Rapport de stage

Lilian Pattier

Utilisation de réseaux de neurones convolutifs pour l'analyse de données de vols

Formations Ingénieur ENSEM / Maîtrise en Informatique UQAC

Stage du 07/10/2019 au 03/04/2020

Réalisé au sein de la société

Safran Aircraft Engines à Moissy-Cramayel

Sous la direction de :

Armand ALIMARDANI – (Tuteur Safran Aircraft Engines)

Valérie LOUIS-DORR – (Tutrice ENSEM)

Sylvain BOIVIN – (Professeur encadrant UQAC)

SAFRAN AIRCRAFT ENGINES

UQAC UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

ensem ÉNERGIES & MOBILITÉS

REMERCIEMENTS

Je tiens à remercier toute l'équipe du Datalab qui m'a accueilli durant ces 6 mois de stage. Je leur suis très reconnaissant pour leur accueil, leur disponibilité et leur sympathie. Cette expérience fut enrichissante en tout point, et l'environnement de travail et la cohésion au sein de cette équipe y sont pour beaucoup.

Je remercie Aurélie et Armand de m'avoir accordé leur confiance et donné l'opportunité d'effectuer mon stage chez Safran Aircraft Engines pour travailler sur un projet passionnant combinant l'analyse de données et l'aéronautique.

Je remercie les Data Scientists pour l'intérêt porté à mon stage et leurs différents conseils tout au long de cette période. Merci à Mohammed, Raphaël, Marc, Jean, Matthieu, Thomas, Céline, Jean-Luc, Armand, Aurore, et Aurélie.

Je remercie également les doctorants qui m'ont souvent aidé sur les aspects théoriques et pratiques des méthodes entreprises durant le stage. Merci à Florent, Alex et Raphael et bon courage pour vos thèses !

Merci également à Jérôme de m'avoir proposé des axes et pistes de recherche et offert sa vision d'expert sur les travaux entrepris et à entreprendre.

Merci à mes professeurs encadrants, Monsieur Boivin (UQAC) et Madame Louis-Dorr (ENSEM).

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

SIGLES ET ACRONYMES

- **BP** : Best Practice
- **CNN** : Convolutional Neural Network (Réseau de neurones convolutifs)
- **QAR** : Quick Access Recorder
- **AOA** : Angle of Attack
- **SAT** : Saturated Air Temperature
- **MLP** : Multi-Layers Perceptron (Perceptron Multi-couches)

RESUME - ABSTRACT

Ce rapport présente le travail effectué dans le cadre de mon stage ingénieur de fin d'études au sein de la société Safran Aircraft Engines. L'objectif est de tester l'implémentation et l'efficacité de l'apprentissage profond, et plus spécifiquement des réseaux de neurones convolutifs dans le cadre de l'analyse de données de vols. Ces structures se sont révélées particulièrement efficaces dans le traitement d'image. Le but est de transposer les méthodes et outils à l'analyse de séries temporelles dans le domaine de l'aviation, et de confirmer ou d'infirmer la pertinence de ce choix au vu des résultats obtenus.

This report presents the work carried out during my end-of-course internship within Safran Aircraft Engines company. The goal is to test the implementation and the efficiency of deep learning models, and more precisely the use of convolutional neural networks in the context of flight-data analysis. These structures have proven to be particularly efficient in image processing. We aim to transpose the methods and tools to time-series analysis in the field of aviation, and to confirm whether this choice is relevant according to the results achieved.

1 INTRODUCTION

Au sein des compagnies aériennes, la consommation de carburant d'une flotte d'avions représente un enjeu majeur, tant sur le plan économique qu'écologique. C'est dans cette optique que Safran Electronics & Defense et Safran Aircraft Engines ont conjointement développé SFCO2®, un service visant à aider ces dernières à optimiser les consommations en carburant de leur flotte. Cet outil utilise une base de données massive de vols et implémente différents algorithmes permettant la génération de « Best Practice » (BP). En pointant les différents gains potentiels, les compagnies qui par la suite, feraient le choix de suivre ces BP peuvent espérer des économies de carburant allant jusqu'à 5% de la consommation totale.

Ce suivi proposé est notamment possible par l'analyse des données de vol de chaque avion d'une flotte donnée. Ces données sont accessibles grâce à un module implanté à bord des aéronefs et nommé QAR : « Quick Access Recorder ».

L'analyse des données QAR peut se faire de différentes manières avec toujours en vue, l'objectif du service SFCO2. L'une des premières étapes pouvant être envisagée est de segmenter le vol des avions en différentes phases, qui pourront ensuite être indépendamment analysées en vue d'aboutir à un algorithme d'optimisation spécifique pour chacune d'elles. Le QAR est déjà pourvu d'un algorithme de segmentation du vol en phases distinctes : chaque instant d'un vol est étiqueté par un nombre correspondant à une phase définie (décollage, atterrissage, croisière, approche…). Cette segmentation, bien que satisfaisante, gagnerait à être plus précise : chaque phase pourrait en théorie, être redivisée en « sous-phases » permettant d'affiner la précision des algorithmes d'optimisation sous-jacents. Un autre avantage de notre implémentation réside dans la généralisation de la procédure à n'importe quels types de turbomachines tandis que le découpage actuel fourni par les algorithmes du QAR repose sur des critères métiers dépendant du type de moteur.

L'objectif de ce stage est de mettre l'intelligence artificielle à profit de cette problématique, en vue d'obtenir un découpage affiné des données de vols. Les données étant assimilables à des séries temporelles, un axe d'approche possible est l'utilisation de réseaux de neurones convolutifs qui ont d'ores et déjà fait leurs preuves dans le domaine de l'imagerie, mais également dans l'étude de signaux temporels comme c'est le cas ici.

2 PRESENTATION DE L'ENTREPRISE

2.1 LE GROUPE SAFRAN

Né en 2005 de la fusion entre le Groupe Snecma et Sagem, Safran est un groupe international de haute technologie dans les domaines de l'aéronautique, de l'espace et de la défense. Comme nous pouvons le voir ci-contre, en 2018, le groupe emploie plus de 92 000 collaborateurs répartis à travers 30 pays sur tous les continents. C'est en prenant le contrôle de Zodiac Aerospace en 2018 que Safran s'est constitué comme le 3ème acteur aéronautique mondial.

Figure 2.1 - Safran dans le monde (2018)

Les activités du groupe sont très vastes comme le montrent les impressionnants chiffres ci-dessous. Safran se place comme un leader mondial dans de nombreux domaines, et se montre aussi être un groupe très innovant à la pointe de la recherche et du développement.

Figure 2.2 - Quelques chiffres à propos du groupe Safran

Les activités du groupe et ses positions au premier plan mondial ou européen sur ses marchés génèrent un chiffre d'affaire de 21 milliards d'euros (2018). Afin de répondre à l'évolution des marchés et pour rester un leader mondial dans ses domaines d'activité, le groupe s'engage dans des programmes de recherche et développement qui ont représenté en 2018 1,5 milliard d'euros d'investissement.

Ce chiffre d'affaire se répartit entre les cinq principales activités du groupe selon le graphique présenté ci-contre (chiffres de 2018). Les livraisons de moteurs civils et militaires portent la majeure partie de ce chiffre d'affaire.

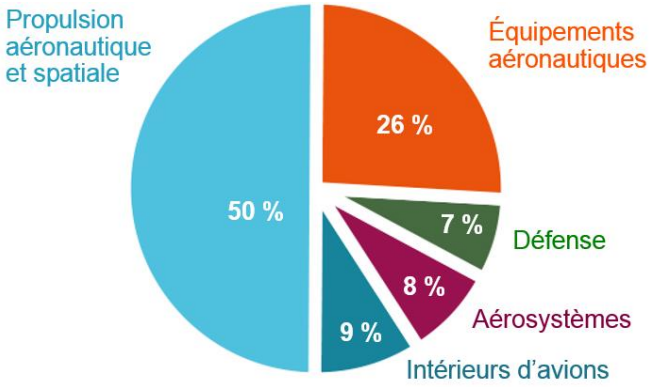

Figure 2.3 - Répartition du chiffre d'affaire de Safran (2018)

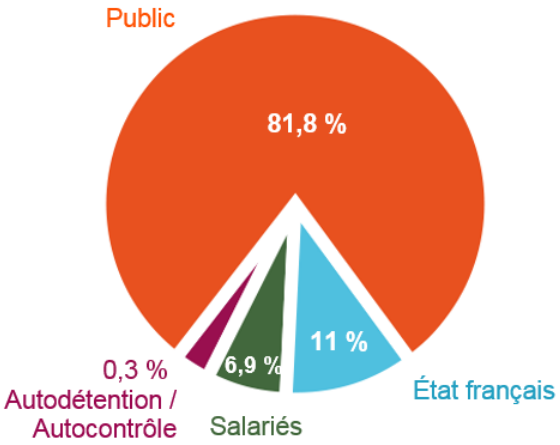

Figure 2.4 - Répartition du capital de Safran au 31/12/2018

Côté finance, le groupe Safran est une société anonyme (SA) dont le capital est réparti tel que présenté sur la Figure 2.4 ci-contre. Observons maintenant les activités du groupe d'un peu plus près.

2.1.1 Propulsion aéronautique et spatiale

La branche propulsion aéronautique et spatiale de Safran regroupe toutes les activités liées aux systèmes de propulsion des avions, hélicoptères et missiles, sur les marchés civil, militaire et spatial. De la conception à la maintenance en passant par les essais et la production, tout est traité chez Safran.

Les principaux marchés de Safran dans ce domaine sont les suivants :

- Les moteurs d'avions civils : n°1 mondial pour les avions court et moyen-courriers, en partenariat 50/50 avec GE,
- Les moteurs d'hélicoptères : n°1 mondial,
- Les moteurs spatiaux : n°1 européen de la propulsion plasmique des satellites

2.1.2 Equipements aéronautiques, aérosystèmes et intérieurs d'avions

Le groupe est également un acteur de premier rang dans les domaines des équipements aéronautiques :

- Les systèmes d'atterrissage : n°1 mondial des trains d'atterrissage,
- Les systèmes de freinage : n°1 mondial des roues et freins carbone (sur avions civils de plus de 100 places),
- Les transmissions de puissance mécanique : n°1 mondial (sur avions civils de plus de 100 places),
- Les systèmes de commande : n°1 mondial des commandes de vol pour hélicoptères,
- Les cabines : n°1 mondial des sièges pour avions commerciaux (classe économique pour avions bi-couloirs).

2.1.3 La Défense

Du côté de la Défense, Safran travaille dans différents segments comme les moteurs d'avions militaires, les drones, l'avionique (ex : systèmes de navigation) et l'optronique. Dans ce dernier segment, les équipements développés sont destinés aux sous-marins, aux navires de surface, aux véhicules de combat ou encore aux plateformes aéroportées.

2.2 SAFRAN AIRCRAFT ENGINES

Safran Aircraft Engines (anciennement Snecma avant mai 2016) est une société du groupe Safran qui conçoit, développe, produit, essaye et commercialise, seule ou en coopération, des moteurs pour avions civils et militaires. Elle propose également aux compagnies aériennes, aux forces armées et aux opérateurs d'avions une gamme complète de services pour leurs moteurs aéronautiques. Avec un chiffre d'affaire de 8,8 milliards d'euros en 2017, elle emploie plus de 16 700 collaborateurs sur 35 sites.

Snecma (Société Nationale d'Étude et de Conception de Moteurs d'Aviation) fut créée en 1945 à la suite de la nationalisation de la société Gnome et Rhône qui fabriquait à l'époque des moteurs d'avions, des motocyclettes et des bicyclettes. Elle intègre par la suite différentes entités, comme la Société Européenne de Propulsion en 1997 qui fait entrer Snecma sur le marché de la propulsion spatiale, marché sur lequel Safran est encore aujourd'hui. En 2000, une structure de *holding* prenant le nom de Groupe Snecma est mise en place pour gérer l'ensemble des participations dans ces diverses entités. La société prend alors le nom de Snecma Moteurs.

Elle reprendra son nom de Snecma lors de la fusion du Groupe Snecma avec Sagem en 2005 pour donner Safran. En mai 2016, Snecma devient Safran Aircraft Engines.

Observons maintenant quelques chiffres à propos de Safran Aircraft Engines :

- Plus de 32 500 moteurs CFM56 livrés en partenariat avec GE,
- Plus de 17 300 commandes ou intentions de commandes au 31/12/2018 pour le LEAP, remplaçant du CFM56,
- Plus de 500 moteurs militaires M88 livrés.

Les activités de Safran Aircraft Engines sont réparties en trois pôles : civil, militaire et spatial.

2.2.1 Les moteurs civils

Les moteurs civils les plus vendus au monde sont produits par Safran en partenariat avec General Electric au sein de la société CFM International. Dans cette société commune 50/50, toutes les activités sont partagées : design, développement, production, ventes et support. Les deux entreprises, animées par le succès du nouveau LEAP, ont dernièrement renouvelé leur partenariat jusqu'en 2040.

La nouvelle génération de turboréacteurs, le LEAP (Leading Edge Aviation Propulsion) remplace la famille des CFM et équipe de nouveaux avions : l'Airbus A320neo (mis en service en août 2016), le Boeing 737 MAX (mis en service en mai 2017) et le Comac C919 (certifié en décembre 2016) (voir Figure 2.5). Ce nouveau turboréacteur offre une consommation de carburant et des émissions de CO_2 réduites de 15% par rapport aux moteurs de la génération précédente.

De plus, Safran Aircraft Engines est également partenaire (ou fournisseur) de GE pour les moteurs de forte poussée : CF6, GE90 et GP7200.

Figure 2.5 - Les trois versions actuelles du LEAP

Parallèlement, Safran Aircraft Engines se place sur le marché de l'aviation régionale avec le SaM146, développé en coopération avec le motoriste russe UEC Saturn et mis en service en 2011 pour équiper le Sukhoï Superjet 100.

La gamme des moteurs civils de Safran Aircraft Engines est présentée dans le tableau suivant :

Figure 2.6 - Gamme des moteurs civils de Safran Aircraft Engines

2.2.2 Les moteurs militaires

La division moteur militaire de Safran regroupe également toutes les activités allant du développement au support après-vente pour les moteurs d'avions de combat ou d'entrainement, ainsi que pour les turbopropulseurs destinés aux avions de transport militaire. Safran Aircraft Engines fabrique les moteurs des avions de combat de Dassault comme le Mirage 2000 avec le moteur M53-P2 ou encore le Rafale avec le M88-2. Ces deux moteurs sont des turboréacteurs double flux avec post-combustion.

Safran Aircraft Engines motorise aussi l'Airbus A400M à l'aide de 4 turbopropulseurs TP400-D6. Cet avion de transport militaire polyvalent est entré en service en 2013. Le TP400-D6 a été développé par le consortium européen Europrop International, regroupant les motoristes MTU Aero Engines, Safran Aircraft Engines, Rolls-Royce et Industria de Turbo Propulsores (ITP).

La gamme des moteurs militaires de Safran Aircraft Engines est présentée ci-dessous :

Figure 2.7 - La gamme de moteurs militaires Safran Aircraft Engines

2.2.3 La propulsion électrique spatiale

Leader européen de la propulsion plasmique, Safran Aircraft Engines conçoit, développe et produit une large gamme de propulseurs électriques. Ils ont pour objectifs de maintenir à poste le satellite, faire un transfert d'orbite, une mise à poste ou encore des missions d'exploration et de navigation. Nous pouvons observer ci-dessous trois exemples de systèmes de propulsion spatiale de la gamme Safran

Figure 2.8 : Exemples de systèmes propulsifs électriques spatiaux : le PPS 1350-E, le PPS 5000 et le PPS 20K

2.3 PRESENTATION DU SECTEUR D'ACCUEIL ET DU SUJET DE STAGE

2.3.1 Le site de Villaroche

Mon stage s'est déroulé sur le site de Villaroche situé à 50 km de Paris. Le site s'étend sur plus de 80 hectares. Initialement créé en 1947 pour les essais au sol et en vol des moteurs civils et militaires, le site s'est fortement développé et réunit aujourd'hui environ 5000 personnes. L'établissement accueille les divisions Moteurs civils et Moteurs Militaires de Safran Aircraft Engines, la direction Commerciale et la direction Technique, Recherche & Technologie.

Figure 2.9 - Site de Safran Villaroche

2.3.2 La Division Transformation & Support DEV-TEC

Le stage s'est déroulé au sein de la direction technique, dans la Division Transformation & Support DEV-TEC. Cette division regroupe différents services autour de plusieurs fondamentaux parmi lesquels on peut citer le pilotage par le processus ou encore la culture du juste besoin. Le but *in fine* est de piloter des projets transverses au service de la transformation et du progrès, de manager l'informatisation de la direction technique et de développer des outils et méthodologies.

2.3.3 Le département Analyse de Données et Statistiques (Datalab)

Safran Aircraft Engines dispose d'importants volumes de données issus de différents secteurs : production, intégration, montage, tests, opérations. Les données sont dans des formats divers et sont souvent stockées sur des postes locaux au sein des différents secteurs ou dans des bases de données hétérogènes avec accès spécialisés.

Les différents secteurs de Safran Aircraft Engines demandeurs de traitement de données (bureaux d'études, Direction Industrielle, Ateliers Innovation, *Product Support Engineering* civils et militaires, etc.) exploitent ces données en relation avec leur contexte métier mais ont besoin d'outils d'analyse et d'un support technique en mathématiques appliqués, et en particulier en analyse de données.

C'est ainsi qu'a été mis en place un laboratoire d'analyse de données, de visualisation et de modélisation stochastique : le « *DataLab* ». Situé au sein de la direction technique et mis en place il y a un cinq ans, il offre les moyens, accès, outils et formations, permettant à chacun de prototyper leurs travaux d'analyses statistiques avec l'aide des *Data Scientists* du département.

Les missions du *DataLab* sont les suivantes :

- Spécifier et prototyper des outils d'accès et de gestion des données,
- Analyser les données et mettre à disposition des solutions prototypes,
- Diffuser la culture « analyse de données »,
- Développer des méthodes et outils avancés.

3 RAPPELS DE MACHINE LEARNING

Cette étude repose sur des concepts d'apprentissage automatique (*Machine Learning*) que nous allons détailler dans cette section.

L'apprentissage automatique est une branche de l'intelligence artificielle visant à donner à un agent intelligent la capacité d'apprendre à partir de données afin d'effectuer par la suite certaines tâches pour lesquels il n'a pas été spécifiquement programmé.

L'apprentissage automatique débute par une phase d'entraînement pendant laquelle l'agent va apprendre à effectuer une tâche. Cet entrainement est effectué de façon à optimiser une fonction « objectif » propre au réseau. A l'issue de cette phase, l'agent sera capable d'effectuer sa mission sur de nouvelles données.

Dès lors, on distingue plusieurs types d'apprentissages :

- **L'apprentissage supervisé** : l'agent dispose de données étiquetées pendant la phase d'entraînement (la réponse à la donnée d'entrée est connue). A l'issue de cette phase, il est capable de prédire la réponse à une nouvelle donnée d'entrée non-étiquetée.
- **L'apprentissage non supervisé** : l'agent dispose de données non-étiquetées pour son entraînement : il apprend lui-même les caractéristiques et structures au sein des données afin d'accomplir sa tâche.
- **L'apprentissage par renforcement** : l'agent est muni d'une fonction de récompense et apprend en prenant des décisions de façon à maximiser sa récompense.

3.1 RESEAU DE NEURONES CLASSIQUE (PERCEPTRON MULTICOUCHE)

Les réseaux de neurones sont des structures qui permettent l'apprentissage supervisé par l'expérience. Les neurones, briques de bases de ces structures, sont regroupés en différentes couches. L'information transite dans le réseau en passant par les différentes couches jusqu'à la sortie du système.

Usuellement, dans un réseau de neurones classique de type « Perceptron Multicouche », chaque neurone d'une couche est connecté à tous les neurones de la couche suivante :

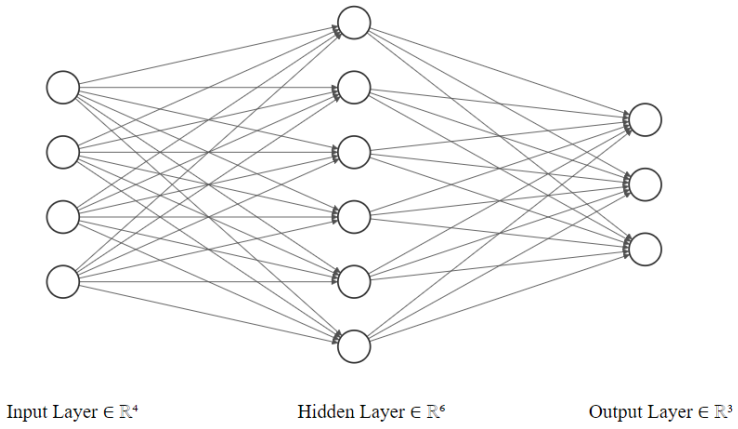

Figure 3.1 - Perceptron multicouche classique

Le réseau peut ainsi être représenté comme un empilement de couches successives. Chaque neurone est alors associé à plusieurs poids synaptiques qui seront modulés pendant la phase d'apprentissage. Le neurone combine ainsi les différents signaux d'entrée pondérés par les poids synaptiques, et fait passer la valeur résultante à travers une fonction d'activation. On dispose finalement en sortie d'une « réponse » donnée par la formule :

$$y = \varphi\left(\sum_{j=1}^{n} x_j w_j\right)$$

Où φ est la fonction d'activation, $(w_i)_{1\leq i\leq n}$ sont les poids synaptiques des neurones et $(x_i)_{1\leq i\leq n}$ les valeurs d'entrée.

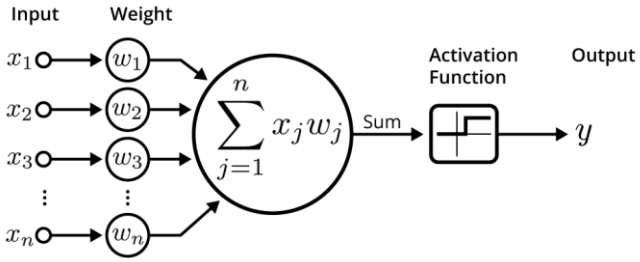

Figure 3.2 - Illustration d'un neurone artificiel

3.2 RESEAUX DE NEURONES CONVOLUTIFS (CNN)

Le fonctionnement d'un réseau de neurones convolutif est légèrement différent. Il repose sur l'utilisation d'opérations de convolution pour accomplir sa tâche. Dans ce cadre, la convolution agit comme un filtrage. Une couche d'un réseau CNN est composée de plusieurs filtres qui vont parcourir les données d'entrée. L'apprentissage consiste alors à définir les différents poids des filtres qui optimisent la fonction « objectif ».

3.2.1 Structure du réseau

Un CNN classique est généralement composé de 4 types de couches différentes :

- **La couche de convolution :**

Le but de cette couche est d'extraire des *features* (caractéristiques) dans les données d'entrée. Cette couche est composée de plusieurs filtres qui vont parcourir ces données. En sortie, elle produit ce qu'on appelle une « *feature map* » qui correspond à la donnée filtrée. Les matrices de poids des différents filtres (appelées *noyaux*) sont actualisées pendant la phase d'apprentissage.

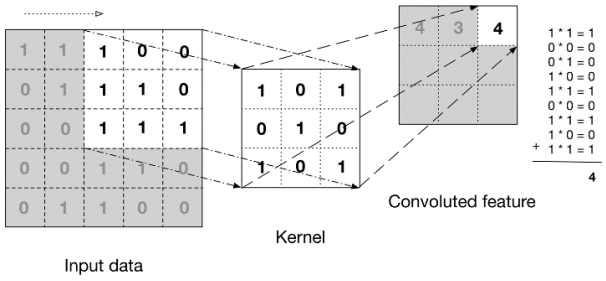

Figure 3.3 - Noyau de convolution

- **La couche de *pooling* :**

A mesure que l'on applique des filtres dans les couches de convolutions, on augmente drastiquement le nombre de paramètres à apprendre. Le but de la couche de *pooling* est de réduire la dimensionnalité spatiale et/ou temporelle (selon le type de données d'entrée) de la donnée intermédiaire tout en conservant les différentes caractéristiques précédemment extraites.

La couche de *pooling* a ainsi pour effet :

- De prévenir l'explosion combinatoire,
- D'éviter le surapprentissage (réseau devenant trop spécifique aux données d'entraînement),
- De désensibiliser le réseau à la position des *features*.

La couche de *pooling* fonctionne en mettant en commun des portions de données suivant différentes stratégies (*max-pooling*, *average pooling*…)

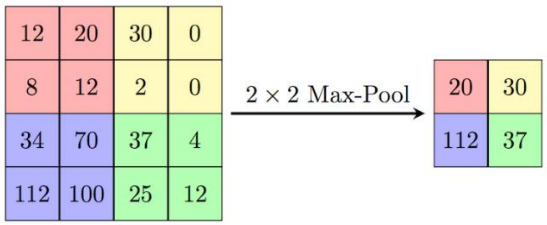

Figure 3.4 - Max-Pooling 2x2

- **La couche d'activation :**

Cette couche correspond à la fonction d'activation classique permettant « d'activer » ou non les neurones. L'implémentation la plus classique est la couche ReLU (unité de rectification linéaire) définie par $f(x) = \max(0, x)$. Elle permet d'augmenter les propriétés non linéaires du réseau.

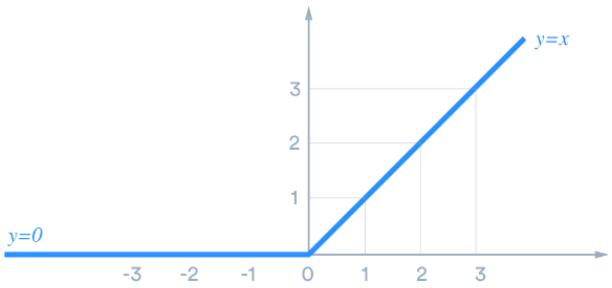

Figure 3.5 - Graphe de la fonction ReLU

- **La couche dense :**

Cette couche est similaire à celle que l'on peut trouver dans les réseaux de type « perceptron multicouche » : tous ses neurones sont connectés aux neurones des couches adjacentes. La sortie est alors une combinaison linéaire des entrées.

Le réseau CNN classique peut ainsi être représenté de la façon suivante :

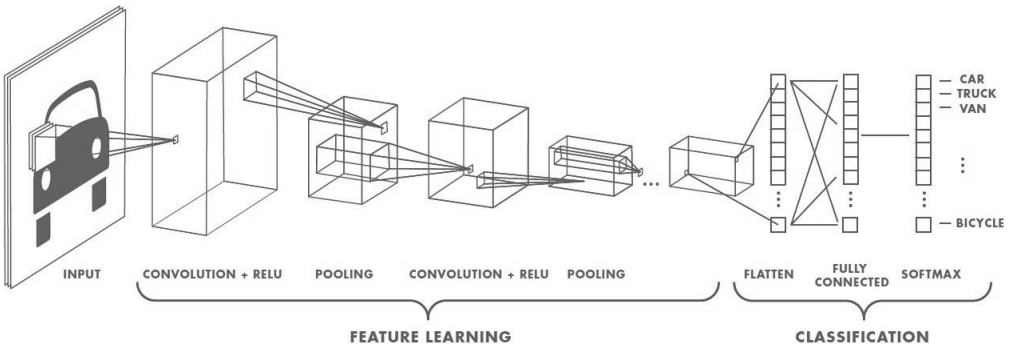

Figure 3.6 - Structure d'un CNN classique

3.2.2 Avantages par rapport à un réseau classique

Le réseau de neurones convolutif est particulièrement efficace lorsque les données présentent une cohérence spatiale ou temporelle (Images, Vidéos, Séries Temporelles). On peut notamment noter 3 grands avantages :

1. **Connectivité locale**

Les noyaux présents dans les couches de convolution limitent le nombre d'entrées des neurones (appelé champ récepteur). Il en résulte une réponse forte aux motifs d'entrée spatialement/temporellement localisés. Ainsi, les caractéristiques locales sont mises en évidence.

2. **Partage des poids**

Dans un réseau classique, il faut apprendre pour chaque neurone un ensemble de poids. Dans un réseau convolutif, pour un noyau donné, chaque poids est utilisé sur l'ensemble de la donnée d'entrée. On apprend ainsi un unique jeu de paramètres et non pas un jeu de paramètres pour chaque neurone.

3. **Invariance à la translation**

Par structure des noyaux, un motif est détecté indépendamment de sa localisation spatiale/temporelle au sein de la donnée.

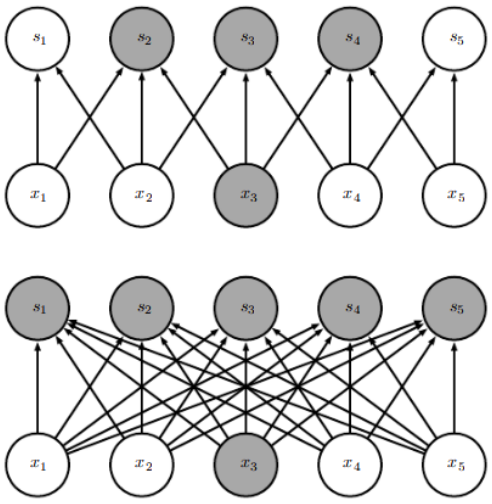

Figure 3.7 - Illustration de la connectivité locale : dans un CNN (en haut) avec une taille de filtre 3, seules les sorties s_2,s_3 et s_4 sont affectées par l'unité d'entrée x_3. Dans un réseau classique (en bas), toutes les sorties sont affectées par l'entrée x_3

4 CONTEXTE DU STAGE

Cette partie présente les objectifs, le périmètre et la démarche de l'étude réalisée dans le cadre de ce stage.

4.1 OBJECTIFS

Ce stage a pour objectif de tester la puissance et l'efficacité des réseaux de neurones convolutifs dans le cadre de l'analyse de données de vols.

On dispose ainsi de fichiers de données correspondant à des vols d'une flotte d'avions d'une même compagnie. Ces données sont détaillées dans la partie suivante.

Le travail réalisé s'articule principalement autour de la détermination des phases d'un vol et au potentiel enrichissement de la signature de chacune de ces phases.

4.2 FORMAT DES DONNEES

Les données dont nous disposons proviennent du QAR, un enregistreur de vols implanté dans les avions. Le QAR enregistre 42 paramètres différents à une fréquence de 1Hz. Ainsi, pour un vol donné, on dispose de 42 valeurs par seconde tout au long du vol. Ces différents paramètres que l'on appellera « caractéristiques » sont détaillés dans le *Tableau A.1 - Récapitulatif des paramètres des fichiers QAR* en annexe.

A titre d'exemple, on peut citer de manière non-exhaustive : l'altitude standard, la vitesse indiquée ou encore le débit de carburant des différents systèmes propulsifs.

Les données se présentent sous la forme de fichiers d'extension CSV (*Comma Separated Values*). Chaque fichier correspond à un vol. Dans notre cas, il s'agit de données de vols provenant d'une unique compagnie et d'un unique type d'avion. Les fichiers CSV peuvent être convertis en tableaux Excel afin d'être représentés de manière plus lisible :

Date	Phase	Altitude	Vitesse	…	N1
19:26:06	0	300	0	…	0
19:26:07	1	300	0	…	10
…	…	…	…	…	…
23:52:08	0	120	0	…	0

Tableau 4.1 - Représentation du format des fichiers de données (valeurs factices à titre d'exemple)

On dispose d'environ 19000 fichiers correspondants tous à des vols différents. La longueur du fichier (vu comme tableau de données) varie donc en fonction du temps de vol. En moyenne, un fichier contient 5500 lignes.

4.3 OUTILS D'ANALYSE

Les différentes implémentations ont été réalisées avec le langage Python et plusieurs librairies dont :

- **Tensorflow**

Tensorflow est une librairie *open source* initiée par Google et utilisée pour développer des scripts de *machine learning*. Elle dispose de nombreuses fonctions de base permettant d'implémenter des réseaux de neurones ainsi que divers types d'apprentissages.

- **Pandas**

Pandas est une librairie dédiée à l'analyse de données. Elle propose entre autres, des structures de données et implémente différentes fonctions pour la manipulation de ces dernières. Il est notamment possible de charger des fichiers CSV grâce à cette librairie, et de les stocker dans des structures adéquates appelées *DataFrame*.

5 TRAVAUX PRELIMINAIRES

5.1 NETTOYAGE DES DONNEES

Le chargement d'un premier fichier de vol met en évidence la nécessité de prétraiter les données. Comme on peut le voir dans la figure suivante, certaines données sont bruitées (*MACH_R, HEAD_TRUE…*). De plus, certains capteurs ont une fréquence d'acquisition inférieure à 1hz, résultant ainsi en des fonctions escaliers qu'il serait pertinent de lisser.

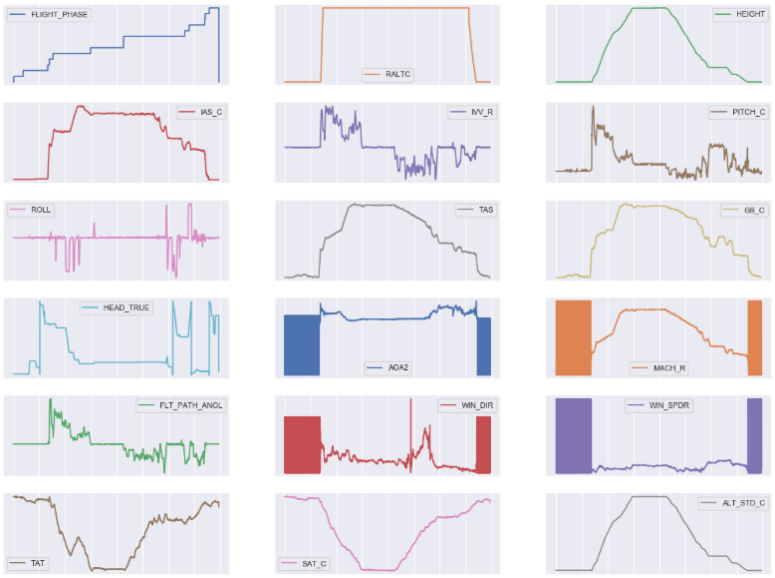

Figure 5.1 - Quelques signaux issus d'un vol

5.1.1 Lissage des signaux

On procède au lissage des signaux par la méthode de la moyenne glissante : on sélectionne une fenêtre de taille $N \in \mathbb{N}^*$ et on calcule la moyenne pour chaque point des N valeurs consécutives et centrées sur lesdits points.

La valeur de N est choisie empiriquement et dépend fortement de l'ordre de grandeur des « évènements » survenant au sein de chaque signal. Ainsi pour de meilleurs résultats, il est pertinent de choisir un N adapté à chaque canal du signal multivarié. N doit être suffisament grand pour que le lissage ait un impact mais pas trop grand pour ne pas « l'applatir »

Cette méthode nous permet d'obtenir des données de meilleure qualité comme on peut le voir sur la figure ci-contre.

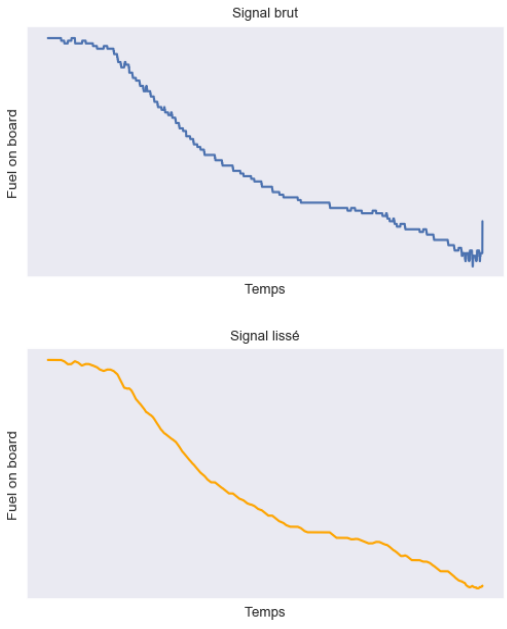

Figure 5.2 - Illustration du lissage des signaux

5.1.2 Suppression des valeurs extrêmes

Certains signaux sont particulièrement bruités, notamment dans les phases où l'avion est relativement lent (Roulage, Taxi…)

La courbe du haut de la Figure 5.3 représente l'AOA en fonction du temps.

L'AOA (*Angle Of Attack*) correspond à l'incidence de l'avion : c'est l'angle formé par la trajectoire (vecteur du vent relatif) et l'axe longitudinal de l'avion (voir figure A.1 de l'annexe).

C'est une valeur critique en aviation. A valeur d'incidence importante (de l'ordre de 20°), l'avion est susceptible de « décrocher » : les filets d'air se « décollent » de la surface de l'aile, résultant en une chute de la portance et dans le pire des cas à une abatée : l'avion chute.

En théorie, le décrochage peut survenir à n'importe quelle vitesse, cependant il a souvent lieu à faible vitesse où l'avion doit, pour maintenir son plan, avoir un certain angle de tangage à cabrer. Dans ce cas, l'incidence est susceptible d'être élevée.

En pratique, la sonde AOA doit avoir l'information correspondante au vecteur vent relatif pour calculer l'incidence. Cependant, à très faible vitesse, cette valeur est proche de 0. Il en résulte un signal complétement bruité comme on peut le voir sur la figure.

Ces différentes considérations aéronautiques nous permettent de prendre une décision quant à la correction des valeurs aberrantes : Il parait légitime de modifier ces valeurs à 0 lors des phases où le signale oscille de manière extrême.

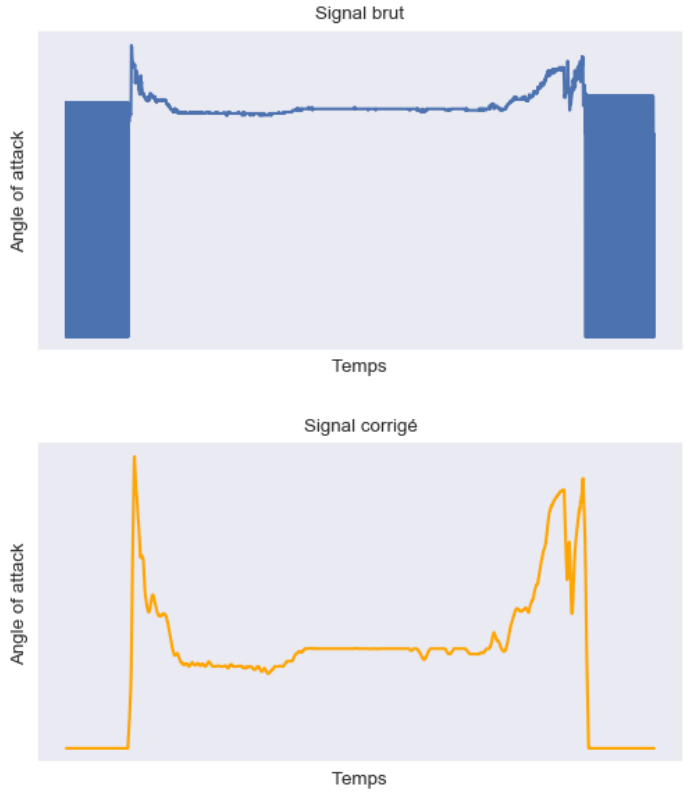

Figure 5.3 - Suppression des valeurs extrêmes

5.1.3 Suppression des valeurs aberrantes

Certaines séries sont sujettes à l'apparition de valeurs aberrantes ponctuelles comme on peut le voir dans la figure ci-dessous. La plupart de ces valeurs aberrantes sont traitées et « aplaties » lors du lissage des signaux comme vu précédemment. Cependant, on peut traiter les valeurs très anormales en amont de façon à être sûr de les supprimer correctement.

Il s'agit ici (figure de gauche) de la température aux abords de l'aéronef (*Static Air Temperature*). On peut en effet distinguer un pic qui n'a pas lieu d'être. En calculant les différences finies de ce signal, on obtient la courbe suivante (figure de droite) :

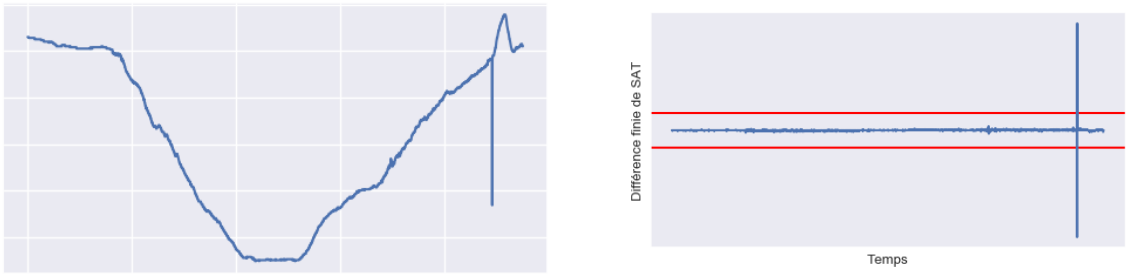

Figure 5.4 – Température SAT et différences finies du signal avec valeur aberrante

On peut définir un seuil S de manière logique, en dessous duquel une valeur peut être considérée comme aberrante. Par exemple, il est peu probable que la température puisse varier de plus de 5° en une seconde (lignes rouges sur le graphe).

Ainsi, si l'on dispose dans nos données de deux valeurs consécutives de température T_{t-1} et T_t dont la différence est supérieure (en valeur absolue) à S, on peut remplacer la valeur à l'instant t, en prenant par exemple la moyenne des deux points autour de cette valeur :

$$T'_t = \frac{T_{t-1} + T_{t+1}}{2}$$

Evidemment, il faut être capable de détecter s'il y a des valeurs aberrantes non pas ponctuelles mais sur un intervalle de temps : dans le cas où T_{t+1} est également une valeur aberrante, la formule précédente devient caduque.

On procède de même pour les autres canaux.

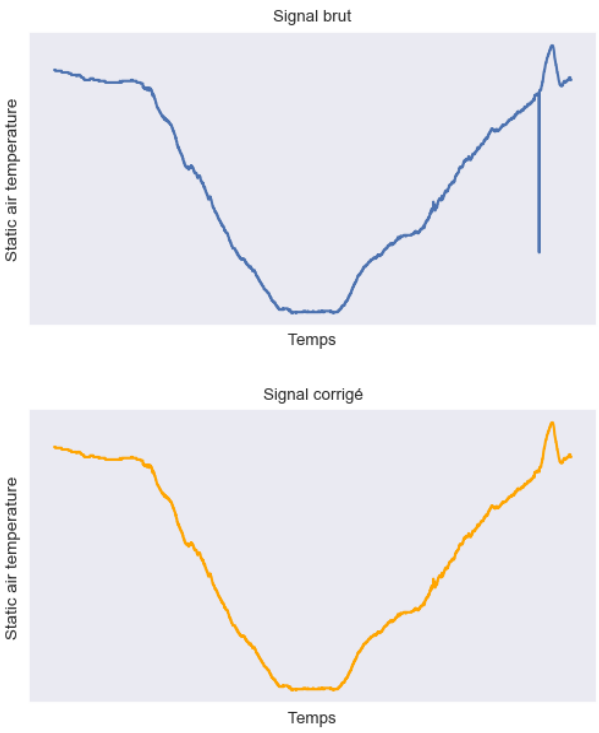

Figure 5.5 - Suppression des valeurs aberrantes

5.2 SELECTION DES PARAMETRES PERTINENTS

Dans un premier temps, on établit la matrice des corrélations des variables quantitatives pour essayer de repérer les éventuelles similarités entre les signaux (Figure 5.7). Cette matrice nous permettra par la suite de définir les variables qu'il est pertinent d'utiliser dans notre implémentation, en garantissant une certaine diversité.

Certaines corrélations importantes sont prévisibles : les caractéristiques de vitesses (*IAS*, *GS_C*, *TAS*) sont dans la pratique très corrélées à quelques différences près (Vent, différence anémo-barométrique), de même pour les caractéristiques d'altitudes (*ALT_STD_C*, *HEIGHT*) qui sont sensiblement identiques.

Le choix des paramètres pertinents pour une étude souhaitée peut être aidé par deux aspects : la connaissance pratique des données et l'observation analytique des corrélations.

Par exemple, pour faire un modèle de régression, il serait peu pertinent de prendre à la fois les valeurs d'IAS et de TAS qui sont extrêmement corrélées et seraient donc redondantes dans le modèle. L'idéal est d'avoir recours à des variables qui couvrent à la fois un large spectre d'applications pratiques (des données de vitesse, d'altitudes, de paramètres moteurs) mais qui sont également pertinentes pour expliquer la variable cible. Dans cette optique, avoir recours à la direction du vent pour prédire la phase de vol n'est *a priori* pas un choix à envisager.

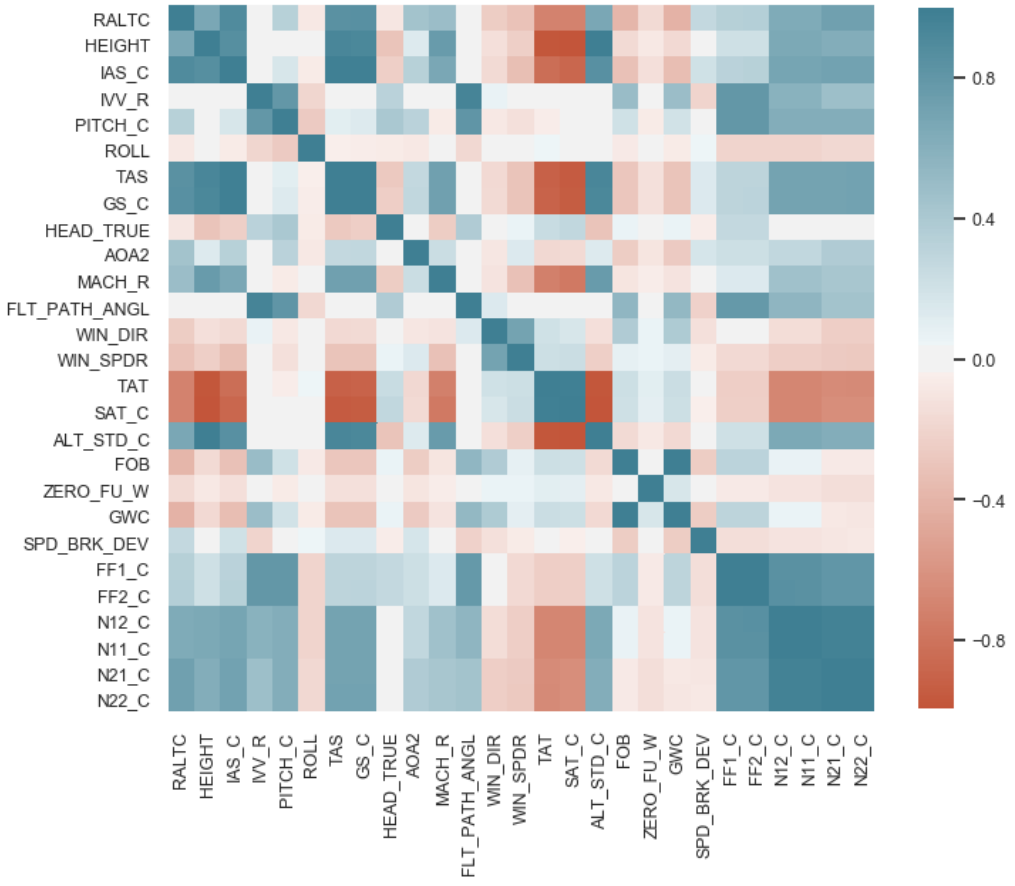

Figure 5.6 - Matrice de corrélations des différentes variables quantitatives

6 IMPLEMENTATION DE RESEAUX DE NEURONES

Après avoir nettoyé les données, nous pouvons passer au vif du sujet et essayer d'inférer des résultats. L'objectif étant toujours d'arriver à « découper » un vol donné en différentes phases distinctes de façon à enrichir sa description.

6.1 PERCEPTRON MULTICOUCHE CLASSIQUE (MLP)

La première structure réalisée est un perceptron multicouche classique, qui prend en entrée un instant d'un vol (une ligne) et classifie la phase de vol correspondant à cet instant. Le but premier était de prendre en main la librairie *Tensorflow* et de tester une implémentation d'un réseau de neurones sur le jeu de données à disposition. A ce stade du stage, nous ne disposions pas encore du nombre de données suffisant pour réaliser un réseau plus complexe.

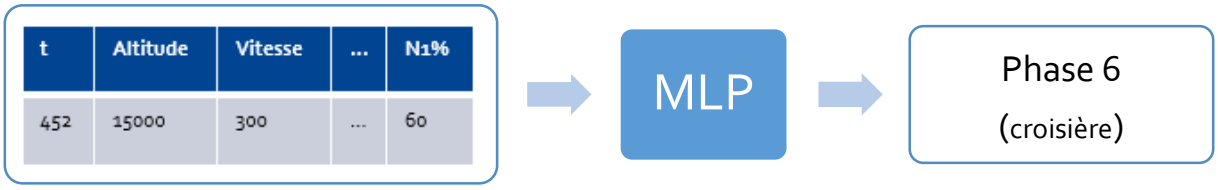

Figure 6.1 – Représentation de la tâche de classification

Nous disposons dans cette partie de 66 vols (pour un total d'environ 400 000 lignes). Ces 66 vols peuvent être représentés dans un tableau, où chaque ligne représente l'instant d'un des vols :

	T	Phase	Altitude	Vitesse	...	N1%
Vol 1	1	0	150	0	...	0
	⋮	⋮	⋮	⋮	⋮	⋮
	5550	13	178	15	...	50
Vol 2	1	0	124	0	...	...
	⋮	⋮	⋮	⋮	⋮	⋮
	4488	13	58	0	...	40
...	⋮	⋮	⋮	⋮	⋮	⋮
Vol 66	1	0	28	0	...	0
	⋮	⋮	⋮	⋮	⋮	⋮
	2847	13	245	5	...	35

Tableau 6.1 - Représentation d'un tableau de données

Nous avons choisi 8 caractéristiques : True Air Speed, Vertical Speed, Pitch, Altitude, Fuel on Board, Total air Temperature, Fuel Flow 1, Config sur la base des critères établis dans la section précédente.

Les sorties sont les différentes phases de vols possibles $\{0, \dots, 13\}$ (Voir Tableaux récapitulatifs des paramètres et phases de vols en annexes). Les phases 14,15 et 16 ne sont pas présentes dans les vols à disposition dans cette partie.

6.1.1 Tâche de classification et datasets

La tâche que nous essayons de réaliser dans cette partie est appelée classification. Elle consiste à associer à chaque individu (une ligne du tableau), la classe correspondante (phase associée). Etant dans le cadre de l'apprentissage supervisé (exemples annotés), le but est d'entraîner le réseau de neurones sur une partie de ces données. A l'issue de l'entraînement, le réseau doit être capable de classer de nouveaux exemples qu'il n'a *a priori* pas encore rencontrés (généralisation).

Dans ce cadre, nous divisons notre jeu de données en trois *datasets* :

- **Le *dataset* d'entraînement** : le réseau de neurones apprend sur ces exemples annotés, en essayant de minimiser sa fonction « objectif ».
- **Le *dataset* de validation** : ponctuellement durant l'apprentissage, le réseau teste ses capacités sur le *dataset* de validation pour voir s'il n'est pas sujet au surapprentissage, cas où le réseau devient trop spécifique au *dataset* d'entraînement. Si les capacités du réseau s'améliorent sur le *dataset* d'entraînement mais diminuent sur le *dataset* de validation, il faut arrêter l'apprentissage. Dans tous les cas, le réseau n'apprend pas sur les exemples de validation, ces derniers jouent simplement un rôle d'indicateur durant l'apprentissage.
- **Le *dataset* de test** : à l'issue de l'entrainement, c'est sur cet ensemble que nous testons l'efficacité du réseau. Nous détaillerons plus tard les indicateurs utilisés à cette fin.

La répartition des données entre ces 3 ensembles sera discutée dans les prochaines parties. D'ores et déjà, on fait face à un problème de balancement que nous allons détailler ci-après.

6.1.2 Jeu de données non balancé :

Visualisons la répartition des données en fonction des phases de vols :

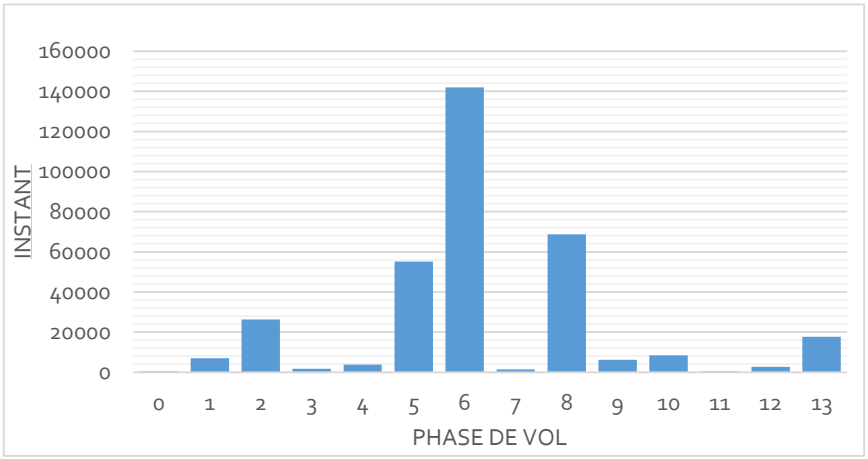

Figure 6.2 - Répartition des phases de vols dans le dataset

La figure ci-dessus illustre très nettement le caractère non balancé des données : au sein des 66 vols, il y a, par exemple, beaucoup plus d'instants où l'avion est en phase 6 (croisière) qu'en phase 3 (décollage). Tout cela est logique : dans un vol, un avion passe presque 60% du temps en phase croisière alors que le décollage ne dure que quelques secondes.

Cette particularité est gênante pour l'entraînement des réseaux de neurones qui risquent de surapprendre les classes majoritaires du fait de leur prépondérance dans le jeu de données d'entraînement.

Il faut donc procéder à un rebalancement du jeu de données. Cela peut se faire de plusieurs façons :

- ***Undersampling*** **:** retirer aléatoirement des observations dans les classes majoritaires,
- ***Oversampling*** : ajouter de nouvelles observations « *artificielles* » dans les classes minoritaires,
- **Hybride** : procéder successivement à de l'*undersampling* et de l'*oversampling.*

Nous avons mis en œuvre la méthode hybride. La partie *oversampling* a été réalisée via un algorithme de suréchantillonnage par ajout d'instances synthétiques (SMOTE) détaillé en annexe.

Ainsi, chacune des classes comporte le même nombre d'individus.

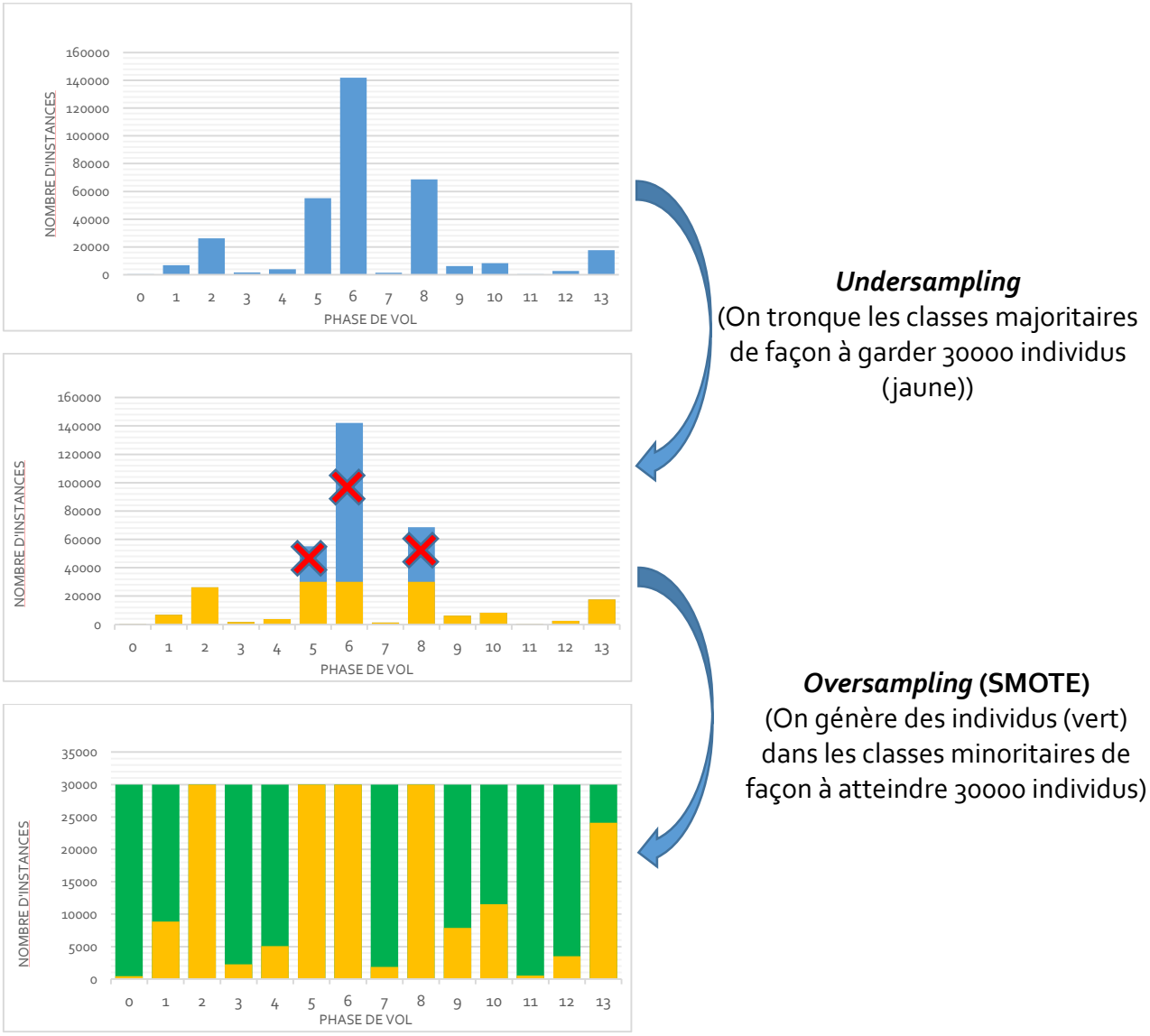

Figure 6.3 - Illustration du balancement des données

Il est important de préciser que le rebalancement se fait uniquement sur le *dataset* d'entraînement. Dans le cas contraire, on risquerait d'avoir des redondances : apparition du même point dans les *datasets* d'entraînement et de test : l'efficacité du réseau testé sur le *dataset* de test serait alors faussée car le réseau aurait déjà rencontré les exemples pendant l'entraînement.

6.1.3 Définition des métriques et efficacité du réseau

Pour estimer la robustesse du modèle, nous devons définir des métriques qui vont quantifier l'efficacité de la classification après la phase d'entraînement. Formalisons tout d'abord mathématiquement le problème :

Pour $C \in \mathbb{N}^*$, on note $\mathcal{C} = \{1, \dots, C\}$ l'ensemble des labels. Dans notre cas, il s'agit des différentes phases de vols.

Un individu l (un instant d'un vol quelconque) peut être vu comme un vecteur $x^{(l)} = (x_1^{(l)}, \dots, x_r^{(l)})$ avec r le nombre de caractéristiques (telles que définies dans la partie Format des données du chapitre 4) et le label associé $y^{(l)}$ un entier dans $\mathcal{C}$.

Si le réseau est caractérisé par la fonction paramétrique de prédiction f, on peut associer à chaque vecteur de « features » $x^{(l)}$ une prédiction associée $\hat{y}^{(l)}$ telle que :

$$\hat{y}^{(l)} = f\left(x^{(l)}\right)$$

A partir de m individus d'un jeu de test, on pose $E = \left\{y^{(1)}, \dots, y^{(m)}\right\}$ ensemble des étiquettes réelles (*True labels*) et $P = \left\{\hat{y}^{(1)}, \dots, \hat{y}^{(m)}\right\}$ l'ensemble des prédictions du réseau (*Predicted labels)* associées aux exemples $l = 1, \dots, m$

Définissons la **matrice de confusion** $N = \left(n_{i,j}\right)_{1 \le i,j \le C}$ telle que :

$$\forall i, j \in \mathcal{C}, \qquad n_{i,j} = \sum_{k=1}^{m} \mathbf{1}_{y^{(k)}=i,\ \hat{y}^{(k)}=j}$$

$n_{i,j}$ correspond au nombre d'exemples issus de la classe i et prédits comme appartenant à la classe j.

Cette matrice est un bon moyen pour résumer l'efficacité du réseau durant la phase de test. Dans le cas d'un classificateur parfait, seuls les coefficients $n_{i,i}, i \in \mathcal{C}$ sont non nuls : tous les exemples d'une classe i donnée sont correctement prédits. On cherche idéalement à s'approcher d'une matrice diagonale, représentative d'un classificateur parfait.

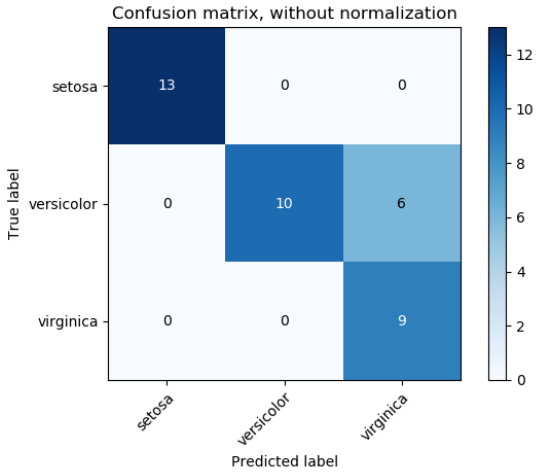

Figure 6.4 - Exemple d'une matrice de confusion d'un problème à 3 classes

A partir de cette matrice on définit deux métriques de précisions :

- **Accuracy classique :**

$$Acc(E,P) = \frac{Tr(N)}{m}$$

Il s'agit du rapport du nombre d'exemples bien classés sur le nombre total d'exemples.

- **Per class accuracy :**

$$Pca(E,P) = \frac{1}{C}\sum_{i}\frac{n_{i,i}}{\sum_j n_{i,j}}$$

C'est la moyenne de l'*accuracy* de chaque classe.

6.1.4 Structure du réseau, fonction « objectif » et répartition du dataset

Le perceptron multicouche implémenté a la structure suivante :

- *Input Layer* (8)
- *Hidden Layer* (32, Activation Relu, Régularisation Dropout, Dense)
- *Output Layer* (14, Softmax)

Le nombre correspond à la dimension de la couche (nombre de neurones). Les dimensions des couches d'entrées et de sorties sont respectivement définies par la dimension des données d'entrée (8 caractéristiques) et le nombre de classes possibles (14). La dimension de la couche cachée a été déterminée empiriquement, à la suite de plusieurs tests d'entraînement.

On applique également une régularisation de type *Dropout* : durant l'entraînement, plusieurs neurones sont désactivés aléatoirement pour éviter le surapprentissage. La couche cachée a pour fonction d'activation une *ReLU* définie précédemment. Finalement, la couche de sortie a une fonction d'activation *softmax* définie par :

$$\sigma : \mathbb{R}^C \to \mathbb{R}^C \quad z \mapsto (\sigma(z)_1, \sigma(z)_2, \ \ldots, \sigma(z)_C) \text{ avec } \sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \ , \ \forall i \in \{1, \ldots, C\}$$

On a ainsi en sortie, un vecteur dont la somme des composantes est égale à 1 ce qui nous donne pour chaque exemple la probabilité d'appartenance à chaque classe.

La sortie de notre réseau peut donc être notée $\hat{y}^{(i)} = f\left(x^{(i)}\right) = \text{indmax} \circ \sigma \circ g\left(x^{(i)}\right)$ pour une certaine fonction g. La fonction $\text{indmax} : \mathbb{R}^C \to \mathcal{C}$ renvoie simplement la classe associée à la composante la plus élevée.

La fonction de coût est une fonction d'entropie croisée définie par :

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} log\, \sigma\left(g\left(x^{(i)}\right)\right)_{y^{(i)}}$$

Cette fonction guide le réseau durant son apprentissage en revoyant une valeur élevée pour les exemples mal classés et une valeur proche de 0 pour les exemples correctement classés.

Finalement, le jeu de données a été séparé en trois *datasets* :

- *Train dataset* : 50 vols
- *Test dataset* : 8 vols
- *Val dataset* : 8 vols

6.1.5 Résultats

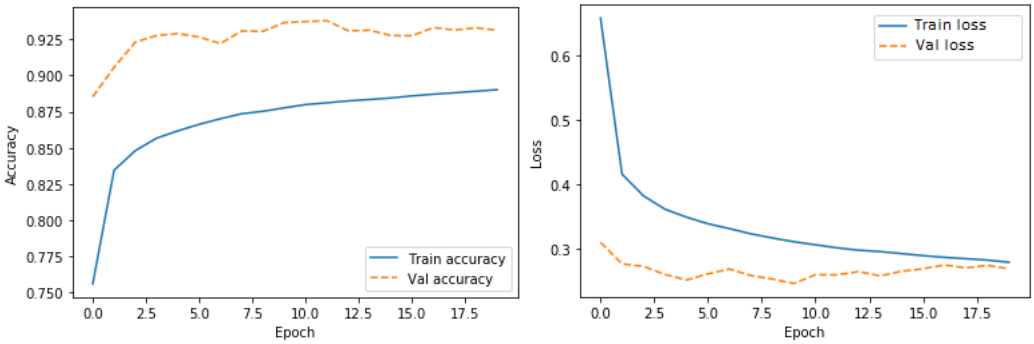

Figure 6.5 - Courbes des accuracies au cours de l'entraînement

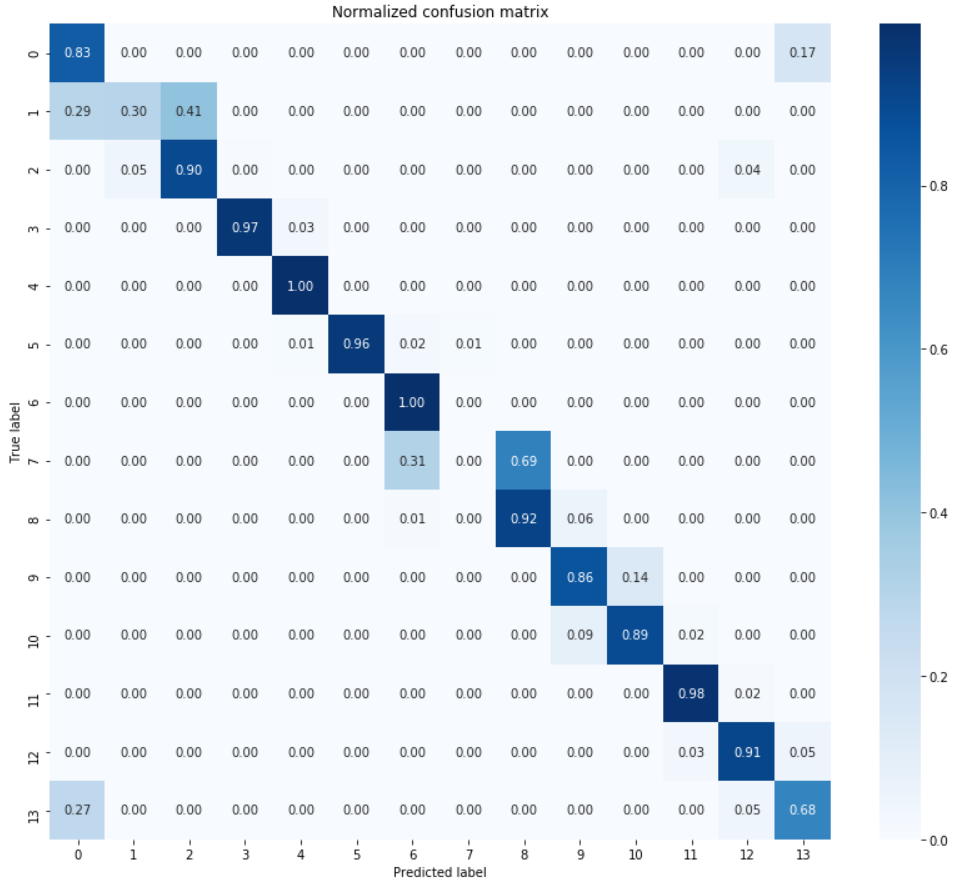

Figure 6.6 - Matrice de confusion normalisée

Test dataset accuracy (Acc)	Test dataset Per Class Accuracy (Pca)
0.93	0.79

Tableau 6.2 - Résultats de la première implémentation

Note : les phases de vol correspondant aux labels se trouvent en annexe.

La matrice de confusion normalisée (cf. figure ci-dessus) démontre la bonne *accuracy* du classement pour certaines classes : par exemple la classe 3 (décollage) qui est prédite correctement à 97%.

Cependant certaines classes sont très mal prédites. Par exemple, 30% des exemples de la classe 7 (*level change*) ont été prédits comme appartenant à la classe 6 (croisière) et près de 70% comme appartenant à la classe 8 (descente). Cette erreur de classification est en partie due au fait que la discrimination d'une phase devrait se faire en analysant le vol sur un intervalle et non ponctuellement. Par exemple, si l'on regarde à un instant *t* fixé, il n'y a aucune raison que les paramètres d'un changement de niveau de vol (7) en descente et d'une descente (8) soient différents. Ce qui discrimine ces deux phases, c'est le contexte dans lequel elles se situent : une descente n'intervient pas en plein milieu d'un vol *a contrario* d'un changement de niveau de vol.

Ce que l'on retient de tout cela est la nécessité, pour avoir une classification correcte, de s'intéresser au vol de manière plus globale, ou au moins sur des intervalles de temps plus longs, et non pas de manière instantanée/ponctuelle.

C'est ce qui a motivé la seconde implémentation réalisée, à l'aide de réseaux de neurones convolutifs.

6.2 SEGMENTATION D'UN VOL PAR RESEAU U-NET

Dans cette partie, nous allons essayer d'extraire de l'information en considérant non plus des instants de vol, mais un vol dans sa globalité. On dispose ainsi d'un ensemble de séries temporelles correspondant aux différents paramètres évoluant au cours de ce vol. Le but ici est toujours de réussir à déduire de ces paramètres, les différentes phases qui composent le vol en question.

Le fait d'interpréter le vol dans sa globalité transforme le problème initial en un problème de segmentation. Ainsi, notre prochain réseau prendra en entrées, les caractéristiques associées à un vol dans sa globalité, et retournera une segmentation de ce vol en différentes phases.

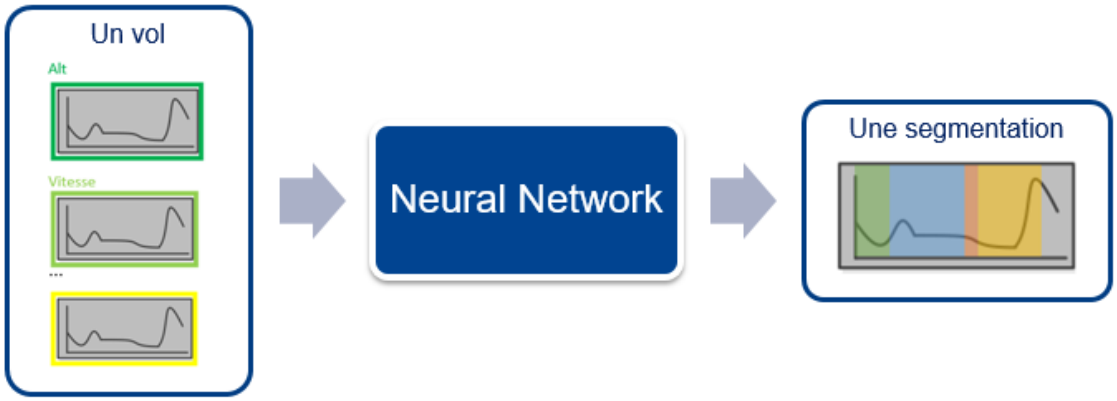

Figure 6.7 - Problème de segmentation d'un vol

6.2.1 La segmentation dans le traitement d'image

Ce problème s'inspire de la segmentation d'image qui est un domaine déjà largement connu et maitrisé. En outre, les réseaux de neurones convolutifs sont très performants pour réaliser cette tâche, pour les raisons mentionnées dans le chapitre 3. Détaillons maintenant la tâche de segmentation d'une image dans le cadre du *machine-learning*. Nous dresserons par la suite une analogie avec notre problème.

Le but de la segmentation d'image est de partitionner une image en rassemblant les différents pixels qui la composent en plusieurs classes suivant certains critères. Par exemple, dans la figure ci-dessous, on peut apercevoir en haut, une image et en bas, la segmentation associée. Les différentes classes correspondent ici aux différents éléments de l'environnement : voitures, routes, arbres…

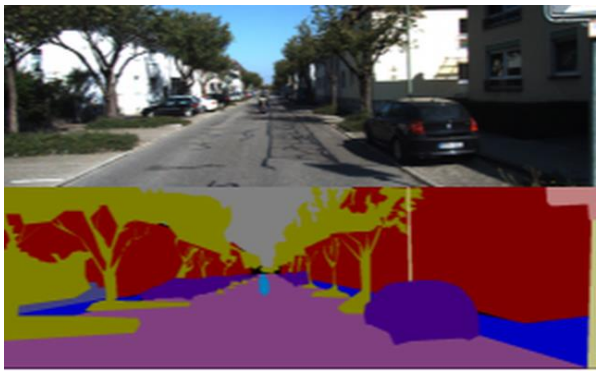

Figure 6.8 - Exemple de segmentation d'image

L'apprentissage dans le cadre de la segmentation supervisée est similaire à ce que l'on a expliqué précédemment. On dispose d'un ensemble d'exemples et de labels associés : la différence ici, c'est que les labels sont un ensemble ordonné de plusieurs classes (de même taille que la donnée d'entrée, car pour chaque pixel de l'image, on associe une classe). L'apprentissage se fait sur un ensemble de couples (Image, Masque) que l'on met à disposition du réseau :

Figure 6.9 - Exemples de données d'entraînement pour la segmentation d'image

Les données sont codées sous formes de tableaux multidimensionnels. Par exemple une image carrée en couleur de 128 pixels sera représentée par un tableau de dimension $3 \times 128 \times 128$ (3 canaux pour les couleurs (RGB), hauteur de 128, largeur de 128) à valeurs dans $[0,255]$ (intervalle de codage RGB). Le masque correspondant sera un tableau de dimension 128×128 à valeurs dans $[1, C]$ avec C le nombre de classes considérées. Ainsi à chaque pixel de l'image, on associe bien une classe.

A l'issue de l'entraînement, le réseau de neurones doit être capable de générer la prédiction d'un masque (i.e. segmentation) sur des exemples d'images inédites :

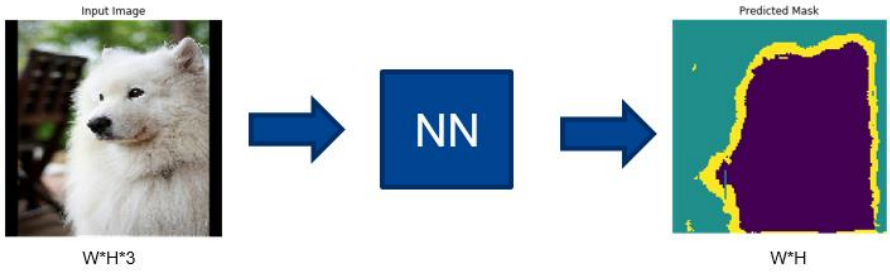

Figure 6.10 - Segmentation d'une image par un CNN

6.2.2 Application à la segmentation de données de vols

Notre problème est quasi-identique à celui de la segmentation d'images : seul le type de données change. Pour un vol donné, nous disposons d'un ensemble de séries temporelles que nous pouvons représenter sous forme d'un tableau de dimension $N \times F$ avec N la durée du vol en secondes et F le nombre de caractéristiques. En sortie, le réseau devra être capable de retourner un tableau de dimension N à valeurs dans l'ensemble des phases de vols possibles, et correspondant à la segmentation du vol.

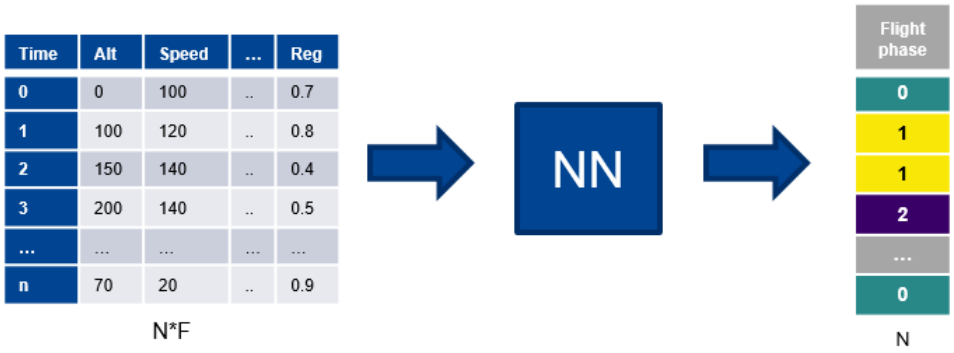

Figure 6.11 - Segmentation du vol

6.2.3 Réseau U-Net

Pour réaliser cette segmentation, nous nous sommes intéressés à un réseau appelé *U-Net*. Ce réseau proposé par Olaf Ronneberger, Philipp Fischer, et Thomas Brox en mai 2015 était initialement utilisé pour la segmentation d'images biomédicales. Son principal avantage est son efficacité même lorsque l'on dispose de peu de données d'entraînement.

Notre but est d'adapter ce réseau conçu pour la segmentation d'images à la segmentation de séries temporelles multivariées.

L'architecture de ce réseau est la suivante:

U-Net est un réseau basé sur une architecture FCN (*Fully Convolutionnal Networks*). Nous détaillerons cette architecture dans les parties suivantes et nous verrons que cette structure présente un réel avantage pour notre étude.

Contrairement aux réseaux convolutifs classiques comme celui présenté dans le chapitre 3, *U-Net* est composé d'une phase contractante et d'une phase expansive lui conférant une forme de U à l'origine de sa dénomination.

La phase contractante est semblable à un CNN classique, elle extrait des caractéristiques. Au fur et à mesure que la donnée « avance » dans cette phase, on assiste à un gain du nombre de caractéristiques extraites (*features*), tandis que la dimension spatiale diminue en conséquence des couches de *pooling*.

La phase expansive permet de retrouver le format de données initiale à l'aide d'opérations d'*upsampling* et d'associer les *features* extraites aux régions géographiques de la donnée. Lors de cette reconstruction, elle combine également par des opérations de concaténations, des caractéristiques hautes résolutions de la voie contractante pour en améliorer le résultat.

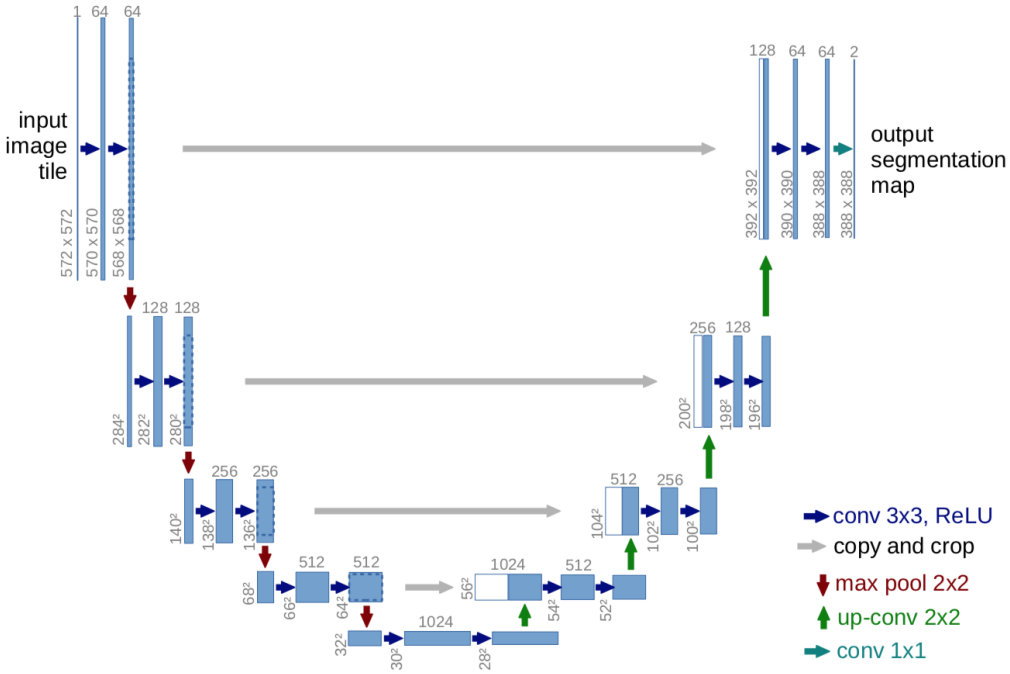

Figure 6.12 - Structure du réseau U-Net

6.2.4 Adaptation du réseau à la segmentation unidimensionnelle

Suivant le type de données, la segmentation peut se faire à des dimensions différentes. Il est important de noter que dans ce cadre, on appelle dimension, le degré de liberté du filtre sur la donnée. Par exemple, pour les images, on réalise une segmentation 2-D car le filtre parcourt l'image de haut en bas et de droite à gauche. La confusion consiste à croire que la dimension de la segmentation représente le format de la donnée d'entrée, ce n'est pas le cas. Par exemple dans le cadre des images en couleurs, le format de la donnée d'entrée est en 3-dimensions (canaux RGB, hauteur, largeur), tout comme les noyaux de convolutions sont des « blocs » en 3 dimensions. Cependant, leur parcours ne se fait pas en profondeur, mais uniquement dans les deux directions spatiales de la donnée.

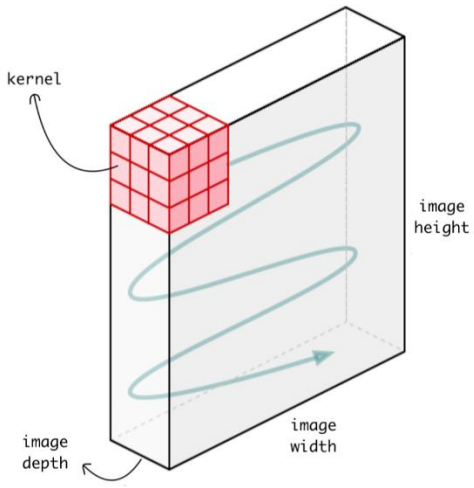

Figure 6.13 - Parcours d'un filtre pour la segmentation 2-D

Dans le cadre des séries multivariées, on se situe dans le domaine de la segmentation 1-D. La donnée n'est plus à support spatial mais à support temporel. Ainsi, peu importe le nombre de canaux (caractéristiques) du signal, on reste dans le domaine de la segmentation unidimensionnelle. Les filtres du réseau convolutif sont de dimensions $k \times f$ avec k le nombre de caractéristiques (canaux du signal multivarié) et f une « taille de filtre » choisie à la discrétion du programmeur. Les filtres se « promènent » ainsi de haut en bas sur la donnée, selon le support temporel de ces dernières.

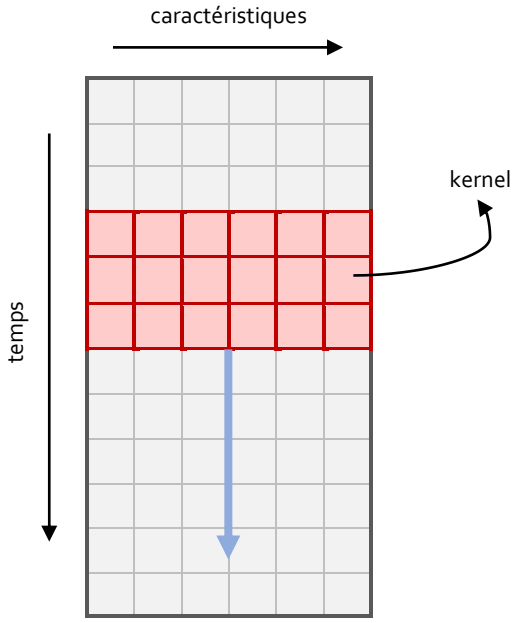

Figure 6.14 - Parcours d'un filtre pour la segmentation 1-D

L'implémentation effectuée nous a permis de convertir le réseau *U-Net* initialement prévu pour de la segmentation bidimensionnelle en réseau prenant en charge la segmentation unidimensionnelle.

6.2.5 Définition des métriques :

En notant comme précédemment $\mathcal{C} = \{1, \dots, C\}$ l'ensemble des classes, on pose, $X = (X_t)_{t\in\mathbb{N}}$ et $Y = (Y_t)_{t\in\mathbb{N}}$ deux processus stochastiques d'espaces d'états respectifs $\mathbb{R}^k$ *et* $\mathcal{C}$ avec $k \in \mathbb{N}^*$ le nombre de caractéristiques.

Pour $i \in \{1, \dots, I\}$, $N_i \in \mathbb{N}^*$, on pose $T_i = \{1, \dots, N_i\}$ l'ensemble des indices temporels associé à l'exemple i, $x^{(i)} = \left(x_t^{(i)}\right)_{t\in T_i}$ (série multivariée des *features*) et $y^{(i)} = \left(y_t^{(i)}\right)_{t\in T_i}$ (série univariée des labels) des réalisations « finies » de X et Y.

Pour simplifier l'écriture, on notera x comme une matrice $\left(x_{t,j}\right)_{1\le t\le N_i,\, 1\le j\le k} \in \mathcal{M}_{N_i,k}(\mathbb{R})$.

Les données peuvent ainsi être vues comme des tableaux :

Features →

Time ↓	x	Alt	Speed	…	N1
	1	$x_{1,1}$	$x_{1,2}$	…	$x_{1,k}$
	2	$x_{2,1}$	$x_{2,2}$	…	$x_{2,k}$
	…				
	N_i	$x_{N_i,1}$	$x_{N_i,2}$	…	$x_{N_i,k}$

Time ↓	y	phase
	1	y_1
	2	y_2
	…	
	N_i	y_{N_i}

Tableau 6.3 - Tableaux des features et labels

La fonction de coût associée à un ensemble $\{(\mathrm{x}^{(1)}, \mathrm{y}^{(1)}), \dots (\mathrm{x}^{(\mathrm{N})}, \mathrm{y}^{(\mathrm{N})})\}$, adaptée de la « *Pixel-Wise Cross-entropy* » couramment utilisée dans la segmentation d'image est donnée par la formule :

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\frac{-1}{T_i}\sum_{j=1}^{T_i}\log\sigma\left(g_j\left(x^{(i)}\right)\right)_{y_j^{(i)}}$$

Pour une certaine fonction $g : \mathcal{M}_{N_i,k}(\mathbb{R}) \to \mathcal{M}_{N_i,C}(\mathbb{R}),\quad x \mapsto \begin{pmatrix} g_1(x) \\ \vdots \\ g_{T_i}(x) \end{pmatrix}$ vérifiant :

$$\hat{y}^{(i)} = f\left(x^{(i)}\right) = \mathrm{indmax} \circ \sigma \circ g\left(x^{(i)}\right)$$

Prenons un exemple simplifié pour expliquer cette fonction « objectif ». On considère :

$C = \{1, \dots, 3\}$, $y^T = (1 \quad 1 \quad 2 \quad 3 \quad 2)$ et $\sigma \circ g(x) = \begin{pmatrix} 0{,}9 & 0{,}05 & 0{,}05 \\ 0{,}4 & 0{,}3 & 0{,}3 \\ 0{,}3 & 0{,}3 & 0{,}4 \\ 0{,}05 & 0{,}9 & 0{,}05 \\ 0{,}2 & 0{,}5 & 0{,}3 \end{pmatrix}$.

Dans ce cas de figure, la prédiction du réseau serait : $\hat{y}^T = (1 \quad 1 \quad 3 \quad 2 \quad 2)$

La fonction de coût associée à cet exemple est :

$$\begin{aligned}\mathcal{L} &= \frac{-1}{5}\sum_{j=1}^{5} log\ \sigma_{y_j}\left(g_j(x)\right) \\ &= -\frac{1}{5}\left(\log\sigma_1\left(g_1(x)\right) + \log\sigma_1\left(g_2(x)\right) + \log\sigma_2\left(g_3(x)\right) + \log\sigma_3\left(g_4(x)\right) + \log\sigma_2\left(g_5(x)\right)\right) \\ &= \frac{1}{5}(-\log(0{,}9) - \log(0{,}4) - \log(0{,}3) - \log(0{,}05) - \log(0{,}5)) = 0{,}51\end{aligned}$$

On a mis en évidence 4 cas de figures :

- Ligne 1 : le réseau prédit correctement sans aucune ambiguïté $(0{,}9 \gg 0{,}05)$: le réseau est sûr de lui, on le récompense avec une très petite valeur ($-\log(0{,}9)$ très proche de 0)
- Ligne 2 : le réseau prédit correctement mais de façon moins certaine $(0{,}4 \approx 0{,}3)$: il n'est pas aussi catégorique mais il a raison : on le récompense avec une petite valeur : ($-\log(0{,}4)$ assez proche de 0)
- Ligne 3 : le réseau se trompe mais de peu $(0{,}3 \approx 0{,}4)$: il est probablement en train d'apprendre la relation. On lui affecte une petite valeur : ($-\log(0{,}3)$ assez proche de 0)
- Ligne 4 : le réseau se trompe complètement : $(0{,}05 \ll 0{,}9)$: on lui affecte une valeur très grande ($-\log(0{,}05) \gg 0$)
- Ligne 5 : …

En résumé, la fonction « objectif » permet, non seulement de rendre compte des fois ou le réseau prédit correctement ou non, mais également de quantifier l'erreur ou l'incertitude du réseau. Cela permet d'actualiser adéquatement les poids lors de la phase de dite de « rétropropagation du gradient ».

6.2.6 Considérations sur la structure du réseau

Lorsque l'on définit la structure d'un réseau de neurones, il est presque tout le temps indispensable de spécifier les dimensions des différentes couches. Ces dimensions sont fixes et adaptées à la donnée d'entrée. Ainsi, dans l'exemple ci-dessous, le réseau pourra accepter en entrée des données d'un ensemble isomorphe à $\mathbb{R}^4$.

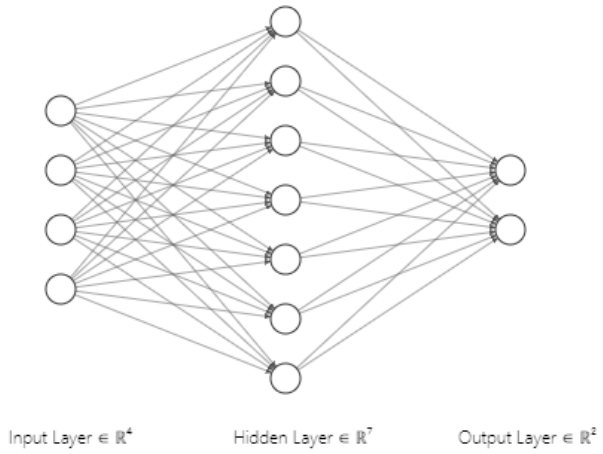

Figure 6.15 - Structure fixe du perceptron multicouche

Le problème qui se pose à nous est le suivant : les vols n'ont pas la même durée. Les données sont donc de tailles variables selon l'axe temporel. Il faut donc trouver un moyen pour utiliser le réseau *U-Net* sur des vols potentiellement plus longs ou plus courts que la taille d'entrée du réseau.

En réalité, *U-Net* est un réseau qui dispose d'une structure dite FCN (« *Fully Convolutionnal Networks* ») qui lui confère la propriété de pouvoir prédire des données en entrée de taille arbitraire. Plus précisément, l'absence de couches denses (*fully connected*) permet cette amplitude d'implémentation : au sein d'une couche de convolution, par structure, le nombre de paramètres à apprendre ne varie pas suivant la taille des données. Le nombre de paramètres est uniquement fonction de la taille et du nombre de filtres (fixés au départ). Ainsi, ce nombre reste constant, qu'on soumette une entrée de taille t ou de taille $t' \neq t$. Dans ce cas, seul le parcours du filtre sur la donnée devient plus long (ou plus court), mais le nombre de poids synaptiques à apprendre reste bien constant. Ceci n'est évidemment pas le cas dans les couches denses, car le nombre de poids est directement fonction du nombre de neurones, et donc de la dimension de la donnée.

Toutes considérations prises, nous disposons de deux solutions :

- Implémenter le *U-Net* à une taille fixe d et trouver un moyen pour l'exécuter sur des données de tailles potentiellement différentes,
- Exploiter la structure FCN pour directement prédire des vols de longueur arbitraire.

Nous détaillons ces deux pistes dans les parties suivantes.

6.2.7 1ère implémentation : U-Net à taille d'entrée fixe d

C'est la solution la plus simple du point de vue implémentation du réseau mais elle nécessite quand même de pouvoir s'entraîner et prédire des vols plus longs et plus courts que la taille d'entrée. Deux solutions s'offrent à nous :

1. Découper les vols « longs » et compléter les vols « courts »[1]
2. Echantillonner les vols

Découpage/complétion des vols :

Dans le cadre de vols « courts », on procède simplement en complétant le vol avec des valeurs nulles à la fin pour obtenir la longueur désirée. On parle de 0-*padding*.

Dans le cadre de vols « longs », on procède :

- En décomposant les vols en plusieurs parties de taille d dans le *dataset* d'entraînement,
- Pour une prédiction, on décompose le vol en plusieurs parties de taille d, on génère une prédiction pour chacune des parties, puis on recompose les prédictions.

La figure ci-dessous présente cette stratégie. On dispose d'un vol de longueur supérieure à d. On découpe ce vol en 4 parties (dont 3 de longueur d et une de longueur inférieure à d). Cette dernière est traitée comme un vol « court » (0-*padding*) puis envoyée au réseau. Les autres parties sont directement envoyées au réseau *U-Net*.

[1] Par vol « long » (resp. « court »), on entend de longueur $\geq$ d (resp. $\leq d$)

On dispose alors de 4 prédictions qui peuvent être re-fusionnées pour construire la prédiction du vol dans sa globalité.

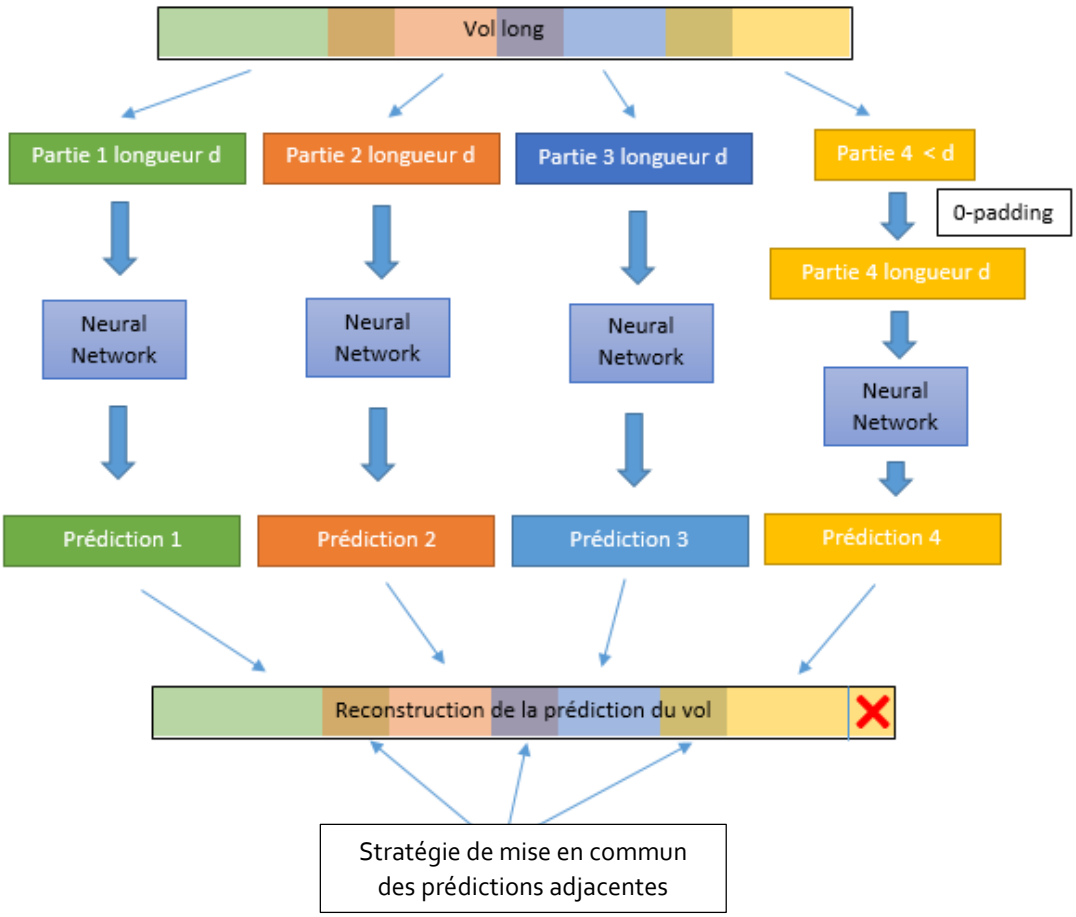

Figure 6.16 - Schéma de reconstruction d'une prédiction

Les parties sont sélectionnées de manière à ce qu'elles se chevauchent : la fin de la partie 1 est identique au début de la partie 2. Ce choix est motivé par le fait que les réseaux convolutifs ont tendance à souffrir des effets de bords : les prédictions sont moins bonnes aux extrémités. Pour pallier à cela, on a choisi de superposer légèrement les parties pour être sûr de toujours disposer de zones de prédictions fiables.

Par conséquent, lors de la reconstruction, on dispose de deux jeux de prédictions (correspondant aux parties qui se chevauchent) pour certaines zones. Il est alors nécessaire de procéder à une mise en commun de ces prédictions, ce que nous faisons de la façon suivante :

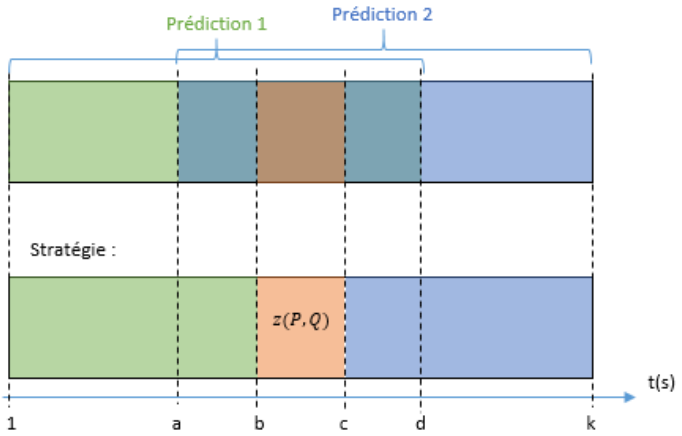

Figure 6.17 - Stratégie de reconstruction des prédictions

On note $\mathcal{C} = \{1, \dots, C\}$ l'ensemble des classes de notre problème.

Si l'on dispose des matrices de prédictions $P_1 \in \mathcal{M}_{C,d}$ et $P_2 \in \mathcal{M}_{C,k-a+1}$ telles que :

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,d} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ p_{C,1} & p_{C,2} & \cdots & p_{C,d} \end{pmatrix} \text{ et } Q = \begin{pmatrix} q_{1,a} & q_{1,a+1} & \cdots & q_{1,k} \\ q_{2,a} & q_{2,a+1} & \cdots & q_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ q_{C,a} & q_{C,a+1} & \cdots & q_{C,k} \end{pmatrix}$$

Avec

- $p_{i,j} \in [0,1]$ la probabilité d'appartenance du vol à la classe i à l'instant j selon la prédiction 1
- $q_{i,j} \in [0,1]$ la probabilité d'appartenance du vol à la classe i à l'instant j selon la prédiction 2

$$Z(P,Q) = \begin{pmatrix} z_{1,b} & z_{1,b+1} & \cdots & z_{1,c} \\ z_{2,b} & z_{2,b+1} & \cdots & z_{2,c} \\ \vdots & \vdots & \vdots & \vdots \\ z_{C,b} & z_{C,b+1} & \cdots & z_{C,c} \end{pmatrix} \in \mathcal{M}_{C,c-b+1} \quad \text{avec } z_{i,j} = \frac{p_{i,j}+q_{i,j}}{2}$$

On dispose ainsi d'une matrice de prédiction pour le vol :

$$V = (v_{i,j})_{1 \le i \le C, 1 \le j \le k} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,b-1} & z_{1,b} & \cdots & z_{1,c} & q_{1,c+1} & \cdots & q_{1,k} \\ p_{2,1} & \cdots & p_{2,b-1} & z_{2,b} & \cdots & z_{2,c} & q_{2,c+1} & \cdots & q_{2,k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{C,1} & \cdots & p_{C,b-1} & z_{C,b} & \cdots & z_{C,c} & q_{C,c+1} & \cdots & q_{C,k} \end{pmatrix} \in \mathcal{M}_{C,k}$$

Il s'ensuit que la prédiction du vol est donnée par :

$$\hat{y} = (\hat{y}_1, \dots, \hat{y}_k) \text{ avec } \hat{y}_i = \text{indmax}(v_{1,i}, v_{2,i}, \dots, v_{C,i})$$

Avec $\text{indmax}(p_1, p_2, \dots p_n) = \max\{\, i \in \{1, n\},\ p_i = \max(p_1, \dots, p_n)\}$ (renvoie la classe associée à la probabilité la plus haute)

Inconvénients de la méthode découpage/complétion :

- Cette méthode demande beaucoup de prétraitements pour prendre en compte des vols longs,
- Le fait de *padder* les vols courts à 0 dans le *dataset* d'entraînement augmente indirectement l'*accuracy* : la surreprésentation induite associée à la facilité du réseau à apprendre des exemples redondants entraîne une surestimation de l'efficacité du réseau.

Échantillonnage des vols :

Une solution plus simple à mettre en œuvre consiste à échantillonner les vols longs et suréchantillonner les vols courts de façon à normaliser leur longueur. Par exemple, pour un vol long donné par la série $(v_t)_{1\leq t\leq d_v}$ $(d_v > d)$, on peut extraire un vol de longueur d :

$$(v_t)_{t\in I_d} \text{ avec } I_d = \left\{\left\lfloor i\frac{d_v}{d}\right\rfloor, i \in \{1, \dots, d\}\right\}$$

On garde uniquement d points régulièrement espacés dans $\{1, \dots, d_v\}$.

De façon à améliorer la qualité de l'échantillonnage, on peut se focaliser sur une fenêtre de taille f autour d'un point que l'on garde : au lieu de considérer la valeur de la série en ce point, on considère la moyenne des points dans cette fenêtre.

$$v' = (v'_t)_{t\in I_d} \text{ avec : } \forall t \in I_d, v'_t = \frac{1}{f}\sum_{i=-f/2}^{f/2} v_i$$

Cela permet d'avoir une meilleure approximation locale des valeurs que l'on conserve. Il faut choisir une fenêtre suffisamment petite pour ne pas fausser le caractère local des valeurs retenues.

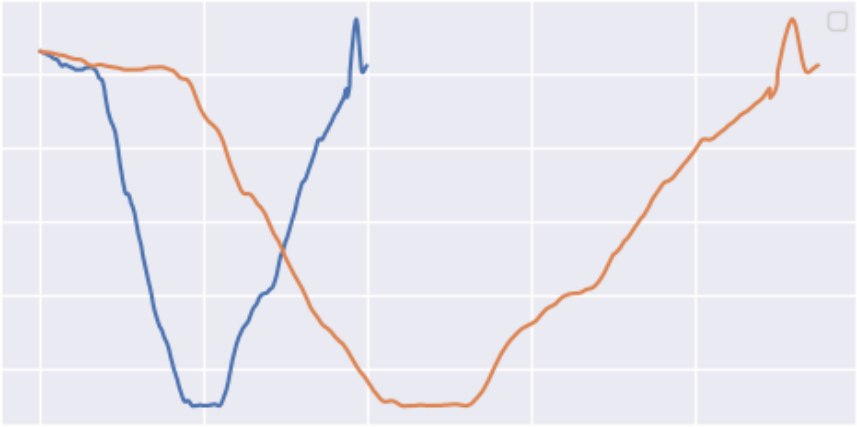

Figure 6.18 - Échantillonnage d'un vol long

On peut voir sur la figure précédente, la température au cours du temps pour un vol initial (orange) et le vol échantillonné correspondant (bleu). Le signal est simplement « compressé » mais garde sa forme initiale.

De manière analogue, on peut procéder au suréchantillonnage des vols courts en rajoutant des points régulièrement espacés dans le temps (dont la valeur peut être la moyenne des points dans le voisinage par exemple) et ce, de façon à atteindre un vol de longueur d.

Inconvénient de la méthode échantillonnage/suréchantillonnage :

- L'échantillonnage des vols est associé à une perte d'information : en échantillonnant nos signaux, on est susceptible de perdre des instants qui pourraient être discriminants pour le réseau de neurones. En particulier dans le cas où l'on dispose de vols de longueur très supérieure à d, le signal échantillonné est susceptible de perdre sa forme originale et donc de fausser l'apprentissage.

6.2.8 2ème implémentation : U-Net à taille d'entrée variable

Comme nous l'avons vu précédemment, il est possible d'implémenter *U-net* avec une taille d'entrée variable. De par sa structure FCN, il est tout à fait possible pour le réseau de traiter des vols de tailles variables.

Cela nécessite cependant quelques prérequis :

- La dimension temporelle des données à l'entrée des couches de *pooling* doit être divisible par 2. En effet, chaque couche de *pooling* fait la division entière de la dimension temporelle de l'entrée par 2, et inversement chaque couche d'*upsampling* de la voie expansive multiplie cette dimension par 2. Si jamais l'hypothèse de divisibilité par 2 n'est pas satisfaite, on risque de se retrouver, pour un même niveau de profondeur, avec des dimensions légèrement différentes entre la branche contractante et la branche expansive du réseau, rendant impossible la concaténation interphase,
- Si l'on procède à un entraînement *mini-batch* (regroupement des exemples en plusieurs paquets durant l'apprentissage) : chaque exemple d'un même *batch* doit avoir la même dimension temporelle. Il s'agit ici d'une restriction de la librairie *Tensorflow.*

Pour répondre à cette problématique, il est nécessaire de procéder à un 0-*padding* des vols.

On peut répondre à la première contrainte de deux manières :

- Soit, on *pad* la donnée avant chaque couche de *pooling* de façon à ce qu'elle soit bien divisible par 2,
- Soit, on *pad* le vol une seule et unique fois à l'entrée du réseau de façon à ce que la dimension temporelle soit divisible par 2^p avec p le nombre de couches de *pooling* dans le réseau.

Enfin, concernant les *batchs*, il suffit de constituer des groupes de vols de dimensions semblables et de *padder* les vols au sein de chaque *batch* de façon à ce qu'ils soient de même taille, tout en respectant la première contrainte.

Illustrons cela sur un exemple :

Pour une taille de batch 3 avec 3 couches de *pooling* dans le réseau, la dimension de la donnée en entrée doit être divisible par $2^3 = 8$. Pour chaque vol, on cherche donc k tel que :

$$d_v + k \equiv 0 \, mod \, 8$$

Puis on pad les vols avec k zéros comme schématisé dans la figure suivante.

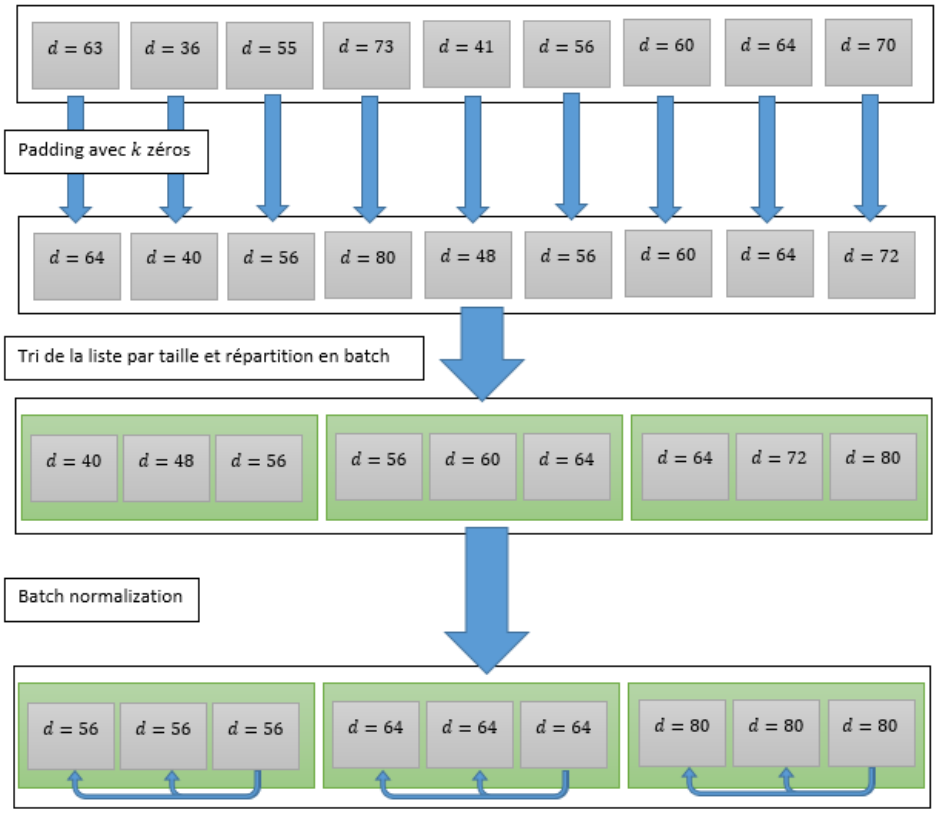

Figure 6.19 - Exemple de la procédure de padding des vols

6.2.9 Résultats

La solution retenue est celle du réseau U-net à taille d'entrée variable. Les tests ont été effectués sur un *dataset* de 1024 vols répartis comme suit :

- *Train dataset* : 924 vols
- *Test dataset* : 100 vols

L'apprentissage a été implémenté en « *mini-batch gradient descent* » avec une taille de *batch* de 32.

Les résultats obtenus sont très satisfaisants en termes de performances comme le montre le tableau ci-dessous :

Test dataset accuracy (Acc)	Test dataset Per Class Accuracy (Pca)
0.976	0.947

Tableau 6.4 - Résultats de la seconde implémentation

Dans l'exemple suivant, on dispose de deux graphiques :

- Le premier graphique présente les segmentations : la partie supérieure est la segmentation réelle du vol, la partie inférieure est la segmentation prédite par le réseau

- Le second graphique montre les courbes de confiance associées aux différentes classes au cours du vol. Elles caractérisent la certitude du réseau sur la prédiction de chacune des phases.

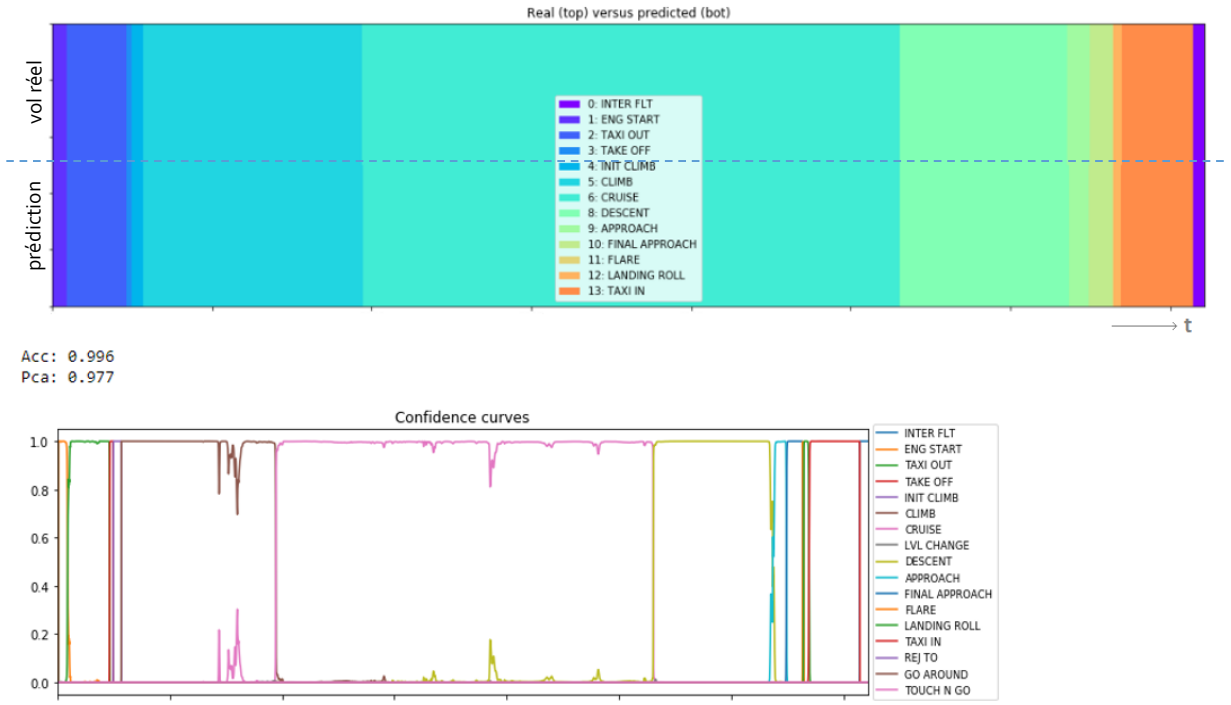

Figure 6.20 - Exemple de segmentation d'un vol

Un des résultats notables est la capacité du réseau à prédire des profils spécifiques non détectés dans la segmentation réelle. On peut voir, par exemple, dans la figure ci-dessous que l'avion a effectué un palier croisière au cours de sa montée. Ce palier est détecté par *U-net* qui a assigné à cette portion de vol le label « Cruise » :

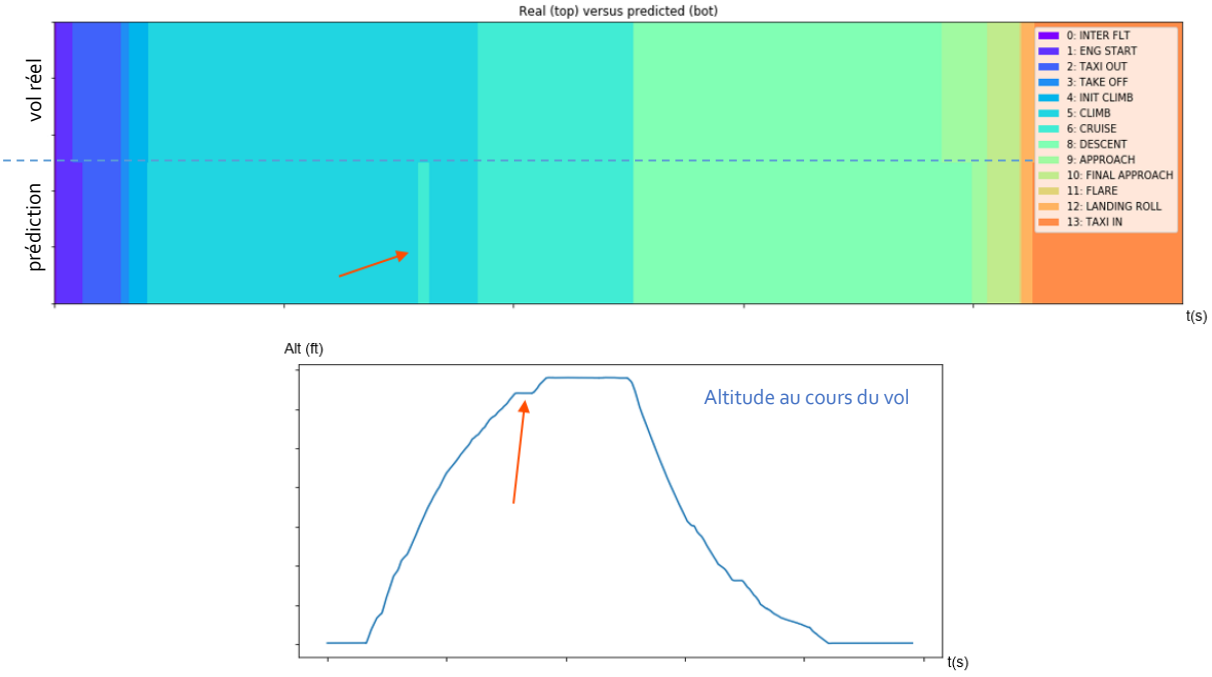

Figure 6.21 - Détection de profils absents dans le masque réel

7 AMELIORATIONS ET AUTRES AXES DE RECHERCHE

Dans les parties précédentes, nous avons vu comment mettre en place un réseau de neurones convolutif capable d'effectuer une segmentation d'un vol en différentes phases distinctes. Au sein de chacun de ces segments, il est encore possible de distinguer des schémas atypiques pouvant eux-mêmes être classés en plusieurs groupes. On peut alors renforcer la description de chacune des phases en se basant sur certains critères spécifiques. Par exemple : distinguer les approches rapides des approches lentes.

La première tentative consistait à réaliser un « *clustering* » des phases de montée. Le but était de soumettre un ensemble de séries temporelles à un algorithme de partitionnement des données appelé *k-means* (à travers la librairie Python *TSLearn*) pour que ce dernier essaie de regrouper les courbes par groupes de similarités. L'idéal aurait été de trouver des schémas de montées de formes variées pour ensuite pouvoir en tirer de l'information.

Après plusieurs tentatives, nous avons obtenu des résultats peu satisfaisants. En effet, les *clusters* trouvés par l'algorithme *k-means* étaient trop dépendants de l'altitude finale des montées plutôt que de la morphologie des trajectoires comme on peut le voir ci-contre avec $k = 2$ *clusters*.

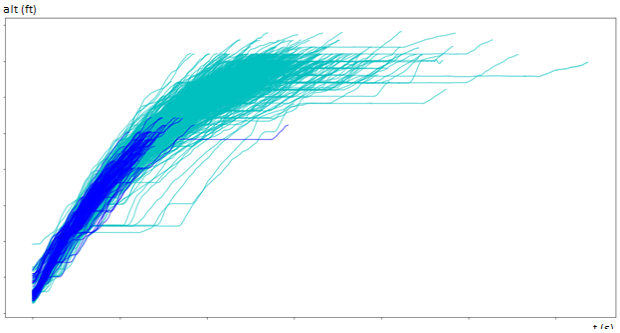

Figure 7.1 - Exemple de clustering des montées pour k=2

Nous avons alors décidé de réaliser un partitionnement des montées en fonction d'un critère spécifique : les paliers. Lors de certaines montées, les avions se stabilisent momentanément en vol horizontal (pour diverses raisons) avant de regagner le régime de montée. Nous avons ainsi implémenté un algorithme qui détecte et recense ces paliers au sein des montées.

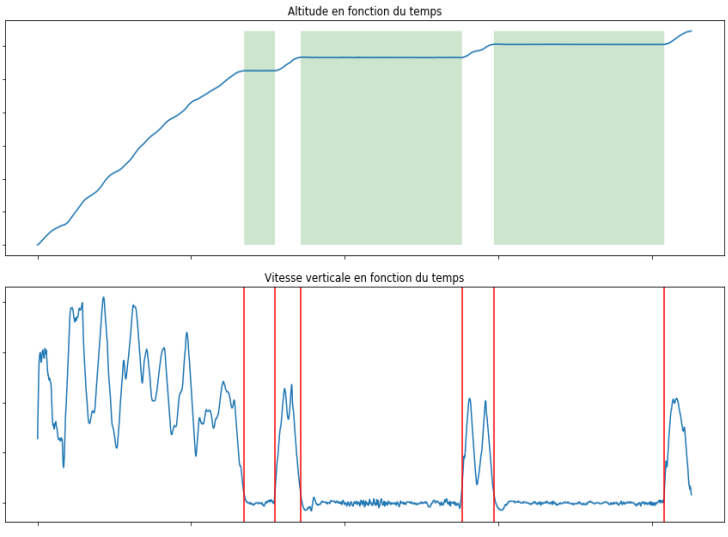

Figure 7.2 - Détection des paliers dans les montées

Cela nous permet plusieurs choses : dans un premier temps, on obtient des *clusters* interprétables en regroupant les différentes montées par catégories distinctes en fonction du nombre de paliers (voir figures Figure 7.3 et Figure 7.4)

Nous pouvons également inférer des résultats statistiques, en établissant, par exemple, un intervalle de confiance sur le nombre de montées avec n paliers dans la population à partir de notre échantillon.

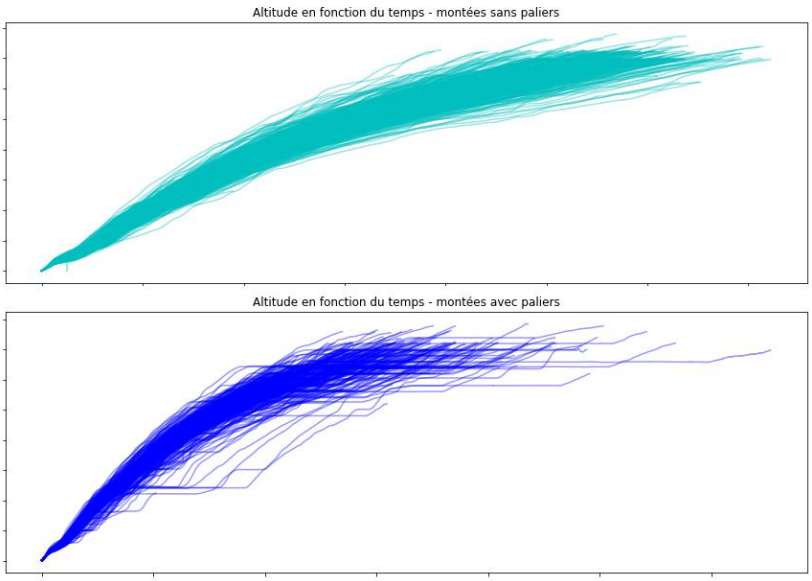

Figure 7.3 - Clustering des montées en fonction de la présence de paliers

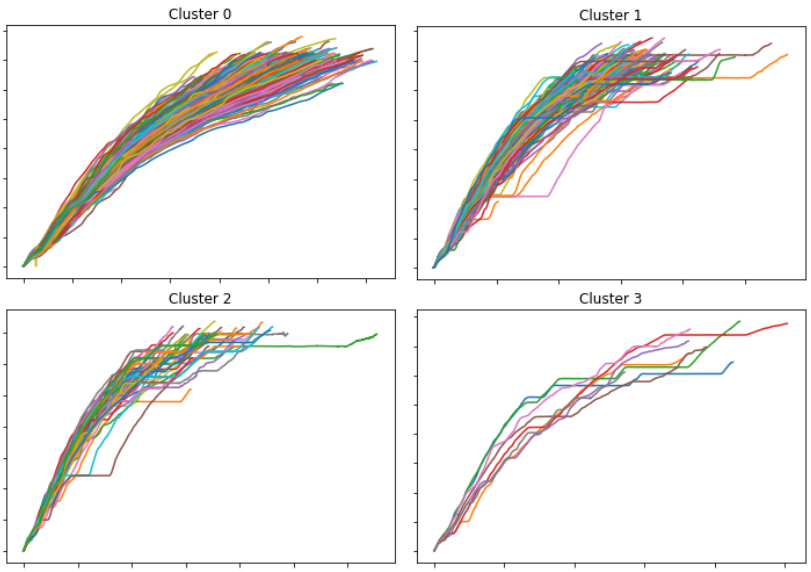

Figure 7.4 - Clustering des montées en fonction du nombre de paliers

Plus spécifiquement, nous pouvons obtenir des résultats sur chaque *cluster* individuellement, et même sur chaque palier au sein des vols d'un même *cluster*. Par exemple, déterminer l'altitude moyenne du 2ème palier dans les vols disposant de 3 paliers, ou encore la durée moyenne des paliers dans les vols disposant de 2 paliers…

Tous ces résultats peuvent être adaptés aux phases de descentes et d'approches.

Les travaux en cours pendant la fin de ce stage concernent l'étude de la consommation de carburant des avions dans les cas de figure énoncés plus haut. En estimant, par exemple, la consommation de carburant sur les phases de montée, on pourra inférer des résultats et à terme, définir si les montées avec paliers doivent être privilégiées ou au contraire proscrites quand cela est possible.

8 ANALYSE DU STAGE

Ce stage réalisé du 7 octobre 2019 au 3 avril 2020 a été une expérience fructueuse sur le plan personnel comme professionnel. D'abord, puisqu'il constitue ma première implication réelle dans le monde professionnel et qu'il m'a permis de connaître en détail le quotidien des ingénieurs dans le domaine de l'analyse de données. Cette partie a pour but de souligner les différents apports que j'ai tirés de cette expérience.

Tout d'abord sur le plan technique, ce stage m'a appris, à mettre en application les connaissances scientifiques que j'ai pu acquérir durant ma scolarité en école d'ingénieur et pendant ma mobilité au Canada, mais plus important encore, de progresser et d'apprendre énormément dans certains domaines ciblés. Je pense en particulier au *Machine Learning*, domaine que j'ai pu découvrir à l'UQAC. J'ai ainsi pu appliquer mes connaissances en implémentant certains résultats théoriques de manière concrète, mais également et surtout de les enrichir en découvrant et en approfondissant plusieurs aspects : architectures des réseaux CNN pour la segmentation, rebalancement des jeux de données, apprentissage par descente de gradient stochastique en *mini-batch*.... D'autres domaines qui m'ont été donnés d'appliquer sont les probabilités et statistiques, ainsi que le traitement du signal dont j'ai suivi les cours en 1ere et 2eme année à l'ENSEM. Cet enseignement m'a été très bénéfique et j'ai pu implémenter les différentes notions de manière concrète sur les données que j'avais à disposition : échantillonnage, étude des corrélations...

Ce stage m'a également permis de voir et d'appréhender le fonctionnement d'une société d'envergure internationale comme Safran. J'ai ainsi pu voir l'articulation entre les différents services, les différentes organisations, les flux de communication et l'implication de chaque acteur dans le bon fonctionnement de la société. La coordination d'un grand nombre de personnes comme à Safran nécessite une organisation « ficelée » et une structure bien définie dont j'ai pu être témoin durant mon stage. En ce qui concerne les communications inter et intra service, j'ai pu assister à des réunions visant à présenter à plusieurs services des projets entrepris par différentes entités, de façon à ce que chaque acteur puisse être au fait des activités des autres, ainsi qu'à l'échelle du service, à des réunions hebdomadaires où chaque personne est amenée à présenter les travaux effectués au cours de la semaine.

Finalement, au-delà des aspects techniques et organisationnels, ce stage m'a éclairé sur les relations humaines au sein de la société. L'atmosphère au sein de mon service était chaleureuse et les relations entre les employés à la fois conviviales et professionnelles, ce qui à mon avis est un vecteur de performance non négligeable dans une société.

9 CONCLUSION

L'objectif de ce stage était de tester l'implémentation et l'efficacité des réseaux convolutifs pour l'analyse de séries temporelles multivariées. Nous avons vu qu'il est possible d'obtenir une segmentation de vol de manière rapide et robuste. En amont de ces travaux, le nettoyage et le prétraitement des données est une étape qui semble indispensable pour exploiter ces dernières et obtenir des résultats satisfaisants. Enfin, différents axes nous permettent de renforcer l'information associée à une phase de vol spécifique comme vu dans la partie « Amélioration et autres axes de recherche ».

Sur le plan personnel, ce stage a été une expérience très enrichissante pour moi. J'ai pu découvrir le monde du travail dans l'ingénierie et plus spécifiquement, l'organisation de la société d'envergure internationale qu'est Safran. J'ai eu l'opportunité de mettre œuvre les connaissances acquises lors de mon *cursus* scolaire, d'élargir et de développer des compétences dans plusieurs domaines, et de m'initier à la vie en entreprise. Dans ce cadre, j'ai pu découvrir en détail le monde de l'analyse de données dans le contexte de l'aviation et de travailler sur un sujet passionnant alliant ces deux domaines. Ce stage me conforte ainsi dans mon projet professionnel et me donne pleinement envie de poursuivre dans cette voie.

A ANNEXES

TABLEAUX RECAPITULATIFS DES PARAMETRES ET PHASES DE VOLS

Paramètre	Signification	Unité	Représentation	Notes
FLIGHT_PHASE	Phase du vol	*qual*[2]	Entier ∈ $\{0, \dots, 16\}$	Voir tableau B
RALTC	Radio-altitude	ft[3]	Entier naturel	Hauteur mesurée par un appareil radio quand l'avion est à proximité du sol
HEIGHT	Hauteur sol	ft	Entier naturel	Hauteur de l'aéronef par rapport au sol
IAS_C	Vitesse indiquée *(Indicated air speed)*	kt[4]	Entier naturel	Vitesse indiquée par l'instrument de mesure anémobarométrique
IVV_R	Vitesse verticale	ft/min	Entier relatif	Vitesse indiquée par le variomètre
PITCH_C	Assiette (Pitch angle)	degré	Réel	Angle formé par l'horizon et l'axe longitudinal de l'avion
ROLL	Inclinaison (Roll angle)	degré	Réel	Angle formé par l'horizon et l'axe transversal de l'avion
TAS	Vitesse vraie *(True air speed)*	kt	Réel positif	Vitesse de l'aéronef par rapport à la masse d'air environnant
GS_C	Vitesse sol (Ground speed)	kt	Réel positif	Vitesse de l'aéronef par rapport au sol
HEAD_TRUE	Cap magnétique	degré	Réel entre 0 et 360	Cap suivit par l'aéronef
AOA	Incidence (angle of attack)	degré	Réel	Angle formé par l'axe longitudinal de l'aéronef et le vecteur vitesse du vent relatif (trajectoire)
MACH_R	Nombre de mach	s.u	Réel entre 0 et 1 (vol subsonique)	Rapport entre la vitesse propre de l'aéronef et la célérité du son à la température considérée
FLT_PATH_ANGLE	Pente	degré	Réel	Angle formé par le vecteur vitesse du vent relatif (trajectoire) et l'horizon
WIN_DIR	Direction du vent (Wind direction)	degré	Réel entre 0 et 360	
WIN_SPDR	Vitesse du vent	kt	Entier naturel	
TAT	Température air (Total air temperature)	°C	Réel	Température mesurée aux abords de l'avion
SAT_C	Static air temperature	°C	Réel	Température mesurée par un thermomètre immobile par rapport à la masse d'air.
ALT_STD_C	Altitude standard	ft	Entier relatif	Altitude indiquée à la pression standard (1013.25 hpa)
AIE1_RW	Anti ice moteur 1	*qual*	Binaire	Système de dégivrage moteur 1
AIE2_RW	Anti ice moteur 2	*qual*	Binaire	Système de dégivrage moteur 2
ENG1_BLD		*qual*	Binaire	
FOB	Fuel on board	kg	Entier naturel	Masse de carburant à bord
ZERO_FU_W	Zero fuel weight	kg	Entier naturel	Masse de l'avion sans carburant
GWC	Gross Weight	kg	Entier naturel	Masse totale de l'avion
CONF	Configuration hypersustentateurs	*qual*	Entier ∈ $\{0, \dots, 5\}$	Voir tableau ci-dessous

[2] Variable qualitative
[3] 1 pied (ft) = 0,3048 m
[4] 1 nœud (kt) = 1,852 km/h

TOUCH_DOWN_CONF	Configuration atterrissage	*qual*	Binaire	
LDG_LOK_DW	Landing gear lock down	*qual*	Binaire	Train verrouillé et sorti
SPD_BRK_DEV	Speed break angle	degré	Réel	Angles aérofrein
SPD_BRK	Spead break	*qual*	Binaire	Déploiement aérofrein
SPOIL_LOUT1	Spoiler out	*qual*	Binaire	Déploiement destructeur de portance
SSP_POINT		*qual*		
DATE_HF	Date	*qual*	jj/mm/aaaaa	
TIME_HF	Heure	*qual*	hh :mm :ss	
FF1_C	Fuel flow motor 1	kg/h		Débit carburant moteur 1
FF2_C	Fuel flow motor 2	kg/h		Débit carburant moteur 2
N12_C	Régime corps basse pression moteur 1		%	
N11_C	Régime corps basse pression moteur 2		%	
N21_C	Régime corps haute pression moteur 1		%	
N22_C	Régime corps haute pression moteur 2		%	

Tableau A.1 - Récapitulatif des paramètres des fichiers QAR

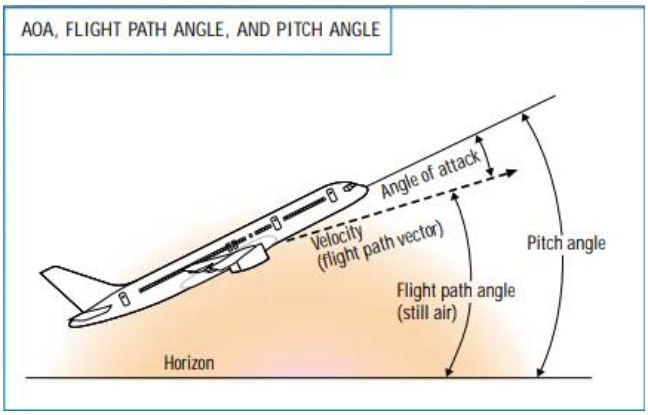

Figure A.1 - Les différents angles caractéristiques en aviation

N°	Phase de vol	TRADUCTION	Info / description icao
0	**INTER FLT**	Pré-vol	
1	**ENG START**	Démarrage des moteurs	
2	**TAXI OUT**	Roulage vers piste	Commences when the aircraft begins to move under its own power leaving the gate, ramp, apron, or parking area, and terminates upon reaching the runway.
3	**TAKE OFF**	Décollage	From the application of takeoff power, through rotation and to an altitude of 35 feet above runway elevation.
4	**INIT CLIMB**	Montée initiale	From the end of the Takeoff sub phase to the first prescribed power reduction, or until reaching 1000 feet above runway elevation or the VFR pattern, whichever comes first.
5	**CLIMB**	Montée	From completion of Initial Climb to arrival at initial assigned cruise altitude.
6	**CRUISE**	Croisière	Any level flight segment after arrival at initial cruise altitude until the start of descent to the destination.
7	**LVL CHANGE**	Changement de niveau de vol	Flight level change FL definition : A measure of altitude (in hundreds of feet) used by aircraft flying above 18,000 feet with the altimeter set at 29.92 "Hg.
8	**DESCENT**	Descente	Descent from cruise to either initial approach fix or VFR pattern entry.
9	**APPROACH**	Approche	From the initial approach fix to the beginning of the landing flare.
10	**FINAL APPROACH**	Approche finale	From the final approach fix to the beginning of the landing flare
11	**FLARE**	Arrondi	Transition from nose low to nose up attitude just before landing until touchdown.
12	**LANDING ROLL**	Roulage vers taxiway	After touchdown until aircraft exits the landing runway or comes to a stop, whichever occurs first
13	**TAXI IN**	Roulage vers parking	Begins upon exiting the landing runway and terminates upon arrival at the gate, ramp, apron, or parking area, when the aircraft ceases to move under its own power
14	**REJ TO**	Interruption décollage	During takeoff, from the point where the decision to abort has been taken until the aircraft begins to taxi from the runway.
15	**GO AROUND**	Interruption atterrissage	An event where an aircraft has begun its decent for landing, discontinues that decent, and does not land from that approach.
16	**TOUCH N GO**	Posé-décollé	Landing on a runway and taking off again without coming to a full stop.

Tableau A.2 – Identification des phases de vols

Source définitions : https://www.ntsb.gov/investigations/data/Documents/datafiles/PhaseofFlightDefinitions.pdf

SMOTE : TECHNIQUE DE SURECHANTILLONNAGE PAR SYNTHETISATION D'INSTANCES MINORITAIRES

Derrière ce grand terme se cache une idée très simple, pouvoir entraîner correctement un réseau de neurones sur un jeu de données non-balancé. Mais avant de rentrer dans le vif du sujet, considérons l'exemple suivant.

On dispose d'un jeu de données de pièces en sortie de chaîne production. Chaque individu est représenté par des caractéristiques X_1 et X_2 et un état E correspondant à la conformité de la pièce.

On souhaite réaliser un réseau qui classifie une nouvelle pièce comme étant défectueuse ou non selon les données de X_1 et X_2

Le jeu de données d'entrainement est le suivant :

X_1	X_2	E
4.9	3.0	Ko
4.4	2.9	Ko
5.7	4.4	Ko
4.6	3.6	Ko
7.0	3.2	Ok
6.4	3.2	Ok
6.9	3.1	Ok
5.5	2.3	Ok
6.5	2.8	Ok
5.7	2.8	Ok
6.3	3.3	Ok
4.9	2.4	Ok
6.6	2.9	Ok
5.2	2.7	Ok
5.0	2.0	Ok
5.9	3.0	Ok
6.0	2.2	Ok

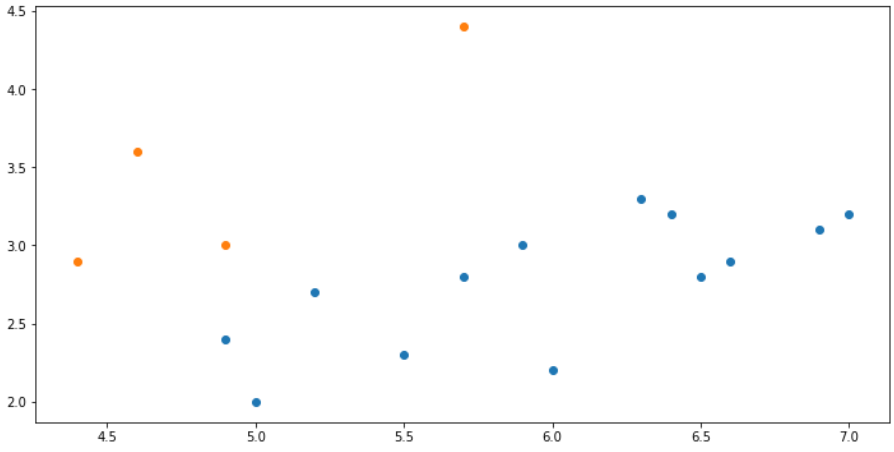

Figure A.2 – SMOTE - Données issues de http://rikunert.com/SMOTE_explained

Constat : Ce jeu de données est non-balancé : la classe Ko est largement en minorité par rapport à la classe Ok :

- Ko : 24%
- Ok : 76%

Problème sous-jacent : voyant que la grande majorité des observations appartient à la même catégorie, le modèle risque de toujours prédire la classe dominante. Or ici, c'est la classe minoritaire qu'il semble pertinent de détecter.

Dans l'optique où l'on cherche à prédire correctement la présence de pièces défectueuses, on préfère minimiser le risque statistique de seconde espèce (test puissant).

Deux solutions s'offrent alors pour rebalancer le jeu de données : augmenter le nombre de représentants des classes minoritaires (*over-sampling*) ou diminuer le nombre de représentants des classes majoritaires (*under-sampling*) :

Under-sampling :

- Retire aléatoirement des observations dans les classes majoritaires,
- Perte d'informations : susceptible de supprimer des observations discriminantes pour l'apprentissage -> augmentation du biais.

Over-sampling :

- Ajouter de nouvelles observations par réplication dans les classes minoritaires,
- Pas de perte d'information mais risque de redondance de l'information menant à du sur-apprentissage.

Méthode hybride : combiner les avantages des deux techniques précédentes

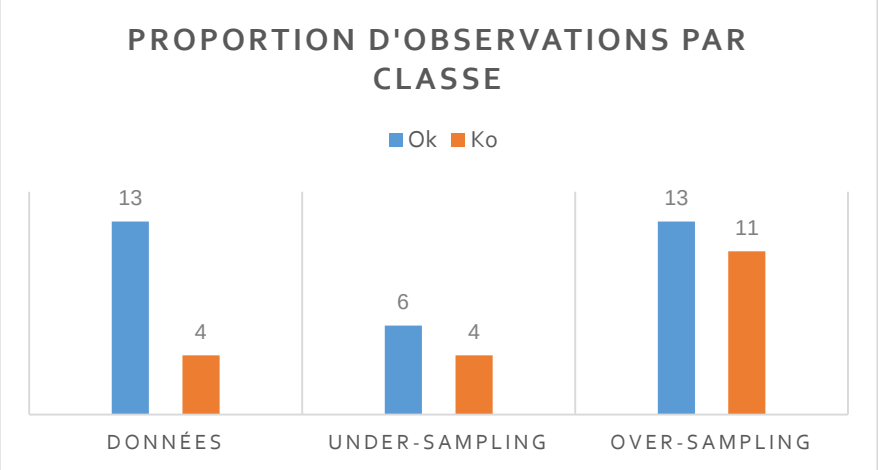

Figure A.3 - SMOTE - Proportion d'observations par classe

C'est ici que SMOTE entre en jeu : SMOTE = *Synthetic Minority Oversampling Technique* : Technique d'*Over-Sampling* qui permet de créer de nouvelles **observations synthétiques** (artificielles) inédites à partir des **observations réelles** de la classe minoritaire.

- Fonctionnement de SMOTE :

On se place dans l'espace des caractéristiques : chaque donnée peut alors être vue comme un point de cet espace. SMOTE fonctionne de la manière suivante :

1. Sélectionne une **observation réelle** A_i de la classe minoritaire
2. Trouve les k plus proches voisins (réels) de cette observation (au sens de la norme euclidienne dans l'espace des caractéristiques) et choisit aléatoirement un de ces k voisins (A_j)
3. Créé un nouveau point aléatoire sur le segment joignant A_i et A_j (Une **nouvelle observation synthétique** est créée)
4. On retourne à l'étape 1. (Répète les opérations précédentes pour chaque point de données (réel) de la classe minoritaire).

Une fois que toutes les observations réelles ont été choisies à l'étape 1, on réitère l'algorithme une nouvelle fois, et ainsi de suite jusqu'à ce que le jeu de données soit balancé.

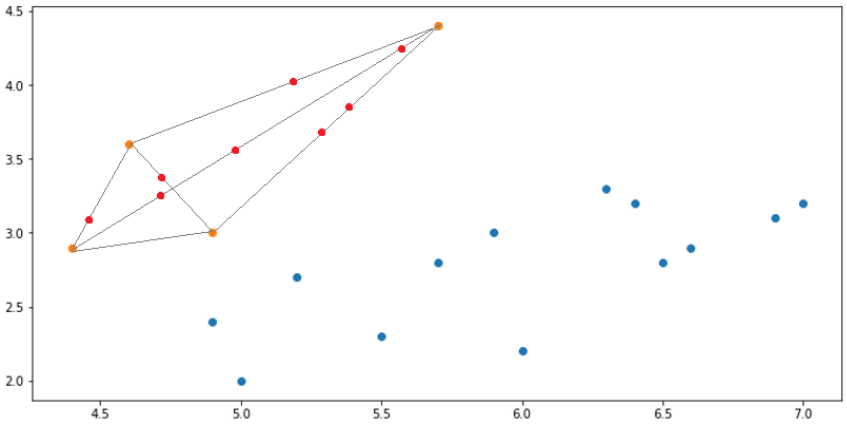

Figure A.4 - SMOTE - Résultat de l'oversampling

Notes :

- Les observations synthétiques ne sont pas considérées, ni dans le choix des k voisins, ni dans le choix des points sélectionnés à l'étape 2
- SMOTE sélectionne les k plus proches voisins uniquement dans la **classe minoritaire.** Sinon, on risquerait d'augmenter le chevauchement des classes : bruit supplémentaire
- On a vu ici, le cas de la classification binaire mais SMOTE s'applique aussi dans la classification multi-classes. Dans ce cas, l'algorithme peut s'effectuer suivant différentes stratégies :
    - *minority* : rééchantillonne uniquement la classe minoritaire;
    - *not-minority* : rééchantillonne toute les classes sauf la classe minoritaire;
    - *not-majority* : rééchantillonne toutes les classe sauf la classe majoritaire (défaut)
    - *all* : rééchantillone toutes les classes

B REFERENCES

Safran. *Présentation institutionnelle groupe Safran* [en ligne]. [Consulté le 18 mars 2019]. Disponible sur : https://insite.collab.group.safran/Pages/default.aspx

Safran Aircraft Engines. *Présentation Safran Aircraft Engines* [en ligne]. [Consulté le 18 mars 2019]. Disponible sur : https://insite.collab.group.safran/Pages/default.aspx

Wikipédia. 18/03/2019. *Safran Aircraft Engines* [en ligne]. [Consulté le 19 mars 2019]. Disponible sur : https://fr.wikipedia.org/wiki/Safran_Aircraft_Engines

Airbus. 2019. *A400M* [en ligne]. [Consulté le 20 mars 2019]. Disponible sur : https://www.airbus.com/defence/a400m.html

Safran Aircraft Engines. 2019. *Villaroche* [en ligne]. [Consulté le 20 mars 2019]. Disponible sur : https://www.safran-aircraft-engines.com/fr/societe/villaroche

Nitesh V. Chawla et al. *SMOTE: Synthetic Minority Over-sampling Technique* [en ligne]. [Consulté le 01/12/2019]. Disponible sur : https://arxiv.org/pdf/1106.1813.pdf

Jason Brownlee. *How to Develop 1D Convolutional Neural Network Models for Human Activity Recognition* [en ligne]. [Consulté le 01/12/2019]. Disponible sur : https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation* [en ligne]. [Consulté le 01/12/2019]. Disponible sur : https://arxiv.org/pdf/1505.04597.pdf

Nils Ackermann. *Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences* [en ligne]. [Consulté le 01/12/2019]. Disponible sur : https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf

Wikipédia. 30/12/2019. *Tensorflow* [en ligne]. [Consulté le 30/01/2020]. Disponible sur : https://fr.wikipedia.org/wiki/TensorFlow

Wikipédia. 20/11/2019. *Pandas* [en ligne]. [Consulté le 30/01/2020]. Disponible sur : https://fr.wikipedia.org/wiki/Pandas

Goodfellow et al. Deep Learning [en ligne]. [Consulté le 05/12/2019].Disponible sur : https://www.deeplearningbook.org/

Jonathan Long et al. Fully Convolutional Networks for Semantic Segmentation [en ligne]. [Consulté le 05/12/2019]. Disponible sur : https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf