Séries temporelles et réseaux de neurones récurrents

Culture Sciences de l'Ingénieur

Édité le 05/09/2022

école normale supérieure paris—saclay

Valentin NOËL

Cette ressource est issue d'une publication du numéro 109 de La Revue 3EI de juillet 2022 et fait partie du « Dossier Intelligence Artificielle » [8] sur Culture Sciences de l'Ingénieur. *Valentin Noël est doctorant au Laboratoire SATIE.*

En opposition aux données dites « statiques « (i.e. images), les données dites « dynamiques » (i.e. vidéos), dont font partie les séries temporelles, sont utilisées dans de nombreux domaines, allant de la météorologie à la finance, en passant l'analyse de sentiments. Du fait de leur temporalité, les séries temporelles ne peuvent pas être fournies au réseau de neurones tel que le sont les données statiques. De plus, de nouvelles problématiques émergent dues elles aussi à leur temporalité, ce qui rend l'approche théorique et pratique de la prédiction de séries temporelles par réseaux de neurones récurrents (RNN) drastiquement différente de celle abordée pour des données statiques.

Dans cette ressource, nous nous efforçons de dépeindre au mieux les caractéristiques, particularités et les complexités d'un tel type de données, que nous illustrons avec une application [7] : l'entrainement d'un RNN (i.e. Long-Short Term Memory) pour la prévision des conditions climatiques.

1 – Introduction

Dans le but de donner une image globale des différentes étapes de la prévision de données dynamiques il est nécessaire de définir les données dont nous voulons prévoir l'évolution. Nous prenons l'exemple de la consommation électrique d'un foyer, ce foyer désirant pouvoir prévoir l'évolution de cette consommation électrique d'une année sur l'autre afin de définir un budget prévisionnel. Ce phénomène n'étant pas parfaitement périodique car sujet à des aléas multiples : conditions météorologiques, nombre d'appareils électroniques, etc. Pour pouvoir pour autant prévoir l'évolution de ces données dynamiques, nous utilisons un réseau de neurones récurrent. Les données d'entraînement seront donc les consommations électriques des années précédentes, tandis que les données de sortie du réseau de neurones seront les données de la consommation électrique de l'année suivante.

2 – Séries temporelles : définition et spécificités

2.1 - Des données statiques aux données dynamiques

Les séries temporelles sont un type de données qui sont échantillonnées en fonction d'une dimension temporelle (jours, mois, années, etc.). Nous qualifions ces données de « dynamiques » car elles sont indexées sur la base d'un attribut de temps, ce qui leur donne donc un ordre temporel implicite. Les données statiques peuvent aussi avoir un attribut de temps, mais la différence réside dans le fait que les données statiques ne seront pas échantillonnées ou indexées en fonction de cet attribut.

Lorsque nous appliquons des algorithmes d'apprentissage automatique sur des données de séries temporelles et que nous voulons faire des prédictions pour des futures valeurs temporelles, par exemple, prédire le nombre de nouveaux étudiants inscrits dans un établissement à partir des données des 10 années précédentes, ou prédire le temps pour un certain jour à partir des données météorologiques de plusieurs années. Ces prédictions sur des données de séries temporelles sont appelées prévisions, ou « forecasting ».

2.2 - Spécificités et contraintes des données temporelles

Une contrainte inhérente à l'apprentissage par réseaux de neurones récurrents est de premièrement assurer la qualité des données dynamiques fournies, c'est-à-dire la continuité des séries temporelles fournies.

Ainsi, l'imputation des données manquantes est une étape clé du prétraitement dans tout projet d'apprentissage automatique tabulaire. Pour les données statiques, on peut utiliser des techniques telles que l'imputation simple, qui permet de combler les données manquantes par la moyenne, la médiane ou le mode des données, selon la nature de l'attribut, ou des méthodes plus sophistiquées telles que l'imputation par le plus proche voisin, qui utilise un algorithme KNN (K Nearest Neighbors) pour identifier les données manquantes.

Dans le cas des données dynamiques, la moyenne statique n'est d'aucune utilité, car il n'est pas logique de combler les valeurs manquantes en prenant des repères dans le futur (en fonction des données disponibles). Nous utilisons ce que l'on appelle la moyenne glissante, la moyenne mobile ou la moyenne de fenêtre, qui consiste à prendre la moyenne des valeurs relatives à une fenêtre prédéfinie, par exemple une fenêtre de 7 jours ou d'un mois. Nous pouvons donc utiliser cette moyenne mobile pour combler les lacunes de nos séries temporelles.

Une méthode plus plébiscitée existe, qui utilise l'ordre implicite des données de séries chronologiques. L'interpolation la méthode la plus utilisée pour déterminer les parties manquantes des données de séries temporelles. Les interpolations utilisent la valeur présente avant et après le point manquant pour calculer la donnée manquante. Par exemple, les interpolations linéaires fonctionnent en calculant une ligne droite entre les deux points, en en faisant la moyenne et en obtenant la donnée manquante. Il existe de nombreux types d'interpolations, comme les interpolations linéaires, de Spline et de Stineman. Dans certains cas où de nombreuses données sont manquantes, d'autres approches basées sur des modèles auto-régressifs (AR), peuvent être utilisés. Les figures 1 et 2 représentent le signal d'entrée avec des données manquantes puis avec le signal reconstruit.

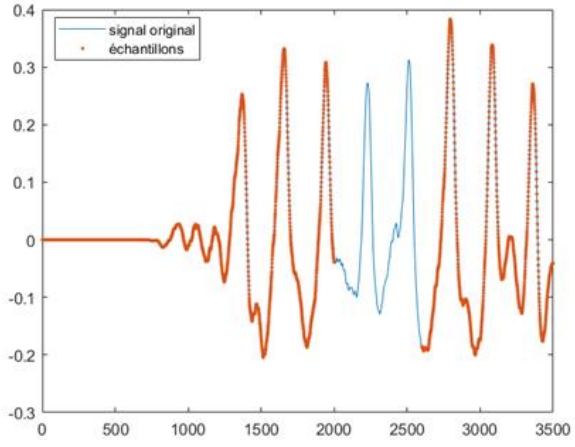

Figure 1 : Représentation du signal original avec le signal possédant des données manquantes

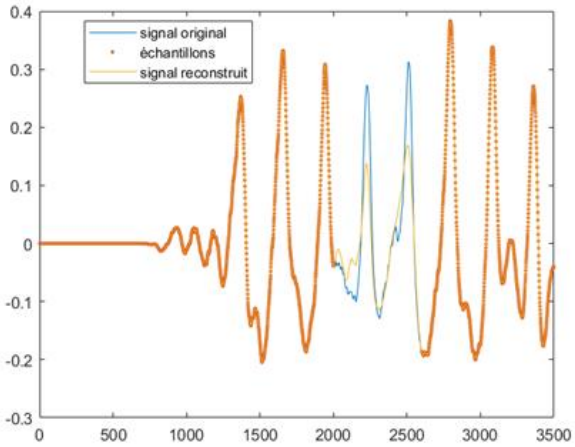

Figure 2 : Représentation du signal original avec le signal possédant des données manquantes et le signal reconstruit

2.3 - Caractéristiques principales des séries temporelles

Manipuler des caractéristiques différencie de nouveau les données dynamiques des données statiques : en effet, les caractéristiques sont traitées différemment dans les données de séries temporelles par rapport aux données statiques.

Dans le cas des données statiques, les techniques les plus communément rencontrées comprennent les transformations de caractéristiques, la mise à l'échelle, la compression, la normalisation, etc.

Quant aux données de séries temporelles, elles ont obligatoirement des composantes de séries temporelles. L'extraction de ces caractéristiques peut s'effectuer via la décomposition STL (Seasonal and Trend decomposition using Loess) et certaines d'entre elles sont définies ci-dessous [1] :

- **Saisonnalité** : La saisonnalité fait référence à une propriété des séries temporelles qui affiche des modèles périodiques se répétant à une fréquence constante.
- **Tendance** : Les données de séries chronologiques présentent une tendance lorsque leur valeur varie dans le temps. Une valeur croissante indique une tendance positive et une valeur décroissante, une tendance négative.
- **Reste** : Après avoir extrait la tendance et la saisonnalité des données, ce qui reste est ce que nous appelons le reste (erreur) ou le résidu. Cela permet de détecter les anomalies dans les séries chronologiques.
- **Cycle** : Les données de séries temporelles sont dites cycliques lorsqu'il existe des tendances sans répétitions fixes ou saisonnalité.
- **Stationnarité** : Les données de séries temporelles sont stationnaires lorsque leurs caractéristiques statistiques ne changent pas dans le temps, c'est-à-dire une moyenne et un écart-type constants.

Une fois extraites, ces composantes constituent la base pour l'analyse de l'évolution de la série temporelle afin de comprendre son comportement et de pouvoir choisir et adapter un modèle de série temporelle approprié.

Les données de séries temporelles peuvent avoir d'autres attributs que les caractéristiques temporelles. Si ces attributs sont temporels, la série temporelle résultante sera multivariée et si elle est statique, elle sera univariée avec des caractéristiques statiques. Les caractéristiques non temporelles peuvent utiliser des méthodes issues des techniques statiques de manière à ne pas nuire à l'intégrité des données.

3 – Approches algorithmiques pour la prévision des séries temporelles

3.1 - Préparation des données dynamiques

Il y a deux choses à établir avant de concevoir un modèle de prévision [2] :

- Les informations disponibles au moment où la prévision est faite (caractéristiques) ;
- La période pendant laquelle l'utilisateur a besoin de valeurs prévisionnelles (objectif).

L'origine de la prévision est le moment auquel une prévision est effectuée. En pratique, l'origine de la prévision est le dernier moment pour lequel des données d'entraînement sont disponibles pour la période à prévoir. Tout ce qui va jusqu'à l'origine peut être utilisé pour créer des caractéristiques.

L'horizon de prévision, qui décrit l'objectif, est la période pour laquelle une prévision va être effectuée. Une prévision est décrite la plupart du temps par le nombre de pas de temps dans son horizon : une prévision "à 1 pas" ou "à 5 pas", par exemple. Ce temps entre l'origine et l'horizon est le délai d'exécution (parfois appelé la latence) de la prévision. Dans la pratique, il peut être nécessaire qu'une prévision commence plusieurs étapes avant l'origine en raison de retards dans l'acquisition ou le traitement des données.

3.2 - Les réseaux de neurones récurrents

Le choix des algorithmes pour les données de séries temporelles est complètement différent de celui des données statiques. Un algorithme applicable aux données temporelles doit être capable d'extrapoler des modèles et d'encapsuler les composantes de séries temporelles en dehors du domaine des données d'apprentissage. Or, la plupart des algorithmes d'apprentissage automatique statiques, comme la régression linéaire ou les SVM (Support Vector Machine), n'ont pas cette capacité car ils généralisent l'espace d'apprentissage pour toute nouvelle prédiction. Dans le but de prévoir les séries temporelles, des réseaux de neurones sont entre autres utilisés. Ces réseaux de neurones font partie de la famille des RNN (Recurent Neural Network). Le terme RNN est utilisé pour désigner la classe des réseaux à réponse impulsionnelle infinie, tandis que le terme CNN (Convolutional Neural Network) désigne la classe des réseaux à réponse impulsionnelle finie. Les deux classes de réseaux présentent un comportement dynamique temporel. Un réseau récurrent à réponse impulsionnelle finie est un graphe acyclique dirigé qui peut être déroulé dans le temps (foldable) et remplacé par un réseau neuronal strictement feedforward, tandis qu'un réseau récurrent à réponse impulsionnelle infinie est un graphe cyclique dirigé qui ne peut pas être déroulé dans le temps (unfoldable). Pour n'en citer que les principaux :

- FRNN (Fully Recurrent Neural Network)
- ESN (Echo State Network)
- Second-order RNNs:
    - LSTM (Long-Short Term Memory)
    - GRU (Gated Recurrent Unit)

3.3 - Fully Recurrent Neural Network et le problème de gradient évanescent

Dans le but de décrire de la façon la plus complète le fonctionnement d'un réseau de neurones récurrent, il est nécessaire de comprendre le comportement du FRNN et de la raison de l'apparition des RNNs de second ordre. Les FRNN connectent les sorties de tous les neurones aux entrées de tous les neurones. Il s'agit de la topologie de réseau neuronal la plus générale. La figure 3 représente les différentes étapes dans le temps du même réseau neuronal entièrement récurrent.

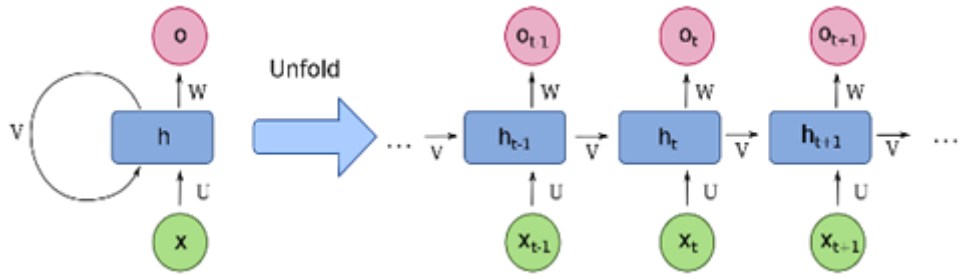

Figure 3 : De gauche à droite : RNN basique non-déplié dans le temps et RNN basique déplié dans le temps [3]

L'élément le plus à gauche de la figure 3 illustre les connexions récurrentes sous la forme de l'arc étiqueté "*v*". Il est "déplié" dans le temps pour produire l'apparence de couches.

Cependant, un tel réseau est victime de ce qui est appelé dans la littérature le gradient évanescent (ou vanishing gradient). Le problème du gradient évanescent est rencontré lors de l'apprentissage des réseaux neuronaux artificiels avec des méthodes d'apprentissage basées sur le gradient et la rétropropagation. Dans ces méthodes, à chaque itération de la formation, chaque poids w_{ij} du réseau neuronal reçoit une mise à jour proportionnelle à la dérivée partielle de la fonction d'erreur par rapport au poids actuel. Les fonctions d'activation traditionnelles, telles que la fonction tangente hyperbolique, ont des gradients compris dans la plage (0,1). La rétropropagation calculant les gradients par la règle de la chaîne, avec n le nombre de couches du réseau, cela a pour effet de multiplier n de ces faibles valeurs pour calculer les gradients des premières couches du réseau. Cela signifie que le gradient diminue exponentiellement avec n alors que les premières couches s'entraînent très lentement. Dans certains cas, le gradient est extrêmement faible, ce qui empêche le poids de changer de valeur et ainsi mener à un apprentissage incomplet.

Le problème du gradient évanescent affecte non seulement les réseaux feedforward à plusieurs couches, mais aussi les réseaux récurrents. Lorsqu'on utilise des fonctions d'activation dont les dérivées peuvent prendre des valeurs plus importantes, on risque de rencontrer le problème connexe du gradient explosif.

3.4 - Long-Short Term Memory, Gated Reccurent Unit: éviter le problème de gradient évanescent

En théorie, les RNN classiques peuvent suivre des dépendances à long terme arbitraires dans les données d'entrée. Le problème des RNN classiques est de nature pratique : lors de l'apprentissage d'un RNN classique par rétropropagation, les gradients à long terme qui sont rétropropagés peuvent « disparaître » (c'est-à-dire tendre vers zéro) ou « exploser » (c'est-à-dire tendre vers l'infini), en raison des calculs impliqués dans le processus, qui utilisent des nombres à précision finie. Dans le but d'éviter ces problèmes de gradient, un réseau neuronal appelé Long-Short Term Memory (LSTM) a été développé.

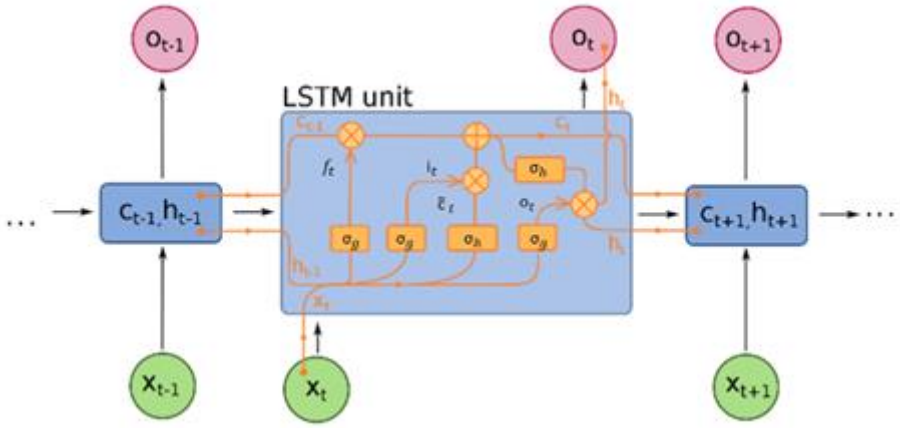

Figure 4 : Long-Short Term Memory Unit [4]

Contrairement aux réseaux neuronaux feedforward standard, LSTM possède des connexions de rétroaction. Un tel réseau neuronal récurrent peut traiter non seulement des points de données statiques, mais aussi des données dynamiques. Une unité LSTM commune est composée d'une cellule, d'une porte d'entrée, d'une porte de sortie et d'une porte d'oubli, comme illustré dans la figure 4. La cellule se souvient des valeurs sur des intervalles de temps arbitraires et les trois portes régulent le flux d'informations entrant et sortant de la cellule, ce qui a pour effet de limiter le problème du gradient évanescent. Les formes compactes des équations pour le forward pass d'une cellule LSTM avec une porte d'oubli sont [4] :

$$f_t=\sigma_g(W_f x_t+U_f h_{t-1}+b_f)$$

$$i_t=\sigma_g(W_i x_t+U_i h_{t-1}+b_i)$$

$$o_t=\sigma_g(W_o x_t+U_o h_{t-1}+b_o)$$

$$\tilde{c}_t=\sigma_g(W_c x_t+U_c h_{t-1}+b_c)$$

$$c_t=f_t \circ c_{t-1}+i_t \circ \tilde{c}_t$$

$$h_t=o_t \circ \sigma_h(c_t)$$

Avec les valeurs initiales $c_0=0$ et $h_0=0$. L'opérateur $\circ$ dénote le produit d'Hadamard (produit par éléments). L'indice t représente le pas temporel. Les autres variables sont listées ci-dessous :

- $x_t \in R_d$: vecteur d'entrée
- $f_t \in (0,1)_h$: vecteur d'activation de la porte d'oubli
- $i_t \in (0,1)_h$: vecteur d'activation de la porte d'entrée/mise à jour
- $o_t \in (0,1)_h$: vecteur d'activation de la porte de sortie
- $h_t \in (-1,1)_h$: vecteur de sortie de l'unité LSTM.
- $\tilde{c}_t \in (-1,1)_h$: vecteur d'activation de l'entrée de la cellule
- $c_t \in R_h$: vecteur d'état des cellules
- $W \in R^{h \times d}$, $U \in R^{h \times h}$ et $b \in R_h$: les matrices de poids et les paramètres des vecteurs de biais qui doivent être appris pendant la formation

Les RNN utilisant des unités LSTM résolvent donc partiellement le problème du gradient évanescent. Cependant, les réseaux LSTM peuvent toujours souffrir du problème du gradient explosif. Quant au GRU, il est comme un LSTM avec une porte d'oubli, mais a moins de paramètres que le LSTM, car il manque une porte de sortie comme illustré dans la figure 5.

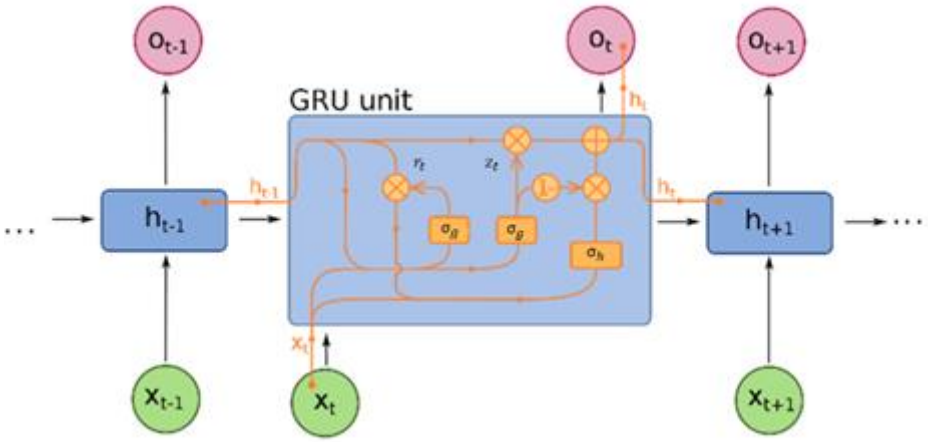

Figure 5 : Gated Reccurent Unit [5]

Malgré cela, il a été démontré que les GRU ont une meilleure performance sur certains ensembles de données plus petits et moins fréquents.

4 – Application : séries temporelles de données climatiques

4.1 - Séries temporelles de données climatiques : le dataset.

Bien qu'il en existe de plus en plus, les bibliothèques de référence pour construire des applications d'apprentissage automatique basées sur LSTM sont Tensorflow et Keras qui jouissent d'une forte

popularité parmi les praticiens de l'apprentissage automatique. D'ailleurs, l'application montée ici est reproductible sur Google Collaboratory [7] : la prévision de séries temporelles pour les prévisions météorologiques.

Index	Features	Format	Description
1	Date Time	01.01.2009 00:10:00	Date-time reference
2	p (mbar)	996.52	The pascal SI derived unit of pressure used to quantify internal pressure. Meteorological reports typically state atmospheric pressure in millibars.
3	T (degC)	-8.02	Temperature in Celsius
4	Tpot (K)	265.4	Temperature in Kelvin
5	Tdew (degC)	-8.9	Temperature in Celsius relative to humidity. Dew Point is a measure of the absolute amount of water in the air, the DP is the temperature at which the air cannot hold all the moisture in it and water condenses.
6	rh (%)	93.3	Relative Humidity is a measure of how saturated the air is with water vapor, the %RH determines the amount of water contained within collection objects.
7	VPmax (mbar)	3.33	Saturation vapor pressure
8	VPact (mbar)	3.11	Vapor pressure
9	VPdef (mbar)	0.22	Vapor pressure deficit
10	sh (g/kg)	1.94	Specific humidity
11	H2OC (mmol/mol)	3.12	Water vapor concentration
12	rho (g/m ** 3)	1307.75	Airtight
13	wv (m/s)	1.03	Wind speed
14	max. wv (m/s)	1.75	Maximum wind speed
15	wd (deg)	152.3	Wind direction in degrees

Tableau 1 : Détail des caractéristiques du set de données

Nous utiliserons le jeu de données climatiques d'Iéna (Allemagne) enregistré par l'Institut Max Planck de biogéochimie. Le jeu de données se compose de 14 caractéristiques telles que la température, la pression, l'humidité, etc., enregistrées une fois toutes les 10 minutes. Le tableau 1 présente les noms des colonnes, leurs formats de valeur et leur description.

4.2 - Visualisation des données brutes

Pour nous donner une idée des données avec lesquelles nous travaillons, chaque caractéristique a été tracée ci-dessous dans la figure 6. Cela montre le modèle distinct de chaque caractéristique sur la période de 2009 à 2016. Il montre également où se trouvent les anomalies, qui seront traitées lors de l'étape de normalisation. De façon à visualiser la corrélation entre différentes caractéristiques, une heat map est disponible à la figue 7. Nous pouvons voir sur la heat map de corrélation que certains paramètres comme l'humidité relative et l'humidité spécifique sont redondants. Par conséquent, nous utiliserons certaines caractéristiques, pas toutes, certaines seront enlevées lors de l'étape de pré-traitement des données.

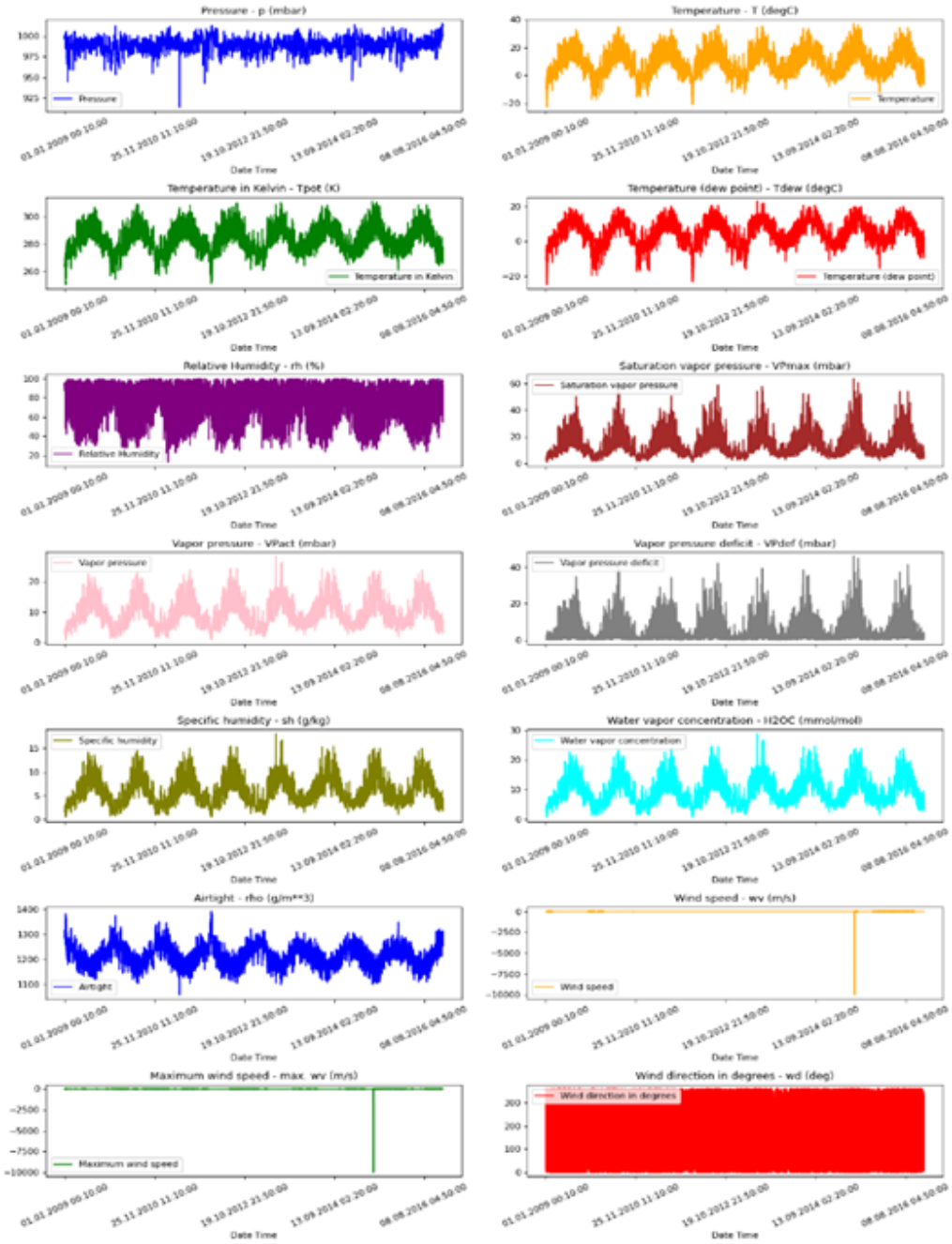

Figure 6 : Visualisation des données brutes du set de données

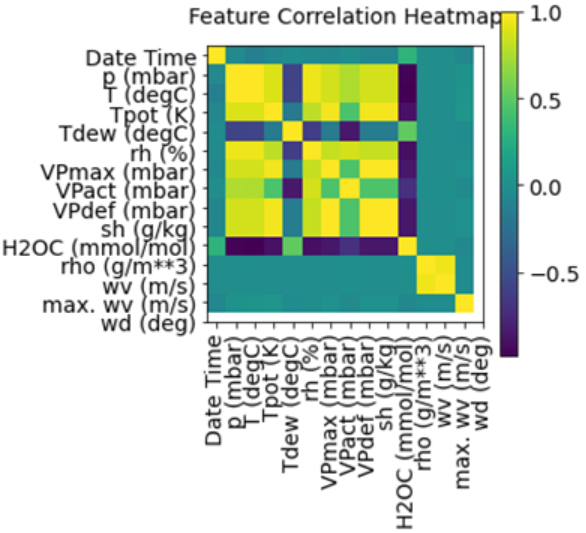

Figure 7 : Heat map des corrélations entre caractéristiques du set de données

4.3 - Prétraitement des données

Ici, ~300 000 points de données sont choisis pour l'entraînement. L'observation est enregistrée toutes les 10 minutes, c'est-à-dire 6 fois par heure. Le rééchantillonnage est mis à un point par heure car aucun changement radical n'est attendu en 60 minutes. Les 1440 derniers horodatages (14400/6=240 heures) sont suivis. Ces données seront utilisées pour prédire la température après 72 horodatages (72/6=12 heures). Comme chaque caractéristique a des valeurs avec des plages variables, une normalisation est effectuée pour confiner les valeurs des caractéristiques dans une plage de [0, 1] avant de former un réseau neuronal. Pour ce faire, la moyenne est soustraite à chaque valeur et est divisée par l'écart type de chaque caractéristique. 70 % des données seront utilisées pour entraîner le modèle, c'est-à-dire 294 385 lignes. Le modèle reçoit les données des 10 premiers jours, soit 1440 observations, qui sont échantillonnées toutes les heures. La température après 72 (12 heures * 6 observations par heure) observations sera utilisée comme étiquette.

4.4 - Les sets d'entraînement et de validation

Pour le set d'entraînement, les labels de l'ensemble de données d'apprentissage commencent à partir de la 1512ème observation (1440 + 72). Une fonction est définie, ayant pour rôle de prendre une séquence de points de données collectés à intervalles égaux, ainsi que des paramètres de séries temporelles tels que la longueur des séquences/fenêtres, l'espacement entre deux séquences/fenêtres, etc., pour produire des lots de sous-séries d'entrées et de cibles échantillonnées à partir de la série temporelle principale.

L'ensemble de données de validation ne doit pas contenir les 1512 dernières lignes, car nous n'aurons pas de données d'étiquette pour ces enregistrements, donc 1512 doit être soustrait de la fin des données. L'ensemble de données de validation doit commencer à partir de 1512 après la dernière donnée utilisée pour le set de test.

Il doit être noté que les paramètres choisis ici sont différents de ceux proposés dans [7], pouvant mener à des résultats non-optimaux. Ce choix a été fait dans le but de pouvoir comparer les résultats obtenus dans les deux configurations mais aussi d'illustrer l'intérêt de l'utilisation de notebook facilement modifiables et très visuels pour la compréhension de thèmes plus ou moins compliqués, comme ici les réseaux de neurones récurrents.

4.5 - Entraînement et prédiction

Le ModelCheckpoint Callback est utilisé pour sauvegarder régulièrement les points de contrôle, et l'EarlyStopping callback pour interrompre la formation lorsque la validation loss ne s'améliore plus. L'évolution de la validation loss et training loss est représentée dans la figure 8.

Figure 8 : Évolution de la validation loss et training loss en fonction du nombre d'epochs.

Nous remarquons que ces paramètres mènent à de meilleurs metrics finaux, comparativement à ceux utilisés en [7].

Le modèle ainsi entraîné est maintenant capable d'effectuer des prédictions pour 5 sets de données issues du set de validation. Nous décidons ici de mettre en avant la prédiction obtenue pour 1 de ces 5 sets dans la figure 9.

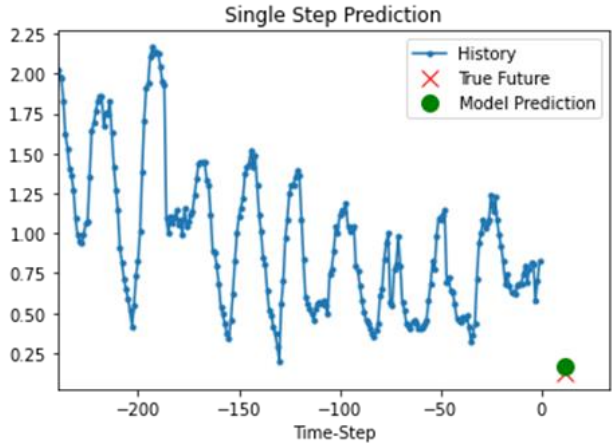

Figure 9 : Prédiction obtenue par le modèle entraîné.

Nous observons que la prédiction pour ce set de données est très bonne. D'autres tests peuvent être menés afin de déterminer les paramètres menant aux meilleurs résultats.

5 – Conclusion

Dans cette ressource ont été mis en avant les fondements des réseaux de neurones récurrents et des séries temporelles. Les principales problématiques liées aux données dynamiques ont donc été abordées, ainsi que leurs principales caractéristiques et donc leur différence avec les données statiques. En suivant le même raisonnement, il a été expliqué pourquoi la majorité des algorithmes applicables aux données statiques ne pouvaient être appliqués aux séries temporelles. Suivant cette explication, différentes architectures de réseaux ont été proposées avec l'objectif d'illustrer clairement le fonctionnement général dans un premier temps, puis spécifique de second ordre.

Cependant, nous ne pouvons que mettre en avant le nombre limité de réseaux neuronaux montrés ici. Il est de la même façon nécessaire de citer d'autres approches tout aussi viables, qui n'ont pas été abordées afin de ne pas augmenter inutilement la complexité de la ressource : Autoregressive-Integrated-Moving-Average (ARIMA), Prophet et Exponential Smoothening (EWMA) sont d'autres approches que nous recommandons d'étudier en parallèle afin d'avoir un aperçu d'autant plus global de la prévision de séries temporelles [6].

Références :

[1]: https://neptune.ai/blog/time-series-prediction-vs-machine-learning

[2]: https://www.kaggle.com/code/ryanholbrook/forecasting-with-machine-learning

[3]: https://en.wikipedia.org/wiki/Recurrent_neural_network

[4]: https://en.wikipedia.org/wiki/Long_short-term_memory

[5]: https://en.wikipedia.org/wiki/Gated_recurrent_unit

[6]: Mahmoud, E., & Pegels, C. C. (1990). An approach for selecting times series forecasting models. *International Journal of Operations & Production Management*

[7]: https://keras.io/examples/timeseries/timeseries_weather_forecasting/

[8]: Dossier Intelligence Artificielle, juin 2022, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/dossier-intelligence-artificielle

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay