MOAYEDPOUR Quentin

ENSAE 2ème année

Stage d'application

Année scolaire 2023-2024

Rapport de stage

ENSTA Laboratoire U2IS

ENSTA, Laboratoire U2IS

Paris, France

Maître de stage : GIANNI FRANCCHI

Juin 2024 - Septembre 2024

ENSAE Paris

TSA 26644

www.ensae.fr

Service des relations entreprises et des stages

5, avenue Henry Le Chatelier - 91764 PALAISEAU CEDEX - FRANCE

Tél : +33 (0)1 70 26 67 39 - Courriel : stage@ensae.fr

Remerciements

J'adresse mes remerciements à toutes les personnes qui ont contribués au bon déroulement de mon stage.

Je remercie tout d'abord mon tuteur d'entreprise Monsieur FRANCHI, enseignant chercheur au sein de l'ENSTA, pour sa disponibilité, ses conseils et son encadrement. Ce stage fut une très bonne expérience pour moi pour découvrir le déroulement de la recherche académique.

Je remercie également l'équipe U2IS qui a pu m'accompagner durant le stage, les doctorants du laboratoires ont su m'accompagner et me conseiller lors de mes recherches.

Table des matières

1 Introduction

1.1 Contexte

Dans le cadre de mon parcours au sein de l'ENSAE, j'ai décidé d'effectuer un stage de recherche afin de découvrir la fonctionnement en interne de la recherche académique dans le domaine de l'apprentissage automatique. Ce stage avait donc pour objectif de me donner une première impression du domaine de la recherche dans le but de conclure mon parcours académique par une thèse en mathématiques appliquées.

L'ENSTA est une école d'ingénieur généraliste appartenant au groupe Institut Polytechnique de paris. L'ENSTA est présent sur de nombreux domaines, tels que la défense, la robotique et l'intelligence artificielle.

J'ai effectué mon stage au sein du laboratoire U2IS. Ce dernier conduit des recherches dans le domaine de la robotique et du Deep Learning. J'ai ainsi été entouré de doctorants et d'étudiants en stage de recherche. J'ai été affecté à un projet de recherche sur la détection d'anomalies appliquée au séries temporelles.

1.2 Objectifs et enjeux du stage

Au sein du laboratoire U2IS de l'ENSTA, à l'aide de M. FRANCHI et M. BELKHIR, le but du stage est d'explorer des méthodes de transfer learning pour obtenir des résultats convainquants pour la tâche de détection d'anomalies sur des séries temporelles. En vue de l'écriture d'un papier, j'ai rejoint l'équipe au début du projet. J'ai donc participé à l'étude préliminaire du sujet, la lecture et la compréhension des méthodes de détection d'anomalies de l'état de l'art, la recherche de données (benchmarks) pour l'évaluation des modèles ainsi que la conception des modèles.

J'ai pu découvrir le développement d'un projet de recherche au sein d'un laboratoire, ses enjeux et ses contraintes. Ce rapport de stage va donc développer et suivre la démarche de recherche au cours du projet. Tout au long du rapport, j'expliquerai de manière intuitive les grands principes des modèles utilisés, mais je n'entrerais pas dans les détails profonds des modèles afin d'alléger le rapport. Les architectures de réseaux de neuronnes bien connus tels que les transformers ou les réseaux de neuronnes convolutionnels ne seront pas présentés en détail.

Enfin, le rapport présente séquentiellement les modèles et les résultats sont présentés à la fin. Je mentionnerai dans le rapport certains résultats afin de justifier le passage à d'autres modèles. Dans le tableau de résultats, certains modèles n'ont pas été testés sur tout les datasets car ils ont été étudiés au debut du stage et n'ont pas été retenus en raison de leurs performances et n'ont donc pas été testés sur tout les datasets étudiés.

1.3 Presentation du sujet

L'analyse des séries temporelles constitue un domaine de recherche fondamental en statistique et en apprentissage automatique, ayant des applications étendues dans des disciplines variées telles que l'économie, la météorologie, l'ingénierie, la biologie et les sciences sociales. Une série temporelle est une suite d'observations ordonnées chronologiquement, permettant d'étudier et de modéliser l'évolution d'un phénomène au fil du temps. L'objectif principal de l'analyse des séries temporelles est de déceler des structures ou des régularités sous-jacentes aux données, afin de prédire les valeurs futures, comprendre les mécanismes générateurs des données ou effectuer des interventions ciblées.

L'intérêt croissant pour l'étude des séries temporelles résulte non seulement de la disponibilité accrue de données chronologiques grâce à l'essor des technologies de l'information et de la communication, mais également des avancées méthodologiques récentes qui permettent de traiter des données de plus en plus complexes. Les modèles traditionnels, tels que les modèles autorégressifs (AR), les modèles à moyenne mobile (MA), ou encore les modèles autorégressifs intégrés à moyenne mobile (ARIMA), ont été enrichis par des techniques plus sophistiquées basées sur l'apprentissage automatique et les réseaux de neurones récurrents (RNN), permettant ainsi une meilleure prise en compte des non-linéarités et des dépendances complexes présentes dans les séries temporelles.

La détection d'anomalies dans les séries temporelles est un domaine de recherche clé aux applications variées et cruciales dans de nombreux secteurs. Par exemple, une anomalie peut réveler une fraude bancaire dans le cas de données financières, la présence d'une maladie cardiaque dans le cas d'électro-cardiogrames [4] etc... L'importance de la détection d'anomalies dans les séries temporelles a considérablement augmenté au sein de la communauté scientifique ces dernières années[3]. Ce regain d'intérêt s'explique notamment par les progrès en matière de puissance de calcul et par l'essor des méthodes d'apprentissage automatique, qui ont ouvert la voie à de nouvelles alternatives aux approches statistiques traditionnelles, souvent fondées sur des hypothèses rigides. Parallèlement, les récentes avancées en transfer learning, largement exploitées dans des domaines comme le traitement du langage naturel et la vision par ordinateur, ont démontré leur efficacité. Les réseaux de neurones, en particulier, se sont révélés capables de capturer des représentations complexes des données. De plus, l'utilisation de modèles pré-entraînés, souvent appelés backbones, permet de tirer parti de larges ensembles de données pour extraire des représentations adaptées à des tâches spécifiques, tout en réduisant le temps et les ressources nécessaires à l'entraînement.

Dans ce contexte, ce travail de recherche s'inscrit dans l'exploration et la proposition de nouvelles méthodes de transfer learning pour la détection d'anomalies dans les séries temporelles. L'objectif est de concevoir des modèles à haute capacité tout en réduisant de manière significative les coûts liés à l'entraînement et à la mise en œuvre de ces modèles,

rendant ainsi leur application plus efficace et accessible.

2 Revue Littérature

Soit une série temporelle $(X_t)_{t\in\mathbf{T}}$ où $\forall t \in \mathbf{T}, X_t \in \mathbb{R}^k$ où $k \in \mathbb{N}^*$. On note $(Y_t)_{t\in\mathbf{Y}}$ le "label" de la série définit par :

$$Y_t = \begin{cases} 1 & \text{si } X_t \text{ est une anomalie,} \\ 0 & \text{sinon.} \end{cases}$$

Pour la suite du rapport, on se place dans le cas où $k = 1$, on parle de série univariées. Le vecteur $\mathbf{X}_s := (X_{s_1}, ..., X_{s_L})$ où $[s_1, ...s_L]$ est une grille de $\mathbf{T}$ tel que $s_1 < s_2 ... < s_L$, est une fenêtre de la série de taille L, qu'on notera $\mathbf{W}_s$. On considère qu'une fenêtre est une fenêtre d'anomalie si la fenêtre contient au moins une anomalie, ainsi : $\mathbf{Y}_s = \arg\max_{s_j \in \{s_1,...,s_L\}} Y_{s_j}$. La tâche de détection d'anomalie vise a définir et estimer une fonction f, qui prends en entrée généralement une fenêtre de la série, et associe un score d'anomalie à un point ou une fenêtre. On peut alors ensuite définir un seuil δ tel que $\hat{y}_{t_L} = \mathbf{1}\{\hat{f}(\mathbf{x}_t) \geq \delta)\}$ [1], afin de simplifier la notation, on notera α_t le "score" d'un point x_t et α_s le score d'une fenêtre $\mathbf{X}_s$.

La littérature scientifique ne s'accorde pas sur une définition précise d'une anomalie dans le cadre de séries temporelles [26]. En reprenant la définition de *Chandola et Al.*[5], une anomalie ou une séquence d'anomalie est un paterne qui ne se conforme pas au "comportement normal" d'une série. Une anomalie peut donc avoir plusieurs formes et plusieurs causes. Dans le cadre de séries temporelles, on regroupe généralement les anomalies en trois types :

- Global outlier : lorsque un point de la série diverge par rapport aux autres observations de la série.
- Contextual outlier : lorsque la distribution d'un point ou d'un groupe de points diffère des autres points dans un même contexte.
- Collective ouliers : lorsqu'aucune observation individuelle n'est en soi une anomalie, mais qu'un groupe de ces observations, une fois combinées, présente un comportement qui diverge du reste des données. Sur le marché boursier, par exemple, la baisse du prix d'un actif ne s'écarte pas de manière significative de la fourchette normale, mais la combinaison de baisses successives indique une anomalie collective.

Les anomalies sont généralement peu présentes et ne partagent pas nécessairement une forme commune (ie deux anomalies dans une série peuvent être totalement différentes et n'ont pour point commun que le fait de dévier de la distribution), les modèles ne peuvent

1. On note en majuscule les variables aléatoires et en minuscules les observations. Un chapeau $\hat{}$ devant une variable désigne qu'elle est estimée

pas se fonder sur une connaissance à priori de la forme des anomalies. De plus, il existe généralement peu d'annotations aux séries temporelles et en particuliers pour la tâche de détection d'anomalie, il peut être très compliqué de définir une anomalie dans une série temporelle lorsqu'il n'y a pas de cause connue (par exemple, erreur de mesure, disfonctionnement d'un capteur etc...) et les données labellisées sont généralement très rares dans les applications concrètes[34]. La tâche de détection d'anomalies diffère donc fortement de la tâche de classification. La plupart des modèles de détections d'anomalies sont non supervisé ou semi-supervisé. On parle de modèle non supervisé lorsque le modèle ne nécessite pas de connaître le label des données sur lequel il s'entraine, semi supervisé lorsque le modèle nécessite de s'entrainer seulement sur des données sans anomalies[26]. On cherche alors à trouver des méthodes générales qui, en s'appuyant sur la flexibilité des méthodes de machine learning, puissent s'appliquer et montrer des résultats convainquants sur des données temporelles provenant de multiples horizons.

On peut distinguer les modèles de détection en différentes catégories, on s'interessera par la suite à trois catégories : Les modèle de forecasting, les modèles de reconstruction et les modèles de représentation.

2.1 Forecasting-Based models

Les modèles forecasting-based sont une manière simple de détecter des anomalies. Ces modèles entraînent une modèle de forecasting f qui prends en entrée une séquence $\mathbf{X}_{s_{t-1}} = (X_{t-L-1}, ...X_{t-1})$ et associe au point X_t un score d'anomalie à l'aide de la distance entre le point prédit et le point observé, ie $\alpha_t = ||X_t - f(\mathbf{X}_{s_{t-1}})||$, où $||.||$ est généralement la norme ℓ_1. L'idée est que un modèle de forecasting pour les séries temporelles aura tendance à faire des erreurs de prédictions plus grandes lorsque le point prédit est une anomalie. Le modèle ARIMA peut donc être utilisé pour faire de la détection d'anomalie et servira de baseline pour évaluer nos modèles. En prenant (X_t), on suppose que on peut écrire :

$$X_t = \phi_0 + \phi_1 X_{t-1} ... + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} ... + \theta_p * \varepsilon_{t-p}$$

où $\varepsilon_t \sim \mathrm{BB}(0, \sigma^2)$, on dit alors que (X_t) suit un modèle ARMA stationnaire au second ordre. On peut alors prendre pour score d'anomalie pour un point t :

$$\begin{aligned} \alpha_t &= ||X_t - \hat{EL}(X_t | X_{t-1}, X_{t-2}, \ldots, X_0)||_1 \\ &= ||X_t - \hat{\phi}_0 - \hat{\phi}_1 X_{t-1} - \cdots - \hat{\phi}_p X_{t-p} - \hat{\theta}_1 \hat{\varepsilon}_{t-1} - \cdots - \hat{\theta}_q \hat{\varepsilon}_{t-q}||_1 \end{aligned}$$

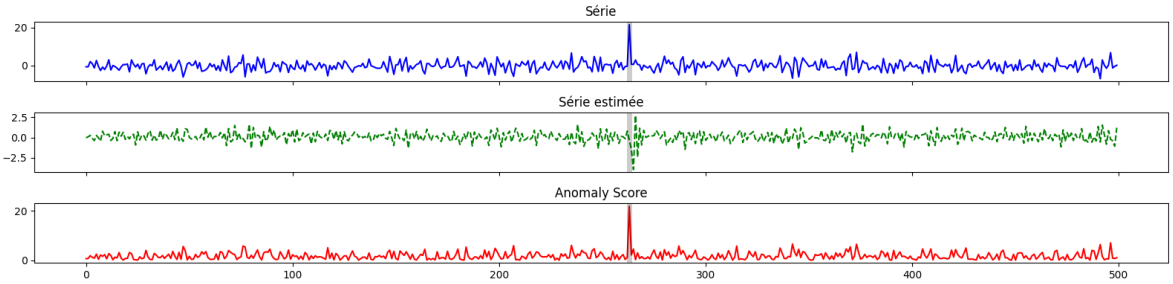


FIGURE 1 – Série ARMA(2,3) simulée

La figure 2 résume la procédure d'utilisation du modèle ARMA. Cette méthode possède cependant des limites : elle repose d'abord sur des hypothèses fortes. En effet le modèle nécessite de faire des transformations sur la série (X_t) (stationnariser, retirer les saisonnalités), de supposer que il existe des relations linéaires entre la variable et son passé. De plus, le modèle suppose une relation "simple" pour définir la trajectoire de la variable aléatoire qui peut ne pas être pertinent pour des données temporelles avec des structures et patternes plus complexes. De plus, le modèle ARMA est sensible aux valeurs "extrêmes" et est moins pertinent pour détecter des "pattern outliers".

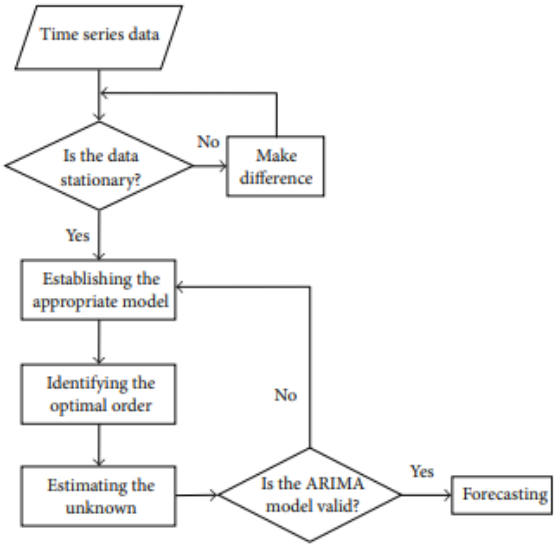


FIGURE 2 – Procédure d'utilisation du modèle ARMA

Les méthodes d'apprentissage profond ont permises de faire émerger des réseaux de neuronnes pour le traitement de données temporelles. Le modèle DeepAnt[20] est un modèle de détection d'anomalie qui entraîne un modèle de prédiction sur les séries et utilise l'erreur de prédiction comme score d'anomalie. Le modèle DeepAnt repose sur une architecture simple, composé de de neuronnes convolutionnels. Un réseau de neuronne est une fonction composé de modules simples qui intéragissent entre eux et appliquent des transformations non linéaires aux données[16]. Les réseaux de neuronnes convolutionnels sont une architec-

ture de réseaux de neuronnes qui ont gagné en popularité grâce à leurs performances pour le traitement d'image. Un neuronne de convolution applique une opération de convolution séquentiellement dans les données. une architecture de réseaux de neuronne peut être représenté de la manière suivante :

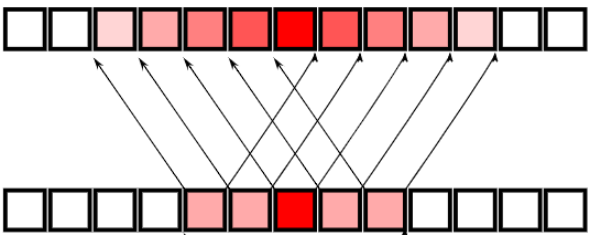
$$z_{ji}^{(l)} = \sum_{i=-k}^{k} W_{jk}^{(l)} a_{i-k}^{(l-1)} + b_j^{(l)} \qquad a_{ji}^{(l)} = \sigma(z_{ji}^{(l)})$$

$$z_j^{(l)} = \sum_{k=1}^{e} W_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)} \qquad a_j^{(l)} = \sigma(z_j^{(l)})$$

Où σ représente une fonction d'activation non linéaire (par exemple la fonction $max(.,0)$), $a_{ji}^{(l)}$ représente l'activation du j^{eme} neuronne de la l^{eme} couche à la position i et $a_j^{(l)}$ représente l'activation du j^{eme} de la couche l. Ici la taille de la fenêtre de données est de N et $k = \lfloor \frac{N-F}{S} \rfloor - 1$ où F est la taille du filtre de convolution et S la valeur du stride (de combien de "pas" se déplace le filtre de convolution). Le grand avantage de cette architecture est que le nombre de paramètre à entraîner pour chaque filtre de convolution $W_{jk}^{(l)}$ ne dépends pas de la dimension des données d'entrées. De plus, les filtres de convolutions permettent d'agréger de l'information locale des données et sont ainsi utiliser pour le traitement de séries temporelles. Enfin, leur utilisation pour les séries temporelles est en partie justifiée par leur capacité à agréger séquentiellement de l'information.

En effet supposons qu'on dispose d'un kernel de taille k (c'est à dire qui prends en entrée k points), à la fin de la première couche de convolution, chaque point dans l'espace latent dépendra de k points. On note R_n le nombre de points de l'input qui influence la sortie d'un neuronne à la n-ème couche, appelé le "champ réceptif" du neuronne. Si on ajoute une autre couche de convolution (toujours de taille de kernel k), chaque points à la sortie de la couche dépendra de k points qui eux même dépendant de k points. En prenant l'exemple d'un "stride" [2] de 1, chaque points de la première couche dispose d'un point différend par rapport à ses voisins. ce qui donne la relation de récurrence :

$$R_n = R_{n-1} + (k-1)$$



2. De combien se déplace la fenêtre de convolution

Ainsi, enchainer plusieurs couches de convolutions permet de faire circuler séquentiellement l'information au sein des couches du réseaux et permet aux premières couches de synthétiser de l'information à l'échelle locale, et aux dernières de l'information à l'échelle globale de l'input.

Le modèle DeepAnt[20] repose ainsi sur l'entrainement d'un réseaux de neuronnes convolutifs pour prendre en entrée une fenêtre de taille N et de prédire la valeur suivante, et utilise comme score d'anomalie l'erreur de prédiction du modèle au carré. L'avantage de cette méthode est qu'elle ne requiert aucune donnée labellisés et nécessite peu de données d'entraînements. Cependant, le défaut du modèle est que celui ci montre des lacunes lorsqu'il est appliqué sur des données sur lesquels il ne s'est pas entraîné. Les performances des modèles de forecasting tendent à se déteriorer lorsque le nombre de points de la série augmente[18]. De plus, de nombreuses séries temporelles ne sont pas prévisibles à partir de leur passé ou possèdent une composante stochastique importante que les modèles de détection d'anomalie forecasting-based ne peuvent distinguer des vraies anomalies[9]. Une autre approche est donc d'utiliser un modèle de reconstruction des données.

2.2 Reconstruction-Based models

Les modèles Reconstruction-Based pour de la détection d'anomalies ont émergés avec le développement des architectures encodeur-decodeur en deep learning. Communément appelés les auto-encodeurs[25], cette famille de modèles entraînent conjointement un encodeur et un decodeur, formellement, on a l'encodeur $f : \mathbb{R}^n \to \mathbb{R}^p$ et le décodeur $g : \mathbb{R}^p \to \mathbb{R}^n$ où généralement $n \gg p$ qui sont entraînés pour satisfaire :

$$\arg\min_{f,g} \mathbb{E}[||X - g \circ f(X)||]$$

Où $||.||$ désigne l'erreur de reconstruction et $X \in \mathbf{R}^n$. f et g sont généralement des réseaux de neuronnes, mais dans le cas où f et g sont des applications linéaires, l'auto encodeur aura les même résultats que une ACP[27]. Les auto-encodeur peuvent être vu comme une généralisation des ACP. Ces modèles sont généralement non supervisés et ont trouver de nombreuses applications, pour la tâche de clustering, de génération de données où encore de détection d'anomalies.

En effet, les encodeurs sont entraînés à apprendre une représentation dans un espace réduit des données tandis que les décodeurs reconstruisent les données à partir de leur représentation dans l'espace latent. Une première idée pour utiliser les auto encodeurs pour de la détection d'anomalie et d'utiliser l'erreur de reconstruction comme score d'anomalie. En entraînant un modèle sur des données ne comprenant aucune anomalies, on s'attends à ce que le modèle présente de moins bonnes performances lorsque la donnée d'entrée X ne suit pas

les même patterns que les données sur lesquels il s'est entraîné et ainsi avoir une erreur de reconstruction plus élevée.

Une autre méthode utilisée pour la tâche de détection d'anomalies est de s'intéresser à la représentation des données dans l'espace latent. En effet la dimension réduite de l'espace latente peut aider à identifier des caractéristiques des données, et de manipuler plus facilement les données afin d'identifier des anomalies, qui seraient par exemple éloignée selon une certaine métrique $||.||$ des autres données. Par exemple, la figure 3 montre la représentation de digits dans un espace latent de dimension 2 à l'aide d'un VAE (Variational Auto Encoder)[15]. Cet exemple montre que les AutoEncodeurs arrivent à identifier la proximité des données et une représentation efficiente dans l'espace latent.

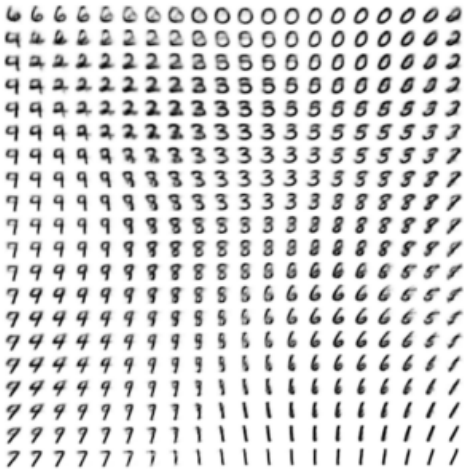


FIGURE 3 – Repartition de données dans l'espace latent d'un VAE de dimension 2. La position de chaque image correspond à sa valeur dans l'espace latent du modèle

Ainsi, les modèles de reconstruction entraînent généralement un auto encodeur sur des samples $\mathbf{X}_s$ et extraient ensuite un score d'anomalie à l'aide de la données reconstruite par le modèle $\hat{\mathbf{X}}_s = g \circ f(\mathbf{X})$. On peut par exemple citer les modèles Donut[32], AER[30]. Cependant, il a été montré que ces modèles peuvent être limités car les modèles peuvent montrer de bonnes performances pour la reconstruction d'anomalies[21]. De plus, ces modèles sont généralement semi-supervisés car ils nécessitent de n'entraîner un modèle uniquement sur des données ne présentant pas d'anomalies. Une autre classe de modèle est alors de ne pas manipuler les données elle même mais d'extraire d'entrainer des modèles afin d'extraire des représentations spécifiques pour aider à la tâche de détection d'anomalies.

2.3 Representation-based models

Les modèles basés sur de la représentation utilisent des modèles afin d'extraire des informations ou synthetiser les informations afin de faciliter la tâche de détection d'anomalie. Ces modèles sont particulièrement utilisés en détection d'anomalies pour les images car

ils permettent par exemple de réduire grandement la dimension des données, ou bien d'utiliser des modèles pré-entrainés et d'opérer sur l'espace latent d'un grand modèle[6][24]. Ici on se réfère de manière générale à l'espace latent d'un modèle de réseaux de neuronne la représentation de données au sein d'une couche intermédiaire du réseaux de neuronne. Bien que l'espace latent se réfère généralement à un espace de dimension réduit (par rapport aux données d'entrée), il peut être pertinent de choisir un espace latent de dimension plus grande, particulièrement pour des données temporelles qui sont généralement unidimensionnelles. Par exemple, le modèle Anomaly Transformer[33] utilise l'architecture des réseaux de neuronnes transformers légerement modifié afin d'y extraire ...

2.4 Evaluation des modèles

La tâche de détection d'anomalie est une tâche de classification, on utilise donc différentes métriques afin d'évaluer les modèles. On a par exemple :

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Rappel} = \frac{TP}{TP+FN}$$

Où TP = nombre de données correctement labellisées en 1, FP = nombre de données 0 labellisées en 1 par le modèle, FN = nombre de données 1 labellisées en 0. En considérant un classificateur de la forme $f(x,\delta) = \mathbf{1}\{s(x) \geq \delta\}$ augmenter δ augmente généralement le Rappel au détriment de la précision. On considère donc le F1 score défini par :

$$\text{F1} = 2 \times \frac{\text{Precision} * \text{Rappel}}{\text{Precision} + \text{Rappel}}$$

On affichera alors pour chaque modèle la precision et le rappel qui maximise le F1 score. De nombreux papier sur la détection d'anomalie n'utilisent pas ces métriques telles quelles. En effet dans le cadre d'anomalies temporelles, une anomalie peut constituer un segment et il est en général pas dérangeant si le modèle a detecté une partie des anomalies sur le segment mais pas toutes[32]. Ainsi de nombreux papier utilisent des métriques corrigés de la manière suivante : En considérant une séquence d'anomalie $y_{t:t+k} = 1$ et en notant s_t le score d'anomalie attribué au modèle, on corrige le score avec $\hat{s}_{t:t+k} = \max_{\tau \in [t,t+k]} s_\tau$. Cette modification permet d'ajuster les scores à un cadre plus pratique.

Cependant, cette métrique mène a des résultat biaisés qui augmentent l'espérance du F1 score, particulièrement lors de la présence de longue séquences d'anomalies.

En effet, prenons un exemple simple avec un prédicteur aléatoire $\mathbf{A}(x_t) \sim \mathscr{U}[0,1]$, on

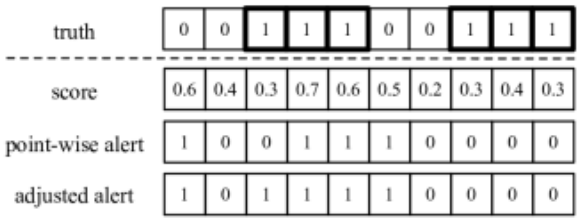


FIGURE 4 – Ajustement du score pour un seuil δ de 0.5.

a donc $\hat{y}_t = \mathbf{1}\{\mathbf{A}(x_t) > \delta\}$, et supposons que la part des anomalies est λ. Ainsi on a :

$$\text{Rappel} = P(\hat{y}_t = 1 | y_t = 1) = 1 - \delta$$

$$\text{Precision} = P(y_t = 1 | \hat{y}_t = 1) = \lambda$$

Etant donné que le predicteur est indépendant de (y_t). On a donc :

$$\begin{aligned} \text{F1 score} &= 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \\ &= 2 \times \frac{(1-\delta)\lambda}{(1-\delta)+\lambda} \end{aligned}$$

Qui est décroissant en δ et est maximal pour $\delta = 0$ et vaut $2 \times \frac{\lambda}{1+\lambda}$. λ est généralement de l'ordre de 1%. Or avec l'ajustement toujours dans le même exemple en supposant que les anomalies sont présentes pour $t \in \mathbf{S}$ (on a donc un seul groupement d'anomalies qui se suivent de taille $|\mathbf{S}|$), on a :

$$\begin{aligned} \text{Recall} &= 1 - P(\hat{y}_t = 0 | y_t = 1) \\ &= 1 - P(\hat{y}_t = 0 | t \in \mathbf{S}) \\ &= 1 - \prod_{t \in \mathbf{S}} P(\mathbf{A}(x_t) < \delta) \\ &= 1 - \delta^{|\mathbf{S}|} \end{aligned}$$

$$\begin{aligned} \text{Precision} &= \frac{P(\hat{y}_t = 1 | y_t = 1) \times P(y_t = 1)}{P(\hat{y}_t = 1)} \\ &= \text{Recall} \times \frac{\lambda}{P(\hat{y}_t = 1, y_t = 1) + P(\hat{y}_t = 1, y_t = 0)} \\ &= \frac{\lambda(1 - \delta^{|\mathbf{S}|})}{\lambda(1 - \delta^{|\mathbf{S}|}) + (1-\delta)(1-\lambda)} \end{aligned}$$

On voit que en faisant grandir la taille de la fenêtre d'anomalies $|\mathbf{S}|$, on augmente l'espérance du F1 score lorsqu'on utilise le score ajusté (utilisé dans de nombreux papiers), on obtient un F1 score qui a pour espérance 0.9 dans certains cas pour un prédicteur totalement aléatoire [Figure 5]. Une telle métrique peut alors afficher des résultats très impressionnants qui pourtant ne sont pas meilleur qu'un prédicteur aléatoire.

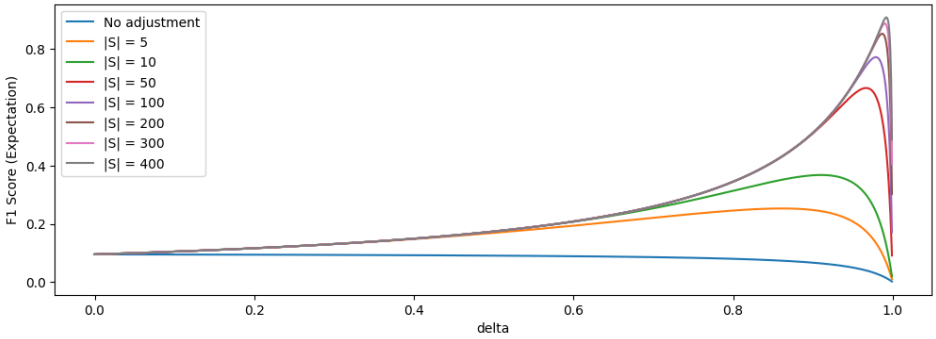


FIGURE 5 – Espérance du F1 score pour un prédicteur aléatoire selon la taille de la fenêtre d'anomalies

Le score ajusté n'est donc pas une bonne métrique et peut donner une fausse impression qu'un modèle est superformant. Il a donc été choisi d'utiliser un autre ajustement qui ne biaise pas l'estimation des métriques (Precision et Rappel). Afin de pouvoir favoriser la détection au sein d'une fenêtre de variable (de taille L), si on a un classificateur qui associe un score d'anomalie pour un point A_{x_t} [3], on définit le score par fenêtre :

$$\mathbf{A}_{W_s} = \max_{s \in W_s} A_{x_s}$$

Où $W_s = \{s_1, \ldots, s_L\}$ est la fenêtre d'index qui contient t. On a $\forall s \neq t, W_s \cap W_t = \emptyset$: on n'a pas d'overlap entre les fenêtres. En créant des fenêtre de cette manière, la part d'anomalies dans le dataset est réduite mais les métriques Precision et Rappel ne sont pas biaisés :

$$\begin{aligned} \text{Rappel} &= 1 - P(\hat{\mathbf{y}}_s = 0 | \mathbf{y}_s = 1) \\ &= 1 - P(\mathbf{A}_{W_s} < \delta') \\ &= 1 - \delta'^{L} \end{aligned} \qquad \begin{aligned} \text{Precision} &= P(\mathbf{y}_s = 1 | \hat{\mathbf{y}}_s = 1) \\ &= P(\mathbf{y}_s = 1) \\ &= \lambda' \end{aligned}$$

Où on voit qu'on obtient les mêmes résultats que dans le cadre d'un prédicteur aléatoire uniforme, à l'exception que la part d'anomalies λ est devenue λ'. L'avantage de cette évaluation est que le score n'est pas ajusté en fonction du label et est donc indépendant du label. La limite est cependant que si on augmente L, on réduit le nombre de données pour l'estimation de la précision et du rappel et on augmente donc leur variance.

Ainsi dans le reste du rapport les métriques présentées (sauf si mention contraire) utiliseront cette évaluation "par fenêtre". Le choix de la taille de la fenêtre L sera expliqué plus tard.

3. Ici le classificateur n'est pas forcément une fonction de x_t uniquement mais est la fonction qui associe au point x_t son score d'anomalie.

3 Transfer learning pour la détection d'anomalies

On cherche à évaluer et explorer différents moyen d'utiliser des méthodes de transfer-learning pour la détection d'anomalie. La notion de transfer-learning se réfère à l'utilisation de connaissances aquises par un modèles pour résoudre ou faciliter le traitement d'un problème annexe. L'avantage du transfer learning et de généralement pouvoir réduire grandement le nombre de données et le coût d'entrainement d'un modèle [Figure 6].

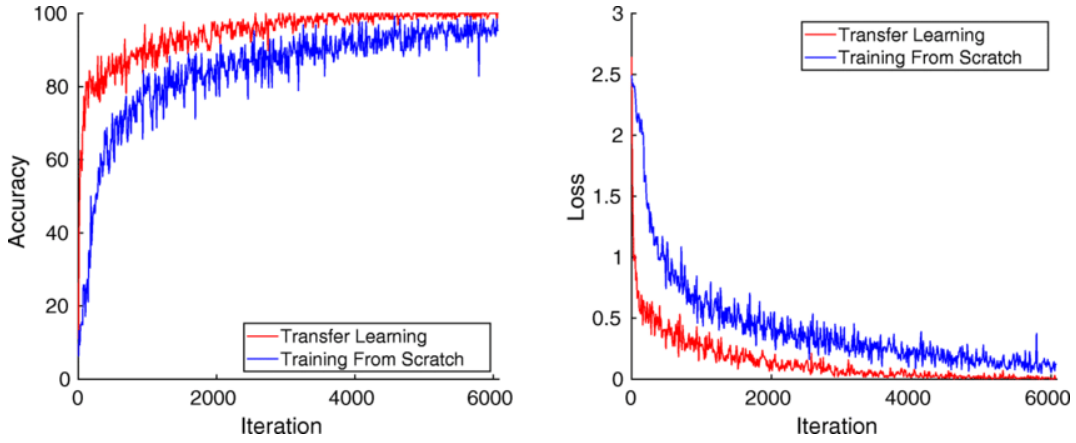


FIGURE 6 – Performance de modèles de NLP en fonction du nombre de données d'entraînements[2].

Les méthodes de transfer learning se sont particulièrement développés dans les méthodes de traitement du langage et de la vision et on dispose alors de nombreuses "backbone" (architecture de réseaux de neuronne pré-entrainés) accessibles librement, tels que Bert[7], ResNet[12] etc... Une première méthode est alors de transformer les séries temporelles en image, et ensuite y appliquer un modèle de détection d'anomalies pour image. Le principal avantage est d'utiliser des grands modèles pré-entrainés de computer vision disponibles en libre accès et qui ont montré de très bonnes performances pour la détection d'anomalies sur image[24][6].

3.1 Modèles de vision pour la détection d'anomalies de séries temporelles

Une première étape est de trouver un moyen de représenter une série temporelle sous forme d'image. Par exemple, supposons qu'on dispose d'une série de taille T et qu'on souhaite découper en samples de taille L, on veut convertir nos samples pour avoir des données de taille $L \times L$ qui représente au mieux notre série temporelle. En augmentant la dimension des données, on souhaite augmenter aussi les informations fournies par les données. La méthode choisie fut la Gramian Angular Field[29].

Soient $v_1, v_2, \ldots, v_n$ des vecteurs dans un espace vectoriel muni d'un produit scalaire $\langle \cdot, \cdot \rangle$. La matrice de Gram associée est définie par :

$$G = \begin{pmatrix} \langle v_1, v_1 \rangle & \langle v_1, v_2 \rangle & \cdots & \langle v_1, v_n \rangle \\ \langle v_2, v_1 \rangle & \langle v_2, v_2 \rangle & \cdots & \langle v_2, v_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle v_n, v_1 \rangle & \langle v_n, v_2 \rangle & \cdots & \langle v_n, v_n \rangle \end{pmatrix}$$

La méthode Grammian Angular Field n'utilise pas directement cette matrice sur les valeurs $x_1, ...x_L$. La Grammian Angular Field utilise une pseudo-matrice de Gram construite de la manière suivante :

1. Mettre à l'échelle le vecteur $\mathbf{X}_s$ avec :

$$\tilde{x}_i = \frac{2 \times x_i - \max_{i \in \mathbf{W_s}} x_i - \min_{i \in \mathbf{W_s}} x_i}{\max_{i \in \mathbf{W_s}} x_i - \min_{i \in \mathbf{W_s}} x_i}$$

2. Ensuite, définir

$$\varphi_i = \arccos(\tilde{x}_i)$$

   Comme $x_i \in [-1, 1]$ l'opération est bien définie

3. Enfin, on définit les termes de la matrice G par :

$$G_{i,j} = cos(\varphi_i + \varphi_j), \quad i, j \in [1, ...L]$$

   En remarquant que on peut écrire :

$$\begin{aligned} cos(\varphi_i + \varphi_j) &= cos(arccos(x_i) + arccos(x_j)) \\ &= x_i \cdot x_j + \sqrt{1 - x_i^2} \cdot \sqrt{1 - x_j^2} \\ &= \langle x_i, x_j \rangle - \sqrt{1 - x_i^2} \cdot \sqrt{1 - x_j^2} \end{aligned}$$

Où $\langle ., . \rangle$ représente le produit scalaire. On parle de pseudo-matrice de gram car l'opération ne conserve pas la linéarité. Cette convertit les valeurs de la série temporelle en angles via la fonction *arccos*, permettant de représenter les relations sous forme de relations angulaires. Cela permet de capturer non seulement les valeurs elles-mêmes, mais aussi comment ces valeurs se combinent et se différencient dans le temps. On peut donc ainsi représenter nos séries temporelles en image et utiliser ainsi des modèles de détection d'anomalies d'image sur nos séries temporelles [4]. La figure 7 montre que les "pics" de la série sont bien représentés

4. Une des raisons de cette approche est que la détection d'anomalie en Computer Vision et particulièrement en Deep Learning est très développée et les algorithmes de l'état de l'art atteignent des résultats à la limite de la perfection

dans l'image.

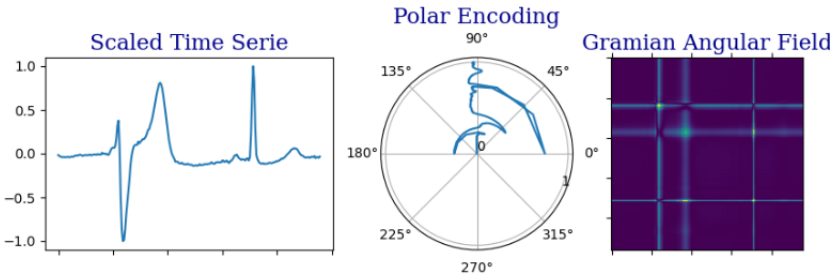


FIGURE 7 – Représentation d'une série temporelle en image avec la méthode GAF

La figure 8 montre que les anomalies peuvent se voir visuellement dans la représentation GAF bien que celles ci soient abstraites. Tout d'abord la présence de valeur anormalement haute ou anormalement basse réduit grandement les variations entre les autres points (étant donné qu'on normalise les données) et permet aux représentations d'être très sensibles aux valeurs extrêmes. La deuxième ligne montre que les patterns récurrents peuvent être bien représentés et un changement de pattern se reflète sur la représentation en image.

Une première limite de cette représentation est qu'on ne peut avoir un score d'anomalie par points de la série. Ici la première raison est que la dimension des données est mise au carré, il n'est donc pas évident de déduire un score d'anomalie pour un point préçis de l'image. De plus, si on définit une image anormale lorsqu'elle à été généré à partir d'un segment qui contient au moins une anomalie, il n'est pas évident de déduire quel pixels dans l'image présente une anomalie.

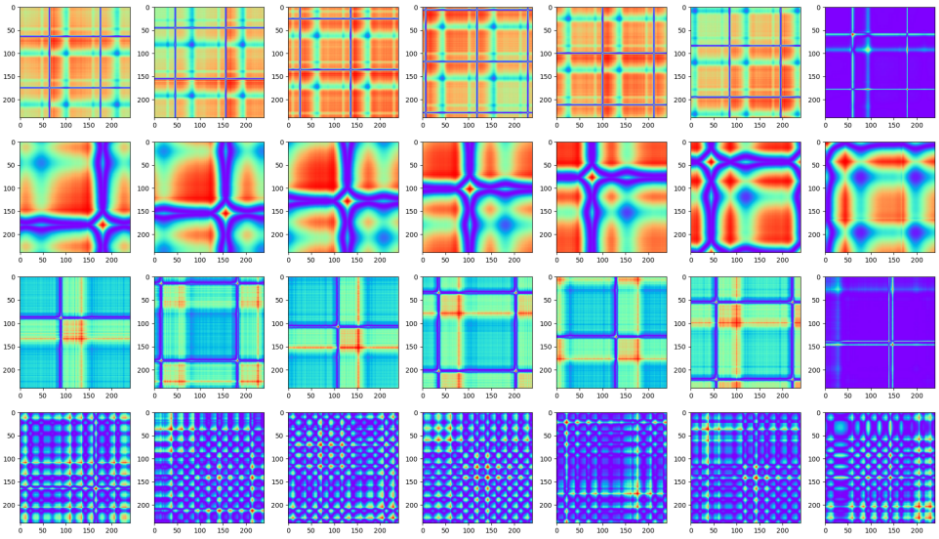


FIGURE 8 – Représentations en image de séries temporelles comportants des anomalies, chaque ligne représente une série, les 6 premières images représentent des fenêtres ne contenant pas d'anomalies et la dernière contient au moins une anomalie.

On utilise donc un modèle de détection d'anomalie nommé PatchCore[24]. Le modèle a d'abord pour avantage d'être semi-supervisé et utilise un modèle pré-entrainé pour extraire et ne nécessite pas de re-entrainer certains paramètres. Le modèle PatchCore utilise les couches intermédiaires d'un réseau de neuronne afin de créer une "banque" de variables à partir d'images ne contenant aucune anomalies. Afin de simplifier les notations supposons que on ne récupère les représentations qu'à un seul niveau du réseaux de neuronne et on note $f_\theta : \mathbb{R}^{L\times L} \to \mathbb{R}^{c\times L\times L}$ la fonction qui associe à une image sa représentation dans l'espace latent, où c représente la dimension de l'espace latent par pixels. Le principe de PatchCore est d'utiliser un dataset d'image sans anomalie pour stoquer les représentations dans un ensemble nommé "memory bank" est d'ensuite comparer les représentations des nouvelles images à celles stoquées dans la memory bank. Le modèle utilise un mécanisme de sélection dit "coreset" pour réduire la dimension des données dans la memory bank et réduire grandement le temps de calcul du modèle. L'algorithme 1 montre le fonctionnement plus en détail du modèle PatchCore.

Les résultats montrent que le modèle ne présente pas de performances meilleures que les autres modèles et peinent à détecter de nombreuses anomalies. Le modèle ne montre pas de mauvaise performances et étant non supervisé, le ROC-AUC supérieur à 0.5 montre que le modèle fait en moyenne mieux qu'un prédicteur aléatoire, mais les résultats ne sont pas assez haut pour justifier l'utilisation d'autant de ressources de calcul.

Une des raisons qui pourraient expliquer les lacunes du modèle est que le backbone pré-entrainé n'a pas été entraîné avec des données semblables (aux séries temporelles représentées en image par GAF). Ce problème est récurrent dans le domaine du Deep Learning : les modèles tendent à montrer de mauvaises performances lorsque les données d'entrées ont une distribution qui diffère des données d'entraînement[22]. En un sens, les données fournies sont ici des anomalies par rapport aux données d'entraînements, ce qui pourrait expliquer les mauvaises performances. Une approche serait alors d'entraîner un modèle de traitement d'image spécifiquement sur des données semblables mais cela dépasse le contexte du stage, notre objectif est de trouver des méthodes qui ne requiert pas l'entraînement d'un modèle de Deep Learning.

D'autre méthode, plus spécifique aux séries temporelles est alors exploré en utilisant cette fois des modèles pré-entrainés spécifiques aux séries temporelles.

Algorithm 1 Algorithme PatchCore Pseudo-Code

Entraînement du modèle
$\mathscr{B} \leftarrow \{\}$
for Image in TrainLoader **do**
    $Z := f_\theta(\text{Image})$
    $\mathscr{B} \leftarrow \mathscr{B} \cup \{\text{Agg}(\{z_{i,j} \mid i,j \in L\})\}$
    #Etant donné que le modèle admet une représentation par pixel les $z_{i,j}$, on agrège les informations par pixels à l'aide d'une fonction "Agg"
end for
$\mathscr{C} \leftarrow \{\}$
for i in N # *N est la dimension de l'espace réduit d'arrivée* **do**
    $p_i^* = \arg\max_{i \in \mathscr{B}} \min_{j \in \mathscr{C}} \|\psi(i) - \psi(j)\|_2$
        # ψ *est une random projection linéaire*
    $\mathscr{C} \leftarrow \mathscr{C} \cup \{p_i^*\}$
end for
# Application du modèle
for Image in TestLoader **do**
    $Z := f_\theta(\text{Image})$
    Score_Anomaly $= g(s)$ # où $s = \|m^{\text{test}} - m\|_2$, m^{test} et m sont les points les plus éloignés entre Z et $\mathscr{C}$ et g est une fonction croissante en s
end for=0

3.2 Modèles de forecasting pré-entrainés pour la détection d'anomalies

La première méthode utilisée repose sur les méthodes de forecasting pour la détection d'anomalies, mais en entraînant pas un modèle sur la série temporelle mais en utilisant un modèle pré-entrainé sur un grand dataset de séries temporelles. On dispose de trois modèles pré-entrainés accessibles librement : LagLlama[23], MoirAi[31] et Moment[10]. Les trois modèles sont basés sur une architecture type transformers[28]. Les transformers ont montré de très bons résultats lorsqu'ils sont entraînés avec un grand nombre de données et sont à la base des modèles tels que GPT, Llama 2 et Mistral large. La différence entre les trois modèles utilisés est essentiellement dù aux nombres de paramètres, et les données qui sont donnés à l'entrée des transformers. Par exemple le modèle Moirai découpe les séquences temporelles en "patch" qui sont fournis ensuite au transformer tandis que le modèle LagLlama traite les points séparemment mais applique une projection linéaire pour étendre la dimension des données avant qu'elles soient traités par le transformers.

Ces trois modèles utilisent cependant une même structure général d'entraînement afin de pouvoir s'adapter à la taille d'entrée des séquences fournies. En effet un modèle pré-entrainés pour la prédiction de données temporelles doit pouvoir performer avec des séries de différentes longueurs. Or, les transformers (contrairement aux réseaux Long Short Term Memory ou Recurrent Neural Network par exemple) nécessitent une taille fixe d'entrée. De

la même manière que de nombreux Large Language Models, les modèles utilisent un "mask" afin d'indiquer au modèle si la donnée doit être prise en compte. Par exemple, supposons que la dimension des données pour le modèle est de 2048, alors afin d'utiliser une donnée de dimension moindre L, on transforme la donnée de la manière suivante :

$$(x_1, x_2, ...x_L) \rightarrow ([mask], [mask], ...[mask], x_1, x_2, ...x_L)$$

L'intégration d'un "mask" permet ainsi de pouvoir gérer des inputs de différentes longueurs, et afin de garentir de bonnes performances lorsque le modèle est appliqué, la taille d'entrée des séquences varie pendant l'entraînement.

Ensuite, pour pouvoir faire varier la taille de la fenêtre de prédiction, les modèles utilisent de manière similaire à de nombreux LLM (tels que GPT et Mistral) une manière auto-regressive :

$$\hat{x}_{t+1|t} = f(x_1, ...x_L)$$

$$\hat{x}_{t+2|t} = f(x_1, ...x_L, \hat{x}_{t+1|t})$$

$$...$$

$$\hat{x}_{t+h|t} = f(x_1, ...\hat{x}_{t+h-2|t}, \hat{x}_{t+h-1|t})$$

Afin de reconstruire une séquence $(\hat{x}_{t+1|t}, ..\hat{x}_{t+h|t})$ de taille h pouvant varier selon les besoins. Dans notre cas, on définit pour chacun des trois modèles le score d'anomalie pour un point x_t de la manière suivante :

$$A_{x_t} = |x_t - f(x_{t-1}, ...x_{t-L-1})|$$

Cependant, cette première méthode assez simple ne dépasse pas significativement les performances d'un simple prédicteur ARMA. Les résultats montrent que en moyenne ces modèles font mieux qu'un prédicteur aléatoire mais les performances ne justifient pas l'utilisation de grand modèles pré-entraînés qui nécessitent beaucoup plus de ressources de calcul qu'un prédicteur fondé sur un modèle ARMA. On va donc s'intéresser à une autre manière d'extraire un score d'anomalie à partir d'un modèle pré-entraîné.

3.3 Représentation latente pour la détection d'anomalie

De manière similaire au modèle PatchCore présenté précédemment, on cherche à extraire un score d'anomalie à partir des représentations latente d'un modèle pré-entraîné. Ici on parle spécifiquement de modèles pour les séries temporelles. Une première méthode est alors d'appliquer le mécanisme de PatchCore aux modèles de séries temporelles. Une autre

méthode qu'on appelera "MovingMahalanobis" utilise la distance de mahalanobis comme critère.

Etant donné une distribution Q sur $\mathbb{R}^p$ d'espérance μ et de variance Σ, un vecteur de données $\mathbf{x}$ la distance de mahalanobis du point $\mathbf{x} \in \mathbb{R}^p$ de Q est définie par :

$$D_M(\mathbf{x}, Q) = \sqrt{(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)}$$

La distance de mahalanobis entre deux points $\mathbf{x}$ et $\mathbf{y}$ par rapport à Q est :

$$D_M(\mathbf{x}, \mathbf{y}, Q) = \sqrt{(\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y})}$$

La distance euclidienne est donc un cas particuliers de la distance de mahalanobis, dans le cas où $\Sigma = I_p$. On remarque que dans le cas où Q est une loi Normale multivariée et $\mathbf{x} \sim Q$, alors $D_M^2(\mathbf{x}, Q) \sim \chi^2(p)$. La distance de mahalanobis a pour avantage d'être très sensible aux outliers et est donc souvent utilisé pour la détection d'anomalies dans le cas de données multivariées[8]. On définit donc la méthode "Moving Mahalanobis" de la manière suivante :

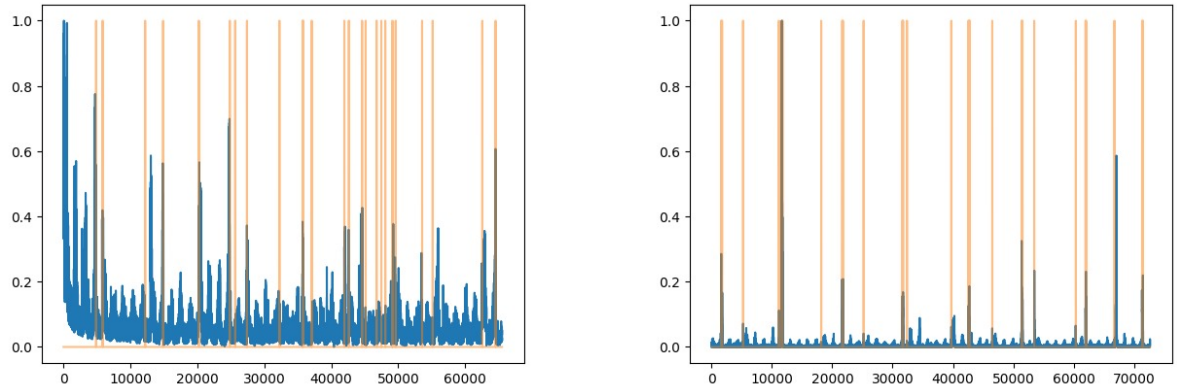
On considère la fonction $f_m : \mathbb{R}^L \to \mathbb{R}^{C_m \times L}$, qui associe à une fenêtre de taille L la représentation latente dans le modèle m de dimension C_m. On applique le modèle à des fenêtres de la série $(\mathbf{X}_1, \cdots, \mathbf{X}_S)$ de taille L sans overlap. On note $Z_s := f_m(\mathbf{X}_s) = (z_1^{(s)}, ...z_L^{(s)})$. Pour chaque fenêtre $\mathbf{X}_s$ on définit le score d'anomalie pour chaque point de $\mathbf{X}_s$ par sa distance (au sens de mahalanobis) par rapport à la moyenne et la covariance estimée à partir des points $z_i^{(s-1)}, \quad z_i^{(s-2)}, \cdots, z_i^{(1)}$.

On suppose donc que les données sans anomalies ont des représentations z_i proches dans l'espace latent du modèle tandis que les anomalies ont une représentation éloignée. Cette est totalement non supervisée, elle ne nécessite pas de disposer d'un jeu de données sans anomalies pour entraîner le modèle.

La dimension C_m de l'espace latent d'un modèle peu être très elevé (pour le modèle LagLlama, elle est par d'environ ≈ 1400) ce qui peut rendre l'algorithme très long (particulièrement pour l'inversion de la matrice $\hat{\Sigma}$) et toute les informations contenues dans l'espace latent ne sont pas forcément utile. Afin de réduire la dimension des z_i, on applique un mécanisme de "coreset selection" de manière similaire au modèle PatchCore. Etant donné une taille désirée p, les $p+1$ premières fenêtres vues par le modèle servent à estimer la moyenne et la covariance des z_i et à la $p+2$ itération, on applique le mécanisem "coreset" pour sélectionner les dimensions les plus pertinentes.

De plus, il est pertinent de ne pas calculer le score des $p+2$ premières estimations car elles sont biaisés et seront plus élevée que les autres estimations. La figure 9a montre que les premiers scores du modèles sont très élevés. La figure 9b montre les scores après un

ajustement du score.



(a) AnomalyScore sans ajustement

(b) AnomalyScore avec ajustement

FIGURE 9 – Score d'anomaly Moving Mahalanobis

Afin d'ajuster le score, on se place dans un cadre hypothétique où les variables z_i sont indépendant et identiquement distribués selon une loi normale. L'hypothèse est forte et particulièrement l'indépendance des données étant donnée qu'on traite des séries temporelles. Pour rappel, on calcule $\overline{z}_i^{(s)} = \frac{1}{s}\sum_{t=0}^{s} z_i^{(t)}$. Les estimations portent donc sur des points espacés de L points, l'hypothèse d'indépendance des z_i est alors plus plausibles si la taille de la fenêtre L est relativement grande.

Soit $X_1, \cdots, X_n \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \Sigma)$ de dimension p. Alors on a :

$$D_M^2(X_i, \mu, \Sigma) \sim \chi^2(p)$$

Et en ordonnant les X_i et en notant :

$$\overline{\mu}_i = \frac{1}{i}\sum_{j=1}^{i} X_j$$

$$\hat{\Sigma}_i = \frac{1}{i-1}\sum_{j=1}^{i} (X_j - \overline{\mu}_i)(X_j - \overline{\mu}_i)'$$

On a :

$$\frac{i-p}{(i-1)p} D_M^2(X_{i+1}, \overline{\mu}_i, \hat{\Sigma}_i) \sim F_{i,i-p}$$

$$E\left(\frac{i-p-2}{i-1} D_M^2(X_{i+1}, \overline{\mu}_i, \hat{\Sigma}_i)\right) = p$$

Pour tout $i \in \{p+3, \cdots, n-1\}$ et en notant $F_{p,k}$ la loi de Fisher de paramètre p, k. Les résultats sont issues du papier Hardin & Rocke 2005[11]. On obtient donc un ajusteur naturel $\frac{i-p-2}{i-1}$ pour le score d'anomalies des points x_j tels que $\lfloor \frac{j}{L} \rfloor = i$. Bien que ces résultats

sont valables sous l'hypothèse forte que les variables z_i sont indépendant et identiquement distribués de lois normales, en pratique l'ajusteur permet bien de diminuer la forte valeur des premiers points estimés par le modèle.

Le pseudo-code ci-dessous résume les étapes du modèle.

Algorithm 2 Algorithme Moving Mahalanobis pseudo-code

Input modèle f, Nombre de couches à garder p

AnomalyScores = { }

EmbeddingBanks = { }

for $\mathbf{X}_s$ in Dataset **do**
  $Z_s \leftarrow f_\theta(\mathbf{X}_s)$
if $s < C_m + 2$ **then**
    EmbeddingBanks $\leftarrow Z_s$
    pass
end if
if $s = C_m + 2$ **then**
    EmbeddingBanks $=$ CoresetSelection(EmbeddingBanks, p)
    pass
end if
  for $i \in [1, \ldots, L]$ **do**
    AnomalyScores $\leftarrow D_m^2(z_i^{(s)}, \overline{z_i}^{(s-1)}, \hat{\Sigma}_i^{(s-1)}) * \frac{s-p-2}{s-1}$
    # Où les $\overline{z_i}^{(s-1)}$ et $\hat{\Sigma}_i^{(s-1)}$ sont calculés à partir de EmbeddingBanks
  end for
  EmbeddingBanks $\leftarrow Z_s$
end for
Return AnomalyScores =0

Une version semi-supervisé de cet algorithme est aussi possible. Plutôt que d'ajuster les estimations à chaque itération du modèle, on peut entraîner le modèle seulement sur des données sans anomalies et on applique ensuite le modèle sur le reste des données sans modifier les estimations des représentations moyenne du modèle. L'ajustement du score ne fait alors plus sens et est retiré du modèle.

Une fois l'algorithme défini permettant d'extraire un score d'anomalie à partir de l'espace latent d'un modèle, il reste maintenant à explorer différends modèles qui pourraient être utiliser pour la détection d'anomalie. Plusieurs modèles ont été étudiés, tels que des modèles de forecasting, des modèles de classification et des modèles de représentation. Les modèles utilisés n'ont pas été entraînés sur les datasets de test pour la tâche de détection d'anomalie et les poids du modèles [5] ont été récupérés sur huggingface lorsque ceux ci étaient disponibles, et dans le cas contraire j'ai entraîné les modèles sur une série annexe. Les modèles utilisées

5. Dans un réseau de neuronne, les poids du modèles correspondent aux paramètres estimées

ont tous pour point commun de pouvoir présenter une représentation latente "par points", c'est à dire que si la dimension d'entrée du modèle est une série temporelle de taille L, alors la représentation latente du modèle sera de taille $L \times C_m$. On se référera à ces modèles en parlant de "BackBone"

3.4 Modèle BackBone pour la détection d'anomalie

Afin de sélectionner des modèles potentiels pour être utilisé en backbone au modèle MovingMahalanobis, de nombreux modèles ont été étudiés et on présente dans ce rapport seulement trois modèles qui diffèrent en de nombreux points : LagLlama[23], LiTE[14] et TS2Vec[35].

a) LagLlama

Comme présenté précedemment, le modèle LagLlama est un modèle de forecasting entraîné pour être utilisé en Zero-Shot où Few-Shot learning. Il utilise une architecture de type "transformer" basée sur LLaMA. Sa particularité est de prédire non seulement des valeurs futures précises, mais aussi des distributions de probabilité, ce qui permet d'évaluer les incertitudes associées aux prédictions. Le modèle utilise des décodeurs d'un réseau type transformers et le transformers s'entraîne à prédir les paramètres d'une distribution t-student (Moyenne, Degrés de libertés et l'échelle) afin de prédire le point suivant.

b) LiTE

Le modèle Light Inception with boosTing tEchniques for Time Series Classification (LiTE) est un modèle de classification de séries temporelles qui utilise une architecture de type Convolutional Neural Network. La particularité du modèle est qu'il utilise différents mécanismes afin de réduire la complexité et le nombre de paramètres du modèle. Un exemple de mécanisme est l'utilisation de couche de convolution à paramètres fixes (c'est à dire qui ne sont pas entrainés par le modèle) afin de fournir L'utilisation de ces mécanismes permet au modèle de montrer de bonnes performances par rapport à son besoin de données et son utilisation dans notre cas est justifié par le fait qu'on espère que le faible nombre de paramètre permet bien de filtrer et de synthétiser les informations importantes contenues dans une série.

c) TS2Vec

Le modèle TS2Vec est un modèle de représentation des séries temporelles. Le principe est fortement inspiré du modèle Word2vec[19]. Ce dernier a permis une grande avancée en traitement du langage en proposant une méthode non supervisé pour apprendre une représentation vectorielle des mots. Le modèle TS2Vec est donc un modèle qui prends en entrée

une série $(X_t) \in \mathbb{R}^{k \times L}$ où k est le nombre de covariables et L la longueur de la série, et renvoie une représentation étendue de la série : $f(X) \in \mathbb{R}^{C \times L}$ où $C \gg k$. Le modèle utilise des neuronnes convolutionnels et sa particularité se trouve dans sa fonction de perte. Le modèle utilise l'apprentissage contrastif hiérarchique : on cherche à obtenir un modèle qui peut apprendre à distinguer des données en maximisant la similarité entre des "paires positives" et en minimisant celle entre des "paires négatives". Les paires positives et négatives désignent des classes qui peuvent être différenciées. Dans l'exemple d'un modèle de computer vision, les représentations de deux images de chats devraient être proches, tandis qu'elles devraient être éloignée de celle d'un chien. Le modèle TS2Vec étant non supervisé, on ne dispose pas de labels, à la place le modèle à chaque itération d'entraînement, on sélectionne deux fenêtres de la série temporelle de manière aléatoire qui possèdent une partie commune, et on considère comme paire positive la partie commune, négative les points présents que dans une seule des deux fenêtres (voir figure 10). Ainsi la place d'un point dans la fenêtre donnée au modèle influence peu sa représentation.

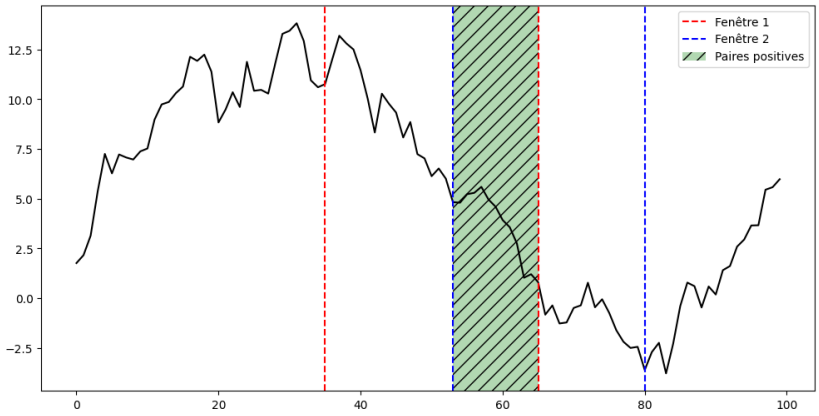
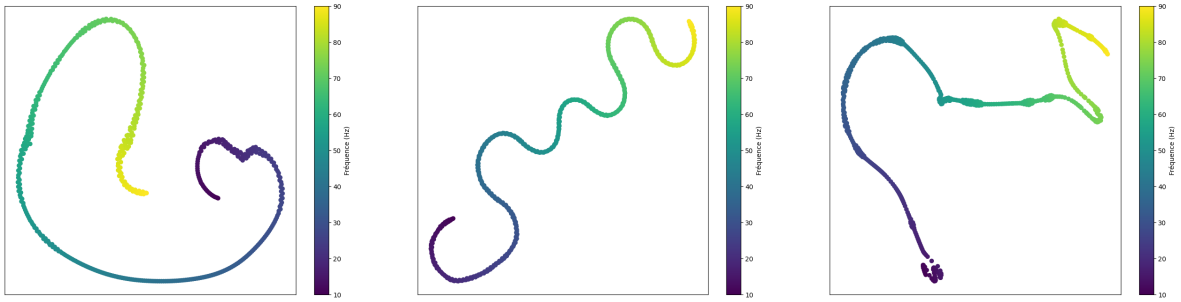


FIGURE 10 – Paires positives pour deux fenêtres aléatoires se chevauchant

Ainsi, le modèle TS2Vec est intéressant dans notre cas car il est totalement non supervisé, son but premier est d'obtenir une représentation efficace de séries temporelles.

Les trois modèles sélectionnés diffèrent en terme de taille (LagLlama $\gg$ TS2Vec $\gg$ LiTE), de type de neuronnes utilisés (CNN & transformers) et de tâches (prédiction, classification et représentation). Afin de s'assurer que la représentations latente agrège bien de l'information, on peut faire une expérience de la manière suivante :

1. Générer N séries sinusoidales de taille L de fréquences allant de 10 à 90 HZ
2. Calculer la représentation latente des séries par le modèle
3. Représenter graphiquement en deux dimensions les résultats à l'aide d'un algorithme de réduction de dimension (t-sne[17])



(a) LagLlama (b) TS2Vec (c) LiTE

FIGURE 11 – Projection T-SNE de la représentation dans l'espace latent de séries sinusoïdales

On voit bien que le graphique contient une certaine cohérence, les données ayant une fréquence proche sont proches dans la représentation en deux dimensions extraite du modèle.

4 Experimentations

Dans cette section, on présente les résultats des modèles sur la tâches de détection d'anomalies. Le code utilisé pour évaluer les modèles est disponible sur un repertoire git privé dont je peux ouvrir l'accès au lecteur si nécessaire.

Le protocole d'évaluation est défini de la manière suivante. Pour les modèles non supervisés, on évalue séquentiellement le score d'anomalie sur chaque point de la série (allant de plus ancien au plus récent) tandis que pour les modèles semi-supervisés. On prends les 20% premières observation de la série qu'on découpe en fenêtre pour l'ensemble d'entraînement et si il y a des fenêtres qui contiennent des anomalies dans l'ensemble d'entraînements, on les retire et les mets dans l'ensemble de test. On normalise toute les séries de tels sortes à ce que leurs valeurs soient comprises entre 0 et 1. Une fois qu'on a un score d'anomalie par points, on convertir le score en score par fenêtre en découpant la série en n fenêtres de taille 32 de la même manière que défini en 2.4. Puisque cette méthode d'évaluation augmente la part d'anomalie dans le dataset, lorsque la part d'anomalie dans le dataset est supérieur à 30% de la série, on ne prends pas en compte le résultat du modèle. Etant donné que les sorties du modèles ne sont pas des prédictions de labels mais le score d'anomalie, on calcul le seuil de classification δ en maximisant le F1 score en fonction de δ. On affiche ensuite les métriques suivantes :

— L'AUC correspond à l'air sous la courbe définie par le taux d'anomalie classifiés correctements en fonction du taux de d'anomalies prédites fausses. Un score de 0.5 correspond à l'espérance d'un prédicteur aléatoire.

— L'AUPr est définie par l'air sous la courbe de la precision en fonction du rappel du modèle. Un classifieur aléatoire a en moyenne un score de λ, qui correspond à la part

d'anomalie dans la série.

— FPR@95% est le taux de prédiction d'anomalie fausse lorsque le rappel atteint 95%. Un modèle peut avoir un AUC et un F1 Score très hauts mais un FPR@95% si il une petite partie des anomalies sont très mal détectées par le modèle.

4.1 Données

Afin d'évaluer les modèles, on dispose de données issues de différents benchmarks. Un benchmark est un test de performance standardisé utilisé pour évaluer, comparer ou mesurer les performances d'un modèle sur une ou plusieurs tâches données. Les données issues d'un benchmark proviennent généralement de diverses sources et ont pour but d'évaluer un modèle sous différents scénarios. Les benchmarks de détection d'anomalies en série temporelles sont composés de différentes séries, ainsi que leur label. Les labels disponibles sont uniquement des labels binaires (1 si anomalie, 0 sinon).

a) Numenta Anomaly Benchmark (NAB)

Le dataset Numenta Anomaly Benchmark[1] est composé de 58 séries, chacune composé de 1000 à 22000 points pour un total de 365000 points. Les données ont été annotés manuellement et certaines séries disposent d'anomalie où on connait la cause. Parmis les séries, 11 sont des données artificielles qui sont composés d'une série simulé ainsi que des anomalies injectées. Le séries ont en moyenne 0.05% de points catégorisés comme anomalie.

Le dataset NAB ne contient aucune séquence d'anomalie. C'est à dire que pour une série donnée, deux anomalies ne sont jamais consécutives dans le cas de ce benchmark. Le benchmark n'est donc pas suffisant car il ne couvre pas suffisamment de types d'anomalies.

b) KPI Dataset

Le dataset KPI est issue d'un challenge sur la détection d'anomalies et présente 29 séries composées de 4000 à 73000 points pour un total de 1.5 millions de points. Les séries ont en moyenne 2% d'anomalies. Le dataset est constitué de plusieurs séries de KPI avec collectées auprès de diverses entreprises comme Sogou, Tencent et eBay. Les points sont espacés de 1 minutes. Les KPI ont généralement des patterns récurrent et les anomalies sont généralement issues d'un changement de pattern où d'une suite de points inhabituels compte tenu du contexte. Le dataset ne contient aucune anomalie composé de seulement un point isolé. Toute les anomalies sont présentes dans des segments d'anomalies composés d'au moins deux anomalies. Ce dataset peu ainsi évaluer les performances du modèles dans le cas de groupements d'annomalies.

c) NASA-SMAP

Le dataset NASA-Soil Moisture Active Passive[13] est un dataset présentant 47 séries de 5000 à 11000 points pour un total de 500000 points qui contient en moyenne 10% d'anomalies. Les séries sont issues de mesures par un satellites et dont on connait les causes réelles des anomalies. De plus, lorsque plusieurs anomalies (ou segments d'anomalies) étaient semblables, les auteurs du datasets n'en gardaient qu'une seule afin d'obtenir un jeu de données avec des anomalies diversifiées.

4.2 Résultats

On présente maintenant les résultats des modèles, on note "MM" a côté d'un modèle si son "backbone" a été utilisé pour l'algorithme Moving Mahalanobis, "FM" (Pour Fixed Mahalanobis) si l'algorithme utilisé est la version fixe et semi-supervisée du modèle Moving Mahalanobis.

Model	**Supervision**	**Training Dataset**	**Tâche**
ARMA	Non supervisé	Train et appliqué sur le même dataset	Forecasting
PatchCore	Semi supervisé	Train sur un dataset d'images	Representation
Donut	Semi-supervisé	Train et appliqué sur le même dataset	Reconstruction
DeepAnt	Non supervisé	Train et appliqué sur le même dataset	Forecasting
LagLlama MM	Non supervisé	Train sur un dataset annexe	Representation
LagLlama forecasting	Non supervisé	Train sur un dataset annexe	Forecasting
TS2Vec MM	Non supervisé	Train sur un dataset annexe (ETTh2)	Representation
LiTE MM	Non supervisé	Train sur un dataset annexe (UCR)	Representation
Moment forecasting	Non supervisé	Train sur un dataset annexe	Forecasting
Moirai forecasting	Non supervisé	Train sur un dataset annexe	Forecasting

TABLE 1 – Résumé des modèles

	NAB				**KPI**				**SMAP**			
Model	F1	AUC	AUPR	FPR	F1	AUC	AUPR	FPR	F1	AUC	AUPR	FPR
Donut	0.41	0.61	0.32	0.52	0.45	0.79	0.37	**0.30**	0.52	0.68	0.43	0.68
ARMA	0.34	0.7	0.25	0.4	0.51	0.8	0.36	0.57	-	-	-	-
DeepAnt	**0.66**	**0.83**	**0.63**	**0.25**	0.67	0.85	0.38	0.35	0.55	0.69	0.42	0.65
PatchCore	0.34	0.61	0.26	0.71	0.44	0.72	0.35	0.51	-	-	-	-
LagLlama MM	0.4	0.70	0.30	0.38	0.47	0.86	0.36	0.41	0.51	0.63	0.41	0.72
LagLlama FM	0.63	0.82	0.55	0.28	0.43	0.70	0.35	0.34	0.52	0.64	0.43	0.71
LagLlama forecasting	0.45	0.79	0.37	0.3	0.36	0.75	0.29	0.65	0.46	0.62	0.34	0.77
TS2Vec MM	0.65	**0.83**	0.62	**0.25**	**0.74**	**0.9**	0.38	0.38	**0.64**	**0.79**	**0.58**	**0.52**
TS2Vec FM	0.55	0.70	0.4	0.33	0.45	0.72	0.27	0.41	0.58	0.71	0.49	0.67
LiTE MM	0.51	0.8	0.32	0.29	0.52	0.86	0.38	0.61	0.53	0.7	0.42	0.68
LiTE FM	0.63	0.82	0.55	0.28	0.43	0.70	0.35	0.34	0.49	0.66	0.4	0.71
Moment forecasting	0.56	**0.83**	0.49	0.29	0.5	0.81	**0.44**	0.48	0.49	0.64	0.39	0.75
Moirai forecasting	0.37	0.78	0.28	0.31	0.34	0.77	0.25	0.59	-	-	-	-

TABLE 2 – Résultats des modèles. Les meilleurs scores sont en gras et les deuxièmes meilleurs en bleu

Les résultats varient selon le dataset présenté. On remarque cependant que la méthode Moving Mahalanobis en utilisant le backcbone TS2Vec obtient de bons résultats sur l'ensemble des datasets. La méthode Moving Mahalanobis a globalement de bons résultats et semble être un bon modèle de détection d'anomalies.

On remarque que la méthode Fixed Mahalanobis (FM) obtient généralement de moins bons résultats que la méthode Moving Mahalanobis. La méthode Fixed Mahalanobis n'estime les paramètres uniquement à partir d'une partie des données sans anomalies. On peut expliquer ce résultats de plusieurs manières. Tout d'abord, lorsque la série ne possède pas un grand nombre de points, le modèle Fixed Mahalanobis dispose de peu d'observations pour estimer les paramètres du modèle et peu avoir des estimations peu précises. De plus, le caractère dynamique de l'algorithme Moving Mahalanobis permet d'ajuster les paramètres du modèle si la distribution des données change au cours de la série. La figure 12 montre l'exemple d'une série issue du dataset KPI et on remarque que la distribution des premières observations (rouge) semble différentes de la distribution de la fin de la série (bleu) sans pour autant que cela est dù à une anomalie. Dans ce cas, les modèles Moving Mahalanobis et Fixed Mahalanobis vont mal classifiés les premiers points (en bleu), mais le score associé par le modèle Moving Mahalanobis décroit au fur et a mesure que les paramètres du modèle s'ajuste, tandis que le modèle Fixed Mahalanobis continue à associé un score trop élevé aux observations.

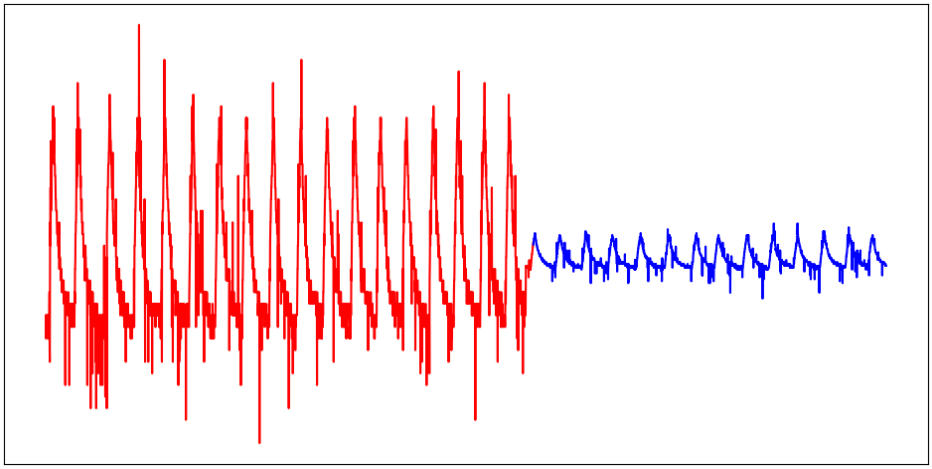


FIGURE 12 – Exemple de série où la distribution change mais ne correspond pas à une anomalie

Ensuite, une raison qui pourrait expliquer les meilleurs performances en utilisant un backbone TS2Vec que LagLlama peut être du au fait que avec la représentation d'un point dans le modèle TS2Vec dépends peu de sa place dans la fenêtre. Prenons un exemple où :

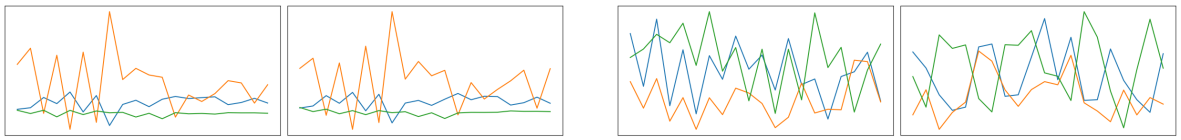
$$\mathbf{X}_s = (x_1, x_2, \quad \dots \quad x_{50})$$

$$\mathbf{X}_t = (x_{20}, x_{21}, \quad \dots \quad x_{70})$$

Alors on aura :

$$(f_{ts2vec}(\mathbf{X}_s)_i)_{i=20}^{50} \approx (f_{ts2vec}(\mathbf{X}_t)_i)_{i=1}^{30}$$

Ce qui n'est pas forcément vérifier pour $f_{LagLlama}$ comme le montre la figure 13. La représentation par points du modèle LagLlama n'est donc pas réellement consistante. De plus, le modèle LagLlama est un modèle de forecasting, contrairement à TS2Vec et LiTE, le modèle n'est pas entraîné à comprendre des caractéristiques des données mais seulement à prédire compte tenu du passé. La place des points au sein du segment à alors une grande importance, et il semble que la représentation latente du modèle ne soit pas adaptée pour représenter une série temporelle



(a) TS2Vec (b) LagLlama

FIGURE 13 – Représentation de même points situés à un endroits différent du segment

Ensuite, les modèles pré-entraînés pour évaluer les anomalies par la méthode du forecasting ne semblent pas bien performer. En effet les modèles n'ont pas de meilleurs performances qu'un "simple" modèle ARMA ou un modèle simple de forecasting tel que DeepAnt. De plus, l'évaluation de ces modèles par forecasting peut être très longue car elle requiert d'appeler autant de fois le modèle qu'on dispose de données afin d'avoir un score d'anomalie par points, tandis que les modèles de reconstruction traitent les données segments par segments (sans overlap) ce qui permet au modèle d'être appelé beaucoup moins de fois (on divise le nombre d'appels du modèles par la taille des séquences, 32 dans notre cas).

La méthode Moving Mahalanobis en utilisant le backbone TS2Vec semble globalement la plus performante et affiche tout le temps des scores au moins meilleurs que le modèle ARMA. La méthode possède un score AUC moyen entre 0.79 et 0.9 sur les 3 datasets ce qui montre que le modèle fait bien mieux que l'aléatoire en moyenne. On note cependant qu'une limite du modèle Moving Mahalanobis est qu'en présence d'un grand nombre d'anomalies (tels que dans le dataset SMAP qui possède en moyenne 10% d'anomalies par séries), les anomalies peuvent biaiser les paramètres du modèles étant donné que le modèle prends toute les données en compte pour estimer les paramètres du modèle.

Ainsi, les résultats montrent la capacité de l'algorithme à montrer de bonnes performances en général pour la tâche de détection d'anomalies. Les résultats montrent que le backbone choisi a une grande importance sur les performances du modèles. Une limite est que les modèles TS2Vec et LiTE ont du être entraîné de mon côté car il n'existe pas de modèles pré-entraînés disponibles publiquement pour TS2Vec et LiTE. Les modèles pré-

entraînés pour les séries temporelles sont en grande majorité des modèles de prédiction et nos résultats montrent que pour le cas de lagllama, la représentation latente du modèle ne semble pas totalement adapté pour extraire les caractéristiques des séries temporelles, contrairement aux modèles de classification et de représentation.

4.3 Conclusion & travail futur

En conclusion, le stage a eu pour but d'explorer dans un premier temps les méthodes de Deep Learning pour la détection d'anomalies appliqués aux séries temporelles. Après avoir étudié différends modèles de détection d'anomalies pour les séries temporelles ainsi que l'état de l'art pour la détection d'anomalies sur image, différentes méthodes de transfer learning pour la détection d'anomalies ont été cherchées. Dans un premier temps, nous avons étudier la méthode d'utiliser l'erreur de prédiction du modèle comme score d'anomalies. Les résultats montrent que pour trois modèles étudiés, les résultats ne dépassent pas ceux d'un simple modèle ARMA pour la détection d'anomalies ou un modèle de Deep Learning simple (DeepAnt). Ensuite, nous avons exploré des méthodes en s'inspirant de ce qui se fait en traitement d'image. L'utilisation d'un modèle de détection d'anomalies d'images en convertissant les séries temporelles en images n'ont pas montrés de résultats qui dépassent les autres méthodes existantes. On a ensuite étudié les méthodes utilisant la représentation latente des données par des réseaux de neuronnes pré-entrainés sur des séries temporelles annexes. On a ainsi définit un algorithme nommé Moving Mahalanobis qui suppose que la représentation dans l'espace latent des données sans anomalies sont similaires, la distance de mahalanobis en utilisant les représentations moyennes vu par le modèles montre de bonnes performances empiriquement.

Une limite de cette algorithme est qu'il est ici valable que pour des séries univariées. Il serait pertinent d'étudier et de modifier l'algorithme afin de voir si le modèle performe lorsque les séries sont multivariées. Ensuite, on a vu que le modèle TS2Vec présentait les meilleurs résultats, on pourrait donc étudier d'autres modèles de représentations de séries temporelles afin de voir si les performances de l'algorithme Moving Mahalanobis peuvent être améliorées. Enfin, il reste l'étude des hyperparamètres du modèle à faire plus formellement. Essentiellement pour des raisons de ressources limités, je n'ai pas eu le temps de faire une étude formelle des hyperparamètres du modèles tels que la dimension du modèle (p dans le rapport), la taille des fenêtres du modèle, la méthode de sélection des couches etc.

Références

[1] Subutai AHMAD et al. « Unsupervised real-time anomaly detection for streaming data ». In : *Neurocomputing* 262 (2017). Online Real-Time Learning Strategies for Data Streams, p. 134-147. ISSN : 0925-2312. DOI : https://doi.org/10.1016/j.neucom.2017.04.070. URL : https://www.sciencedirect.com/science/article/pii/S0925231217309864.

[2] Elif BAYKAL et al. « Transfer learning with pre-trained deep convolutional neural networks for serous cell classification ». In : *Multimedia Tools Appl.* 79.21–22 (juin 2020), p. 15593-15611. ISSN : 1380-7501. DOI : 10.1007/s11042-019-07821-9. URL : https://doi.org/10.1007/s11042-019-07821-9.

[3] Paul BONIOL, John PAPARRIZOS et Themis PALPANAS. « An Interactive Dive into Time-Series Anomaly Detection ». In : mai 2024. DOI : 10.1109/ICDE60146.2024.00409.

[4] Mohammad BRAEI et Sebastian WAGNER. *Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art*. 2020. arXiv : 2004.00433 [cs.LG]. URL : https://arxiv.org/abs/2004.00433.

[5] Varun CHANDOLA, Arindam BANERJEE et Vipin KUMAR. « Anomaly Detection: A Survey ». In : *ACM Comput. Surv.* 41 (juill. 2009). DOI : 10.1145/1541880.1541882.

[6] Thomas DEFARD et al. *PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization*. 2020. arXiv : 2011.08785 [cs.CV]. URL : https://arxiv.org/abs/2011.08785.

[7] Jacob DEVLIN et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv : 1810.04805 [cs.CL]. URL : https://arxiv.org/abs/1810.04805.

[8] Hamid GHORBANI. « MAHALANOBIS DISTANCE AND ITS APPLICATION FOR DETECTING MULTIVARIATE OUTLIERS ». In : *Facta Universitatis Series Mathematics and Informatics* 34 (oct. 2019), p. 583. DOI : 10.22190/FUMI1903583G.

[9] Abbas GOLESTANI et Robin GRAS. « Can we predict the unpredictable? » In : *Scientific Reports* 4 (oct. 2014). DOI : 10.1038/srep06834.

[10] Mononito GOSWAMI et al. *MOMENT: A Family of Open Time-series Foundation Models*. 2024. arXiv : 2402.03885 [cs.LG]. URL : https://arxiv.org/abs/2402.03885.

[11] Johanna HARDIN et David M ROCKE. « The Distribution of Robust Distances ». In : *Journal of Computational and Graphical Statistics* 14.4 (2005), p. 928-946. DOI : 10.1198/106186005X77685.

[12] Kaiming HE et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv : 1512.03385 [cs.CV]. URL : https://arxiv.org/abs/1512.03385.

[13] Kyle HUNDMAN et al. « Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding ». In : *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. KDD '18. ACM, juill. 2018, p. 387-395. DOI : 10.1145/3219819.3219845. URL : http://dx.doi.org/10.1145/3219819.3219845.

[14] Ali ISMAIL-FAWAZ et al. « LITE: Light Inception with boosTing tEchniques for Time Series Classification ». In : *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*. 2023, p. 1-10. DOI : 10.1109/DSAA60987.2023.10302569.

[15] Diederik P KINGMA et Max WELLING. *Auto-Encoding Variational Bayes*. 2022. arXiv : 1312.6114 [stat.ML]. URL : https://arxiv.org/abs/1312.6114.

[16] Yann LECUN, Y. BENGIO et Geoffrey HINTON. « Deep Learning ». In : *Nature* 521 (mai 2015), p. 436-44. DOI : 10.1038/nature14539.

[17] Laurens van der MAATEN et Geoffrey HINTON. « Visualizing Data using t-SNE ». In : *Journal of Machine Learning Research* 9.86 (2008), p. 2579-2605. URL : http://jmlr.org/papers/v9/vandermaaten08a.html.

[18] Pankaj MALHOTRA et al. « Long Short Term Memory Networks for Anomaly Detection in Time Series ». In : *The European Symposium on Artificial Neural Networks*. 2015. URL : https://api.semanticscholar.org/CorpusID:43680425.

[19] Tomas MIKOLOV et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv : 1301.3781 [cs.CL]. URL : https://arxiv.org/abs/1301.3781.

[20] Mohsin MUNIR et al. « DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series ». In : *IEEE Access* 7 (2019), p. 1991-2005. DOI : 10.1109/ACCESS.2018.2886457.

[21] Pramuditha PERERA, Ramesh NALLAPATI et Bing XIANG. *OCGAN: One-class Novelty Detection Using GANs with Constrained Latent Representations*. 2019. arXiv : 1903.08550 [cs.CV]. URL : https://arxiv.org/abs/1903.08550.

[22] Marco A.F. PIMENTEL et al. « A review of novelty detection ». In : *Signal Processing* 99 (2014), p. 215-249. ISSN : 0165-1684. DOI : https://doi.org/10.1016/j.sigpro.2013.12.026. URL : https://www.sciencedirect.com/science/article/pii/S016516841300515X.

[23] Kashif RASUL et al. *Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting*. 2024. arXiv : 2310.08278 [cs.LG]. URL : https://arxiv.org/abs/2310.08278.

[24] Karsten ROTH et al. *Towards Total Recall in Industrial Anomaly Detection*. 2022. arXiv : 2106.08265 [cs.CV]. URL : https://arxiv.org/abs/2106.08265.

[25] D. E. RUMELHART, G. E. HINTON et R. J. WILLIAMS. « Learning internal representations by error propagation ». In : *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA : MIT Press, 1986, p. 318-362. ISBN : 026268053X.

[26] Sebastian SCHMIDL, Phillip WENIG et Thorsten PAPENBROCK. « Anomaly Detection in Time Series: A Comprehensive Evaluation ». In : *Proceedings of the VLDB Endowment (PVLDB)* 15.9 (2022), p. 1779-1797. DOI : 10.14778/3538598.3538602.

[27] Joshua B. TENENBAUM, Vin de SILVA et John C. LANGFORD. « A Global Geometric Framework for Nonlinear Dimensionality Reduction ». In : *Science* 290.5500 (2000), p. 2319.

[28] Ashish VASWANI et al. *Attention Is All You Need*. 2023. arXiv : 1706.03762 [cs.CL]. URL : https://arxiv.org/abs/1706.03762.

[29] Zhiguang WANG et Tim OATES. *Imaging Time-Series to Improve Classification and Imputation*. 2015. arXiv : 1506.00327 [cs.LG]. URL : https://arxiv.org/abs/1506.00327.

[30] Lawrence WONG et al. *AER: Auto-Encoder with Regression for Time Series Anomaly Detection*. 2022. arXiv : 2212.13558 [cs.LG]. URL : https://arxiv.org/abs/2212.13558.

[31] Gerald WOO et al. *Unified Training of Universal Time Series Forecasting Transformers*. 2024. arXiv : 2402.02592 [cs.LG]. URL : https://arxiv.org/abs/2402.02592.

[32] Haowen XU et al. « Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications ». In : *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. WWW '18. ACM Press, 2018, p. 187-196. DOI : 10.1145/3178876.3185996. URL : http://dx.doi.org/10.1145/3178876.3185996.

[33] Jiehui XU et al. *Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy*. 2022. arXiv : 2110.02642 [cs.LG]. URL : https://arxiv.org/abs/2110.02642.

[34] Yiyuan YANG, Haifeng ZHANG et Yi LI. « Long-Distance Pipeline Safety Early Warning: A Distributed Optical Fiber Sensing Semi-Supervised Learning Method ». In : *IEEE Sensors Journal* 21.17 (2021), p. 19453-19461. DOI : 10.1109/JSEN.2021.3087537.

[35] Zhihan YUE et al. *TS2Vec: Towards Universal Representation of Time Series*. 2022. arXiv : 2106.10466 [cs.LG]. URL : https://arxiv.org/abs/2106.10466.