RÉPUBLIQUE DU CAMEROUN
****
Paix - Travail - Patrie
****
UNIVERSITÉ DE YAOUNDÉ I
****
FACULTÉ DES SCIENCES
****
DÉPARTEMENT D'INFORMATIQUE
****

REPUBLIC OF CAMEROON
****
Peace - Work - Fatherland
****
UNIVERSITY OF YAOUNDE I
****
FACULTY OF SCIENCE
****
DEPARTMENT OF COMPUTER SCIENCE
****

MÉTHODE EXPLICABLE BASÉE SUR LES AUTOENCODEURS POUR LA DÉTECTION D'ANOMALIES DANS LES FLUX DE DONNÉES

En vue de l'obtention du diplôme de Master Recherche en Informatique

Option :
Science des données

Présenté par :
NAKAM YOPDUP Manuella Kristeva

Matricule :
19M2233

Sous la direction de : *Pr. Norbert TSOPZE*, UY1

Année académique 2023 / 2024

Université de Yaoundé 1
Département d'Informatique

MÉTHODE EXPLICABLE BASÉE SUR LES AUTOENCODEURS POUR LA DÉTECTION D'ANOMALIES DANS LES FLUX DE DONNÉES

Présenté par :
NAKAM YOPDUP Manuella Kristeva

Matricule :
19M2233

Superviseur : *Pr. Norbert TSOPZE*, UY1

Année académique 2023 / 2024

DÉDICACES

Je dédie ce mémoire à ma très chère famille.

REMERCIEMENTS

Avant toute chose, je remercie notre créateur Dieu le père pour la grâce qu'il m'a accordé d'être en santé afin de pouvoir effectuer ce travail et son soutien indéfectible durant toute cette période. Je témoigne ensuite ma gratitude et ma reconnaissance à toutes ces personnes qui ont contribué à la réalisation de ce travail :

À mon directeur, Pr Norbert TSOPZE, pour m'avoir donné l'opportunité d'explorer un thème très intéressant, d'avoir toujours été disponible face aux préoccupations et difficultés rencontrées.

Aux membres de mon jury de soutenance, pour m'avoir fait l'honneur d'accepter d'évaluer ce travail.

Au chef de département informatique Dr Halidou AMINOU qui m'autorise à soutenir en ligne.

Au Dr Armel NZEKON et Dr Thomas MESSI pour leurs encouragements et leurs conseils en particulier d'éviter de trainer inutilement.

À l'Université de Yaoundé 1 et au département d'informatique en particulier, pour la qualité de la formation que j'ai reçue.

Au Pr Emmanuel KAMGNIA pour ses multiples encouragements, conseils et son incitation au travail acharné.

À mes aînés académiques Germès OBIANG, Marivonne KEUBOU, Audrey FONGUE et autres pour leurs remarques, conseils et leur disponibilité.

À mes très chers parents pour leur immense amour, leur patience, leur soutien indéfectible durant toute cette période, leurs encouragements et leurs conseils. A toutes mes soeurs, ma marraine, mes oncles et tantes pour leurs encouragements et conseils.

À mes camarades particulièrement Victor DJIEMBOU qui m'a toujours assisté avant et après mon depart, Belviane KEUFACK qui m'a aidé à déposer ce mémoire, Melvin FOKAM qui m'a accompagné dans ce travail et tous les autres pour leur soutien.

Enfin, j'exprime toute ma gratitude à mes amis Dorine NGUEUMEN, Merveille DJEUMEZA, Sarah DOMO, Voltaire TCHAMBA, Gaby NJOUNOU, Aurel TOUKAM, Benjamin FOSSO et tous ceux que je n'ai pas cité pour tout leur amour et leur soutien.

RÉSUMÉ

La détection d'anomalies en temps réel dans les données est cruciale dans divers domaines tels que la santé, la surveillance satellitaire et les systèmes industriels. L'objectif est d'identifier le comportement anormal ou inattendu d'un système ou d'un processus. Dans des contextes critiques, identifier ce comportement anormal peut s'avérer insuffisant ; il devient donc nécessaire d'expliquer ou justifier cette prédiction. Plusieurs algorithmes ont été proposés pour expliquer les anomalies détectées dans les flux de données notamment en s'appuyant sur les autoencodeurs qui se sont revélés particulièrement efficaces dans la détection d'anomalies. Cependant, ces algorithmes peuvent s'avérer inefficaces à la longue en raison de la non-prise en compte du changement de la distribution des données dans le temps. Dans ce mémoire, nous proposons une solution à cette limitation en intégrant la mise à jour de l'autoencodeur utilisé pour la détection en s'appuyant sur les prototypes calculés à partir de la représentation latente de l'autoencodeur. Nous proposons deux algorithmes appelés "Update Based Prototypes" pour la detection d'anomalies avec une mise à jour basée sur les prototypes et "ARCANA Prototypes" pour expliquer les anomalies détectées dans le temps. Les résultats expérimentaux montrent que l'algorithme "Update Based Prototypes" proposé permet non seulement de détecter 79% des anomalies mais aussi d'obtenir un taux de fausses alarmes équivalent à 21 % . Contrairement à l'implémentation sans mise à jour qui a un taux de fausses alarmes égal à 49%. Ceci nous indique que le modèle est capable de maintenir de bonnes performances de détection, même face à des variations dans les caractéristiques des données ou à l'apparition de nouveaux types d'anomalies. Pour ce qui est de "ARCANA Prototype", les expérimentations montrent que les explications fournies par le modèle sont fidèles ce qui permet de valider l'utilisation des prototypes pour l'explicabilité des anomalies.

Mots-clés : Autoencodeurs, Flux de données, Détection d'anomalies, Méthode explicable.

ABSTRACT

The detection of real-time anomalies in data is crucial, with many applications in diverse fields such as health, satellite monitoring and industrial systems. The aim is to identify abnormal or unexpected behavior in a system or process. In critical contexts, identifying such abnormal behavior may prove insufficient, so it becomes necessary to explain or justify this prediction. Several algorithms have been proposed to explain anomalies detected in data streams, notably based on autoencoders, which have proved particularly effective in anomaly detection. However, these algorithms can prove inefficient in the long run, as they fail to take into account changes in data distribution over time. In this report, we propose a solution to this limitation by integrating the updating of the autoencoder based on prototypes computed from the latent representation of the autoencoder. We propose two algorithms : "Update Based Prototypes" for anomaly detection based on prototype updates, and "ARCANA Prototypes" for detecting and explaining anomalies over time. over time. Experimental results show that the proposed "Update Based Prototypes" algorithm not only detects 79% of anomalies, but also achieves a false alarm rate equivalent to 21%. This contrasts with the non-update implementation, which has a false alarm rate of 49%. This tells us that the able to maintain good detection performance, even in the face of variations in data in data characteristics or the appearance of new types of anomalies. As for "ARCANA Prototype", experiments show that the explanations provided by the model are accurate, validating the use of prototypes for anomaly explicability.

Keywords : Autoencoders, Data stream, Anomaly detection, Explainable method.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION GÉNÉRALE

Le monde est de plus en plus connecté et numérique ; qu'ils s'agissent de capteurs IoT, des réseaux sociaux ou encore des transactions financières, les données sont produites en continu avec pour contrainte, l'impossibilité de tout stocker avant de les traiter . Cette production de données en continue est connue sous le nom de flux de données (data stream en anglais). [1] Dans la vie réelle, les montres connectées constituent un des exemples concrets des flux de données car elles collectent en temps réel des données sur la fréquence cardiaque, le sommeil, ou l'activité physique d'une personne. Ces données peuvent être analysées instantanément afin de détecter des anomalies de santé, ou alerter en cas d'urgence médicale. Aussi, il existe des satellites équipés de radars et les capteurs sismiques dont le but est de surveiller les tremblements de terre, les tsunamis, les éruptions volcaniques, et les glissements de terrain. Ces données sont collectées et analysées en temps réel pour la détection précoce des catastrophes naturelles.

L'analyse de ces flux de données pour l'identification des observations, des événements ou des points de données qui s'écartent de la norme ou des attentes constitue une branche appelée la détection d'anomalies [2]. Par ailleurs, dans certains domaines sensibles comme la finance, la santé, ou la sécurité, détecter une anomalie n'est pas suffisant ; il faut fournir aux experts du domaine la raison de ces comportements inhabituels cela permet d'augmenter la confiance du modèle et d'assurer ainsi un usage éthique. Ainsi, fournir des raisons qui ont conduit à ces décisions est appelée l'explicabilité[17].

Contrairement aux données statiques qui sont entièrement stockées dans les bases de données et fichiers, les flux de données ne peuvent pas être entièrement stockés. Des méthodes adaptées ont été développées pour capturer, stocker et analyser ces données. Plusieurs travaux de recherche basés sur le machine learning ont été proposés pour la détection d'anomalies. Les méthodes basées sur les autoencodeurs [3][14] sont de plus en plus prisées en raison de leur robustesse bien que leurs résultats soient difficiles à expliquer.

Ce mémoire traite de l'explicabilité des modèles de détection d'anomalies. Dans la littérature, de nombreux travaux traitent de cette problématique notamment celle de l'explicabilité des anomalies detectées par les autoencodeurs tel que Anomaly Root CAuse ANAlysis (ARCANA) [19]. ARCANA est méthode d'optimisation qui est utilisée pour expliquer les anomalies detectées par un autoencodeur. Le but de cette méthode est de trouver un vecteur biais représentant la déviation des caractéristiques ayant causées l'anomalie. Cependant bien qu'étant dans un contexte de flux de

1. https://www.1min30.com/data-marketing/flux-de-donnees-1287547970(17-08-2024)
2. https://www.ibm.com/fr-fr/topics/anomaly-detection(30-09-2024)

données ce qui implique que la distribution des données peut changer dans le temps ce phénomène est connu sous le nom de dérive conceptuelle (concept drift en anglais) , les auteurs d'ARCANA ne mentionnent pas la mise à jour de l'autoencodeur utilisé pour la détection d'anomalies ; cela pourrait influencer les explications données par ARCANA car l'autoencodeur intervient dans la formule de l'erreur de reconstruction qu'on vise à minimiser . Le problème adressé dans ce mémoire est donc : La prise en compte de la dérive conceptuelle par la mise à jour de l'autoencodeur de détection d'anomalies au fil du temps afin d'améliorer les explications données par ARCANA .

Dans le but d'atteindre cet objectif, nous proposons le modèle supervisé "Update Based Prototypes" pour la détection d'anomalies Notre approche repose sur les prototypes qui correspondent à une représentation centrale des caractéristiques de la classe qu'ils définissent. Nous utilisons donc une extension des autoencodeurs classiques à savoir un autoencodeur variationnel qui apprend une représentation latente des données permettant ainsi d'obtenir des prototypes des différentes classes. Cette représentation latente est utilisée pour extraire des prototypes caractéristiques des différentes classes des données. Ensuite, ces prototypes seront utilisés pour détecter des changements dans la distribution des données, indiquant un éventuel concept drift. Le modèle sera donc mis à jour lorsqu'un changement dans la distribution des données sera détectée. Pour ce qui est de l'explicabilité, nous proposons d'utiliser également les propriétés des prototypes afin d'obtenir une extension de la méthode ARCANA qui permettra d'expliquer les anomalies détectées par le modèle "Update Based Prototypes". Cette extension est nommée "ARCANA prototype".

Le contenu de ce mémoire est organisé comme suit : le chapitre 1 présente d'une part les bases nécessaires à la compréhension de la recherche, couvrant les flux de donnees, les autoencodeurs, la detection d'anomalies et l'explicabilite. Et d'autre part une revue de la littérature sur la détection d'anomalies et l'explicabilité dans les flux de données. Dans le chapitre 3, nous détaillons la méthodologie utilisée ensuite dans le chapitre 4 nous présentons nos expérimentations et les résultats obtenus et enfin, nous concluons ce document en résumant les principaux résultats et en discutant des implications futures de notre recherche.

CHAPITRE 2

GÉNÉRALITÉS ET ÉTAT DE L'ART

Dans ce chapitre, il est question pour nous de fournir les bases nécéssaires à la bonne compréhension des concepts du domaine. Ensuite, nous explorons diverses approches, méthodologies et résultats obtenus par les chercheurs dans le domaine de la détection d'anomalies et de l'explicabilité .

2.1 Généralités

Nous commencerons par présenter les flux de données et ses propriétés. Ensuite nous parlerons des anomalies et des différentes approches de détection puis nous donnerons les éléments fondamentaux des autoencodeurs. Enfin il s'agira de définir ce qu'on entend par explicabilité.

2.1.1 Flux de données

Un flux de données (ou "data stream" en anglais) est une séquence continue de données, souvent générée en temps réel, qui peut être traitée et analysée au fur et à mesure de sa production. Contrairement aux ensembles de données statiques, les flux de données sont dynamiques, ce qui signifie qu'ils évoluent avec le temps et peuvent être infinis [4][10]. Dans la vie réelle, les flux de données peuvent provenir de diverses sources [1] :

— Les capteurs et appareils : les montres intelligentes et les trackers de fitness peuvent fournir des informations sur l'activité physique et le sommeil.

— Les transactions : chaque transaction, qu'elle soit effectuée en ligne, par carte de crédit ou autre, génère des données. Elles peuvent être utilisées pour mieux comprendre le comportement des consommateurs et prévoir les tendances futures.

Définition formelle

Un flux de données S est défini par [2] comme une séquence $\langle s_1, s_2, s_3, \ldots, s_\infty \rangle$. Nous considérons un scénario supervisé $s_i = (X, y)$, où :

— $X = [x_1, x_2, \ldots, x_D]$ avec D comme dimensionnalité de l'espace des caractéristiques, et

— y comme variable cible, qui peut être disponible ou non.

1. https://www.1min30.com/data-marketing/flux-de-donnees-1287547970(17-08-2024)

Chaque instance du flux est indépendante et tirée au hasard d'une distribution de probabilité stationnaire.

Flux de données vs Séries temporelles

Les flux de données (data streams) et les séries temporelles (time series) sont deux concepts apparentés mais distincts dans le domaine de l'analyse des données. Dans les flux, les données arrivent de manière continue et en temps réel, sans limite de temps prédéfinie contrairement aux series temporelles où elles sont organisées de manière séquentielle en fonction du temps, avec un intervalle régulier entre les observations. De plus, l'analyse des flux de données se concentre souvent sur la détection d'événements en temps réel, l'identification d'anomalies, et l'adaptation continue des modèles. Les series temporelles quant à elles, se concentrent généralement sur la prévision des valeurs futures, l'identification des tendances à long terme, et l'analyse des cycles saisonniers.

Défis liés aux flux de données

Le traitement des flux de données posent un certains nombres de contraintes en raison de leur volume, leur vitesse et de la qualite des données [8]. Aussi, le phénomène de concept drift qui affecte les distributions statistiques des données au fil du temps est un élément à prendre en compte dans le traitement de flux de données.

Dérive conceptuelle

La dérive conceptuelle ou concept drift en anglais est le fait que la distribution sous-jacente des données change avec le temps[11].

Ainsi, la figure 2.1 [1] nous présente les différents scénarios liés au concept drift. On observe 6 principales distributions qui peuvent être interpretées comme suit :

a) Original Distribution of Data : Ce graphique presente la distribution initiale des données, avec des points de deux classes différentes (cercles et triangles). Les données sont bien séparées par une frontière de décision, ce qui permet un bon apprentissage et une classification efficace.

b) Virtual Concept Drift : Ici, la distribution des données dans les classes a changé, mais la frontière de décision n'a pas été affectée. Bien que les données semblent avoir changé, la relation sous-jacente n'a pas évolué. Cela peut entraîner des erreurs si le modèle ne prend pas en compte ce changement.

c) Real Concept Drift : Dans ce graphe, nous observons que la distribution des données a changé de manière significative, nécessitant une nouvelle frontière de décision. Ce type de dérive est problématique car les modèles précédemment entraînés peuvent devenir obsolètes. Un nouvel apprentissage est nécessaire pour s'adapter à ces nouveaux schémas.

d) Class Imbalance : La figure montre une situation où une classe (ici, les cercles) est beaucoup plus représentée que l'autre (les triangles). Un déséquilibre de classe peut conduire à des modèles biaisés qui privilégient la classe majoritaire, rendant la classification de la classe minoritaire moins précise.

e) Novel Class : Il s'agit du scénario où une nouvelle classe (représentée par un point vert) apparaît dans la distribution. Cette apparition de nouvelles classes peut poser des défis pour les modèles existants, qui n'ont pas été formés pour reconnaître ces nouvelles catégories. Cela nécessite souvent une réévaluation ou un réentraînement des modèles.

f) Fusion of Existing Class : Cette figure montre comment deux classes existantes (cercles et triangles) peuvent fusionner en une seule classe. Une telle fusion peut compliquer la classification,

car la frontière de décision doit être ajustée pour refléter cette nouvelle réalité. Cela peut également nécessiter un réentraînement du modèle.

Ces scénarios illustrent divers défis que les modèles d'apprentissage automatique peuvent rencontrer dans des environnements dynamiques. La gestion du concept drift est essentielle pour maintenir la performance et la précision des systèmes de classification.

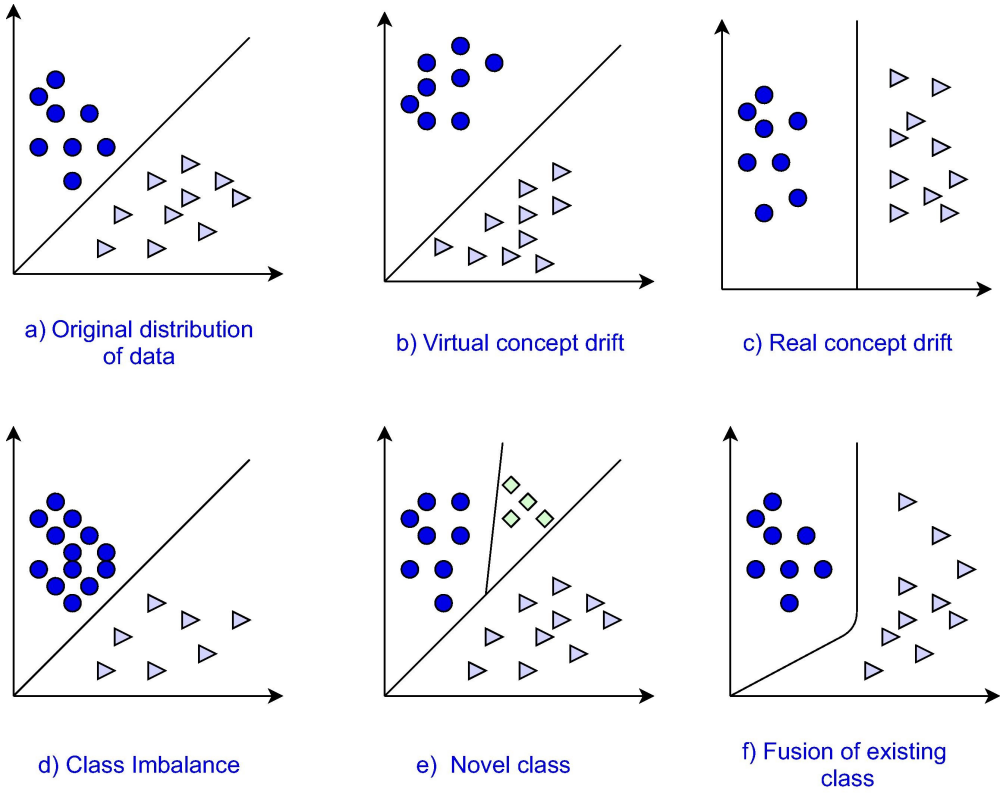


FIGURE 2.1 – Les types de dérives conceptuelles

2.1.2 Détection d'anomalies

La détection d'anomalies, est une branche importante de l'exploration de données qui vise à identifier les modèles de données qui s'écartent des comportements attendus au sein des données[15]. Le choix de la méthode de détection est fonction du type d'anomalies à détecter. Ainsi, on distingue :

— **Les anomalies ponctuelles** : Il s'agit des observations qui s'ecartent de manière significative du comportement des autres observations. Dans la vie courante, ce type d'anomalie peut correspondre à une transaction d'un montant très élevé contrairement aux transactions financières habituelles.

— **Les anomalies collectives** : Elles font reférence à une suite d'observations qui s'ecartent de facon importante de la distribution normale des données. Il faut noter que prise individuellement ces données peuvent paraitre normales bien que collectivement elles constituent une anomalie. Dans le E-commerce par exemple, un utilisateur achète un article coûteux, puis plusieurs articles de faible valeur juste après. Ce comportement pourrait être normal, mais une séquence répétée de tels achats peut indiquer un schéma de fraude, où les petits achats sont effectués pour vérifier la validité de la carte après un achat important.

— **Les anomalies contextuelles :** En raison de la propriété du concept drift liées aux flux de données, certains points de données considerés comme anormaux dans un contexte peuvent être considérés comme normaux dans un autre contexte. En santé, un patient à une fréquence cardiaque normale entre 60 et 80 battements par minute (bpm) au repos. Pendant un exercice léger, une augmentation de la fréquence cardiaque à 120 bpm est normale. Cependant, si la même fréquence cardiaque de 120 bpm est observée alors que le patient est allongé et au repos la nuit, cela pourrait être une anomalie contextuelle, indiquant un problème cardiaque tel qu'une arythmie nocturne.

Ces différents types d'anomalies peuvent être visualisés sur la figure 2.2 [6] :

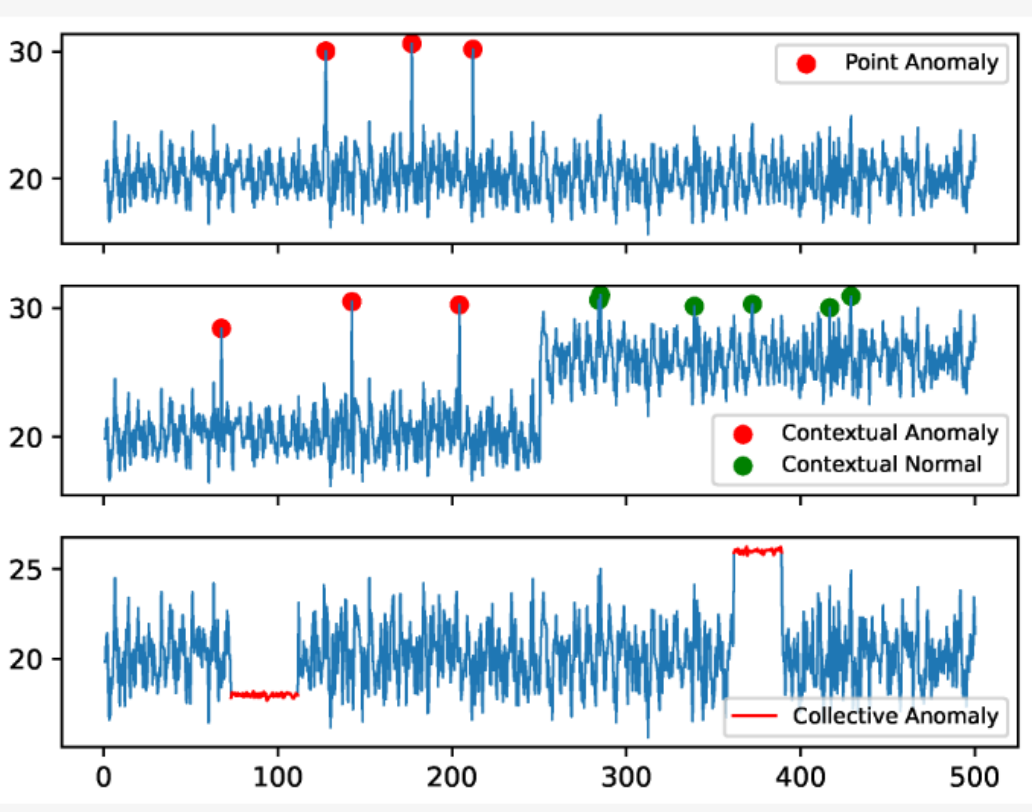


FIGURE 2.2 – Les types d'anomalies

2.1.3 Autoencodeurs

Un autoencodeur est un type d'algorithme dont le but principal est d'apprendre une représentation « informative » des données qui peut être utilisée pour différentes applications tout en apprenant à reconstruire suffisamment bien un ensemble d'observations d'entrée [5].

Le schéma 2.3 décrit la structure générale d'un autoencodeur [2] ; elle est constituée principalement de :

— **Un encodeur** : Il s'agit généralement d'un réseau de neurones qui prend les données en entrée et les transforme en une représentation latente (ou code) de plus petite dimension.

— **La couche de Bottleneck** : C'est la couche centrale de l'autoencodeur, qui contient la représentation latente des données d'entrée. Cette représentation de plus faible dimension

2. https://www.linkedin.com/pulse/coupled-attention-revolutionizing-time-series-anomaly-prajapati/ (23-08-2024)

peut être utilisée pour d'autres tâches comme la visualisation, la classification, etc.

— **Un decodeur** : C'est également un réseau de neurones dans la plupart des cas qui prend cette représentation latente et tente de reconstruire les données d'origine.

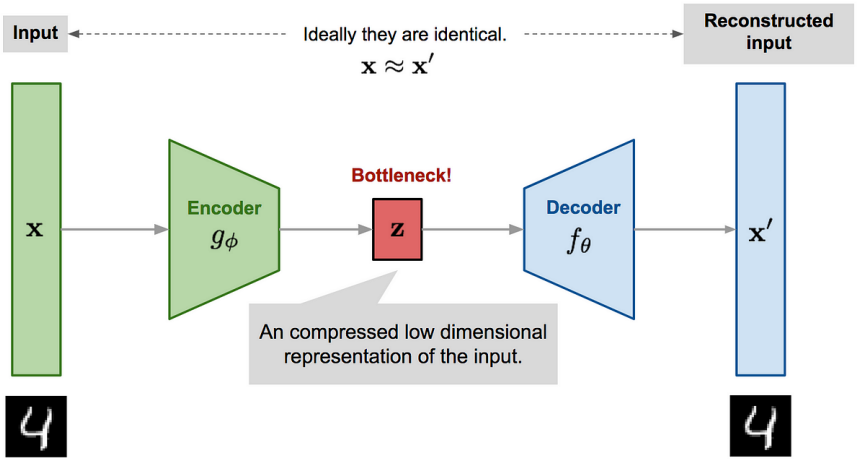


FIGURE 2.3 – Structure d'un autoencodeur

De façon formelle, l'encodeur peut être écrit comme une fonction g qui dépend de certains paramètres :

$$z_i = g(x_i) \quad (1)$$

où $z_i \in \mathbb{R}^Q$ (la représentation des caractéristiques latentes) est la sortie du bloc encodeur dans la Figure 1 lorsque nous l'évaluons sur l'entrée x. Notez que nous avons $g : \mathbb{R}^N \to \mathbb{R}^Q$. Le décodeur (et la sortie du réseau que nous indiquerons par x_i') peut alors être écrit comme une seconde fonction générique f des caractéristiques latentes :

$$x_i' = f(z_i) = f(g(x_i)) \quad (2)$$

où $x_i' \in \mathbb{R}^n$. L'entraînement d'un autoencodeur signifie simplement trouver les fonctions $g(\cdot)$ et $f(\cdot)$ qui satisfont :

$$\arg\min_{f,g} \langle \Delta(x_i, f(g(x_i))) \rangle \quad (3)$$

où Δ indique une mesure de la différence entre l'entrée et la sortie de l'autoencodeur : C'est l'erreur de reconstruction. Ceci peut donc s'apparenter à un problème de regréssion linéaire. Cette erreur de reconstruction peut se calculer de diverses façon mais l'erreur la plus utilisée l'erreur quadratique moyenne (Mean Square Error, MSE). Elle est indépendante de la fonction d'activation de la couche de sortie ou de la manière dont les données d'entrée sont normalisées et aussi, il est facile de montrer que le minimum de L_{MSE} est atteint pour $x_i' = x_i$.

$$L_{\text{MSE}} = \text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \|x_i - x_i'\|^2 \quad (4)$$

Le symbole $\|\cdot\|$ indique la norme d'un vecteur, et N est le nombre d'observations dans l'ensemble de données d'entraînement.

Bien que l'autoencodeur soit entraîné à reproduire un vecteur d'entrée $x \in \mathbb{R}^D$ en sortie, la couche de bottleneck force le réseau à apprendre une représentation compacte des données d'entrée.

Mais il se pose souvent le problème de conservation de la sémantique des données dans l'espace latent. C'est à dire les relations entre les différents concepts peuvent ne pas être correctement préservées dans la représentation latente. Par exemple, l'espace latent peut representer les cercles plus proches des triangles que des carrés ce qui est illogique. Les autoencodeurs variationnels ou VAE viennent donc régler ce problème.

Un autoencodeur variationnel ou VAE peut être considéré comme une version probabiliste d'un AE, où un vecteur latent aléatoire z et des variables d'observation/vecteur x, $p_\theta(z \mid x)$ est connu sous le nom de distribution postérieure ; $p(z)$ est la distribution à priori, $p_\theta(x)$ est la preuve du modèle ou la probabilité marginale, et $p_\theta(x \mid z)$ est la vraisemblance. Les mises à jour sur $p(z)$ sont faites en utilisant la règle bayésienne.

La fonction de perte totale du VAE, qui combine ces deux termes, est donnée par :

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[-\log p_\theta(x \mid z)] + D_{\mathrm{KL}}(q_\phi(z \mid x) \parallel p(z)) \quad (5)$$

où :

$\mathbb{E}_{q_\phi(z|x)}[-\log p_\theta(x \mid z)]$ est la perte de reconstruction, et $D_{\mathrm{KL}}(q_\phi(z \mid x) \parallel p(z))$ la divergence de Kullback-Leibler qui mesure la différence entre la distribution a posteriori approximée $q_\phi(z \mid x)$ et la distribution a priori $p(z)$.

En minimisant cette fonction de perte, l'objectif est d'obtenir un compromis optimal entre une bonne reconstruction des données et une régularisation adéquate de l'espace latent.

2.1.4 Explicabilité des modèles

L'intelligence artificielle tend à s'imposer de plus en plus dans des domaines sensibles tels que la santé, la justice et la sécurité. Ainsi, il ne suffit plus de produire de bonnes prédictions mais également d'expliquer comment le modèle est arrivé à la prédiction (le pourquoi). De plus en plus, on se tourne vers l' apprentissage automatique interprétable ou explicable car les utilisateurs et les décideurs ont besoin de comprendre comment un modèle arrive à ses conclusions pour faire confiance à ses recommandations.. Selon Molnar, l'apprentissage automatique explicable fait référence aux méthodes et aux modèles qui rendent le comportement et les prédictions des systèmes d'apprentissage automatique compréhensibles pour les humains [17].

Les modèles d'apprentissage automatique explicables permettent de verifier que les modèles respectent les propriétés suivantes :

- **Équité** : garantir que les décisions prises sont impartiales et non discriminatoires envers certains groupes (sexe, race).
- **Confidentialité** : garantir que les informations sensibles contenues dans les données sont protégées.
- **Fiabilité ou robustesse** : garantir que de petits changements dans les données d'entrée n'entraînent pas de changements importants dans la prédiction.
- **Causalité** : Vérifier que les relations identifiées par le modèle sont causales plutôt que simplement corrélatives.

Les méthodes d'interprétabilité de l'apprentissage automatique peuvent être classées selon divers critères [17] :

— **Modèles Interprétables Intrinsèques** : Ce sont des modèles qui sont interprétables par nature, sans nécessiter de méthodes supplémentaires pour expliquer leurs décisions. **Exemple** : Les arbres de décision, les régressions linéaires .

— **Méthodes Post-hoc** : Ces méthodes sont appliquées après l'entraînement du modèle, en particulier pour les modèles complexes comme les réseaux de neurones profonds ou les forêts aléatoires, pour expliquer leur comportement. **Exemple** : SHAP [16] et LIME [18].

2.2 État de l'art

De plus en plus, les chercheurs se tournent vers les techniques basées sur les autoencodeurs pour la détection d'anomalies en raison de leur robustesse permettant de gérer les variations et le bruit dans les données, ce qui les rend efficaces dans des environnements réels où les données peuvent être imparfaites. Cependant les résultats des autoencodeurs peuvent être difficiles à interpréter. Contrairement à certains modèles plus simples, il peut être compliqué de comprendre pourquoi certaines anomalies ont été détectées. Ainsi, des méthodes d'explicabilité sont de plus en plus explorées afin de pouvoir expliquer les anomalies détectées par les autoencodeurs. Dans cette section, nous analysons les travaux sur la détection d'anomalies à l'aide des autoencodeurs ainsi que les méthodes d'explicabilité afin mieux comprendre les défis rencontrés dans ces domaines .

2.2.1 Méthode de détection d'anomalies à l'aide d' un autoencodeur

En 2015, Jinwon An et al.[3] proposent une approche probabiliste pour la détection d'anomalies à partir d'un autoencodeur variationnel. L'objectif est de tirer parti de la probabilité de reconstruction pour mieux détecter les anomalies. En effet, lorsque les données sont hétérogenes ou que les anomalies sont rares et difficiles à distinguer des données normales, les méthodes traditionnelles ont du mal à les détecter à cause de leurs limites dans la capture des variabilités dans les données. Les auteurs proposent donc d'utiliser le calcul d'une probabilité de reconstruction comme un score d'anomalie. Sachant que la probabilité de reconstruction mesure la probabilité que les données d'origine soient générées par la distribution du décodeur. La méthode proposée contrairement aux autoencodeurs classiques permet de modeliser la variabilité des flux de données ce qui améliore la detection d'anomalies. Cependant, le modèle a du mal à reconnaitre les anomalies lorsqu'elles sont faiblement representées mais aussi et surtout les prédictions ne sont pas interprétables .

Aussi, Longyuan Li et al.[14] propose donc en 2020 le Smoothness-Inducing Sequential Variational Autoencoder (SISVAE) qui est une extension du Variational Autoencoder (VAE) traditionnel qui intègre une modélisation de données séquentielles tout en favorisant la fluidité de la représentation de l'espace latent. Il s'agit d'une méthode de détection d'anomalies au niveau ponctuel sur des séries temporelles multidimensionnelles. qui permet de détecter efficacement les anomalies dans les séries temporelles tout en prenant en compte de la structure temporelle et de la représentation "normale" des données. Ici, plutôt que de minimiser aveuglément l'erreur de reconstruction entre les donnees d'entrées et les données de sortie, l'entraînement vise à apprendre le modèle sur des donnees normales à partir de l'espace latent .

Le Smoothness-Inducing Sequential Variational Auto-Encoder en plus d'être une méthode non supervisée a pour avantage d'etre robuste dans la détection d'anomalies car modelise les relations séquentielles dans les données tout en réduisant l'impact du bruit non stationnaire grâce à la contrainte de lissage. Cependant les performances du modèle dependent des hyperparamètres notamment du paramètre de régularisation. Aussi, comme pour tous les reseaux de neurones, les resultats du SISVAE sont difficiles à interpréter.

2.2.2 Méthode explicable de détection d'anomalies

Dans un contexte ou la plupart des travaux sur l'explicabilité dans la détection d'anomalies comportent deux modèles à savoir : un modèle pour la détection d'anomalies et un autre pour l'explicabilité des anomalies détectées . Penny Chong et al.[7] dans "Toward Scalable and Unified Example-based Explanation and Outlier Detection" se proposent d'unifier l'explicabilité des prédictions et la détection des outliers dans un même modèle. En s'appuyant sur le fait que l'explication par exemples, telle que l'utilisation d'exemples prototypiques, est souvent préférée aux explications par superposition pour les utilisateurs finaux non techniques . Ils proposent donc un "réseau étudiant" basé sur des prototypes capable d'éffectuer simultanément la tâche d'explication et la tâche de détection d'anomalies. Pour ce faire, un modèle de reférence baseline appelé ici modèle enseignant est entrainé pour la classification d'images ; ensuite par l'apprentissage par transfert les prédictions du modèle enseignant sont utilisées pour entraîner le modèle étudiant afin d'obtenir de meilleures performances.

Afin de maintenir la pertinence des prototypes au fur et à mesure, ils sont mis à jour par un algorithme itératif. Ainsi, plus une donnée est eloignée de son prototype plus elle est considerée comme un outlier. Les auteurs de cet article ont proposé une approche innovante qui permet de fournir des explications claires et intuitives pour les utilisateurs. Cependant la limite réside principalement dans la capacité du modèle à trouver le conpromis entre la détection et l'explication.

En 2021, Cyriana Roelofs et al. [19]proposent un algorithme d'optimisation dont le but est de trouver quelles caractéristiques d'entrée ont contribué à une anomalie détectée par le modèle AE. En s'appuyant sur l'hypothèse selon laquelle une erreur dans une caractéristique d'entrée d'un auto-encodeur se propage à travers le réseau et affecte toutes les sorties du réseau, les auteurs proposent la méthode ARCANA ou Anomaly Root Causes ANAlysis.

L'algorithme recherche des caractéristiques d'entrée qui ont le plus contribuées à l'anomalie détectée par un autoencodeur. Cela se fait en minimisant la fonction de perte :

$$\text{Loss} = (1-\alpha)\frac{1}{2}\|x_{\text{corr}} - g(h(x_{\text{corr}}))\|_2 + \alpha\|x_{\text{corr}} - x\|_1, \tag{2.1}$$

où x est un échantillon de données (vecteur de caractéristiques d'entrée à un moment donné), $\alpha \in (0, 1]$ est un hyperparamètre, $x_{\text{corr}} = x + x_{\text{bias}}$ et x_{bias} est le vecteur de biais d'ARCANA.

La minimisation de cette fonction permet de trouver le vecteur biais qui représente la déviation de chaque caractéristique causant l'anomalie et, par conséquent, leurs contributions à l'anomalie. Il peut également être interprété comme le RE 'corrigé'. L'optimisation de la norme L_2 trouve une entrée 'corrigée' x_{corr} c'est à dire une entrée qui peut être dite 'normale'. La norme L_1 assure que le nombre de caractéristiques non nulles du vecteur de biais est aussi bas que possible. Pour optimiser la fonction de perte, la rétropropagation est utilisée pour calculer les gradients de la fonction de perte par rapport à x_{bias}. Pour mettre à jour x_{bias} et pour l'optimisation, l'algorithme Adam [13] est utilisé. Le vecteur x_{bias} est calculé pour chaque échantillon, mais peut etre utilisé pour expliquer une tranche d'anomalies en moyennant les valeurs absolues du vecteur de biais sur la fenêtre temporelle d'anomalie et en normalisant le résultat, de sorte que la somme des importances soit égale à 1.

La méthode ARCANA ou Anomaly Root Causes ANAlysis produit des interprétations compréhensibles par des humains en les permettant d'identifier les causes profondes de l'anomalie. Cependant, bien qu'étant appliquée dans un contexte de flux de données impliquant le fait que la distribution des données peut évoluer au fil du temps, la mise à jour régulière du modèle d'auto-encodeur n'est pas abordée dans la description du modèle.

2.2.3 Tableau récapitulatif

Le tableau 3.1 résume les différentes travaux de notre revue de la littérature ayant pour principal objectif la détection d'anomalies et/ou l'explicabilité. Il présente donc pou chacun d'eux les auteurs, les approches, les algorithme utilisés, les avantages et les limites.

Auteurs	Principe	Algorithme	Avantages	Limites
Jinwon An et al. [3] (2015)	Utiliser la probabilité de reconstruction pour la détection d'anomalies	**Variational Autoencoder (VAE)**	Améliore la détection d'anomalies (en prenant compte de la variabilité dans les données)	Les prédictions ne sont pas interprétables
Longyuan Li et al.[14] (2020)	Récuperer la distribution normale des données en présence d'anomalies	**Smoothness-Inducing Sequential Variational Autoencoder (SISVAE)**	Robuste dans la détection d'anomalies	Les prédictions ne sont pas interprétables
Penny Chong et al. [7] (2021)	Utiliser les prototypes pour effectuer simultanément la tâche de détection et d'explication	**Prototypes**	Fournit des explications claires et intuitives pour les utilisateurs.	Difficile de trouver le compromis entre la détection et l'explicabilité
Cyriana Roelofs et al. [19] (2021)	Minimiser une fonction d'erreur afin de trouver un vecteur de déviations des caractéristiques	**Anomaly Root Causes ANAlysis(ARCANA)**	Fournit une explication basée sur les caractéristiques	Le modèle ne prend pas en compte le changement de la distribution au fil du temps

Table 2.1 – Tableau récapitulatif des algorithmes détection d'anomalies

2.3 Bilan du chapitre

De ce chapitre, nous pouvons constater que la principale limite des méthodes de détection d'anomalies basées sur les réseaux de neurones est la faible explicabilité. Penny Chong et al. [7] propose une approche d'explicabilité basée sur les prototypes . Aussi, Cyriana Roelofs et al. [19] propose la methode d'explicabilité ARCANA cependant la méthode ne prend pas en compte le concept drift. On se pose donc la question de recherche suivante : Comment prendre en compte le changement de la distribution des données dans le processus de détection et d'explication des anomalies ?

CHAPITRE 3

MÉTHODE PROPOSÉE

Ce chapitre présente de manière détaillée l'approche proposée pour la détection d'anomalies à l'aide d'un autoencodeur et son explicabilité. Notre approche se presente en deux étapes à savoir la detection d'anomalies à partir de l'algorithme "Update Based Prototypes" qui s'appuie sur un autoencodeur et l'explicabilité à l'aide l'algorithme "ARCANA-Prototype" qui est une extension de l'algorithme ARCANA.

3.1 Architecture générale du cadre de travail

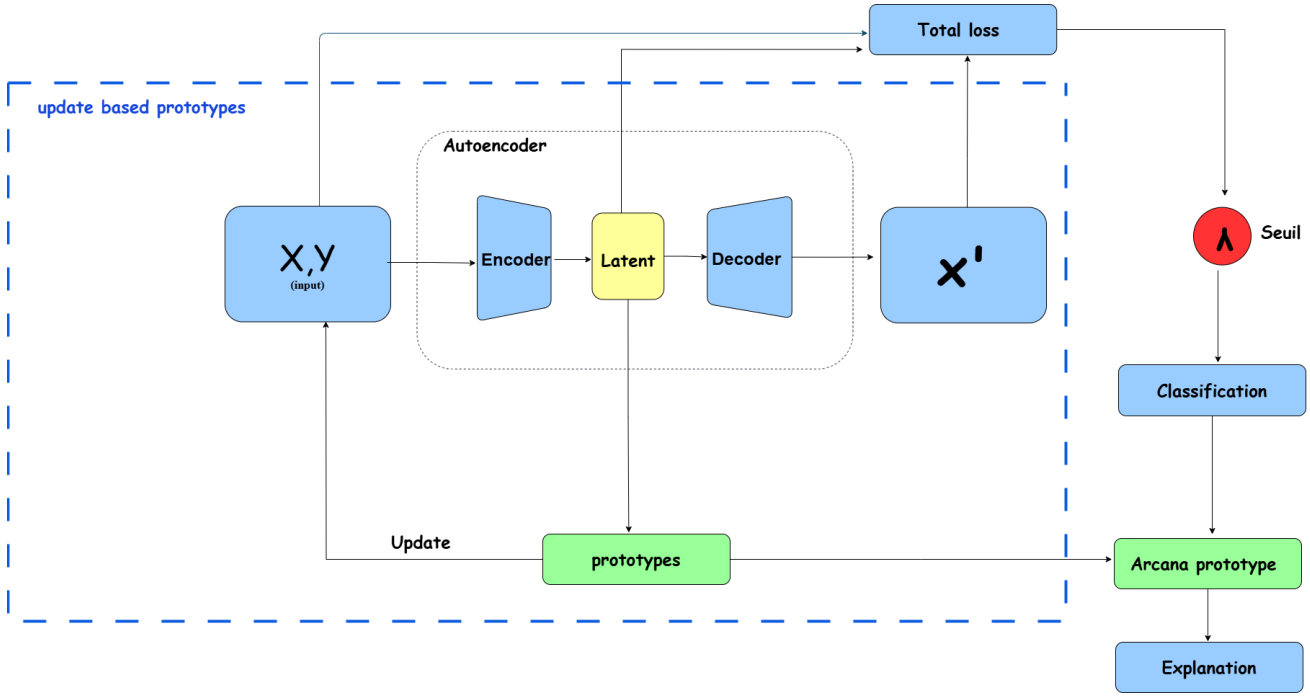


FIGURE 3.1 – Architecture générale

L'architecture principale de la méthode proposée dans ce mémoire se constitue de deux principales étapes et est representée sur la figure 3.1 :

— La détection d'anomalies à l'aide de notre modèle "Update Based Prototypes" : Ici, les données sont passées à l'autoencodeur avec les étiquettes pendant l'entrainement ; les étiquettes sont utilisées afin de séparer l'espace latent en fonction des classes et de pouvoir obtenir les prototypes de chaque classe à partir de l'espace latent. Pendant la prediction, l'erreur est calculée non seulement en s'appuyant sur la différence entre les données d'entrée et les données reconstruites ; mais aussi sur la classification de l'espace latent. Ensuite cette erreur est comparée à un seuil et sont considérés comme anomalies les données pour lesquelles l'erreur est supérieure au seuil. Lorsque une donnée est mal classée par le modèle, on mesure la difference entre la donnée et le prototype de la classe associée ; si la différence est grande, ceci implique un changement dans la distribution des données donc le modèle est donc réentrainer afin de s'adapter à la nouvelle distribution des données. Tout en mettant à jour les prototypes.

— Après la phase de détection, vient la phase des explications ; les anomalies détectées seront passées au modèle ARCANA-Prototypes afin d'identifier les caractéristiques qui ont le plus contribuées à l'anomalie. Cette approche utilise les prototypes extraits pendant l'entrainement.

3.2 Algorithme Update Based Prototypes

3.2.1 Autoencodeur

Pour la détection d'anomalies, notre modèle s'appuie sur un autoencodeur. L'idée principale est d'entrainer l'autoencodeur afin de le rendre capable de reconstruire au mieux les données de facon à utiliser l'erreur de reconstruction pour détecter les anomalies.

Préparation des données

Étant dans un contexte de données etiquetées, il a donc fallu séparer les caractéristiques des etiquettes. Aussi, afin d'éviter que certaines variables n'aient une influence disproportionnée sur le modèle à cause de leurs échelles nous avons normalisé les données. Nous avons opté pour la standardisation en raison de sa capacité à traiter de manière équitable les différentes caractéristiques indépendamment de leur échelle.

Définition de l'architecture de l'autoencodeur

Dans le cadre de ce travail, la détection s'appuie sur un autoencodeur variationnel en raison du terme de régularisation (divergence de Kullback-Leibler, KL) qui permet à la distribution des variables latentes d'être proche d'une distribution normale standard ; permettant ainsi d'obtenir une organisation plus sèmantiquement correcte de l'espace latent.

L'autoencodeur prend les données étiquettées et les représente dans l'espace latent en fonction de leur classes. Pour ce qui est de l'espace latent une fonction sampling réalise un tirage aléatoire de l'espace latent en utilisant les paramètres produits par l'encodeur.

Le fait d'échantillonner aléatoirement dans l'espace latent permet d'obtenir des représentations plus régulières et robustes. Cela permet d'obtenir un vecteur latent z qui suit la probabilité gaussienne de l'espace latent. On a donc z comme suit :

$$\mathbf{z} = \mathbf{z}_{\text{mean}} + \exp(0.5 * \mathbf{z}_{\text{log\_var}}) * \epsilon, \text{où} \epsilon \sim \mathcal{N}(0, 1)$$

Le décodeur doit reconstruire les données d'entrée à partir de l'espace latent .

Entrainement du modèle

L'entrainement de l'encodeur variationnel visera à apprendre à reconstruire les données d'entrée tout en apprenant une représentation compacte et regulière dans l'espace latent.

Pour cela, apprendre à bien reconstruire les données d'entrée reviendra à minimiser l'erreur de reconstruction de l'autoencodeur. Pour l'erreur de reconstruction le choix s'est porté sur l'erreur quadratique moyenne ou Mean Square Error(MSE). La MSE en raison de ses propriétés statisques, de sa facilité d'optimisation et de son interprétation intuitive. De facon formelle, apprendre à bien reconstruire reviendra à minimiser la fonction d'erreur suivante :

$$\text{MSE}(\mathbf{x}_i, \mathbf{x'}_i) = \frac{1}{m} \sum_{i=1}^{m} (x_i - x'_i)^2 \tag{3.1}$$

où :

— m est le nombre de dimensions,

— x_i est la valeur réelle de la i-ème caractéristique de x,

— x'_i est la valeur de la i-ème caractéristique de x reconstruit.

Apprendre une représentation compacte et regulière de l'espace latent consiste à minimiser la divergence de Kullback Leibler. La divergence de Kullback-Leibler (KL) est une mesure de la différence entre deux distributions de probabilité. Dans notre cas on calculera la divergence de KL entre la distribution latente apprise et une distribution de référence (généralement une gaussienne standard) qui agit comme un terme de régularisation. Ce terme encourage le modèle à apprendre des représentations latentes qui suivent la distribution de référence, rendant l'espace latent plus structuré et régulier. Dans notre cas elle est définie comme :

$$\text{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\right) = -\frac{1}{2} \sum_{i=1}^{d} \left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right) \tag{3.2}$$

où :

— $q_\phi(\mathbf{z}|\mathbf{x})$ est la distribution latente apprise par l'encodeur,

— $p_\theta(\mathbf{z})$ est la distribution latente prior (souvent une normale standard),

— μ_i est la moyenne de la distribution latente pour la i-ème dimension,

— σ_i^2 est la variance de la distribution latente pour la i-ème dimension,

— d est la dimension de l'espace latent.

L'erreur totale est la somme ponderée des deux erreurs en donnant plus d'importance à l'erreur de reconstruction :

$$\mathcal{L}_{\text{VAE}} = \frac{1}{n} \sum_{j=1}^{n} \left[\alpha \text{MSE}(\mathbf{x}_j, \mathbf{x'}_j) + \beta \times \text{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}_j) \parallel p(\mathbf{z})\right)\right] \tag{3.3}$$

où :

— $\mathbf{x}_j$ est l'entrée réelle,
— $\mathbf{x'}_j$ est la reconstruction de l'entrée,
— $\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\right)$ est la divergence de Kullback Leibler,
— $\mathrm{MSE}(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ est l'erreur de reconstruction'
— α, β sont des coefficients d'importance des deux erreurs

3.2.2 Prototypes

Dans le contexte de l'apprentissage automatique et de la détection d'anomalies, un prototype est souvent utilisé comme une référence pour comparer de nouvelles instances de données afin de déterminer si elles appartiennent à la classe en question ou si elles sont anormales. L'autoencodeur est entrainé de façon supervisée et les étiquettes sont utilisées afin de séparer l'espace latent en fonction des classes grâce à la divergence de Kullback Liebler . Ainsi, à partir des étiquettes, il est possible d'extraire les prototypes des données normales et des anomalies car l'espace latent aété entrainer afin de differencier les anomalies des données normales . Il est donc question pour nous d'utiliser un algorithme de clustering qui réussit à capter la distribution des classes dans l'espace latent . Le centre de chaque cluster obtenu à partir de cet algorithme de clustering représentera donc le prototype associé à la classe concernée. Ainsi, le prototype d'une classe est calculé par la moyenne des points du clusters et donné par la formule :

$$\mathbf{P}_c = \frac{1}{n} \sum_{i \in c}^{m} \mathbf{x}_i \tag{3.4}$$

où :
— n est le nombre d'observations,
— c est la classe associée'
— $\mathbf{x}_i$ le vecteur de la donnée appartenant à la classe

Apres l'exécution de cet algorithme on obtient des prototypes qui correspondent à la representation moyenne des données d'entrainement.

3.2.3 Détermination du seuil

Le choix du seuil est l'étape importante dans la détection d'anomalies car un mauvais seuil peut donner de mauvais resultats.

Nous avons donc utilisé plusieurs techniques de seuillage et avons choisi le seuil qui donne les meilleurs resultats. Les critères d'evaluation de notre seuil seront :

— **F1-score :** Le F1-score est la moyenne harmonique de la précision (precision) et du rappel (recall). Il est particulièrement adapté dans des situations où il y a un déséquilibre entre les classes (c'est-à-dire beaucoup plus de données normales que d'anomalies), ce qui est souvent le cas dans la détection d'anomalies.
— **Taux de fausses alarmes :** Le taux de fausses alarmes (False Alarm Rate) mesure la proportion d'instances normales qui ont été incorrectement classées comme anomalies (faux positifs). Il est utilisé pour évaluer la capacité d'un système à minimiser les alertes inutiles.
— **Taux d'alarmes manquées :** Le taux d'alarmes manquées (aussi appelé taux de faux négatifs) est une métrique plus générale qui prend en compte à la fois les faux positifs et les faux négatifs. Il est souvent utilisé pour évaluer la performance globale du modèle de détection d'anomalies.

3.2.4 Mise à jour de l'autoencodeur

Comme nous l'avons mentionné plus haut notre principale contribution était d'intégrer la prise en compte du concept drift dans notre modèle. Pour cela, nous proposons d'utiliser l'apprentissage supervisé en s'appuyant sur les prototypes comme éléments de base de notre approche.

Nous supposons que le prototype d'une classe représente l'élément qui représente le mieux les données d'une classe . Ainsi, lorsque les données d'une classe commencent à être distante du prototype de la classe associée on considéra qu'il y'a eu un changement dans la distribution des données et on mettra à jour notre modèle sur la base de ce changement. Pour ce faire, on suivra les étapes suivantes :

— Pour chaque fenêtre de données on les reconstruit à l'aide de l'autoencodeur tout en s'assurant de bien representer l'espace latent.

— On calcule l'erreur totale de l'autoencodeur sur chaque donnée de la fenêtre

— Les anomalies sont détectées en comparant l'erreur de reconstruction de chaque donnée au seuil ; Si l'erreur est supérieure au seuil, la donnée est considérée comme une anomalie

— Aussi, pendant l'entraînement, les prototypes de chaque classe sont appris à partir de l'espace latent.

— Lorsque une donnée est mal classée, nous calculons la distance entre la donnée et le prototype de la classe associée.

— Si la donnée est très eloignée de son prototype, cela veut dire que la distribution des données a changée et nous faisons une mise à jour du prototype associé

— Ensuite, on met à jour le modèle *vae* en le réentraînant sur la fenêtre de données et les étiquettes et on obtient de nouveaux prototypes

On obtient l'algorithme suivant 1 :

Algorithm 1 Update Based Prototypes

Require: $X \in \mathbb{R}^{n \times d}$: Données d'entrée, $y \in \mathbb{R}^n$: Étiquettes des données, τ : Seuil de distance
Ensure: Modèle VAE mis à jour, Prototypes $\{p_c \in \mathbb{R}^2 | c = 1, \ldots, C\}$ mis à jour, où C est le nombre de classes

for chaque donnée i dans X **do**
  Prédire la classe de la donnée X avec le modèle VAE : $\hat{y}_i = \text{vae.predict}(x_i)$
  if $\hat{y}_i \neq y_i$ **then**
    Calculer la distance d_i entre la donnée x_i et le prototype de sa classe y_i :

$$d_i = \|x_i - p_{y_i}\|_2$$

    if $d_i > \tau$ **then**
      Mettre à jour le modèle VAE en l'entraînant sur la donnée X_i avec l'étiquette y_i
      Mettre à jour le prototype p_{y_i} de la classe y_i en recalculant la moyenne des données de cette classe dans l'espace à 2 dimensions :

$$p_{y_i} = \frac{1}{n_i} \sum_{j:y_j = y_i} x_i$$

      où n_i est le nombre de données de la classe y_i
    end if
  end if
end for
return Modèle VAE mis à jour et prototypes mis à jour

3.3 ARCANA Prototypes

Concernant l'explicabilité, nous proposons d'utiliser le fait que les prototypes soient une représentation typique ou moyenne des éléments appartenant d'une classe. et nous construisons une extension de la méthode ARCANA .En remplacant le x corrigé par le prototype de la classe normale. Car dans l'équation de perte de ARCANA, le x corrigé représente le x supposé normal cependant le calcul du x corrigé se fait par une méthode d'optimisation ; on a donc pas la garantie de tomber sur le minmum global de la fonction d'erreur. Pour pallier à ce problème, nous proposons de remplacer le x corrigé par le prototype des données normales qui est l'élément qui represente au mieux les donées normales. Ceci nous permet d' éviter les risques liés aux minima locaux. Nous proposons donc la formule suivante :

$$\text{Loss} = (1 - \alpha)\frac{1}{2}\|x_{\text{proto}} - g(h(x_{\text{corr}}))\|_2 + \alpha\|x_{\text{corr}} - x\|_1, \tag{3.5}$$

où

— Le premier terme Ce terme mesure la différence entre le prototype et la sortie du modèle pour la donnée corrigée

— Ce terme mesure la différence entre l'entrée corrigée et l'entrée d'origine x, afin d'encourager des changements minimaux dans les données.

— x est un échantillon de données (vecteur de caractéristiques d'entrée à un moment donné),

— $\alpha \in (0, 1]$ est un hyperparamètre qui permet de contrôler l'importance de chaque terme,

— $x_{\text{corr}} = x + x_{\text{bias}}$,

— x_{proto} est le prototype des données normales extrait de l'espace latent pendant l'entrainement

— x_{bias} est le vecteur de biais d'ARCANA.

On obtiendra donc un vecteur biais qui corrige les anomalies parrapport au prototype des donnees normales.

Les étapes de notre méthode peuvent etre présentées à partir de l'algorithme 2 dont les points principaux sont :

— Les données sont traités par fenêtres de taille N. Cela permet de ne pas charger tout l'ensemble de données en mémoire à la fois, ce qui peut être nécessaire si les données sont volumineuses.

— Pour chaque élément x dans une fenêtre, un vecteur de correction est appliqué : $x_{\text{corr}} = x + x_{\text{bias}}$. Ce biais permet de compenser les différences ou erreurs systématiques dans les données avant de les transmettre à l'autoencodeur.

— L'erreur de reconstruction est calculée entre x_{proto} (le prototype des données normales) et x_{pred}. L'idée ici est que les données normales devraient être reconstruites de manière plus précise par rapport aux prototypes

— On régularise l'écart entre la correction x_{corr} (les données corrigées avec le biais) et les données d'origine x

— Après avoir calculé la fonction de perte, l'algorithme met à jour les paramètres de l'autoencodeur (c'est-à-dire les poids des couches de l'encodeur et du décodeur) via un algorithme de descente de gradient, comme Adam.

— Le vecteur de biais x_{bias} est ajusté en fonction des erreurs détectées dans la fenêtre de données actuelles, afin d'améliorer les reconstructions dans les fenêtres suivantes.

— L'algorithme renvoie la perte minimale $loss_{\text{min}}$ et le vecteur biais x_{bias} associé .

— Le vecteur biais est un vecteur de taille égale aux caractéristiques de départ des données . Ainsi, chaque valeur du vecteur biais représente la contribution de la caractéristique correspondante dans l'anomalie.

— Les explications sont obtenues en lisant vecteur biais qui est un vecteur de taille égale aux caractéristiques de départ des données . Ainsi, chaque valeur du vecteur biais représente la contribution de la caractéristique correspondante dans l'anomalie.

Algorithm 2 ARCANA-Prototype

Require: $X \in \mathbb{R}^{n \times d}$: Données d'entrée, α : Hyperparamètre de régularisation, N : Taille de la fenêtre, x_{proto} : Le prototype des données normales

Ensure: Vecteur de biais $x_{\text{bias\_min}}$ et perte minimale $loss_{\text{min}}$

Initialiser le vecteur de biais $x_{\text{bias}} = 0$
for chaque fenêtre de taille N **do**
    for chaque élément x dans la fenêtre **do**
        Calculer $x_{\text{corr}} = x + x_{\text{bias}}$
        Reconstituer x_{corr} à partir de l'autoencodeur :

$$x_{\text{pred}} = g(f(x_{\text{corr}}))$$

        Calculer la fonction de perte :

$$\text{Loss} = (1 - \alpha) \cdot \frac{1}{2} \|x_{\text{proto}} - x_{\text{pred}}\|_2^2 + \alpha \|x_{\text{corr}} - x\|_1$$

        Mettre à jour les paramètres du modèle via un algorithme de descente de gradient (Adam)
        Mettre à jour le vecteur biais x_{bias}
    end for
end for
return $x_{\text{bias\_min}}$, $loss_{\text{min}}$

3.4 Bilan du chapitre

Ce chapitre offre une vue d'ensemble exhaustive de la méthodologie adoptée dans cette étude. Nous avons présenté l'architecture générale de l'autoencodeur, la technique de détection d'anomalies et la méthode d'explicabilité choisie . Ces informations serviront de fondement pour la compréhension, la mise en œuvre des expériences et l'analyse des résultats présentés dans la suite de ce travail.

CHAPITRE 4

ÉTUDE EXPÉRIMENTALE

Le but de ce chapitre est la description des jeux de données utilisés pour l'expérimentation de notre détection et explicabilité des anomalies, la présentation du protocole expérimental utilisé, puis la présentation des résulats obtenus suivis d'une interprétation de ceux-ci.

4.1 Description du jeu de données

Le jeu de données utilisé pour nos expérimentations se nomme le Skoltech Anomaly Benchmark (SKAB)[12] qui est un ensemble de données réelles de référence d'un système de circulation d'eau de laboratoire (SKAB) (Katser et Kozitsin, 2020). Il représente un système industriel bien décrit avec plusieurs capteurs et des états de fonctionnement et de défaut bien définis caractérisés par des anomalies collectives ou des points de changement, ainsi que des transitions entre ces états conçu pour évaluer les algorithmes de détection des anomalies. SKAB permet de travailler sur deux problèmes principaux : la détection des anomalies ponctuelles et la détection des anomalies collectives. Il contient l'ensemble de données et un ensemble de modules python permettant de l'évaluer.

Pour ce qui est de l'ensemble de données, la version actuelle que nous avons utilisée comporte 34 datasets dont un dataset contenant les données normales et les 33 autres correspondants aux données issues des expérimentations qui créent des anomalies. Chaque dataset des données des expériences contient 10 features dont 8 caractéristiques et 2 étiquettes à savoir : "Anomaly" qui prend 1 lorsqu'il s'agit d'une anomalie et 0 sinon et "Changepoint" qui prend 1 lorsqu'il s'agit d'un point de début d'anomalies collective et 0 sinon. Les données sont de la nature des séries temporelles multidimensionnelles.

Pour nos expérimentations nous avons utilisé les 33 datasets issus des expériences on travaillera donc avec 37401 points de données avec 13067 points d'anomalies soit un total de 34% d'anomalies. Nous avons divisé notre jeu de données en 3 parties à savoir un jeu de données d'entrainement, de validation et de test.

4.2 Protocole expérimental

Nos expériences ont été réalisées sur une machine Intel Intel Core i7-6600U CPU 2.60GHz × 4 avec un processeur 4 cores, et une RAM de 16GB. Le code a été implementé en langage python version 3.8.10 et été executé sur VisualStudio Code .

Nous avons utilisé comme librairies suivantes :

— Pandas : pour le chargement, la transformation, et l'analyse de grands ensembles de données sous forme de DataFrames.

— Tensorflow : pour la définition et l'entraînement de l'architecture de l'autoencodeur.

— Keras : pour la création, l'entraînement, et les tests de l'autoencodeur .

— Seaborn et Matplotlib : pour la création des graphiques, des tracés statistiques et des visualisations de résultats des experimentations.

— SKlearn : pour les tâches d'apprentissage automatique et pour l'évaluation des modèles.

4.3 Mesures d'évaluation

Pour évaluer notre modèle nous nous sommes appuyés sur l' erreur d'entrainement : pour se rassurer que le modèle ait bien appris ; l' erreur de validation : pour se rassurer que le modèle peut s'adapter aux nouvelles données et la visualisation de l'espace latent pour se rassurer que l'espace latent est sémantiquement cohérent.

Pour le choix du seuil, nous avons testé les différents seuils sur le jeu de données de validation et avons evalué la F1-Mesure, le taux de faux negatifs et le taux de faux positifs afin de choisir un seuil qui trouve un compromis entre le taux de de fausses alarmes et le taux d'alarmes manquées.

Pour ce qui est de la détection d'anomalies nous avons evalué notre modèle sur la F1-Mesure, le taux de fausses alarmes et le taux d'alarmes manquées .

Les formules sont :

— **F1-score :** Le F1-score est une métrique qui combine la précision et le rappel en une seule valeur situe entre 0 et 1 , avec 1 correspondant à la meilleure performance possible. L'objectif sera donc de trouver un modèle qui maximise au plus cette mesure. Le F1-score est calculé comme suit :

$$\text{F1-score} = 2 \times \frac{\text{précision} \times \text{rappel}}{\text{précision} + \text{rappel}} \tag{4.1}$$

où :

$$\text{précision} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux positifs}}$$

$$\text{rappel} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux négatifs}}$$

— **Taux de fausses alarmes :** Le taux de fausses alarmes (aussi appelé taux de faux positifs) est la proportion d'observations normales qui ont été détectées à tort comme étant des anomalies. L'objectif est d'avoir un faible taux de fausses alarmes afin de donner confiance au modèle. Il se calcule comme suit :

$$\text{Taux de fausses alarmes} = \frac{\text{faux positifs}}{\text{vrais négatifs} + \text{faux positifs}} \tag{4.2}$$

— **Taux d'alarmes manquées :** Le taux d'alarmes manquées (aussi appelé taux de faux négatifs) est la proportion d'anomalies qui n'ont pas été détectées par le système. L'objectif est d'avoir un faible taux de faux négatifs afin de laisser passer des anomalies moins d'anomalies possible sans détection. Il se calcule comme suit :

$$\text{Taux d'alarmes manquées} = \frac{\text{faux négatifs}}{\text{vrais positifs} + \text{faux négatifs}} \tag{4.3}$$

Concernant l'explicabilité, nous avons mesuré la fidelité de deux façons en mettant une caracteristique à 0 et en observant si l'algorithme allait identifier la caractéristique mise à 0 comme anormale et aussi nous avons fait le chemin inverse en mettant les caractéristiques identifiées comme déviantes à zero afin de se rassurer que la prédiction du modèle change lorsque les caracteristiques anormales sont annulées .

4.4 Résultats et interprétations

4.4.1 Update Based Prototypes : Détection d'anomalies

Pour l'implémentation de notre modèle, nous commencons par entrainer notre modèleafin de le rendre capable de reconnaitre les données normales ; par la suite nous extrayons les prototypes des differentes classes à partir de l'espace latent. Après cela, nous déterminons le meilleur seuil adapté à notre contexte et c'est ce seuil qui est utilisé pour la détection d'anomalies en mettant à jour le modèle lorsque un changement dans la distribution des données est observée.

Entrainement de l'autoencodeur

En s'appuyant sur les différents avantages des autoencodeurs variationnels comme énumérés dans [6], nous avons choisi d'implémenter une approche supervisée basée sur un autoencodeur variationnel pour notre détection d'anomalies.

Pour ce faire, nous avons premièrement construit notre encodeur en définissant le nombre de couches, l'architecture et les fonctions d'activation ; Ensuite, nous avons défini une fonction d'echantillonage de l'espace latent. Il s'en est suivi la construction du décodeur que nous avons choisi symétrique à l'encodeur. La définition de perte aussi a été une étape importante. Par la suite s'en est suivi l 'entrainement du modèle pour un nombre d'époch défini

Dans le cadre de notre modèle nous avons implementé un autoencodeur symétrique c-a-d la structure du décodeur est le miroir symétrique de celle de l'encodeur. L'encodeur est constitué de 4 couches denses réduisant progressivement la dimension des données et d'une couche qui produit respectivement la moyenne et la variance logarithmique de la distribution de probabilité de l'espace latent. La structure de l'autoencodeur qui a donné les meilleurs resultats est consignée dans le tableau suivant 4.1 :

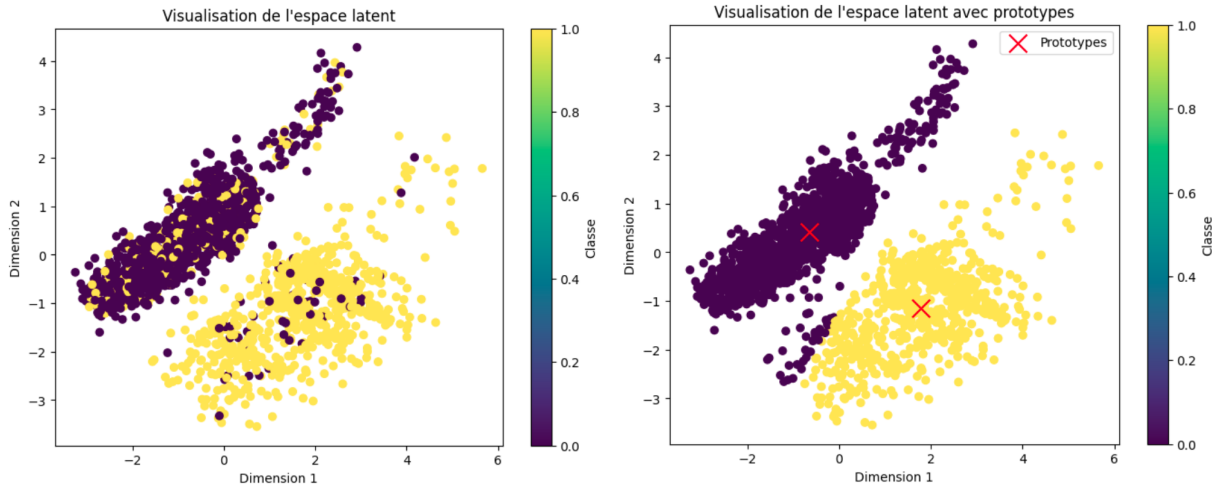
Configuration	Valeur
Nombre d'époques	300
Normalisation	Standard
Taille des lots	100
Optimiseur	Adam
Taux d'apprentissage	0,0005
Dimension des données d'entrées	8*N
Réseau encodeur	5 couches dense de taille de noyau : (128, 64, 32, 16, 8)
Dimension de l'espace latent	4*N
Réseau decodeur	5 couches dense de taille de noyau : (8, 16, 32, 64, 128)
Taille de sortie de l'autoencodeur	8*N

TABLE 4.1 – Configurations de notre autoencodeur

Représentation de l'espace latent et extraction des prototypes

L'entrainement de l'autoencodeur proposé nous a permis d'obtenir la séparation de l' espace latent suivant en fonction des classes comme nous pouvons le voir sur la figure 4.1a. Ici, les points jaunes représentent les anomalies et les points violets representent les points normaux. On constate que notre autoencodeur a réussi à séparer au mieux les deux classes. Ensuite nous avons utiliser l'algorithme Kmeans pour séparerer l'espace latent en cluster et ainsi trouver les centres des clusters qui representent les points qui representent au mieux chaque classe et nous avons obtenu les prototypes de la classe associée comme nous pouvons le voir sur la figure 4.1b.

En utilisant l'Analyse en Composantes principales(ACP) comme outil de réduction de dimension pour projeter ces clusters dans un espace 2D , nous avons obtenu la représentation suivante 4.1 :



(a) Visualisation de l'espace latent (b) Visualisations des prototypes

FIGURE 4.1 – Visualisation en 2D de l'espace latent et des prototypes

Détermination du seuil

Pour la détection de notre seuil, nous avons utilisé les données de test sur lesquels nous avons appliquer plusieurs techniques de seuillage d'apprentissage supervisée [9] :

— **Area Under the Curve (AUC)** : Ici, le seuil optimal peut être déterminé en recherchant le point sur la courbe ROC qui maximise le TPR tout en minimisant le FPR. Cela peut être fait en calculant la distance à partir du point (0,1) sur la courbe ROC.

— **Best-F-score** : Détermine le seuil qui maximise le F1- score d'un ensemble donne . Cette méthode est utilisée pour évaluer la performance maximale qu'un algorithme de détection d'anomalies peut atteindre avec un seuil fixe.

— **Top-k** : Ici, il s'agit de trouver un seuil qui permet de marquer exactement k points temporels comme étant anormaux, où k correspond au nombre réel d'anomalies présentes dans l'ensemble de test.

— **Threshold tuning** : pour évaluer la performance d'un modèle de classification binaire en fonction de différents seuils d'erreur.

— **Méthode de seuillage basée sur un quantile** : est une technique utilisée pour classifier des valeurs (comme des erreurs de reconstruction) en utilisant un seuil défini par un certain quantile d'une distribution statistique.

Pour chaque algorithme nous avons evalué le F1-score 4.1, taux d'alarmes manquées 4.2 et taux d'alarmes manquées 4.3

Ainsi, nous avons obtenu les resultats suivant 4.2 :

Algorithme	**F1-score**	**FAR**	**MAR**	**Meilleur seuil**
Area Under the Curve (AUC)	0.6337	0.20	0.39	0.1127
Best-F-score	0.6425	0.35	0.27	0,1010
Top-k	0.6362	0.24	0.36	0.1057
Quantile(65%)	0.6189	0.17	0.43	0.1208
Threshold tuning	0.6395	0.24	0.36	0.1045

TABLE 4.2 – Détermination du seuil

Ainsi, le meilleur seuil est donné par la methode "le Best-F-score" avec un seuil de 0,1010, un F1-score de 0.6425, un taux de faux positifs(fausses alarmes) de 0.35 et un taux de faux négatifs(manque d'alarmes) : 0.27 car il minimise non seulement le nombre de faux positifs mais aussi le nombre de négatifs ce qui est adapté dans notre contexte car un bon système de détection d'anomalies se doit non seulement de détecter efficacement les anomalies mais aussi de minimiser au plus le nombre de fausses alarmes.

Détection d'anomalies

Pour ce qui est de la détection d'anomalies, nos expérimentations ont été faites sur notre modèle "Update Based Prototypes" en utilisant la valeur de seuil "0,1010" obtenue à partir de la méthode best F1-score car elle a donné les meilleurs resultats sur nos données de validation à savoir : un F1-score de 64.25, un taux de manques d'alarmes de 0.27 et un taux de fausses alarmes de 0.35 .

Pour nos experimentations, nous avons executer les algorithmes du Benchmark de SKAB dataset à savoir : Conv-AE, LSTM-AE, LSTM-VAE et Vanilla-AE ensuite nous avons utiliser notre autoencodeur pour la détection sans le mettre à jour puis nous vons testé notre modèle "Update Based Prototypes" . Pour chacun de ces modèles nous avons evaluer le F1-score, le taux de manques d'alarmes, le taux de fausses alarmes et le temps d'execution. Les résultats obtenus ont été consignés dans le tableau suivant 4.3 :

Algorithme	F1-score	FAR(%)	MAR(%)	Temps d'execution(s)
Conv-AE	0.78	13.55	28.02	420
LSTM-AE	0.74	29.96	25.92	520
LSTM-VAE	0.56	9.13	55.03	509
Vanilla AE	0.39	2.59	75.15	415
VAE Sans mise à jour	0.59	48.8	21.85	0.6
Update Based Prototypes	0.60	21.1	21.6	546

TABLE 4.3 – Résultats des expérimentations

Nous observons que bien que le F1-score de notre modèle ne soit pas maximale, Update Based Prototypes est le modèle qui trouve mieux le compromis entre le taux de fausses alarmes et le taux d'alarmes manquées. Ce qui nous permet de conclure de l'éfficacité de la solution proposée à savoir détecter efficacement les anomalies.

4.4.2 ARCANA-Prototypes : Explicabilité

Dans cette partie, nous avons tenté d'expliquer les anomalies détectées par notre modèle Update Based Prototypes. Nos expériences autour de l'explicabilité ont été faites essentielement autour du modèle ARCANA et ARCANA-Prototype. Pour tester la fidelité de notre modèle nous avons tenté deux expériences à savoir : la mise à 0 d'une caractéristique pour voir si le modèle sera capable d'identifier la caractéristique comme cause de l'anomalie et la mise à 0 des caractéristiques identifiées comme cause de l'anomalie pour vérifier que sans elles l'observation est normale . Nous avons repété les expériences afin de vérifier la stabilité des explications fournies par le modèle.

La mise à zéro d'une caractéristique

Pour notre première expérience nous avons mis la caracteristique " Current" à 0 comme indiqué sur la figure 4.2 afin de vérifier que les modèles d'explicabilité puisse la reconnaitre car la mise à zero de la caractéristique "Current" change la prédiction du modèle.

Nous avons donc exécuter nos deux algorithmes ARCANA et ARCANA-prototype deux fois et pour la première exécution nous avons obtenu les explications de la figure 4.3. Nous pouvons donc remarquer que ARCANA identifie la température et la pression comme étant les caracteristiques principales de l'anomalie et classe "Current" parmi les catractéristiques ayant le moins d'influence. ARCANA-prototype quant à lui classe Pressure et current comme les principales caractéristiques de l'anomalie . ARCANA-Prototype réussit donc à identifier Current comme une caractéristique principale dans la cause de l'anomalie.

Lors de la deuxième exécution ARCANA accorde plus d'importance à la caracteristique "Current" de même que l'algorithme ARCANA-Prototype comme nous pouvons l'observer sur la figure 4.4. Mais nous observons que les explications données par ARCANA-Prototype diffère significativement des explications de la première experience ce qui peut traduire la non stabilité du modèle ARCANA-Prototype

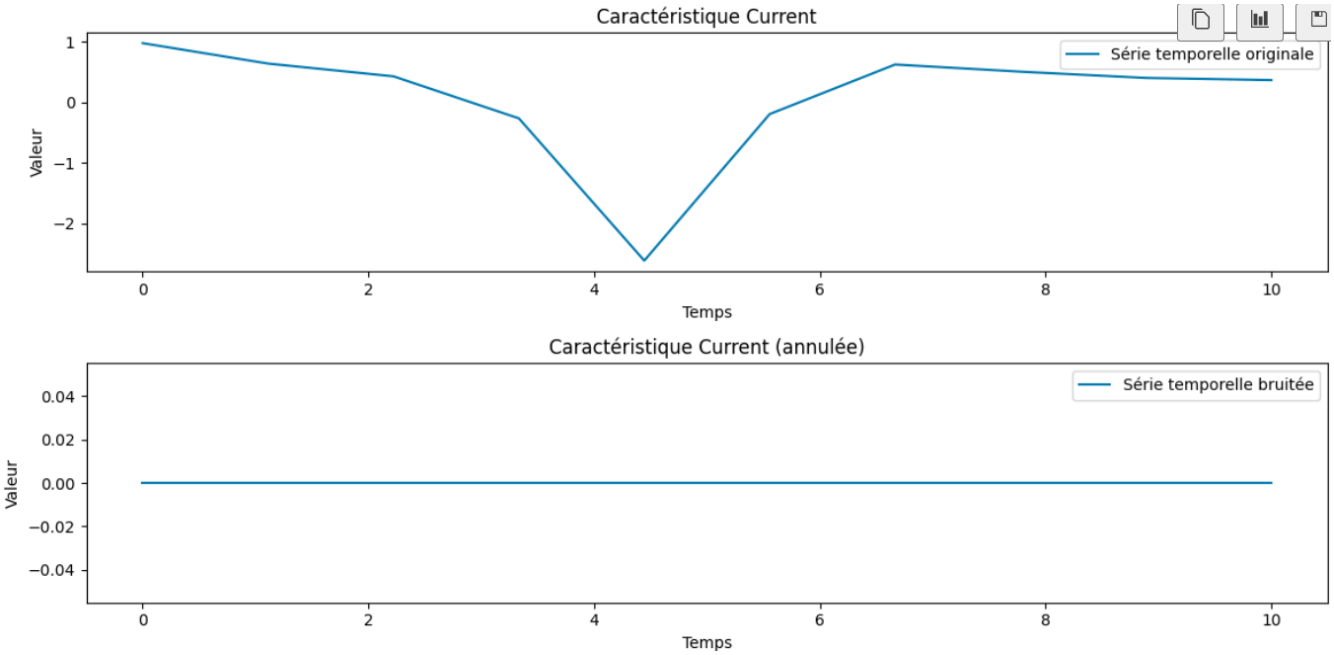
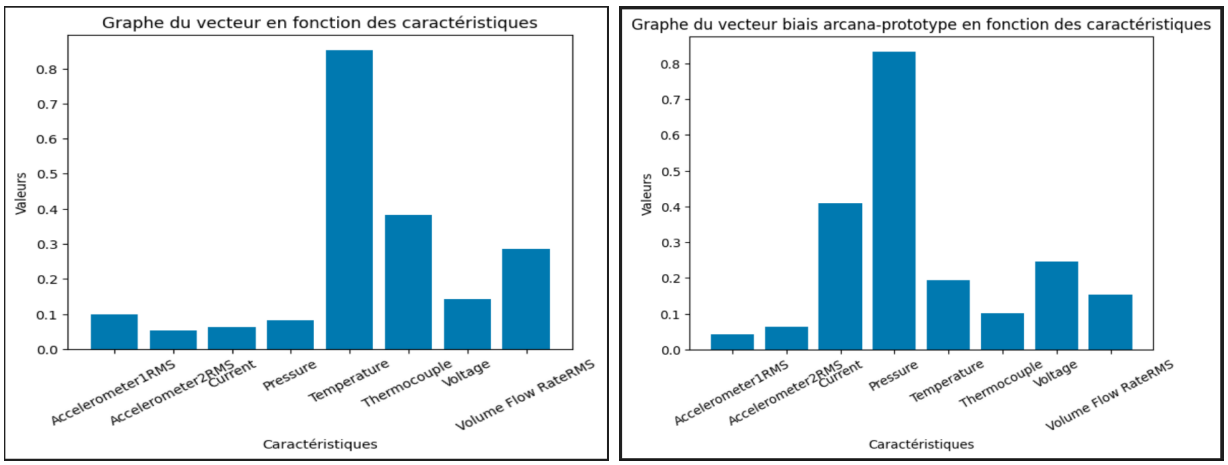
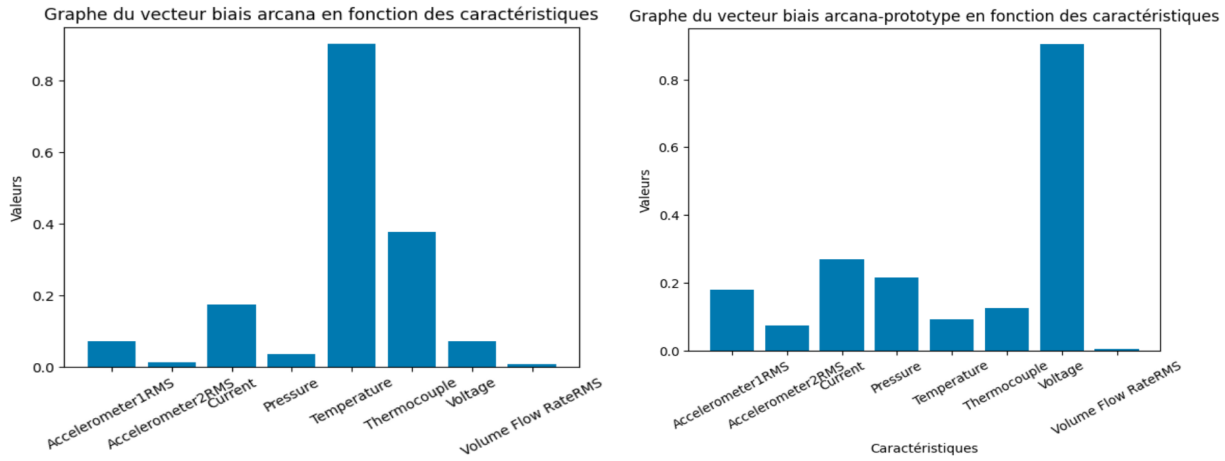


FIGURE 4.2 – Mise à zero de la caracteristique "Current"



(a) ARCANA (b) ARCANA-Prototype

FIGURE 4.3 – Première expérience de la mise à zéro de la caracteristique "Current"

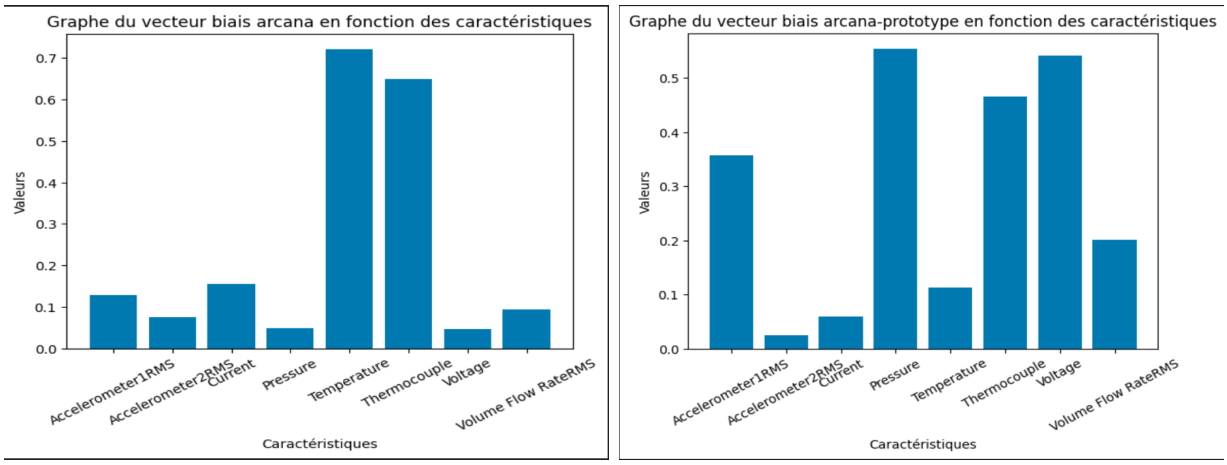


(a) ARCANA (b) ARCANA-Prototype

FIGURE 4.4 – Deuxième expérience de la mise à zéro de la caractéristique "Current"

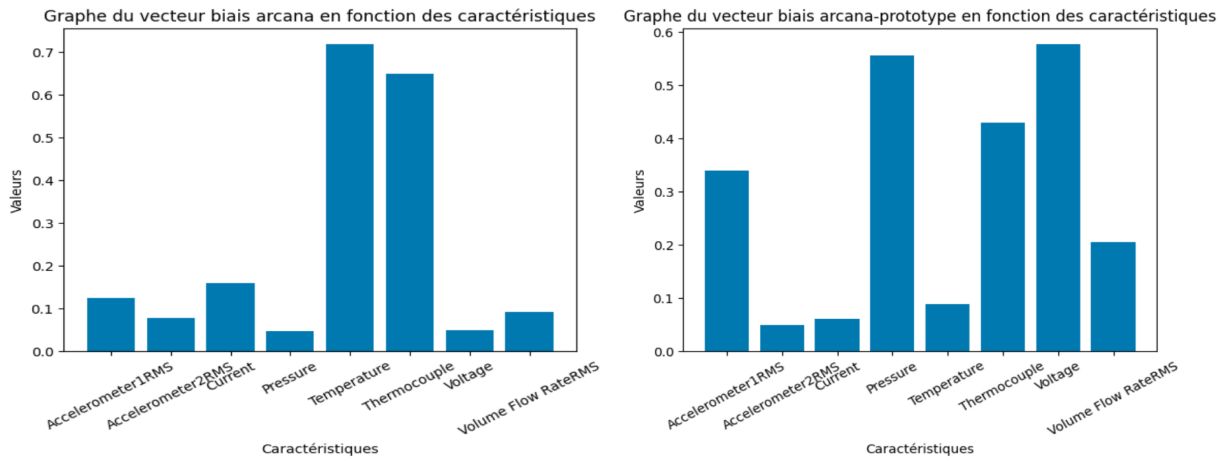
La mise à zéro des caractéristiques identifiées

Pour cette expérience nous avons choisie une donnée anormale reconnue par le modèle comme anormale et nous avons tenté de l' expliquer puis nous avons mis à zéro les caractéristiques identifiées pour voir si cela changerait la prédiction. Nous constatons que les prédictions des modèles pour la première expérience 4.5 et celles de la deuxième expérience 4.6 sont similaires. La mise à zéro des caractéristiques "Temperature" et "Thermocouple" ont permis de changer la prédiction du modèle. De même, la mise à zéro des caractéristiques : Accelerometer1RMS, Pressure, Thermocouple et Voltage met également la prédiction à FALSE. ARCANA identifie donc spécifiquement deux caractéristiques importantes dans l'apparition de l'anomalie.



(a) ARCANA (b) ARCANA-Prototype

FIGURE 4.5 – Première expérience de mise à zéro des caractéristiques identifiées



(a) ARCANA (b) ARCANA-Prototype

FIGURE 4.6 – Deuxième expérience de mise à zéro des caractéristiques identifiées

4.5 Bilan du chapitre

Il était question dans ce chapitre de présenter les résultats obtenus après quelques expérimentations de notre détection et explication d'anomalies. Les résultats obtenus sur SKAB dataset ont permis de voir que en général, avec un seuil supérieur à 0.10, on obtient une bonne détection mais aussi l'ajout de la mise à jour du modèle permet d'améliorer davantage la détection bien que cela prenne beaucoup de temps. Aussi, la fonction ARCANA-Prototype permet d'obtenir de bons résultats et des explications fidèles.

CHAPITRE 5

CONCLUSION ET PERSPECTIVES

Il était question pour nous dans ce mémoire d'étudier la détection d'anomalies et leur interprétation dans un contexte de flux de données. Pour cela, notre principale contribution était de prendre en compte l'évolution de la distribution des données dans le temps. Nous avons proposé une approche en deux temps basée sur les prototypes à savoir : "Update Based Prototypes" qui permet une détection des anomalies tout en prenant compte du concept drift dans les données à partir de la distance entre les données et le prototype de la classe associée permettant ainsi de mettre le modèle à jour lorsque cette distance est grande. La deuxième phase est celle de l'explicabilité des anomalies détectées par notre autoencodeur, nous avons utilisé les prototypes comme représentation cible des données normales et proposé une extension du modèle ARCANA nommée "ARCANA-Prototypes". De ces expériences, il en ressort que la mise à jour du modèle par notre approche "Update Based Prototypes" permet d'améliorer les résultats de la détection d'anomalie ; néanmoins la mise à jour continue du modèle pose un problème de temps bien que cela soit normal dans un contexte de flux de données. Aussi, l'utilisation des prototypes pour l'explication des anomalies à travers le modèle "ARCANA-Prototypes" permet de donner des résultats fidèles bien que les explications ne soient pas très stables au cours des expériences.

Pour la suite, nous comptons essayer de réduire le temps d'exécution en utilisant la parallélisation afin de rendre la mise à jour plus rapide. Toutefois, une étude approfondie sur les paramètres des prototypes notamment le nombre de prototypes nécéssaires pour représenter éffectivement tous les élements d'une classe est envisagée.

BIBLIOGRAPHIE

[1] Supriya Agrahari and Anil Kumar Singh. Concept drift detection in data stream mining : A literature review. *Journal of King Saud University-Computer and Information Sciences*, 34(10) :9523–9540, 2022.

[2] Gabriel Aguiar, Bartosz Krawczyk, and Alberto Cano. A survey on learning from imbalanced data streams : taxonomy, challenges, empirical study, and reproducible experimental framework. *Machine learning*, 113(7) :4165–4243, 2024.

[3] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1) :1–18, 2015.

[4] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM, 2002.

[5] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook : data mining and knowledge discovery handbook*, pages 353–374, 2023.

[6] Mohammed Ayalew Belay, Sindre Stenen Blakseth, Adil Rasheed, and Pierluigi Salvo Rossi. Unsupervised anomaly detection for iot-based multivariate time series : Existing solutions, performance analysis and future directions. *Sensors*, 23(5), 2023.

[7] Penny Chong, Ngai-Man Cheung, Yuval Elovici, and Alexander Binder. Toward scalable and unified example-based explanation and outlier detection. *IEEE Transactions on Image Processing*, 31 :525–540, 2021.

[8] Joao Gama, Pedro Pereira Rodrigues, Eduardo Spinosa, and Andre Carvalho. Knowledge discovery from data streams. In *Web Intelligence and Security*, pages 125–138. IOS Press, 2010.

[9] Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6) :2508–2517, 2021.

[10] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *ACM SIGMOD Record*, 32(2) :5–14, 2003.

[11] T Ryan Hoens, Robi Polikar, and Nitesh V Chawla. Learning from streaming data with concept drift and imbalance : an overview. *Progress in Artificial Intelligence*, 1 :89–101, 2012.

[12] Iurii D. Katser and Vyacheslav O. Kozitsin. Skoltech anomaly benchmark (skab). `https://www.kaggle.com/dsv/1693952`, 2020.

[13] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[14] Longyuan Li, Junchi Yan, Haiyang Wang, and Yaohui Jin. Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder. *IEEE transactions on neural networks and learning systems*, 32(3) :1177–1191, 2020.

[15] Tianyuan Lu, Lei Wang, and Xiaoyong Zhao. Review of anomaly detection algorithms for data streams. *Applied Sciences*, 13(10), 2023.

[16] Scott Lundberg. A unified approach to interpreting model predictions. *arXiv preprint arXiv :1705.07874*, 2017.

[17] C. Molnar. *Interpretable Machine Learning*. Leanpub, 2020.

[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you ?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[19] Cyriana MA Roelofs, Marc-Alexander Lutz, Stefan Faulstich, and Stephan Vogt. Autoencoder-based anomaly root cause analysis for wind turbines. *Energy and AI*, 4 :100065, 2021.