Modélisation de l'usure de moteurs d'avions

Article réalisé dans le cadre du cours de modélisation statistique de l'ENSAE en partenariat avec l'entreprise Safran

Louise Blart, Vinciane Desbois

Mars-Mai 2021

Table des matières

Résumé

Les séries chronologiques multivariées sont présentes dans de nombreux secteurs et peuvent profondemment impacter l'économie. Dans ce document, on s'intéresse à la problèmatique du groupe industriel Safran, à savoir de prédire les pentes de la marge EGT (indicateur de l'usure des moteurs d'avions) à partir des catactéristiques du moteur et de l'environnement dans lequel il vole.

De manière générale, le traitement non-paramétrique des séries temporelles se fait à l'aide de Réseaux de Neurones de type Récurrents. Ils ont l'inconvénient de nécessiter un temps d'entrainement conséquent. Dans ce papier nous traitons ce problème à l'aide de Réseaux de Neurones Convolutifs (CNN), connus pour leur capacité à détecter les caractéristiques locales des séries.

Nous présentons trois modèles: un CNN simple, dont les prédictions sont assez stables mais proches de la moyenne, un CNN-LSTM, dont les prédictions sont pertinentes mais ne permettent pas de détecter les valeurs extrêmes, et un CNN à plusieurs branches, dont l'input est décomposé en Tendance, Saisonnalité, Bruit et Variables catégorielles. Ce dernier modèle est basé sur des hypothèses de périodicité fortes. Il est instable en fonction des échantillons proposés, mais permet d'agrandir le champ des prédictions.

1 Introduction

Une série temporelle ou série chronologique est une séquence X_t d'observations de la même quantité indexée par une date $t \in T$ (où T est un ensemble discret de dates). L'objectif principal de l'analyse d'une série temporelle est la **prévision de ses réalisations futures** en se basant sur les valeurs passées. Les données temporelles (ex : encombrement du transport routier) sont différentes des autres types de données car elles ont des caractéristiques très particulières à prendre en compte :

- Les éléments de proximité (ex : bouchons, pluie)
- La périodicité locale de la séquence (ex : sur 24 heures – heures de pointe)
- La périodicité globale de la série (ex : sur 7 jours – circulation moins intense le weekend)
- Une tendance peut être présente (ex : augmentation du nombre de véhicules en circulation)
- Du bruit peut encombrer la série

Depuis plusieurs décennies, de nombreuses recherches sont effectuées pour analyser les séries temporelles, notamment dans les secteurs de l'énergie, de la finance, du transport ou de la météorologie. L'étude effectuée dans cet article repose sur la problématique du groupe industriel Safran, à savoir la modélisation de l'usure de ses moteurs d'avions. En effet, les moteurs se dégradent en fonction de l'environnement, du type de vol, du nombre de vols, de l'état du moteur, etc. Un indicateur nommé **la marge EGT**, lié à la température du moteur, permet de suivre l'évolution de l'usure. Régulièrement, des actions de maintenance sont effectuées afin de restaurer la marge EGT, comme les Water Washers.

Notre problématique est alors la suivante : **Comment modéliser la vitesse d'usure d'un moteur d'avion entre deux événements de maintenance, en fonction de l'environnement dans lequel il vole ?**

La suite de ce papier est construite de la manière suivante. La section 2 décrit les travaux effectués dans la littérature pour traiter les séries temporelles multivariées. La section 3 présente les différents modèles que nous avons choisi d'étudier. Dans la section 4 nous parlerons des résultats obtenus. La section 5 conclut l'étude et propose des pistes envisageables pour de futures recherches.

2 Revue de littérature

De multiples approches peuvent être utilisées pour aborder les séries temporelles. On peut diviser les méthodes de traitement de ces séries en deux catégories :

- Les **méthodes statistiques classiques** (modèles paramétriques)
- Les **algorithmes de Machine Learning** (modèles non-paramétriques)

2.1 Les méthodes de traitement des séries chronologiques

De nombreuses recherches ont tout d'abord favorisé les modèles traditionnels, qui tentent de reconnaître une fonction connue sur les données passées, afin de prédire le futur (modèles paramétriques). Ils utilisent notamment le modèle Autorégressif à moyenne mobile intégrée (ARIMA), pour sa simplicité et ses caractéristiques statistiques inhérentes. Différents sous-modèles en ont découlé comme les modèles Autoregressifs (AR), Moyenne Mobile (MA) ou Autorégressif à Moyenne Mobile (ARMA). Cependant, ces modèles reposent sur des hypothèses de stationnarité, qui sont généralement très restrictives.

Des méthodes alternatives, traitant les séries temporelles à l'aide des régressions (en incluant des paramètres temporels) peuvent être utilisées. C'est le cas des méthodes LASSO, Ridge Regression, Random Forest et Vector Regression. Elles sont performantes et poussées par une communauté dynamique, mais peinent à identifier les relations non-linéaires entre les variables.

Face à la difficulté de ces méthodes à capturer les dépendances temporelles des séries chronologiques, les réseaux de neurones profonds (DNN), généralement utilisés pour le NLP ou la Computer Vision, ont pris une place de plus en plus importante dans la prédiction des séries temporelles.

L'article "Multivariate Temporal Convolutional Network : A deep Neural Networks Approach for Multivariate Time Series Forecasting" (1) compare les méthodes précédemment énoncées à des modèles de DNN. On parle notamment des Réseaux de Neurones Récurrent (RNN) et de son variant : le réseau de neurones récurrents à mémoire court-terme et long terme (LSTM). Malgré une littérature étendue sur l'emploi des RNN dans le traitement des séries temporelles, ces méthodes ne permettent pas de mettre en valeur l'information globale de la séquence. De plus, ils sont relativement lents aux changements dynamiques, le temps d'entraînement est coûteux, et ils sont sensibles au problème d'évanescence du gradient.

Au contraire, les CNN ont une force d'entraînement rapide (calculs sur GPU) et une structure simple pour le traitement des séries chronologiques. De plus, selon l'article "Temporal Convolutional Networks Applied to Energy-Related Time Series Forecasting" (2) les prédictions des CNN se sont montrées similaires à celles du LSTM pour des prédictions de court terme, avec un temps de calcul plus court.

Après une étude de la littérature scientifique sur les différentes approches adaptées aux séries temporelles *(Voir Annexe – Comparaison des modèles)*, et la prise en compte des travaux précédemment effectués par Safran, nous nous concentrons sur les réseaux de neurones convolutifs pour traiter notre problématique. Nous nous restreignons pour la suite de cette revue à l'étude des CNN.

2.2 Les Réseaux de Neurones Convolutifs

Un **Réseau de Neurones Convolutifs** est un type de réseau artificiel composé de neurones, répartis en plusieurs couches connectées entre elles. Chaque couche reçoit en input des données, et les transforme à l'aide d'une combinaison linéaire et d'une fonction d'activation. Les coefficients de la combinaison linéaire définissent les poids de la couche. Un réseau de neurones est construit en empilant les couches, pour finalement en extraire des prédictions en output. Les valeurs des poids des couches sont apprises par rétropropagation du gradient : on calcule progressivement les paramètres qui minimisent la fonction de perte. Dans la littérature, de nombreuses approches basées sur les CNN se sont développées.

Dans l'article "Conditional time series forecasting with convolutional neural networks"(3), les auteurs comparent l'efficacité des CNN par rapport aux modèles AR et LSTM, sur des séries financières particulièrement bruitées. Ils mettent en avant un atout considérable des CNN pour traiter le bruit de ces séries, grâce à leur structure en couches. Les auteurs s'inspirent du réseau de neurones WaveNet, principalement utilisé pour de la reconnaissance d'audio. Ils considèrent les CNN comme **simples, efficaces, facilement interprétables et pouvant servir de base solide dans le domaine de la prédiction**.

Dans l'article intitulé "Multiple convolutional neural networks for multivariate time series prediction"(4) les auteurs utilisent un **CNN multiple**. Ils décomposent la série temporelle selon trois sous-séries ; les composantes de proximité, de court terme et de long terme. Ces dernières sont traitées dans trois branches d'un même modèle selon une première couche de convolution. Ils mettent en place un réseau résiduel (ResNet) pour éviter le problème de l'évanescence du gradient, et ajoutent une partie linéaire à chaque convolution, pour rendre l'output sensible à la taille de l'input. Les auteurs mettent en valeur l'efficacité de la division de la série temporelle. On s'inspirera de cet article lors de la confection de notre CNN Multiple.

2.3 Les extensions

Les réseaux de neurones sont souvent **complétés par d'autres méthodes afin d'optimiser les prédictions**. Nous en présentons quelques-unes ici, dont nous pourrons nous inspirer pour notre propre modèle.

Un cas particulier des CNN combinant convolution et récurrence est explicité dans l'article "Temporal Convolutional Networks (TCN) Applied to Energy-Related Time Series Forecasting" (2) . Il met en valeur les TCN comme une alternative aux RNN, plus rapide et moins sensible aux choix des paramètres. Les auteurs conseillent finalement d'utiliser les LSTM pour les entrées de petite taille et de privilégier les TCN pour les inputs de grande taille. L'article "Multivariate Temporal Convolutional Network : A deep Neural Networks Approach for Multivariate Time Series Forecasting"(1) vient compléter l'article précédent en traitant les TCN multivariés. Ces modèles ont une structure à plusieurs branches et utilisent la convolution dilatée 1D pour éviter la perte d'informations de long terme par la couche de pooling.

Dans la littérature, de nombreux auteurs complètent leurs modèles en hybridant les CNN avec d'autres approches. Dans l'article "Envolving Deep CNN-LSTMs for Inventory Time Series Prediction" (6), les auteurs emploient à la fois un LSTM pour capturer les dépendances de long terme et un CNN pour détecter les caractéristiques locales. Ils en concluent une certaine complémentarité entre les deux modèles : les CNN permettent de réduire les variations et les LSTM sont performants pour modéliser la temporalité. On s'inspirera de cet article pour confectionner notre modèle CNN-LSTM.

Dans l'article intitulé "Multi-Scale Convolutional Neural Networks for Time Series Classification" (7), les auteurs constatent la difficulté des CNN à prendre en compte les échelles de temps différentes. Pour y remédier, il conçoivent un réseau dont la première couche effectue des transformations sur la série temporelle. Elle est ensuite répartie en trois branches : la série temporelle originale, la série lissée à l'aide de modèles MA et un sous-échantillon de la série représentant les caractéristiques principales.
Dans l'article de Penghui Liu et Jing Liu (8), les auteurs combinent un CNN avec un FCM (Fuzzy Cognitive Maps). Le FCM décompose le signal de la série, étudie les corrélations entre ces signaux, et fait une prédiction à l'aide d'un modèle de régression. Son but est de stabiliser les prédictions lorsque les données d'entrée s'éloignent des données d'entraînement.

Les auteurs de l'article "Convolutional Neural Network Couple with a Transfer-Learning Approach for Time-Series Flood Predictions" (9) utilisent la méthode du Transfer Learning. C'est une méthode très utilisée dans la reconnaissance d'image, qui consiste en l'utilisation sur des données particulières d'un modèle pré-entrainé sur d'autres données. L'un de ses grands avantages, en plus de sa précision, est son gain de temps d'entraînement.

3 La méthodologie

3.1 Les données

La base de données fournie et anonymisée par Safran contient un peu moins de 3 000 000 de vols associés à 26 paramètres (immatriculation, date, marge EGT ...). Ces 3 millions de vols représentent 1 397 moteurs. Pour la période de temps donnée, chaque moteur effectue en moyenne 3.8 Water Washer (écart type de 3.6 , minimum de 0 et maximum de 18).

Notre objectif est de prédire la pente de la marge EGT entre deux événements de maintenance (Water Washer ou autre) . Pour cela, nous traitons la base de données initiale afin d'obtenir des données conformes aux inputs d'un CNN. De ce fait, nous effectuons plusieurs prétraitements :

- Traitement des valeurs manquantes
- Traitement des formats, des variables catégorielles et mise à l'échelle
- Échantillonnage de la base pour obtenir le même nombre de vols entre deux événements de maintenance

A la fin de cette étape, nous obtenons une base de données de 6 612 périodes entre deux événements de maintenance. Chaque échantillon est représenté par 100 vols. Pour illustration, voici la représentation de la marge EGT pour un échantillon tiré au hasard :

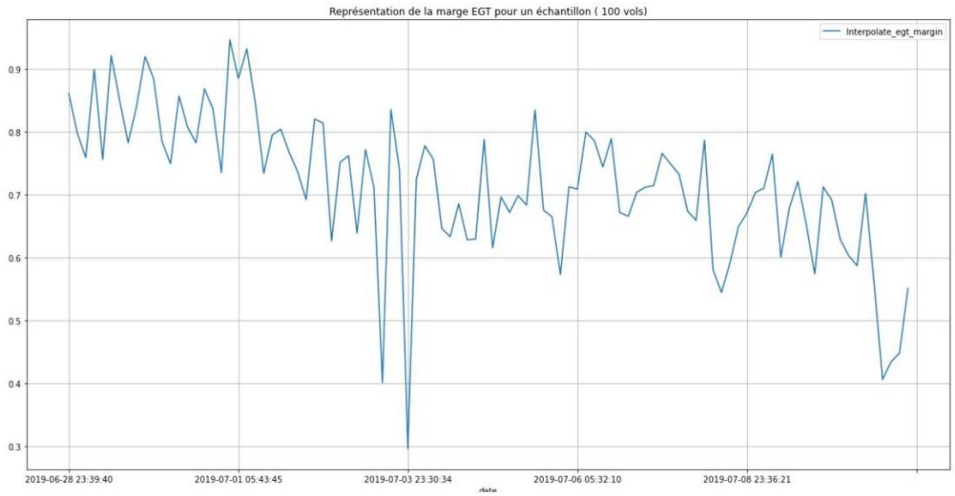


FIG. 1 – *Représentation de la marge EGT pour un échantillon (100 vols)*

La variable d'intérêt est la pente de la marge EGT (*egt_slope*) entre deux événements de maintenance. Pour prédire ces pentes nous utilisons les variables explicatives suivantes :

- Le type de moteur (*Engine_series*) - variable catégorielle à 7 modalités
- La configuration de A (*Config_A*) - variable catégorielle à 5 modalités
- La configuration de B (*Config_B*) - variable catégorielle à 4 modalités
- L'age du moteur (*Cycles_counter*) - variable continue
- La puissance du moteur au décollage (*Var_mot_1*) - variable continue
- Le nombre d'heures de vols (*flight_leg_hours*) - variable continue
- Les variables environnementales (*var_env_1 à var_env_5*) - variables continues sauf *var_env_4* catégorielle à 4 modalités.

3.2 Les modèles

Pour modéliser l'usure des moteurs d'avions, nous mettons au point 3 modèles différents : un premier modèle CNN classique, puis un modèle mêlant CNN et LSTM, et enfin un CNN à plusieurs branches. Nous nous concentrons sur les 6 612 échantillons de 100 vols effectués précédemment pour les 27 variables explicatives et l'unique variable d'intérêt. On effectue une base d'entraînement de 80% (5290 échantillons) et une base de test de 20% (1322 échantillons).

Paramètres communs à tous nos modèles

1. **La fonction objective:**
   Dans cet article, nous nous concentrons sur une problèmatique de régression (par opposition à la classification, couramment traitée à l'aide des CNN). Notre objectif est de prédire une valeur de pente qui, théoriquement, peut appartenir à l'intervalle $[-\infty; +\infty]$. Dans notre cas, les pentes appartiennent à un intervalle réduit n'excedant pas $[-4; 4]$. La fonction *Mean Squared Errors (MSE)* est employée pour mesurer la distance entre la prédiction et l'information réelle de la série chronologique. Elle pénalise davantage les erreurs de grande taille. Ainsi, notre objectif est de minimiser la MSE pendant la période d'apprentissage du CNN. La MSE s'écrit de la manière suivante :

   $$\frac{1}{N}\sum_{i=1}^{N}(Y_t - \hat{Y}_t)^2$$

   où $\hat{Y}_t$ et Y_t correspondent respectivement à la prédiction et à la vraie valeur de la série temporelle en t. N est le nombre total d'échantillons.

2. **La stratégie d'optimisation**
   Nous utilisons dans ce modèle l'algorithme *Adam* comme fonction d'optimisation du CNN. C'est un optimiseur très couramment utilisé dans la littérature, réputé non seulement pour sa capacité à ajuster les paramètres mais aussi pour sa robustesse, son efficacité, sa stabilité et son faible besoin en mémoire pendant la régression.

3.2.1 Le modèle de CNN simple

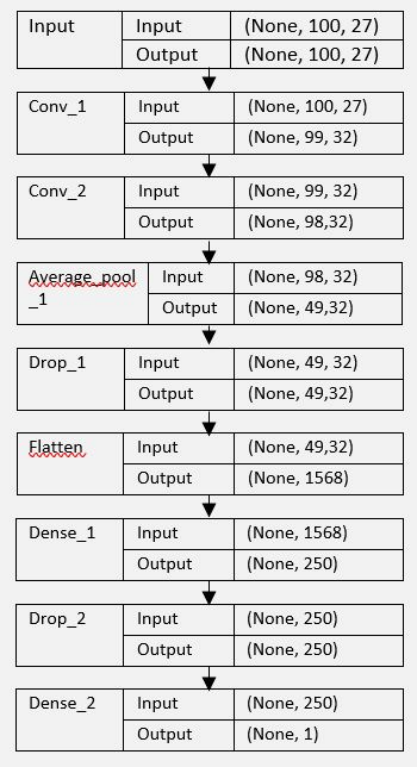
Pour notre premier modèle, nous mettons au point un réseau de neurones convolutionnels classique.

Architecture

L'apprentissage de ce modéle de CNN simple s'effectue sur une base d'entrainement de 5 290 matrices $(X_1,X_2,...,X_{5290})$ où X_i est de taille (100,27) $\forall i \in \{1,..,5290\}$ représentant les 27 variables explicatives pour les 5290 échantillons. Notre objectif est de prédire une pente de la marge EGT pour chaque matrice : $(Y_1,...,Y_{5290}) = f(X_1,X_2,...,X_{5290})$ où f illustre les différentes transformations effectuées par le réseau.

Pour recueillir les caractéristiques communes aux matrices d'entrée, nous appliquons une couche de convolution. Les filtres de cette couche correspondent aux caractéristiques que l'on souhaite capter : on applique donc un filtre assez faible en début de réseau, que l'on augmente au fil des couches de convolution. On y associe une fonction d'activation ReLU : $f(x) = max(x,0)$. Elle est couramment utilisée pour ses propriétés de stabilité lors de la rétropopagation du gradient.

On ajoute à notre modèle des couches de pooling permettant de réduire la taille des matrices. Finalement, et comme pour tous les réseaux de neurones, la dernière couche est une fully-connected. Elle reçoit en entrée un vecteur, et réduit sa dimension afin d'obtenir la taille de sortie souhaitée (1 dans notre cas). Contrairement aux précédentes couches, on applique ici une fonction d'activation linéaire (et non ReLu) pour prédire des pentes aussi bien positives que négatives. Pour limiter le sur-apprentissage, on utilise également des stratégies de Dropout et de Callbacks. Les Callbacks utilisés interrompent le modèle ou réduisent le learning rate en cas de non-apprentissage sur plusieurs périodes. L'architecture complète du modèle final est schématisée ci-après.



Architecture Modèle CNN Simple

Le processus d'entrainement

1^e **étape :** Créer le modèle manuellement par combinaison de différentes couches

2^e **étape :** Identifier les hyperparamètres à l'aide du package *Tuner* de Keras

3.2.2 Le modèle de CNN-LSTM

Dans ce modèle, nous combinons un réseau de type CNN à un dérivé des réseaux de neurones récurrents : le LSTM. Comme étudié à la section 2, ces modèles sont dits complémentaires : le CNN capture les tendances locales, tandis que le LSTM repère les dépendances de long-terme. Le modèle LSTM possède une mémoire interne, qui lui permet de retenir des informations d'une prédiction à l'autre. Les couches LSTM ont tout de même un inconvénient : elles sont 4 fois plus lourdes que les couches Dense, et demandent donc plus de temps de calcul.

Nos notations sont les suivantes :

- X_t correspond aux données entrantes dans le réseau de neurones
- h_{t-1} correspond à la valeur de sortie préditée par la couche précédente
- W_f correspond au poids des neurones
- B_t correspond au biais, c'est-à-dire le degré de liberté du réseau de neurons
- f_t correspond à la fonction d'activation

On peut schématiser la mémoire des réseaux LSTM avec trois portes:

1. **La porte d'oubli** : elle filtre les informations contenues dans la cellule mémoire précédente.
   L'équation est :
   $$f_t = \sigma(W_f * [h_{t-1},X_t] + b_t)$$
   où f_t est un tableau de valeurs comprises entre 0 et 1
   Ensuite, Le modèle LSTM multiplie f_t et la cellule mémoire précédente C_{t-1}. Lorsque les valeurs de sortie sont proches de 1, alors la valeur contenue dans C_{t-1} est autorisée à rester, et inversement lorsque les valeurs sont proches de 0, l'information contenue dans C_{t-1} est ignorée. Les valeurs restantes sont stockées dans C_1.
2. **La porte d'entrée** : elle décide quels inputs sont autorisés à rentrer dans le modèle. Les données les plus pertinentes vont être stockées dans la cellule mémoire, qui va servir à la décision à l'instant t et peut-être à l'instant *t+1* La fonction i_t décide quel input va être stocké dans la cellule mémoire :

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

   où i_t est un tableau avec des valeurs comprises entre 0 et 1.
   $\widetilde{C}$ contient les valeurs candidates à être stockées dans la cellule mémoire.

   $$\widetilde{C} = \sigma(W_c * [H_{t-1},X_t] + b_c)$$

   La multiplication terme à terme entre i_t et $\widetilde{C}$ nous indique quelles valeurs vont être stockées dans la cellule mémoire.La cellule mémoire est dorénavant:

   $$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}$$

3. **La porte de sortie** : après avoir pris en compte les informations entrantes, elle décide de l'output h_t . La prise de décision h_t est calculée de la sorte :

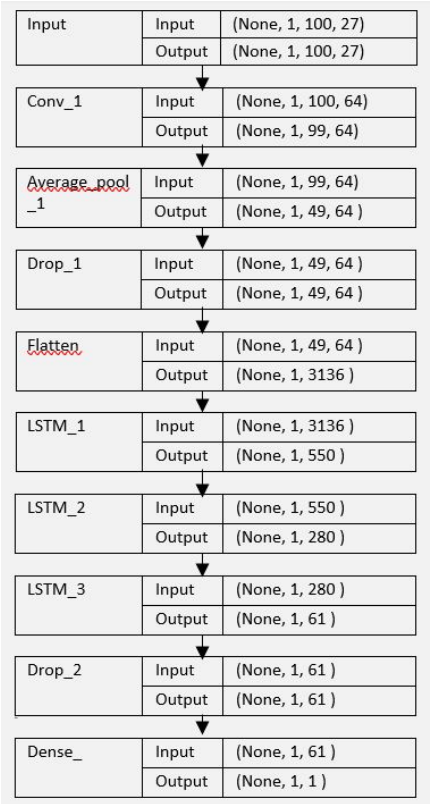
   $$o_t = \sigma(W_0 * [H_{t-1},X_t] + b_0)$$

   $$h_t = o_t \times \sigma(C_t)$$

   Ainsi, l'output h_t dépend des informations présentes dans la cellule mémoire C_t et des informations entrantes x_t.

Architecture

On crée des couches de convolutions identiques à celles utilisées dans le modèle CNN simple, à une différence de notation près : on utilise *TimeDistributed* avant chaque couche de convolution, pour que chaque donnée temporelle subisse le même traitement. On ajoute ensuite trois couches LSTM pour réduire au fur et à mesure le nombre d'outputs jusqu'à 1. L'architecture complète du modèle final est schématisée ci-contre.



Architecture du Modèle CNN-LSTM

Le processus d'entrainement

1^e **étape :** Créer le modèle avec d'abord des couches CNN puis des couches LSTM

2^e **étape :** Identifier les hyperparamètres à l'aide du package *Tuner* de Keras

3.2.3 Le modèle de CNN multiple

Les variables que nous utilisons peuvent être regroupées en deux catégories : les variables environnementales continues X_{env} et les variables caractèristiques de l'appareil (configuration du moteur, famille du moteur...) ou catégorielles X_{car}. Où $\{E1,E2,E3,E5\} \in X_{env}$ avec X_{env} de taille (6612,100,4) et $\{Serie_1,...,Serie_7,ConfigA_1,...ConfigA_5,ConfigB_1,...ConfigB_4,E4_1,...,E4_4,Counter,Mot,Flight\} \in X_{car}$ où X_{car} est de taille (6612,100,23)

Architecture

La décomposition des séries temporelles en amont de l'entrée dans le CNN a su faire ses preuves dans certains articles scientifiques. Etant donné les trajets réguliers des avions, il est probable qu'une saisonalité se répercute sur les variables environnementales. Nous décomposons ainsi les variables environnementales continues en tendance (T), saisonalité (S) et bruit (e) en fonction d'un modéle additif :

$$Y[t] = T[t] + S[t] + e[t]$$

En vue de cette décomposition et en guise de simplification, on emet deux hypothèses importantes :

- Les variables environnementales ont des périodes de répétition semblables. C'est une hypothèse faible puisque les variables environnementales sont corrélées entre elles.
- Les échantillons ont des périodes de répétition similaires. C'est une hypothèse forte puisque les moteurs sont très différents les uns des autres. On prendra comme fréquence initiale une valeur de 15. En effet, étant donné les 1400 moteurs initiaux, échantillonnés en tranches de 100 vols, un moteur est en moyenne représenté 5 fois dans les échantillons. En admettant qu'un avion répète ses trajets tous les 3 jours, on obtient une fréquence de 15. Cette valeur initiale est cependant discutable puisque l'étape d'échantillonnage ne prend pas en compte les fréquences de répétition. C'est pourquoi nous testerons notre modèle pour différentes fréquences.

Chaque variable environnementale continue est maintenant décomposée en trois séries : $\forall i \in \{1,2,3,5\}$: $Ei = Ei_T + Ei_S + Ei_e$

Dans cette partie, on aborde le problème de prédiction à l'aide d'un réseau de neurones convolutifs à plusieurs branches. Chaque branche est composée d'une entrée spécifique :

- Branche 1 (23,100) : les variables caractéristiques ou catégorielles X_{car}
- Branche 2 (4,100) : la tendance des séries environnementales $\forall i \in \{1,2,3,5\} : Ei_T$
- Branche 3 (4,100) : la saisonnalité des séries environnementales $\forall i \in \{1,2,3,5\} : Ei_S$
- Branche 4 (4,100) : le bruit des séries environnementales $\forall i \in \{1,2,3,5\} : Ei_e$

Chaque branche est composée d'une première couche de convolution où l'activation est la fonction ReLU, connue pour sa simplicité, ses propriétés non linéaires et son gradient fini en tous points (permet de contrôler le problème de l'evanescence du gradient).
La première branche contient davantage de couches puisque les variables caractéristiques ne sont présentes que dans une seule branche ; le modèle a donc besoin d'un réseau plus profond pour appréhender l'intégralité des caractéristiques de ces variables.
Les branches 2, 3 et 4 sont construites de manière semblable. On appuie plus faiblement sur les couches 2 et 4 : un apprentissage plus intense sur la couche 2 (tendance) entraînait une variance trop faible des prédictions, et l'augmentation de la profondeur de la couche 4 (bruit) éloignait les prédictions de 0.
On représente la sortie de chaque branche par l'output :

- $H_1 = f_1(W_1 * X_{car} + b_1)$ pour la branche 1
- $H_2 = f_2(W_2 * X_{env\_T} + b_2)$ pour la branche 2
- $H_3 = f_3(W_3 * X_{env\_S} + b_3)$ pour la branche 3
- $H_4 = f_4(W_4 * X_{env\_e} + b_4)$ pour la branche 4

Où H_i est l'output de la branche i , $X_{env\_T} = \begin{pmatrix} E1_T & E2_T & E3_T & E4_T \end{pmatrix}$, $X_{env\_S} = \begin{pmatrix} E1_S & E2_S & E3_S & E4_S \end{pmatrix}$, $X_{env\_e} = \begin{pmatrix} E1_e & E2_e & E3_e & E4_e \end{pmatrix}$, W_i et b_i correspondent aux paramétres de poids du modèle $\forall i \in$ {1,...,4}
On fusionne le tout pour obtenir l'output du CNN : $Y = H_1 + H_2 + H_3 + H_4$. Cet output final sera noté $\hat{Y}$. Comme pour les deux modèles précédents, on utilise des stratégies de Dropout et de Callbacks pour éviter le sur-apprentissage.

Le processus d'entrainement
1^e **étape :** Décomposer les séries des variables environnementales en sous-séries de tendance, saisonnalité et bruit à l'aide d'une fréquence initiale à 15.
2^e **étape :** Créer le modèle en y ajoutant manuellement les couches souhaitées afin d'obtenir une MSE la plus faible possible.
3^e **étape :** Identifier les hyperparamètres à l'aide du package *Tuner* de Keras.
4^e **étape :** Tester le modèle pour différentes fréquences afin d'identifier la période de temps qui décompose le mieux les séries chronologiques environnementales. On veillera à utiliser les mêmes échantillons de *test* et *train* pour une comparaison la plus juste possible. On selectionne finalement la fréquence 36 qui offre la meilleure perte.
L'architecture complète du modèle final est schématisée ci-après :

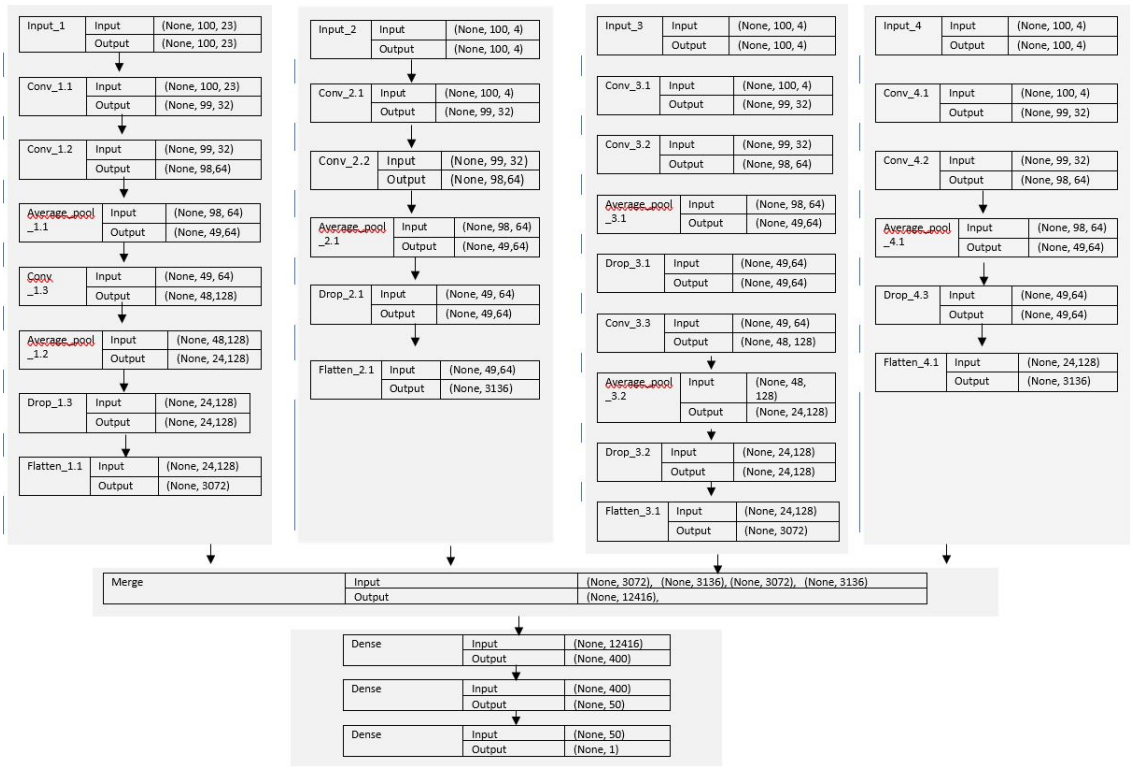


FIG. 2 – *Architecture du Modèle CNN multiple*

4 Les résultats

Dans cette section, on évalue et compare la performance de nos différents modéles. Pour cela nous utilisons un serveur Windows dont les caractéristiques sont les suivantes :

- Intel(R) Core(TM) i7 2.8GHz
- 16Go RAM
- Sans carte graphique NVidia

Nous utilisons les mêmes échantillons d'entraînement (80%) et de test (20%) pour la comparaison des trois modèles, en se basant sur une base de données initiales de taille (6 612, 100, 27).

4.1 Métriques

Dans cette expérience, nous adoptons plusieurs métriques d'évaluation conventionnelles :

1. **Mean Squared Error (MSE) :**

$$\frac{1}{N}\sum_{t=1}^{N}(Y_t - \hat{Y}_t)^2$$

où $\hat{Y}_t$ et Y_t correspondent respectivement à la prédiction et à la vraie valeur de la série temporelle en t. N est le nombre total d'échantillons. Cette métrique pénalise les erreurs de grandes tailles pour les erreurs de prédiction supérieures à 1 ou inférieures à -1.

2. **Mean Absolute Error (MAE) :**

$$\frac{1}{N}\sum_{t=1}^{N}|Y_t - \hat{Y}_t|$$

Plus de 50% des valeurs de pente appartiennent à l'intervalle $]-1{,}1[$, les erreurs de prédictions dans cet intervalle son minimisés par la MSE (conséquence de la fonction carré). C'est pourquoi nous introduisons cette seconde métrique : la MAE.

3. **Relative Absolute Error (RAE) :**

$$\frac{\sum_{t=1}^{N}|Y_t - \hat{Y}_t|}{\sum_{i=1}^{N}|Y_t - \bar{Y}_t|}$$

où $\bar{Y}_t$ correspond à la moyenne des observations prédites.
Cette métrique nous permet de constater l'efficacité du modèle par rapport à l'erreur faite en prédisant par la moyenne.

4. **Root relative Squared Error (RSE) :**

$$\frac{\sqrt{\sum_{t=1}^{N}(Y_t - \hat{Y}_t)^2}}{\sqrt{\sum_{t=1}^{N}(Y_t - \bar{Y}_t)^2}}$$

Métrique relative à la MSE, en comparaison à l'erreur faite en prédisant la moyenne.

5. **Relative Standard Deviation (RSTD) :**

$$\frac{\sqrt{\sum_{t=1}^{N}(\hat{Y}_t - \bar{\hat{Y}}_t)^2}}{\sqrt{\sum_{t=1}^{N}(Y_t - \bar{Y}_t)^2}}$$

Cette métrique permet de constater si l'écart-type des valeurs prédites se rapproche de l'écart-type des valeurs réelles. Contrairement aux autres métriques d'évaluation, on souhaite ici une valeur proche de 1.

4.2 Evaluation

En utilisant l'échantillon de test initial, on obtient les densités suivantes :

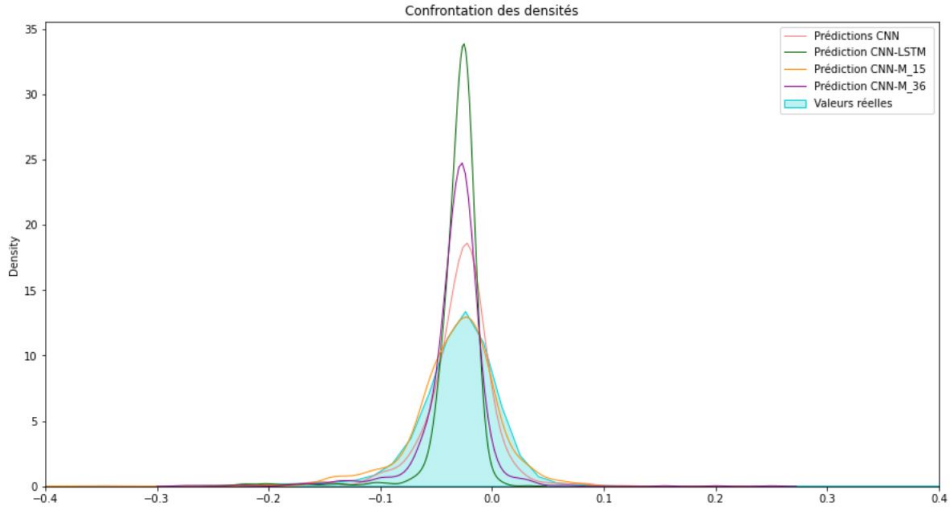


FIG. 3 – *Confrontation des densités pour l'échantillon initial de test*

On constate que la densité des prédictions du CNN Multiple (fréquence 15, en orange) est proche de la densité des valeurs réelles (en bleu). Les autres modèles ont tendance à rester proches de la moyenne et peinent à prédire les valeurs de pente s'en écartant. C'est notamment le cas pour le CNN-LSTM, dont l'écart-type est très faible et dont les valeurs positives sont presque inexistantes.

	MSE	MAE	RSE	RAE	RSTD
CNN	0.008440	0.035372	1.061467	1.251326	0.448334
CNN-LSTM	0.006602	0.025235	0.938819	0.892704	0.283774
CNN-M_15	0.007650	0.034330	1.010564	1.214448	0.407018
CNN-M_36	0.006836	0.028187	0.955284	0.997151	0.321365

FIG. 4 – *Evaluation - Echantillon de test initial*

On remarque que la MSE du CNN-LSTM est particulièrement faible. En effet, au vu des valeurs de pente concentrées autour de la moyenne, s'en éloigner augmente les erreurs de prédiction. C'est également pour ce modèle que la RAE est la plus faible : les erreurs de prédiction de ce modéle représentent 0.89 des erreurs de prédiction par la moyenne. Dans l'ensemble, les RAE des modèles sont assez élevées : les erreurs de prédiction à l'aide des modéles sont semblables aux erreurs faites en prédisant par la moyenne. Des valeurs plus faibles pour la RAE seraient souhaitables. Les écarts-types des modèles de CNN simple et CNN Multiples semblent être les plus en accord avec les écarts-types des vraies valeurs (RSTD autour de 0.4). Ces modèles permettent de prédire des valeurs s'éloignant de la moyenne.

En analysant ces mêmes modèles sur un échantillon de test différent, on obtient des MSE plus faibles. Cela est prévisible puisque cette base de test contient certains échantillons d'entrainement. Afin de constater la stabilité et l'efficacité de nos modèles, on les évalue sur 50 échantillons de test différents (pouvant contenir certaines données de la base d'entraînement). Il est évident que de nouveaux échantillons inconnus aux modèles seraient plus adaptés pour cette comparaison, cependant, la base initiale n'étant pas extensible, nous avons préféré l'exploiter au maximum pendant l'entrainement.

	MSE_mean	MSE_std	MAE_mean	MAE_std	RSE_mean	RSE_std	RAE_mean	RAE_std	RSTD_mean	RSTD_std
CNN	0.006334	0.003151	0.030236	0.001599	0.883576	0.153184	1.086094	0.045401	0.519967	0.116547
CNN-LSTM	0.004618	0.002761	0.022589	0.001244	0.750409	0.172852	0.811539	0.038508	0.613218	0.229228
CNN-M_15	0.010513	0.006089	0.040123	0.001973	1.095844	0.054936	1.440684	0.033775	0.427698	0.133951
CNN-M_36	0.010022	0.006014	0.034043	0.001941	1.061721	0.036552	1.221699	0.020662	0.337693	0.105762

FIG. 5 – *Résultats sur 50 échantillons de test*

Les résultats sont plutôt décevants pour les modèles de CNN multiples puisque la moyenne des MSE est élevée malgrè une base de test contenant des données d'entrainement. L'écart type est lui aussi élevé; les CNN multiples prédisent des résultats très variés en fonction des données reçues, ils ne sont donc pas très fiables. Le modèle de CNN sans pré-traitement est plus efficace ou semblable pour toutes les métriques. De plus, le choix de la fréquence 36 au lieu de 15, ne semble pas améliorer les prédictions. Il faut cependant rappeler que les modèles de CNN multiples sont basés sur une hypothèse forte ; les échantillons ont des périodes de répétition similaires. Ces résultats nous incitent à penser que cette hypothèse n'est pas vérifiée.

Le modèle de CNN-LSTM montre quant à lui des caractéristiques intéressantes. Sur les 50 échantillons testés, la MSE est de 0.0046 avec un écart-type de 0.002. Il semble être plus efficace que la prédiction par la moyenne (RSE et RAE < 1). La RSTD est relativement haute par rapport aux autres modèles, les prédictions du CNN-LSTM sont moins concentrés autour de la moyenne que pour les autres modèles. Il faut cependant rappeler que ces 50 échantillons tests contiennent des observations d'entraînement. On peut donc seulement affirmer que ce modèle a une plus grande capacité à prédire efficacement sur des données déjà vues. C'est une condition nécessaire mais insuffisante pour conclure la fiabilité de ce modéle. Il faudrait le tester sur d'autres bases de données inconnues afin de connaître ses capacités d'adaptation à de nouvelles données.

5 Conclusion

La prédiction des séries temporelles multivariées est un problème récurrent. Dans cet article, nous utilisons des algorithmes de Deep Learning pour résoudre la problèmatique du groupe Safran : la modélisation de l'usure des moteurs d'avions. Nous utilisons trois méthodes basées sur les Réseaux de Neurones Convolutifs. Les résultats nous permettent de conclure que les capacités prédictives des CNN sont réelles. Chaque modèle utilisé a néanmoins ses avantages et ses inconvénients.

Premièrement, le CNN simple a une capacité prédictive relativement stable entre différents échantillons mais peine à s'éloigner de la moyenne. Son complément, le CNN-LSTM, est d'autant plus stable ; les prédictions autour de la moyenne sont particulièrement justes. Il est cependant difficile de prédire les pentes extrêmes avec ce modèle. Le CNN multivarié est quant à lui peu pertinent pour certains échantillons. Il est cependant le plus à même de prédire des valeurs extrêmes, notamment les pentes positives. Ce réseau est également basé sur une hypothèse forte de périodicité commune entre les moteurs. Il serait alors intéressant de revoir ce modèle en changeant la phase de décomposition des séries chronologiques. Nous pourrions alors, soit prendre en compte les saisonnalités lors de l'échantillonnage puis décomposer, soit intégrer ce prétraitement à la première couche du modèle.

De plus, pour augmenter la performance des CNN, il serait souhaitable d'envisager d'autres prétraitements. Par exemple, nous pourrions considérer la méthode *Embedding* pour traiter les variables catégorielles, utiliser des tests d'inclusion-exclusion de certaines variables (en s'inspirant des indicateurs BIC ou AIC) ou effectuer de la *Data Augmentation* pour multiplier les capacités d'apprentissage. De la même façon, nous pourrions davantage exploiter les réseaux en testant d'autres combinaisons de couches et/ou d'autres couches. Il serait également avantageux d'utiliser une carte graphique pour optimiser le temps d'entrainement, notamment pour les Réseaux de Neurones Récurrents. Finalement, et pour aller plus loin, d'autres modèles sont envisageables comme par exemple combiner nos modèles afin de créer un CNN-LSTM multiple, ou s'inspirer de la littérature pour expérimenter d'autres combinaisons de modèles (TCN, FCM, AR, Resnet ...)

Références

[1] Renrhuo Wan, Shuping Mei, Jun Wang, Min Liu, Fan Yang. "Multivariate Temporal Convolutional Network : A deep Neural Networks Approach for Multivariate Time Series Forecasting", *Journal Electronics*, (2019).

[2] Pedro Lara-Benitez, Manuel Carranza-Garcia, José M. Luna-Romera, José C. Riquelme. "Temporal Convolutional Networks Applied to Energy-Related Time Series Forecasting", *Journal Applied Sciences*, (2020).

[3] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. "Conditional time series forecasting with convolutional neural networks", (2018).

[4] Kang Wanga, Kenli Li, Liqian Zhoub, Yikun Hua, Zhongyao Chengc, Jing Liude, Cen Chena. "Multiple convolutional neural networks for multivariate time series prediction", *Journal Neurocomputing*, (2019).

[5] Shaojie Bai, J.Zico Kolter, Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", (2018).

[6] Ning Xue, Isaac Triguero, Grazziela P. Figueredo, Dario Landa-Silva. "Envolving Deep CNN-LSTMs for Inventory Time Series Prediction", *Journal IEEE Xplore*, (2019).

[7] Zhicheng Cui, Wenlin Chen, Yixin Chen. "Multi-Scale Convolutional Neural Networks for Time Series Classification", (2016).

[8] Penghui Liu, Jing Liu. "CNN-FCM System modeling promotes stability of deep learning in time series prediction", *Journal Knowledge-based Systems*, (2020).

[9] Kimura Nobuaki, Yoshinaga Ikuo, Sekijima Kenji, Azechi Issaku, Baba Daichi. "Convolutional Neural Network Coupled with a Transfer-Learning Approach for Time-Series Flood Predictions Water", (2020).

Annexe

Comparaison des modèles

Modèles	Avantages	Inconvénients
ARIMA	- Simple - Efficace pour les modèles stationnaires	- Hypothèse de stationnarité
VAR	- Simple - Capture les interdépendances entre plusieurs séries temporelles	- Sensible au sur-apprentissage - Coût de calculs élevé
Modèle linéaire	- Simple	- Difficile d'identifier les relations non-linéaires
RNN	- Beaucoup de progrès - Efficace pour les données séquentielles	- Considère la dernière période pour prédire - Dispersion du gradient - Entraînement lent
LSTM	- Plus performant que les RNN car prend en compte plusieurs périodes pour prédire - Identifie les caractéristiques de court terme et long terme	- Entraînement lent - Non robuste pour les données apériodiques
CNN	- Calculs rapides (GPU) - Plutôt connu pour détecter les caractéristiques locales	- Taille fixe de l'input - Difficile de prendre en compte les dépendances de long terme
TCN	- Performant pour les séquences d'entrée de grande taille - Stabilité du gradient	- Mémoire requise pour l'évaluation : prise en compte de l'ensemble de la période passée