



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Time series and XAI

Approches multivariées à travers l'apprentissage non supervisé

Group 6

25 mars 2022

OMAR ALDAKAR
JASON CARNEIRO
LOUIS DUBOIS LEPROU
THÉO LE GALL

Table des matières

1	Introduction	3
2	Contexte du projet	4
3	Approche Algorithmique : Méthode de Matrix Profile	5
3.1	Fonctionnement de la Matrix Profile	5
3.1.1	Série Temporelle	5
3.1.2	Fenêtre, sous-série et motif	5
3.1.3	Matrix Profile	5
3.1.4	Généralisation au cas multivarié	7
3.2	Points faibles de la Matrix Profile	8
3.2.1	Des motifs redondants	8
3.2.2	Des motifs constants	9
3.2.3	Des motifs peu convaincants	9
3.3	Concept de motif intéressant	9
3.3.1	Définitions	10
3.3.2	Cas Univarié	10
3.3.3	Cas multivarié	12
3.4	Résultats Obtenus	13
3.4.1	Cas Univarié	13
3.4.2	Cas multivarié	14
3.5	Choix du nombre de dimensions	15
3.5.1	Réduction du nombre de dimensions	16
3.5.2	Nombre de dimensions : Méthode naïve	16
3.6	Perspectives d'amélioration	18
4	Approche Deep Learning : Utilisation d'Autoencoders pour la detection de motifs	18
4.1	Présentation générale	18
4.2	Spécificités du réseau de neurones	20
4.3	Application du modèle sur un jeu de données test	20
4.3.1	Présentation des données	20
4.4	Application aux séries temporelles	23
4.5	Limites et perspectives d'amélioration	25
5	Conclusion	26
	Bibliographie	27

1 Introduction

Notre projet s'inscrit dans un nouveau cadre et dans une volonté mondiale de pouvoir expliquer les décisions ou les prédictions réalisées par une Intelligence Artificielle. En effet, depuis peu la législation oblige les entreprises réalisant du profilage ou de la prise de décision automatisée sur des personnes, à fournir un droit à l'intervention humaine afin de pouvoir expliquer les décisions prises par un modèle d'apprentissage automatisé. De même, les décideurs d'entreprises ne peuvent se permettre de prendre des décisions sur des recommandations d'un modèle, sans aucune explication ou justification.

Dans un même temps, les modèles de Machine Learning et de Deep Learning sont de plus en plus complexes, cependant ils sont aussi de véritable boîte noire inexplicable.

Ainsi au vu des nouvelles exigences sociétales et des complexités grandissantes des modèles d'apprentissage, le domaine du XAI (eXplainable Artificial Intelligence) se développe afin de rendre les processus de modélisation plus explicable et transparent. Elle a pour but de créer une relation de confiance entre le modèle et l'utilisateur final à travers certains piliers présentés en *figure 1*.



FIGURE 1 – Graphe de connaissance reliant tous les piliers nécessaires à l'établissement d'une relation de confiance entre le modèle et l'utilisateur.[4]

2 Contexte du projet

Le contexte de notre projet s'inscrit à travers le besoin d'une entreprise de logiciel pour le pilotage de performance énergétique. Cette dernière dispose de nombreux capteurs situés sur différentes machines d'une usine de production énergétique. Ces derniers collectent de nombreuses données numériques indexées dans le temps et qui décrivent le fonctionnement des machines. Elle utilise ces dernières en association avec des modèle d'IA à des fins de prédictions et de compréhension. Son besoin aujourd'hui, est de pouvoir expliquer les résultats d'un modèle d'IA d'analyse de séries temporelles, afin de comprendre notamment les écarts pouvant être observés entre la prédition et le réel ou pouvoir tirer une compréhension de ces séries temporelles. Une autre complexité, intervient dans notre projet. Nous sommes amenés, à travailler dans un cadre d'analyse multivariée, c'est-à-dire que nous ne cherchons pas à expliquer une à une chaque série temporelle, mais à expliquer plusieurs séries temporelles dans leur ensemble.

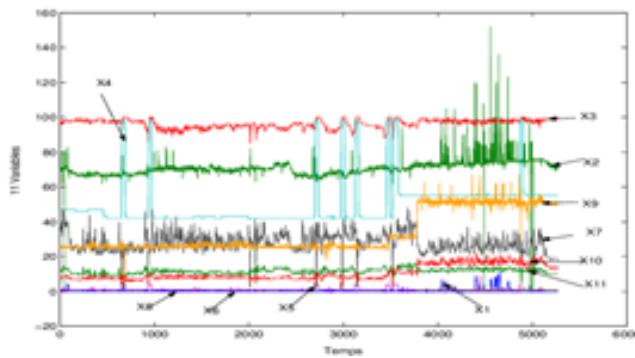


FIGURE 2 – Exemple de série temporelle multivariée.

Ainsi, pour résumer, nous allons devoir définir et réaliser une approche explicable d'un modèle d'analyse de séries temporelles multivariées de notre choix, appliquer sur des données fournies par l'entreprise.

Dans ce rapport nous développerons deux approches de détection de motifs au sein de séries temporelles multivariées : une approche algorithmique et une approche faisant appel au Deep Learning.

3 Approche Algorithmique : Méthode de Matrix Profile

Dans le domaine des séries temporelles, l'algorithme de Matrix Profile permet de détecter des anomalies ou des motifs récurrents au sein d'une ou plusieurs séries temporelles. Cette technique a été introduite en 2016 par Eamonn Keogh à l'université de Californie et Abdullah Mueen à l'université de New Mexico. La méthode Matrix Profile fournit une solution rapide, exacte et cela seulement à partir de quelques paramètres. Tout au long de notre étude, nous nous sommes appuyés sur l'article [2] et nous avons décidé d'utiliser la bibliothèque Python STUMPY [3], qui est une bibliothèque très récente fournissant un grand nombre d'outils optimisés et scalables pour l'étude de séries temporelles multivariées et la découverte de motifs.

3.1 Fonctionnement de la Matrix Profile

Dans cette section nous allons introduire quelques définitions et notations nécessaire à la compréhension de la méthode de Matrix Profile.

3.1.1 Série Temporelle

Une série temporelle est une suite finie $[x_1, x_2, \dots, x_n]$ de données (dans notre cas ces données sont numériques) indexées par le temps.

3.1.2 Fenêtre, sous-série et motif

Une fenêtre de taille m sur une série temporelle $X = [x_1, x_2, \dots, x_n]$ est un cadre d'observation qui va permettre d'isoler m observations de X , afin d'en obtenir une sous-série $[x_k, \dots, x_{k+m}]$ tel que $0 \leq k \leq n - m$.

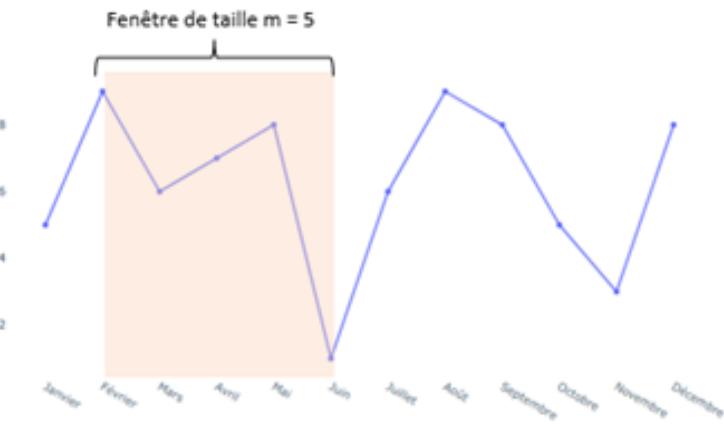


FIGURE 3 – Exemple de fenêtre.

Ainsi on définit un motif comme étant une sous-série apparaissant plusieurs fois au sein d'une même série temporelle. Inversement une anomalie est une sous-série avec une faible fréquence d'apparition.

3.1.3 Matrix Profile

L'algorithme Matrix Profile détecte les motifs et les anomalies au sein d'une série temporelle en comparant deux sous-séries sur une même fenêtre de taille m à partir de la distance Euclidienne Z-normalisée définie de la manière suivante :

Soit X et Y deux sous-séries temporelles de taille m alors :

$$d(X, Y) = \sqrt{2 * m * \left(1 - \frac{\sum_{i=0}^m x_i y_i - m\mu_x \mu_y}{m\sigma_x \sigma_y} \right)} \quad (1)$$

$$\text{où } \mu_x = \sum_{i=0}^m \frac{x_i^2}{m} \text{ et } \sigma_x^2 = \sum_{i=0}^m \frac{x_i^2}{m} - \mu_x^2 \quad (2)$$

$$\text{corrPearson}(X, Y) = \frac{\sum_{i=0}^m x_i y_i - m\mu_x \mu_y}{m\sigma_x \sigma_y} \quad (3)$$

La corrélation de Pearson (3) est un indice reflétant une relation linéaire entre deux variables continues. Ce coefficient normalise et centre chaque sous-série pour comparer leur évolution au cours du temps et ainsi définir une similarité.

Maintenant que la distance a été définie, la stratégie est de sélectionner dans un premier temps une sous-série de référence (fenêtre verte sur la *figure 4*) de taille m dans notre série temporelle et de calculer la distance euclidienne Z-normalisée entre cette sous-série de référence et toutes les autres sous-séries de taille m , comme le montre la *figure 4*.

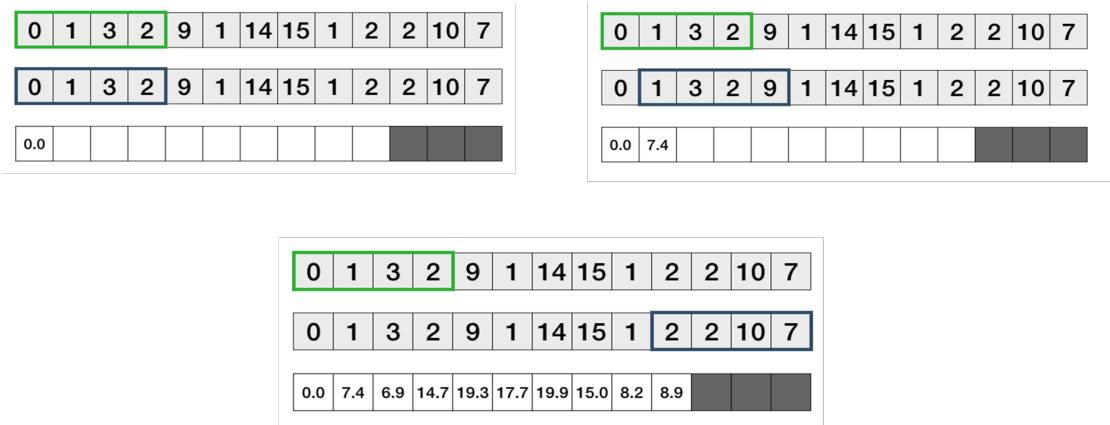


FIGURE 4 – Calcul de la distance entre une sous-série de référence et toutes autres sous-série de taille $m=4$.[3]

En décalant à chaque fois la sous-série de référence d'un indice et en répétant le processus, nous pouvons donc obtenir une matrice stockant toutes les distances entre chaque sous-série. Ainsi le coefficient (i,j) de la matrice de distances, est la distance entre la sous-série commençant à l'indice i et la sous-série commençant à l'indice j .

Distance Matrix

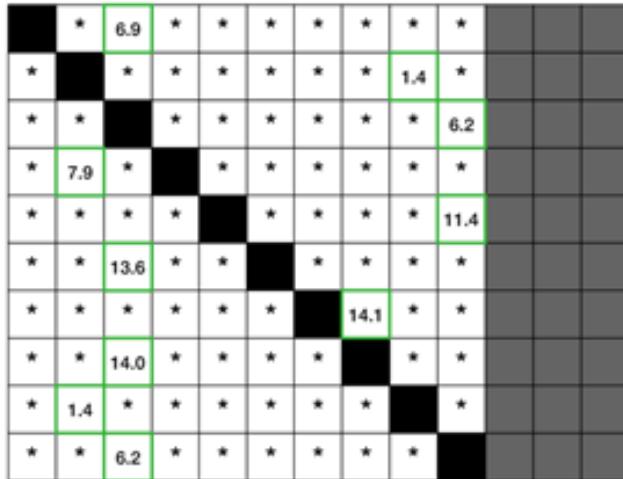


FIGURE 5 – Exemple de matrice de distances obtenue.[3]

On obtient alors la Matrix Profile en récupérant uniquement la plus petite valeur de chaque ligne de la matrice de distances. Ainsi les couples d'indices définissant les potentiels motifs sont les indices de la matrice où la plus petite valeur de distance a été obtenue sur chaque ligne.

Matrix Profile

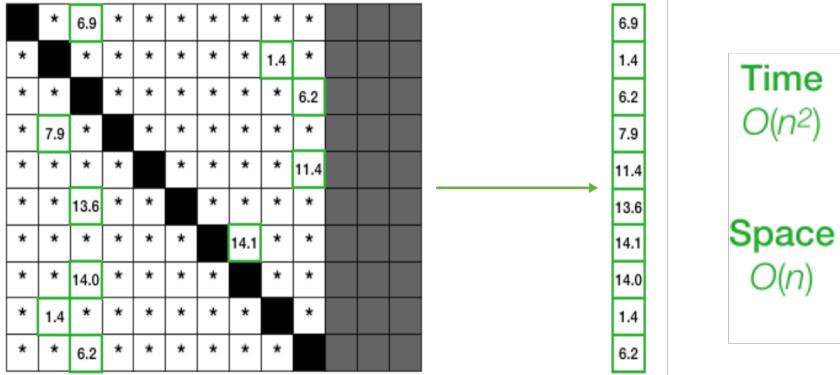


FIGURE 6 – Exemple de Matrix Profile. [3]

3.1.4 Généralisation au cas multivarié

Dans le cas d'une série temporelle multivariée $X = [X^{(1)}, X^{(2)}, \dots, X^{(d)}]$ à d dimensions, c'est à dire que d séries temporelles univariées évoluent en même temps et doivent être étudiées dans leur ensemble, le calcul de la Matrix Profile est similaire au cas univarié.

Une sous-série $X_{t,m} = [X_{t,m}^{(1)}, X_{t,m}^{(2)}, \dots, X_{t,m}^{(d)}]$ de X est un regroupement de sous-séries univariées, de longueur m et commençant à l'indice t . Le calcul de la distance entre deux sous-séries multivariées est simplement l'agrégation des distances des deux sous-séries sur chacune de leur dimension.

Soit $X_{t,m} = [X_{t,m}^{(1)}, X_{t,m}^{(2)}, \dots, X_{t,m}^{(d)}]$ et $Y_{t,m} = [Y_{t,m}^{(1)}, Y_{t,m}^{(2)}, \dots, Y_{t,m}^{(d)}]$ deux sous-séries multivariées alors la distance de ces deux sous-séries est :

$$dist(X_{t,m}, Y_{t,m}) = \sum_{i=1}^d dist(X_{t,m}^{(i)}, Y_{t,m}^{(i)}) \quad (4)$$

avec $dist$ la distance définie en (1).

Ainsi à travers la définition de cette distance nous pouvons reproduire le processus de calcul de la Matrix Profile défini dans le cas univarié au cas multivarié.

3.2 Points faibles de la Matrix Profile

Même si cet algorithme permet de bien réaliser une première tentative de découverte de motifs, nous avons observé quelques faiblesses.

3.2.1 Des motifs redondants

Comme expliqué au-dessus, la Matrix Profile calcule la distance Euclidienne Z-normalisée entre une sous-série de référence et toutes les autres sous-séries, en se décalant d'indice en indice. Cependant lorsque l'on décale la fenêtre de seulement quelques indices par rapport à la sous-série de référence, la distance calculée peut être très faible du fait que les deux sous-séries partagent une grande majorité de leurs observations, entraînant donc la détection d'un motif comme le montre la figure 7.

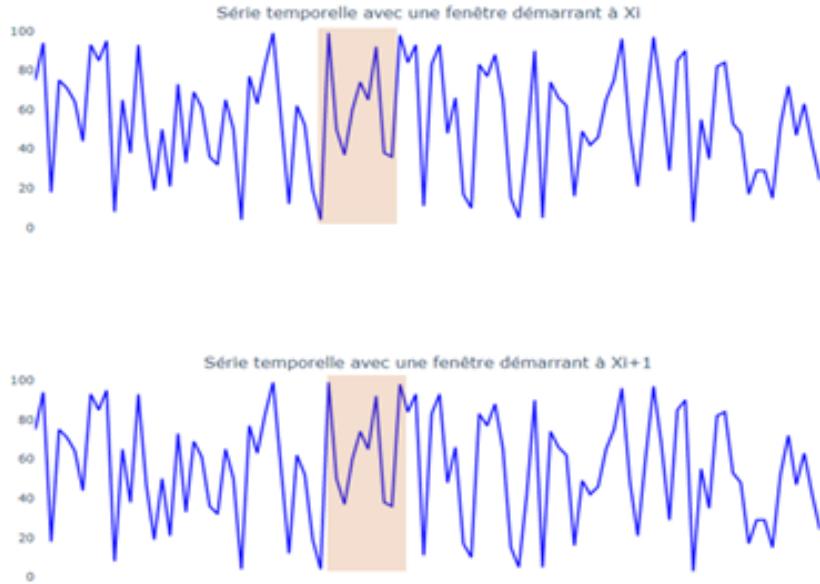


FIGURE 7 – Exemple d'un motif redondant après le décalage d'un indice de la fenêtre.

Un autre type de motif redondant est que plusieurs sous-séries consécutives renvoient à une même sous-série décalée ou non de quelques indices. Par exemple une sous-série d'indice 200 est associée à une sous-série d'indice 300 pour faire un motif, de même la sous-série d'indice 201 est associée à la sous-série d'indice 300 ou 301 et ainsi de suite. Nous considérons que c'est en fait un seul et même motif et nous souhaitons donc conserver que sa meilleure représentation qui est celle qui offre la plus petite distance de motif.

Ainsi on définit un motif distinct comme étant une sous-série qui apparaît plusieurs fois dans la série temporelle et dont les sous-séries qui sont rattachées à ce motif, ne partagent pas de mêmes

observations.

Désormais nous prendrons en compte uniquement les motifs distincts de la Matrix Profile afin d'éviter la répétition de motifs redondants.

3.2.2 Des motifs constants

Dans certains cas de figure la méthode Matrix Profile va détecter un motif au sein de notre série temporelle correspondant à différents enchaînements de valeurs constantes. Cependant ces motifs sont intérressants car il est très difficile d'en tirer des informations.

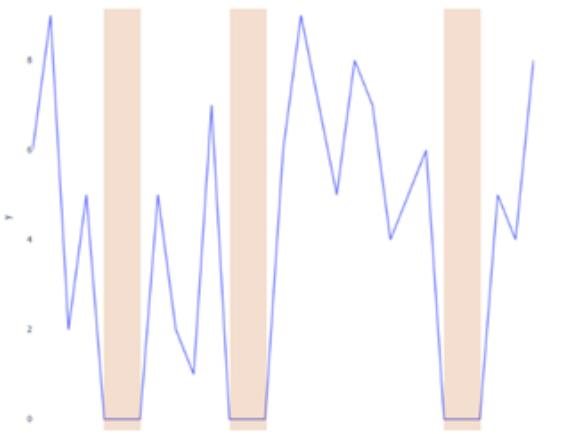


FIGURE 8 – Exemple de motifs constants.

3.2.3 Des motifs peu convaincants

Dans quelques situations, il se peut que la série temporelle ne présente aucun motif. Cependant l'algorithme Matrix Profile va toujours associer une sous-série à une autre, à partir de la plus petite distance calculée alors que les deux ne se ressemblent pas et ne peuvent donc constituer un motif. Il manquerait donc à l'algorithme de Matrix Profile une distance seuil qui permettrait d'éliminer les motifs identifiés avec une distance supérieure à cette distance seuil.

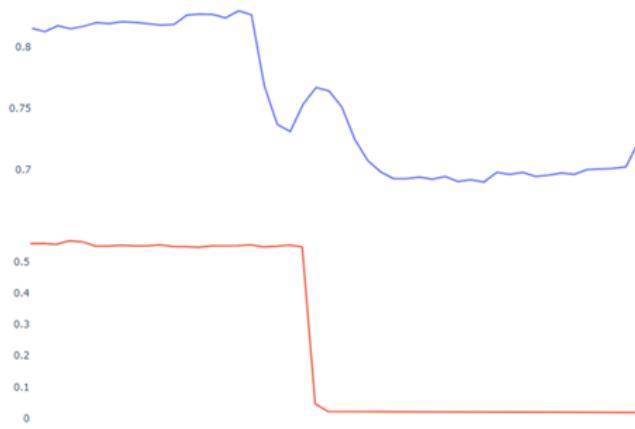


FIGURE 9 – Exemple de motif non convaincant.

3.3 Concept de motif intéressant

Dans cette partie nous développons le concept de motif intéressant, qui est un concept issu de notre propre réflexion au regard des quelques faiblesses que nous avons pu observer sur la méthode Matrix Profile.

3.3.1 Définitions

Définition 1 : La qualité d'un motif est désignée par la distance calculée entre la sous-série représentant ce motif et son plus proche voisin. Ainsi plus la sous-série représentant un motif et son plus proche voisin sont similaires, plus la distance est proche de 0 et donc plus le motif est de bonne qualité.

$$A_1 \text{ est de meilleur qualité que } A_2 \text{ ssi } d(A_1, A_{1\text{ plus proche voisin}}) < d(A_2, A_{2\text{ plus proche voisin}}) \quad (5)$$

Définition 2 : La variation d'un motif est caractérisée par la valeur de son écart-type sur l'intervalle de temps considéré.

Définition 3 : Nous appelons motif intéressant un motif ayant une variation dépassant un certain seuil, afin d'éviter tous motifs constants ou à faible variation :

$$\sigma > \sigma_{seuil} \quad (6)$$

De même un motif intéressant ce doit d'avoir une qualité minimum :

$$d < d_{seuil} \quad (7)$$

Ainsi définir un motif intéressant, c'est définir un seuil de variation σ_{seuil} , ainsi qu'un seuil de qualité d_{seuil} .

3.3.2 Cas Univarié

Soit T une série univariée de longueur n . La longueur d'une sous séquence ou sous-série de T est désignée par m .

Établir σ_{seuil}

Afin d'établir notre σ_{seuil} , nous avons imaginé un motif ayant une variation minimale mais suffisante pour que nous le considérons comme intéressant. Ce motif aurait pour forme une évolution constante autour d'une valeur c sur toutes les observations, excepté sur une où la valeur serait écartée de e par rapport à c , comme le montre la figure 10.

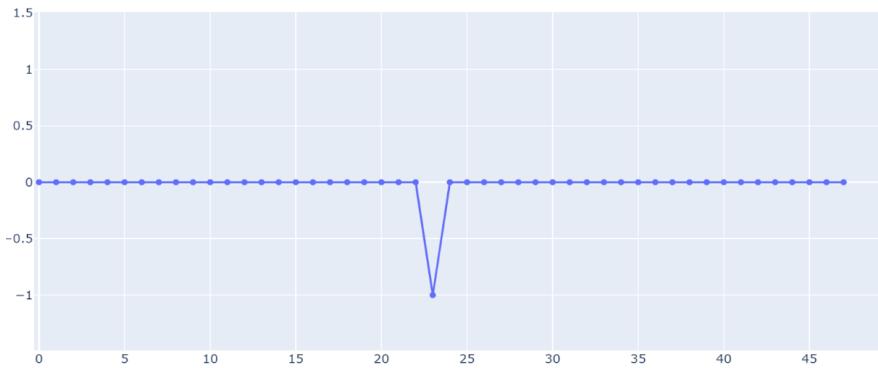


FIGURE 10 – Exemple d'un motif ayant une variation minimale mais suffisante pour être considéré comme intéressant.

Ainsi l'écart-type de ce genre de motif est égal à :

$$\sigma = \sqrt{\frac{1}{m} \sum (X_i - \mu)^2} = \sqrt{\frac{1}{m} \times ((m-1) (c - \frac{cm \mp e}{m})^2 + (c \mp e - \frac{cm \mp e}{m})^2)} \quad (8)$$

$$\sigma = \frac{e}{m} \times \sqrt{m-1} \quad (9)$$

Notre σ_{seuil} est donc défini par uniquement l'amplitude de la variation e que nous fixerons par la suite comme étant l'écart-type de la série globale étudiée :

$$\sigma_{seuil} = \frac{\sqrt{m-1}}{m} \sigma_{global} \quad (10)$$

Ainsi cette définition du σ_{seuil} permet une comparaison facile à l'écart-type d'un motif quelconque sans aucune transformation, mais uniquement par connaissance de l'écart-type de la série temporelle σ_{global} .

Établir d_{seuil}

Afin d'établir notre d_{seuil} , il nous a fallu dans un premier temps définir la similarité minimum à avoir entre deux motifs à travers la proposition suivante :

Soit $X_t = [x_t, x_{t+1}, \dots, x_{t+m}]$ et $Y_t = [y_t, y_{t+1}, \dots, y_{t+m}]$ deux sous-séries temporelles, de longueur m et commençant à l'indice t , et c une constante définissant l'écart de mesure accepté sur chaque observation du motif.

On dit que X_t est similaire à Y_t si $\forall i \in [t, \dots, t+m]$, $y_i - c \leq x_i \leq y_i + c$.

Ainsi afin d'établir notre distance seuil, nous prenons un motif trouvé lors du calcul de la matrix profile, que nous appelons motif de référence que nous centrons et réduisons afin d'obtenir le même d_{seuil} pour chaque motif. Puis nous créons une nouvelle sous-série en se basant sur le motif de référence mais où chaque observation du motif est décalée aléatoirement de $-c$, c ou 0. Nous calculons ensuite la distance entre le motif de référence et cette nouvelle sous-série en utilisant la distance euclidienne Z-Normalisée. On répète le processus sur l'ensemble des motifs distincts identifiés lors du calcul de la matrix profile. La distance seuil est alors définie comme étant la moyenne des distances calculées.

Dans notre cas nous avons essayé de déterminer une bonne constante c .

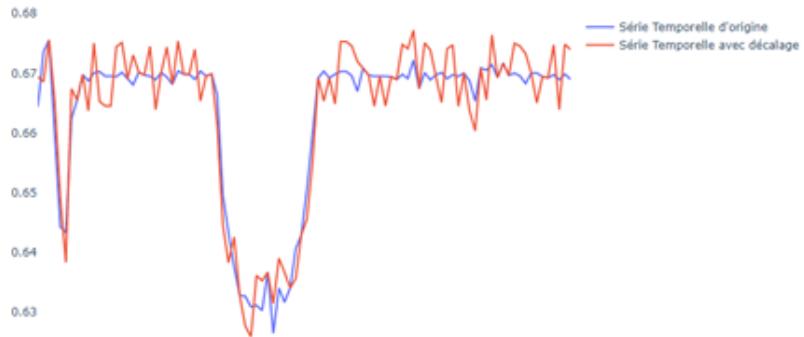


FIGURE 11 – Exemple de 2 série temporelles similaires avec un $c=0.005$.

Comme on peut le voir sur la figure 11, $c = 0.005$ semble être satisfaisant, les deux sous-séries ont les mêmes formes sans pour autant être identiques et on obtient une distance euclidienne Z-Normalisée relativement faible de 2,98.

3.3.3 Cas multivarié

Soit $T = [T^{(1)}, T^{(2)}, \dots, T^{(d)}]$ une série multivariée de longueur n et de dimension d . La longueur d'une sous séquence de T est désignée par m .

Établir σ_{seuil}

De façon similaire au cas univarié, nous avons défini notre σ_{seuil} , en imposant une variation minimum sur chaque dimension du motif multivarié. Ainsi, ce motif aurait pour forme minimale une évolution constante autour d'une valeur c sur toutes les observations, excepté sur une où la valeur serait écartée de e par rapport à c , et ce pour chaque dimension du motif comme le montre la figure 12.

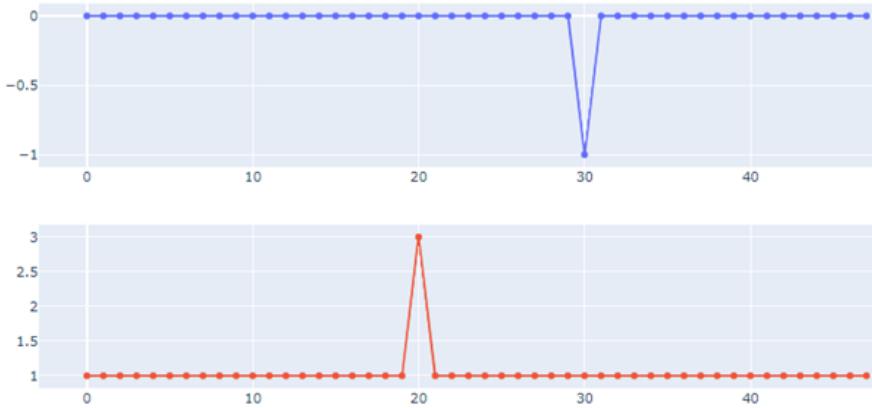


FIGURE 12 – Exemple d'un motif à $d=2$ dimensions ayant une variation minimale mais suffisante pour être considéré comme intéressant.

Ainsi un σ_{seuil} est établi pour chaque dimension :

$$\sigma_{seuil}^{(i)} = \frac{\sqrt{m-1}}{m} \sigma_{global}^{(i)} \quad (11)$$

Établir d_{seuil}

De façon similaire au cas univarié, nous avons défini la similarité minimum à avoir entre deux sous-séries multivariées pour être considérée comme un motif intéressant à travers la proposition suivante :

Soit $X_{t,m} = [X_{t,m}^{(1)}, X_{t,m}^{(2)}, \dots, X_{t,m}^{(d)}]$, $Y_{t,m} = [Y_{t,m}^{(1)}, Y_{t,m}^{(2)}, \dots, Y_{t,m}^{(d)}]$ deux sous-séries temporels de d dimensions et m valeurs, commençant à l'indice t . Une constante c définit l'écart de mesure accepté sur chaque observation.

$$X_{t,m}^{(j)} = [x_t^{(j)}, x_{t+1}^{(j)}, \dots, x_{t+m}^{(j)}]$$

$$Y_{t,m}^{(j)} = [y_t^{(j)}, y_{t+1}^{(j)}, \dots, y_{t+m}^{(j)}]$$

On dit que $X_{t,m}$ est similaire à $Y_{t,m}$ si :

$$\forall j \in [1, \dots, d] \text{ et } \forall i \in [t, \dots, t+m], \quad y_i^{(j)} - c \leq x_i^{(j)} \leq y_i^{(j)} + c.$$

Ainsi afin d'établir notre distance seuil, nous prenons un motif multivarié trouvé lors du calcul de la matrix profile, que nous appelons motif de référence. Puis nous créons un nouveau motif multivarié se basant sur le motif de référence mais où chaque observation du motif sur chaque

dimension est décalée aléatoirement de $-c$, c ou 0. Nous calculons ensuite la distance entre le motif de référence et ce nouveau motif en utilisant la distance euclidienne Z-Normalisée. On répète le processus sur l'ensemble des motifs distincts identifiés lors du calcul de la Matrix Profile. La distance seuil est alors définie comme étant la moyenne des distances calculées.

3.4 Résultats Obtenus

Dans cette partie nous présentons les différents résultats que nous avons obtenu à travers notre programme Python permettant de supprimer les motifs que nous avons défini comme redondant et de ne conserver uniquement les motifs que nous entendons intéressants dans le cas univarié et multivarié.

3.4.1 Cas Univarié

Nous avons donc appliqué en complément de la Matrix Profile obtenu à travers STUMPY notre définition de motif intéressant en calculant et fixant une distance et une variance seuil. Nous avons fait varier la fenêtre d'observation d'un motif ainsi que la période totale de la série temporelle afin d'observer l'évolution du temps d'exécution de notre programme.

Fenêtre de 8H avec $d_{seuil} = 3.05$ et $\sigma_{seuil} = 2.35 * 10^{-2}$

	6 mois	1 ans	2 ans
Nombre de motifs intéressants moyen	9	15	19
Temps d'exécution moyen (secondes)	3.51	8.67	31.27

Fenêtre de 16H avec $d_{seuil} = 3.05$ et $\sigma_{seuil} = 2.35 * 10^{-2}$

	6 mois	1 ans	2 ans
Nombre de motifs intéressants moyen	12	16	23
Temps d'exécution moyen (secondes)	4.22	7.95	31.55

Ainsi on observe que pour le même d_{seuil} et σ_{seuil} les résultats obtenus avec une fenêtre de 16h sont meilleurs puisque nous avons recueillis plus de motifs intéressants avec un temps d'exécution similaire.

Nous sommes très satisfaits des résultats obtenus dans le cas univarié car les motifs que nous obtenons sont en accord avec la définition de motif intéressant que nous avons faite comme le montre la *figure 13*. Le motif est non constant, possède une variation significative et les deux sous-séries sont très similaires.

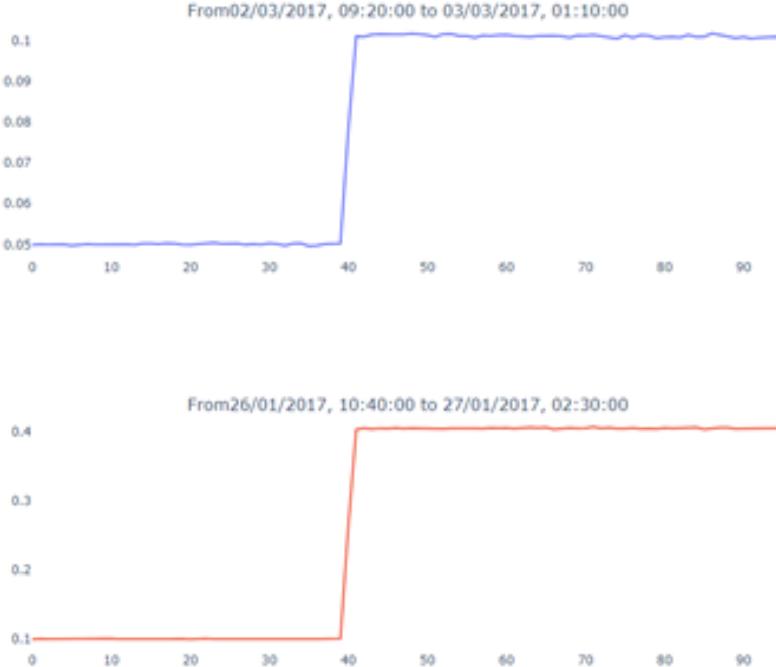


FIGURE 13 – Exemple motif à 1 dimension.

À partir des résultats obtenus au-dessus, toutes nos analyses seront basées sur une période de 6 mois (afin de limiter le temps de calcul au vu des ressources qui sont à notre disposition), une fenêtre de 16 heures, $d_{seuil}=0.5$ et $\sigma_{seuil}=2.35 \times 10^{-2}$.

3.4.2 Cas multivarié

Nous avons adapté notre programme appliqué au cas univarié au cas multivarié afin de pouvoir de même observer si les résultats qui en résultent sont en accord avec la définition de motif intéressant. Notre programme fonctionne quel que soit le nombre de dimension de la série temporelle fournie. La *figure 14* présente quelques résultats obtenus.

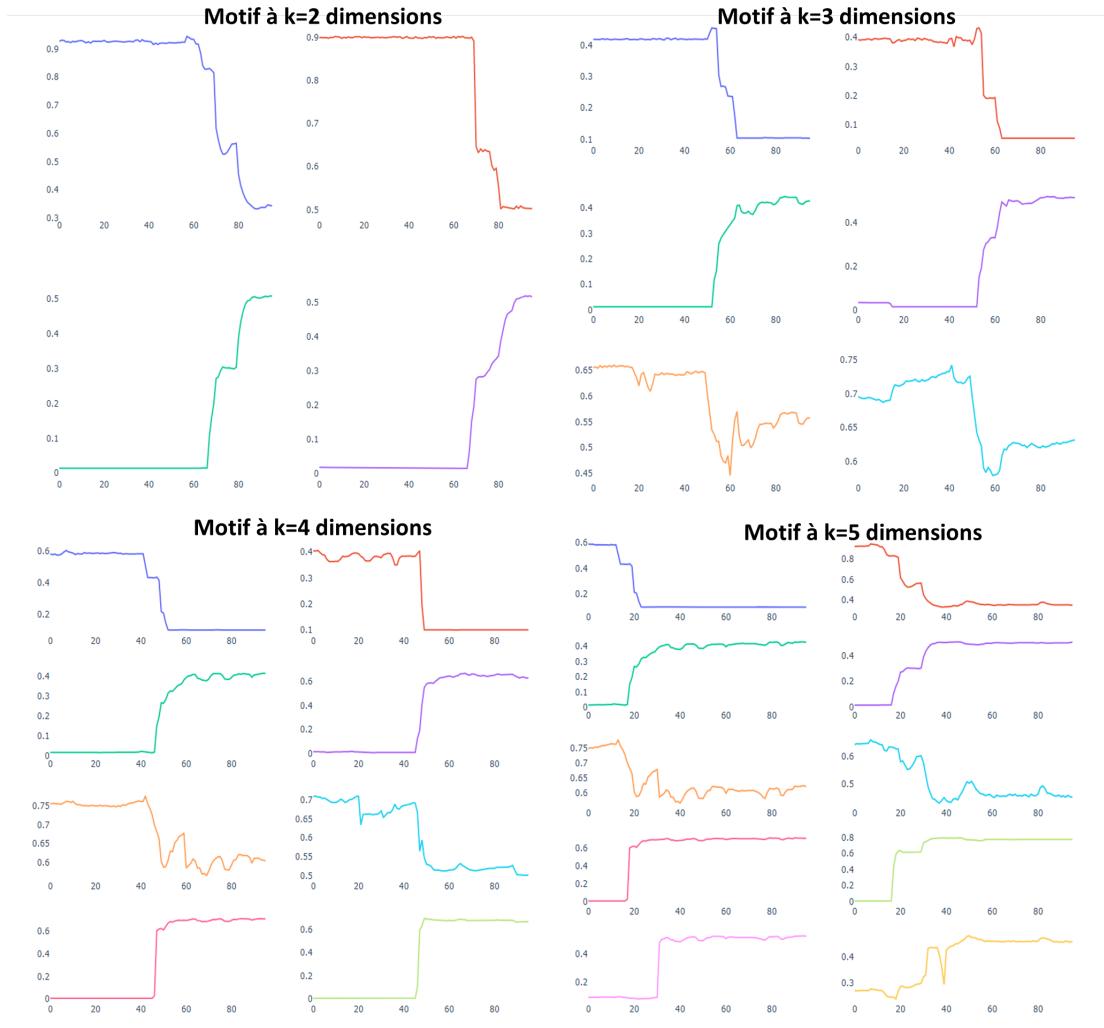


FIGURE 14 – Exemple de motifs obtenus pour une fenêtre de 16 heures et sur une série temporelle de 6 mois, pour respectivement 2, 3, 4, 5 dimensions.

La encore nous sommes très satisfaits des motifs qui ressortent de notre programme. En effet, nous observons à chaque fois des variations sur chaque dimension du motif ainsi qu'une forte similarité entre les sous-séries. Quasiment aucun motif que nous avons considéré comme redondant apparaît. Nous avons été de même surpris de continuer à observer des motifs intéressants jusqu'en dimension 8.

3.5 Choix du nombre de dimensions

Pour rappel nous disposons de 92 capteurs sur la chaudière GNF, ce qui signifie que nous disposons d'une série temporelle à 92 dimensions. Il serait vain de vouloir trouver un motif sur autant de dimension, les résultats garantissent à coup sûr d'obtenir des motifs de très mauvaises qualités sans aucun sens. En effet nous faisons face à un problème courant et complexe qui est de savoir parmi les dimensions que nous possédons lesquelles sont pertinentes pour obtenir des motifs interprétables ou encore combien de dimensions sont nécessaires.

Il faut garder en tête qu'en rajoutant une dimension, on perd potentiellement des motifs. En effet comme le montre la figure 16, en considérant les deux premières séries on trouve un motif qui n'existe pas si l'on considère les 3 séries.

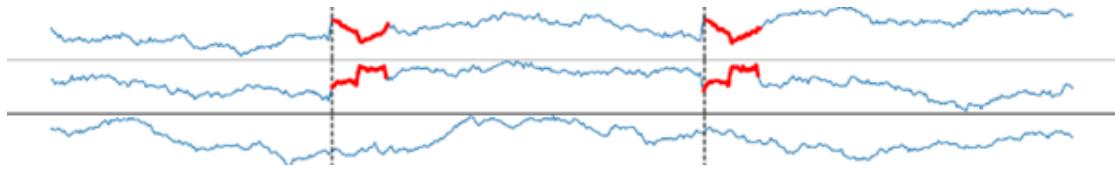


FIGURE 15 – Exemple de motif sur 2 dimensions qui disparaît à l'ajout d'une 3ème dimension.[2]

3.5.1 Réduction du nombre de dimensions

Nous avons dans un premier temps décider de réduire le nombre de dimensions en retirant les séries temporelles comportant plus de 3% de valeurs manquantes afin de se concentrer sur les données les plus riches en informations. Nous sommes alors passé de 92 à 20 dimensions et c'est sur celles-là que nous nous concentrerons le reste du rapport.

3.5.2 Nombre de dimensions : Méthode naïve

Pour répondre à la question de combien de dimensions sont nécessaires lorsque nous avons une série temporelle multivariée à d dimensions, une méthode naïve est de calculer dans un premier temps les Matrix Profil pour 1, 2, 3,..., d dimensions. Pour obtenir la Matrix Profile pour i dimensions, avec $i < d$, lorsque l'on a une série à d dimensions, il faut calculer la Matrix Profile pour toutes les combinaisons possibles à i dimensions puis pour chaque indice récupérer la valeur minimale parmi toutes les combinaisons.

Un exemple ci-dessous pour une série multivariée à 3 dimensions :

Soit $T = [T^{(1)}, T^{(2)}, T^{(3)}]$ une série multivariée à 3 dimensions. Si je souhaite obtenir la Matrix Profile sur 2 dimensions de T (soit $MP_{2d}(T)$), il faut calculer la Matrix Profile sur 2 dimensions de toutes les combinaisons possibles des dimensions de T , soit $MP_{2d}([T^{(1)}, T^{(2)}]), MP_{2d}([T^{(1)}, T^{(3)}])$ et $MP_{2d}([T^{(2)}, T^{(3)}])$. Ainsi pour chaque index k $MP_{2d}(T)_k = \min_{i,j} MP_{2d}([T^{(i)}, T^{(j)}])_k$

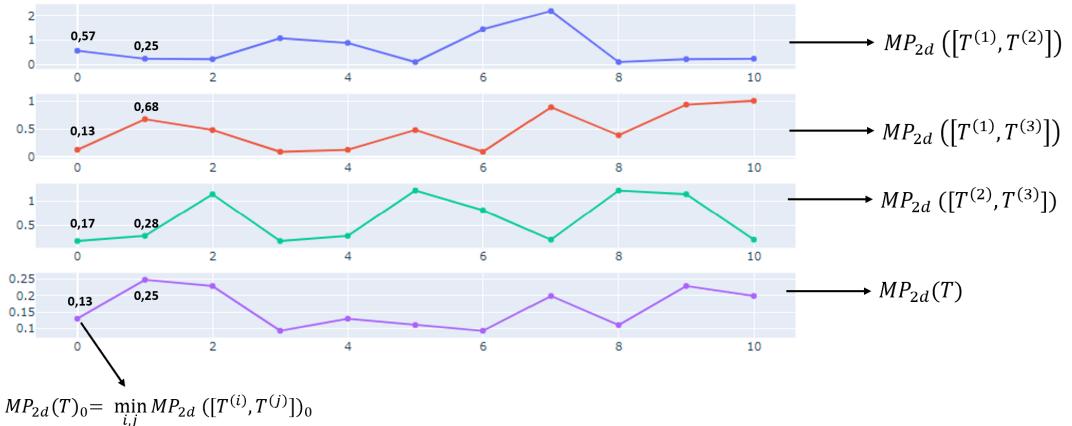


FIGURE 16 – Exemple de calcul d'une Matrix Profile à 2 dimensions sur une série multivariée à 3 dimensions.

Une fois les calculs de la Matrix Profil obtenus pour toutes les k -dimensionnel série avec k compris entre 1 et d , on récupère la valeur minimale de chacune de ces Matrix Profile et on trace la courbe qui en résulte.

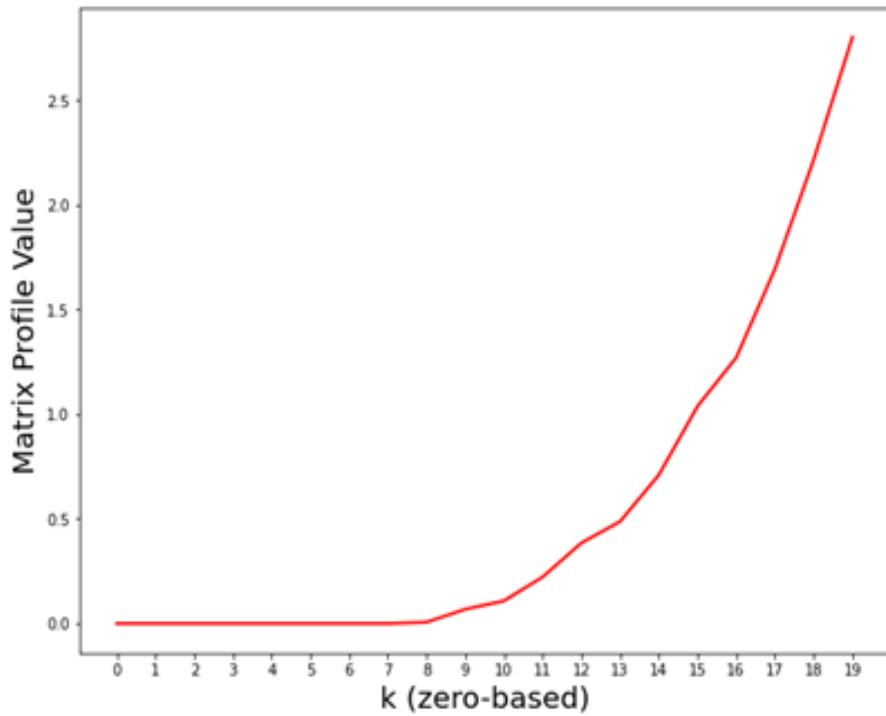


FIGURE 17 – Exemple de courbe obtenue en récupérant la valeur minimale de chaque Matrix Profile pour chaque nombre de dimensions.

On se ramène alors à un problème de localisation d'un "coude", c'est-à-dire qu'il faut identifier le point de courbure maximum. Le nombre de dimensions pour lequel est atteint ce point de courbure maximum est alors le nombre de dimensions à conserver pour obtenir les motifs les plus pertinents sur la série multivariée. Dans l'exemple de la *figure 17*, ce point de courbure se situe pour $k=8$. Ainsi les motifs les plus pertinents de cette série multivariée à 20 dimensions sont obtenus en réalisant l'étude entre 1 et 8 dimensions.

Cependant plus la dimension est grande plus le temps de calcul augmente, voici *figure 18* le temps d'exécution de notre algorithme pour chaque dimension avec un ordinateur portable de 4.0 Go de ram :

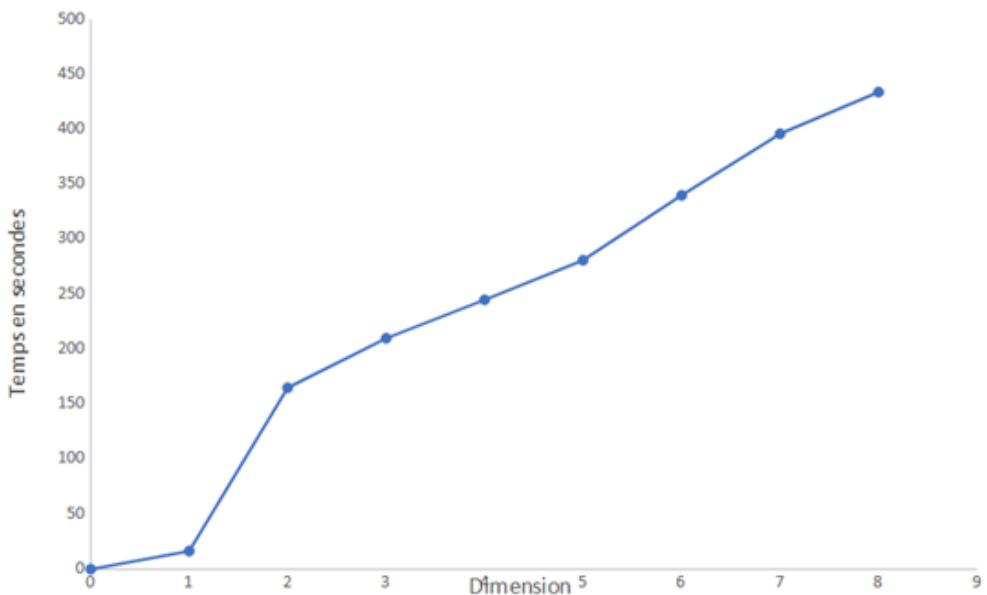


FIGURE 18 – Temps d'exécution de la Matrix Profile en fonction de la dimension.

3.6 Perspectives d'amélioration

Comme nous l'avons expliqué, il est très rare de trouver un motif qui apparaît simultanément en considérant toutes les dimensions. Il faut donc les filtrer afin de trouver un sous-ensemble de dimensions qui nous donnera les meilleurs motifs.

Il en ressort alors une question d'optimisation, en effet il faut analyser des combinaisons de séries temporelles (très souvent de grandes tailles) ce qui peut devenir très couteux en termes de temps d'exécution et de stockage.

Cette problématique est un sujet d'actualité à part entière, il pourrait donc être intéressant de s'y pencher afin d'obtenir les meilleurs motifs intéressants multivariées ainsi que les séries à analyser et donc avoir une meilleure compréhension de comment interagissent les séries temporelles entre elles.

4 Approche Deep Learning : Utilisation d'Autoencoders pour la detection de motifs

Cette partie se base sur une approche Deep Learning faisant appel aux Autoencoders. Elle a pour but de détecter les motifs afin de tirer une compréhension de nos séries temporelles multivariées.

4.1 Présentation générale

Au cours du projet, c'est l'article [1] qui a été étudié en profondeur. L'article présente une architecture de réseau *figure 19* neuronal convolutif qui vise à extraire des motifs récurrents d'une série temporelle multivariée. Pour rappel, une série temporelle multivariée est simplement une concaténation de plusieurs séries temporelles se produisant au même moment. Pour être plus précis, l'architecture est un autoencodeur, c'est-à-dire un réseau neuronal qui cherche à reproduire ce qui est donné en entrée en passant par un espace de données de dimension inférieure que

l'on appelle espace latent. L'article précise que les motifs sont extraits des filtres de convolution et de déconvolution de ce réseau.

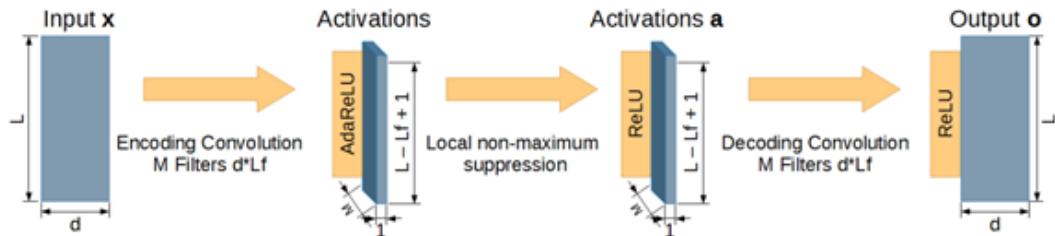


FIGURE 19 – Architecture de l'Autoencoder.[1]

L'article souhaite aborder un problème d'interprétabilité des autoencoders convolutifs. Le problème est que le nombre de motifs trouvés correspond au nombre de filtres de convolution définis manuellement, de sorte que les motifs sont souvent redondants et corrélés. Afin de récupérer des motifs propres, l'article introduit différents éléments dans l'architecture, notamment une "fonction" adaptative rectifiée (adarelu) qui vise à améliorer l'interprétabilité des motifs et un régularisateur qui aide à trouver automatiquement le nombre pertinent de motifs. L'article illustre le besoin de ces éléments sur des données synthétiques qui sont des images avec des motifs récurrents.



FIGURE 20 – Exemple de série utilisée par [1]

Il faut voir ces images comme des séries temporelles, ou chaque ligne de l'image est une série temporelle ayant 500 valeurs variant entre 0 et 255.

Une des grandes faiblesses de cet article est le manque de clarté sur la façon d'interpréter les filtres, en effet nous avons d'abord pensé que les filtres peuvent être visualisés directement en prenant la valeur des poids comme la valeur d'un pixel, en faisant cela nous n'avons pas pu trouver les résultats de l'article. Nous avons d'abord implémenté un code nous permettant de trouver l'image maximisant la sortie d'un filtre de convolution. Nous avons ensuite implémenté une fonction nous permettant de visualiser un filtre de déconvolution en mettant une grande valeur à l'élément correspondant à ce filtre dans l'espace latent et en donnant cette entrée à notre réseau de déconvolution.

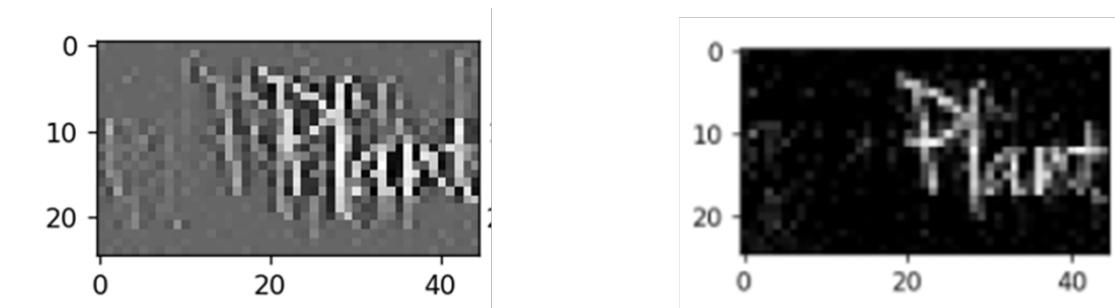


FIGURE 21 – Visualisation d'un filtre de convolution (à gauche) et d'un filtre de déconvolution (à droite).

4.2 Spécificités du réseau de neurones

Comme précisé précédemment ce réseau a été pensé avec une approche IA interprétable, pour se faire l'auteur a introduit différents mécanismes permettant d'avoir des filtres interprétables.

Imposer des filtres de décodage non négatifs. Comme la sortie de l'AE est définie comme une combinaison linéaire des filtres de décodage, alors ces filtres peuvent représenter. Ainsi, comme l'entrée est non-négative, les poids des filtres de décodage sont forcés d'être non-négatifs en les seuillant à chaque itération SGD.

La sparsification des filtres. La régularisation traditionnelle L2 permet de nombreuses valeurs petites mais non nulles. Pour forcer ces valeurs à zéro et ainsi obtenir des filtres plus sparses, la norme L2 est remplacée par la norme L1 de promotion de la sparsité connue sous le nom de lasso. Un coefficient λ_{lasso} est alors introduit plus il est important plus les coefficients sont forcés à valoir 0.

Encourager les activations éparses. La divergence KL traditionnelle vise à rendre toutes les "hidden units" également utiles en moyenne, alors que notre objectif est de faire en sorte que la couche d'activation soit aussi épars que possible pour chaque document d'entrée donné. Pour se faire l'article encourage les activations maximales, c'est-à-dire de faible entropie. L'article introduit alors un coefficient de régularisation λ_{ent} .

Trouver le nombre réel de motifs. L'un des principaux avantages et apport de cet article est la possibilité de découvrir le "vrai" nombre de motifs. Une façon d'y parvenir est d'introduire un grand nombre de filtres dans le réseau et d'espérer que l'apprentissage ne conduise qu'à quelques filtres non nuls capturant les motifs intéressants. Cependant, en pratique, les termes de régularisation et les optimisations standard ont tendance à produire des réseaux qui "utilisent" tous les filtres ou plusieurs d'entre eux, ce qui donne des modèles moins interprétables. Pour surmonter ce problème, l'article propose un terme de régularisation λ_{grp} , qui pénalise les modèles comportant de nombreux filtres non nuls.

Gestion des corrélations de filtres distants avec AdaReLU. La couche gaussienne ne peut pas gérer les corrélations non locales (dans le temps). Pour gérer cela, l'article remplace la fonction d'activation traditionnelle ReLU par une nouvelle fonction appelée ReLU adaptative. AdaReLU travaille sur des groupes d'unités et met à 0 toutes les valeurs qui sont inférieures à un pourcentage (par exemple, 60%) de la valeur maximale du groupe.

4.3 Application du modèle sur un jeu de données test

La partie théorique étant établie, nous pouvons désormais étudier plus en détail le code de l'article. Comme nous avons pu le voir, le modèle dispose de beaucoup d'hyperparamètres qui ont tous un impact plus ou moins élevé sur les performances du modèle et sur la visualisation des patterns. L'article détaille l'utilité d'une partie d'entre eux, mais dans un souci de compréhension nous allons lancer plusieurs expérimentations pour mieux appréhender le comportement du modèle. Pour cela, nous allons dans un premier temps utiliser le jeu de données utilisé dans l'article. Celui-ci est composé de pattern sous forme de mots manuscrits, il a l'avantage d'être très visuel et donc facilement interprétable. Ensuite nous utiliserons ces résultats pour les appliquer sur nos séries temporelles multivariées.

4.3.1 Présentation des données

Le code de l'article fournit en effet un jeu de données. Ce jeu de données est un ensemble de mots manuscrits qui sont ajoutés ensemble les uns à la suite des autres. L'idée est alors d'utiliser une suite de mots pour former une série qui ira en entrée du modèle *figure 19*.

Le modèle va alors s'entraîner à reconstituer cette série et nous pourrons, à la fin de l'entraînement, extraire les filtres de déconvolution du modèle. Ces filtres, en théorie, sont censés contenir

les patterns détectés par le modèle, il s'agit ici des mots utilisés dans la série : on s'attend alors à avoir un mot filtre.

Dans un premier temps, nous allons jouer avec le nombre de filtres utilisés par le modèle. Nous allons augmenter progressivement le nombre de filtres pour étudier l'influence sur le nombre et la qualité des patterns découverts par le modèle. Nous allons entraîner plusieurs modèles avec 2, 4, 6, 12 et 18 filtres de déconvolution *figure 22*.

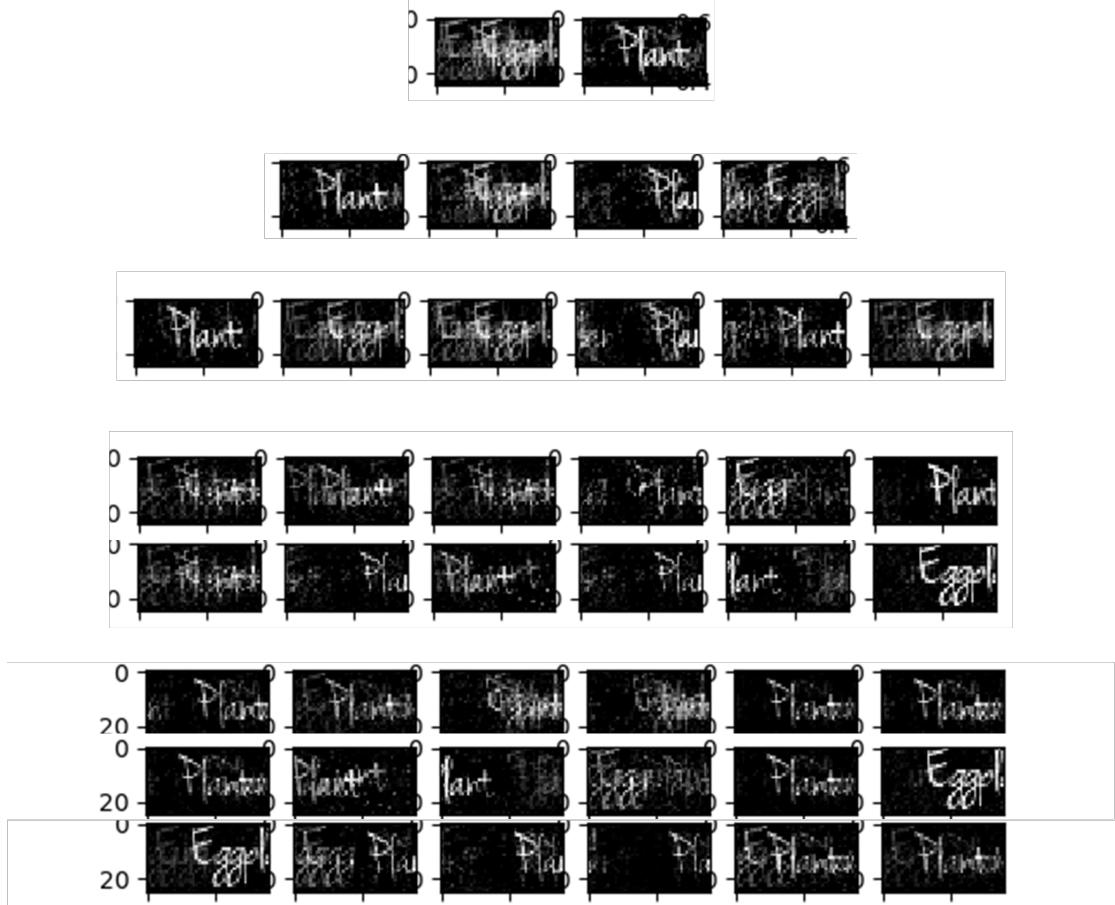


FIGURE 22 – Visualisation des filtres de déconvolution d'un modèle entraîné sur la série présentée en *figure 19*. Entraînements réalisés avec 2, 4, 7, 8 et 16 filtres de déconvolution.

Ainsi, on peut noter qu'augmenter le nombre de filtres n'influence pas la qualité des patterns découverts. Si l'on ne connaît pas le nombre de pattern à extraire à l'avance, une bonne méthode peut être de rajouter des filtres au modèle afin de ne pas manquer des patterns. Pour pénaliser le nombre élevé de filtres par rapport à un nombre plus faible de patterns, l'article utilise un terme de régularisation : λ_{grp} . Plus ce paramètre est élevé, plus nous aurons de filtres nuls. Nous avons entraîné le modèle avec différents λ_{grp} (2, 3, 4, 5, 6, 7) pour 18 filtres de déconvolution :

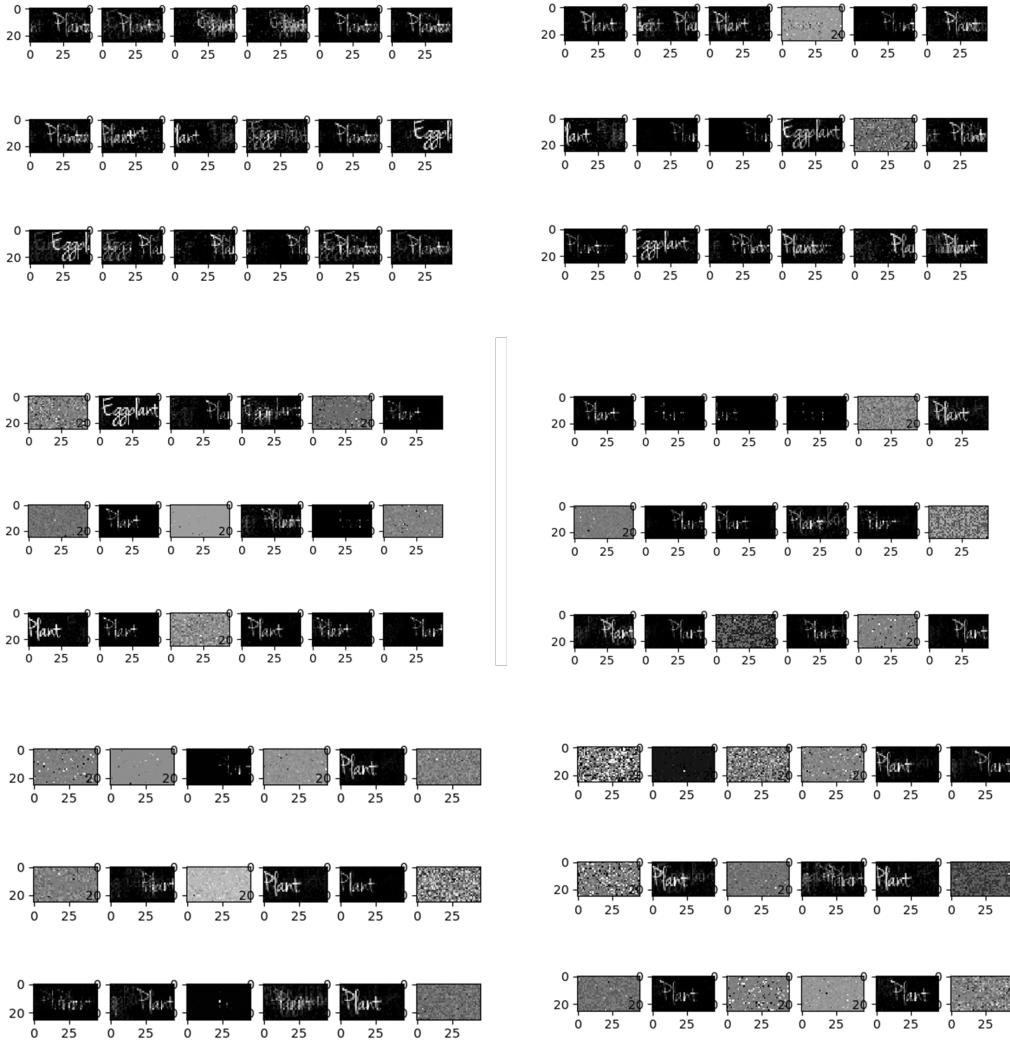


FIGURE 23 – Visualisation des filtres de déconvolution d'un modèle entrainé sur la série présentée en Figure 1. Entrainements réalisés avec un paramètre λ_{grp} à 2 (défaut), 3, 4, 5, 6, 7.

On a bien le comportement attendu : plus λ_{grp} est élevé, plus on a de filtres nuls. On peut noter qu'une valeur > 4 ici pose problème car le pattern "Egg" disparaît. Se pose alors la question de comment trouver une valeur optimale pour λ_{grp} sachant qu'on ne connaît pas le nombre de patterns attendus à l'arrivée. L'article indique qu'une méthode possible est d'augmenter progressivement le paramètre λ_{grp} jusqu'à avoir une augmentation conséquente de la loss lors de l'entraînement. Voici alors les loss correspondantes aux entraînements précédents :

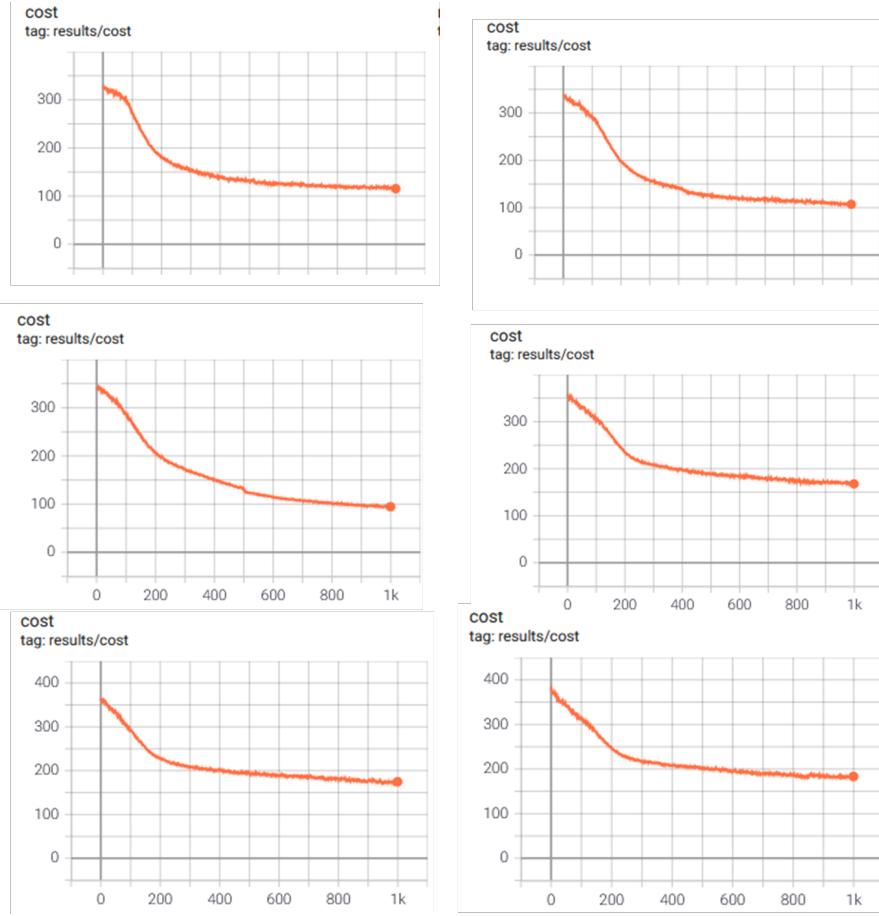


FIGURE 24 – Visualisation des loss correspondantes aux entraînements de la *figure 23*

Effectivement, la 4ème loss est bien plus élevée que la précédente, ce qui correspond bien au seuil où le pattern “Egg” disparaît. Nous sommes donc en mesure d’établir un protocole pour utiliser ce modèle sur nos séries temporelles. Nous allons mettre un nombre élevé de filtre (par rapport au nombre de patterns que nous pensons extraire), puis nous allons progressivement augmenter le paramètre λ_{grp} jusqu’à ce que la loss augmente considérablement.

4.4 Application aux séries temporelles

Les séries en notre possession représente l’évolution d’une donnée sur plusieurs années. Cela représente beaucoup d’information, ainsi nous allons réduire cette série à 90 points (d’abord réduit à 1000 mais par manque de ressources nous nous contenterons de 90), correspondant à 90 relevés de données à intervalle de temps réguliers. Pour faire le parallèle avec les travaux précédents, il faut comparer une série temporelle multivariée à une image. En effet, une image correspond à une matrice de pixels, on peut alors voir les séries temporelles multivariées comme tel : une sous série correspond à une ligne de pixels et donc la série multivariée correspond à un ensemble de lignes (soit une matrice) *figure 25*.

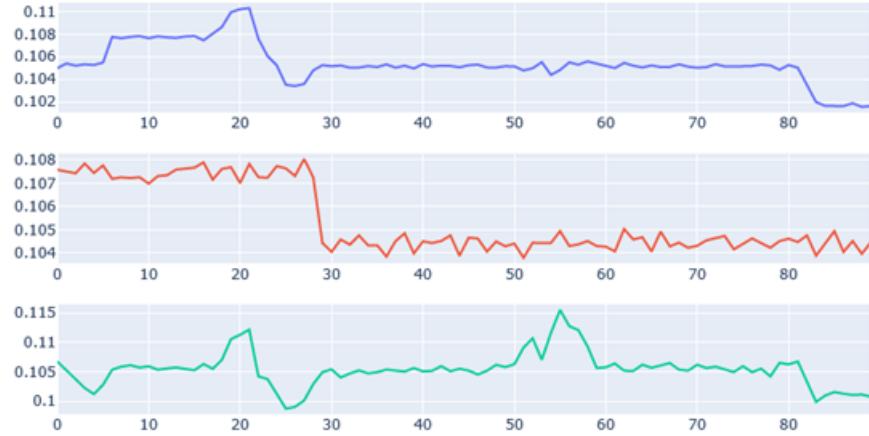


FIGURE 25 – Exemple de série temporelle multivariée en entrée du modèle.

Ainsi, nous avons tenté de mettre en place le protocole établi au préalable pour extraire des patterns de cette série multivariée. Nous avons entraîné un modèle avec les paramètres suivants : Série multivariée composée de 3 sous séries, 12 filtres de déconvolution de taille 45, un $\lambda_{grp}=2$ (paramètre par défaut : peu/pas de filtres à 0) et 1000 itérations (comme pour les images issues de l'article) *figure 26*.

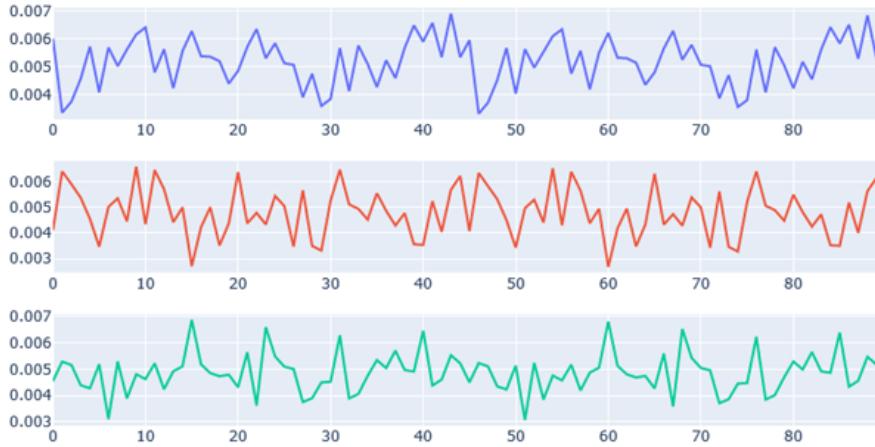


FIGURE 26 – Reconstruction de la série *figure 25* par le modèle.

Que ce soient les motifs, l'amplitude de ces trois séries, rien ne coïncide avec la série d'entrée.

Comme nous pouvons le voir sur la *figure 26* les résultats obtenus ne sont pas très encourageants, en effet la série de sortie de notre autoencodeur n'est pas similaire à la série d'entrée. Nous avons essayé de modifier les différents paramètres de régularisation sans résultats concluants. Nous pensons que cela vient du fait que les termes de régularisation sont trop forts et non adaptés à notre type de données, en effet ils sont destinés à supprimer le bruit en mettant le coefficient à 0 mais dans notre cas pour supprimer le bruit on ne met pas simplement la série à 0. Nous remarquons également que la série a des valeurs proches de 0 (autour de 0.001), ce qui nous conforte dans l'idée que la régularisation dans ce réseau est trop importante. Pour avoir un autoencodeur plus adapté à nos données nous aurions pu tester un modèle utilisant un autoencodeur de type LSTM, c'est une piste d'amélioration.

Ainsi, il existe plusieurs hypothèses pour expliquer ce phénomène. Cependant, les entraînements de ce modèle sont relativement exigeants en termes de ressources de calcul, or nous utilisons un

GPU en notre possession et nous ne sommes donc pas en mesure de mettre en place l'ensemble des tests : ces entraînements représenteraient un certain coût financier et énergétique. Cela nous permet aussi de mettre en avant un problème actuel au cœur de la recherche qui est l'impact énergétique (et environnemental) des modèles de Deep Learning. En effet, depuis la mise en lumière de ce domaine, les entreprises et les entités de recherches n'ont cessé d'essayer d'atteindre des performances toujours plus élevées avec leur modèle. Cela au prix de modèles toujours plus complexes, plus lourd et plus énergivore. Nous pouvons par exemple citer les modèles de NLP, GPT entraînés par OpenAI qui à ce jour comptent plus de 175 milliards de paramètres (pour la dernière version GPT-3). Nous pouvons voir [Figure 9], une estimation de l'impact environnemental lié aux modèles de Deep Learning. Heureusement, beaucoup de travaux démontre que nous pouvons conserver des performances similaires avec des modèles de plus petite taille. Cela nécessite un travail supplémentaire lors de l'entraînement de ces derniers mais prouve bien que des solutions existent.

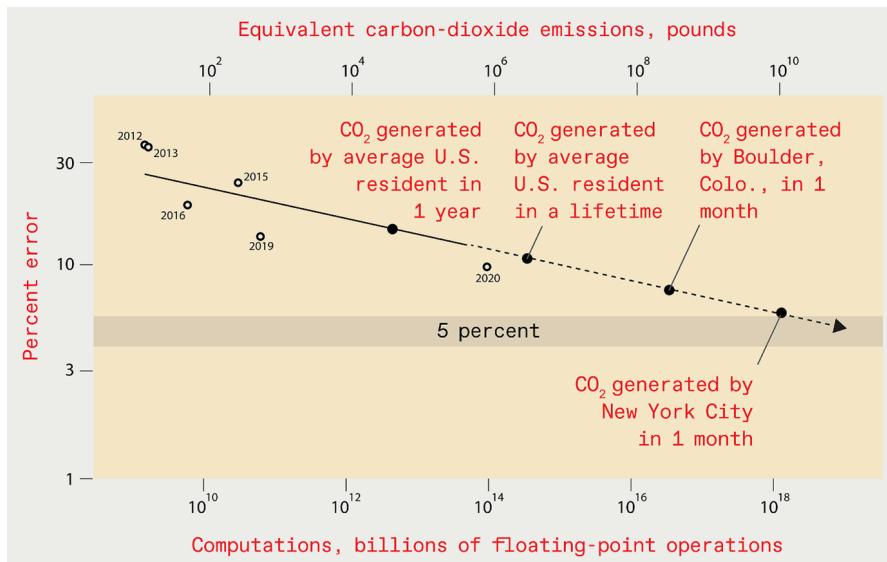


FIGURE 27 – Estimation de l'impact environnemental futur des modèles de Deep Learning. [5]

4.5 Limites et perspectives d'amélioration

Pour conclure sur ces travaux, nous avons choisi ce modèle car il est bien représentatif du tradeoff explicabilité/performances. Nous avions beaucoup d'attentes par rapport à l'aspect explicable de ce dernier. Si dans un premier temps, les résultats sur des images étaient très encourageants, les résultats sur les séries temporelles ne furent pas à la hauteur de nos espérances. Nous avons plusieurs hypothèses quant aux mauvaises performances, mais nous ne sommes pas en mesure de les confirmer par de nouveaux entraînements faute d'un manque de ressources. Une autre piste à explorer sont les LSTM, déjà présents dans la littérature et réputés pour leurs performances sur les séries temporelles mais qui sont aussi moins explicables.

5 Conclusion

Pour conclure, nous avons exploré dans ce rapport deux méthodes d'analyse de séries temporelles multivariées, et des résultats encourageant ont été obtenus. Le sujet du XAI et de l'analyse de séries temporelles multivariées sont d'actualité et beaucoup de travaux sont en cours. Peu de ressources sont présentes et c'est ce qui nous a amenés à formaliser certains concepts issus de notre réflexion personnelle.

A travers ce projet nous avons noté qu'il était important de réfléchir à quelles méthodes implémenter. En effet, l'utilité du Machine Learning ou Deep Learning n'est pas toujours nécessaire et de très bon résultats peuvent être obtenus à travers des approches algorithmiques moins coûteuses et plus performantes.

Bibliographie

- [1] Kevin BASCOL et al. “Unsupervised Interpretable Pattern Discovery in Time Series Using Autoencoders”. In : (2016). Sous la dir. d’Antonio ROBLES-KELLY et al., p. 427-438.
- [2] Chin-Chia Michael YEH, Nickolas KAVANTZAS et Eamonn KEOGH. “Matrix Profile VI : Meaningful Multidimensional Motif Discovery”. In : (2017), p. 565-574. DOI : 10.1109/ICDM.2017.66.
- [3] Sean LAW. “STUMPY Basics”. In : (2020). URL : https://stumpy.readthedocs.io/en/latest/Tutorial_STUMPY_Basics.html.
- [4] Thomas ROJAT et al. “Explainable Artificial Intelligence (XAI) on TimeSeries Data : A Survey”. In : (2021). DOI : 10.48550/ARXIV.2104.00950. URL : <https://arxiv.org/abs/2104.00950>.
- [5] Neil THOMPSON et al. “Deep Learning’s diminishing returns : The cost of improvement is becoming unsustainable”. In : (2021). URL : <https://spectrum.ieee.org/deep-learning-computational-cost>.