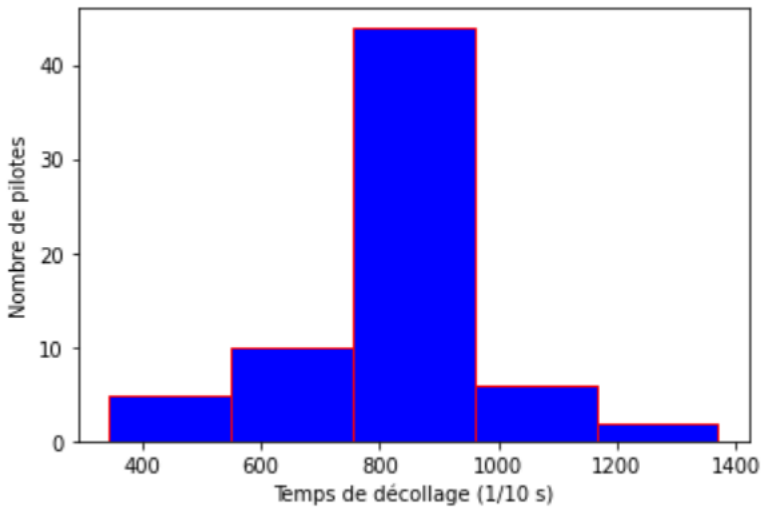


FIGURE 3.1 Temps des décollage de pilotes.

Nous avons 4 options pour le choix du temps de décollage : le temps de décollage minimum qui est de 343, le temps maximum qui est de 1371, le temps médian qui est de 854 et le mode qui est de 895. Pour chaque cas de figure, nous avons calculé la moyenne des écarts entre les différents temps de décollage et les différents choix (médiane, mode, minimum , maximum) que nous appelons ici écart moyen. L'écart moyen par rapport au temps minimum est 8,82. l'écart moyen par rapport au temps maximum est 6,52. L'écart moyen par rapport au temps médian est 0,58. L'écart moyen par rapport au mode est 1,19. Le temps médian est optimal.

Échantillonnage : sous-échantillonnage, sur-échantillonnage

Cette opération implique un changement de la fréquence temporelle des données. Le module utilisé est la fonction *scipy.signal*, qui est lié au traitement des formes d'ondes dans la bibliothèque *Scipy*. Puisque *scipy.signal.resample*[1] utilise la trans-

1. https ://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.resample.html

formée de Fourier, il est basé sur l'hypothèse que le signal est périodique. Ici, le ré-échantillonnage se fait pour obtenir le temps de collecte de données égal à 854 dixièmes de secondes (temps médian). Le graphique présenté à la figure 3.2 montre l'exemple d'un cas de sous-échantillonnage de la valeur du paramètre vent arrière (*tailwind*) de l'avion pour un pilote particulier.

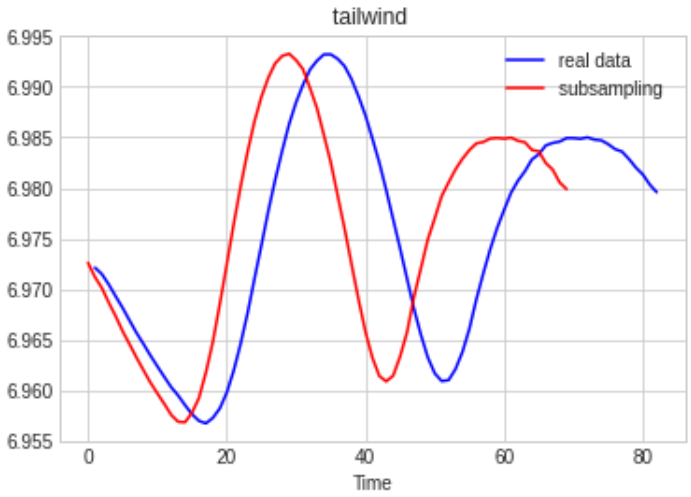


FIGURE 3.2 Exemple de sous-échantillonnage de la valeur du paramètre vent arrière.

3.2.2 Autoencodeur LSTM

Nous allons maintenant créer notre autoencodeur LSTM (voir annexe A), l'ajuster et le tester. La figure 3.3 présente l'architecture de celui-ci. L'entrée de ce modèle se compose d'un vecteur de taille $(854, 24)$. La première valeur est le temps de décollage, et la deuxième valeur correspond aux différents paramètres. La sortie du modèle est pareil que l'entrée.

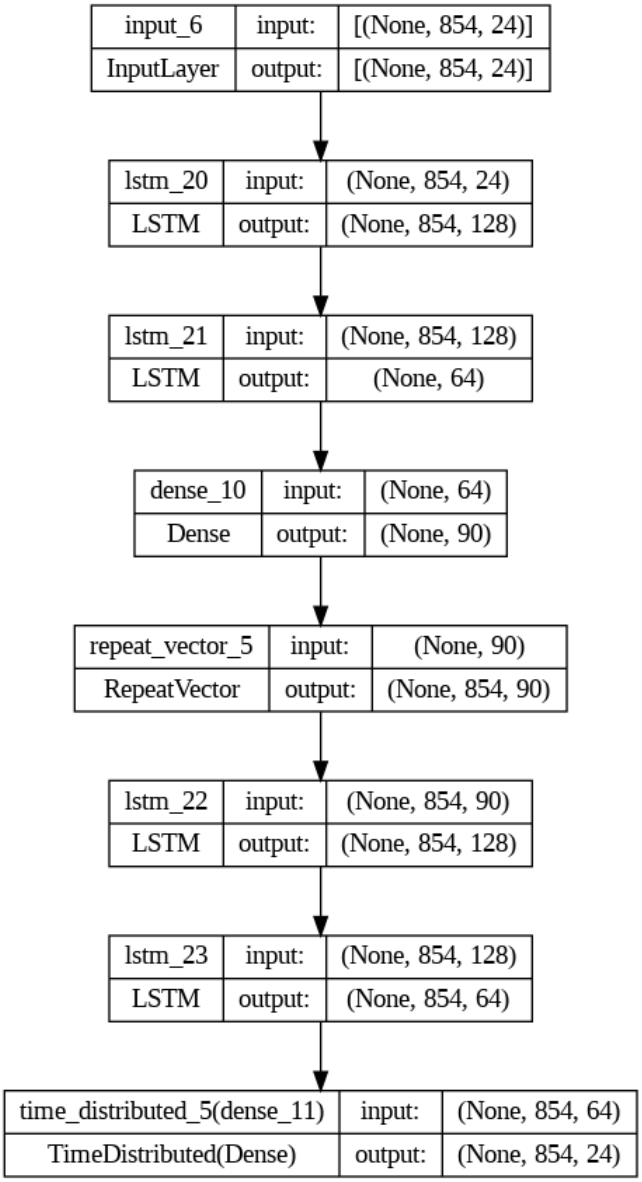


FIGURE 3.3 Architecture de l'autoencodeur LSTM poposée.

La fonction de perte utilisée est le MSE (erreur quadratique moyenne) qui permet de mesurer les erreurs dans notre modèle. On peut remarquer qu'à chaque époque la performance du modèle est améliorée (figure 3.4). Le modèle étant bien ajusté, nous utilisons le codeur en tant que modèle autonome qui a en sortie un vecteur

de taille 90 (vecteur réduit de nos données d'origine). C'est sur ces vecteurs qu'on va appliquer notre algorithme de regroupement.

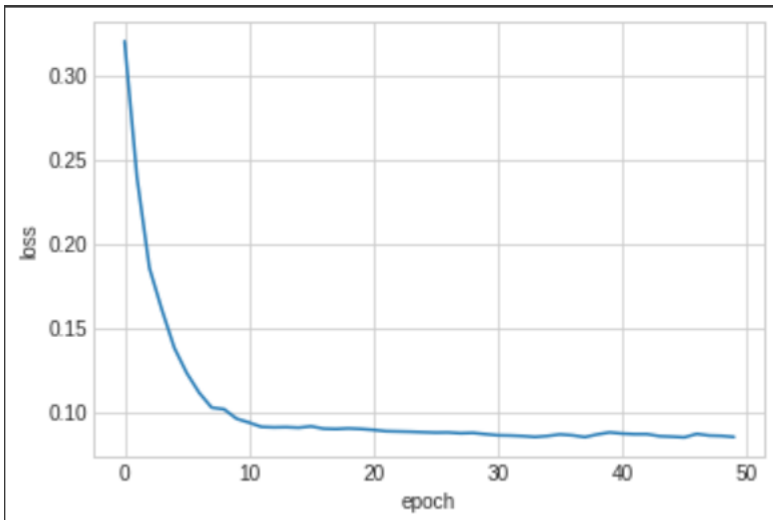


FIGURE 3.4 Courbe de perte.

Regroupement

En général, il n'y a pas vraiment de méthode pour déterminer la valeur exacte du nombre k de *clusters* optimal. Toutefois, la connaissance du domaine et l'intuition peuvent aider. Dans la méthodologie de regroupement, nous pouvons évaluer les performances des modèles en fonction de différents *clusters* k, car les *clusters* sont utilisés dans la modélisation en aval. Les métriques telles que la méthode du coude (*Elbow*) et l'analyse de silhouette sont souvent utilisées comme techniques d'évaluation complémentaires plutôt que préférer l'une à l'autre (Jain, 2010).

La méthode du coude : *Elbow*

L'idée de cette méthode est d'exécuter un regroupement en k-moyennes pour une plage de groupes k (disons de 1 à 10 par exemple) pour chaque valeur, calculer la

somme des distances au carré de chaque point à son centre assigné (distorsions). Lorsque les distorsions sont tracées et que le tracé ressemble à un bras, le « coude » ou le point d'inflexion sur la courbe, est la meilleure valeur de k. La figure 3.5 présente le tracé des distorsions dans notre cas. Le nombre de groupes est 2.

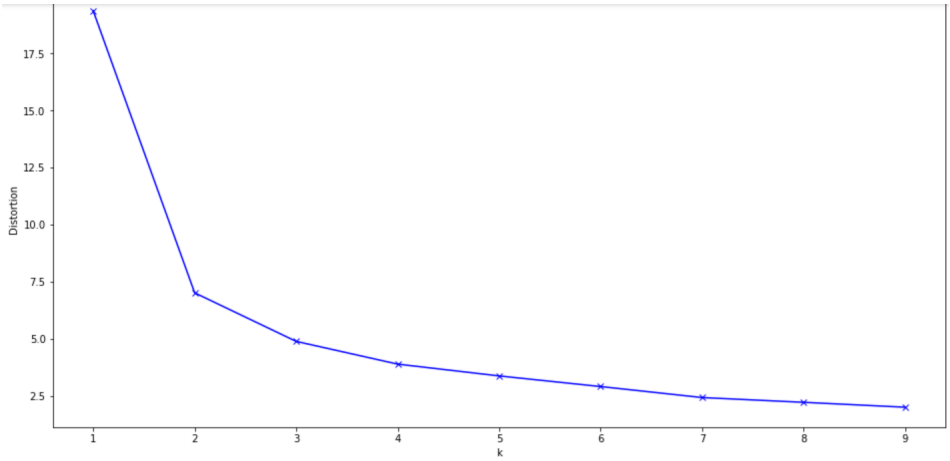


FIGURE 3.5 Tracé des distorsions

La méthode du coefficient de silhouette

Nous utilisons ici le coefficient de silhouette (*silhouette score*) qui est un indicateur de qualité d'une partition de *clustering*. Il mesure la similarité d'un point à ses points de groupes voisins par rapport aux autres groupes. Plus précisément, pour chaque point dans un groupe, le coefficient de silhouette calcule la différence entre la distance moyenne du point à tous les autres points dans le même groupe et la distance moyenne du point à tous les points dans le groupe le plus proche. Ce coefficient varie entre -1 et 1, avec des valeurs proches de 1 indiquant que les points sont fortement groupés avec d'autres points similaires dans le même groupe et éloignés des points dans les autres groupes, tandis que des valeurs proches de -1 indiquent une forte confusion entre les groupes et des valeurs proches de 0 indiquent une indétermination quant à l'affectation du point à un groupe en

particulier. Le meilleur score est obtenu lorsque k est 2 (tableau 3.1).

TABLEAU 3.1 Scores silhouette

Nombre K de clusters	2	3	4	5	6	7	8	9
Score silhouette	0.63	0.48	0.3	0.28	0.05	-0.11	0.01	0.019

Résultats et discussions

Après avoir obtenu les différents groupes, nous décodons ces valeurs afin d'avoir nos données chronologiques initiales. Les figures 3.6 et 3.7 présentent les manœuvres effectuées par les pilotes au fil du temps sur le manche de tangage (*pitch stick*), le manche de roulis(*roll stick*) et les pédales de direction (*rudder pedals*) des 2 groupes obtenus :

— Le groupe 1 comprend 25 pilotes, le tracé en bleu présente le centroïde du groupe et en noir les différents pilotes du groupe.

— Le groupe 2 comprend 32 pilotes, le tracé en vert présente le centroïde du groupe et en noir les différents pilotes du groupe.

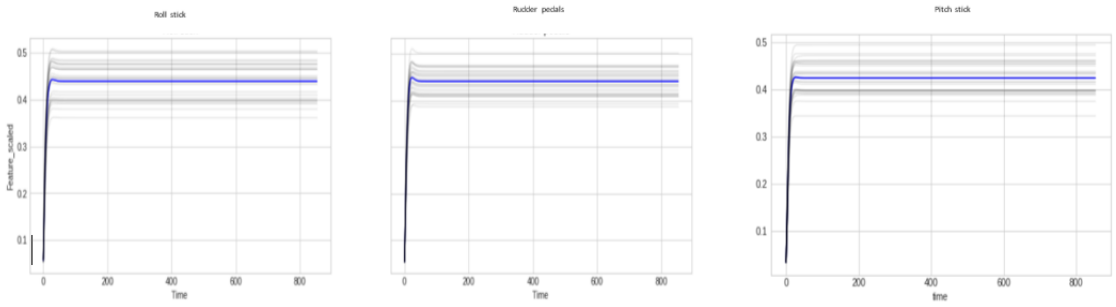


FIGURE 3.6 Groupe 1 obtenue de la méthode de réduction de dimensions (LSTM) suivie du K-moyennes

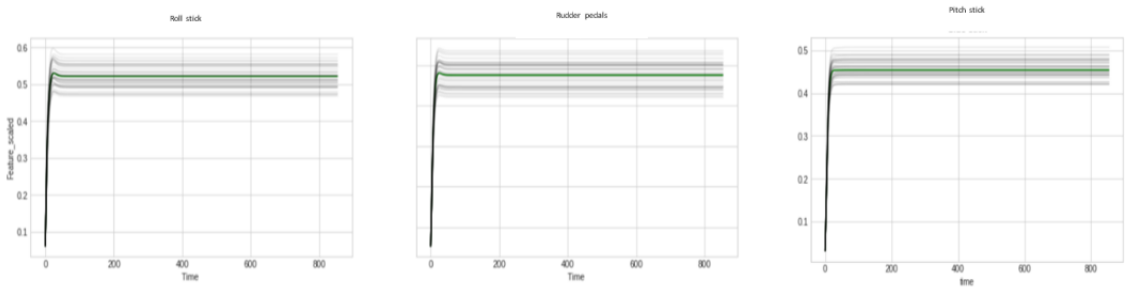


FIGURE 3.7 Groupe 2 obtenue de la méthode de réduction de dimensions (LSTM) suivie du K-moyennes

Lorsqu'on superpose les centroïdes des différentes classes de pilotes (figure 3.8), on n'observe presque le même mouvement dans les différents groupes sur les trois manœuvres des pilotes *roll stick* (manche de roulis), *rudder pedals* (pédales de direction) et *pitch stick* (manche de tangage) à l'exception d'une pression plus accentuée dans le groupe 1 que le groupe 2.

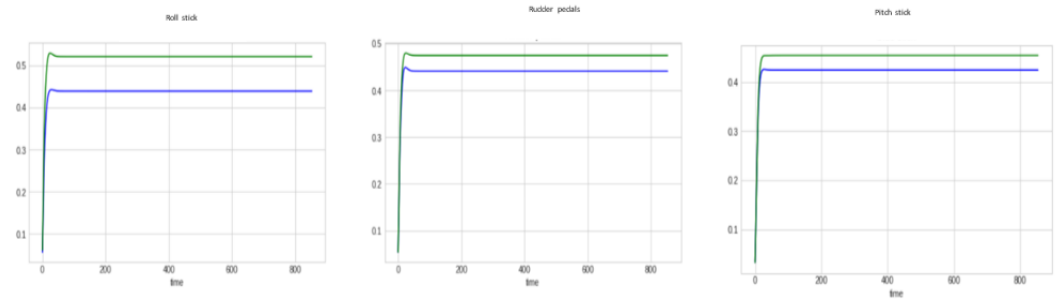


FIGURE 3.8 Superposition des 2 groupes obtenus de la méthode de réduction de dimensions (LSTM) suivie du K-moyennes

3.2.3 K-moyennes pour les séries chronologiques

Approche

La similarité entre les points de données est mesurée avec une métrique de distance, généralement la distance euclidienne (Amidon, 2020). Le K-moyennes combiné avec la distance de séquence dynamique (*Dynamic Time Warping, DTW*) est une approche efficace pour le regroupement de séries chronologiques. Nous remplacerons ici la mesure de distance euclidienne standard du K-moyennes avec la distance de DTW. Cela permettra de mesurer la distance entre les séries chronologiques de manière plus appropriée et de prendre en compte les décalages dans le temps. On part sur le même jeu de données qu'avec l'auto encodeur (données échantillonnées).

Le nombre k de groupes

L'analyse de silhouette est utilisée pour déterminer le degré de séparation entre les groupes. Par conséquent, nous voulons que les coefficients soient aussi grands que possible pour avoir un bon groupe. Le tableau 3.2 présente le score de silhouette avec différentes valeurs de k testé. Le nombre de groupes optimal est 2.

Tableau 3.2 Scores silhouette pour les séries chronologiques.

Nombre K de groupes	2	3	4	5	6	7	8	9
Score silhouette	0.65	0.23	0.23	0.23	0.01	-0.111	0.061	0.013

Résultats et discussion

Les figures 3.9 et 3.10 présentent les 2 groupes de pilotes ressortis.

— Le groupe 1 comprend 58 pilotes, le tracé en bleu présente le centroïde du

groupe et en noir les différents pilotes du groupe.

— Le groupe 2 comprend 9 pilotes, le tracé en vert présente le centroïde du groupe et en noir les différents pilotes du groupe.

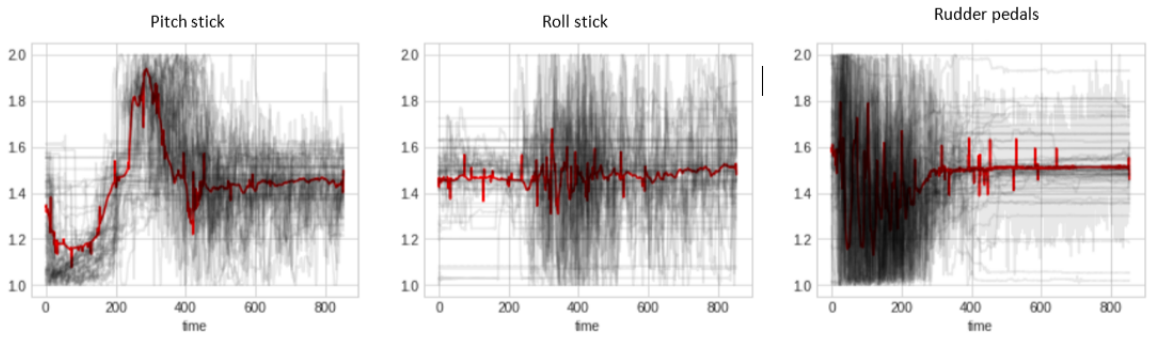


FIGURE 3.9 Groupe 1 obtenue par la méthode des K-moyennes pour les séries chronologiques

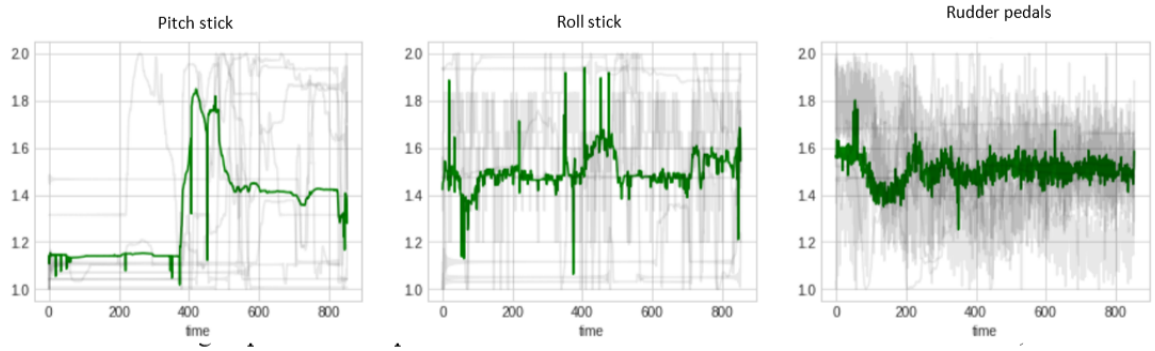


FIGURE 3.10 Groupe 2 obtenue par la méthode des K-moyennes pour les séries chronologiques

La superposition des 2 groupes (figure 3.11) nous permet de constater que :

— La manipulation du manche de tangage (*pitch stick*) a une tendance plus agressive dans le groupe 1 que dans le groupe 2.

— La manipulation du manche de roulis quant à elle (*roll stick*) a des valeurs généralement stables dans les 2 groupes. Cependant, on observe des pics très prononcés dans le groupe 2.

— Une manœuvre excessive (dent de scie) est observée chez les pilotes du groupe 1 comparé à ceux du groupe 2.

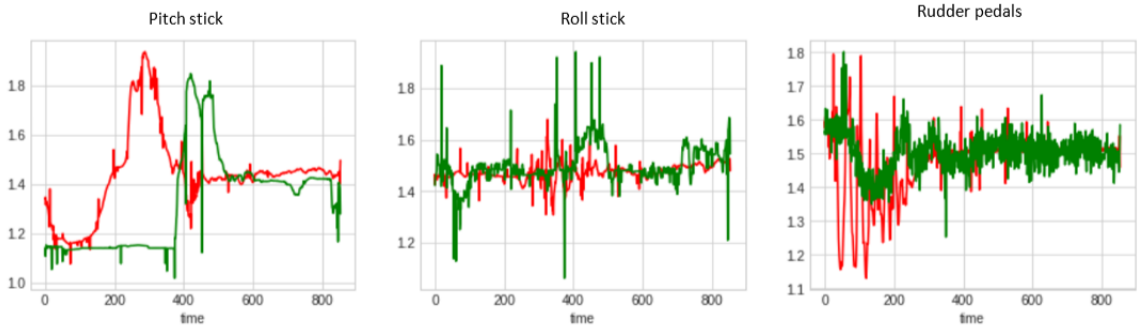


FIGURE 3.11 Superposition des 2 clusters .

3.2.4 Analyse des résultats

Plusieurs constats sont effectués :

— Si on observe les groupes ressortis en utilisant la méthode de réduction de dimensions (LSTM) suivie des K-moyennes et les K-moyennes pour les séries chronologiques, on constate que les manœuvres des pilotes sont très lisses dans la méthode qui consistait à encoder les données. Ce qui montre clairement une perte d'informations. Difficile alors d'avoir des groupes fiables.

— Les groupes extraits à partir de la deuxième méthode semblent être convaincants, mais jusqu'ici on se retrouve avec des pilotes d'un même groupe agissant agressivement sur une pédale à un moment donné et l'effet inverse ensuite.

3.3 Expérience 2 : Regroupement de pilotes sur les séquences de décollages

Un effort concerté avec des experts du domaine conduit à la conclusion qu'en réalité, un pilote peut avoir différents profils comportementaux dépendamment des phases de décollage. Cette deuxième expérience vise deux objectifs tout d'abord, procéder à un regroupement segment par segment compte tenu du constat des experts et dans un second temps d'hypothéser ces constats pour en faire la vérification selon les résultats.

3.3.1 Séquençage de données

Avec l'aide des experts, nous avons recensé 10 segments dans un décollage :

Segment 1 : Avant le décollage

Chaque décollage commence avec l'avion positionné sur le seuil de piste avec un cap correspondant au cap de la piste. Les manettes de poussée sont au ralenti et les freins sont appliqués. Les volets sont réglés sur 1 ou 2. Les spoilers au sol sont armés. Le levier du train d'atterrissage est abaissé. La pression n'est pas appliquée sur le manche latéral ou les pédales de direction.

Segment 2 : Vérification de la symétrie de poussée

Le pilote déplacera les manettes de poussée des moteurs pour obtenir des indications de moteur $N1$ de 50 pour cent. Ce segment est nécessaire pour vérifier si la poussée est symétrique avant d'appliquer la poussée de décollage. Les freins restent appliqués.

Segment 3 : Poussée au décollage

Dès que la symétrie de poussée est vérifiée, le pilote relâchera les freins et réglera les manettes de poussée sur FLEX (35 degrés) ou TOGA (45 degrés).

Segment 4 : Décollage

Une fois que l'avion commence à avancer, le pilote utilisera les pédales de direction pour maintenir le cap de l'avion sur l'axe de la piste. Cela continuera jusqu'au décollage.

Segment 5 : Procédure du manche de tangage pour contrer l'effet du réglage de la poussée au décollage (procédure spécifique à Airbus).

Le segment 4 continue d'être appliqué. L'Airbus dispose d'une procédure pour contrer l'effet du réglage de la poussée moteur au décollage. Cette procédure est appliquée lors de la course au décollage. Elle consiste à régler le *pitch stick* à mi-course si le vent de travers est inférieur à 20 kts (noeuds), sans vent arrière, ou à fond si le vent de travers est supérieur à 20 kts ou si un vent arrière est présent. Une fois que l'avion atteint une vitesse indiquée de 80 kts, le pilote relâche progressivement la pression sur le manche latéral pour atteindre le neutre de 100 kts.

Segment 6 : Rotation

Le segment 4 continue d'être appliqué. Lorsque la vitesse indiquée atteint VR, le pilote tirera sur le manche latéral de manière à atteindre un taux de tangage de 3 degrés/s vers un angle de tangage de 15 degrés. Le segment suivant commence dès que l'avion quitte le sol.

Segment 7 : Décollage pour la phase de montée

Une fois en vol, le pilote relâchera progressivement la pression sur les palonniers (à partir du segment 4). Le niveau des ailes sera maintenu via les commandes du manche de roulis (*roll stick*). Lorsqu'il atteint 15 degrés, le pilote s'inclinera pour correspondre à l'inclinaison commandée par le directeur de vol. La commande de pas du directeur de vol changera pour aider le pilote à maintenir une vitesse indiquée de $V2 + 10$. Le pilote effectuera également un roulis pour correspondre au roulis commandé par le directeur de vol.

Segment 8 : Taux positif

Dès que l'avion quitte le sol et qu'un taux positif est observé sur l'affichage primaire de vol (vitesse verticale), le levier de train d'atterrissage est levé. La procédure du segment 7 s'applique toujours.

Segment 9 : Altitude d'accélération

Une fois que l'avion a dépassé l'altitude d'accélération, le pilote règle les TLA sur l'écran CLB (25 degrés). La procédure du segment 7 s'applique toujours.

Segment 10 : Rétraction du volet

Le pilote lèvera les volets selon les indications sur l'écran de vol. Aux vitesses F, le levier des volets est réglé sur 1. À la vitesse S, le levier des volets est réglé sur 0. Les spoilers au sol sont désarmés après avoir réglé le levier des volets sur 0. La procédure du segment 7 s'applique toujours.

3.3.2 Regroupement des séries chronologiques

Un échange avec les experts a mené à la conclusion que parmi les différents segments de vols cités plus haut, juste 3 sont les plus pertinents en matière de manœuvre, de comportement de pilotage (nécessitant réellement un apprentissage). Nous tenterons de ressortir les différents profils comportementaux de pilotage sur ces 3 segments-là.

Segment 5

L'idée ici est d'observer l'action du pilote sur le manche de tangage (*pitch stick*) dépendamment du vent qu'il soit arrière ou de travers. Pour ce segment, nous n'avons eu que 7 pilotes qui décollent dans des conditions sensibles de vents. La figure 3.12 présente les différents groupes obtenus.

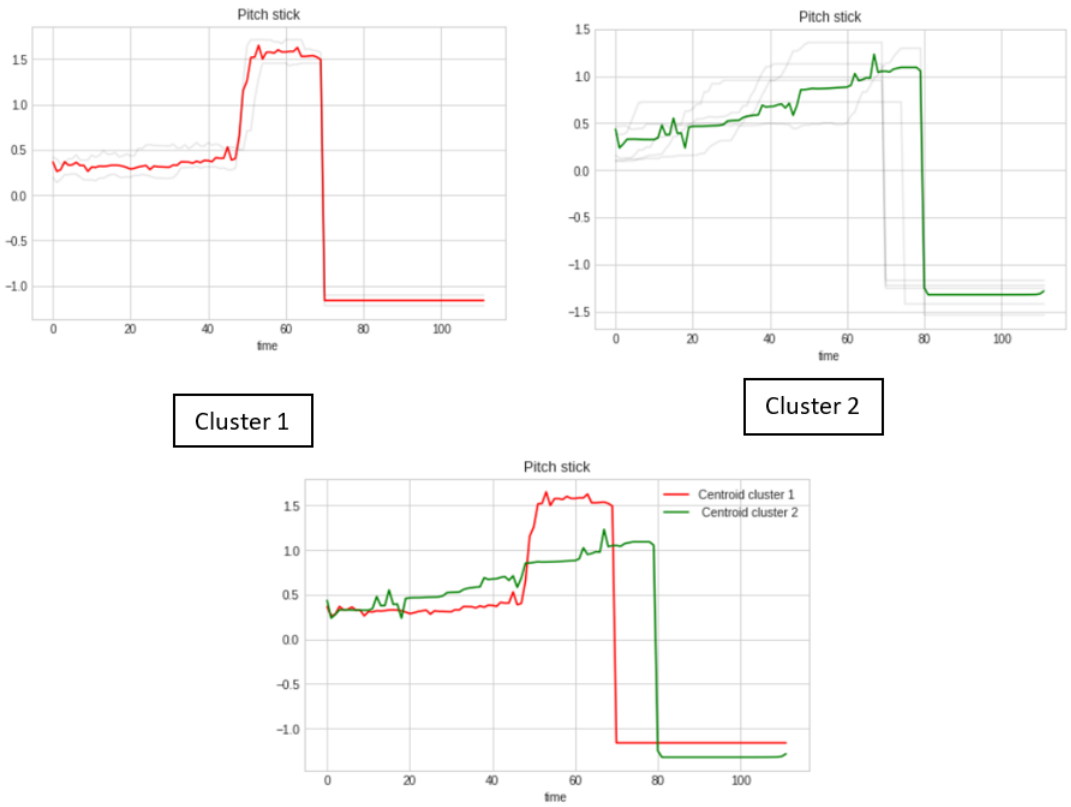


FIGURE 3.12 Les groupes extraits du segment 5

— Le groupe 1 comprend 2 pilotes, le tracé en rouge présente le centroïde du groupe et en noir les différents pilotes du groupe.
— Le groupe 2 comprend 5 pilotes, le tracé en vert présente le centroïde du groupe et en noir les différents pilotes du groupe.

Lorsqu'on superpose les différents centroïdes des différentes classes de pilote, on observe presque le même mouvement dans les différents groupes à l'exception d'un décalage dans le temps. On pourra penser à des pilotes qui agissent plus lentement que d'autres. Également, les pilotes du groupe 2 ont tendance à manipuler le manche latéral moins agressivement que ceux du groupe 1.

Segment 6

Ce segment est juste avant le décollage. L'idée est de voir comment le pilote agit sur le *Pitch stick* pour atteindre un certain taux de tangage vers un angle de tangage. L'étude est faite sur les données de 53 pilotes. La figure 3.13 présente les 2 groupes de comportements observés.

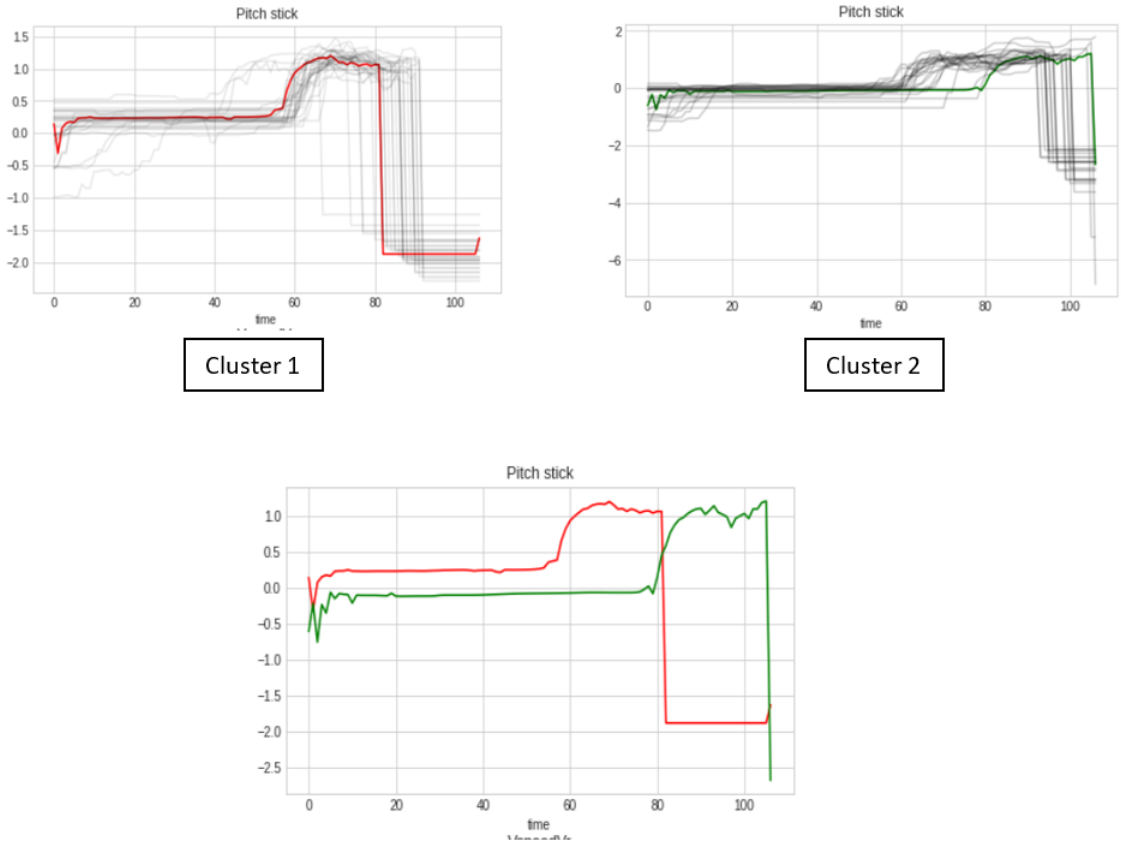


FIGURE 3.13 Les groupes extraits du segment 6

Groupe 1 : il comprend 31 pilotes, le tracé en rouge présente le centroïde du groupe et en noir les différents pilotes du groupe. Groupe 2 : il comprend 22 pilotes, le tracé en rouge présente le centroïde du groupe et en noir les différents pilotes du

groupe. Pareil que pour le groupe précédent, on a presque le même mouvement dans les différents groupes à l'exception d'un décalage dans le temps lorsqu'on superpose les 2 centroïdes.

Segment 7

Cette phase c'est le décollage proprement dit. On s'intéresse à l'action du pilote sur les pédales (*pitch stick, roll stick, rudder pedals*) lorsque l'altitude croit dépendamment de l'inclinaison commandée par le directeur de vol. L'étude est faite sur 67 pilotes. Les figures 3.14 et 3.15 présentent les 2 groupes de comportements observés. Groupe 1 : il comprend 41 pilotes, le tracé en rouge présente le centroïde du groupe et en noir les différents pilotes du groupe. Groupe 2 : il comprend 26 pilotes, le tracé en rouge présente le centroïde du groupe et en noir les différents pilotes du groupe. Les pilotes en rouge (groupe 1) ont en général une manœuvre plus instable et plus lente que ceux du groupe 2 (figure 3.16).

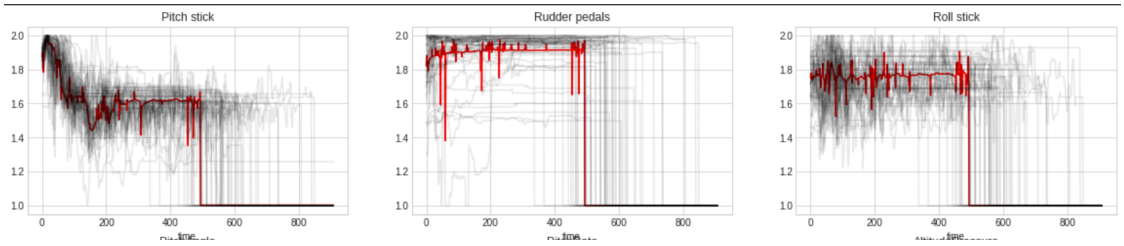


FIGURE 3.14 Segment 7 : groupe 1.

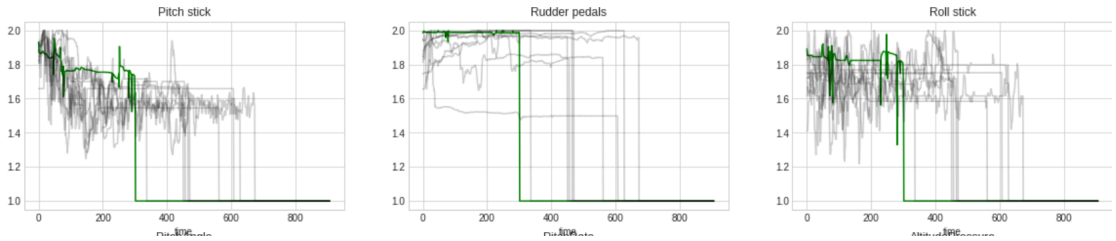


FIGURE 3.15 Segment 7 : groupe 2

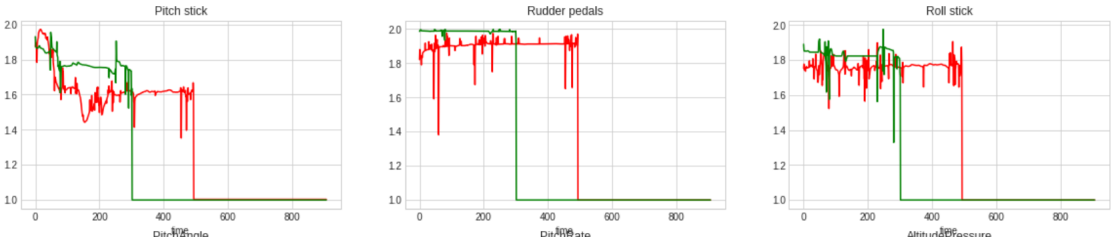


FIGURE 3.16 Superposition des centroïdes des groupes 1 et 2 du segment 7

Cette deuxième expérience a été réalisée sur un jeu de données de 67 décollages d'A320 se déroulant dans des conditions normales. À la différence de la première expérience, l'extraction des profils comportementaux de pilotage est basée sur les segments de décollage, ce qui permet d'avoir des résultats plus spécifiques.

3.4 Conclusion

Nous avons dans ce chapitre identifié à travers l'apprentissage automatique des catégories de pilotes induites des données. Pour ce faire nous avons adopté deux approches : la réduction de dimensions de données en utilisant les Autoencodeurs LSTM couplés au regroupement avec les K-moyennes standard et les K-moyennes pour les séries chronologiques. L'approche basée sur les K-moyennes des séries

chronologiques semble beaucoup plus intéressante que la première, car elle permet de ressortir toutes les variations du comportement de pilotage (profil de pilote) durant les différents segments de décollage. Il en ressort 2 profils comportementaux de pilotage qui représentent en quelque sorte le niveau d'agressivité lors de l'intervention des pilotes (contrôle de direction).On retrouve effectivement des pilotes ayant des classes de profils différentes selon les segments de décollage. Néanmoins, même si nous croyons que malgré les variabilités du gain observé, il devrait y avoir un profil dominant qui serait alors caractéristique du pilote, des études supplémentaires sont nécessaires pour valider cette croyance en adéquation avec ce qui est connu dans la littérature.

CHAPITRE IV

MODÈLES APPRIS DES COMPORTEMENTS DE PILOTAGE

Nous allons dans ce chapitre construire un modèle appris des données réelles de décollage. C'est un modèle générique de tâches de pilotage (plus précisément de décollage) qui va prendre en entrée un ou plusieurs paramètres de vol (état de l'avion, variables contextuelles, variables environnementales,etc) qui ont un lien avec l'action que va effectuer le pilote et produire des actions à faire en fonction des évènements observés et selon les segments de décollage bien défini pour l'Airbus 320. Ce modèle peut prédire l'action de contrôle à effectuer au fil du temps à partir des données de séries temporelles. Le modèle résultant vise à imiter les compétences d'un pilote humain pour une tâche de décollage. Pour chaque segment, nous allons mettre en place un modèle qui apprend comment un groupe de pilotes se comporte. Le modèle apprend les comportements de pilotage et peut prédire les actions à effectuer sur la base d'observations spécifiques qui représentent les entrées du modèle.

4.1 Implémentation

Pour chaque segment de décollage, nous avons fait deux expériences et obtenu deux modèles de prédiction différents. La première expérience consiste à faire un apprentissage sur des données modifiées dans ce sens ou on aura à faire dépen-

damment du cas de figure un étirement (suréchantillonnage) ou une compression (sous-échantillonnage) des données d'apprentissage. La seconde expérience est effectuée sur les données de vols sans aucune modification. Tout au long du travail, tous les modèles générés avec l'expérience 1 seront identifiés par $M1$ et ceux de l'expérience 2 seront identifiés par $M2$. Nous rappelons que l'apprentissage ne s'effectue que sur quelques segments du décollage identifiés dans le chapitre précédent, où l'apprentissage est possible.

4.1.1 Segment5 : Procédure du *pitch stick* pour contrer l'effet du réglage de la poussée du décollage (procédure spécifique à Airbus)

Prétraitement

- Collecte des données : La durée pour ce segment va de 60 à 90 dixièmes de seconde dépendamment des pilotes. La Figure 4.1 présente la distribution des pilotes selon la durée prise pour effectuer ce segment. En abscisse on a le temps en dixième de seconde et en ordonnée le nombre de pilotes ayant effectué le segment avec cette durée là. On peut remarquer que la majorité effectue ce segment en sensiblement 70 dixièmes de seconde.
- Échantillonnage des données : Pour l'expérience 1, nous avons reéchantillonné les données pour que tous les vecteurs aient la même longueur. Nous avons ajusté toutes les données pour que les pilotes aient un nombre fixe de secondes de décollage. Nous avons 4 options pour le choix du temps de décollage : le temps de décollage minimum, le temps maximum, le temps médian et le mode. Pour chaque cas de figure, nous avons calculé la moyenne des écarts entre les différents temps de décollage et les différents choix (médiane, mode, minimum, maximum) que nous appelons ici écart moyen. L'écart moyen par rapport au temps médian s'est avéré le plus petit pour ce cas et pour tous les autres. Le temps médian est donc le plus optimal. Pour

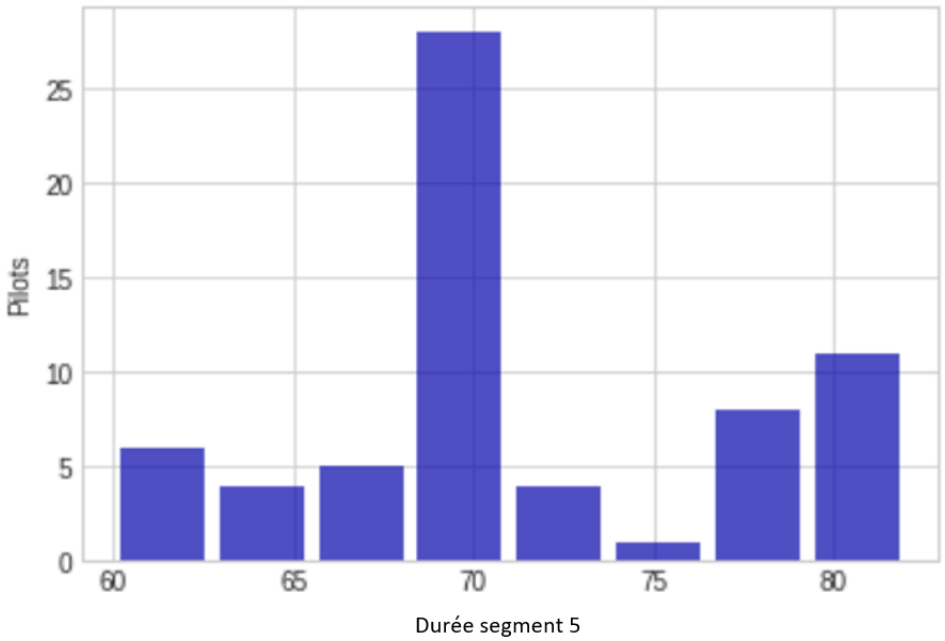


FIGURE 4.1 Durée mise par les pilotes pour effectuer le segment 5.

ce segment, le reéchantillonnage se fait pour obtenir le temps de collecte de données égal à 70 dixièmes de secondes (temps médian). La prédiction est faite après une dixième de seconde d'observation au vu des intervalles de sorties de données collectées par le simulateur X-Plane qui sera utilisé pour les tests.

Construction du modèle

Le modèle qui est construit pour ce segment prédit la valeur du manche de tangage ou *pitch stick* en fonction des entrées (voir Figure 4.2).

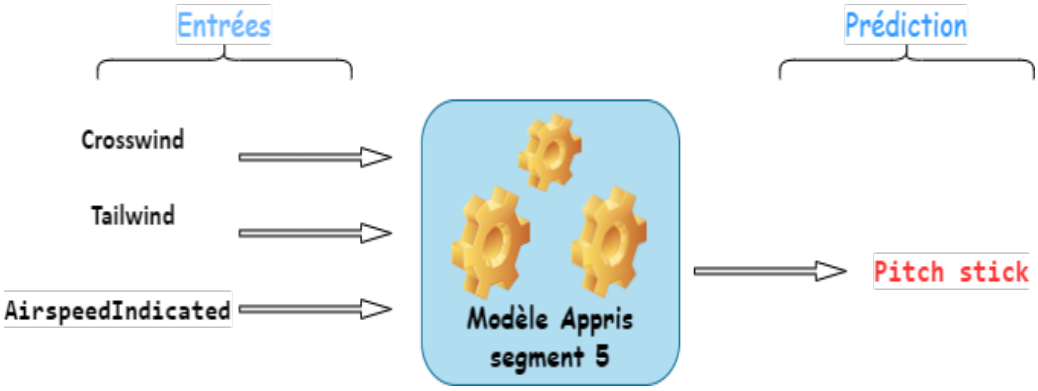


FIGURE 4.2 Illustration du modèle appris sur le segment 5.

Une fois l'étape de prétraitement des données terminée, les réseaux neuronaux récurrents sont utilisés pour générer différents modèles d'apprentissage à partir des données de pilotage capturées. Ces données ont été partitionnées en données d'entraînement qui représentent 60 pilotes, et en données de tests qui représentent 7 pilotes et qui serviront pour l'évaluation du modèle. La méthode pour choisir les paramètres des modèles dans ces expériences est basée sur une règle empirique (Heaton, 2008) qui indique qu'en général, ce genre de problème ne nécessite pas beaucoup de couches cachées. Cette règle suit une approche qui tente d'éviter le sous-ajustement causé par trop peu de neurones dans la couche cachée, ou le surajustement causé par trop de neurones, en ayant un nombre de neurones cachés inférieur ou égal à 2 fois la taille de la couche d'entrée.

La Figure 4.3 montre l'architecture générale du modèle appris du segment 5 qui prédit la valeur du *pitch stick*, ainsi que les entrées et sorties de chaque couche. L'entrée de ce modèle consiste en un vecteur de taille (1,3). La première valeur est la longueur de la séquence qui est le temps d'observation du modèle avant la prédiction (dixième de seconde), et la deuxième valeur correspond au nombre de paramètres d'entrée (figure 4.2).

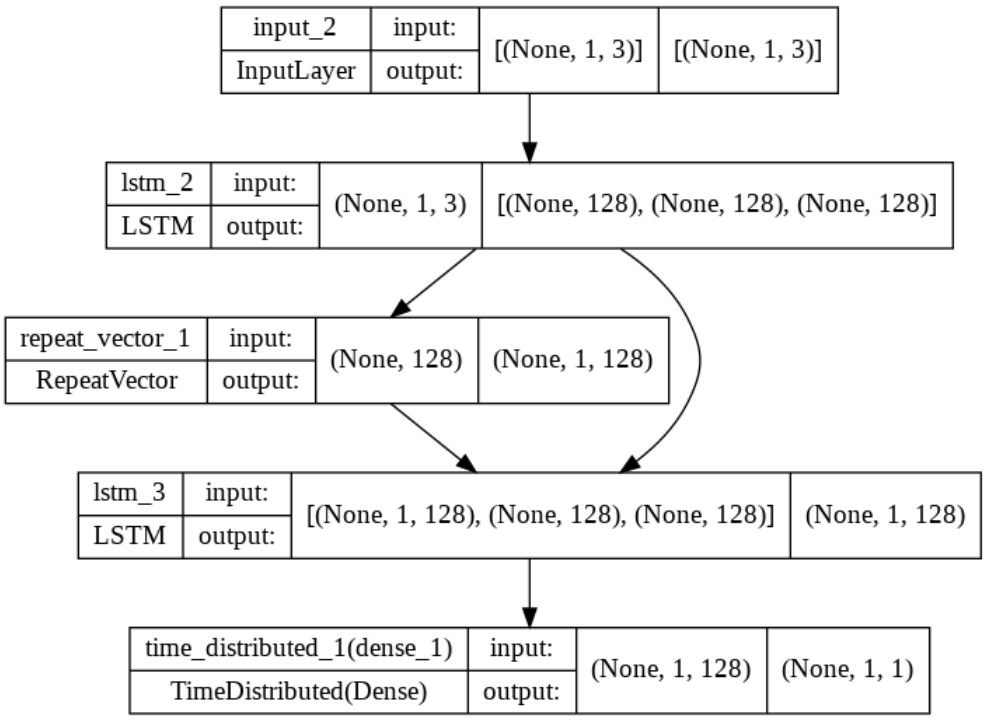


FIGURE 4.3 Architecture du modèle appris du segment 5.

Évaluation préliminaire du modèle : MAE (*Mean Absolute Error*)

L'erreur absolue moyenne (MAE) représente la différence moyenne entre la valeur réelle et la valeur prédite en valeur absolue. Plus il est bas, mieux est le modèle étant donné qu'un MAE=0 signifie que le prédit et le réel sont égaux. En pratique, on ne vise pas un MAE=0 de peur du surapprentissage. Nous évaluons notre modèle sur les données de tests. Notre MAE pour les 2 expériences est de 5. Un MAE de 5 ici veut dire que : si on prend un *pitch stick* réel égal à 50, la prédiction sera de $50 + 5 = 55$ ou de $50 - 5 = 45$. La perte est peut-être ou pas significative, car les données sont sélectionnées au hasard dans le jeu de données de tests, mais ils restent dans le cadre de référence (intervalle acceptable de valeurs (voir annexe B). De plus, nous ne nous limitons pas à cette métrique pour évaluer les différents

modèles.

4.1.2 Segment 6 : Rotation

Prétraitement des données

— Collecte des données : Les données pour ce segment sont relevées 3 secondes avant que la vitesse indiquée (*AirspeedIndicated*) n'atteigne la vitesse de rotation (*VspeedVr*) jusqu'à ce que l'altitude (*AltitudePressure*) croît (début du segment7 : l'avion quitte le sol). Comme on peut le voir sur la figure 4.4, la durée va de 17 à 107 dixièmes de secondes dépendamment des pilotes. En abscisse on a le temps en dixième de secondes et en ordonnée le nombre de pilotes ayant effectué le segment en ce temps-là. On peut remarquer que la majorité de pilotes effectue ce segment en un temps situé entre 80 et 110 dixièmes de secondes. Mais le temps médian est de 90 dixièmes de secondes.

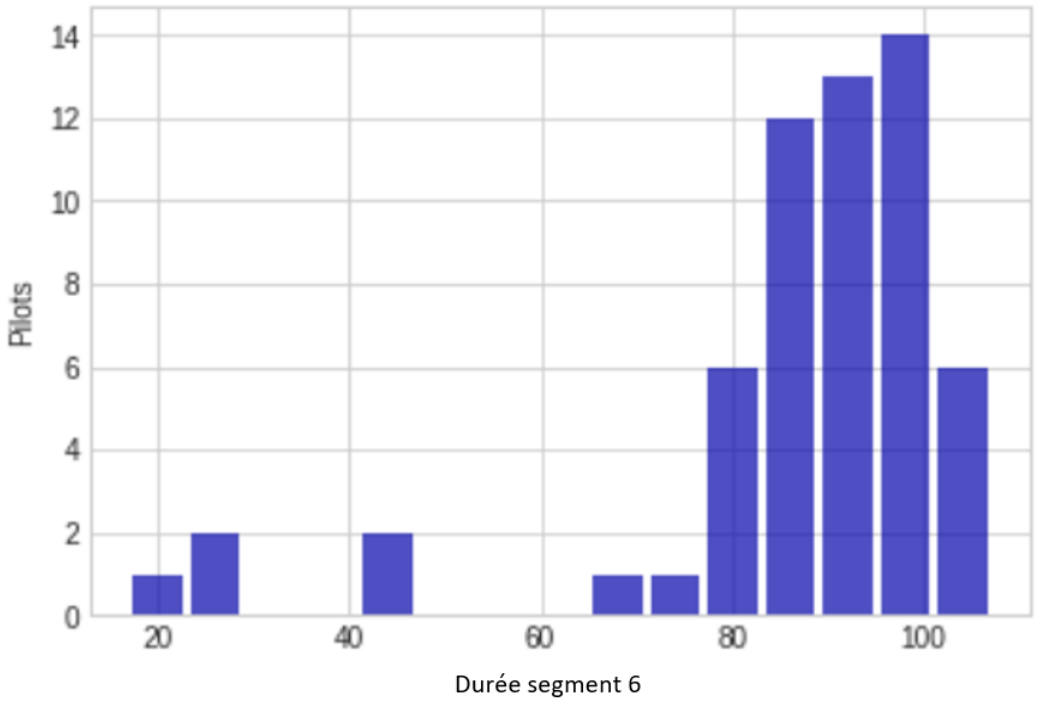


FIGURE 4.4 Durée mise par les pilotes pour effectuer le segment 6

— Échantillonnage des données : Comme le montre la Figure 4.4 les pilotes effectuent ce segment de décollage avec une durée différente. Pour l'expérience 1, nous avons reéchantillonné les données pour que les vecteurs aient la même longueur. Nous avons ajusté toutes les données pour que tous les pilotes aient un nombre fixe de secondes de décollage (90 dixièmes de seconde qui est le temps médian).

Construction du modèle

Le modèle qui est construit pour ce segment prédit la valeur du *pitch stick* en fonction des entrées (voir Figure 4.5).

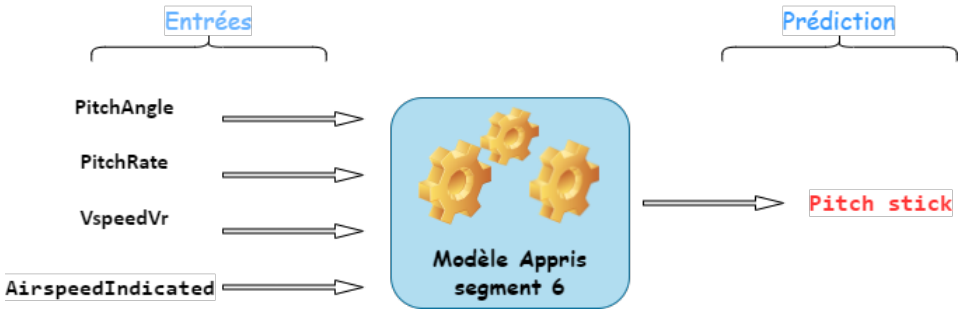


FIGURE 4.5 Illustration du modèle appris sur le segment 6.

Une fois les données prétraitées, ceux-ci ont été partitionnées en données d'entraînement et données de test (qui serviront pour l'évaluation du modèle). La figure (4.6) montre l'architecture générale du modèle appris du segment 6 qui prédit la valeur du *pitch stick*, ainsi que les entrées et sorties de chaque couche.

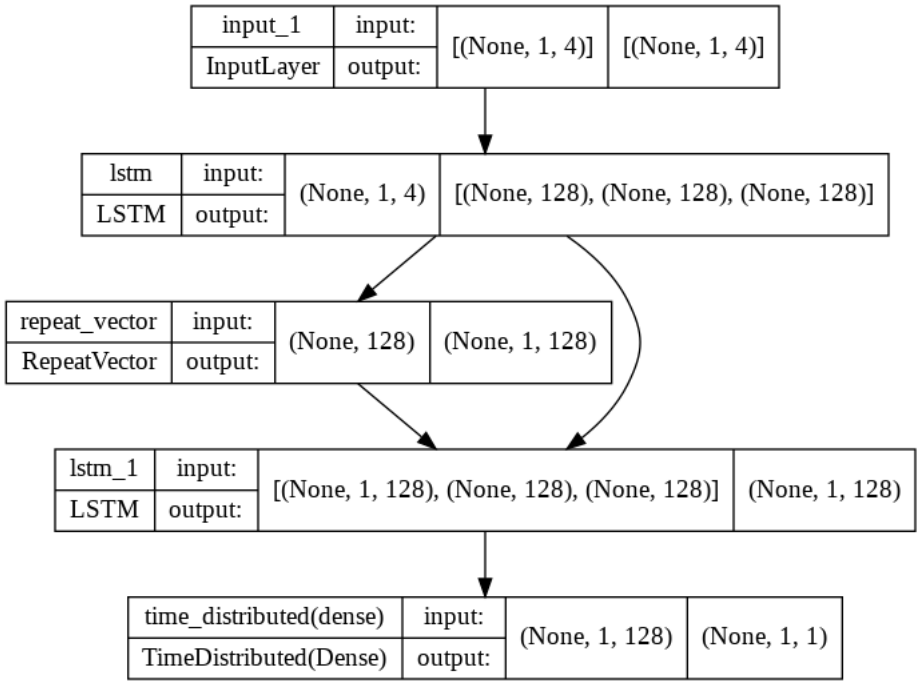


FIGURE 4.6 Architecture du modèle appris sur le segment 6.

L'entrée de ce modèle consiste en un vecteur de taille (1,4). La première valeur est la longueur de la séquence (le dixième de seconde qui est le temps d'observation du modèle avant la prédiction), et la deuxième valeur correspond au nombre de paramètres d'entrée .

Évaluation préliminaire du modèle : MAE (*Mean Absolute Error*)

Le modèle est ensuite évalué sur les données de test. Notre MAE est de : 2 pour l'expérience 1 et 2.5 pour l'expérience 2. Un MAE de 2 ici veut dire que : si nous prenons un *pitch stick* réel égal à 50, la prédiction sera de $50 + 2 = 52$ ou de $50 - 2 = 48$. Si on ne s'en tient qu'au MAE. Le modèle dans lequel les données ont été reéchantillonnées (M1) est plus performant que celui sans modification de données (M2).

4.1.3 Segment 7 : Décollage

Prétraitement

— Collecte des données : Les données sont relevées à partir du moment où l'avion quitte le sol jusqu'à la fin du décollage. La durée va de 105 à 908 dixièmes de secondes (voir figure 4.7) dépendamment des pilotes. En abscisse on a le temps en dixième de seconde et en ordonnée le nombre de pilotes ayant effectué le segment en ce temps-là. Le temps médian ici est de 535 dixièmes de secondes.

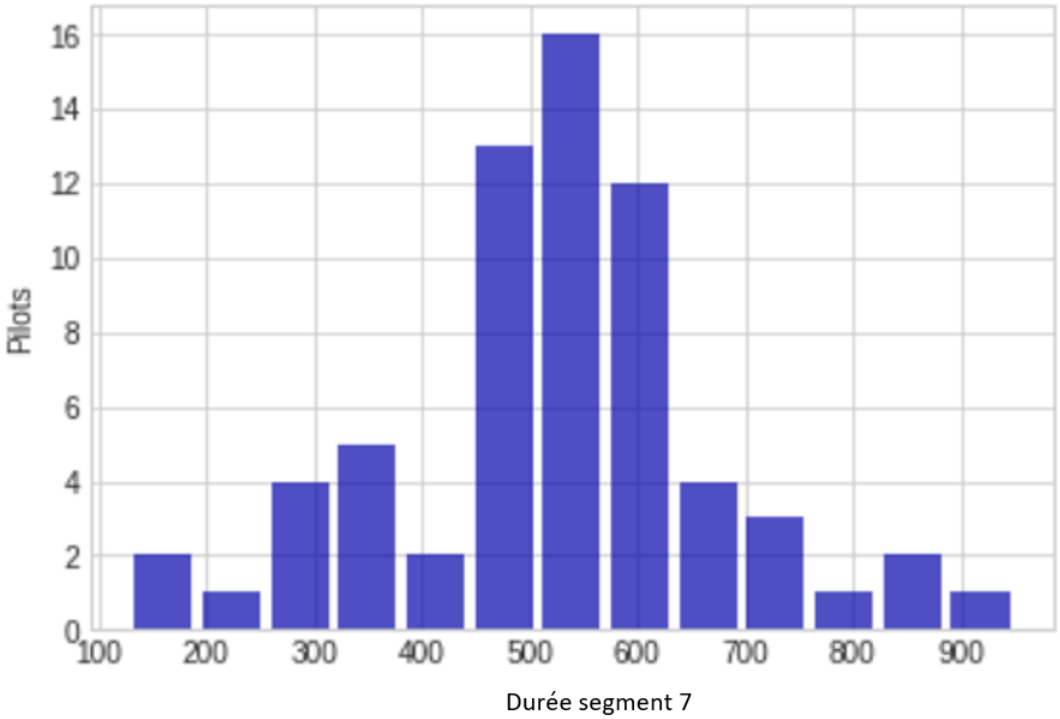


FIGURE 4.7 Durée mise par les pilotes pour effectuer le segment 7

— Échantillonnage des données : comme le montre la Figure 4.7, les pilotes effectuent ce segment de décollage avec une durée différente. Pour la première expérience, nous avons ajusté toutes les données pour que tous les pilotes aient un nombre fixe de secondes de décollage. Pour ce segment, le rééchantillonnage se fait pour obtenir le temps de collecte de données égal à 535 dixièmes de seconde.

Construction du modèle

Le modèle appris pour ce segment est un ensemble de trois modèles d'apprentissage automatique prédisant respectivement les valeurs du manche de tangage, le

manche de roulis et les pédales de direction en fonction des entrées. La figure 4.8 illustre les entrées et les sorties du modèle appris.

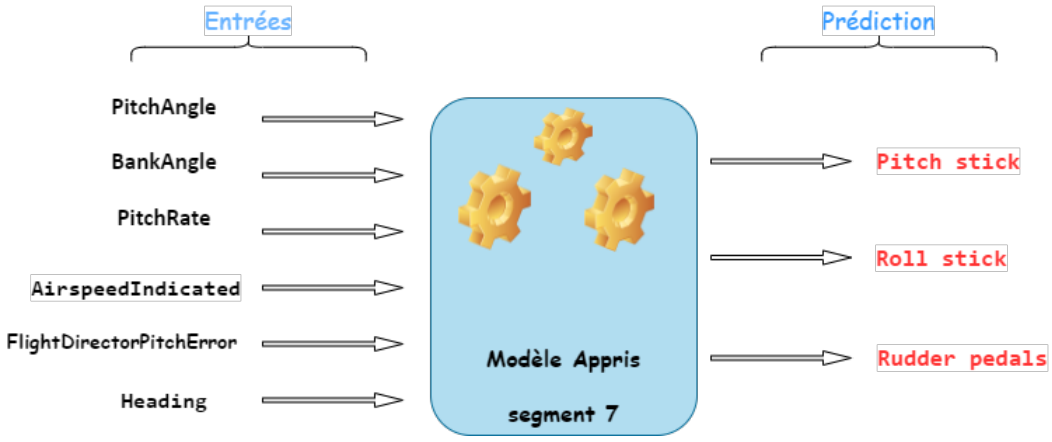


FIGURE 4.8 Illustration du modèle appris sur le segment 7

Les entrées correspondent à l'état et au contexte de l'avion, qui doivent être observés pendant la phase de décollage et de montée. Les actions à exécuter ou les sorties des modèles correspondent aux actions du pilote sur le manche de tangage, manche de roulis et pédales de direction. En d'autres termes, le modèle appris basé sur l'état actuel de l'avion doit prédire les valeurs des trois entrées du pilote afin que l'action à exécuter (décollage vers la montée) soit bien exécutée.

Une fois les données prétraitées, elles ont été partitionnées en données d'entrainement et données de test. La figure 4.9 présente l'une des 3 architectures des modèles appris du segment 7 qui prédit la valeur du *pitch stick*, ainsi que les entrées de chaque couche. L'entrée de ces modèles consiste en un vecteur de taille (1,6). La première valeur est la longueur de la séquence qui représente le dixième de seconde et qui est le temps d'observation avant la prédiction. La deuxième valeur correspond au nombre de paramètres d'entrée (voir Figure 4.8).

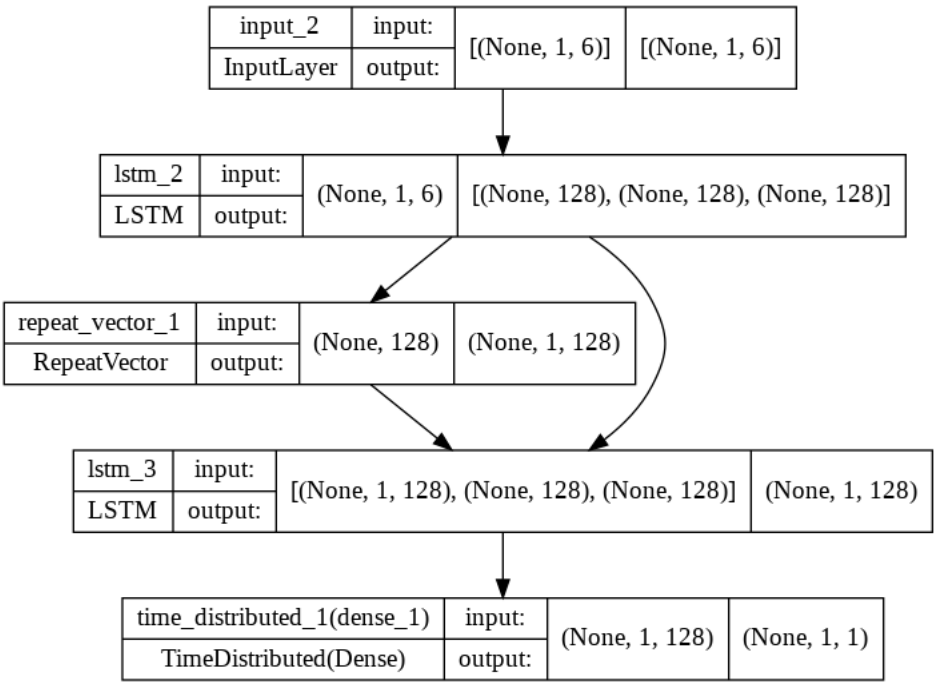


FIGURE 4.9 Architecture du modèle appris sur le segment 7

Évaluation préliminaire du modèle en utilisant le MAE

Le modèle est ensuite évalué sur les données de test. Notre MAE est de : 4,6,1 respectivement pour les différentes prédictions de l'expérience 1 et 3, 2.5 ,2.5 pour l'expérience 2. Si on ne s'en tient qu'au MAE, le modèle dans lequel les données ont été reéchantillonnées (M1) est plus peformant pour la prédiction du *pitch stick* et du *rudder pedals* de performant que celui sans modification de données (M2)..

4.2 Résultats et discussions

4.2.1 Segment5

La figure 4.10 présente un exemple concret de prédiction pour ce segment. Dans cette figure, les données du pilote représentées n'ont pas fait partie des données d'apprentissage, mais plutôt utilisées comme données de test. D'une part (les 3 graphiques du haut de la figure 4.10) nous avons les paramètres d'entrée du modèle qui sont le vent arrière, le vent de travers et la vitesse indiquée en bleu et d'autre part (les 2 graphiques en bas) les paramètres de sorties qui sont le *pitch stick* réel et le prédit. La courbe en rouge représente le *pitch stick* réel du pilote, les courbes en vert et noir représentent les valeurs prédites de celui-ci selon les 2 expériences.

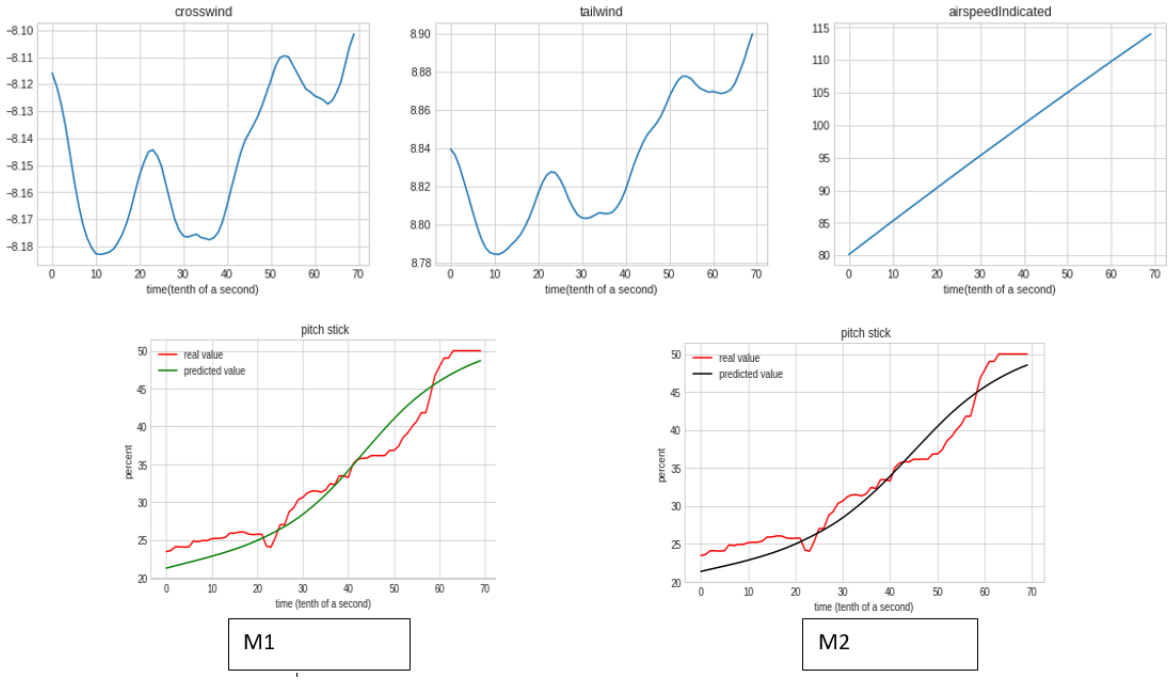


FIGURE 4.10 Exemple de prédiction sur le segment 5.

Lorsqu'on s'intéresse aux entrées du pilote, on peut remarquer la présence du vent arrière et du vent de travers inférieur à 20 noeuds. Le pilote essaie de régler son *pitch stick* à 50 (courbe en rouge). La prédiction du *pitch* selon les différents modèles est pratiquement identique et se rapproche plutôt bien de la manœuvre du pilote. Le but ici n'étant pas que la courbe en rouge soit exactement celle en vert ou en noir mais que la prédiction soit un comportement normal et acceptable.

4.2.2 Segment6

Les figures 4.12 et 4.13 présentent deux exemples concrets de prédiction pour ce segment. Nous avons considéré les données de deux pilotes n'ayant pas participé à l'apprentissage. D'une part nous avons pour chaque pilote les paramètres d'entrée du modèle (la vitesse indiquée, le taux de tangage, l'angle de tangage et la vitesse VR) en bleu et d'autre part le paramètre de sortie qui est le*pitch stick* réel et prédit. La courbe en rouge représente le *pitch stick* réel du pilote, les courbes en vert et noir représentent les valeurs prédites.

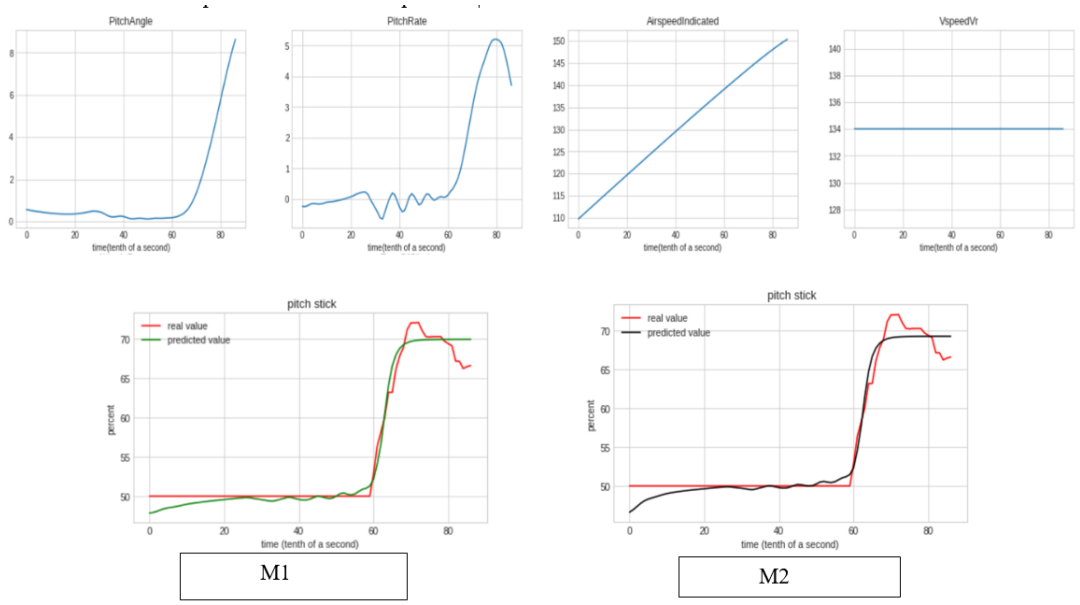


FIGURE 4.11 Exemple 1 de prédiction sur le segment 6.

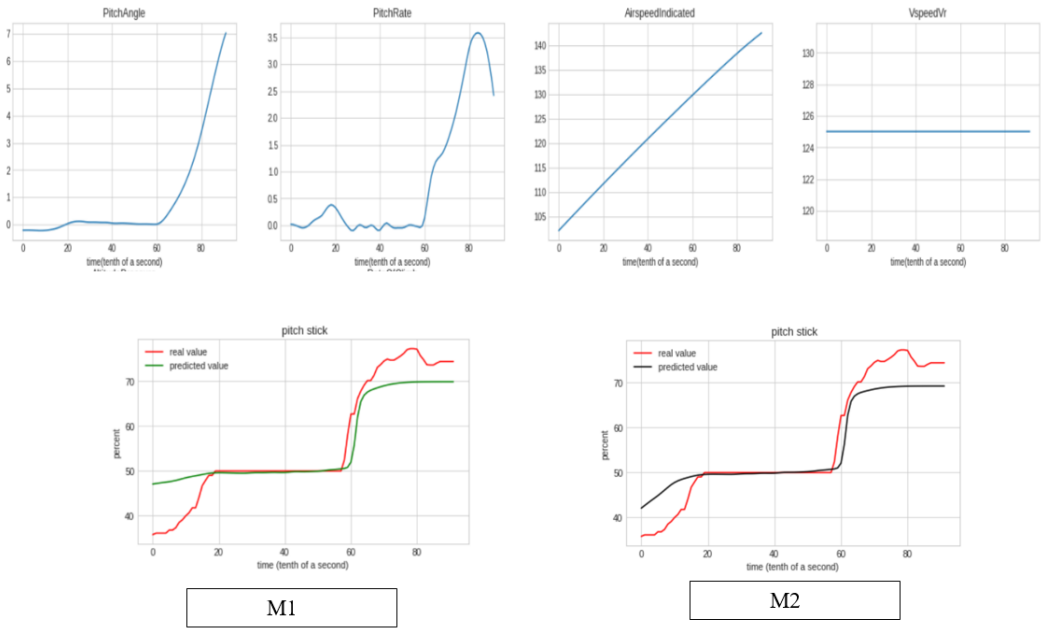


FIGURE 4.12 Exemple 2 de prédiction sur le segment 6.

Dans les 2 cas de figure, on constate que les pilotes tirent sur le manche de manière à dépasser les 50 pour cent et atteindre un taux de tangage ($PitchRate$) d'environ trois degrés lorsque la vitesse indiquée atteint la vitesse Vr qui est de 134 pour le pilote 1 et 125 pour le pilote 2. Au regard des prédictions faites par le modèle $M1$, les dix premières dixièmes de secondes se rapprochent le mieux de la réaction du pilote 1 et inversement pour le pilote 2. Mais tous deux tentent de faire une rotation. Les simulations sur X-Plane permettront de visualiser ces actions.

4.2.3 Segment7

La figure 4.13 présente un exemple concret de prédiction pour ce segment. D'un côté nous avons les paramètres d'entrée du modèle en bleu et d'un autre côté les paramètres de sorties et les prédictions selon différents modèles (à l'échelle de données réelle : comme fournie dans les données). Si on ne s'en tient qu'au MAE, le modèle 2 (expérience effectuée sans modification de données) performe mieux ou alors s'approche le mieux du comportement réel du pilote lors de la prédiction du *pitch stick* et du *rudder pedals* contrairement au *roll stick* du pilote.

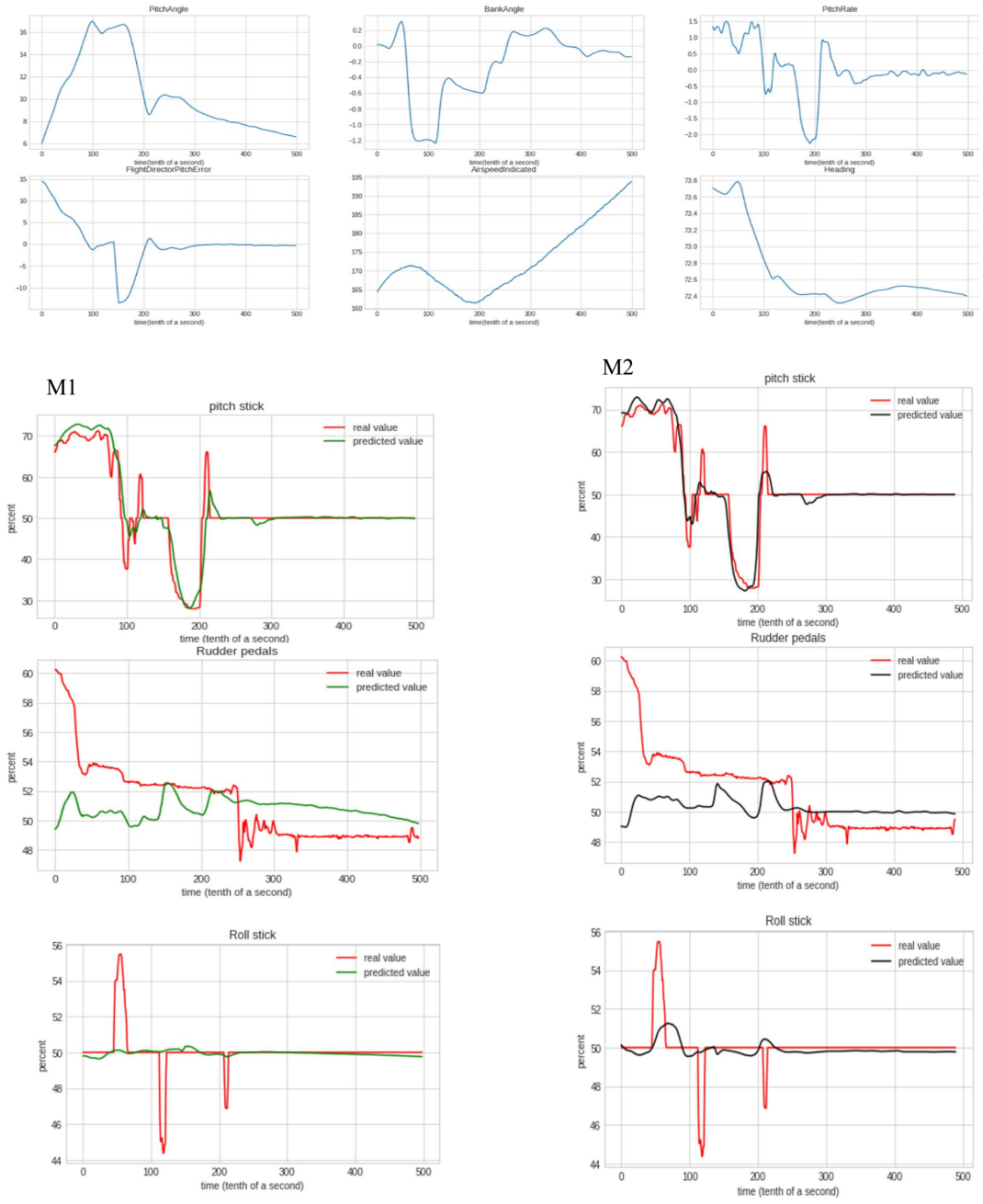


FIGURE 4.13 Exemple de prédiction sur le segment 7.

On peut constater ici que, le pilote relâche progressivement la pression sur les pé-

dales de direction (*rudder pedals*). Par contre, ce relâchement n'est pas réellement observé au niveau des prédictions que ce soit lors de la première ou deuxième expérience. Les prédictions reflètent plutôt une constance sur les pédales de direction. Le pilote essaye de maintenir le niveau des ailes via le *roll stick*. On observe plus cette constance dans les données prédites que l'action réelle du pilote (principalement dans l'expérience 1 : $M1$). Lorsque l'avion atteint 15 degrés d'angle de tangage, le pilote ajuste l'inclinaison de l'avion pour suivre les commandes du directeur de vol pour aider le pilote à maintenir une vitesse indiquée, on observe donc les variations du *pitch stick* à cet effet. Les *pitch* prédits ici se rapprochent beaucoup du *pitch* réel. Comme précisé plus haut, le but n'est pas d'imiter le pilote en rouge, car le modèle de prédiction est une sorte de représentation (au niveau du comportement) de tous les pilotes utilisés lors de l'apprentissage.

4.3 Conclusion

À partir des modèles neuronaux entraînées de type LSTM, nous avons pu ressortir pour chaque segment un comportement général de pilotes appris sur les données fournies. Les expériences ont montré la capacité du modèle appris à capturer les taches de haut niveau. Les prédictions faites jusqu'ici passent l'évaluation préliminaire dans ce sens ou les valeurs prédites se situent dans les intervalles de valeurs de référence spécifiés par les experts(voir annexe B). Plus il y aura de données à fournir aux algorithmes d'intelligence artificielle, mieux l'apprentissage sera. Les modèles développés sont destinés à être utilisés à des fins de simulation, par exemple, dans le cas où nous voulons prédire comment un pilote se comportera devant une situation spécifique. La prochaine étape de cette étude consistera à valider notre hypothèse en réalisant des exécutions directement dans le simulateur : X-Plane.

CHAPITRE V

VERS UN CADRE D'EXÉCUTION DU MODÈLE APPRIS

Maintenant que nous avons un modèle construit à partir des données réelles de pilotage, il peut être intéressant de le situer dans un contexte d'exécution d'une phase ou d'un segment de vol complet. Un agent simulant une telle exécution du modèle appris (que nous appelons pilote synthétique) serait utile pour simuler des tests ou pour permettre une assistance aux pilotes (suggérer ou corriger les actions lors d'une formation). Dans le cadre du projet piloteIA, il est d'ailleurs prévue d'utiliser le modèle appris pour étendre la mémoire procédurale d'un agent cognitif de pilotage (Tchio *et al.*, 2023). l'objectif du pilote synthétique n'est certainement pas de remplacer ni le pilote humain ni le pilote automatique mais d'être utilisé pour synthétiser et démontrer des comportements de pilotage qui peuvent être d'un certains intérêt. Ce chapitre présente une conception et une première réalisation de ce cadre. Le but est de permettre l'exécution automatique du modèle appris sur une tache complète de décollage d'un Airbus320 dans X-Plane.

5.1 Interfaçage avec X-Plane

5.1.1 Définition

X-Plane est une série de moteurs de simulation de vol, développée et publiée par *Laminar Research* depuis 1995. X-Plane est utilisé par plusieurs organisations et industries telles que la NASA, Boeing, Cirrus, Cessna Piper, Japan Airlines, etc. X-Plane peut communiquer avec des applications externes en envoyant et en recevant des données sur l'état du vol et les commandes de contrôle (Baomar et Bentley, 2016). X-Plane est préemballé avec plusieurs avions commerciaux et militaires, ainsi que des paysages mondiaux, qui couvrent la majeure partie de la terre. Il dispose également d'une architecture de *plugin* qui permet aux utilisateurs de créer leurs propres modules, étendant les fonctionnalités du logiciel en permettant aux utilisateurs de créer leurs propres mondes ou répliques de lieux sur Terre (Ross, 2022). Pour ce travail, le simulateur (la version 11) est configuré pour envoyer et recevoir des paquets contenant les données souhaitées toute les 0.1 seconde.

5.1.2 API de communication avec X-Plane

Pour supporter la communication avec l'environnement de simulation X-Plane, une API a été développée (par un autre étudiant faisant partie du projet) pour permettre d'obtenir les valeurs des paramètres de l'environnement et modifier ceux-ci. Les fonctions phares qui seront utilisées sont les suivantes :

— *getDataRef()* : cette fonction est celle qui permet d'obtenir la valeur d'une référence donnée de l'environnement de simulation. Pour obtenir une valeur, il suffit d'identifier la référence donnée du paramètre voulu. Une fois identifiée, la fonction retourne la valeur en temps réel. La fonction retourne

la valeur d'un paramètre, l'intérêt est de l'utiliser pour l'évaluation des contraintes de l'environnement liées aux segments de décollage.

— *setDataRef()* : cette deuxième fonction permet au client de modifier la valeur d'un paramètre (*dataref*) de l'environnement. L'intérêt de modifier une valeur est de pouvoir représenter ou faire jouer les actions du pilote prédit par les modèles.

5.2 X-Plane Runner

5.2.1 Architecture

La figure 5.1 présente l'architecture d'implémentation du cadre d'exécution du modèle appris qui est une exécution du modèle appris et qui couple ce modèle appris à un modèle de référence. Cette implémentation comprend plusieurs étapes.

Charger les ontologies

Nous nous concentrons sur une approche basée sur des ontologies créées au sein du projet. La création de ces ontologies, dont le but est la modélisation des connaissances procédurales complexes liées aux procédures de pilotage d'aéronefs, se base sur les connaissances extraites à partir de la littérature et provenant d'experts du domaine. Les ontologies utilisées (Courtemanche *et al.*, 2022) ont été entièrement élaborées par un étudiant faisant partie du projet. Elles présentent le cadre de référence pour l'exécution des procédures de pilotage d'aéronef. Cette référence d'exécution représente la théorie de pilotage qui doit être appliquée en réponse à diverses situations posées par l'environnement. Autrement dit, il s'agit d'un dispositif capable d'identifier l'action qu'un pilote doit accomplir selon ce que prévoit la théorie de pilotage. L'avantage de l'utilisation des ontologies ici est le haut niveau

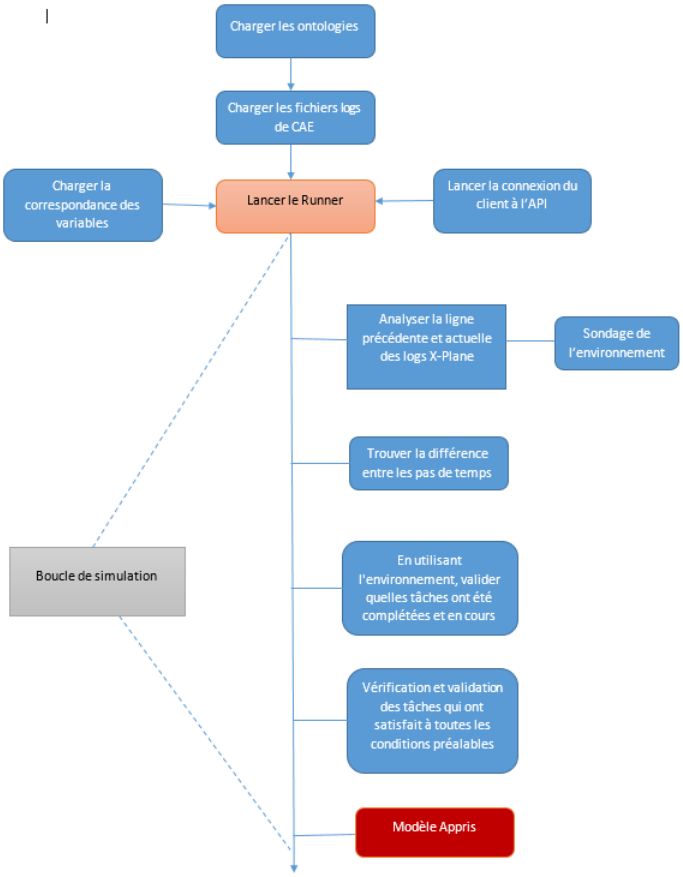


FIGURE 5.1 Architecture du runner.

d'expressivité que permettent les différents langages et leur sémantique. Il y a 2 ontologies qui ont été créées dans ce projet : l'ontologie du domaine et l'ontologie de tâches.

L'ontologie du domaine : il s'agit d'un ensemble de paramètres de l'environnement d'exécution de la tâche. Autrement dit, il représente les composantes de vol extérieures, les éléments du poste de pilotage et les systèmes de l'aéronef. La figure 5.2 présente les différents sous-environnements de l'ontologie du domaine.

— Environnement : il s'agit de la classe la plus générale d'où est spécialisé

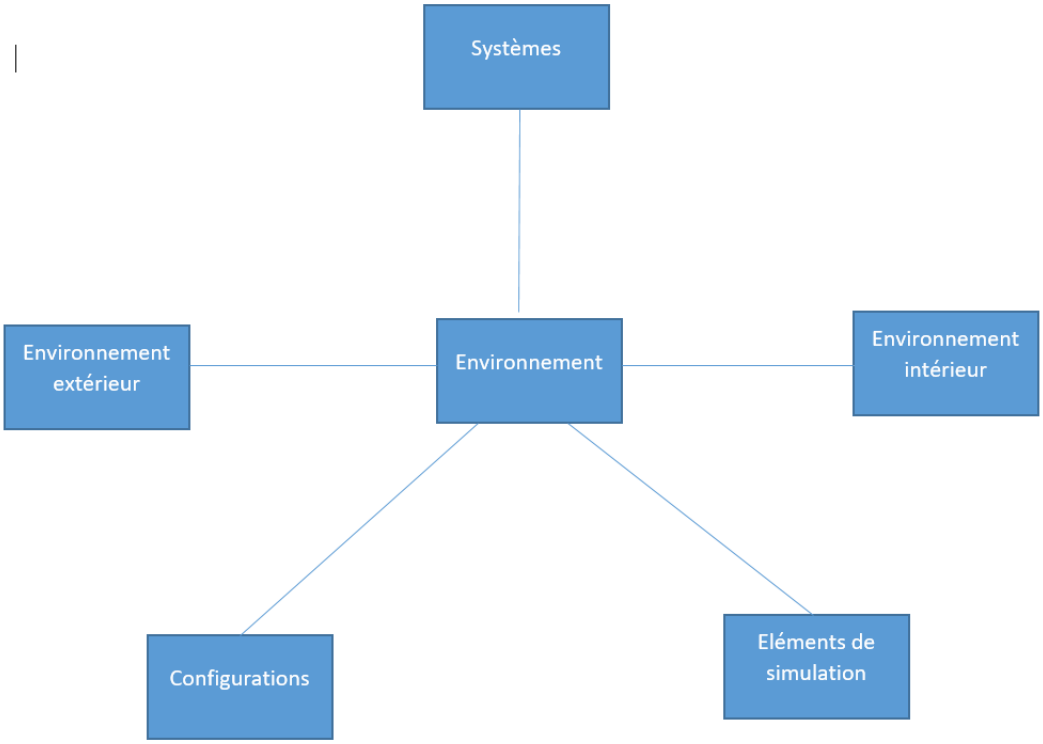


FIGURE 5.2 Représentation simplifiée de l'ontologie du domaine.

chacun des prochains sous-environnements.

— Environnement intérieur : représente l'ensemble des éléments de l'environnement à l'intérieur de l'aéronef.

— Environnement extérieur : l'ensemble des paramètres de vol liés à l'extérieur de l'aéronef.

— Configuration : Nous pouvons y retrouver un ensemble de paramètres variables en fonction du contexte et qui est utilisé au travers des scénarios.

— Élément de simulation : Cette spécialisation contient certains paramètres essentiels pour assurer le bon fonctionnement d'une simulation.

L'ontologie de taches (voir annexe D) : L'ontologie de tâche est un ensemble terminologique qui structure le processus d'exécution d'une procédure de pilotage. Autrement dit, les procédures de pilotage sont décomposées à partir de l'ensemble terminologique de l'ontologie de tâche en incluent de forts liens sémantiques vers l'ontologie du domaine.

Dans le but de rendre l'interpréteur plus accessible, la bibliothèque *OWLReady* (Jean-Baptiste, 2021) a été sélectionnée. Le cadre d'exécution du modèle appris (pilote synthétique) (*Runner*) charge donc les fichiers d'ontologies et y accède via *owlready2*. La première étape du module nécessite le chargement de l'ontologie de l'avion. Cette ontologie permettra au *runner* d'identifier les entrées analogiques du cockpit et les classer en fonction de ses actions et tâches.

Charger les logs CAE

Les paramètres qu'utilise l'ontologie sont directement liés à ceux du simulateur de CAE (d'où proviennent les données des différentes expérimentations). Les logs CAE sont chargés dans un fichier à la fois (ces fichiers sont spécifiés à l'initialisation). Le journal est divisé en trois sections : entrées du pilote, contexte de l'aéronef et état de l'aéronef. En conséquence, il y a une séparation de données pour chacune de ces catégories. Cela permet aux modèles qui interagissent avec le "*runner*" de vérifier rapidement les changements dans les entrées, le contexte ou l'environnement.

Charger la correspondance des variables

Afin de valider avec succès les tâches ontologiques, ces contraintes doivent être comparées en utilisant les mêmes unités et étiquettes. Les variables du simulateur X-Plane et du simulateur de CAE n'ont pas la même nomenclature ni la même

valeur. Il est question ici de charger une bibliothèque faisant le pont entre les deux. Il est important de noter que chacun de ces dictionnaires contient également une correspondance de leurs facteurs d'échelle respectifs. L'annexe E présente la structure du dictionnaire de correspondance des deux variables.

Connexion avec l'API

La connexion avec l'API consiste à :

- démarrer X-Plane ;
- attendre d'être dans le cockpit de l'avion (prêt à décoller) ;
- accéder au dossier *XplanePackage1.0.0* ;
- modifier le fichier subscribe.txt pour ajouter ou supprimer les datarefs au besoin
- Exécuter *libXplane-udp-client.exe* afin de communiquer avec le serveur.

Boucle de simulation

La simulation en fin de compte consiste à analyser la ligne actuelle et précédente des logs afin de capturer les changements d'états (l'évolution des variables) entre les pas de temps. Ces changements d'état, qui correspondent aux entrées du modèle, vont nourrir le modèle appris qui va à son tour prédire les actions à effectuer. L'utilisation de l'environnement va permettre de valider quelles tâches ont été accomplies et quelles tâches sont en cours d'exécution et aussi valider les tâches qui ont réussi.

Modèle appris

La figure 5.3 représente la boucle de prédiction (architecture globale du modèle appris basé sur les différents modèles présentés dans le chapitre 4). Pour chaque modèle sont précisés les paramètres d'entrées et ceux prédits. Le segment 5 est déclenché une fois que l'avion atteint une vitesse indiquée de 80 kts et se termine lorsque cette vitesse atteint le neutre qui est 100 kts. Le segment 6 est déclenché 2 à 3 secondes avant que la vitesse indiquée n'atteigne la vitesse Vr et se termine lorsque le segment 7 commence : altitude à 5 pieds. Le segment 7 est déclenché une fois que l'avion atteint une altitude de 5 pieds et se termine lorsque l'avion atteint une altitude de 3200 pieds. Les fonctions *getDataref* et *setDataref* servent successivement à récupérer et à renvoyer les paramètres à l'environnement. Tout seul, ce modèle ne pourra pas s'exécuter. Il faut le coupler au modèle de référence pour l'encadrer et afin que celui-ci effectue toutes les autres tâches connexes du pilotage dont les segments non pris en compte par le modèle appris.

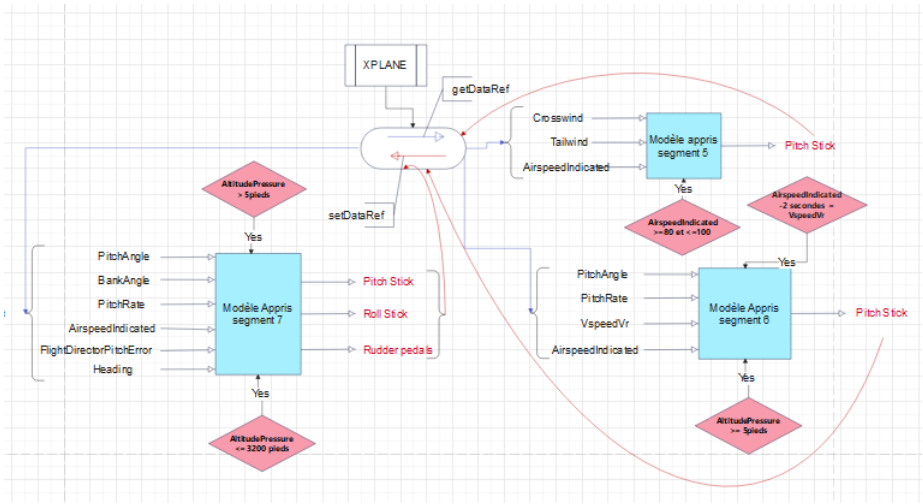


FIGURE 5.3 Architecture finale du modèle appris.

TABLEAU 5.1 Entrées du modèle appris du segment 5.

Nom	Dataref de CAE (x)	Dataref de X-Plane (y)	Conversion
Entrées météo			
crosswind	StandardAircraft/ WindV	sim/weather/ wind_speed_kt[1]	x = y
tailwind	StandardAircraft/ WindU	sim/weather/ wind_speed_kt[0]	x = y
Sortie de l'avion			
AirspeedIndicated	StandardAircraft/ AirspeedIndicated	sim/flightmodel/position/ indicated_airspeed	x = y

5.3 Exécution du modèle appris

5.3.1 Mise en correspondance des paramètres des différents modèles

Comme dit précédemment, le décollage comprend dix segments, mais seuls trois nécessitent un apprentissage automatique, à savoir les segments 5, 6 et 7. Différents modèles ont été développés pour chaque segment, chacun avec ses entrées et ses sorties. Pour chaque modèle, nous faisons correspondre leurs paramètres c'est-à-dire des données du simulateur de CAE (données d'origine) à ceux de X-Plane (simulateur utilisé pour les expérimentations).

Modèle appris sur le segment 5

— Les entrées sont présentées dans la table 5.1.

— La sortie est présentée dans le tableau 5.2.

Tableau 5.2 Sortie du modèle 5

Action du pilote	Dataref de CAE (x)	Dataref de X-Plane (y)	Conversion
pitch stick	StandardAircraft/ Control- LongitudinalPosition	a320/Aircraft/ Cockpit/Panel/ SidestickLonL	y = (x-50)/50

Tableau 5.3 Entrées du modèle appris du segment 6

Nom	Dataref de CAE(x)	Dataref de X-Plane (y)	Conversion
Sorties de l'avion			
PitchAngle	StandardAircraft/ PitchAngle	a320/Aircraft/Control/ ELAC1/Pitch	x = y
PitchRate	StandardAircraft/ PitchRate	sim/flightmodel/-position/Q	x = y
AirspeedInd	StandardAircraft/ PitchRate	sim/flightmodel/position/ indicated_airspeed	x = y
VspeedVr	StandardAircraft/ VspeedVr	sim/cockpit2/gauges/- indicators/- ground_speed_kt	x = y

Modèle appris du segment 6

— Les entrées sont présentées dans le tableau 5.3.

— La sortie est présentée dans le tableau 5.4.

Modèle appris du segment 7

— Les entrées sont présentées dans le tableau 5.5.

— Les sorties sont présentées dans le tableau 5.7.

TABLEAU 5.4 Sortie du modèle appris du segment 6

Action pilote	Dataref de CAE (x)	Dataref de X-Plane (y)	Conversion
pitch stick	StandardAircraft/ Control- LongitudinalPosition	a320/Aircraft/ Cockpit/Panel/ SidestickLonL	y = (x-50)/50

TABLEAU 5.5 Entrées du modèle appris du segment 7

Nom	Dataref de CAE(x)	Dataref de X-Plane (y)	Conversion
Entrées de l'avion			
PitchAngle	StandardAircraft/ PitchAngle	a320/Aircraft/ Control/ELAC1/Pitch	x =y
PitchRate	StandardAircraft/ PitchRate	sim/flightmodel/ position/Q	x =y
BankAngle	StandardAircraft/ BankAngle	a320/Aircraft/Control/ ELAC1/Roll	x =y
Heading	StandardAircraft/ Heading	a320/Aircraft/Heading	x =y
Airspeed Indicated	StandardAircraft/ AirspeedIndicated	sim/flightmodel/position/ indicated_airspeed2	x =y
FlightDirector PitchError	StandardAircraft/ FlightDirector PitchError	a320/Aircraft/FMGS/ FMGC1/FlightDirPitch	x =y

Tableau 5.6 Sorties du modèle appris du segment 7

Actions du pilote	Dataref de CAE(x)	Dataref de X-Plane (y)	Conversion
pitch stick	StandardAircraft/ Control- LongitudinalPosition	a320/Aircraft/Cockpit/ Panel/SidestickLonL	y = (x-50)/50
rudder pedals	StandardAircraft/ Control- DirectionalPosition	a320/Aircraft/Cockpit/ Panel/RudderPedals	y = x/100
Roll Stick	StandardAircraft/ Control- LateralPosition	a320/Aircraft/Cockpit/ Panel/SidestickLatL	y = (x-50)/50

Tableau 5.7 Sorties du modèle appris du segment 7

5.3.2 Expérience 1 : tests indépendants

Nous testons dans cette expérience de façon indépendante les différents modèles développés afin de s'assurer que la connexion à X-Plane et les différents modèles fonctionnent correctement. Les segments statiques du décollage qui n'ont pas requis d'apprentissage sont effectués manuellement notamment :

- positionner l'avion sur le seuil de piste avec un cap correspondant au cap de la piste ;
- mettre les manettes de poussée au ralenti et appliquer les freins ;
- régler les volets sur 1 ou 2, armer les spoilers au sol et abaisser le levier du train d'atterrissage ;
- déplacer les manettes de poussée des moteurs pour obtenir des indications de moteur N1 de 50 pour cent. Cette manœuvre est nécessaire pour vérifier

si la poussée est symétrique avant d'appliquer la poussée de décollage. Les freins restent appliqués.

Une fois que l'avion commence à avancer, le pilote est censé utiliser les pédales de direction pour maintenir le cap de l'avion sur l'axe de la piste. C'est ce rôle que vont jouer nos modèles.

Pour chaque modèle, plusieurs étapes sont effectuées pour permettre leur exécution. Nous présentons ci-dessous les étapes pour le modèle appris du segment 7.

— Création d'un client et connexion avec l'API

Rappelons ici que l'API va nous permettre de faire interagir nos modèles avec X-Plane. Le client créé ici est le point d'entrée. Nous allons tester si la connexion est bien établie, car tous part de là.

— Lecture des entrées nécessaires

Une fois connecté à X-Plane , nous récupérons les valeurs de différents paramètres de l'avion qui sont : le *pitchAngle*, *Pitch rate*, *BankAngle*, *AirspeedIndicated*, *FlightDirectorPitchError*, *Heading*. Ces paramètres sont les entrées dont le modèle a besoin pour prédire les comportements du pilote.

— Définition des conditions de segments

Chaque segment de décollage débute à un moment bien précis à définir. Le segment 7 est généralement appelé décollage proprement dit, car il commence lorsque l'avion prend de l'altitude ou quitte le sol. L'altitude va en général se trouver entre 5 et 3200 pieds dans ce segment.

— Prédiction de comportement de pilotage

Une fois les entrées du modèle collectées, la prochaine étape est de prédire les différents comportements de pilotage, notamment la valeur du manche de roulis, du manche de tangage et pédales de direction sans toutefois oublier la mise à l'échelle de ceux-ci.

— Mise à jour des valeurs prédites

Les valeurs prédites doivent être retournées à l'environnement (X-Plane) pour observer l'évolution ou les variations de celui-ci d'où l'utilisation de la fonction *setDataref*.

Résultat et discussion

Cette première expérience a consisté à tester de façon préliminaire les modèles développés. En effet, il est important de s'assurer que les valeurs prédites par les différents modèles soient des valeurs raisonnables (intervalles de valeurs prescrits dans la référence) et surtout que les prédictions puissent être faites avec les valeurs réelles de l'environnement. Étant donné qu'une bonne partie des manœuvres est statique, on n'observe pas beaucoup de variation de l'environnement. Mais les valeurs prédites sont des valeurs acceptables d'après la référence (annexe B). La prochaine étape serait d'intégrer les modèles appris au modèle de référence.

5.3.3 Expérience 2 : test complet du Runner

Cette deuxième expérience implique l'intégration du modèle appris avec le modèle de référence qui est l'ontologie. L'objectif ici est de pouvoir effectuer un décollage complet en utilisant les modèles appris comme pilote. Ceux-ci tenteront de prédire à chaque fois les valeurs de pédales afin de manœuvrer tout le décollage. D'un autre côté, l'ontologie se charge de gérer tous les segments statiques du décollage ou toutes les opérations qui ne nécessitent pas les manœuvres particulières du pilote et qui sont en général des actions standard ou prédéfinies. Pour ce faire , nous allons :

Créer une classe pour les différents modèles

Pour ça, nous utilisons le mot-clé *class*, suivi du nom de la classe : `GabModel`. Cette classe comprend :

— Initialisation de la classe en chargeant les différents modèles (5.4). Il s'agit du constructeur par défaut de la classe.

```
class Gab_Model:
    def __init__(self):
        self.model5 = load_model('LearnedModel/"model5_new.h5')
        self.model6 = load_model('LearnedModel/"model6_new.h5')
        self.model7_1 = load_model('LearnedModel/"model7_1_new.h5')
        self.model7_2 = load_model('LearnedModel/"model7_2_new.h5')
        self.model7_3 = load_model('LearnedModel/"model7_3_new.h5')

        print(" Models Initialized")
```

FIGURE 5.4 Initialisation de la classe

— Définir la fonction *Run* qui prend en entrée l'état de l'environnement et prédit les commandes de pilotes selon les différents segments(5.5).

```
def run(environement):
    sidestick_x = sidestick_y = pedals = 0
    #segment 5
    if (float(environement['AirspeedIndicated']) <= 80) and (float(environement['AirspeedIndicated']) >= 100) :

        tab = np.array([float(environement['CrossWind']), float(environement['TailWind']), float(environement['AirspeedIndicated'])])
        tab = tab.reshape(-1, 1, 3)
        # prediction of pilot action
        sidestick_y = self.model5.predict(tab)
        #Conversion
        sidestick_y = (sidestick_y - 50) / 50

        sidestick_x = float(environement['SideStickPositionX'])
        pedals = float(environement['PedestalRudderTrimButtonPosition'])
    #segment6
    elif (float(environement['AirspeedIndicated']) - 2 == float(environement['VspeedVr'])) and (float(environement ['Altitude'])<= 5):

        tab = np.array([float(environement['PitchAngle']), float(environement['PitchRate']), float(environement['AirspeedIndicated']), float(environement['VspeedVr'])])
        tab = tab.reshape(-1, 1, 4)
        # prediction of pilot action
        sidestick_y = self.model6.predict(tab)
        # conversion
        sidestick_y = (sidestick_y - 50) / 50

        sidestick_x = float(environement['SideStickPositionX'])
        pedals = float(environement['PedestalRudderTrimButtonPosition'])
    #segment 7
    else :
        tab = np.array([float(environement['PitchAngle']), float(environement['PitchRate']), float(environement['BankAngle']), float(environement['AirspeedIndicated']), float(environe
        tab = tab.reshape(-1, 1, 6)

        # prediction of pilot action
        sidestick_y = predict(tab, model7_1)
        pedals = predict(tab, model7_2)
        sidestick_x = predict(tab, model7_3)
        # conversion
        sidestick_y = (sidestick_y3 - 50) / 50
        pedals = pedals / 100
        sidestick_x= (sidestick_x3 - 50) / 50
    return [sidestick_x,sidestick_y,pedals]
```

FIGURE 5.5 Fonction de prédiction de comportement

Simulation de décollage

— Effectuer la procédure de démarrage à froid dans le cockpit (voir annexe F). Cette procédure consiste de façon générale à la mise en marche de l'avion.
— La connexion à l'API qui permet la connexion avec X-Plane (figure 5.6). Le client python a été configuré avec un référentiel PyPi, qui permet aux utilisateurs d'importer cette API dans leur code python avec une relative facilité. Tout ce qu'il faut, c'est effectuer une installation *pip* du paquet : `pip install XPlaneApi`. Ensuite, l'API peut être importée avec les éléments suivants `from XPlaneApi; import XPlaneClient`.

```
api_client = XPlaneClient("Runner")
#Call the connect command
if not api_client.connect():
    print("Connection client 1 failed")
    exit()
```

FIGURE 5.6 Connexion du runner

— Charger et analyser des ontologies dans un modèle hiérarchique (figure 5.7). Comme dit précédemment, l'ontologie est utilisée comme référence pour les tâches et actions de pilotage. Celui-ci nous permet de retrouver toutes les taches, les actions et les contraintes des différentes taches. Ces contraintes sont utilisées de manière récursive pour valider l'évolution de l'environnement et les différentes actions.
— *Mapping* des paramètres. Afin de valider avec succès les tâches ontologiques, ces contraintes doivent être comparées en utilisant les mêmes unités et étiquettes. Le simulateur de CAE, et X-Plane ont chacun leurs propres références d'étiquettes pour des systèmes d'aéronef particuliers. Ces systèmes doivent être correctement mappés à l'ontologie de l'avion au moyen d'un dictionnaire (annexe E).
— Lancer une boucle de simulation en temps réel et mettre à jour l'environ-

```
ontology_rebuild = False

sim_runner = runner.Runner(hWnd, sim_yaml_path, api_client, 10, 0)
if ontology_rebuild == True:
    print("Rebuilding Ontology...")
    sim_runner.build_ontology_hierarchy(ONT_DIR, ONTOLOGY_NAMES)  # Run this to load ontology from owl files
    sim_runner.save_ontology_hierarchy("ontology_obj.pkl")
else:
    sim_runner.build_ontology_hierarchy(ONT_DIR, ONTOLOGY_NAMES, "ontology_obj.pkl") # Run this to load from pickle

sim_runner.modify_ontology_class()
sim_runner.generate_sequences_from_ontology()
```

FIGURE 5.7 Chargement des ontologies

nement de X-Plane (figure 5.8). À chaque cycle, une différenciation est effectuée entre la ligne de journal actuelle et la précédente. Les différences entre les deux pas de temps sont ensuite mises en évidence et renvoyées à la simulation. Le modèle reçoit à chaque pas de temps l'état de l'environnement et renvoi les différentes manoeuvres pilote à effectuer (figure 5.9).

```
def run_simulation_loop(self):
    start_time_sec = time.time_ns() * Runner.nanos_to_sec
    next_iter_time = start_time_sec + self.runner_frequency
    self.runner_stats_dict["runner_heartbeat"] = 0
    while(True):
        # Start runner loop
        if self.runner_stats_dict["runner_heartbeat"] % 100 == 0:
            Runner.display_runner_stats_dict(self)

        current_time = time.time_ns() * Runner.nanos_to_sec
        self.runner_stats_dict["runner_heartbeat"] += 1
        self.runner_stats_dict["running_time"] = next_iter_time - start_time_sec

        # ============== Start Runner Work ==============
        Runner.update_environment(self) # Update the environment
        Runner.perform_ontology_tasks(self) # Perform task actions
        Runner.audit_mode(self) # Audit the cockpit and perform deviation assessments

        if self.completion_flag == True:
            print("Takeoff Complete! Comencing Runner Termination...")
            return
```

FIGURE 5.8 Boucle de simulation

```
self.actions_task_mapping_dict = {} # Used to map type of actions to their tasks for quicker searche
self.constraints_task_mapping_dict = {} # Used to map type of constrain to their tasks

self.learned_model_output = []
self.learned_model_output = gab_model.run(self.environment)

self.completion_flag = False

Runner.extract_yaml_objs(self, path_to_yaml) # For environment
```

FIGURE 5.9 Intégration de modèles appris

5.3.4 Résultats

Notre environnement de simulation (figure 5.10) comprend en plus du simulateur X-Plane, des manettes de contrôles d'actions de pilotage, etc.

Plusieurs tests de décollages sont effectués en changeant à chaque fois les conditions de ceux-ci. L'avion décolle normalement. Il est maintenant important d'avoir des analyses plus poussées à ce propos.

5.4 Conclusion

Nous avons dans ce chapitre crée un cadre de validation des modèles appris développés qui est l'exécution de ceux-ci. Nous avons effectué 2 expériences. La première expérience qui évalue de façon individuelle les modèles développés et la seconde qui intègre les modèles dans une boucle de simulation. Afin d'évaluer l'efficacité des modèles proposés, nous avons tenté des décollages normaux (aucune panne) avec des temps orageux et des temps calmes. Les modèles stabilisent parfaitement l'avion et agissent bien. Une procédure de validation et d'approbation par les experts est en cours.

FIGURE 5.10 Environement de simulation

CHAPITRE VI

CONCLUSION

Dans cette recherche, nous avons tout d'abord analysé des comportements de pilotage afin de mettre en évidence des profils comportementaux de pilotage (gain du pilote). Ensuite, nous avons développé un modèle d'apprentissage des actions de pilotage (plus précisément, les actions effectuées par les pilotes au décollage d'un Airbus 320) à l'aide de réseaux de neurones récurrents. Enfin, nous avons exécuté le modèle appris des actions de pilotage dans le simulateur X-Plane (ce qui constitue un cadre de validation des modèles développés).

À partir des données brutes de décollages d'un A320 (Airbus), nous avons grâce aux différentes architectures que propose l'apprentissage automatique dont les algorithmes de regroupement, tenter d'extraire efficacement des informations cachées importantes pour la discrimination et la modélisation des profils comportementaux de pilotage. Une première expérience a consisté à réduire la dimension des données en utilisant des LSTM auto-encodeurs et d'y appliquer un algorithme standard de regroupement qui est les K-moyennes. Il en ressort 2 limites : premièrement la perte d'informations due à la réduction de dimension et enfin la difficulté à ressortir un seul profil de pilotage sur un décollage complet. La seconde expérience, à la différence de la première, a consisté à extraire les profils comportementaux de pilotage basé sur les segments de décollage. Cette approche

a été plus intéressante et a permis de ressortir 2 catégories de comportements de pilotage.

Le second volet de cette recherche a consisté à développer un modèle appris des comportements de pilotage capable de prédire ceux-ci. Une approche de pilotage est proposée pour apprendre le comportement de pilotage à partir des données (réseaux de neurones à mémoires à long court terme). Le modèle est développé sur la base de segments de décollage identifiés par des experts du domaine. Les expériences ont montré la capacité du modèle appris à capter les tâches de haut niveau. Les prédictions faites passent l'évaluation préliminaire dans ce sens ou les valeurs prédites se situent dans les intervalles de valeurs de référence spécifiés par les experts.

Enfin, dans l'optique de créer un cadre de validation des modèles mis en place, nous avons développé un pilote synthétique capable d'exécuter les actions prédites par le modèle appris dans le simulateur X-Plane. L'exécution de ce modèle fait appel à un modèle de référence (ensemble d'ontologies) qui est un dispositif capable d'identifier l'action qu'un pilote doit accomplir selon ce que prévoit la théorie de pilotage. Les différentes expériences effectuées ont montré que le système proposé imite le comportement humain dans l'exécution des tâches de pilotages au décollage, ce qui est l'objectif visé.

6.1 Limitations

L'ensemble des objectifs visés dans ce mémoire, notamment la catégorisation des comportements de pilotage, le développement d'un modèle appris capable de prédire les actions de pilotage et enfin, la création d'un cadre de validation des modèles développés ont été atteints. Toutefois, il est important de mentionner quelques limites de cette recherche :

— la quantité de données :Le modèle aurait peut-être pu être amélioré s'il y avait plus de données.
— correspondance des données : les données utilisées pour entraîner nos modèles proviennent du simulateur de CAE. Pour des soucis de généralisation de la solution, le cadre de validation a été le simulateur X-Plane. Ainsi, la correspondance non parfaite des données de ces différents simulateurs peut avoir eu un impact sur les performances obtenues.

6.2 Travaux futurs

Plusieurs ajouts et améliorations sont envisagés notamment :

— Effectuer de multiples décollages dans un environnement de simulations préétablies afin de collecter les données pour enrichir les différents modèles développés.
— Développer des modèles appris spécifiques à différentes classes de comportements/profils des pilotes.
— Étendre l'apprentissage des comportements de pilotage aux autres phases de vol (croisière, atterrissage, etc.), avec un accent sur les phases les plus critiques.
— Étendre la prédiction de comportement de pilotage sur d'autres types d'avion.
— Utiliser le modèle appris pour enrichir la mémoire procédurale d'un agent cognitif de type ACT-R jouant le rôle d'un pilote synthétique.

Ce travail a mené à deux publications, une qui porte sur la catégorisation des profils comportementaux (Tato *et al.*, 2022), et l'autre qui présente les résultats du modèle appris construit (Tato *et al.*, 2023).

APPENDICE A

RÉDUCTION DE DIMENSION

```
visible = Input(shape=(timesteps,n_features))
e = LSTM(128, activation='relu',return_sequences=True)(visible)
e = LSTM(64, activation='relu', return_sequences=False)(e)
out_e = Dense(90)(e)
bottleneck = RepeatVector(timesteps) (out_e)
d = LSTM(128, activation='relu', return_sequences=True)(bottleneck)
d = LSTM(64, activation='relu', return_sequences=True)(d)
output = TimeDistributed(Dense(n_features))(d)
model = Model(inputs=visible, outputs=output)
ADAM = tensorflow.keras.optimizers.Adam(learning_rate = 0.00001,

beta_1=0.9, beta_2=0.999, epsilon=None,

decay=0.0, amsgrad=False, clipnorm=1.0 , clipvalue=0.5)

model.compile(optimizer= ADAM , loss='mse', metrics=['accuracy'])
```

APPENDICE B

TABLEAU DE RÉFÉRENCE

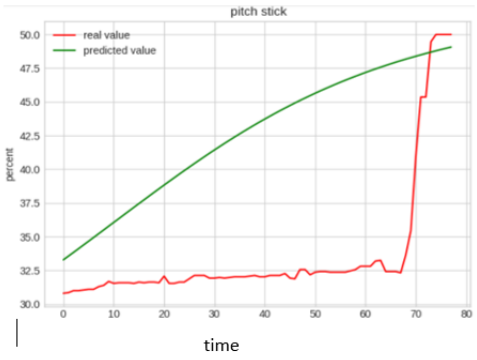
URI	Description	Range	Unit	Comments
ControlLateral - Position (roll stick)	Stick(Roll)	0 to 100	percent	Full left = 0 (bank left), Full right = 100 (bank right), Neutral (center released by pilot)=50.
ControlLongitudinal -Position (pitch stick)	Stick(Pitch)	0 to 100	percent	Full Forward = 0, full aft = 100, Neutral (center released by pilot) = 50
ControlDirectional - Position (Rudder Pedals)	Rudder pedals	0 to 100	percent	Right Fwd = 0, Left Fwd = 100, Neutral (center released by pilot) = 50
StandardAircraft /BankAngle	Bank Angle	-45 to 45	degrees	bank left = negative, bank right = positive
StandardAircraft /PitchAngle	Pitch Angle	-13 to 25	degrees	pitch up = positive, pitch down=negative

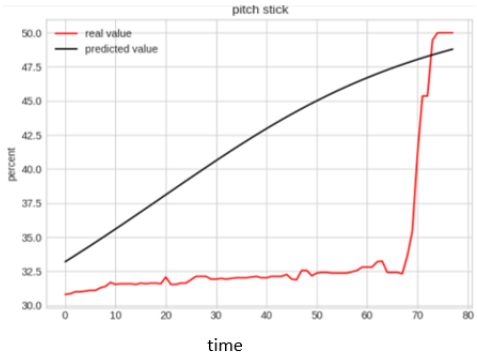
APPENDICE C

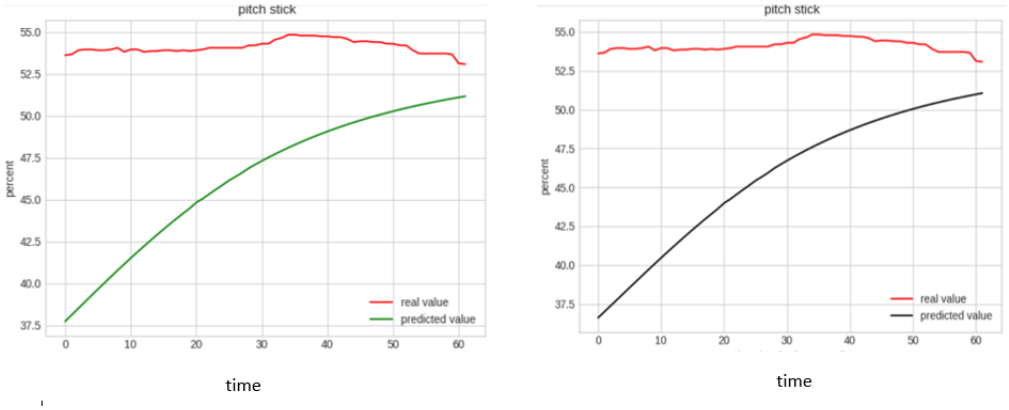
AUTRES EXEMPLES DE PRÉDICTION

C.1 Model 5

Les figures présentées ci-dessous montrent d'autres exemples de prédiction pour le segment 5. La courbe en rouge représente le *pitch stick* réel du pilote, les courbes en noir et vert représentent les valeurs prédites de celui-ci selon les 2 expériences(1 et 2).

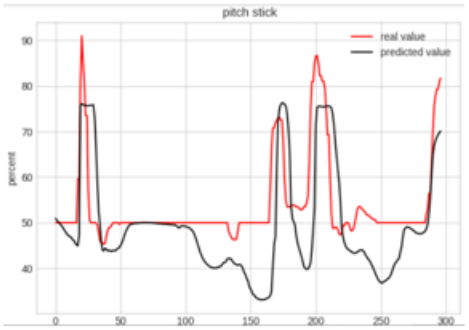


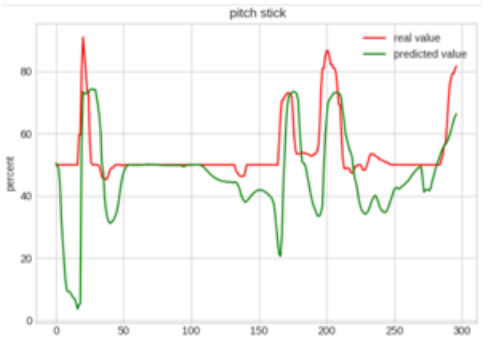


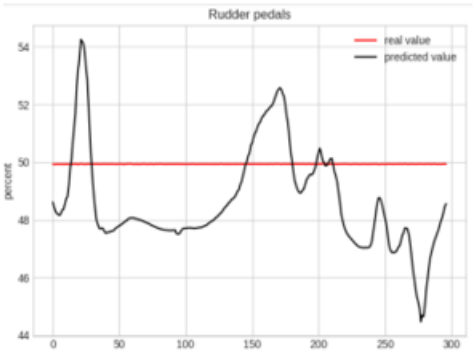


C.2 Model 7

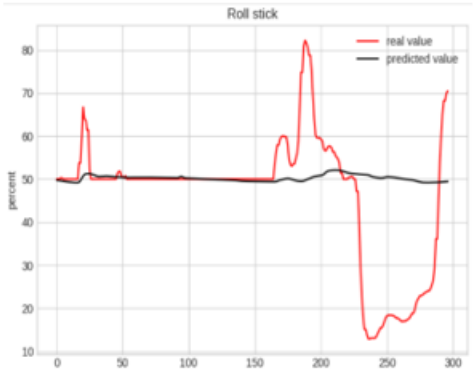
Les figures ci-dessous présentent deux exemples de prédiction pour le segment 7 selon différentes expériences.Les courbes en rouge représentent les valeurs réelles du pilote et celles en noir et en vert les valeurs prédites.

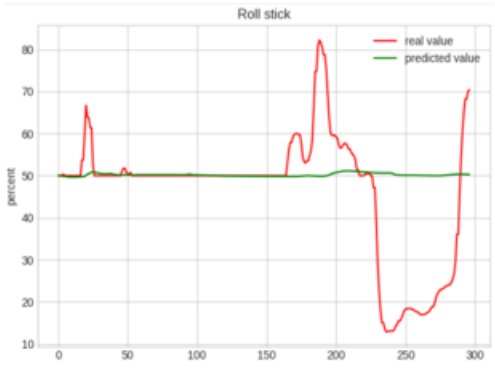

pitch stick
real value
predicted value
percent


pitch stick
real value
predicted value
percent


Rudder pedals
real value
predicted value
percent


Rudder pedals
real value
predicted value
percent

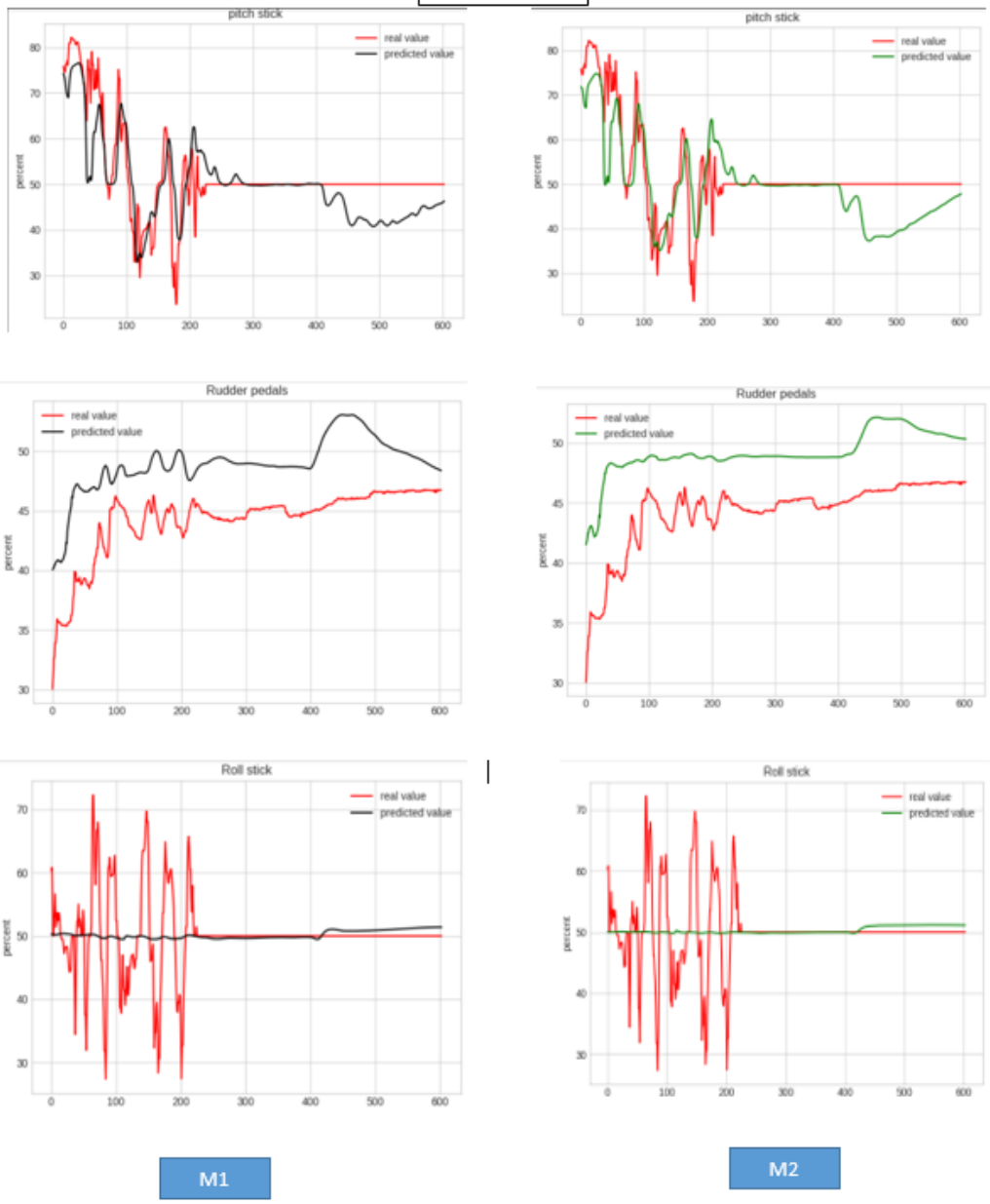

Roll stick
real value
predicted value
percent


Roll stick
real value
predicted value
percent

M1

M2

Pilote 2

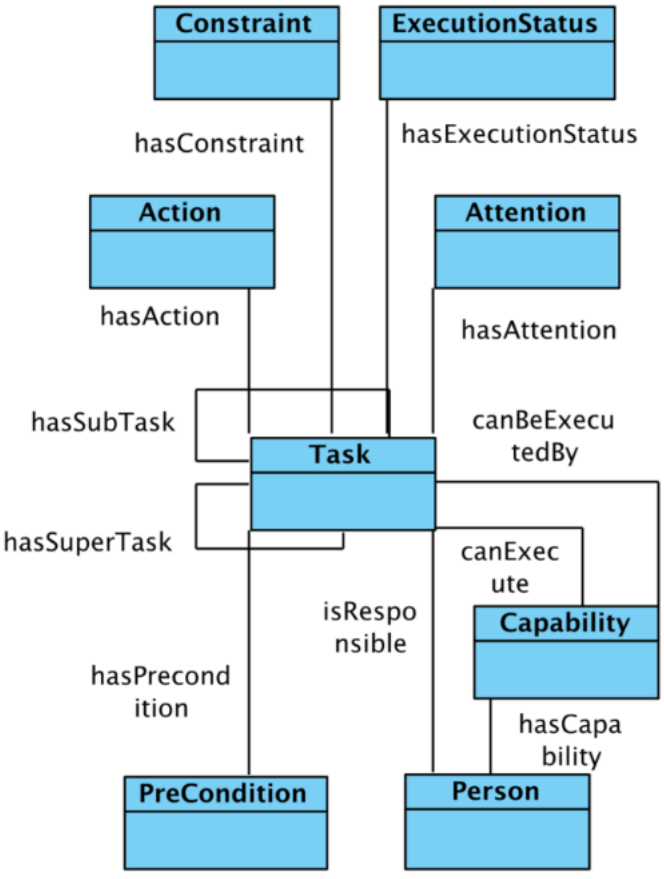


M1

M2

APPENDICE D

ONTOLOGIE DE TACHE



APPENDICE E

FICHIER DE MAPPING

```
:
tag: "sim/time/zulu_time_sec"
frequency: 10
description: "EMPTY"
raft_Description:
tag: "sim/aircraft/view/acf_descrip[0][40]"
frequency: 10
description: "EMPTY"
StickPositionY:
tag: "a320/Aircraft/Cockpit/Panel/SidestickLonL"
frequency: 10
description: "EMPTY"

StickPositionX:
tag: "a320/Aircraft/Cockpit/Panel/SidestickLatL"
frequency: 10
description: "EMPTY"

ne1FireSwitch:
```

```
20  tag: "a320/Overhead/FireEngine1"
21  frequency: 1
22  description: ""
23  ne2FireSwitch:
24  tag: "a320/Overhead/FireEngine2"
25  frequency: 1
26  description: ""
27  ne2MasterSwitchPosition:
28  tag: "a320/Pedestal/EngineMaster2_switch+"
29  frequency: 1
30  description: ""
31  ne1MasterSwitchPosition:
32  tag: "a320/Pedestal/EngineMaster1_switch+"
33  frequency: 1
34  description: ""
35
36  tag: "sim/aircraft/engine/acf_EGT_is_C"
37  frequency: 1
38  description: ""
39  ThrustMode:
40  tag: "a320/Panel/FCU_AutoThrust"
41  frequency: 1
42  description: ""
43  PilotMasterSwitch:
44  tag: "a320/Panel/FCU_AutoPilot1"
45  frequency: 1
46  description: ""
47  t2Switch:
```

```
48  tag: "a320/Overhead/FireEngine1_Agent2"
49  frequency: 1
50  description: ""
51  t1Switch:
52  tag: "a320/Overhead/FireEngine1_Agent1"
53  frequency: 1
54  description: ""
55  BrakePosition:
56  tag: "sim/flightmodel/controls/parkbrake"
57  frequency: 1
58  description: ""
59  dBrakeLeverPosition:
60  tag: "a320/Panel/BrakeAuto1"
61  frequency: 1
62  description: ""
63  eLeft:
64  tag: "sim/cockpit2/controls/left_brake_ratio"
65  frequency: 1
66  description: ""
67  eRight:
68  tag: "sim/cockpit2/controls/left_brake_ratio"
69  frequency: 1
70  description: ""
71  ingGearLeverPosition:
72  tag: "sim/cockpit2/controls/gear_handle_down"
73  frequency: 1
74  description: ""
75  d:
```

```
76  tag: "sim/flightmodel/position/indicated_airspeed"
77  frequency: 10
78  description: ""
79  ndication:
80  tag: "sim/cockpit/autopilot/vertical_velocity"
81  frequency: 10
82  description: ""
83  oAltitude:
84  tag: "sim/cockpit2/gauges/indicators/radio_altimeter_height_ft_pilot"
85  frequency: 10
86  description: ""
87  Mode:
88  tag: "sim/cockpit2/EFIS/EFIS_tcas_on"
89  frequency: 10
90  description: ""
91  Direction:
92  tag: "sim/flightmodel2/controls/rudder_trim"
93  frequency: 10
94  description: "Rudder trim values"
95  erSwitchWarningSwitchPress:
96  tag: "sim/cockpit/warnings/annunciator_test_pressed"
97  frequency: 10
98  description: ""
99  oTransmitting:
100 tag: "sim/atc/user_aircraft_transmitting"
101 frequency: 10
102 description: ""
103 Wind:
```

```
104 tag: "sim/weather/wind_speed_kt[0]"
105 frequency: 10
106 description: ""
107 sWind:
108 tag: "sim/weather/wind_speed_kt[1]"
109 frequency: 10
110 description: ""
111 ne1Failure:
112 tag: "sim/operation/failures/rel_engfai1"
113 frequency: 10
114 description: ""
115 ne2Failure:
116 tag: "sim/operation/failures/rel_engfai2"
117 frequency: 10
118 description: ""
119 ingGearPosition:
120 tag: "sim/aircraft/prop/acf_prop_gear_rat"
121 frequency: 10
122 description: ""
123 peed:
124 tag: "sim/flightmodel/position/indicated_airspeed"
125 frequency: 10
126 description: ""
127 ThrustLever_write:
128 tag: "sim/cockpit2/engine/actuators/throttle_ratio[0]"
129 frequency: 10
130 description: "Engine 1 Thrust value ratio between 0-1"
131 tThrustLever_write:
```

```
132 tag: "sim/cockpit2/engine/actuators/throttle_ratio[1]"
133 frequency: 10
134 description: "Engine 2 Thrust value ratio between 0-1"
135 ThrustLever_Read:
136 tag: "a320/Aircraft/Cockpit/Pedestal/EngineLever1"
137 frequency: 10
138 description: "Engine 2 Thrust value ratio between 0-1"
139 tThrustLever_Read:
140 tag: "a320/Aircraft/Cockpit/Pedestal/EngineLever2"
141 frequency: 10
142 description: "Engine 2 Thrust value ratio between 0-1"
143 irection:
144 tag: "sim/cockpit/gyros/psi_ind_degm3"
145 frequency: 10
146 description: ""
147 tude:
148 tag: "sim/flightmodel/misc/h_ind"
149 frequency: 10
150 description: ""
151 Lever:
152 tag: "sim/multiplayer/controls/flap_request"
153 frequency: 10
154 description: "FLAP values"
155 Alert:
156 tag: "sim/cockpit2/tcas/indicators/tcas_alert"
157 frequency: 10
158 description: "TCAS Alert"
159 h:
```

```
160 tag: "sim/cockpit/gyros/the_vac_ind_deg"
161 frequency: 10
162 description: ""
163 ShearWarning:
164 tag: "sim/cockpit2/annunciators/windshear_warning"
165 frequency: 10
166 description: "Windshear warning values"
167
```

APPENDICE F

PROCÉDURE DE MISE EN MARCHE POUR LE DÉCOLLAGE

1. Aller sur la tablette pour gérer le fioul et le chargement ;
2. Bat 1 et Bat 2 à ON ;
3. Exit power ;
4. Aligner les addir (ADR1, ADR2, ADR3) ;
5. Mettre les DUME au milieu ;
6. AWLT juste à côté du DUME pour tester toutes les lumières du cookpit ;
7. Mettre les EMER EXIT LT à ON ;
8. Mettre les NO SMOKING à côté de EMER EXIT LT sur ON ;
9. Les SEATBELTS (proche de NO SMOKING) on les mets sur AUTO ;
10. On peut tester le système antiincendie de l'APU (cliquer sur tester au centre) ;
11. Allumer les pompes : L TK PUMPS 1 et 2, PUMP 1, PUMP2 ,R TK PUMPS 1 et 2 ;
12. Mettre le NAV LOGO à 1 ;
13. Mettre le MASTER SW à ON ;
14. Après quelque temps, cliquez sur START juste en bas ;
15. Mettre les PROBE /WINDOW HEAT à ON ;

16. Mettre API BLEED à ON : le PACK 1 et PACK 2 vont s'activer ;
17. Configuration du FMS et on va finir le chargement ;
18. Enfin arrêter le EXIT PWR ;
19. Mettre les SEATBELTS sur ON.

APPENDICE G

PLUGIN ULTIME AIRBUS A320

Le *plugin Ultime Airbus A320*[1] doit être téléchargé et installé pour pouvoir utiliser l'avion A320 dans X-Plane. Les étapes d'installation sont les suivantes :

- télécharger le fichier A320Plugin.zip depuis la boutique.
- Décompressez le fichier et localisez le dossier "Aircraft". Copiez le "*FlightFactor A320 Ultimate*".
- ouvrir le dossier principal de X-Plane 11 contenant les fichiers d'installation et localiser le dossier "Aircraft" (celui qui se trouve dans le dossier principal).
- coller le dossier « *FlightFactor A320 Ultimate* » dans le dossier « *Aircraft* » du dossier principal de X-Plane 11.
- retourner à nouveau dans le dossier décompressé, ouvrir le dossier "Ressources" et identifier le dossier nommé « CEF ».
- ouvrir à nouveau le dossier principal de X-Plane 11 et localiser le dossier "Ressources", puis accéder au dossier "plugins" et coller le dossier "CEF" précédemment copié à l'intérieur.
- charger l'avion et allumer l'unité d'alimentation au sol sur votre EFB (moniteur situé au-dessus du manche latéral du siège gauche).

---

1. https ://store.x-plane.org/A320-Ultimate-XP11_p_688.html

— allumer l'alimentation externe. Ce bouton indiquera le mot "DISPONIBLE" avant qu'il ne soit cliqué puis "ON" une fois qu'il a été cliqué.

— tous les systèmes doivent maintenant être démarrés et entrer le code de série de l'achat dans le MCDU.

RÉFÉRENCES

Aghabozorgi, S., Shirkhorshidi, A. S. et Wah, T. Y. (2015). Time-series clustering–a decade review. *Information systems*, *53*, 16–38.

Amidon, A. (2020). How to apply k-means clustering to time series data. *Towards Data Science*.

Baker, K. (2005). Singular value decomposition tutorial. *The Ohio State University*, *24*.

Baomar, H. et Bentley, P. J. (2016). An intelligent autopilot system that learns piloting skills from human pilots by imitation. Dans *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1023–1031. IEEE.

Bauschat, J.-M., Leissling, D. et Gestwa, M. (2004). A score monitoring system to support evaluation pilots in flight. Dans *AIAA Modeling and Simulation Technologies Conference and Exhibit*, p. 5444.

Belyavin, A., Woodward, A., Nguyen, D., Robel, G. et Woolworth, J. (2005). Development of a novel model of pilot control behavior in balked landings. Dans *AIAA Modeling and Simulation Technologies Conference and Exhibit*, p. 5878.

Bienkowski, M., Feng, M. et Means, B. (2012). Enhancing teaching and learning through educational data mining and learning analytics : An issue brief. *Office of Educational Technology, US Department of Education*.

Bodin, C. (2020). Automatic flight maneuver identification using machine learning methods.

Bontempi, G., Ben Taieb, S. et Le Borgne, Y.-A. (2013). Machine learning strategies for time series forecasting. *Business Intelligence : Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2*, 62–77.

Brunt, K., Neumann, T. et Smith, B. (2019). Assessment of icesat-2 ice sheet surface heights, based on comparisons over the interior of the antarctic ice sheet. *Geophysical Research Letters*, *46*(22), 13072–13078.

Chan, A. (2022). Gpt-3 and instructgpt : technological dystopianism, utopianism, and "contextual" perspectives in ai ethics and industry. *AI and Ethics*, 1–12.

Courtemanche, M.-A., Tato, A. et Nkambou, R. (2022). Ontological reference model for piloting procedures. Dans *Intelligent Tutoring Systems : 18th International Conference, ITS 2022, Bucharest, Romania, June 29–July 1, 2022, Proceedings*, 95–104. Springer.

Cregger, J., Mahavier, K., Holub, A., Machek, E., Crayton, T., Patel, R. et Suder, S. (2022). *Automation in Our Parks : Automated Shuttle Pilots at Yellowstone National Park and Wright Brothers National Memorial*. Rapport technique.

Gilbreath, G. P. (2001). Prediction of pilot induced oscillations (pio) due to activator rate limiting using the open-loop onset point (olop) criterion.

Gray, W. (2009). Handling qualities evaluation at the usaf test pilot school. Dans *AIAA Atmospheric Flight Mechanics Conference*, p. 6317.

Gray, W. R. (2007). A boundary avoidance tracking flight test technique for performance and workload assessment. Dans *Proceedings of the 38th Symposium of Society of Experimental Test Pilots, San Diego*.

Hansen, J., Murray, J. et Campos, N. (2004). The nasa dryden aar project : a flight test approach to an aerial refueling system. Dans *AIAA atmospheric flight mechanics conference and exhibit*, p. 4939.

Heaton, J. (2008). *Introduction to neural networks with Java*. Heaton Research, Inc.

Hess, R. A. (2006). Simplified approach for modelling pilot pursuit control behaviour in multi-loop flight control tasks. *Proceedings of the Institution of Mechanical Engineers, Part G : Journal of Aerospace Engineering*, *220*(2), 85–102.

Jain, A. K. (2010). Data clustering : 50 years beyond k-means. *Pattern recognition letters*, *31*(8), 651–666.

Jean-Baptiste, L. (2021). Ontologies with python. *Apress, Berkeley, CA*.

Kayte, S., Mundada, M. et Gujrathi, J. (2015). Hidden markov model based speech synthesis : A review. *International Journal of Computer Applications*, *130*(3), 35–39.

Khan, K., Rehman, S. U., Aziz, K., Fong, S. et Sarasvady, S. (2014). Dbscan : Past, present and future. Dans *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, 232–238. IEEE.

Klyde, D., Brenner, M. et Thompson, P. (2001). Wavelet-based time-varying human operator models. Dans *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, p. 4009.

LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lee, S., Choi, H. S. et Seong, K.-J. (2014). Automatic flight control system development for optionally piloted vehicle. *Journal of the Korean Society for Aeronautical & Space Sciences*, *42*(11), 968–973.

Li, L., Das, S., John Hansman, R., Palacios, R. et Srivastava, A. N. (2015). Analysis of flight data using clustering techniques for detecting abnormal operations. *Journal of Aerospace information systems*, *12*(9), 587–598.

Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition*, *38*(11), 1857–1874.

Lin, J., Keogh, E., Lonardi, S. et Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. Dans *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2–11.

Lowe, C. et How, J. P. (2015). Learning and predicting pilot behavior in uncontrolled airspace. In *AIAA Infotech@ Aerospace* p. 1199.

Mangortey, E., Monteiro, D., Ackley, J., Gao, Z., Puranik, T. G., Kirby, M., Pinon-Fischer, O. J. et Mavris, D. N. (2020). Application of machine learning techniques to parameter selection for flight risk identification. Dans *AIAA Scitech 2020 Forum*, p. 1850.

Mathews, J. A pilot recommender system using k-means clustering jithin mathews, sandeep kumar, dr. ch. sobhan babu1 prajwal eachempati department of computer science department of it systems iit hyderabad iim rohtak.

Matsumoto, T. T., Vismari, L. F., Gimenes, R. A. V., De Almeida, J. R. et Camargo, J. B. (2015). A learning-based autonomous control system approach for collision avoidance within an unmanned aircraft. Dans *2015 IEEE International Conference on Dependable Systems and Networks*

Workshops, 96–103. IEEE.

McRuer, D. T. (1995). *Pilot-induced oscillations and human dynamic behavior*. Rapport technique.

Mitchell, D. G., Aponso, B. L. et Hoh, R. H. (1990). *Minimum Flying Qualities. Volume 1. Piloted Simulation Evaluation of Multiple Axis Flying Qualities*. Rapport technique, SYSTEMS TECHNOLOGY INC HAWTHORNE CA.

Mitchell, D. G. et Klyde, D. H. (2019). This is pilot gain. Dans *AIAA Scitech 2019 Forum*, p. 0562.

Moormann, D. (2000). Physical model-building leading to automatic code generation. in : Flight control design-best practices.

Neal, T. P. et Smith, R. E. (1971). A flying qualities criterion for the design of fighter flight-control systems. *Journal of Aircraft*, *8*(10), 803–809.

Niewind, I. (2011). What the hell is pilot gain ?

Niewind, I. (2012). Pilot gain and the workload buildup flight test technique : about the influence of natural pilot gain on the achievable pilot gain range.

Niewind, I. (2013). A new approach for the validation of potential pilot gain measures. Dans *EuroGNC 2013, 2nd CEAS Specialist Conference on Guidance, Navigation and Control*.

Nittala, S. K., Elkin, C. P., Kiker, J. M., Meyer, R., Curro, J., Reiter, A. K., Xu, K. S. et Devabhaktuni, V. K. (2018). Pilot skill level and workload prediction for sliding-scale autonomy. Dans *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1166–1173. IEEE.

Palit, A. K. et Popovic, D. (2006). *Computational intelligence in time series forecasting : theory and engineering applications*. Springer Science & Business Media.

Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y. et Sykes, C. (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, *173*(7-8), 789–816.

Ross, G. A. (2022). *Extended reality (xR) flight simulators as an adjunct to traditional flight training methods : a thesis presented in partial fulfilment of the requirements for the degree of Master of Aviation at Massey University, New Zealand*. (Thèse de doctorat). Massey University.

Sagheer, A. et Kotb, M. (2019). Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems. *Scientific reports*, *9*(1), 1–16.

Sheridan, K., Puranik, T. G., Mangortey, E., Pinon-Fischer, O. J., Kirby, M. et Mavris, D. N. (2020). An application of dbscan clustering for flight anomaly detection during the approach phase. Dans *AIAA Scitech 2020 Forum*, p. 1851.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D : Nonlinear Phenomena*, *404*, 132306.

Simm, A. (2012). Technical and psychological aspects of pilot gain.

Sinaga, K. P. et Yang, M.-S. (2020). Unsupervised k-means clustering algorithm. *IEEE access*, *8*, 80716–80727.

Srivastava, N., Mansimov, E. et Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. Dans *International conference on machine learning*, 843–852. PMLR.

Tato, A., Nkambou, R. et Nana Tato, G. J. (2022). Towards adaptive coaching in piloting tasks : Learning pilots' behavioral profiles from flight data. Dans *Intelligent Tutoring Systems : 18th International Conference, ITS 2022, Bucharest, Romania, June 29–July 1, 2022, Proceedings*, 105–114. Springer.

Tato, A., Nkambou, R. et Nana Tato, G. J. (2023). Automatic learning of piloting behavior from flight data. Dans *Augmented Intelligence and Intelligent Tutoring Systems. 19th International Conference, ITS 2023, Corfu, Greece, June 2–5, 2023, Proceedings*, 541–552. Springer.

Tchio, G. C. T., Courtemanche, M.-A., Tato, A. A. N., Nkambou, R. et Psyché, V. (2023). Integrating an ontological reference model of piloting procedures in act-r cognitive architecture to simulate piloting tasks. Dans *International Conference on Intelligent Tutoring Systems*, 183–194. Springer.

Van der Weerd, R. (2000). *Pilot-Induced Oscillation Suppression Methods and their Effects on Large Transport Aircraft Handling Qualities*. DUP Science.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. et Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, Q. et Megalooikonomou, V. (2008). A dimensionality reduction

technique for efficient time series similarity analysis. *Information systems*, *33*(1), 115–132.

Yadav, M. et Alam, M. A. (2018). Dynamic time warping (dtw) algorithm in speech : a review. *International Journal of Research in Electronics and Computer Engineering*, *6*(1), 524–528.

Yang, X.-S. (2016). *Engineering mathematics with examples and applications*. Academic Press.