Apprentissage auto-supervisé pour la détection d'anomalies dans des séries temporelles

Romain Bailly

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : STEP - Sciences de la Terre de l'Environnement et des Planètes
Spécialité : Sciences de la Terre et de l'Environnement
Unité de recherche : Grenoble Images Parole Signal Automatique

Apprentissage auto-supervisé pour la détection d'anomalies dans des séries temporelles

Self supervised learning for anomaly detection in time series

Présentée par :

Romain BAILLY

Direction de thèse :

Jérôme MARS PROFESSEUR DES UNIVERSITES, Université Grenoble Alpes	Directeur de thèse
Marielle MALFANTE Docteur en sciences, Université Grenoble Alpes	Co-encadrante de thèse

Rapporteurs :

Philippe DELACHARTRE
PROFESSEUR DES UNIVERSITES, INSA Lyon
Bertrand LUVISON
DOCTEUR EN SCIENCES HDR, CEA centre de Paris-Saclay

Thèse soutenue publiquement le **15 décembre 2023**, devant le jury composé de :

Jérôme MARS PROFESSEUR DES UNIVERSITES, Grenoble INP	Directeur de thèse
Philippe DELACHARTRE PROFESSEUR DES UNIVERSITES, INSA Lyon	Rapporteur
Bertrand LUVISON DOCTEUR EN SCIENCES HDR, CEA centre de Paris-Saclay	Rapporteur
Régine LE BOUQUIN JEANNES PROFESSEURE DES UNIVERSITES, Université de Rennes	Examinatrice
GIFFARD ROISIN SOPHIE CHARGEE DE RECHERCHE, IRD délégation régionale Sud Est	Examinatrice
Pascal PERRIER PROFESSEUR DES UNIVERSITES, Grenoble INP	Président

Invités :

Marielle Malfante
DOCTEURE EN SCIENCES, Université Grenoble Alpes

Image artificielle générée à l'aide de l'intelligence artificielle DALL·E [RPG+21] avec pour consigne *"Illustration for the cover of my PHD thesis about artificial intelligence, digital art, 4k"*

Remerciements

Ca y est, les dernières fautes sont corrigées, il est maintenant venu pour moi le temps de rédiger la dernière partie de mon manuscrit : celle-ci. Comme tout le reste pendant cette thèse, avec beaucoup d'avance comparé à la deadline (sarcasme).

Tout d'abord, je souhaite remercier le jury, qui a fait l'effort de lire ce manuscrit et de m'écouter palabrer le jour de la soutenance.

Ensuite, je souhaiterai remercier mes encadrants, sans qui cette thèse n'aurait pas eu lieu. Jérome, pour tes conseils sur la gestion de la thèse et pour avoir su me recadrer quand j'en avais besoin. Marielle pour avoir toujours eu ta porte ouverte pour discuter quand c'était nécessaire, et ce jusqu'à la dernière minute.

Je voudrais ensuite remercier les experts des données qui m'ont donné de leur temps pour m'expliquer les problématiques qu'ils rencontrent tous les jours. Merci à Cédric, Chiara, Lamya, Eleonore et Taichi.

Viennent ensuite mes collègues, qui m'ont supporté pendant 3 ans, afin d'éviter d'oublier quelqu'un, je vais le faire par ordre géographique des bureaux. Merci à Pascal, Tin, Régis, Romain (pour ce monitoring parfait de la température du labo), Laurent, David (que je n'ai pas vu beaucoup au labo, mais que je revoie avec plaisir aux soirées jeux) Quentin, Jeremy, Sovan (bien que cette pile n'ai jamais atteint le plafond), Maxime, le vrai Paul, le faux Paul (merci pour tous ces memes !), Marion, Miguel, Yanis (je me souviens encore de ces moments passés au Raid, ou sur le tandem), Phan, Maxime (l'autre), Mathias, Raphaël, Pierre, Carolynn, Anca, Marina, Olivier, Laurent (lire *Laur-un*), Louis, Thomas et Diego. Un remerciement tout particulier à Aurélien, pour tout ce temps passé ensemble, les énigmes, les LEDS, les jeux de société et autres discussions non professionnelles et à Nils, de m'avoir accompagné en tant que co-bureau pendant ces trois ans, dans les bons jours comme dans les mauvais, pour toutes ces discussions sur les derniers jeux sortis, et toutes les autres choses qui ont rendu le quotidien de cette thèse si agréable. Merci aussi de m'avoir aidé à monter ce billard au 7ème étage dans l'escalier Nils.

Je tiens à féliciter Laurent, pour s'être beaucoup trop impliqué dans ce qui n'était qu'une blague au départ, 36 canards 🐤 à ton actif au moment où j'écris ces lignes, tu les as bien massacrés !

Merci à mes amis de Discord, dit *la mafia du trade*, Lim', Galaad, Ticot, Calife, Celine512, Jean, Rizflex et soulpass.

Merci à Esteban, mon binôme de toujours (qui aurait pu être en train de rédiger ces lignes si ce n'était pas moi qui avait été choisi pour ce stage) et à Quentin, pour toutes ces parties de Rocket League, même si je n'étais clairement pas des plus constants.

Merci aux Kaktous d'avoir été là, d'avoir traversé les mêmes galères, au même moment, ensemble. Le pastagave, qui n'était censé être qu'une blague, c'est transformé en une véritable amitié. Merci à Cédric, Benjamin, Paul, Marie (pour ces fins de tacos bien trop tardive),

Jeya (pour ces idées farfelues), Nils (tu as déjà été mentionné mais bon, tu mérites bien une deuxième mention), Adèle - *Adéloulou* - et Marc - *Marcpépette* - pour tout ces moments passés ensemble (souvent autour d'une bière, je dois le reconnaitre).

Enfin, merci à Pauline, ma copine, *mon coeur*, pour tout le reste. Pour m'avoir sorti quand tu savais que j'en avais besoin mais aussi pour m'avoir laissé travailler quand il le fallait. Merci pour tout ces moments passés ensemble. Pour toutes ces activités que tu m'as fait et que te me fais encore découvrir. Pour tous ces moments de distraction au milieu du travail intense. Pour l'adoption du plus mignon des petichats. Et pour bien plus encore que je ne saurais exprimer par des mots.

Notations

Notation	Signification
	Notations génériques
$\mathbb{X}$	Ensemble des échantillons x_i
x	Un échantillon quelconque de $\mathbb{X}$
x_i	i^{eme} échantillon de $\mathbb{X}$
$\mathbb{Y}$	Ensemble des annotations
y	Un échantillon quelconque de $\mathbb{Y}$
y_i	i^{eme} échantillon de $\mathbb{Y}$
	L'annotation y_j est associée à l'exemple x_j
$\mathcal{W}$	Ensemble des paramètres (poids) d'un modèle
W	Ensemble des poids w d'une couche spécifique d'un réseau profond
w	Poids quelconque d'un modèle
N	Nombre de couches d'un réseau de neurones
φ	Fonction d'activation
$\mathcal{L}$	Fonction de cout
$\mathcal{N}$	Nombre de classes de la tâche considérée
	Exposants et indices
(n)	Couche de l'objet considéré
i	Indice lié à l'entrée i
j	Indice lié à la sortie j
	Jeux de données
$\mathcal{A}$	Nom du jeu de données
$\mathcal{A}_{\text{train}}$	Sous jeu d'entrainement de $\mathcal{A}$
$\mathcal{A}_{\text{val}}$	Sous jeu de validation de $\mathcal{A}$
$\mathcal{A}_{\text{test}}$	Sous jeu de test de $\mathcal{A}$
$\mathcal{A}_p$	Jeu de données augmenté avec un fenêtrage glissant de pas p
$a \leftarrow b$	La valeur b est assignée à la variable a

Les notes en pied de page correspondent aux nominations anglaises exactes qui ont pu être traduites dans ce document.

Table des matières

Table des figures

Liste des tableaux

CHAPITRE

I

Introduction

Table des matières

A Contexte

Le nombre d'objets connectés est en constante croissance depuis les dernières décennies [LXZ15, REC15]. Cette évolution d'internet baptisée web 3.0 [HHA+] consiste en une globalisation et une généralisation de l'acquisition de données à grande échelle et proche des utilisateurs. Alors que la collecte automatique de données était historiquement réservée aux expériences scientifiques, au contrôle de chaînes de productions ou à la défense, un grand nombre d'objets connectés se sont popularisés auprès du grand public : montres connectées, balances connectées, lampes connectées, enceintes connectées, portes de garages connectées, robots-cuiseur connectés, réfrigérateurs connectés ou encore toilettes connectées ! Cette profusion de données permet alors aux différentes entreprises commercialisant ces produits de proposer des services d'analyses des données issus des capteurs de leurs objets connectés.

Les capteurs de ces objets connectés peuvent acquérir des données de plusieurs types. Des données à une dimension ou séries temporelles, mais aussi des images qui sont des ensembles de pixels à 3 dimensions hauteur, largeur, couleur ; des vidéos à 4 dimensions hauteur, largeur, couleur, temps qui peuvent être assimilées à des séries temporelles d'images ou encore des champs de points comme ceux pouvant être obtenus par des Laser Imaging Detection and Rangings (LIDARs) dans le cas de la voiture autonome.

Remarque 1 : Les capteurs d'une montre connectée

Une montre connectée de dernière génération[1] dispose des capteurs suivants :

- Champ magnétique (x, y, z) pour la *boussole*,
- Pression pour l'*altimètre barométrique*,
- Luminance pour le *détecteur de lumière ambiante*,
- Accéléromètre (x, y, z),
- Gyroscope (x, y, z),
- Capteur de fréquence cardiaque optique,
- Capteur de fréquence cardiaque électrique,
- Capteur d'oxygène sanguin

Soit 14 séries temporelles différentes au sein du même objet connecté !

1. `https://www.apple.com/fr/watch/`

Il est possible de catégoriser les capteurs intelligents en fonction de la proximité physique de la plateforme de calcul permettant de réaliser les traitements de données nécessaires à l'obtention de l'information d'intérêt. Il est important de noter ici que plus les données sont loin au sens de la distance entre le capteur et la plateforme de calcul permettant le traitement des données, plus le cout matériel de stockage et/ou de transmission de ces données sera important [AMKF18, KW21].

D'un côté de la chaîne se trouve le calcul dans le nuage *, consistant à envoyer les données collectées à un serveur, une ferme de calcul hébergée par le fournisseur de l'objet disposant de plusieurs centaines, voire milliers [LS22] d'ordinateurs capables de transformer les informations brutes du capteur afin d'en extraire les données importantes qui sont ensuite renvoyées à l'utilisateur. De l'autre côté, le calcul à la marge † consiste à réaliser les opérations de traitement des données plus proche des capteurs. Bien que cette

*. cloud computing
†. edge computing

solution propose des avantages, elle reste néanmoins contrainte par la puissance de calcul bien plus faible qui peut être déployée au plus proche du capteur sans augmenter son cout, sa taille ou encore sa consommation électrique. [Say19] montre sur la figure I.1 une catégorisation intermédiaire appelée l'informatique en brouillard * qui consiste à utiliser des infrastructures plus proches que les énormes datacenters du nuage. La frontière entre les capteurs intelligents, embarquant directement l'intelligence nécessaire aux traitements des données et le calcul à la marge est encore floue aujourd'hui. Il existe autant de traitements possibles que de cas applicatifs et il n'est en fait pas toujours possible de classer certains cas dans des cases prédéfinies. Cette classification représente en fait un gradient continu de l'ensemble des possibilités de calcul entre le capteur intelligent et le calcul dans le nuage.

Chacun de ces scenarii présente des avantages et des inconvénients : le calcul dans le nuage dispose d'une puissance de calcul bien supérieure, mais nécessite le stockage local, la transmission des données, puis le stockage distant des données acquises. Ce stockage ainsi que la transmission des données peut alors poser des questions de gestion et d'utilisation des données des utilisateurs, notamment au regard de la RGPD [PC16]. Par opposition, le calcul à la marge réalise l'interprétation des acquisitions proche du capteur, sur une plateforme embarquée, mais dispose d'une puissance de calcul ainsi que de capacités de stockage plus faibles. C'est pourquoi les algorithmes de traitement des données doivent alors être adaptés, optimisés et réfléchis pour être déployés sur ces plateformes (localisées). Faut-il adapter des algorithmes déjà existants pour les rendre plus légers et pouvoir les déployer sur des plateformes embarquées ou alors développer, à partir de zéro, des nouveaux algorithmes expressément développés pour fonctionner avec peu de ressources ? Les deux approches évoluent aujourd'hui en parallèle au sein de la communauté scientifique, en particulier pour les méthodes reposant sur l'IA [MPI20, SG23].

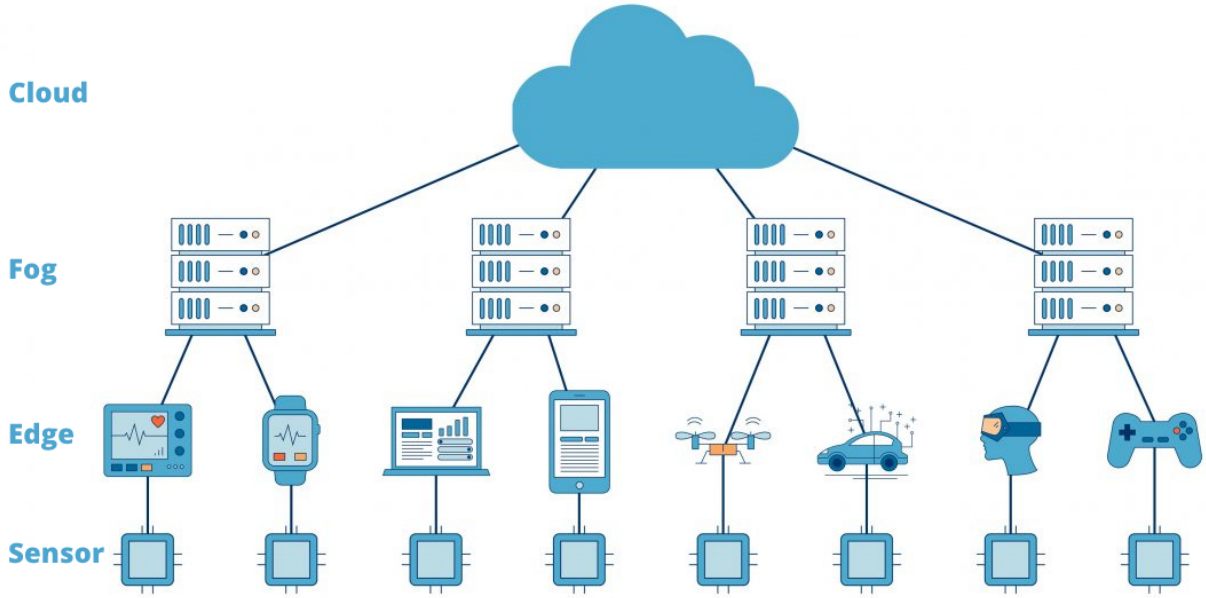


Figure I.1 : Illustration des différents niveaux de calculs pour l'internet des objets allant du calcul dans le nuage, puissant mais éloigné physiquement des utilisateurs, aux capteurs intelligents proches du capteur, mais avec de faibles ressources [Say19]

Les capteurs intelligents poussent la proximité entre l'acquisition des données et leur traitement au maximum. C'est cette "*intelligence*", autrement dit cette puissance de calcul et ces algorithmes de traitements optimisés, qu'embarquent le capteur, qui le rendent

*. fog computing

capable de réaliser les opérations nécessaires à la conversion des données brutes en données utiles. C'est finalement la donnée utile, d'intérêt pour l'utilisateur, qui peut ensuite lui être transmise à moindre cout. Le capteur ne renvoie plus les données brutes, mais directement l'information utile.

Cette thèse, intitulée *apprentissage auto-supervisé pour la détection d'anomalies dans des séries temporelles*, s'inscrit dans cette démarche de vouloir rapprocher les algorithmes au plus proche du capteur afin de réaliser les traitements nécessaires à l'interprétation des données avant leur envoi à l'utilisateur. En effet, effectuer directement le traitement des données sur le capteur permet de diminuer la quantité de données à stocker ou à transmettre à l'utilisateur. Il ne faut cependant pas que le cout calculatoire impliqué par le traitement local des données surpasse les couts de stockage ou d'envoi des données non traitées. Il est donc nécessaire de développer des algorithmes efficaces énergétiquement pour effectuer le traitement de ces données brutes, et dans le meilleur des cas, au plus proche du capteur.

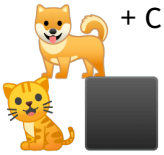
Dans cette optique de développement de méthodes algorithmiques adaptées pour les capteurs intelligents, nous avons choisi de nous concentrer sur le développement d'algorithmes et de méthodes spécifiques au traitement de séries temporelles. Une série temporelle est une suite de valeur(s) mesurant une ou plusieurs quantités au cours du temps. Le traitement des données à une seule dimension diminue radicalement la puissance de traitement nécessaire comparée au traitement d'image ou de vidéos par exemple. En effet, bien que les problématiques abordées dans le cadre du traitement d'image soient diverses et s'éloignent parfois des problématiques abordées en traitement de signal, l'utilisation de méthode d'IA, héritée du traitement d'image peut aussi être bénéfique au traitement de séries temporelles. Attention cependant, les vidéos peuvent être considérées comme des séries temporelles d'images, elles ne sont néanmoins pas l'objet des études menées dans ce manuscrit.

Plaçons-nous dans un contexte d'un capteur déployé sur le terrain. Imaginons qu'un capteur déployé sur le terrain de manière autonome compte les animaux passant devant lui. Son algorithme repose sur un modèle d'apprentissage profond ayant été entrainé à différencier les chiens des chats. Ce modèle est très performant à la détection et la classification des chiens et des chats comme le montre l'illustration tableau I.1. Cependant, dans un cas limite, le capteur acquiert la photo d'une souris  . Le modèle ayant été entrainé à discriminer chien et chat, il n'est pas capable de reconnaitre une nouvelle classe et tente quand même de lui associer une des deux classes apprises. Il peut même être particulièrement *confiant* sur sa prédiction. Le modèle n'est pas capable de s'adapter à une nouveauté, une anomalie à laquelle il est confronté.

L'étude de méthodes permettant de détecter ces anomalies devient donc cruciale, surtout pour un capteur intelligent fonctionnant en autonomie, afin qu'il puisse rejeter les informations pour lesquelles il n'est pas capable de répondre. Dans un cadre encore plus poussé, il serait possible de coupler cette détection d'anomalies à un modèle capable d'apprendre de nouvelles classes au fur et a mesure tel que le propose les méthodes d'apprentissage incrémental [HRRP20, AS19, Sol21, MSR+21].

La détection d'anomalies se révèle être une méthode d'une importance cruciale dans le domaine du traitement des séries temporelles. En effet, les séries temporelles contiennent souvent des variations et des schémas complexes, et l'identification rapide et précise des anomalies au sein de ces données peut avoir des implications significatives dans divers domaines. Les méthodes de détection d'anomalies, notamment celles basées sur l'apprentissage profond (mais pas uniquement), offrent la possibilité de modéliser les schémas normaux de manière adaptative et de détecter des déviations significatives, souvent imper-

Table I.1 : Cas limitant d'un modèle de classification. Lorsqu'un élément d'une nouvelle classe lui est présenté, il le classifie quand même dans l'une des classes apprises et peut être très confiant à propos de cette classification

Entrainement	Usage classique	Nouvelle classe
 + Chien + Chat		Chien : 98% Chat : 2% Chien : 3% Chat : 97%		Chien : 82% Chat : 18%

ceptibles à l'œil humain. L'apprentissage ne répond donc pas à la même problématique que les méthodes de détection reposant sur un savoir expert. Dans le cas où le phénomène observé n'est pas complètement compris, l'apprentissage permet de tirer parti d'un ensemble de données afin de déterminer automatiquement, à partir de ces données, des règles sous-jacentes permettant de décrire les observations. Cela s'oppose à des systèmes experts où un modèle est méthodique conçu grâce à la connaissance du phénomène pour décrire la réalité physique observée.

Le dictionnaire Le Robert [É23] donne la définition suivante d'une anomalie :

> anomalie *n.f.* : Écart par rapport à la normale ou à la valeur théorique, exception à la règle

On comprend bien qu'il n'est possible de définir une anomalie que par opposition à une certaine *norme*. S'il n'y a pas d'observation que l'on peut définir comme normale, la notion d'anomalie perd alors tout son sens. Dans certains cas applicatifs particuliers, cette norme peut être mal définie, voire non définissable. Le fait de ne pas pouvoir différencier, même en amont, les sujets normaux des sujets anormaux, pose la question suivante : Comment détecter des anomalies, sans connaitre la définition des anomalies? En apprentissage automatique, l'association d'un objet et de sa classe normal/anormal constitue une annotation, ayant souvent été manuellement fournie par un expert du domaine considéré. Dans le cas de la détection d'anomalies ces annotations ne sont pas toujours disponibles et c'est pourquoi nous proposons dans ce manuscrit des méthodes de détection d'anomalies ne nécessitant pas d'annotations.

B Un peu de vocabulaire

Cette section présente quelques mots de vocabulaire utilisés dans ce manuscrit ainsi que leur définition. Elle n'est pas aussi exhaustive que le glossaire section F, mais détaille cependant les concepts les plus importants à connaître avant de lire ce manuscrit.

Modèle D'après la CNIL, "*Le modèle [...] est la construction mathématique générant une déduction ou une prédiction à partir de données d'entrée*" [CNI23]. Le mot modèle est le terme le plus générique que l'on puisse utiliser pour décrire un algorithme. Il peut aussi bien être utilisé pour parler de modèle de détection d'anomalie, que de réseaux de neurones profonds.

Architecture L'architecture est un terme spécifique à l'apprentissage profond. Elle définit la structure interne du modèle à proprement parler.

Hyperparamètre L'architecture d'un réseau de neurones est elle-même définie par un ensemble de paramètres qui régissent comment doivent s'organiser les couches du réseau ainsi que leur contenu.

Entrainement L'entrainement des réseaux de neurones est le processus d'ajustement des poids et des biais du réseau en utilisant des données d'apprentissage afin de minimiser l'erreur entre les prédictions du réseau et les annotations attendues, permettant ainsi au réseau d'apprendre à effectuer une tâche spécifique.

Inférence L'inférence d'un réseau de neurones désigne l'utilisation du modèle déjà entrainé pour effectuer des prédictions sur de nouvelles données d'entrée, sans modifier les paramètres du réseau. C'est le régime de fonctionnement du modèle.

Série temporelle Dans ce manuscrit le terme "série temporelle" fait référence à la mesure d'une ou plusieurs modalités pendant toute la durée de l'expérience par opposition au terme "fenêtre" qui correspond à une partie tronquée de la série temporelle complète.

C Cadres applicatifs

La science des données et en particulier l'IA repose en grande partie sur les données utilisées, observées et interprétées par les modèles. Cette section présente les contextes applicatifs des données utilisées dans cette thèse. Des acquisitions biologiques de masse sèche de cellules issues de microscopie sans lentille sont présentées section C.1 et des données sismiques de la planète Mars section C.2. Nous attachons un intérêt particulier à l'utilisation de données réelles afin de répondre à des problématiques applicatives en parallèle des questions méthodologiques auxquelles nous répondons dans ce document. L'application y est un support qui guide les réflexions méthodologiques nécessaires à la réponse à des problématiques applicatives.

C.1 Masse sèche de cellules et microscopie sans lentille

Cette étude a eu lieu dans le cadre du projet CARNOT Micro3DN en collaboration avec le Département micro-Technologies pour la Biologie et la Santé (DTBS) du CEA Grenoble et en particulier Cédric Allier, Chiara Paviolo et Lamya Ghenim.

Contexte des données : Le CEA LETI développe depuis 2009 une technologie de microscopie sans lentille innovante [AMV+17]. Cette dernière, en rupture avec les technologies d'imagerie classiques permet d'obtenir des images à haute définition et à très grand Champ de vision (FOV) * de cellules [MML+13]. Cette méthode permet l'acquisition d'images contenant des milliers de cellules ce qui n'est pas possible avec des méthodes classiques d'imagerie. Par ailleurs, cet objet étant bien plus compact qu'un microscope classique (figure I.2), il peut facilement être transporté. Par exemple, il peut être utilisé dans un dispositif permettant de contrôler l'environnement de croissance des cellules, appelé un incubateur. Il est alors possible de suivre la croissance de cellules en temps réel. En plus d'une compacité accrue, ces dispositifs sont beaucoup moins couteux que des microscopes optiques classiques [WO18]. Ils sont donc particulièrement utiles pour des applications liées à l'analyse du vivant au sens large [ABA+17, AYW+22, CBC+20].

Les microscopes sans lentille fonctionnent de la manière suivante : une Diode ÉlectroLuminescente (DEL) est positionnée au-dessus de l'échantillon biologique. La lumière émise est ensuite diffractée par l'échantillon, ce qui engendre la formation d'une image holographique. Cette image, dite holographique, est capturée par un capteur CMOS situé sous l'échantillon. Cependant, cette image d'interférence ne peut être directement interprétée visuellement. Un algorithme ([AMV+17] dans notre cas) est capable de reconstruire une

*. Field Of View

image à large FOV des cellules observées. La figure I.3 montre un exemple d'image reconstruite acquise grâce à un dispositif de microscopie sans lentille. Chacun des points blancs sur l'image est une cellule de l'échantillon observé.

Figure I.2 : Dispositif de microscopie sans lentille

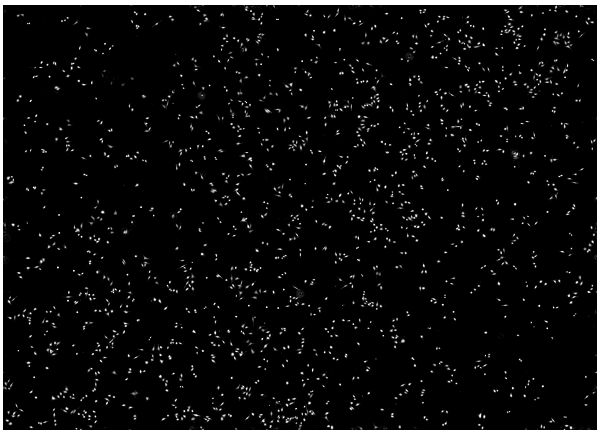


Figure I.3 : Image d'environ 30 mm^2 acquise par un microscope sans lentille. Les points gris sont des cellules

Les données en pratique : L'utilisation d'une telle technologie, couplée à des algorithmes de segmentation et de suivi cellulaire permet d'extraire des séries temporelles décrivant la vie des cellules au cours du temps. En particulier la masse sèche des cellules qui correspond à la masse des cellules si elles avaient été privées de toute leur eau. Cette métrique représente donc le matériel génétique de la cellule [AMV+17] qui est une image de la santé globale de la cellule.

Problématiques : Trois problématiques ont été identifiées sur ces données d'origines biologiques :

- La détection automatique de cellules anormales sans label et sans aucun *a priori*. Les méthodes de détection d'anomalies actuelles reposent sur la définition manuelle de métriques permettant de suivre l'évolution de la vie de cellules et de quantifier leur *normalité*. Cependant, ces techniques reposent sur la définition de la normalité définie par l'utilisateur qui introduit alors un biais humain dans la détection d'anomalies. La détection de nouvelles anomalies est alors plus complexe.
- La prédiction du destin * de la cellule. Être capable de prédire le futur de la cellule et notamment sa division pour pouvoir mieux comprendre son comportement.
- L'évaluation de la méthode de reconstruction des images à partir des images holographiques de l'imagerie sans lentilles.

C.2 Données sismiques de Mars

Cette étude est l'issue d'une collaboration avec l'Institut de Physique du Globe de Paris (IPGP) et en particulier Eléonore Stutzmann, Taichi Kawamura et Grégory Sainton.

Contexte des données : La sonde InSight (Interior Exploration using Seismic Investigations, Geodesy and Heat Transport), lancée en 2018, est une sonde ayant pour objectif de poser sur la planète Mars un sismomètre unique, aussi perfectionné que résistant, pour

*. cell fate

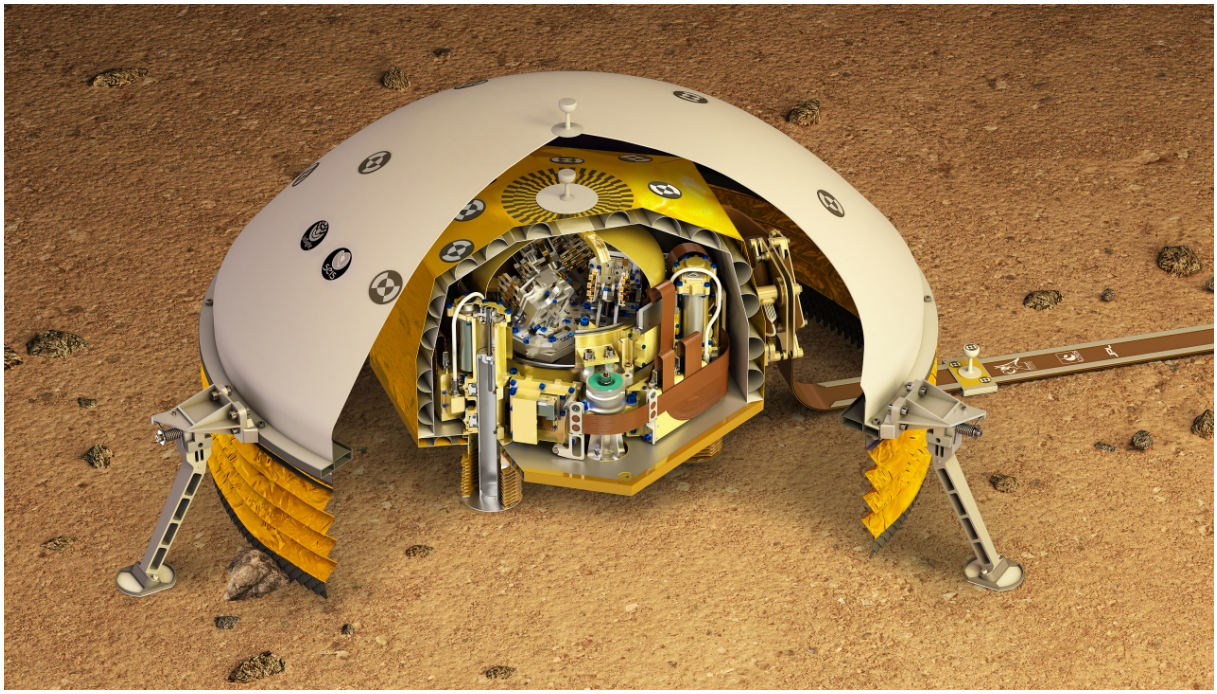


Figure I.4 : Le sismomètre SEIS : de l'extérieur vers le centre, on distingue les boucliers thermique et éolien, puis l'enceinte de confinement sphérique qui protège les pendules du sismomètre au centre (©IPGP/David Ducros [Duc18])

enregistrer pour la première fois l'activité sismique de MARS [GWG+20]. Équipée du sismomètre SEIS [LBG+19, BSB+20], sa sensibilité est mille fois supérieure aux sondes de Viking [ADL+76].

Le sismomètre SEIS a été fourni par le Centre national d'études spatiales (CNES) en France, en collaboration avec des institutions internationales. Il a été installé sur la surface martienne par le bras robotique d'InSight, à proximité de la sonde elle-même. On voit sur l'illustration figure I.4 qu'il est protégé des conditions environnementales par un bouclier thermique afin de minimiser les interférences induites par les vents ou la température.

Les données en pratique : Les données sismologiques de Mars sont des acquisitions d'accélérométrie 3 axes, mais aussi des données de pression, acquises sur une période d'un mois entre juin et juillet 2019. [LBG+19]

Problématiques :

- L'objectif premier de la mission est une meilleure compréhension de la géologie et sismologie de Mars sans avoir accès au sol et uniquement à partir des mesures effectuées *in situ* [Duc18, GLB+20].
- La détection des séismes martiens est un défi en soi, car les mouvements du sol peuvent être très faibles et difficiles à distinguer du bruit de fond. Le vent, les variations thermiques, les vibrations mécaniques de la sonde et d'autres facteurs peuvent introduire du bruit dans les données sismiques, ce qui rend la détection et l'analyse des séismes plus complexes [GLB+20, LBP+20, SWD+20].
- Les données sismiques sont envoyées sur terre, dans un premier temps, en faible qualité. Lorsque ces données sont analysées, les experts peuvent envoyer une requête au lander pour obtenir certaines données avec une fréquence d'échantillonnage plus élevée. Cependant, la transmission de nouvelles données prend du temps

et est couteuse en énergie. La détection automatique d'évènement sismique, sans avoir besoin d'analyse experte permettrait d'envoyer directement en haute définition les données d'intérêt sismique [CCv+21, CCG+22].

C.3 Différences et complémentarités entre les jeux de données

Le point clé reliant ces deux jeux de données est l'étude de séries temporelles. Cependant, bien que ces deux études semblent totalement décorrélées, elles prennent tout leur sens au regard des sujets abordés dans ce manuscrit :

— Alors que les données cellulaires sont accessibles (au sens de la proximité physique) et ne présentent pas une forte contrainte quant à l'embarquabilité des algorithmes développés car ce dispositif peut être directement relié à un ordinateur, la contrainte pour le développement d'algorithmes pour une plateforme embarquée comme un lander martien est forte. On comprend aussi l'intérêt de capteur intelligent, totalement autonome dans ce cas où l'accès aux données est très compliqué.
— Les données cellulaires contiennent un nombre important de séries temporelles relativement courtes alors que les données martiennes contiennent une seule série temporelle de très longue durée en comparaison. Les méthodes proposées sont assez génériques afin de s'adapter aux deux jeux de données bien que la sémantique derrière ces séries temporelles soit totalement différente.
— Les données cellulaires semblent plus simples, car une unique modalité est étudiée à la fois alors que les données sismiques sont composées d'au moins 3 modalités de vitesse (U, V, W).

La comparaison entre les points communs et les similarités de ces deux jeu de données seront par ailleurs discutées en conclusion section B.2 au regard des résultats obtenus.

D Problématique

La thèse présentée dans ce manuscrit se concentre principalement sur le développement de méthodes de détection d'anomalies pour les séries temporelles. Cette détection doit pouvoir être réalisée automatiquement et donc sans avoir à disposition des annotations correspondant aux classes des sujets observés. Ces annotations peuvent être indisponibles à la fois dans le cas d'un capteur intelligent qui doit fonctionner en autonomie, mais aussi dans le cas de découverte de nouvelles classes, qui n'ont jamais été identifiées auparavant. Comme nous l'avons déjà exposé, la détection de nouvelles classes peut elle-même rentrer dans des considérations de fiabilité de l'IA. Ces problématiques peuvent être abordées toutes les deux sous le même angle d'approche qu'est la détection d'anomalie.

En particulier, nous proposons l'utilisation et le développement de méthode d'apprentissage de représentation autosupervisée (ne nécessitant donc pas d'annotations) comme solution à cette contrainte d'annotation forte.

Problématique :

Comment l'intelligence artificielle peut répondre à l'absence de modèle physique, en apprenant de données non annotées, pour détecter des anomalies lorsque la définition d'une anomalie est inconnue ?

En effet, nous cherchons dans ce manuscrit à mettre en avant l'utilisation de l'IA, non pas à la place de modèles physiques, mais bien en complément de ces derniers. Comment

procéder pour réaliser cette détection d'anomalies lorsqu'il n'est pas possible de décrire le phénomène observé grâce à un modèle mathématique ? Nous proposons d'utiliser l'IA comme solution à ce problème, cette dernière étant capable d'apprendre automatiquement des motifs au sein des données et de les utiliser à leur avantage pour réaliser une tâche donnée. En effet, nous cherchons dans ce manuscrit à mettre en avant l'utilisation de l'IA, non pas à la place de modèles physiques, mais bien en complément de ces derniers. Comment procéder pour réaliser cette détection d'anomalies lorsqu'il n'est pas possible de décrire le phénomène observé grâce à un modèle mathématique ? Nous proposons d'utiliser l'IA comme solution à ce problème, cette dernière étant capable d'apprendre automatiquement des motifs au sein des données et de les utiliser à leur avantage pour réaliser une tâche donnée.

Cette problématique soulève d'autres questions qui seront en partie abordées dans ce manuscrit : Comment détecter des anomalies lorsque nous n'avons aucun a priori sur les anomalies que l'on recherche ? En particulier, quand aucune annotation n'est disponible. Comment bénéficier de la puissance de représentation des réseaux de neurones profonds pour détecter des anomalies ? L'apprentissage auto-supervisé se démocratisant ces dernières années pour le traitement d'image peut-il s'appliquer à la tâche de détection d'anomalies ? Spécifiquement dans le cas du traitement des séries temporelles, partie de l'état de l'art encore peu développée aujourd'hui.

E Contributions

Afin de répondre à la problématique de ce manuscrit énoncée ci-dessus, nous proposons un certain nombre de contributions méthodologiques.

1. Le modèle StArDusTS est la plus importante contribution de cette thèse. Nommé StArDusTS pour Self-supervised Anomaly Detection on Time Series, c'est un modèle adapté pour la détection d'anomalies, sans annotations en particulier dans les séries temporelles. Il repose sur deux blocs distincts. Un premier permettant d'apprendre une représentation des données grâce à des réseaux de neurones convolutionnels. Comme nous cherchons à travailler dans ce manuscrit sans annotations, StArDusTS repose sur un paradigme émergent appelé apprentissage auto-supervisé ne nécessitant pas d'annotations pour apprendre une représentation des données. Ces méthodes reposent sur une tâche auxiliaire, appelée tâche prétexte. Nous étudions deux tâches prétextes pour l'apprentissage de représentation de séries temporelles :
   — La prédiction du futur de la série temporelle
   — L'utilisation de l'apprentissage contrastif

   Un second bloc de détection d'anomalies est ensuite utilisé sur la représentation apprise grâce à l'apprentissage auto-supervisé.
2. Nous proposons le calcul score d'anomalie permettant d'agréger des anomalies détectées au niveau d'une fenêtre d'analyse donnée.
3. Les réseaux de neurones présentent une grande complexité dans le choix des valeurs qui définissent leur architecture, appelé hyperparamètres. Nous proposons une méthode d'optimisation successive des hyperparamètres qui reprend et formalise l'optimisation souvent réalisée manuellement par les chercheurs du domaine.
4. Un outil algorithmique interne pour l'entrainement et la comparaison des multiples assemblages possibles de bloc d'apprentissage de représentation et de détection d'anomalies pour StArDusTS a été développé.

Nous présentons ci-dessous un résumé des conclusions, aussi bien méthodologiques qu'applicatives tirées des expériences menées sur les jeux de données cellulaires et sismiques de Mars. Nous présentons dans un premier temps les conclusions obtenues grâce aux données cellulaires puis grâce aux données sismiques de Mars.

Données cellulaires

Conclusions applicatives

— Il est possible de prédire le futur de la masse sèche de cellules grâce à des réseaux de neurones CNN1D
— Les données cellulaires contiennent à la fois des anomalies biologiques et des anomalies d'acquisition,
— Parmi les anomalies biologiques, 6 sous-catégories d'anomalies cellulaires ont pu être identifiées,

Conclusions méthodologiques

— Le modèle StArDusTS est capable de détecter des anomalies uniquement à partir de séries temporelles,
— La représentation apprise grâce à la tâche prétexte de prédiction est assez générique pour s'adapter à des données de cellules différentes de celles sur lesquelles le modèle a été entrainé,
— La tâche prétexte de prédiction ne permet pas de séparer les données normales de données anormales dans l'espace latent.

Données sismiques de Mars :

Conclusions applicatives

— Les données sismiques de Mars sont beaucoup plus complexes que les données cellulaires

Conclusions méthodologiques

— Les CNN1D ne sont pas capables de prédire les données sismiques de Mars,
— Le choix des données utilisées est crucial au succès de l'apprentissage de représentation,
— Le choix et le développement de méthodes d'augmentation de données spécialisées aux données traitées sont nécessaires au bon fonctionnement de l'apprentissage contrastif
— Le modèle StArDusTS ne peut pas être utilisé tel quel sur n'importe quel jeu de données et nécessite une adaptation, parfois conséquente.

Liste des contributions

Ce doctorat a donné lieu à plusieurs publications détaillées ici :

— Bailly, Romain, Marielle Malfante, Cédric Allier, Lamya Ghenim, et Jerome I. Mars. « Self-supervised learning for anomaly detection on time series : Application to cellular data ». *Conférence sur L'apprentissage Automatique*. Saint Etienne, France, 2021.

— « Deep Anomaly Detection Using Self-Supervised Learning : Application to Time Series of Cellular Data ». *ASPAI 2021 - 3rd International Conference on Advances in Signal Processing and Artificial Intelligence*, 2021.
— Bailly, Romain, Marielle Malfante, Cédric Allier, Lamya Ghenim, et Jérôme Mars. « Comparaison des capacités prédictives de réseaux de neurones, application à la masse sèche de cellules ». *XXIXème colloque francophone de traitement du signal et des images (GRETSI 2023)*, 1, 2023.
— Romain Bailly, Marielle Malfante, Cédric Allier, Chiara Paviolo Lamya Ghenim, Kiran Padmanabhan, Sabine Bardin, Jérôme Mars « StArDusTS, a self-supervised model for anomaly detection on time series. Application to the detection of abnormal behaviors in cell dry mass time series ». *Scientific Reports*, 2023. **SOUMIS**

F Organisation du document

Afin de répondre à la problématique de cette thèse, un état de l'art des méthodes existantes est présenté chapitre II. Cet état de l'art permet d'un côté de présenter l'ensemble des méthodes existantes qui seront utilisées dans ce manuscrit, mais aussi de présenter un ensemble de méthodes qui ne seront pas utilisées afin de pouvoir positionner le travail présenté dans les sections suivantes par rapport au reste de la littérature.

Les contributions méthodologiques de cette thèse présentées dans la section précédente sont détaillées dans le chapitre III.

Le chapitre IV est consacré à l'étude des données cellulaires. Les jeux de données utilisés ainsi que les problématiques inhérentes à cette étude de cas sont détaillées avant de présenter deux jeux d'expériences. Le modèle StArDusTS présenté chapitre III est appliqué aux données cellulaires. Il est couplé à la méthode d'optimisation successive des hyperparamètres implémentée dans l'outil algorithmique de comparaison. Les expériences menées dans ce chapitre permettent à la fois de valider les capacités de détection d'anomalies du modèle proposé, mais aussi l'étude des anomalies détectées d'un point de vue applicatif.

Identiquement, le chapitre V se penche sur l'étude de cas des données sismiques de Mars en commençant par une présentation des données et des problématiques qui leur sont associées. Les expériences sur ces données se concentrent sur la comparaison de la tâche prétexte utilisée dans StArDusTS, en particulier la tâche prétexte de apprentissage contrastif, mais aussi de l'algorithme de détection d'anomalies qui lui est couplé. Les résultats obtenus permettent de mettre en perspective les capacités du modèle StArDusTS.

Enfin le chapitre VI permet de conclure ce document et de mettre en perspective l'ensemble des résultats présentés.

Additionnellement, un glossaire, rappelant l'ensemble des définitions des termes techniques et des abréviations, est disponible page 13.

G Glossaire

acquisition Constitue l'ensemble des données acquises par le capteur ou la méthode d'acquisition avant d'avoir été prétraitées pour pouvoir être utilisées par un algorithme d'apprentissage automatique xv, 6, 8, 15, 51, 52, 71, 72, 74, 75, 83, 88, 92, 103, 104, 126, 131

annotation Aussi appelées étiquettes ou labels, les annotations consistent en des informations supplémentaires contenues au sein des données observées. Ces annotations sont souvent le résultat d'une analyse manuelle des données permettant de leur attribuer une classe par exemple. 5, 6, 9, 10, 13, 15, 20, 39–43, 45, 47, 58, 59, 61, 75, 85, 90, 102, 104–106, 120, 123, 127, 130, 131, 135, 136, 138, 139, 165

anomalie Ce qui s'écarte de la norme, de la régularité, de la règle. Une anomalie n'est définissable que par opposition à ce qui est normal xii, 4, 10, 20, 22–24, 58, 74, 78, 80, 88, 89, 102, 103, 126, 130–133, 136, 139

apprentissage auto-supervisé L'apprentissage auto-supervisé est souvent considéré comme un entre-deux entre l'apprentissage supervisé et l'apprentissage non supervisé. Bien qu'il ne repose pas sur l'utilisation d'annotations résultant d'une annotation manuelle des données mais sur des pseudos-annotations, générées automatiquement à partir des données 4, 10, 20, 39, 40, 43, 45–48, 55, 58–60, 91, 102, 126, 127, 130, 135, 138, 162

apprentissage automatique Branche de l'intelligence artificielle cherchant à modéliser un phénomène à partir des données observées grâce à des algorithmes capables d'apprendre. Ces algorithmes fonctionnent la plupart du temps en deux étapes, la première est une phase d'apprentissage pendant laquelle l'algorithme cherche à s'améliorer à réaliser la tâche qui lui a été confiée 5, 13, 15, 26, 36, 51, 62, 104, 130, 137

apprentissage contrastif Méthode d'apprentissage de représentation reposant sur une tâche prétexte contrastive. Cette technique consiste à entrainer un modèle en lui présentant des paires d'exemples similaires et dissemblables, afin d'apprendre à maximiser la similarité dans l'espace latent entre les exemples similaires et à minimiser la similarité entre les exemples dissemblables, permettant ainsi de capturer des représentations significatives des données xii, xiii, xv, 10–12, 47–49, 53–55, 60, 92, 102, 115, 116, 120–123, 125–127, 130, 132

apprentissage de représentation Ensemble de méthodes permettant d'apprendre, à partir d'un jeu de données, une représentation différente, souvent plus adéquat pour la tâche à réaliser xii, 9–11, 13, 20, 39, 55, 58–60, 64, 68, 76, 90, 91, 96, 106, 115, 116, 121–124, 126, 127, 132, 139

apprentissage en environnement ouvert *Open set recognition* en anglais. C'est une sous-catégorie de la tâche de classification automatique qui consiste à détecter des classes qui n'ont pas été vues lors de l'entrainement 138, 139

apprentissage profond Branche de l'apprentissage automatique reposant sur les capacités d'apprentissage d'algorithmes hautement non linéaires tels que les réseaux de neurones profonds xv, 4, 5, 26–28, 39, 46, 51, 52, 97, 101, 137, 162

architecture L'architecture d'un réseau de neurones est définie par de nombreux paramètres comme la topologie des connexions entre les neurones, les couches utilisées ou les fonctions d'activation utilisées viii, ix, xii, xv, 5, 10, 16, 20, 26, 27, 30–36, 38, 51, 55, 57–65, 68, 71, 76, 78, 83, 86, 88, 89, 92–95, 97, 102, 105–108, 112, 113, 115, 118, 120, 124, 126, 127, 131–133, 138, 171, 172

augmentation de données Ensemble des méthodes pour augmenter la quantité de données utilisées lors de l'entrainement. Ces méthodes permettent par ailleurs de rendre les modèles plus robustes xii, xiii, xv, 11, 20, 47, 49, 52–55, 77, 79, 115–117, 119, 121, 123–127, 132, 168

calcul dans le nuage L'informatique dans le nuage consiste à héberger, stocker et traiter les données acquises sur des serveurs de calculs, distants de l'utilisateur ou de l'application considérés xi, 2, 3, 14

calcul à la marge Ensemble de méthodes d'optimisation visant à repousser le traitement des données en périphérie du réseau et non plus dans des data centers par opposition au calcul dans le nuage. L'analyse des données est réalisé au plus proche de la source d'acquisition de ces dernières 2, 3

capteur intelligent Capteur capable de traiter les données brutes acquises par ce type de capteur afin de renvoyer (uniquement ou non) une information d'intérêt à l'utilisateur xi, 2–4, 9, 52, 104, 106, 122, 130, 133

chute de pression type d'événements présent dans les données sismologiques de Mars 101, 103, 105, 123, 126, 132, 136

dust devil phénomènes météorologiques martien, similaires aux tourbillons de poussière ou aux mini-tornades que l'on peut observer sur Terre 101–104, 109, 126

décodeur Composant, matériel ou logiciel permettant de transformer un code en une information. Dans le cadre des réseaux de neurones profonds, les décodeurs permettent de projeter les données de l'espace latent vers l'espace de sortie $\mathbb{Y}$ xi, 14, 26, 27, 76, 106, 107, 112

détection d'anomalies Tâche qui isole les anomalies du reste des données viii, xii, xv, 4, 5, 7, 9, 10, 12, 20–23, 30, 42, 49, 55, 57–61, 64, 65, 67, 71, 75–77, 81, 83, 87–89, 92, 95–97, 102, 112, 120–124, 126, 130, 131, 133, 134, 138, 139, 165, 186

encodeur Composant, matériel ou logiciel permettant de transformer une information en un code. Dans le cadre des réseaux de neurones profonds, les encodeurs permettent de projeter les données d'entrées $\mathbb{X}$ dans un espace latent xi, 14, 26, 27, 76, 106, 107, 115, 119, 121

espace latent Espace permettant de compresser l'information des données. Cet espace permet d'extraire les variables latentes ou *cachées* des données. Les données peuvent être projétes dans un espace latent à l'aide d'un encodeur xiii, 13, 26, 61, 90, 91, 115, 116, 119–123, 132

exactitude De l'anglais *accuracy*, C'est le rapport du nombre de classifications justes sur toutes les classifications. Elle prend donc en compte aussi bien les vrais positifs que les vrais négatifs. Elle mesure, parmi toutes les détections faites, combien sont justes 14, 23, 49, 123

exactitude équilibrée Version modifiée de l'exactitude qui prend en compte un fort déséquilibre dans les données et le compense. xiii, 23, 123, 125, 181–183

feature Ensemble des caractéristiques liées à un échantillon xiii, 16, 22, 26, 33, 61, 89, 91, 102, 106, 116, 120, 123, 125, 126, 167, 181–183

fonction d'activation Fonction permettant d'introduire des non linéarités dans les réseaux de neurones. Les principales sont `reLu`, `tanh`, `sigmoid` ou `softmax` mais d'autres fonctions peuvent être utilisées xv, 13, 28, 29, 31, 115

fonction de cout Fonction cible permettant de réaliser l'optimisation des réseaux de neurones 26, 27, 48, 115, 118, 162

glitch Altération temporaire des données martiennes, notamment à cause des températures extrêmes subies par le lander 101–105, 123, 126, 132, 136

hyperparamètre Ensemble des valeurs des réseaux de neurones à déterminer en amont de l'entrainement comme le nombre de couches N, leur taille u^n ou encore les fonctions d'activation à utiliser viii, xii, xvi, 5, 10, 15, 20, 26, 29, 30, 32, 33, 35–38, 51, 55, 57, 58, 62, 63, 65, 66, 71, 76, 93, 107, 108, 115, 131, 132, 139, 162, 168

intelligence artificielle Ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine ix, 39

internet des objets L'internet des objets est défini comme l'interconnexion d'objets physiques à un réseau afin de disposer de services évolués. Les objets peuvent prendre des formes diverses telles que des objets du quotidien ou tout type de capteur, connectés à un réseau par n'importe quel type de communication (filaire, Bluetooth, Wi-Fi, 5G, etc.) xi, 3, 15

jeu de données Constitue l'ensemble des données prêtes à être utilisées par un algorithme d'apprentissage automatique. Ces données ont été prétraitées avant : c'est le passage d'acquisitions à un jeu de données xii, xiii, xv, xvi, 9, 11–13, 15, 20, 22, 23, 27, 30, 43, 44, 47–49, 51–53, 55, 59–61, 65, 68, 71, 74–77, 79–81, 83–90, 92–94, 96, 97, 102–105, 108–112, 115, 116, 121–123, 126, 127, 131, 132, 135–139, 184

masse sèche Masse de la cellule si elle avait été privée de toute son eau. C'est une image du contenu génétique de la cellule xii, xv, 6, 7, 11, 68–78, 81, 85, 88, 92, 94, 96, 131, 136

microscopie sans lentille Méthode d'imagerie biologique en rupture permettant de capturer des images contenant des milliers de cellules 6, 68

objet connecté On parle d'objet connecté pour désigner un objet ordinaire tel qu'une montre, un bracelet, un jouet, un robot de cuisine, une porte de garage etc...capable de communiquer des informations diverses à un autre objet ou à internet. Un objet connecté est la conjugaison d'un *objet physique* et d'une *intelligence virtuelle associée* 2

pas d'apprentissage *learning rate* en anglais. C'est le pas avec lequel un algorithme d'optimisation se déplace vers un minimum local 107

précision Aussi appelée *taux de vrai positif*. Dans le cas d'une classification (binaire), la précision est le rapport du nombre d'événements classés comme positifs par rapport au nombre d'événements effectivement positifs dans le jeu de données. Formulé autrement, la précision mesure, parmi les éléments classés comme anormaux, combien sont réellement anormaux ? 20, 23, 25, 79, 84, 85, 88, 89, 102

pseudo-label annotation associée à une tâche prétexte pouvant être générée automatiquement à partir des données afin de réaliser un apprentissage supervisé d'une représentation avec des données non annotées xi, 40, 43–46, 130

rappel C'est le nombre d'anomalies détectées sur le nombre d'anomalie total. Il représente le taux d'anomalies que l'on a été capable de détecter parmi toutes les anomalies 20, 23, 25, 75, 88, 89, 102

recherche d'architecture neuronale Ensemble des méthodes permettant de définir l'architecture mais aussi les hyperparamètres du réseau à utiliser pour résoudre une tâche donnée 20, 36–38, 55, 58, 65

représentation c'est la manière dont les données sont transformées afin d'être comprises par un algorithme. Une *bonne* représentation capture les caractéristiques pertinentes des données, facilitant ainsi la découverte de motifs et la prise de décisions xiii, xvi, 10, 13, 15, 20, 26, 33, 34, 39, 43, 46, 48, 49, 59–61, 68, 71, 76, 86, 88, 91, 96, 102, 113, 115, 116, 118–120, 122–124, 126, 127, 130–133, 138, 139, 181

réseau de neurones Abus de langage faisant référence aux réseaux de neurones profonds vii, ix, xi, xv, 5, 6, 11, 13, 16, 19, 20, 26–36, 39, 43–45, 47, 51, 52, 55, 58–62, 65, 68, 71, 76, 81, 87, 96, 103, 106–113, 115, 119, 126, 127, 130–136, 171, 172

réseau de neurones convolutif Architecture de réseaux de neurones reposant sur des opérations de convolutions entre chaque couche 30

réseau de neurones profond Architectures de réseaux de neurones possédant au moins une couche cachée entre l'entrée et la sortie. Aujourd'hui, l'ensemble des réseaux de neurones sont en fait des réseaux de neurones profonds. xi, 5, 10, 14–16, 26, 27, 29, 30, 36, 52, 59, 74, 92, 93, 103

réseau de neurones récurrent Type de réseau de neurones profond contenant une ou plusieurs boucles de rétroaction 31, 35, 60

sol de "jour solaire", correspond à un jour sur Mars. C'est la durée qui s'écoule pour que le soleil retourne au même point dans le ciel. Un sol martien correspond en moyenne à 24 heures, 39 minutes et 35 secondes. Le premier sol est défini par le jour de l'atterrissage de la mission concernée xiii, 103, 105

sous-échantillonnage par valeurs maximales Couche de réseau de neurones utilisée pour diminuer la taille des features tout en conservant la cohérence spatiale ou temporelle de ces dernières 30, 31

sur-apprentissage Le sur-apprentissage, de l'anglais *overfiting*, se produit lorsque le modèle d'apprentissage automatique s'est adapté si étroitement aux données d'entrainement qu'il perd sa capacité à généraliser efficacement à de nouvelles données 35, 51, 59

série temporelle Suite de valeurs (numériques ou non) décrivant l'évolution d'une quantité physique au cours du temps. Généralement l'étude des séries temporelles consiste à analyser le comportement passé afin d'en déduire le comportement futur ix, xii, xiii, xvi, 2, 4, 6, 7, 9, 10, 20, 30, 32, 35, 43, 46–49, 51–53, 57, 59–62, 65, 68, 70, 71, 74, 75, 77–83, 85, 89–92, 96, 97, 102, 105–107, 112, 113, 116, 122, 123, 127, 129–132, 136, 137, 168, 171, 172, 174–179

tâche prétexte Tâche qui permet d'apprendre une représentation des données mais qui ne correspond pas à la tâche que l'on souhaite réaliser au final viii, ix, xi, xiii, 10–12, 15, 39, 40, 42–47, 55, 57, 59, 60, 64, 65, 68, 71, 74, 76, 83, 87, 89, 91, 92, 97, 102, 106, 108, 112, 113, 115, 116, 119, 121–127, 130–132, 135, 171, 172

vision par ordinateur Ensemble de méthodes algorithmiques visant à traiter des images afin d'en comprendre leur contenu. La vision par ordinateur comprend un grand nombre de tâches, dont entre autres la segmentation ou la classification 30, 33–35, 43, 79, 137, 138

web 3.0 Aussi appelé web sémantique. Il est défini par le W3C [HHA+] comme un format permettant d'agréger des données de sources diverses afin qu'elles puissent être réutilisées là où le web des objets se concentre sur l'échange de documents. C'est une interconnexion entre des objets physiques et les données y étant reliées 2

H Acronymes

ACI Analyse en Composantes Indépendantes 50, 89

ACP Analyse en Composantes Principales 49, 50, 89

CNN réseau de neurones convolutif 30, 32, 35, Glossaire : réseau de neurones convolutif

CNN 1D CNN à une dimension 58, 59, 61, 62, 68, 76, 83, 86, 89, 90, 92, 97, 102, 106, 107, 115, 116, 126, 131–133, 164, 168

DEL Diode ÉlectroLuminescente 6, 69

EEG Electroencéphalogramme 30

EQM Erreur Quadratique Moyenne xii, xvi, 61, 62, 77, 80, 81, 83, 93–97, 109, 110, 114, 131, 132, 172

FOV Champ de vision 6, 7, 68, 69

IA Intelligence Artificielle ix, 3, 4, 6, 9, 10, 39, 80, 129, 130, 133–135, 137–139, Glossaire : intelligence artificielle

LIDAR Laser Imaging Detection and Ranging 2

LMST Local Mean Solar Time 103, 104

LOF Local Outlier Factor 21, 123, 126, 132

LSTM Long short Term Memory xi, 31, 32, 35, 94, 95, 113, 164

OCSVM Machine à Vecteurs Supports (SVM) à une classe 21, 24, 61, 123, 126, 132

PR Précision Rappel xiii, 25, 89, 90

ROC Receiver Operating Characteristic xiii, 89, 90

RVB Rouge Vert Bleu 69

StArDusTS Self-supervised Anomaly Detection on Time Series xii, 10–12, 58–60, 64, 68, 71, 74, 76, 77, 79, 80, 83, 85, 86, 88–92, 96, 97, 102–104, 106, 115, 121–124, 126, 127, 130–133, 139

SVM Machine à Vecteurs Supports 17, 21, 39

TALN Traitement Automatique de la Langue Naturelle 34, 35, 46, 47, 137

TDNN Réseau de neurones à délai d'attente 30

UMAP Uniform Manifold Approximation and Projection for Dimension Reduction xiii, 120–122

CHAPITRE

II

État de l'art

Table des matières

Introduction

Cette section présente un état de l'art relatif aux travaux effectués lors de cette thèse. Les méthodes de détection d'anomalies utilisées dans le reste du document sont présentées section A. Nous présentons ensuite section B un rappel sur les réseaux de neurones et les différentes architectures. Le choix des hyperparamètres de ces architecture peut être fait grâce à des méthodes de recherche d'architecture neuronale présentées section C.

Nous nous concentrons après section D sur l'apprentissage de représentation et en particulier l'apprentissage auto-supervisé, permettant d'apprendre une représentation sans pour autant disposer d'annotations des données. Enfin nous présentons la manière dont seront gérés les jeux de données dans cette étude et les méthodes d'augmentation de données spécifiques aux séries temporelles section E.

A Méthodes de détection d'anomalies

La détection d'anomalies correspond à une identification d'éléments ou d'événements déviants par rapport à une ou des normes définies en fonction du cadre applicatif concerné. Dans cette section, nous présentons d'abord section A.1 les détecteurs utilisés dans la suite de ce manuscrit avant de présenter les métriques permettant d'évaluer des détecteurs d'anomalies section A.2.

A.1 Détecteurs

Cette section présente quelques détecteurs de l'état de l'art, utilisés pour la détection d'anomalies. Il est important de garder à l'esprit que, dans le contexte de cette thèse, les annotations des anomalies ne sont pas disponibles. Une distinction est alors faite entre les méthodes supervisées, nécessitant les annotations associées aux données, les méthodes non supervisées ne nécessitant aucune annotation et les méthodes semi-supervisées nécessitant seulement *une partie* des annotations comme un ensemble de données normales par exemple. Dans cette étude, nous nous restreindrons aux méthodes non supervisées et semi-supervisées.

Le tableau II.1 (page 24) synthétise les avantages, les inconvénients ainsi que les contraintes liées à chacun de ces algorithmes.

Détecteur à seuil Un détecteur à seuil [WR78, Wes78, SSW88] est un outil fondamental utilisé dans l'analyse de données temporelles pour identifier des comportements anormaux. Son principe repose sur l'établissement d'une valeur limite, appelée seuil, au-delà de laquelle les observations sont considérées comme inhabituelles ou déviantes par rapport au modèle attendu. Lorsque les données dépassent ce seuil, le détecteur signale la présence potentielle d'une anomalie. En d'autres termes, le détecteur surveille en permanence les données et compare chaque nouvelle observation à cette limite prédéfinie. Si la valeur observée dépasse le seuil défini, une anomalie est détectée.

D'un côté, la définition d'un seuil bas permet de ne manquer aucune anomalie et donc d'avoir un très bon rappel. De l'autre, en choisissant un seuil élevé, très peu d'éléments sont détectés comme anormaux, mais leur détection est bonne (VP), menant à une très bonne précision. C'est un compromis entre les deux qu'il faut réussir à trouver. Cet équilibre est alors constitué de peu d'éléments anormaux manqués (un bon rappel) et peu de faux positifs parmi les éléments détectés (une bonne précision).

Machine à Vecteur Support (SVM) à une classe (OCSVM) Les SVMs [BGV92, CV95] sont une classe d'algorithmes d'apprentissage automatique utilisés pour la classification et la régression. Les SVMs cherchent un hyperplan qui sépare au mieux les données en fonction de leurs classes ou de leurs valeurs cibles. Cet hyperplan est choisi de manière à maximiser la marge, c'est-à-dire la distance entre les exemples d'entrainement les plus proches de chaque classe. En d'autres termes, les SVMs cherchent à trouver l'hyperplan qui présente la plus grande marge possible entre les différentes classes, ce qui les rend robustes face aux données bruitées ou aux cas marginaux. Pour les problèmes de classification non linéaire, les SVMs utilisent des techniques de transformation de l'espace pour projeter les données dans un espace de dimension supérieure où une séparation linéaire devient possible. La méthode de projection est alors qualifiée de fonction noyau. [CGRL20] présentent dans leur étude de nombreuses applications des SVMs tels que la catégorisation de textes, reconnaissance d'images [CB21], ou encore le contrôle prédictif.

Les SVM à une classe (SVM) [SWS+99], sont une variation spécifique des SVMs conçues pour traiter les problèmes de détection d'anomalies. Contrairement aux SVMs classiques qui effectuent une classification binaire en séparant deux classes distinctes, les OCSVMs visent à modéliser et à détecter la distribution des données d'une seule classe. L'objectif principal est d'identifier les observations qui sont très différentes ou inhabituelles par rapport au modèle attendu de la classe.

Dans les OCSVMs, l'algorithme tente de trouver une hypersphère qui capture au mieux la majorité des données de la classe cible, tout en maintenant une marge maximale par rapport aux données marginales ou aberrantes. La distance entre les données et l'hyperplan est utilisée pour évaluer l'ajustement des observations à la distribution de la classe. Les données qui se trouvent en dehors de cette marge sont considérées comme des anomalies.

Les OCSVM sont utiles dans les situations où les exemples d'anomalies sont rares ou difficiles à obtenir pour l'entrainement, mais où les exemples normaux sont disponibles en abondance. Elles sont également efficaces pour la détection d'anomalies dans des espaces de grande dimension, où il peut être difficile de définir des frontières de décision précises. [ARS+20] exposent de nombreux domaines applicatifs des OCSVMs telles que la classification de documents, des applications liées au domaine audio, vidéos, ou à la détection de faute pour la maintenance prédictive.

Local Outlier Factor (LOF) Le Facteur Local d'Anomalie mesure la déviation de densité locale d'un élément par rapport à la densité locale de ses voisins [BKNS00].

Grâce à son aspect local, le LOF permet de détecter un unique point à faible distance d'un regroupement à haute densité de la même manière qu'un point très loin d'un regroupement à faible densité.

L'un des inconvénients est qu'il est difficile de définir un seuil pour lequel le LOF doit considérer le point comme anormal. Autrement dit, à partir de quelle différence de densité ce point est considéré anormal. Par ailleurs il a tendance à mal s'adapter à des données à haute dimension notamment à cause du fléau de la dimension * (voir remarque A.1).

[AASM21] propose une review extensive des méthodes LOF, toutes leurs variantes ainsi que des domaines applicatifs qui utilisent cette méthode.

*. Curse of dimensionality

Remarque 2 : Le fléau de la dimension

Le fléau de la dimension [Bel66], de l'anglais *curse of dimensionality*, est un phénomène problématique qui se produit lorsqu'on travaille avec des ensembles de données de très grande dimension. À mesure que le nombre de dimensions (variables) dans un ensemble de données augmente, les points dans cet espace se retrouvent rapidement isolés, entrainant des défis significatifs dans l'analyse et le traitement de ces données.

Il est possible d'illustrer cet éloignement avec un exemple. Afin d'échantillonner régulièrement l'intervalle $[0,1] \in \mathbb{R}$ avec une distance maximale entre deux points de 10^{-2}, seulement $10^2 = 100$ points sont nécessaires. Si l'on cherche à réaliser la même grille pour un hypercube $[0,1]^{10}$ sce sont alors $10^{2^{10}} = 10^{20}$ points qui sont nécessaires. À l'inverse si l'on considère seulement 100 points dans cet espace de dimension 10, ces derniers sont principalement entourés de vide.

En outre, la malédiction de la dimension peut entrainer une augmentation de la complexité computationnelle. Les calculs nécessaires pour analyser et traiter les données deviennent plus intensifs en termes de temps et de ressources, ce qui peut rendre les méthodes d'analyse et d'apprentissage automatique beaucoup plus lentes et moins pratiques.

Isolation Forest Pour introduire le concept de l'Isolation Forest, il est important de commencer par comprendre ce qu'est une Random Forest. Une Random Forest [Ho95, Ho98] est un algorithme d'apprentissage automatique qui appartient à la famille des méthodes ensemblistes. Elle combine plusieurs modèles d'arbres de décision [RM05] pour effectuer des tâches telles que la classification et la régression. Chaque arbre de décision est formé sur un sous-ensemble aléatoire des données d'entrainement et utilise un sous-ensemble aléatoire des features des données. Ensuite, les prédictions de chaque arbre individuel sont agrégées pour obtenir la prédiction finale de la Random Forest. Attention cependant, les Random Forest au sens large ne sont pas uniquement des méthodes de détection d'anomalies à proprement parler, contrairement aux Isolation Forest.

L'Isolation Forest [LTZ08, LTZ12] est une technique similaire, qui vise à identifier rapidement les anomalies en les isolant du reste des données normales. Afin d'isoler un point, l'algorithme sépare aléatoirement l'espace en 2 sur un attribut sélectionné aléatoirement lui aussi.

Il construit alors un *arbre isolateur* de manière récursive en réalisant des coupes aléatoires à chaque branche qui n'est pas une feuille jusqu'à ce que ce que tous les points du jeu de données soient isolés sur une feuille de l'arbre. L'hypothèse est alors qu'il faudra en moyenne moins de coupes pour isoler une donnée anormale. On calcule alors le nombre moyen de coupes nécessaires à l'isolation d'un point du jeu de données en utilisant un grand nombre d'arbres isolateurs.

L'avantage de cet algorithme est qu'il peut être utilisé de manière totalement non supervisée et fonctionne même en haute dimension.

A.2 Méthodes d'évaluation

Avant de présenter les différentes méthodes de détection d'anomalies, concentrons-nous sur les métriques permettant d'évaluer et de comparer ces modèles. Pour évaluer la détection d'anomalies, on peut se baser sur la matrice de confusion de la classification tableau II.2. Les bonnes classifications sont les événements détectés comme des anomalies (vrais positifs) ainsi que les éléments normaux, classifiés comme normaux (vrais négatifs).

La détection d'anomalies est évaluée selon 3 critères principaux :

La précision (eq. II.1) aussi appelée taux de vrai positif. C'est la mesure, parmi les éléments classés comme anormaux, de combien sont effectivement anormaux.

$$\text{Précision} = \frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux positifs}} \tag{II.1}$$

Le rappel (eq. II.2) quant à lui mesure le nombre d'anomalies détectées, comparé au nombre total d'anomalies. Il représente le taux d'anomalies que l'on a été capable de détecter parmi toutes les anomalies.

$$\text{Rappel} = \frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux négatifs}} \tag{II.2}$$

L'exactitude * (eq. II.3), C'est le rapport du nombre de classifications justes sur toutes les classifications. Elle prend donc en compte aussi bien les vrais positifs que les vrais négatifs. Elle mesure, parmi toutes les détections faites, combien sont justes

$$\text{Exactitude} = \frac{\text{Vrais positifs} + \text{Vrais négatifs}}{\text{Total}} \tag{II.3}$$

L'exactitude équilibrée (eq. II.4) L'exactitude mesure le pourcentage total de prédictions correctes sur l'ensemble du jeu de données, qu'elles soient normales ou anormales. Lorsque le jeu de données est fortement déséquilibré avec une grande majorité d'éléments normaux, l'exactitude peut être trompeuse et donner l'impression que le modèle est très performant, car il prédit correctement la classe majoritaire (normale) dans la plupart des cas. Cependant, cela peut masquer le fait que le modèle ne détecte pas efficacement les éléments anormaux, ce qui est souvent l'objectif principal dans la détection d'anomalies.

L'exactitude équilibrée prend en compte la distribution des classes dans le jeu de données. Elle calcule la moyenne des taux de vrais positifs (éléments anormaux correctement détectés) et des taux de vrais négatifs (éléments normaux correctement identifiés). Cela équilibre l'impact des classes déséquilibrées et fournit une mesure plus juste de la performance du modèle, en donnant autant d'importance aux prédictions correctes pour les deux classes (normale et anormale).

$$\begin{aligned}
\text{Exactitude équilibrée} &= \frac{1}{2}\left(\text{Sensibilité} + \text{Spécificité}\right) && \text{(II.4)}\\
&= \frac{1}{2}\left(\text{Taux de vrais positifs} + \text{Taux de vrais négatifs}\right) && \text{(II.5)}\\
&= \frac{1}{2}\left(\frac{\text{VP}}{\text{VP} + \text{FN}} + \frac{\text{VN}}{\text{VN} + \text{FP}}\right) && \text{(II.6)}
\end{aligned}$$

Table II.1 : Comparaison des méthodes de détection d'anomalie

Méthode	Avantages	Inconvénients	Type de données appropriées
Seuillage	- Ne nécessite pas d'hypothèse de distribution	- Choix du seuil difficile, - Nécessite le choix d'une valeur à seuiller	- Données peu bruitées - Données avec des anomalies globales.
LOF	- Peut détecter des anomalies locales et globales. - Fonctionne bien avec des données de haute dimension. - Évite les hypothèses de distribution de données.	- Sensible aux paramètres de distance et de voisinage. - Peut être couteux en temps de calcul pour de grandes quantités de données.	- Données à haute dimension. - Données avec des anomalies locales.
OCSVM	- Peut détecter des anomalies globales. - Peut être efficace même avec peu de données d'entrainement. - Ne nécessite pas d'hypothèse de distribution.	- La qualité de la détection dépend fortement des paramètres du modèle, comme la marge et le noyau. - Peut être sensible aux valeurs aberrantes dans les données d'entrainement. - Peut être couteux en temps de calcul pour de grandes quantités de données.	- Données de faible ou moyenne dimension. - Données avec des anomalies globales.
Isolation Forest	- Peut détecter des anomalies globales. - Peut être efficace même avec peu de données d'entrainement. - Peut gérer les données avec des valeurs manquantes. - Le calcul de chaque arbre est parallélisable. - N'a pas besoin d'hypothèses de distribution.	- Peut être couteux en temps de calcul pour de grandes quantités de données. - Peut être peu efficace avec des données qui ont peu de variabilité. - Les résultats peuvent être influencés par les paramètres du modèle, tels que la profondeur de l'arbre et le nombre d'arbres.	- Données de faible, moyenne ou haute dimension. - Données avec des anomalies globales.

Table II.2 : Matrice de confusion pour une classification a deux classes normal/anormal

		Valeur Prédite	
		Anormal	Normal
Vraie valeur	Anormal	Vrai positif	Faux négatif
	Normal	Faux positif	Vrai négatif

Ces trois métriques mesurent bien des quantités différentes et peuvent toutes être intéressantes à maximiser en fonction du cas applicatif traité. C'est souvent un compromis entre une précision élevée et un rappel élevé qu'il faut trouver.

Lorsqu'un détecteur fonctionne à l'aide d'un seuil quelconque, faire varier ce seuil engendre des détections différentes et donc des performances différentes. Par exemple, augmenter le seuil de détection peut permettre d'être plus précis, au risque de manquer certaines anomalies et donc d'avoir un rappel plus faible. Des méthodes existent pour pouvoir se défaire de la valeur de seuil lors de la comparaison de plusieurs modèles. La courbe Précision Rappel (PR) correspond à l'ensemble des valeurs précision, rappel calculées pour différentes valeurs de seuil. L'aire sous la courbe PR permet d'intégrer la variabilité de la précision et du rappel en fonction du seuil et de pouvoir comparer plusieurs modèles avec une unique valeur. Cette dernière est parfois appelée score de précision moyen [GG05].

B Réseaux de neurones et architectures

Le cœur de cette thèse repose sur l'utilisation de réseaux de neurones comme méthodes algorithmiques pour apprendre une représentation des données. Nous cherchons en effet à bénéficier de la capacité qu'ont les réseaux de neurones à extraire automatiquement des features appropriées à la résolution de la tâche qui leur incombe.

Afin de mieux les comprendre, nous présentons section B.1 les concepts clés des réseaux de neurones. Nous détaillons ensuite sections B.2 à B.6 différentes architectures proposées dans la littérature ainsi que leurs avantages et inconvénients. Cette section repose en partie sur les livres d'apprentissage automatique et d'apprentissage profond [HTF09] et [GBC16].

Comme pour la section précédente, un tableau synthétisant les différentes architectures de réseaux de neurones ainsi que leurs avantages et leurs inconvénients est disponible à la fin de la section (tableau II.4 page 35).

B.1 Introduction aux réseaux de neurones

Il est possible de décrire les réseaux de neurones profonds comme un ensemble de deux blocs distincts. Un premier nommé encodeur permettant de projeter les données X dans un espace latent décrivant des propriétés initialement *cachées* de ces données. Les données X projetées dans l'espace latent sont usuellement notées Z. Les données dans l'espace latent sont ensuite présentées en entrée du second bloc : le décodeur. Son rôle, est de réaliser une tâche d'intérêt comme de la classification [Den12, RDS+15] de la régression [MHWR91, S+91]. Comme montré sur la figure II.1, le décodeur est le bloc qui permet de passer de la représentation latente Z aux données à prédire Y. Ces dernières peuvent prendre plusieurs formes en fonction de la tâche que l'on cherche à accomplir, si c'est une régression, Y peut être une unique valeur, ou encore dans le cas d'une classification, un identifiant de classe associé à l'entrée X.

La manière dont sont traitées les données au sein de l'encodeur et du décodeur constitue l'architecture du réseau de neurones utilisée. C'est cette architecture qui va définir la complexité du réseau de neurones ainsi que sa capacité à répondre à des problèmes plus ou moins complexes. Les hyperparamètres architecturaux sont l'ensemble des valeurs qui permettent de définir cette architecture.

Cependant, une architecture bien conçue n'est que le premier pas vers la création d'un réseau de neurones performant. Au-delà des connexions complexes et de la disposition des couches, un autre élément crucial entre en jeu : l'algorithme d'apprentissage. Si l'architecture d'un réseau définit son potentiel, c'est l'algorithme d'apprentissage qui permet de le réaliser. Les architectures sophistiquées ne peuvent exploiter leur plein potentiel sans une stratégie d'apprentissage pertinente et bien ajustée. L'algorithme d'apprentissage agit en tant que pilier fondamental, guérissant le modèle de ses erreurs et ajustant les poids des connexions neuronales à partir des données d'entrainement. Cet algorithme dispose lui-même de paramètres qui peuvent affecter la vitesse de convergence (et même simplement la convergence) d'une architecture. Ces paramètres sont nommés hyperparamètres mais afin de les distinguer des hyperparamètres architecturaux, ils sont parfois nommés hyperparamètres d'entrainement.

Au cœur des algorithmes d'apprentissage pour les réseaux de neurones se trouve la méthode de la descente du gradient. Cette technique repose sur l'optimisation itérative des poids du réseau en minimisant une fonction de cout, représentant généralement l'écart entre les prédictions du modèle et les valeurs cibles attendues. Une contrainte forte sur la fonction de cout utilisée est que cette dernière doit être dérivable, ce n'est donc pas tou-

jours la métrique que l'on cherche à optimiser qui est utilisée comme fonction de cout mais parfois des alternatives. On fait donc la différence entre la métrique, qui fait référence à la valeur que l'on cherche vraiment à minimiser (ou maximiser en fonction des métriques) et la fonction de cout qui est utilisée pour entrainer un réseau de neurones. La descente du gradient ajuste progressivement les poids en calculant la direction dans laquelle la fonction de cout décroît le plus rapidement. L'optimisateur [KB15] (Adaptive Moment Estimation) est une amélioration notable de la descente du gradient standard. ADAM permet de calculer un taux d'apprentissage adaptatif par paramètre, ce qui facilite la convergence plus rapide et robuste des réseaux de neurones. De plus, ADAM intègre des mécanismes de correction de biais et d'adaptation aux moments du gradient pour éviter les variations excessives des taux d'apprentissage, ce qui lui confère une efficacité accrue. C'est aujourd'hui l'optimisateur qui fait consensus au sein de la communauté scientifique [Pos20, CSN+20].

Remarque 3 : Émergence des réseaux de neurones profonds

Bien qu'il soit difficile d'attribuer la paternité des réseaux de neurones à un unique article, on voit émerger vers la fin des années 80 [Wai89, LBD+89] les premières utilisations concrètes de l'apprentissage profond. Il faudra cependant attendre deux avancées pour voir l'utilisation des réseaux de neurones se démocratiser :

— La mise à disposition auprès du grand public de jeux de données assez conséquents pour permettre une diversité suffisante dans les données. Avec par exemple [RDS+15],

— L'émergence de puissances de calcul bien supérieures avec la démocratisation dans les années 2010 des cartes graphiques, aussi appelées GPU pour Graphics Processing Unit, auprès du grand public et par la même occasion de la communauté de l'apprentissage profond. Le premier article mettant en avant l'utilisation de GPU montre en 2009 un gain de temps grâce à celui-ci d'un facteur 72.6, passant d'un entrainement de 2.5 jours à 29 minutes [RMN09].

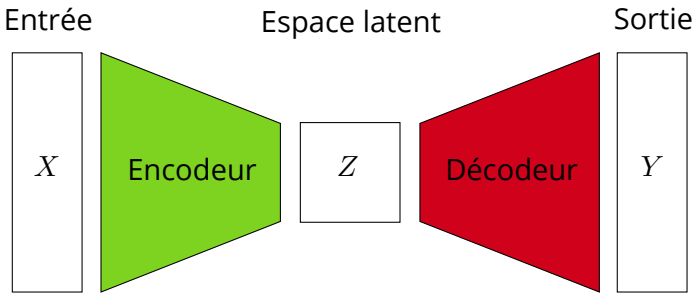


Figure II.1 : Schéma d'un réseau de neurones avec un encodeur et un décodeur

Nous présentons dans les sections suivantes un ensemble d'architectures de réseaux de neurones utilisées dans la littérature. Ces architectures sont ordonnées de la plus simple à la plus complexe. Cette section n'a pas pour vocation d'être une liste exhaustive des architectures de réseaux de neurones, mais permet de comprendre les architectures utilisées dans ce manuscrit.

B.2 Perceptrons Multicouches

La figure II.2 schématise un unique neurone : le neurone artificiel numéro j de la couche (n) nommé $\Sigma_j^{(n)}$. Ce dernier dispose d'une entrée $X^{(n-1)}$ (eq. II.7), constituée des $u^{(n-1)}$ sorties $\left[x_0^{(n-1)}, \dots, x_{u^{(n-1)}-1}^{(n-1)}\right]$ de la couche précédente. Chacune des valeurs de cette entrée

est pondérée par un poids $w_{i,j}^{(n)}$ correspondant au i^{eme} poids du neurone j de la couche (n). Ces poids peuvent être réunis en un vecteur $W_j^{(n)}$ (eq. II.8)

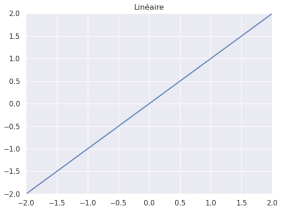
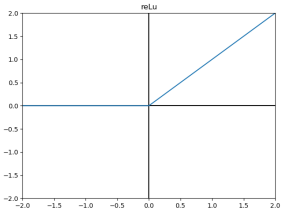
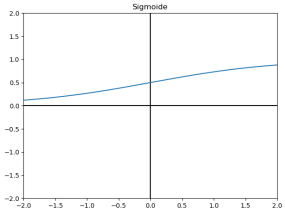
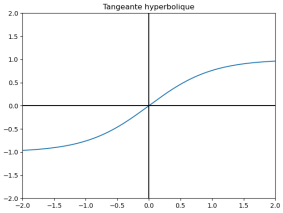
La sortie $x_j^{(n)}$ produite par ce neurone j de la couche n dépend donc de ses poids $W_{j^{(n)}}$ et de ses entrées $X^{(n-1)}$ comme le montre l'équation (II.9). L'opération réalisée par le neurone figure II.2 est une *simple* combinaison linéaire des entrées de ce dernier. Cependant, les neurones sont souvent suivis d'une fonction d'activation φ permettant d'introduire une non-linéarité entre son entrée et sa sortie [GBC16]. Cette non-linéarité permet de mimer le comportement d'un neurone humain qui accumule des charges sur ses synapses jusqu'à ce que la charge soit assez grande pour dépasser son seuil d'activation. Le tableau II.3 présente 3 fonctions d'activation : La rectified Linear Unit ou reLu, la sigmoide et la tangente hyperbolique.

$$X^{(n-1)} = \left[x_0^{(n-1)}, x_1^{(n-1)}, \dots, x_{u^{(n-1)}-1}^{(n-1)}\right]^T \tag{II.7}$$

$$W_j^{(n)} = \left[w_{0,j}^{(n)}, w_{1,j}^{(n)}, \dots, w_{u^{(n-1)}-1,j}^{(n)}\right]^T \tag{II.8}$$

$$x_j^{(n)} = \varphi\left(W_j^{(n)^T} \cdot X^{(n-1)}\right) = \varphi\left(\sum_{i=0}^{u^{(n-1)}-1} w_{i,j}^{(n)} \cdot x_i^{(n-1)}\right) \tag{II.9}$$

Table II.3 : Quelques exemples de fonctions d'activation utilisées classiquement en apprentissage profond

Linéaire	reLu	Sigmoide	tanh
			
$f(x) = x$	$f(x) = \begin{cases} 0 & \text{si } x <= 0 \\ x & \text{sinon} \end{cases}$	$f(x) = \frac{1}{1+e^{-x}}$	$f(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Un réseau de neurones correspond à la mise en parallèle de neurones uniques dans des couches distinctes puis à *empiler* différentes couches afin de concevoir un réseau profond comme montré sur la figure II.3. En pratique, les poids $[W_0^{(n)}, \dots, W_{u^{(n)}-1}^{(n)}]$ des neurones 0 à $u^{(n)}-1$ peuvent être concaténés dans une matrice $W^{(n)}$ des poids des neurones de la couche (n). Ainsi les sorties de la couche n peuvent être calculées comme le produit de la matrice des poids de la couche (n) $W^{(n)}$ et du vecteur des entrées $X^{(n-1)}$.

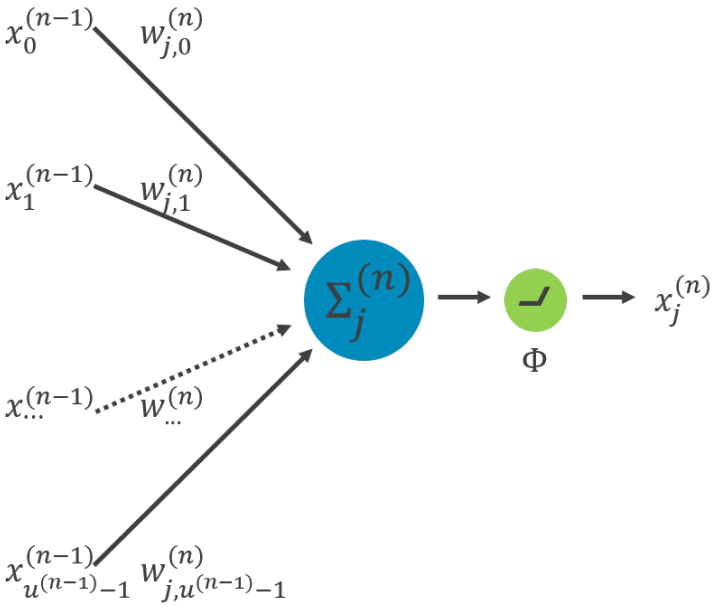


Figure II.2 : Schéma d'un neurone. Il prend en entrée toutes les valeurs provenant de la couche précédente $X^{(n-1)}$ et les pondère respectivement par chacun de ses poids $W^{(n)}$

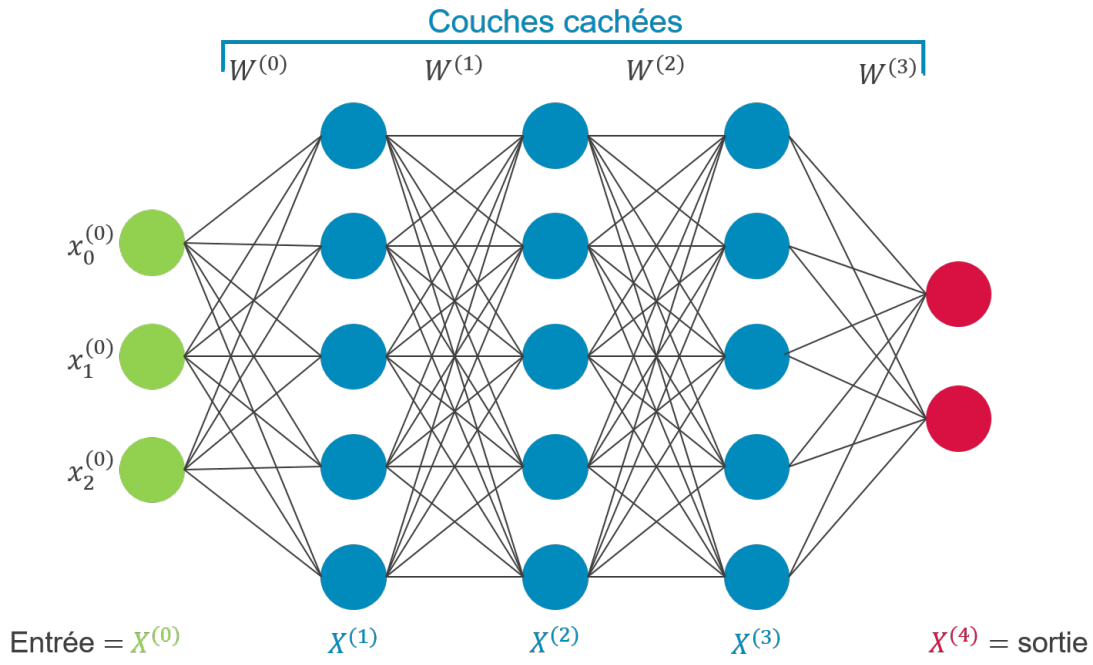


Figure II.3 : Schéma d'un réseau de neurones profond complètement connecté aussi appelé perceptron multicouches

Hyperparamètres des Perceptrons Multicouches : Les hyperparamètres principaux permettant de définir l'architecture des réseaux de neurones denses sont donc :

(i) Le nombre de couches cachées

(ii) Le nombre de neurones dans chaque couche

(iii) Les fonctions d'activation de chaque couche

Remarque 4 : Théorème d'approximation universelle

Le théorème de Cybenko [Cyb89] montre que les Perceptrons multicouches peuvent (sous certaines conditions) approximer n'importe quelle fonction et donc être utilisés pour modéliser n'importe quelle tâche de régression ou de classification.

En particulier, le théorème énonce qu'un réseau dense d'une seule couche cachée contenant un nombre fini de neurones (c'est-à-dire, un perceptron multicouche) peut approximer des fonctions continues sur des sous-ensembles compacts de $\mathbb{R}^n$

B.3 Réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont des architectures neuronales principalement utilisées pour des tâches de vision par ordinateur. Ces derniers reposent sur l'utilisation de noyaux de convolution, dont les valeurs sont apprises pendant la phase d'apprentissage du réseau de neurones. Ils émergent à la fin des années 80 [LBD+89], mais tombent dans l'oubli à cause de la puissance de calcul nécessaire à leur entrainement. C'est au début des années 2010, grâce à l'émergence de nombreux jeux de données publiquement accessibles [Kri09, RDS+15] que des architectures compétitives de Réseaux de neurones convolutifs s apparaissent [KSH12, SZ15, HZRS16, SVI+15] et surpassent les techniques utilisées jusqu'à lors.

Les CNNs, dis 2D pour la vision par ordinateur, permettent d'effectuer un traitement invariant par translation (*i.e.* le même traitement n'importe où sur l'image). Ainsi, réaliser des convolutions avec des noyaux de taille fixe (par exemple 3×3 pixels) appris, permet d'extraire des propriétés dans les images pour ensuite les classifier. En plus des couches convolutives, les CNNs utilisent aussi des couches de sous-échantillonnage par valeurs maximales qui permettent de diminuer la dimension des données au fur et à mesure de la profondeur du réseau de neurones. Une alternative aux couches de sous-échantillonnage par valeurs maximales est proposée par [vKE+16] en utilisant des convolutions à trous, ce qui permet d'augmenter le champ réceptif des convolutions au fur et à mesure des couches sans pour autant diminuer la taille des entrées. Le pas de dilatation définit le nombre de points à sauter dans le calcul de la convolution.

Remarque 5 : Les CNN et le nombre de poids

L'utilisation de couches convolutives réduit radicalement le nombre de poids $\mathcal{W}$ à apprendre. Pour un réseau de neurones profond tel que AlexNet [KSH12] par exemple, plus de 90% des paramètres à apprendre sont dus aux 3 couches complètement connectées, et donc seulement 10% des poids sont associés aux cinq couches convolutives.

Bien que les Réseau de neurones à délai d'attente (TDNN)* aient été introduits par [Wai89], c'est grâce aux performances obtenues en traitement d'image des CNNs 2D que la communauté en traitement de signal a commencé à regagner de l'intérêt pour les CNNs à une dimension. Ils ont été utilisés pour répondre à de nombreuses problématiques concernant des séries temporelles tels que la classification de signaux d'Electroencéphalogramme (EEG) [KIG17, LZZW17], la détection de pannes ou de défauts [AAK+18, AAKI17, EIK19, IKE+16, KKLK19, ZLP+18] ou la reconnaissance de la parole [vDZ+16, vKE+16]. Mais aussi pour de la prédiction de séries temporelles ou la détection d'anomalies [YWX+18].

Hyperparamètres des CNN :

*. Time Delay Neural Network

(i) Le nombre de couches convolutives $\mathcal{C}$
(ii) La taille des noyaux de convolutions $\mathcal{K}$
(iii) Le pas de convolution * $\mathcal{P}$
(iv) Le type de padding
(v) Le nombre de couches convolutives avant une couche de sous échantillonnage
(vi) Le type de couche de sous échantillonnage
(vii) Les fonctions d'activation utilisées
(viii) L'utilisation de couches de sous-échantillonnage par valeurs maximales (a) de convolution à trou (b)
  (a) Nombre de filtres dans chacune des couches convolutives $\mathcal{F}$
  (b) Le pas de dilatation $\mathcal{DR}$ des couches convolutives

B.4 Long Short Term Memory (LSTM)

Les réseaux de neurones récurrents sont une classe d'architecture de réseaux de neurones conçus spécifiquement pour traiter des données séquentielles. Contrairement aux réseaux de neurones classiques, les réseaux de neurones récurrents introduisent des connexions récurrentes entre les couches du réseau. Ces récurrences leur permettent de conserver une mémoire interne et de prendre en compte le contexte temporel des données. Cette capacité à modéliser des dépendances temporelles a conduit à l'émergence des RNN dans de nombreux domaines [LBE15, YSHZ19], tels que la traduction automatique, la génération de texte, l'analyse de sentiments et la reconnaissance de la parole.

Lors de l'entrainement d'un réseau de neurones la mise à jour des poids est guidée par les valeurs du gradient lors de la phase de rétropropagation. Cependant, plus le réseau de neurones est profond, plus ces gradients sont faibles et tendent vers 0 [HS97]. Cela entraine une mise à jour négligeable des poids dans les couches profondes, rendant difficile l'apprentissage de relations complexes. Ce phénomène, appelé évanescence du gradient, limite la capacité des réseaux très profonds à capturer des dépendances à long terme dans les données séquentielles ou spatiales. C'est ce qui a conduit au développement de techniques telles que les LSTM pour atténuer ce problème [HS97].

Ils intègrent des portes qui permettent de contrôler le flux d'informations dans la cellule du réseau, ce qui leur permet de mémoriser des informations pendant de longues séquences et d'apprendre des dépendances à long terme. Comme l'ensemble des réseaux de neurones récurrents, les Long short Term Memorys (LSTMs) disposent d'un état caché de l'instant précédent h_{t-1} et de l'instant présent h_t. Chaque bloc dispose en plus d'un état de la cellule c_t jouant le rôle de mémoire. La figure II.4 présente un schéma de cellule LSTM dont le fonctionnement repose sur un ensemble de portes :

— La porte d'oubli † définit la quantité d'information à oublier de la part la cellule précédente c_{t-1}
— La porte d'entrée ‡ permet d'ajouter l'information de l'instant présent x_t
— La porte de sortie § permet de calculer l'état de la cellule courante c_t à passer à la prochaine cellule.

*. stride
†. Forget gate
‡. Input gate
§. Output Gate

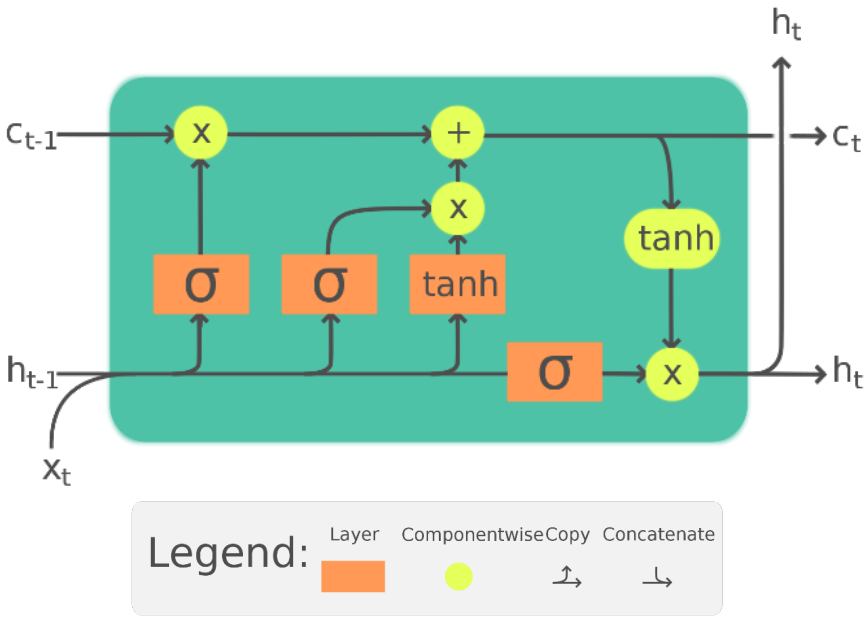


Figure II.4 : Schéma d'une cellule LSTM [Che18]

Leur conception les rend particulièrement utiles lors de l'apprentissage de représentation pour des données ayant des longues dépendances temporelles et donc en particulier l'étude de séries temporelles[XLD15, DLYH19, BKK18, HTJS22].

Bien que le concept de portes ait été introduit pour pallier le problème de gradient évanescent * [HS97], les LSTMs peuvent être plus complexes à entrainer par rapport aux réseaux de neurones présentés précédemment en raison de leur architecture plus sophistiquée et de leur plus grand nombre de paramètres. Cette augmentation du nombre de paramètres les rend par la même occasion plus lourds que les CNNs par exemple et donc moins embarquables.

Hyperparamètre des LSTM :

(i) Le nombre de couches LSTM $\mathcal{L}$
(ii) Le nombre de cellules par couches $\mathcal{N}$
(iii) Les fonctions d'activations des portes internes

B.5 Réseaux à connexions résiduelles

Les réseaux à connexions résiduelles, également connus sous le nom de ResNet (Residual Neural Networks) [HZRS16], sont des architectures de réseaux de neurones qui ont apporté une avancée significative dans la formation des réseaux de neurones. Contrairement aux architectures traditionnelles où les informations sont transmises de manière séquentielle à travers les couches, ils introduisent des connexions résiduelles † qui permettent un flux direct d'informations à travers le réseau. Ces connexions résiduelles sont créées en ajoutant une connexion directe qui permet à l'entrée d'être ajoutée directement à la sortie d'une ou plusieurs couches intermédiaires comme montré sur la figure II.5. Le module de base présenté par [HZRS16] est constitué de deux enchainements de couches convolutives et de couches de normalisation par batch ‡[IS15]. La sortie de la deuxième couche de normalisation est alors additionnée à l'entrée du module.

*. vanishing gradient
†. Skip connection
‡. Batch Normalization

L'introduction de ces connexions résiduelles présente plusieurs avantages. Tout d'abord, elles permettent de résoudre le problème de la dégradation du réseau, où l'ajout de couches supplémentaires entraine une diminution des performances du modèle. Grâce aux connexions résiduelles, les réseaux à connexions résiduelles peuvent apprendre des représentations plus profondes sans compromettre les performances [HZRS16]. De plus, ces connexions facilitent le flux de l'information et permettent aux gradients de se propager plus facilement lors de la rétropropagation, facilitant ainsi l'entrainement des réseaux profonds et diminuant les risques d'évanescence du gradient [HS97].

Grâce à leur architecture beaucoup plus profonde, les réseaux à connexions résiduelles ont atteint des performances de pointe dans de nombreux domaines de l'apprentissage automatique, notamment la vision par ordinateur, la reconnaissance d'objets et la segmentation d'images [SG22]. Leur capacité à former des réseaux plus profonds et plus performants a contribué à l'amélioration des résultats dans des tâches complexes en permettant une meilleure capture des features.

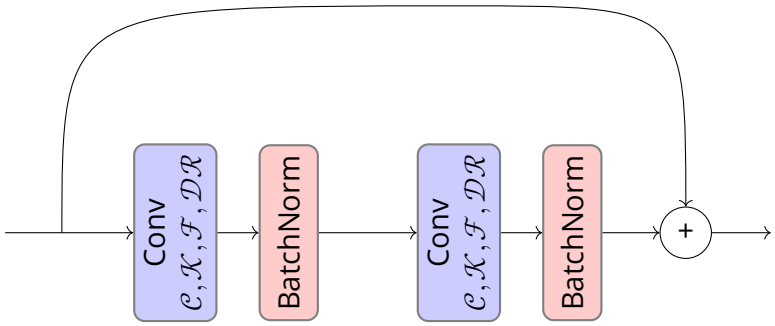


Figure II.5 : Un Bloc Resnet est composé de deux paires de couches convolutives 1D chacune couplée à un bloc de normalisation par batch

Hyperparamètre des réseaux à connexions résiduelles : Les connexions résiduelles sont une surcouche qui peut être apportée sur les couches totalement connectées (section B.2) ou sur les couches convolutionnelles (section B.3) par exemple. Nous ne nous concentrons pas ici sur les hyperparamètres de ces couches, mais bien sur les modifications apportées par les connexions résiduelles. Leurs hyperparamètres sont donc :

(i) Le nombre de blocs à connexion résiduelle utilisés à la suite $\mathcal{B}$

(ii) La constitution d'un module à connexion résiduelle. Le nombre de couches de convolution au sein d'un bloc peut différer de celui présenté figure II.5

B.6 Transformers

Les réseaux Transformers sont une architecture révolutionnaire dans le domaine de l'apprentissage automatique, principalement utilisés pour les tâches de traitement du langage naturel. en 2017 [VSP+17] et ont depuis lors dominé de nombreux domaines, notamment la traduction automatique, la génération de texte et la compréhension du langage [BMR+20, Ope23, TLI+23].

Contrairement aux réseaux de neurones récurrents (RNN) qui dépendent de connexions récurrentes, les Transformers se basent sur des mécanismes d'attention et plus particulièrement des mécanismes à auto-attention. L'auto-attention permet au réseau de se concentrer sur différentes parties de la séquence d'entrée lors de la prise de décision, en donnant plus de poids aux informations les plus pertinentes. Cela permet aux Transformers de modéliser les relations non seulement entre les éléments séquentiels, mais aussi entre tous

les éléments de la séquence en parallèle, ce qui les rend plus efficaces pour capturer des dépendances à long terme.

La clé de l'architecture Transformer réside dans son module d'auto-attention multi-têtes. Ce module permet au réseau de calculer plusieurs *têtes* d'auto-attention simultanément, ce qui lui permet de modéliser différentes relations et aspects de la séquence d'entrée. De plus, les Transformers utilisent des couches dites totalement connectées pour transformer les représentations intermédiaires, introduisant ainsi de la non-linéarité dans le réseau.

Enfin, [VSP+17] introduisent en complément de leur architecture un bloc d'encodeur positionnel, permettant de rajouter une notion de temporalité aux données observées et d'en informer le réseau de neurones.

Ce sont ces architectures novatrices composées de plusieurs têtes auto-attentives, l'utilisation d'encodeurs positionnels et la mise en place d'une procédure d'entrainement propre qui font la nouveauté de ces architectures ainsi que leur force.

Les Transformers traitent à la fois de problématiques liées au Traitement Automatique de la Langue Naturelle (TALN) [VSP+17, DCLT19, BMR+20], mais aussi depuis plus récemment, les Transformers pour la vision sont devenus l'état de l'art pour toutes les tâches de vision par ordinateur [DBK+21, CMS+20].

Une caractéristique notable des Transformers est leur capacité à être parallélisés, ce qui les rend plus efficaces lors de l'entrainement sur des architectures distribuées. D'un certain point de vue, ils peuvent être plus facilement interprétables, car les mécanismes d'auto-attention permettent de comprendre quelles parties de l'entrée sont les plus influentes pour les prédictions. Néanmoins l'empilement des modules auto-attentifs peuvent rendre le modèle particulièrement complexe à appréhender.

Les Transformers ont tendance à être plus couteux en termes de puissance de calcul et de ressources de mémoire par rapport à d'autres architectures, en particulier lorsqu'ils sont utilisés sur de grandes séquences ou dans des tâches à grande échelle. Cela peut limiter leur applicabilité dans des scénarios où les ressources sont limitées. De plus les modèles Transformers nécessitent souvent des ensembles de données volumineux pour capturer des représentations riches et générales. L'acquisition de telles quantités de données peut être un défi dans certaines applications ou domaines spécifiques. C'est pourquoi ce type d'architecture n'est pas utilisé dans la suite de ce manuscrit.

Table II.4 : Tableau de comparaison des différentes architectures de réseau de neurones

Architecture	Description	Utilisation typique	Avantages	Inconvénients
Perceptron multicouche	Un réseau de neurones complètement connecté avec plusieurs couches cachées	- Classification - Régression - Regroupement - Reconstruction	- Architecture simple - Facilité de mise en œuvre - Peu d'hyperparamètres	- Sur-apprentissage - Pas de notion de spatialité/-temporalité
CNN	Un réseau de neurones utilisant des couches de convolution pour extraire des informations ayant une cohérence spatiale ou temporelle	- Traitement d'image - Classification - Segmentation - Traitement de séries temporelles - Reconstruction	- Performances excellentes en vision par ordinateur - Prise en compte de la structure spatiale des données - Réutilisation des poids via la convolution	- Sur-apprentissage - Evanescence du gradient dans les réseaux profonds
LSTM	Un type de cellule de réseau de neurones récurrent utilisant des portes pour contrôler l'accès à son état interne et conserver une mémoire à long terme des éléments passés	- TALN - Traitement de séries temporelles - Génération de texte - Reconnaissance vocale	- Modélisation des dépendances à long terme - Performances sur les séquences - Gestion des problèmes de gradients	- Plus complexe à entrainer - Nécessite plus de ressources de calcul
Réseaux à connexion résiduelle	Un réseau de neurones utilisant des connexions résiduelles pour simplifier la propagation des gradients à travers les couches	- Traitement d'image - Classification - Segmentation	- Réseaux de neurones très profonds sans perte de performances - Meilleure propagation des gradients - Résultats de pointe en vision par ordinateur	- Beaucoup d'hyperparamètres - Nécessite plus de ressources de calcul - Moins approprié pour d'autres types de données
Transformers	Un réseau de neurones utilisant des modules auto-attentifs pour traiter des séquences de données	- TALN - Traitement de séries temporelles - Vision par ordinateur	- Modélisation des dépendances à long terme - Parallélisme élevé - Interprétabilité relative	- Complexité computationnelle élevée - Nécessite beaucoup de données d'entrainement

C Recherche d'architecture neuronale

La recherche d'architecture neuronale * englobe à la fois la recherche de l'architecture de réseaux de neurones la plus propice pour répondre à une problématique donnée, mais aussi les valeurs des hyperparamètres (aussi bien d'apprentissage que d'architecture) à choisir pour tirer les meilleures performances possibles de ces architectures. Les domaines du AutoML, de l'optimisation d'hyperparamètre, du meta-apprentissage [HKV19], y sont fortement liés. *"Le processus de choix des hyperparamètres dépend souvent d'une intuition ou d'une expérience personnelle d'une manière qu'il est difficile de quantifier ou décrire."* [BYC13]. La comparaison de modèles entre eux n'est en effet possible que sur la base de leurs performances à achever une tâche d'intérêt. Il est alors difficile de dire si un modèle est simplement meilleur qu'un autre ou s'il est mieux paramétré. L'élaboration d'algorithmes permettant d'optimiser les hyperparamètres comme l'architecture des modèles pour pouvoir les comparer au meilleur de leur performance est un point clé de l'entrainement de réseau de neurones profond et d'algorithmes d'apprentissage automatique.

Comme proposé dans [EMH19], on peut classer les techniques de recherche d'architecture neuronale selon trois axes : l'espace de recherche (section C.1), la stratégie d'exploration de l'espace de recherche (section C.2) et la méthode d'estimation des performances (section C.3).

C.1 Espace de recherche

La définition de l'espace de recherche, c'est-à-dire les hyperparamètres à explorer et les valeurs que ces derniers peuvent prendre est un point clé de la recherche d'architecture neuronale. D'un côté, l'espace de recherche peut être restreint par les connaissances apportées par les précédentes études ayant été menées pour une architecture ou une application donnée. De l'autre, la restriction de l'espace de recherche peut induire des biais humains et brider la découverte de nouveaux blocs intéressants. Il faut donc trouver un compromis entre la taille de l'espace à explorer et le temps de recherche dans cet espace. En effet, plus l'espace de recherche est grand, plus la stratégie d'exploration utilisée risque de ne pas explorer des parties clés de ce dernier.

L'espace de recherche peut alors être formalisé sous la forme d'arbre [BYC13] qui pourra être exploré selon diverses méthodes.

Les différentes améliorations apportées par la communauté scientifique consistent en un élargissement croissant de l'espace de recherche exploré. Ces derniers peuvent simplement aller du nombre de neurones contenus dans une couche dense [BBBK11] jusqu'à des briques plus complexes constituées de plusieurs couches neuronales [ZVSL18, EMH19, CZH19]. Un exemple simple est l'opposition des simples couches convolutionnelles des CNN face aux modules constitués de plusieurs couches convolutives des réseaux à connexion résiduelles ou encore des modules auto-attentifs des Transformers. Les études récentes montrent que l'utilisation de ces blocs élaborés permettent d'obtenir de meilleures performances. Néanmoins, ils posent de nouvelles questions : comment concilier à la fois l'ordonnancement des blocs complexes d'un point de vue macroscopique et le contenu interne de ces blocs qui lui aussi peut être optimisé. L'utilisation de tels blocs ne faisant qu'augmenter exponentiellement le nombre d'hyperparamètres et donc la taille de l'espace de recherche si l'on veut être exhaustif.

*. NAS : Neural Architecture Search

C.2 Stratégies d'exploration

La stratégie d'exploration est la méthode utilisée pour rechercher les meilleures combinaisons d'hyperparamètres dans l'espace de recherche proposé. Le but de la stratégie d'exploration est de trouver la meilleure architecture le plus rapidement possible. Les méthodes développées ne doivent évidemment pas converger trop rapidement vers un minimum local sans jamais en sortir, au risque d'obtenir des performances suboptimales. C'est le compromis d'exploration face à l'exploitation * formalisé par [BK97]. Il faut bien comprendre qu'il n'y a alors pas 2 métriques distinctes, mais bien un compromis à trouver entre le temps de recherche et les performances du modèle.

La première méthode de recherche d'architecture neuronale est la recherche par grille † [LBOM98, LEC+07, Hin12]. Elle consiste à positionner toutes les valeurs possibles des hyperparamètres sur une grille et à tester toutes les combinaisons possibles de ces valeurs. C'est la méthode la plus exhaustive, mais aussi la plus couteuse. En effet, le nombre de modèles à entrainer augmente exponentiellement avec le nombre d'hyperparamètres. Elle est souvent couplée à l'expertise apportée par les experts du domaine applicatif afin de restreindre la taille de l'espace de recherche à explorer. La recherche par grille est particulièrement intéressante grâce à sa facilité d'implémentation et ses possibilités de parallélisation, tous les modèles pouvant être entrainés indépendamment.

La recherche aléatoire ‡ [BB12, BYC13] propose une sélection aléatoire des valeurs d'hyperparamètres dans l'espace défini par l'utilisateur. Dans le cas où les hyperparamètres ne peuvent pas prendre de valeur continue, les valeurs peuvent être tirées aléatoirement sur une grille définie par l'utilisateur. Cette méthode permet d'obtenir de très bons résultats, mais ne permet pas de trouver le meilleur modèle de l'espace proposé.

Une méthode populaire de recherche d'architecture neuronale est l'optimisation Bayesienne [HHL11, SLA12, BBBK11, DSH15]. Le but est de mettre à jour une distribution *a priori* de la fonction cible (la métrique) en fonction des hyperparamètres grâce aux points auxquels cette fonction est évaluée. Une fonction d'acquisition dicte ensuite en quel point de l'espace des hyperparamètres la fonction cible doit être évaluée afin de maximiser/minimiser un critère donné. En fonction des modèles, le critère à optimiser peut être :

- La Probabilité d'Amélioration [Kus64] qui cherche à maximiser les chances d'améliorer le résultat précédemment obtenu,
- l'Amélioration Attendue § qui cherche à maximiser les chances d'améliorer les performances
- La borne haute de l'intervalle de confiance du Procédé Gaussien [SKKS10] qui dispose d'un paramètre κ permettant de réguler la part d'exploitation (étude de l'espace autour des points connus afin d'améliorer encore plus les performances) face à l'exploration de l'espace (nécessaire afin de ne pas explorer uniquement un minimum local)
- [BBBK11] propose de voir le choix des hyperparamètres comme un arbre décisionnel et propose donc une nouvelle fonction d'acquisition permettant de définir le prochain modèle à entrainer nommée Tree Parzen Estimator (TPE)

Un pan de la littérature se concentre sur l'exploration de l'espace des hyperparamètres grâce à l'apprentissage par renforcement [BGNR17, ZL17, ZYW+18]. L'exploration de l'espace de recherche correspond à l'espace des actions que peut effectuer un agent. La récompense de l'agent dépend alors des performances du modèle sélectionné. Les approches

*. Exploitation-exploration trade-off
†. GridSearch
‡. RandomSearch
§. Expected Improvement

proposées se différencient alors sur la politique de récompense et l'optimisation de cette dernière. Bien que ces méthodes permettent d'obtenir les meilleurs résultats, elles sont cependant très couteuses : [ZL17] utilisent par exemple 800 GPUs pendant 3 semaines afin de réaliser leur benchmark sur le modèle CIFAR-10 [Kri09].

L'utilisation d'algorithmes évolutifs [MTH89] et d'algorithmes génétiques est aussi étudiée. Ces derniers sont différenciés sur la manière de sélectionner les populations parentes, de mettre à jour les populations ou d'insérer des mutations au sein de ces populations. Les modèles, les plus anciens [ASP94, SM02, SDG09] utilisent des algorithmes génétiques à la fois pour la recherche d'architecture neuronale et pour l'optimisation des paramètres de ces architectures alors que les méthodes les plus récentes [Des17, RMS+17, SSN17, LSV+18, RAHL19b, GZM+20] utilisent une méthode basée sur la descente de gradient pour l'optimisation des paramètres de ces architectures une fois les valeurs d'hyperparamètres choisies par l'algorithme génétique.

L'étude [RAHL19a] compare ces approches d'apprentissage par renforcement, de recherche aléatoire et de méthodes évolutives. Ils concluent que les méthodes d'apprentissage par renforcement obtiennent des performances similaires aux méthodes évolutives bien que les secondes soient en général marginalement meilleures et avec moins de paramètres. La recherche aléatoire obtient quant à elle des résultats moins bons de 3 à 4% sur CIFAR-10.

C.3 Estimation des performances

Le point clé de la recherche d'architecture neuronale réside dans l'aspect couteux et long de l'estimation des performances d'un modèle, car celui-ci doit être entrainé avant de pouvoir être évalué. Les méthodes présentées dans cette section permettent d'accélérer les entrainements individuels de chaque modèle.

- L'estimation des performances basse fidélité *[LZZW17, RAHL19a] consiste à réduire le temps d'entrainement des réseaux en entrainant les modèles sur peu d'époques, sur un sous-ensemble des données ou sur des données de plus petites tailles (dans le cas d'images).
- L'Extrapolation de la courbe d'apprentissage [SSA14, DSH15, BGRN17] permet d'accélérer l'apprentissage en n'entrainant les réseaux que sur quelques époques. En extrapolant les performances des modèles [DSH15] propose d'arrêter l'entrainement des modèles dont l'extrapolation de la courbe d'apprentissage mène à de mauvaises performances.
- L'héritage des poids ou morphisme neuronal [CGS16, EMH17], est un ensemble de méthodes permettant de ne pas avoir à ré-entrainer l'ensemble des poids d'une architecture neuronale. Pour ce faire, les architectures sont explorées dans un ordre *croissant*. Des opérations permettent d'augmenter la profondeur des réseaux en rajoutant une couche ou sa largeur en augmentant le nombre de neurones (pour les couches denses) ou de filtres (pour les couches convolutives). Les poids des couches déjà existants sont alors hérités de l'architecture précédente. [EMH19] propose d'ajouter l'opération ajout de skip-connections aux morphismes possibles.

*. Lower fidelity estimate

D Apprentissage de représentation

L'apprentissage de représentation est la base sur laquelle tous les modèles d'Intelligence Artificielle contemporains reposent. Cette discipline vise à extraire des informations significatives et exploitables à partir de données brutes. En d'autres termes, elle cherche à créer des représentations abstraites et informatives des données, permettant aux systèmes informatiques de mieux les interpréter et ainsi résoudre des problèmes complexes. Le mot "apprentissage" sous-entend alors que cette extraction d'information peut être réalisée automatiquement par un modèle lors d'une phase d'apprentissage.

Nous nous concentrons en particulier sur une méthode dédiée à l'apprentissage de représentation appelée apprentissage auto-supervisé et présentée section D.1. Elle repose notamment sur la notion de tâche prétexte qui est détaillée section D.2.

L'apprentissage de représentation s'oppose à d'autres méthodes permettant d'extraire une représentation des données, mais ne reposant pas sur l'apprentissage profond, usuellement des méthodes de réduction de dimension ou de décomposition. Quelques-unes sont présentées

D.1 Apprentissage auto-supervisé

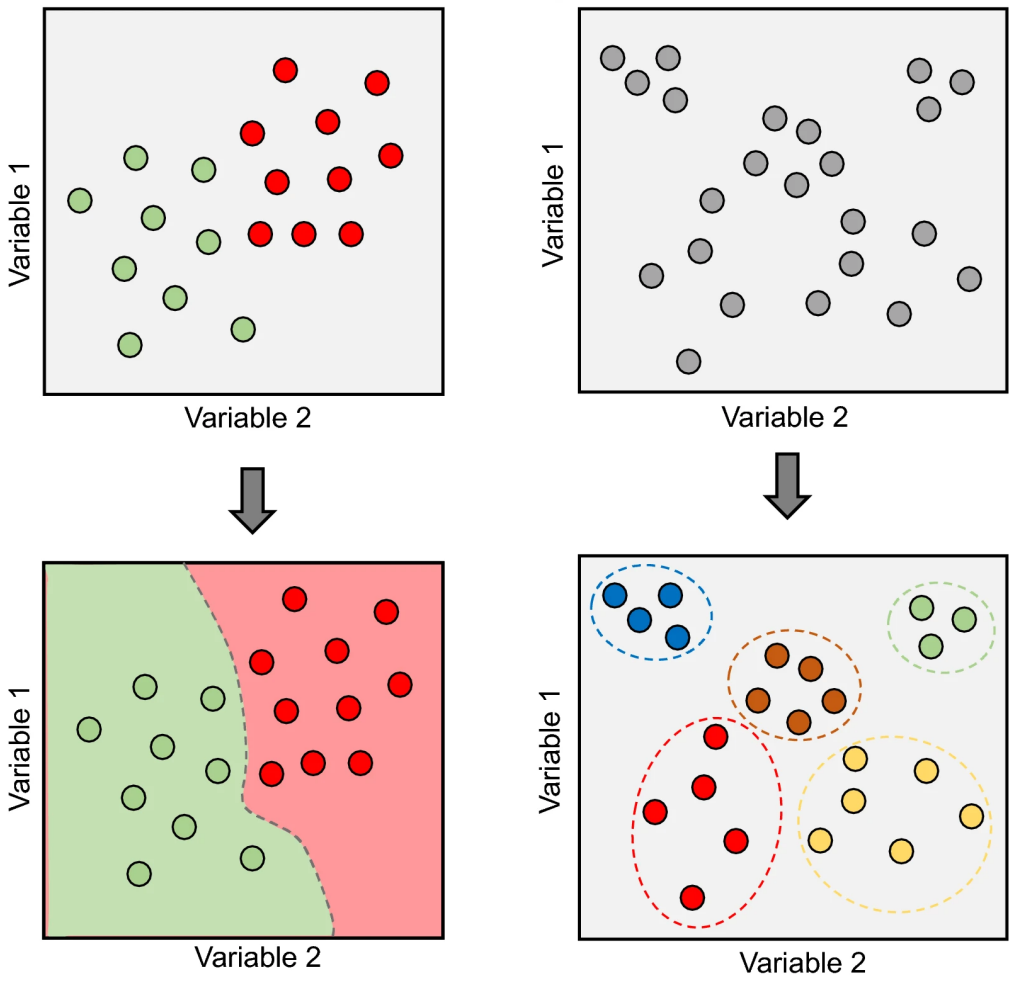
Historiquement, deux types d'entrainement de réseaux de neurones sont opposés dans la littérature, les entrainements supervisés face aux entrainements non supervisés. L'entrainement supervisé consiste à fournir au modèle des exemples annotés, c'est-à-dire des données d'entrée associées à des annotations (parfois appelées étiquettes ou labels) ou des résultats connus. L'objectif principal de l'entrainement supervisé est de permettre au modèle d'apprendre à prédire avec précision les annotations des données non étiquetées, en se basant sur les seules informations fournies par les exemples annotés. Les réseaux de neurones ne sont pas les seuls algorithmes à utiliser un entrainement supervisé : les Random forests ou encore les SVMs déjà présentés section A utilisent un entrainement supervisé.

Par opposition, l'entrainement non supervisé est utilisé lorsque les données ne sont pas étiquetées. Dans ce cas, l'objectif principal est de découvrir des structures, des modèles ou des relations intrinsèques dans les données sans aucun guidage extérieur. Les algorithmes non supervisés peuvent être utilisés pour regrouper les données * en fonction de leurs similitudes, réduire la dimensionnalité des données ou encore découvrir des motifs récurrents. La figure II.6 illustre cette opposition entre entrainement supervisé et non supervisé à la fois du point de vue de leur fonction et des contraintes d'annotation associées.

Bien que l'entrainement supervisé et non supervisé diffèrent par leur approche, ils ont en commun le fait de tirer parti des données d'entrée pour construire un modèle prédictif ou découvrir des informations utiles. Cependant, les contraintes qui y sont associées sont différentes. L'entrainement supervisé nécessite des exemples annotés, ce qui peut représenter un défi en termes de collecte de données et de couts associés à l'annotation. En revanche, l'entrainement non supervisé ne nécessite pas d'exemples annotés, mais il peut être difficile d'évaluer les résultats obtenus, car il n'y a pas de réponse attendue, de *vérité terrain*, à laquelle comparer les prédictions. Ces différences fondamentales sont illustrées figure II.6.

L'apprentissage auto-supervisé est une méthode d'apprentissage automatique qui se situe entre l'entrainement supervisé et non supervisé. Dans l'apprentissage auto-supervisé,

*. Clustering



(a) Apprentissage supervisé. Le modèle doit associer une classe (vert ou rouge) aux points qui lui sont présentés. Les annotations sont connues pendant l'entrainement.

(b) Apprentissage non supervisé. Le modèle doit regrouper les point similaires qui lui sont présentés. Les annotations ne sont pas connues pendant l'entrainement.

Figure II.6 : Illustration sur un exemple de la différence entre les apprentissages non supervisés et supervisés [MP21]

le modèle est formé à partir de données non annotées en créant une tâche prétexte pour générer des annotations *"artificielles"*, appelées pseudo-labels, à partir des données elles-mêmes. Contrairement à l'entrainement supervisé où les annotations sont fournies par des experts humains, l'apprentissage auto-supervisé exploite les structures intrinsèques des données pour générer des annotations de manière automatique [DGE16].

L'avantage de l'apprentissage auto-supervisé est qu'il peut tirer parti de grandes quantités de données non annotées disponibles, ce qui est moins couteux et plus facile à obtenir que des exemples étiquetés. De plus, en utilisant des tâches prétextes variées, l'apprentissage auto-supervisé peut permettre au modèle d'apprendre des représentations riches et plus générales des données, pouvant être utilisées dans diverses tâches d'intérêt.

Le tableau II.5 synthétise les avantages, inconvénients, contraintes et objectifs de ces 3 cadres d'apprentissage.

Remarque 6 : Pourquoi apprendre sans annotations ?

[RBL+07] propose une application numérique afin de mettre en avant l'intérêt de l'apprentissage sans annotations. Il compare la quantité d'information annotée nécessaire à l'apprentissage des connaissance d'un humain :

> « Un humain adulte dispose d'en moyenne 10^{14} synapses et vit environ 10^9 secondes. Ainsi, si chaque bit synapses est paramétrée par un bit. Il faudrait alors à un algorithme d'apprentissage $\frac{10^{14}}{10^9} = 10^5$ bit d'information par seconde pour *apprendre* toutes les connexions du cerveau. Il est très peu probable qu'un aussi grand montant d'information annotée soit disponible (sous-entendu de la part des parents ou professeurs pendant l'enfance) »

On comprend au travers de cette citation que l'entrainement de modèle supervisé ne mime pas parfaitement l'apprentissage réalisé par les humains et que d'autres méthodes doivent être utilisées afin de tirer profit des données non annotées.

Table II.5 : Tableau comparatif de l'apprentissage supervisé, non supervisé et auto-supervisé

	Apprentissage supervisé	**Apprentissage non supervisé**	**Apprentissage auto-supervisé**
Annotations	Oui	Non	Non
Définition	Un modèle est entrainé sur des données annotées, avec les sorties attendues associées aux entrées présentées	Un modèle est entrainé sur des données non annotées	Un modèle est entrainé sur des données non annotées, mais avec des méthodes pour créer des tâches auxiliaires qui guident l'apprentissage.
Objectif	Prédire les sorties correctes pour de nouvelles données d'entrée.	Découvrir des structures, des modèles ou des relations intrinsèques dans les données sans étiquettes.	Découvrir des représentations utiles et des structures latentes dans les données non annotées en utilisant des tâches auxiliaires.
Exemples	- Classification d'images - Prédiction de valeurs numériques - Détection d'objets	- Regroupement - Réduction de dimension - Apprentissage de représentations	- Pré-entrainement de modèles, apprentissage de représentations, prédiction de contexte, etc.
Applications courantes	- Reconnaissance d'images - Traduction automatique - Voitures autonomes	- Détection d'anomalies - Segmentation d'images - Apprentissage de représentations	- Pré-entrainement de modèles - Apprentissage de représentation
Avantages	- Performances élevées	- Peut découvrir des structures et des modèles intrinsèques dans les données - Moins couteux car il n'est pas nécessaire d'étiqueter les données manuellement.	- Peut exploiter des structures latentes et des représentations apprises à partir des données sans annotations - Peut permettre une meilleure généralisation en apprenant des représentations plus riches et générales.
Inconvénients	- couteux car nécessite l'étiquetage manuel des données - Dépend de la disponibilité des annotations	- La généralisation peut être plus difficile en l'absence de supervision explicite	- La création des tâches prétextes peut nécessiter un effort supplémentaire

D.2 Tâches prétextes

La tâche prétexte joue un rôle essentiel dans l'apprentissage auto-supervisé. Elle consiste à définir une tâche auxiliaire qui permet au modèle d'apprendre une représentation utile à partir des données non étiquetées. La tâche prétexte est choisie de manière à ce qu'il soit possible d'extraire automatiquement des pseudo-labels des données afin de pouvoir entrainer le modèle de manière supervisée. La qualité de la tâche prétexte est cruciale, car elle détermine la qualité des représentations apprises par le modèle.

L'importance de la tâche prétexte réside dans le fait qu'elle doit être conçue de manière à encourager le modèle à capturer les caractéristiques pertinentes des données. Elle doit être suffisamment complexe pour que le modèle ait besoin d'apprendre des informations riches et discriminantes, tout en étant réalisable sans annotations externes. En concevant une tâche prétexte pertinente, le modèle peut extraire des caractéristiques utiles et générales des données non annotées, ce qui améliore sa capacité à généraliser à d'autres tâches d'intérêt.

En traitement d'image

La littérature autour de l'apprentissage auto-supervisé pour le traitement d'image est extensive et couvre un grand nombre de tâches prétextes. Nous présentons dans cette sous-section des tâches prétextes ayant fait leur preuve dans le contexte de la vision par ordinateur. Il sera possible de s'inspirer de ces tâches prétextes pour les transférer au traitement des séries temporelles.

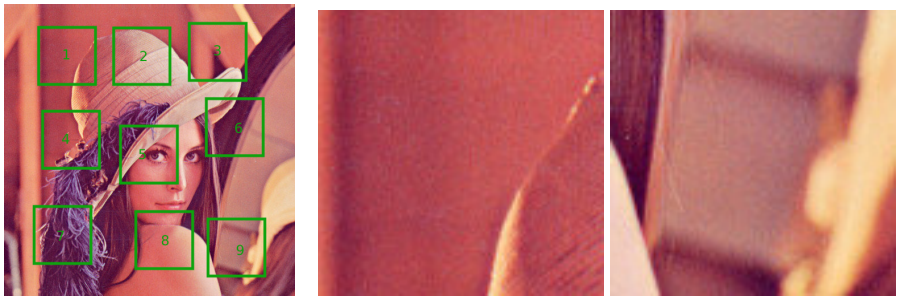
(i) Prédiction de rotation : [GSK18] propose de tourner aléatoirement les images de leur jeu de données de 0, 90, 180 ou 270 °. La tâche prétexte associée est alors la prédiction de la rotation de l'image.

(a) Entrée

(b) Pseudo-label

Figure II.7 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de prédiction de rotation

(ii) Puzzle : [DGE16, NF17] proposent de découper les images en pièces comme un puzzle. La tâche prétexte consiste à prédire la position relative de deux pièces présentées au réseau de neurones.



(a) Prétraitement **(b)** Entrées **(c)** Pseudo-label

Figure II.8 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de puzzle

(iii) Colorisation : [LMS17] propose de transformer les images RGB du jeu de données en noir et blanc et de demander au réseau de neurones comme tâche prétexte de prédire l'image en couleur.

(a) Entrée **(b)** Pseudo-label

Figure II.9 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de colorisation

(iv) Peinture : [PKD+16] propose de retirer le centre des images. La tâche prétexte consiste à repeindre l'intérieur des images qui a été retirée.

(a) Entrée **(b)** Pseudo-label

Figure II.10 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de peinture

(v) Détection des artéfacts : [JF18] dégrade artificiellement certaines images afin de créer des annotations liées à la dégradation *ou non* de ces images. La tâche prétexte est une classification image normale ou image modifiée.

(a) Entrée **(b)** Pseudo-label

Figure II.11 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de détection d'artéfacts

(vi) Reconstruction : D'autres tâches peuvent être englobées au sein de l'apprentissage auto-supervisé. Les auto-encodeurs ont pour but d'apprendre une représentation (un encodage) des données, généralement pour réduire leur dimension. Pour être entrainé, chaque image est alors aussi sa propre annotation. Dans la mesure ou un entrainement d'autoencoder est un entrainement réalisé de manière supervisée sans aucune annotation manuelle des données, cette tâche prétexte de reconstruction correspond bien au cadre de l'apprentissage auto-supervisé.

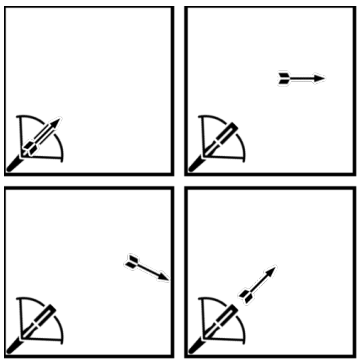
(a) Entrée

(b) Vecteur dans l'espace latent

(c) Pseudo-label

Figure II.12 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de reconstruction

(vii) Ré-arrangement temporel des images d'une vidéo : [LHSY17, MZH16] proposent d'apprendre une représentation de vidéos grâce à la tâche prétexte de ré-arrangement temporel des images d'une vidéo. Les images sont présentées dans un ordre aléatoire au réseau de neurones qui doit alors prédire leur ordre véritable.



(0,2,3,1)

(a) Entrée **(b)** Pseudo-label

Figure II.13 : Exemple d'entrée et de pseudo-label associé pour la tâche prétexte de réarrangement temporel des images d'une vidéo

En Traitement Automatique de la Langue Naturelle

On retrouve parfois dans la communauté de l'apprentissage profond un parallèle entre le traitement de séries temporelles et le NLP (NLP). Il découle de la similitude fondamentale entre les structures séquentielles présentes dans ces deux domaines. Dans le TALN, les séquences de mots dans un texte sont organisées de manière linéaire, reflétant la progression du discours ou de la signification. De même, dans le traitement de séries temporelles, les données sont enregistrées dans un ordre chronologique, où chaque instant de temps porte l'information contextuelle des instants précédents.

Bien que le traitement de séries temporelles et le TALN partagent des similarités dans leur nature séquentielle, ils se distinguent par leurs caractéristiques fondamentales. Les séries temporelles se concentrent sur l'évolution de valeurs numériques à travers le temps alors que le TALN traite de l'analyse et de la compréhension du langage humain, en explorant la signification sémantique des mots et leur contexte. Les séries temporelles mettent l'accent sur la prédiction de tendances et de modèles temporels, tandis que le NLP vise à extraire des informations, générer du texte ou comprendre la signification linguistique [TLI+23]. Bien que ces domaines diffèrent par leurs objectifs et leurs méthodologies spécifiques, leur convergence réside dans l'utilisation de modèles séquentiels pour capturer des dépendances contextuelles.

L'apprentissage auto-supervisé en TALN est devenu une méthode populaire pour l'apprentissage de représentations sémantiques à partir de grandes quantités de données textuelles non annotées. Les modèles sont entrainés à prédire certaines parties du texte à partir d'autres parties du même texte. Par exemple, ils peuvent être formés à prédire le mot manquant dans une phrase en fonction du contexte précédant et suivant [DCLT19]. Une autre approche courante est l'utilisation de l'encodage et du décodage, où le modèle est formé à encoder une phrase dans un espace latent et à décoder cette représentation pour reconstruire la phrase d'origine [MCCD13].

Ces représentations pré-entrainées peuvent ensuite être utilisées dans diverses tâches de TALN, telles que la classification de texte, l'analyse de sentiments ou la traduction automatique. Le TALN illustre parfaitement l'intérêt de l'apprentissage auto-supervisé : en utilisant des données textuelles non annotées, il est possible d'exploiter les grandes quantités de textes disponibles sur le web, ce qui permet d'améliorer les performances des modèles sur des tâches spécifiques même avec des données annotées limitées.

L'apprentissage auto-supervisé en TALN a été largement utilisé dans des modèles pré-entrainés tels que BERT (Bidirectional Encoder Representations from Transformers) [DCLT19]

dont la figure II.14 est extraite, ou GPT (Generative Pre-trained Transformer) [BMR+20] sur lequel repose le récent modèle de langage ChatGPT proposé par OpenAI. Ces modèles qui reposent sur une architecture de Transformers (section B) ont montré des performances remarquables dans diverses tâches de TALN, surpassant souvent les approches précédentes [BCE+23, KBGA23].

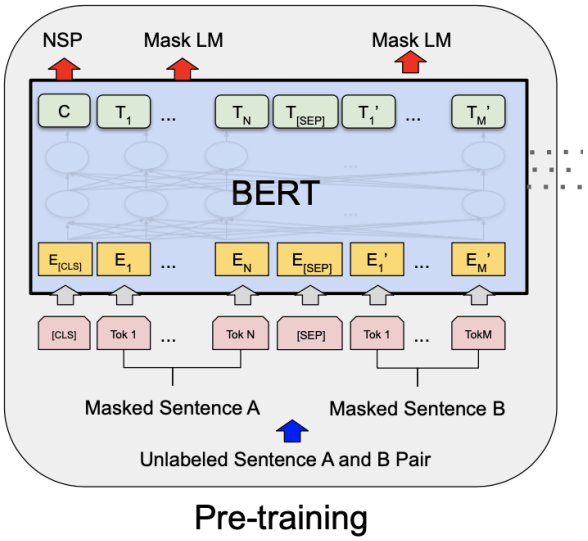


Figure II.14 : Exemple d'entrainement du modèle BERT. Le modèle doit prédire le token en rouge étant donné le reste des tokens de la phrase [DCLT19]

Les méthodes d'apprentissage auto-supervisé utilisées en TALN sont spécifiquement conçues pour exploiter les structures linguistiques présentes dans les données textuelles. Elles sont basées sur des hypothèses spécifiques sur la nature du langage, comme la continuité des mots dans une phrase ou les relations entre les mots. En général, l'entrainement de ces réseaux reprend le principe des phrases à trous. Le réseau de neurones doit prédire un mot au milieu de la phrase connaissant le contexte. Cette idée peut être étendue à d'autres tâches comme la prédiction du mot suivant, puis être généralisée à des phrases ou des paragraphes entiers de manière récursive.

Pour appliquer des méthodes d'apprentissage auto-supervisé à d'autres séries temporelles il est nécessaire d'adapter ces méthodes pour tenir compte des caractéristiques spécifiques des données temporelles. Cela peut impliquer la conception de tâches prétextes appropriées qui captent les propriétés temporelles de la série.

D.3 Apprentissage contrastif

L'apprentissage contrastif est une sous-classe de l'apprentissage auto-supervisé. Il se concentre sur la similitude et la différence entre les exemples. Dans l'apprentissage contrastif, l'objectif pour le réseau de neurones est d'apprendre à maximiser la similarité entre des paires d'exemples positifs tout en minimisant la similarité avec des exemples négatifs [vLV19, CKNH20, LHS20]. Les paires positives sont composées d'exemples provenant de la même classe, tandis que les paires négatives sont formées d'exemples provenant de classes différentes. Comme aucune annotation n'est disponible, il n'est pas possible de créer des paires positives en cherchant dans le jeu de données deux images appartenant à la même classe. Il est cependant possible d'utiliser 2 versions légèrement modifiées d'une même image afin de générer deux images différentes appartenant cependant à la même classe. Les techniques d'augmentation de données utilisées pour créer des versions mo-

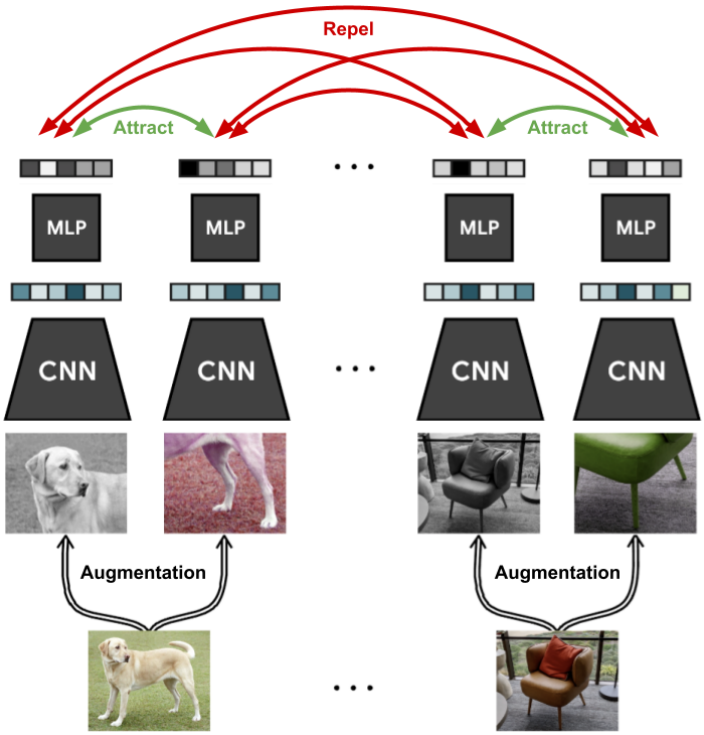


Figure II.15 : Illustration sur un exemple de paires positives et négatives présentées sur le github de SimCLR [CKNH20]. Les deux illustrations de chien constituent ensemble une paire positive, comme pour les deux images de chaise. Les chiens comparés aux chaises forment alors des images négatives

difiées de la même image sont au cœur de l'apprentissage contrastif. Les paires négatives peuvent-elles être construites en associant des éléments pris aléatoirement au sein du jeu de données.

La fonction de cout de l'apprentissage contrastif essaie donc de minimiser la distance entre les échantillons positifs tout en maximisant la distance entre les échantillons négatifs. Elle peut prendre différentes formes, mais la plus utilisée reste la InfoNCE équation (II.10) [CKNH20] avec $sim(u,v) = s_{u,v} = \frac{u^T \cdot v}{||u|| \cdot ||v||}$ la similarité entre les vecteurs u et v et $l_{i,j}$ la valeur de la fonction de cout pour la paire positive i,j. La valeur au numérateur mesure la similarité entre les représentations latentes de la paire positive face à la similarité des représentations des paires négatives. La fonction $\delta_{k,i}$ est la fonction delta de Kronecker [kro68] qui vaut 1 si $k = i$ et 0 sinon. Elle permet de ne pas prendre en compte la paire positive au dénominateur. Le paramètre τ est un paramètre de température. La fonction de cout $\mathcal{L}$ par batch est calculée en moyenne sur l'ensemble des paire positive $2k, 2k-1$ du batch et dans les 2 sens $l_{2k,2k-1}$ $l_{2k-1,2k}$

$$l_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} (1-\delta_{k,i}) \cdot \exp(sim(z_i, z_k)/\tau)} \tag{II.10a}$$

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} l_{2k,2k-1} + l_{2k-1,2k} \tag{II.10b}$$

Au début de cette thèse, comme le montre par exemple cet état de l'art [JBZ+21] sur l'apprentissage auto-supervisé et en particulier l'apprentissage contrastif, l'utilisation de méthodes contrastives pour le traitement de séries temporelles n'était alors pas démo-

cratisé. Nous avons cependant vu, au cours de cette thèse, l'utilisation de l'apprentissage contrastif émerger pour le traitement de séries temporelles [WKMJ22, YWD+22], [YZC22] spécialisé dans le traitement de séries temporelles univariées ou encore [AM21] pour le traitement de données audio. Cette émergence montre l'aspect très récent et novateur de l'utilisation de telles méthodes pour le traitement de séries temporelles.

En particulier [PCS22] propose une étude sur l'impact de l'apprentissage d'une représentation grâce à un modèle contrastif pour la détection d'anomalies dans les séries temporelles. L'utilisation d'un pré-entrainement contrastif permet d'obtenir un gain en exactitude de $+25$%, passant de 55.2% à 80.2% d'exactitude sur le jeu de données *Tennessee Eastman Process* [RATC18]. Nous présentons dans la section E.3 les méthodes d'augmentation de données utilisées dans cet article pour la génération des paires positives.

D.4 Représentations non basées sur l'apprentissage profond

D'autres techniques de représentation peuvent être utilisées afin de transformer des données d'entrées et de les projeter dans un espace plus propice à la résolution de la tâche à accomplir. Ces méthodes peuvent être divisées en deux catégories, celles reposant sur les données et celles, génériques, ne dépendant pas des données disponibles. Elles sont présentées dans cette section à titre indicatif et ne seront pas utilisées dans cette thèse. Cependant, il nous semblait important de mentionner leur existence pour pouvoir positionner les méthodes développées dans les chapitres suivants par rapport à ces dernières.

Méthodes ne dépendant pas des données

Projection aléatoire : La Projection aléatoire consiste à projeter les données dans un espace tiré aléatoirement [BM01]. Empiriquement, cette méthode permet de conserver les distances entre les entrées tout en gagnant des performances computationnelles comparées à une Analyse en Composantes Principales (ACP) section D.4.

Transformée de Fourier Discrète La Transformée de Fourier [FD+22] permet de projeter les données d'entrées dans l'espace des fréquences afin d'en avoir une compréhension différente. Elle permet de décomposer le signal sur une base de sinus et de cosinus de fréquences croissantes. Sa version discrète permet de l'appliquer à des données échantillonnées et quantifiées.

Transformée en Cosinus Discrète La transformée en cosinus permet de ne pas décomposer le signal sur une base complexe comme le propose la transformée de Fourier, mais sur une base de cosinus dans $\mathbb{R}$. Introduite par [ANR74], elle est aujourd'hui utilisée dans des méthodes de compression comme le format d'image JPEG [ISO].

Décomposition en Ondelettes : La décomposition en ondelettes [Chu92, Mey92], similaire à la transformée de Fourier à court terme, permet de décomposer le signal dans une base différente d'une base de sinus ou de cosinus.

Elle est notamment utilisée pour la compression de données dans des formats avec ou sans perte [ISO]

Méthodes dépendant des données

Analyse factorielle L'analyse factorielle [Spe04, Ste35] est une méthode analytique permettant de décrire la variabilité (faible ou forte) au sein d'un jeu de données . Elle est par-

ticulièrement utilisée dans le cas de données sous forme de tableau. La méthode calcule des variables latentes comme combinaison linéaire des variables observées.

Analyse en composantes principales L'ACP consiste à transformer des variables corrélées en nouvelles variables décorrélées les unes des autres [Pea01]. Ces nouvelles variables sont nommées les composantes principales. Les composantes principales permettent de décrire un maximum de la variance des données en un minimum de variables. L'ACP est utilisée pour de nombreuses applications dont la visualisation de données, le débruitage ou la réduction de dimension [Rin08, AW10].

Analyse en composante principale à noyau L'ACP repose sur l'hypothèse d'une corrélation linéaire entre les différentes variables latentes. L'ACP à noyau [SPS+01] permet d'étendre la technique de l'ACP en utilisant un noyau permettant de projeter les données qui seraient non séparables linéairement dans un espace où elles le sont dans lequel une ACP pourra être réalisée.

Analyse en Composantes Indépendantes L'Analyse en Composantes Indépendantes (ACI) [Com94], contraste avec l'ACP. Au lieu de maximiser la variance sur chacun des axes de la décomposition, cette méthode cherche à maximiser l'indépendance statistique des différentes composantes.

E Acquisitions et jeux de données

Dans ce document, une différence fondamentale sera faite entre les termes acquisitions et jeux de données. Alors que les acquisitions constituent l'ensemble des données brutes acquises par le capteur ou la méthode d'acquisition, les jeux de données correspondent aux données prétraitées, prêtes à être utilisées par un algorithme d'apprentissage automatique. Passer de l'un à l'autre est une phase de préparation des données * qui dépend grandement des données elles-mêmes mais aussi des traitements qu'elles vont subir par la suite.

E.1 Jeux d'entrainement, de validation et de test

La procédure utilisée lors d'un entrainement en apprentissage automatique prévoit de décomposer un jeu de données en 3 ensembles distincts [HTF09]. Le premier, le jeu d'entrainement sert à optimiser les poids du modèle. C'est grâce à ce dernier que sont optimisées les valeurs des poids du réseau de neurones lors de la phase de rétropropagation dans le cas de l'apprentissage profond. C'est aussi pourquoi ce doit être le jeu de données le plus conséquent.

Le jeu de validation permet d'obtenir une valeur de métrique permettant de comparer différents modèles et différents choix d'hyperparamètres entre eux. On utilise un jeu de validation différent du jeu d'entrainement afin d'éviter le sur-apprentissage du modèle sur le jeu de données et d'avoir des résultats comparables entre les modèles. Il peut aussi être utilisé dans le cas d'un entrainement avec arrêt anticipé † [Pre98, YRC07]. Cette méthode consiste à arrêter l'entrainement lorsque le modèle ne progresse plus ou commence à régresser.

Finalement le jeu de test est conservé jusqu'à la fin pour confirmer les performances du modèle sélectionné sur un jeu de données n'ayant servi ni à l'entrainement du modèle ni à la sélection des hyperparamètres.

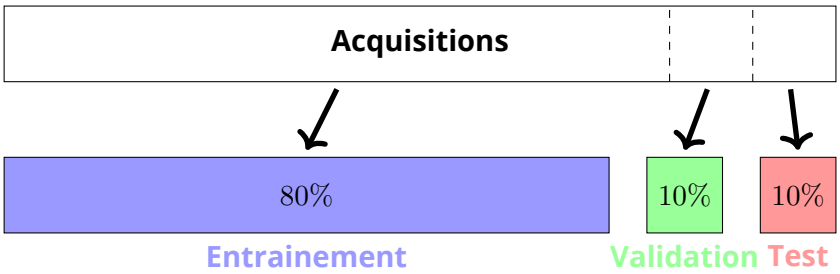


Figure II.16 : Découpage d'un jeu de données en jeux d'entrainement, de validation et de test

En résumé, le jeu d'entrainement sert à optimiser les poids d'un modèle, en particulier d'un réseau de neurones. Le jeu de validation permet de comparer différents modèles et de sélectionner, un modèle, une architecture, une valeur d'hyperparamètre. Enfin le jeu de test permet d'estimer de manière réaliste les performances que peut obtenir le modèle sélectionné.

Sauf contre-indication, les jeux de données sont toujours générés en attribuant 80% des séries temporelles des acquisitions au jeu d'entrainement puis 10% chacun au jeu de validation et de test (voir figure II.16). Tous les prétraitements qui pourront être décrits

*. preprocessing
†. Early stopping

dans ce manuscrit sont appliqués après le découpage des données dans ces 3 jeux de données.

E.2 Normalisation

Afin d'éviter des problèmes d'explosion du gradient lors de l'entrainement des réseaux de neurones profonds [GBC16], il est courant d'utiliser des méthodes de normalisation des données.

La littérature oppose 2 types de normalisations :

— la normalisation standard * ou standardisation qui consiste à centrer (moyenne égale à 0) et réduire (écart type égal à 1) les données

$$\mathbb{X}_{\text{normalisé}} = \frac{\mathbb{X} - \mu_{\mathbb{X}}}{\sigma_{\mathbb{X}}} \tag{II.1}$$

avec $\mu_{\mathbb{X}}$ et $\sigma_{\mathbb{X}}$ respectivement la moyenne et l'écart type de $\mathbb{X}$ calculés sur chaque instant de chaque série temporelle.

— la normalisation min-max qui consiste à comprimer les valeurs des acquisitions dans l'intervalle $[0, 1]$

$$\mathbb{X}_{\text{normalisé}} = \frac{\mathbb{X} - \min \mathbb{X}}{\max \mathbb{X} - \min \mathbb{X}} \tag{II.2}$$

L'échelonnage min-max étant linéaire, il ne permet pas de conserver l'information contenue dans la variance du signal. De plus, il est moins robuste lorsque les grandeurs d'entrée varient beaucoup. C'est pourquoi sauf contre-indication, les acquisitions de ce manuscrit seront standardisées.

E.3 Techniques d'augmentation des données

Parmi les prétraitements qui peuvent être effectués sur un jeu de données, l'augmentation de données et l'une des plus importantes. En effet, l'une des faiblesses de l'apprentissage profond est la quantité massive de données nécessaires pour entrainer un réseau de neurones. Il n'existe pas de définition claire des données *minimales* nécessaires pour entrainer un réseau, mais, en général, le nombre de paramètres du réseau est un bon indicateur. Le consensus est que le nombre de données d'entrées doit être bien supérieur au nombre de paramètres du réseau de neurones. Cependant, cela n'est pas toujours possible car les jeux de données peuvent être très petits. Par exemple, parmi les 128 jeux de données des archives de l'UCR [DBK+19], 101 jeux de données contiennent 500 éléments ou moins dans le jeu d'entrainement. Afin de répondre à ce manque de données, [CCC16] propose une technique d'augmentation de données pour les séries temporelles appelée fenêtrage temporel. Comme l'extraction de plus petits morceaux du signal ne change pas sa classe, le signal original est découpé en fenêtres (plus petites), ce qui peut augmenter considérablement la taille du jeu de données. Le fenêtrage apporte deux avantages supplémentaires. Il permet de se placer dans un cas ou l'ensemble des données ne seraient pas disponible simultanément, comme pour un capteur intelligent déployé sur le terrain bénéficiant d'une quantité de mémoire définie. Par ailleurs, le fenêtrage permet aussi d'harmoniser la durée des séries temporelles lorsque ces dernières ont une durée variable dans le jeu de données considéré.

*. standard normalisation

Alternativement, [GMT16] propose une solution de déformation des fenêtres * : une partie du signal est soit accélérée, soit décélérée sur l'axe temporel. [UPP+17] étudient l'impact de plusieurs méthodes d'augmentation des données telles que la rotation, la permutation, la mise à l'échelle, l'ajout de bruit, la déformation des fenêtres, le fenêtrage glissant et leurs combinaisons, atteignant une performance de $+10\%$ sur une tâche de classification.

Par opposition aux méthodes précédentes qui modifiaient les signaux existants pour augmenter le jeu de données, [FFW+18, FPD+17] génèrent de nouvelles séries temporelles de la même classe en faisant la moyenne d'un ensemble de séries temporelles en une nouvelle série temporelle. Le fait de pondérer les séries temporelles différemment conduit à un nombre presque infini de possibilités d'augmentation.

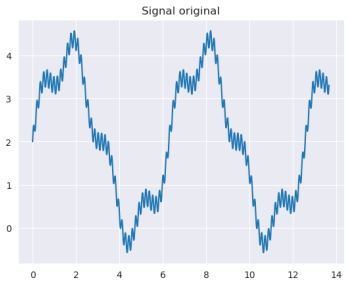
Contrairement aux images, il est plus difficile pour les données temporelles de définir des méthodes d'augmentation de données qui ne changent pas la sémantique de la série temporelle augmentée. C'est pourquoi au cours de ce manuscrit, nous nous limiterons, dans le cas général à la seule utilisation d'un fenêtrage glissant [CCC16]. Dans ce manuscrit le terme "série temporelle" fait référence à la mesure d'une ou plusieurs modalités pendant toute la durée de l'expérience par opposition au terme "fenêtre" qui correspond à une petite partie de la série temporelle complète. En somme, un grand nombre de fenêtres peuvent être extraites d'une série temporelle.

Les méthodes d'augmentation de données peuvent avoir un deuxième objectif : la génération d'une paire positive pour l'apprentissage contrastif (présenté section D.3). [PCS22] propose 9 méthodes d'augmentation de données pour la génération de paire positives pour les séries temporelles. Ces méthodes sont décrites dans le tableau II.6 et illustrées sur la figure II.17. Nous nous attardons particulièrement sur ces transformations car elles seront utilisées dans le manuscrit chapitre V.

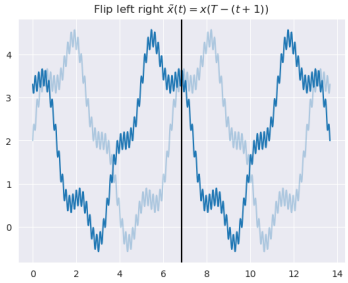
*. window warping

Table II.6 : Tableau comparatif des différentes méthodes d'augmentation de données pour l'apprentissage contrastif proposées dans [PCS22]

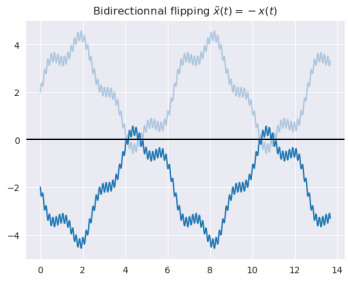
Nom	Description	Équation
Signal original Figure II.17a		$x(t)$
Retournement gauche-droite Figure II.17b	Le signal est renversé tel que les premiers éléments deviennent les derniers et inversement	$\tilde{x}(t) = x(T-t+1)$
Renversement bidirectionnel Figure II.17c	Un signal miroir est généré autour de l'abscisse	$\tilde{x}(t) = -x(t)$
Permutations aléatoires	L'ordre des modalités du signal est changé de manière aléatoire	
Bruit aléatoire Figure II.17d	Ajout d'un bruit uniforme positif ou négatif de l'amplitude de l'écart type du signal	$\tilde{x} = x + R * \lambda * std(x)$ $R \in [-1,1]^T$
Blocage Figure II.17e	Remplace un bloc carré (temps -) par des 0	
Rogner et redimensionner Figure II.17f	Une fenêtre de taille $T/2$ est choisie aléatoirement dans le signal. Le signal est interpolé 1 point sur 2 dans cette nouvelle fenêtre pour garder un nombre de points constant	
Déformation de l'amplitude Figure II.17g	Ajout au signal d'un sinus de période T. L'amplitude la modulation est définie par l'utilisateur par le paramètre λ. Chaque modalité est modulée par le même sinus déphasé de θ_i choisi aléatoirement et différent pour chaque modalité	
Déformation de la temporalité Figure II.17h	Ajout de jitter sur l'axe temporel	
Lissage aléatoire Figure II.17i	Utilisation d'un filtre FIR pour lisser le signal. λ est tiré aléatoirement dans $[0,1]$	$H(z) = \frac{\lambda}{2} z^1 + (1-\lambda)\frac{\lambda}{2} z^{-1}$



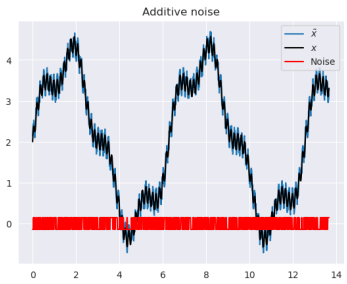
(a) Signal original



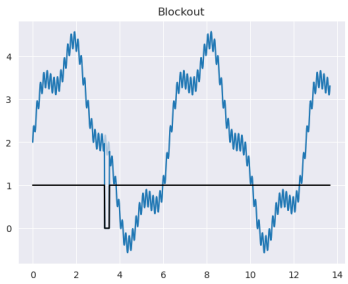
(b) Retournement gauche-droite



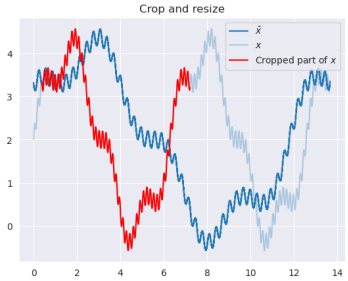
(c) Renversement bidirectionnel



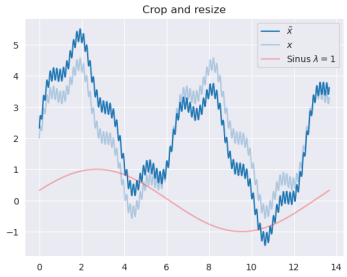
(d) Bruit aléatoire



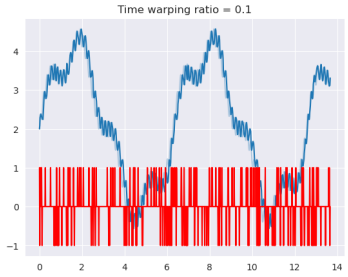
(e) Blocage



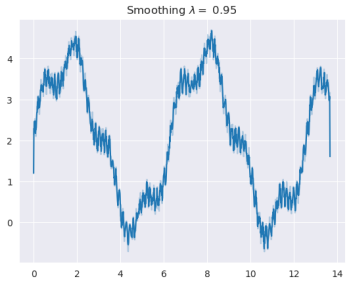
(f) Rogner et redimensionner



(g) Déformation de l'amplitude



(h) Déformation de la temporalité



(i) Lissage aléatoire

Figure II.17 : Illustration des différentes méthodes d'augmentation de données pour l'apprentissage contrastif proposées dans [PCS22]

Conclusion

Nous avons introduit dans ces sections les concepts nécessaires à la compréhension du reste du manuscrit et à son positionnement au sein de l'état de l'art. En partant du plus générique au plus spécifique et utile, nous avons d'abord présenté les méthodes de détection d'anomalies section A. Nous avons ensuite présenté les architectures de réseaux de neurones les plus utilisées dans la littérature section B et les manières de choisir les hyperparamètres de ces architectures grâce à la recherche d'architecture neuronale section C. Les méthodes d'apprentissage de représentations et notamment l'apprentissage auto-supervisé ont été introduits dans la section D. La notion de tâches prétextes pour l'apprentissage auto-supervisé ainsi que celle d'apprentissage contrastif qui sont les plus importantes utilisées au cours de ce manuscrit. Enfin la section E présente les méthodes utilisées pour la gestion des acquisitions et des données ainsi que les méthodes de l'état de l'art pour l'augmentation de données.

Maintenant que les concepts connus ont étés introduits, le chapitre III va présenter les contributions méthodologiques de ce manuscrit avant de les appliquer à deux jeux de

données différents dans les chapitres IV et V.

CHAPITRE

III

Méthode de détection d'anomalie : mise en œuvre

Table des matières

A Introduction

Afin de répondre aux problématiques posées en introduction (Chapitre 1 section D) et en complément des méthodes de l'état de l'art, nous proposons dans ce chapitre trois contributions.

Nous proposons section B le modèle Self-supervised Anomaly Detection on Time Series (StArDusTS). Ce modèle de détection d'anomalies repose sur un premier bloc d'apprentissage de représentation ne nécessitant pas d'annotation manuelle grâce à une méthode d'apprentissage auto-supervisé, puis un bloc de détection d'anomalies.

Ensuite, section C, une méthode d'optimisation des hyperparamètres d'un réseau de neurones reposant sur une optimisation successive de ces derniers. Elle formalise et automatise l'approche intuitive d'optimisation successive des hyperparamètres utilisés comme première approche pour la recherche d'architecture neuronale.

Enfin nous présentons un outil logiciel implémentant les deux contributions précédentes afin de pouvoir comparer les différentes méthodes d'apprentissage de représentation couplées à des architectures et des algorithmes de détection d'anomalies.

B StArDusTS : Self-Supervised Anomaly Detection on Time Series

B.1 NEW Le modèle

SCIENTIFIC REPORTS Le modèle StArDusTS a été soumis pour une publication dans scientific reports [BMA+23b]

Afin de pouvoir détecter des anomalies, sans connaitre les annotations de ces dernières, nous proposons le modèle StArDusTS, pour Self-supervised Anomaly Detection on Time Series.

Ce modèle, repose sur deux blocs distincts comme le montre la figure III.2. Il se compose d'abord d'un bloc d'apprentissage de représentation ne nécessitant pas d'annotation manuelle puis second bloc de détection d'anomalies. Sa nouveauté réside au centre de l'utilisation de méthodes auto-supervisées, à base de CNN à une dimension (CNN 1D) pour la détection d'anomalie comme le montre la figure III.1.

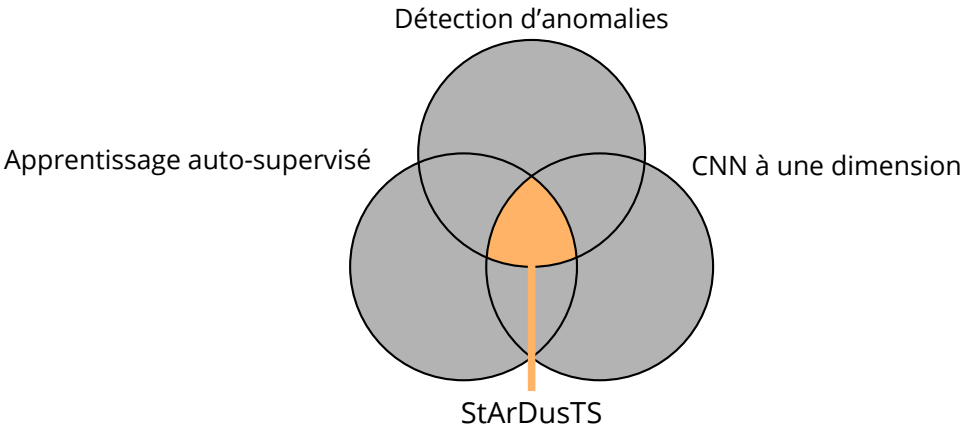


Figure III.1 : Illustration du positionnement central de StArDusTS

Le bloc d'apprentissage de représentation repose sur l'apprentissage auto-supervisé pour apprendre une représentation des données qui est ensuite utilisée pour la détection d'anomalies. Une fois une représentation des données apprises, sans annotations, nous proposons d'utiliser un algorithme de détection d'anomalies de l'état de l'art. Le détecteur est alors appliqué sur la représentation afin d'identifier les données anormales.

Comme pour le bloc d'apprentissage de représentation, le contenu bloc de détection d'anomalies peut être différent en fonction du cas applicatif traité. Par ailleurs, en fonction de la représentation apprise, un algorithme de détection d'anomalies peut s'avérer plus efficace qu'un autre, c'est pourquoi nous ne restreignons pas StArDusTS à l'usage d'un détecteur en particulier. Les détecteurs présentés dans l'état de l'art chapitre II section A seront utilisés dans ce manuscrit.

La nouveauté de StArDusTS provient du couplage d'un bloc d'apprentissage de représentation apprentissage auto-supervisé avec une méthodologie de détection d'anomalies non supervisée plus classique. Il tire parti de la puissance de représentation des réseaux de neurones profonds et de la quantité de données disponible pour pouvoir extraire automatiquement, sans aucune annotation préalable, les éléments anormaux du jeu de données.

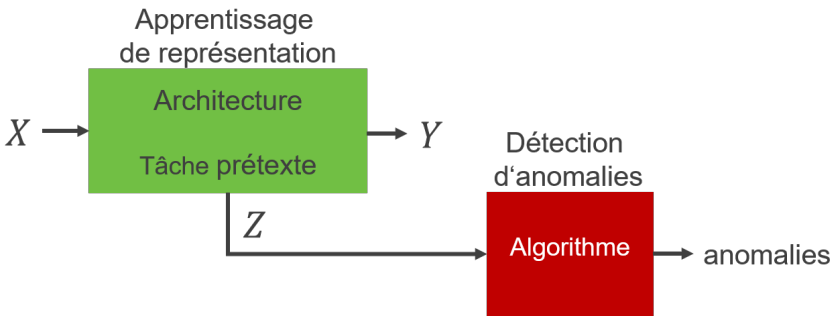


Figure III.2 : Présentation schématique du modèle StArDusTS, basé sur un bloc d'apprentissage de représentation et un bloc de détection d'anomalies

B.2 Apprentissage de représentation

Le bloc d'apprentissage de représentation utilise l'apprentissage auto-supervisé présenté chapitre II section D.1. Dans le contexte de l'apprentissage auto-supervisé, deux éléments clés fonctionnent conjointement et doivent être choisis :

— L'architecture des réseaux de neurones utilisée,
— La tâche prétexte utilisée pour entrainer ce réseau de neurones.

Quelle architecture neuronale ? Il n'y a aujourd'hui pas de consensus sur l'architecture à utiliser dans un problème donné.

Les recherches bibliographiques menées dans la section B nous ont poussés à nous diriger principalement vers l'utilisation d'architecture CNN 1D.

Grâce à leurs couches de convolution, de sous-échantillonnage et de couches entièrement connectées, les CNN 1D sont capables de capturer des motifs locaux et globaux dans les séries temporelles, tout en réduisant le risque de sur-apprentissage. Ils sont également adaptés pour gérer des séries temporelles de longueurs variables, ce qui est courant dans de nombreux domaines de traitement de série temporelle. En outre, l'utilisation de filtres convolutifs permet de détecter des caractéristiques temporelles invariantes, ce qui en fait des candidats idéaux pour la détection d'anomalies, la classification et la prédiction dans les séries temporelles. Enfin, leur faible complexité les rend plus simples à entrainer que

des architectures plus complexes comme les réseaux de neurones récurrents ou les transformers.

Quelle tâche prétexte pour des séries temporelles ? Les tâches prétextes présentées chapitre II section D.2, et de manière générale celles disponibles dans la littérature, reposent principalement sur des jeux de données d'images et ne sont pas forcément adaptées pour le traitement de séries temporelles. Nous avons cherché dans ce manuscrit à transposer les méthodes présentes dans l'état de l'art pour le traitement d'image au traitement de série temporelle. Nous nous concentrons dans ce document sur deux tâches prétextes :

1. La prédiction du futur d'une série temporelle. En effet, une série temporelle peut être découpée en une *entrée* x correspondant aux premiers instants de la série temporelle connue et une sortie y pour les instants suivants.
2. L'apprentissage contrastif présenté chapitre II section D.1, pour tirer parti d'une représentation qui tente d'organiser son espace latent, dès son entrainement.

Hypothèses : Le fonctionnement du modèle StArDusTS repose sur deux hypothèses majeures, classiquement faites dans la communauté scientifique. Elles sont exposées ci-dessous.

1. Comment évaluer une représentation ? Ou formulé autrement, comment comparer deux représentations ? Il n'est pas possible dans l'absolu d'affirmer la supériorité d'une représentation par rapport à une autre. En effet, l'une ou l'autre peuvent avoir des intérêts différents. Tant que la tâche finale, dans notre cas la détection d'anomalies, n'est pas réalisée, il n'est pas possible d'affirmer quelle représentation sera la plus efficace. Cependant, afin de découpler le problème d'apprentissage de représentation du problème de détection d'anomalies nous prenons l'hypothèse suivante :

   Hypothèse 1 :

   Plus la métrique sur la tâche prétexte est bonne, plus la représentation sera efficace pour la tâche finale que l'on cherche à réaliser.

   Ainsi, les différentes représentations apprises sur une même tâche prétexte sont comparées par rapport à leurs performances sur ladite tâche prétexte. Cette hypothèse est établie à partir du constat suivant : si le réseau de neurones n'est pas capable d'accomplir la tâche prétexte pour laquelle il a été entrainé, c'est qu'il n'a pas appris une représentation des données utile. Par contraposition, il serait donc préférable de conserver les architectures obtenant les meilleures performances sur la tâche prétexte. Cette hypothèse nous permet de découpler le problème d'apprentissage de représentation du problème de détection d'anomalies et de *comparer* les représentations apprises par apprentissage auto-supervisé.
2. Une seconde hypothèse est nécessaire pour le bon fonctionnement de StArDusTS :

   Hypothèse 2 :

   Les données d'entrainement $\mathbb{X}_{\text{train}}$ utilisées pour l'apprentissage de représentation sont majoritairement des données normales.

   Cela permet d'utiliser l'oubli catastrophique [MC89, MSPT21] des réseaux de neurones à notre avantage. Si les données anormales sont en minorité dans les jeux de données, les réseaux de neurones ne seront pas capables d'apprendre correctement

une représentation des données anormales. La représentation étant mauvaise pour ces données, elles seront facilement séparables des données normales.

Cette hypothèse permet de transformer un problème totalement non supervisé en un problème où les annotations normales sont connues, qui peut donc être qualifié de semi-supervisé. Cette différence fondamentale, nous permet, au besoin d'appliquer des méthodes telles que les OCSVM qui présupposent un cadre semi-supervisé où seulement des observations normales sont disponibles lors de l'entrainement du modèle.

Dans les deux cas applicatifs considérés dans ce manuscrit, cette hypothèse est vérifiée.

B.3 Détection d'anomalie

Une fois l'entrainement du réseau de neurones du bloc précédent entrainé, ses poids sont fixés et permettent de projeter les données dans un espace latent Z plus propice à réaliser la détection d'anomalies. Les features dans l'espace latent sont alors utilisées en entrée d'un des algorithmes de détection d'anomalies présenté dans le chapitre II section A.1.

B.3.1 Détecteur à seuil sur les fenêtres

Dans un premier temps, un détecteur à seuil (chapitre II section A.1) est utilisé sur l'Erreur Quadratique Moyenne (EQM) entre la prédiction du futur de la série temporelle et sa véritable valeur.

Ce dernier compare la prédiction réalisée grâce à un réseau de neurones avec une architecture CNN 1D à la véritable valeur du signal considéré. Si la prédiction est *trop loin* de la vérité terrain, la fenêtre observée est détectée comme anormale. Cette notion de *trop loin* est mesurée grâce à l'EQM entre la prédiction et la vérité terrain.

Le seuil τ d'EQM au-delà duquel une unique fenêtre est considérée comme anormale est calculé équation (III.1) de manière à ne conserver que les fenêtres dont la valeur d'EQM appartient à l'intervalle de confiance à $97.5\,\%$[Bal96, DKLM05].

$$\tau = \mu_{\text{train}} + 2 \cdot \sigma_{\text{train}} \tag{III.1}$$

Cette valeur correspond aux échantillons en dehors de l'intervalle de confiance à $95\,\%$ par excès. Si l'on suppose la distribution gaussienne des données, classifier les données en dehors de l'intervalle de confiance à $97.5\,\%$ comme anormales revient à choisir les $2.5\,\%$ des données dont les valeurs d'EQM sont les plus grandes. La moyenne μ_{train} des EQMs et l'écart-type σ_{train} sont calculés sur le jeu d'entrainement $\mathbb{X}_{\text{train}}$.

Nous calculons cette valeur du seuil sur le jeu d'entrainement afin d'être cohérent avec les autres algorithmes de détection d'anomalies comme les OCSVM et les random forest qui seront entrainés sur ce jeu de données.

B.3.2 NEW Score d'anomalie sur les séries temporelles complètes

Nous proposons donc un second détecteur, le score d'anomalie S, qui agrège l'ensemble des résultats individuels des anomalies détectées sur les fenêtres d'une même série temporelle. Il permet de considérer les multiples fenêtres d'une série temporelle ensemble et non plus indépendamment et donc de surpasser le problème du détecteur à

seuil exposé ci-dessus. En ce sens, il permet d'intégrer la temporalité des données en une unique valeur de score.

Ce score permet aussi d'augmenter la robustesse de la détection en ne considérant pas les séries temporelles *peu* anormales.

Le score d'anomalie équation (III.2) permet de comparer la quantité de fenêtres anormales à la quantité totale de fenêtres d'une série temporelle. Plus la série temporelle a de fenêtres pour lesquelles la prédiction est mauvaise, plus son score d'anomalie est élevé. En effet, un score d'anomalie de 1 correspond à 100% des fenêtres observées supérieures à τ.

$$\mathcal{S}_c = \frac{\text{Nombre de fenêtres avec EQM} > \tau}{\text{Nombre de fenêtres}} \tag{III.2}$$

Ordonner les séries temporelles avec un score d'anomalie du plus élevé au plus faible permet donc d'obtenir une *classification* des séries temporelles de la plus anormale à la moins anormale. Nous proposons donc de considérer comme anormales uniquement les séries temporelles dont plus d'un quart des fenêtres *vues* par le modèle ont une EQM supérieure à τ.

Cette valeur de $25\,\%$ a été fixée empiriquement de manière à conserver un nombre acceptable de séries temporelles à annoter dans les expériences du chapitre IV. Choisir un seuil sur le score d'anomalie plus élevé reviendrait à sélectionner uniquement les séries temporelles les plus anormales, mais retirer des données *moins* anormales à observer. Le choix de cette valeur correspond dans le cas des expériences de ce manuscrit à un bon compromis.

Le détecteur à score est un complément du détecteur à seuil et pas uniquement une amélioration. En effet, en fonction du cas applicatif, le traitement indépendant de multiples fenêtre peut s'avérer être une tâche intéressante. Par exemple, lors d'une étude où les données arrivent en continu, le traitement indépendant des fenêtres peut s'avérer plus profitable, notamment à cause des contraintes de stockage de données et de puissance de calcul. D'un autre côté, dans le cas d'une étude *a posteriori*, obtenir des détections plus robustes, intégrant l'ensemble de l'historique d'une série temporelle, mais à un cout calculatoire plus élevé peut être préférable.

C Optimisation successive des hyperparamètres

Nous exposions dans les sections précédentes que nous nous restreignons à l'usage de réseau de neurones avec une architecture de CNN 1D. Cependant, la question du choix des hyperparamètres de cette architecture reste encore à définir. Nous proposons dans cette section de suivre une démarche d'optimisation successive afin de définir les meilleures valeurs d'hyperparamètres du réseau de neurones utilisées. Cette méthode reprend le principe d'optimisation des hyperparamètres réalisée manuellement par les chercheurs du domaine et la formalise mathématiquement. À notre connaissance, bien que cette méthode soit implicitement utilisée, elle n'a jamais été formalisée pour le domaine de l'apprentissage automatique.

Nous illustrons sur les figures III.3a et III.3b respectivement la recherche par grille et la recherche aléatoire, déjà présentée section C.

Remarque 7 : Recherche par grille, combien de temps ?

Soit une architecture à 5 hyperparamètres pouvant chacun prendre 5 valeurs. Le nombre de combinaisons unique d'entrainement est alors de $5^5 = 3125$ réseaux de neurones différents à entrainer.

En supposant un temps d'entrainement moyen d'une heure par architecture, cela représente $\mathbf{130}$ jours d'entrainements sans interruption pour une unique architecture sur une unique tâche prétexte.

NEW **Recherche par optimisation successive :** Le pseudo code 1 explicite l'algorithme d'optimisation successive des hyperparamètres.

Dans un premier temps, l'utilisateur définit les hyperparamètres qu'il souhaite optimiser et les valeurs que ces derniers peuvent prendre, de la même manière que pour la recherche par grille. L'utilisateur définit aussi une valeur par défaut $\mathcal{H}_{\text{défaut}}[i]$ pour chaque hyperparamètre $\mathcal{H}_i$.

Ensuite, une première série de modèles, entrainée avec les valeurs par défaut pour les paramètres 2 à n_h sont entrainés en faisant varier la valeur du paramètre 1 parmi les choix $\mathcal{H}_1$ proposés par l'utilisateur. Ensuite la valeur du paramètre 1 est fixée à la valeur pour laquelle la meilleure performance a été obtenue. Les valeurs possibles du paramètre 2 sont explorées avec le paramètre 1 fixé à sa *meilleure* valeur et les paramètres 3 à N fixés à leur valeur par défaut. Le processus est alors répété jusqu'à ce que toutes les valeurs du dernier hyperparamètre aient été testées.

Algorithm 1 Optimisation successive des hyperparamètres, cas à 3 hyperparamètres

$\mathcal{H}_1 \leftarrow \left[h_1^1, h_1^2 \cdots h_1^{n_1}\right]$
$\mathcal{H}_2 \leftarrow \left[h_2^1, h_2^2 \cdots h_2^{n_2}\right]$
$\vdots$
$\mathcal{H}_{n_h} \leftarrow \left[h_{n_h}^1, h_{n_h}^2 \cdots h_{n_h}^{n_{n_h}}\right]$
$\mathcal{H}_{\text{défaut}} \leftarrow \left[h_1^1, h_1^2 \cdots h_{n_h}^1\right]$
Pour tout $i \in n_h$: **Faire**
  $\mathcal{L}_i \leftarrow []$
  Pour $j \in n_i$: **Faire**
    modèle $\leftarrow$ archi(h_i^j, $\mathcal{H}_{\text{défaut}}$sauf $\mathcal{H}_{\text{défaut}}[i]$)
    modèle.fit($\mathbb{X}_{train}$)
    $\mathcal{L}_i[j] \leftarrow$ modèle.evaluate($\mathbb{X}_{val}$)
  Fin Pour
  $\mathcal{H}_{\text{défaut}}[i] \leftarrow \arg\max_j(\mathcal{L}_i)$
Fin Pour

Exemple : Détaillons ci-dessous sur un exemple les étapes suives par la méthode d'optimisation successive des hyperparamètres. Sur la figure III.3c, chaque point représente un modèle entrainé. Le numéro correspond à l'ordre dans lequel les modèles sont entrainés. Pour une meilleure compréhension, il est recommandé de suivre cet exemple simultanément sur la figure III.3c.

Le modèle m est défini par les valeurs de ses 2 paramètres p_1 et p_2 : $m(p_1, p_2)$.

(i) L'utilisateur définit les paramètres à utiliser et leur valeur possible et leur valeur par défaut :
   — Paramètre 1 $\in \{\underline{3}, 5, 9, 13, 17, 21\}$
   — Paramètre 2 $\in \{3, 5, 9, 13, \underline{17}, 21\}$

(ii) Tous les modèles pour chaque valeur possible du paramètre 1 sont entrainés à la valeur des autres paramètres fixés par l'utilisateur. Ce sont les combinaisons $m(3,17)$, $m(5,17)$, $m(9,17)$, $m(13,17)$, $m(17,17)$, $m(21,17)$

(iii) Le meilleur modèle, c'est-à-dire celui avec la meilleure métrique, pour l'optimisation du paramètre 1 est le modèle $m(17,17)$, celui pour lequel le paramètre 1 vaut 17. C'est cette valeur du paramètre 1 qui sera utilisée par défaut pour les modèles suivants.

(iv) La deuxième série de modèles pour l'optimisation de paramètre 2 est entrainée à la valeur du paramètre 1 fixée par l'étape précédente $m(17,3)$, $m(17,5)$, $m(17,9)$, $m(17,13)$, $m(21,5)$. Le modèle $m(17,17)$, c'est-à-dire le meilleur modèle de l'étape précédente, n'a pas besoin d'être ré-entrainé puisqu'il a déjà été entrainé lors de l'étape précédente.

(v) Le meilleur modèle de tous est celui qui obtient la meilleure métrique pour l'optimisation du dernier paramètre soit le modèle $m(17,13)$ dans cet exemple.

En pratique, la valeur par défaut d'un paramètre est la première valeur dans la liste des valeurs à explorer.

Remarque 8 : Et l'optimisation successive ?

Reprenons l'exemple précédent d'une recherche par grille de 5 paramètres pouvant prendre 5 valeurs.

Pour optimiser le premier paramètre 5 entrainements sont à réaliser. Ensuite, pour chaque paramètre suivant, il ne reste plus que 4 entrainements à réaliser soit un total de $5+4*4=\mathbf{21}$ entrainements au total.

On passe donc de 3125 heures d'entrainements à 21 heures soit une division du temps d'entrainement par un facteur 148.

D NEW Outil algorithmique pour la comparaison des tâches prétextes, des architectures neuronales, des algorithmes de détection d'anomalies et de leurs combinaisons

Dans le modèle StArDusTS proposé section B, deux blocs clés sont mis en avant. Un premier, permettant d'apprendre une représentation des données sans avoir besoin d'annotation manuelle et un second permettant de réaliser la détection d'anomalies, tâche d'intérêt. Cependant, le nombre de méthodes *équivalentes* permettant de réaliser ces tâches intermédiaires sont nombreuses et le nombre de combinaisons possibles est d'autant plus élevé.

Nous avons ainsi voulu développer un outil capable de comparer les différentes combinaisons de méthodes d'apprentissage de représentation, et de détection d'anomalies. Afin de diminuer la quantité totale de comparaisons à réaliser et de modèles à entrainer et optimiser, l'optimisation des blocs d'apprentissage de représentation doit pouvoir être indépendante de l'utilisation des blocs de détection d'anomalies. Cette optimisation indépendante est possible grâce à l'hypothèse 1.

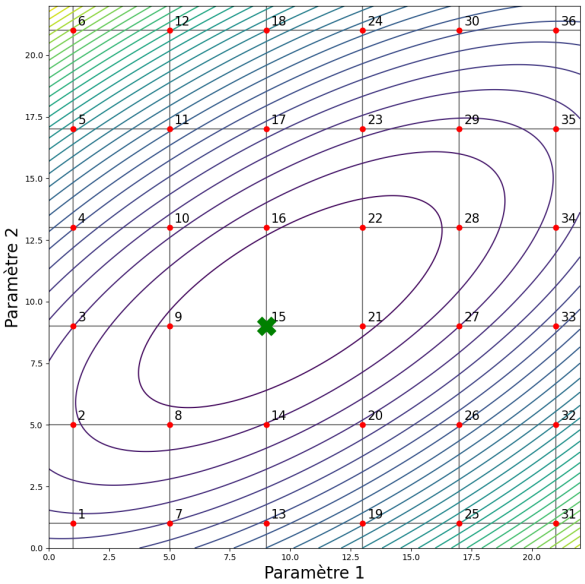
Le développement de cet outil a été réalisé avec en tête un certain nombre de fonctionnalités nécessaires à sa bonne utilisation :

— L'outil doit pouvoir déterminer automatiquement les meilleures valeurs d'hyperparamètres des réseaux de neurones,
— Il doit être assez générique pour pouvoir être utilisable sur n'importe quel jeu de données,
— L'espace des briques méthodologiques utilisées doit pouvoir être agrandi, c'est-à-dire rajouter de nouvelles méthodes facilement,
— Les méthodes à comparer sont sélectionnées à l'aide de fichiers de configuration,
— La méthode d'optimisation successive des hyperparamètres présentée section C doit être implémentée pour la recherche des meilleurs hyperparamètres d'une architecture neuronale

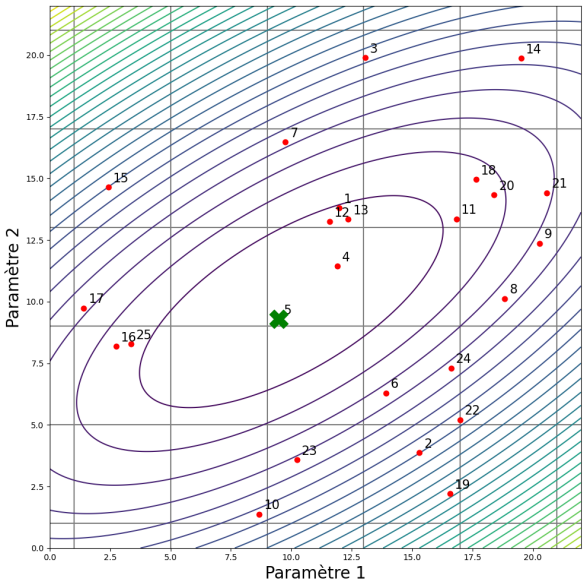
L'annexe A propose une description exhaustive des classes et méthodes disponibles dans cet outil de comparaison. Un total de 5 tâches prétextes, 6 architectures de réseaux de neurones avec de nombreux hyperparamètres (de 4 à 12 en fonction des architectures) ainsi que 4 algorithmes de détection d'anomalies sont aujourd'hui implémentés dans l'outil de comparaison.

Cet outil a été utilisé à de nombreuses reprises pour mener les expériences présentes dans ce manuscrit. Il sera rappelé au cours du manuscrit quand cet outil a été utilisé.

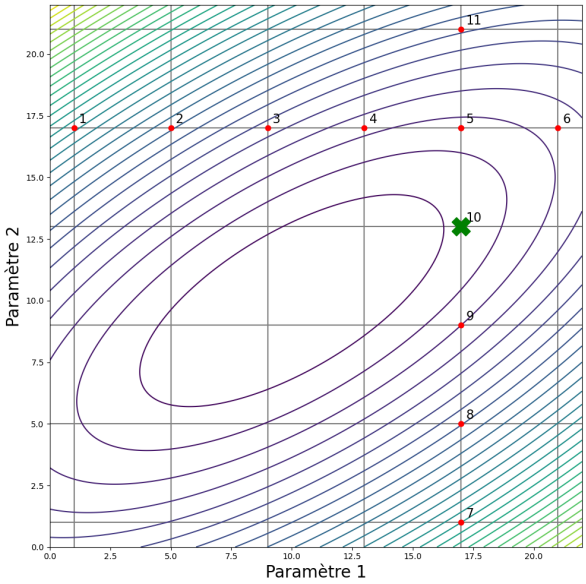
Il a par ailleurs été utilisé au sein du laboratoire d'accueil pour mener d'autres expériences et de la recherche d'architecture neuronale pour la prédiction de série temporelle.



(a) Exemple de recherche par grille. Le meilleur modèle trouvé est le n°15



(b) Exemple de recherche aléatoire. Le meilleur modèle trouvé est le n°5



(c) Exemple de recherche par optimisation successive à 2 paramètres Le meilleur modèle trouvé est le n°10

Figure III.3 : Comparaison de la recherche par grille (III.3a), de la recherche aléatoire (III.3b) et de la recherche par optimisation successive (III.3c) sur un exemple naïf à 2 paramètres. La meilleure combinaison d'hyperparamètre est définie par la croix verte sur ces figures

CHAPITRE

IV

Application : données cellulaires

Table des matières

A Introduction

Nous présentons dans ce chapitre une application des méthodes et algorithmes développés dans le chapitre III. Cette application est principalement orientée autour de la détection de cellules anormales à partir de leur masse sèche.

Nous présentons tout d'abord le contexte applicatif de la microscopie sans lentille et le calcul de la masse sèche de cellule dans cette introduction. Dans un second temps, la manière dont sont générés les jeux de données utilisés dans les expériences est présentée. Les deux sections C et D constituent le cœur des expériences menées sur les données cellulaires. La section C s'attarde sur l'application du modèle StArDusTS avec un apprentissage de représentation effectué grâce à un CNN à une dimension (CNN 1D) et une tâche prétexte de prédiction du futur de la masse sèche des cellules. Cette représentation est ensuite couplée à deux types de détecteurs à seuil. La section D est une étude comparative de différentes architectures de réseaux de neurones pour la tâche prétexte de prédiction de séries temporelles. Une conclusion générale du chapitre est proposée section E.

A.1 Microscopie sans lentille

Le CEA LETI développe depuis 2009 une technologie de microscopie sans lentille innovante. Cette dernière, en rupture avec les technologies de microscopie classiques, permet d'obtenir des images à haute définition, avec une résolution d'environ $2\,\mu\mathrm{m}$ à $4\,\mu\mathrm{m}$, et à très grand Champ de vision (FOV)*[MML+13]. Ce FOV d'environ $30\,\mathrm{mm}^2$ permet l'acquisition d'images contenant des dizaines de milliers de cellules au sein d'une même image ce qui n'est pas possible avec des méthodes traditionnelles qui ont des FOV plus faibles dus aux systèmes de lentilles utilisés. Par ailleurs, ce microscope sans lentille étant bien plus compact qu'un microscope classique (figure IV.1), il peut facilement être utilisé dans un incubateur. Il permet de contrôler parfaitement l'environnement de la culture cellulaire, avec une température de $37\,^{\circ}\mathrm{C}$, un taux de CO_2 de $5\,\%$ et à $100\,\%$ d'humidité par exemple. Des images peuvent alors être acquises de manière régulière, dans notre cas toutes les 10 minutes, pour suivre l'évolution de la culture cellulaire.

Figure IV.1 : Dispositif de microscopie sans lentille

*. Field Of View

Le système d'émission d'un microscope sans lentille émet une lumière grâce à une Diode ÉlectroLuminescente (DEL) Rouge Vert Bleu (RVB), placée au-dessus de l'échantillon. La lumière est ensuite diffractée par l'échantillon biologique générant une figure holographique capturée par le capteur CMOS placé en dessous de l'échantillon. Cette figure d'interférence n'est pas directement interprétable. Un premier algorithme [AMV+17] permet de reconstruire l'image large FOV des cellules. La figure IV.2 montre un exemple d'image reconstruite, chacun des points blancs sur l'image est une cellule de l'échantillon observé.

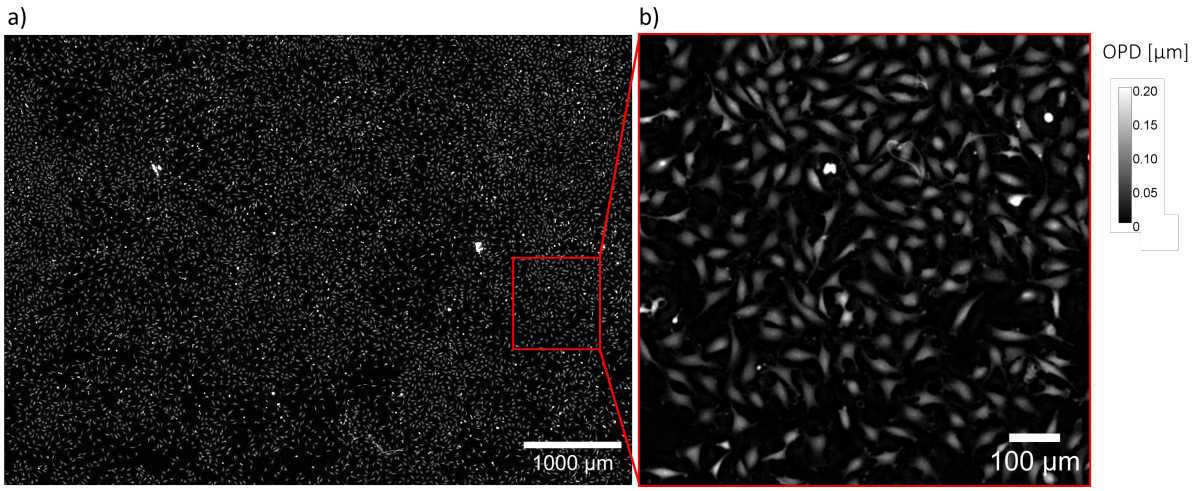


Figure IV.2 : (a) Image de microscopie sans lentille reconstruite complète et (b) extrait d'une zone de 30 mm². Chaque point blanc sur l'image est une cellule

Un second algorithme [AMV+17] est ensuite utilisé afin de segmenter, c'est-à-dire d'identifier une masse de pixels blancs sur chaque image et de l'associer à une cellule. Ce dernier repose sur des méthodes de segmentation par ligne de partage des eaux [NS94]. Des images étant acquises toutes les 10 minutes, ces segmentations sont ensuite comparées afin de relier une cellule à l'instant t à la même cellule sur l'image suivante. C'est le rôle de l'algorithme de suivi * [CSd+14].

Finalement, pour chacune de ces cellules, l'algorithme extrait différentes informations telles que la position (x, y) de la cellule dans l'image, sa surface, son épaisseur, sa vitesse ou encore sa masse sèche, qui va nous intéresser dans cette étude. L'intérêt de la masse sèche d'une cellule et son calcul à partir de l'image acquise par le microscope sans lentille sont détaillés dans la section suivante.

A.2 Masse sèche des cellules

La masse sèche d'une cellule est l'évaluation de la masse de la cellule si elle a été vidée de toute son eau. Cette métrique représente donc le contenu cellulaire, en particulier les protéines, l'ADN, l'ARN et les lipides contenus dans la cellule. La phase détectée grâce au microscope sans lentille est proportionnelle à la densité optique des couches traversées. L'intégration sur la surface de la cellule permet de mesurer la masse sèche de la cellule $\mathcal{C}$:

$$\text{Masse sèche}_{\mathcal{C}} = \iint_{x,y \in \mathcal{C}} \frac{\lambda \cdot \phi(x,y)}{2\pi \cdot \alpha(x,y)} dx \cdot dy \tag{IV.1}$$

Avec $\phi(x, y)$ le décalage de phase reconstruit (voir figure IV.1), $\lambda = 450nm$ la longueur d'onde et α un coefficient relié au changement de l'indice de réfraction de la masse sèche.

*. tracking

Comme les protéines et les acides nucléiques sont les principaux contributeurs de la masse sèche et ont un indice de réfraction similaire, on définit ce coefficient $\alpha \overset{\text{def}}{=} 1.8 \times 10^{-4}\text{m}/\text{kg}$ [ZT14, PPL+08]. De la même manière, on peut calculer l'épaisseur de la cellule $\mathcal{E}$ à partir du décalage de phase maximum de la cellule selon la formule suivante :

$$\mathcal{E}_{\mathcal{C}} \simeq \frac{\lambda}{2\pi \cdot \Delta n} \max_{x,y \in \mathcal{C}} [\phi(x,y)] \tag{IV.2}$$

Avec $\Delta n = n - n_0 = 0.025$ la différence d'indice de réfraction entre le milieu de culture et les cellules.

A.3 Problématiques et intérêts

Plusieurs problématiques sont reliées au traitement de ces données, mais voici celles qui ont été identifiées comme prioritaires pour les experts :

Problématique 1 : Détection d'anomalie

La détection automatique de cellules au comportement anormal est d'une grande importance pour les experts : l'étude approfondie et la mise en évidence de telles cellules leur permet de mieux comprendre leur comportement.

Pour les images contenant plusieurs (dizaines de) milliers de cellules, il est très fastidieux pour les experts d'identifier manuellement des cellules anormales, c'est pourquoi une méthode permettant de les détecter automatiquement est nécessaire. De plus la détection manuelle de ces cellules anormales repose sur une définition bien établie de la *normalité* et l'*anormalité* des cellules, définitions n'étant pas toujours claires pour les experts en fonction des cas d'usage. La détection automatique pourrait aussi permettre une détection sans aucun *a priori* des cellules anormales. Elle pourrait donc permettre de mettre en lumière des anomalies qui ne sont pas encore identifiées comme telles et de découvrir ou de consolider des connaissances concernant le développement des cellules et en particulier des cellules anormales.

Problématique 2 : Prédiction du futur de la cellule

La prédiction du destin (ou du futur) de la cellule. Être capable de prédire le futur de la cellule et notamment sa division permet de mieux comprendre son comportement.

En effet, les mécanismes poussant la cellule à se diviser ne sont pas encore bien compris aujourd'hui [NPJ+22]. Prédire le futur de la cellule pourrait permettre de mieux comprendre son comportement. Cette problématique est donc double, c'est à la fois une question de la prédictibilité du destin des cellules, mais aussi une question de compréhension de leur comportement de manière plus globale.

Problématique 3 : Évaluation de la chaine de traitement

L'évaluation de la chaine de traitement, allant de la reconstruction des images holographiques à la génération de la série temporelle de masse sèche.

La méthode de reconstruction d'image est en constante amélioration au sein du laboratoire où elle est développée. Au travers d'une tâche alternative, nous pouvons proposer une mesure de performance de leur algorithme de reconstruction alors que les mesures

utilisées sont plutôt qualitatives que quantitatives. Par ailleurs, la détection d'anomalies de l'algorithme de reconstruction peut permettre de cibler ces anomalies afin de les corriger. Ainsi une anomalie au sens de la détection que nous cherchons à effectuer peut tout à fait être une anomalie dans la série temporelle analysée due à une erreur dans le pipeline de génération des séries temporelles de masse sèche.

Organisation du chapitre

Dans la section C, nous mettons en œuvre le modèle StArDusTS pour la détection d'anomalies pour répondre à la fois à la problématique applicative n°1 (la détection de cellules anormales) et aussi démontrer les capacités du modèle. Par ailleurs, la tâche prétexte utilisée pour apprendre la représentation des séries temporelles est une tâche prétexte de prédiction du futur de la masse sèche de la cellule qui nous permet de répondre partiellement à la deuxième problématique applicative.

Nous mettons ensuite à profit l'algorithme d'optimisation successive des hyperparamètres ainsi que l'outil d'optimisation automatique des hyperparamètres pour tester et comparer différentes architectures de réseau de neurones pour la tâche prétexte de prédiction du futur de la masse sèche des cellules section D.

Enfin, nous revenons sur les résultats obtenus lors des expériences et les discutons section E.

B Des acquisitions aux jeux de données

Nous présentions chapitre II section E la différence fondamentale entre les acquisitions et les jeux de données. Cette section est consacrée à la présentation des acquisitions cellulaires et à la génération des jeux de données associés. Ce processus est le traitement effectué pour transformer les séries temporelles de masse sèche de cellules en jeu de données, nous ne parlerons pas ici de la chaine de traitement permettant d'extraire les séries temporelles à partir des images de microscopie sans lentille, cette problématique ayant été abordée dans la section A.2.

B.1 Présentation des acquisitions

Deux lignées cellulaires ont été étudiées et ont permis de générer des acquisitions pour cette étude :

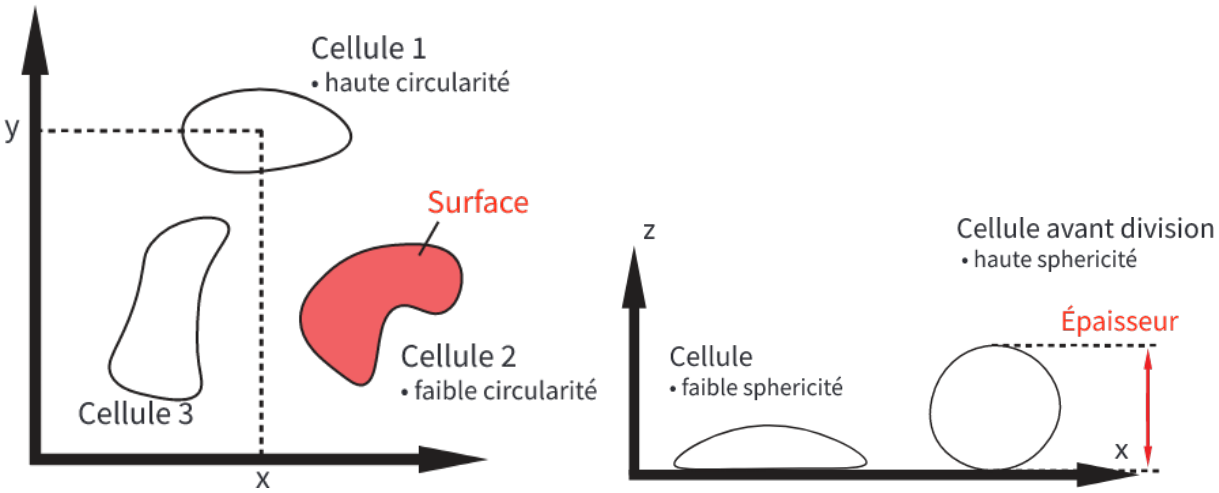
HeLa : Les cellules HeLa sont des cellules très utilisées en biologie cellulaire. Ces cellules épithéliales cancéreuses ont été prélevées sur **He**nrietta **La**cks en 1951 (sans son consentement) et ont pour propriété de continuer de se diviser à l'infini en présence d'un milieu de culture contrairement aux cellules *classiques*. On estime que plus de 50 millions de tonnes de ces cellules ont été produites depuis 1951 [Skl10]. Ce sont les cellules des acquisitions a et b. Les premières ont été acquises au sein d'un laboratoire du CEA Grenoble alors que les secondes résultent d'une expérience à l'institut Marie Curie de Paris.

Fibroblastes : Ces cellules sont des cellules présentes dans le tissu conjonctif, parfois appelées cellules de soutien, utilisées dans le domaine de la recherche. Les acquisitions c sont générées à partir de ces cellules. Les images de ces cellules ont été acquises lors d'un ensemble d'expériences mené à l'ENS Lyon. Les expériences menées sur ces cellules nous permettent d'avoir à disposition des cellules dans leur état naturel ainsi qu'une seconde série d'acquisitions pour laquelle les cellules ont été génétiquement modifiées afin de perturber leur rythme circadien.

Lors des acquisitions, une image est capturée toutes les 10 minutes par le microscope sans lentille. Pour chaque cellule dans chacune des images, une ligne est entrée dans le fichier d'acquisition. Le tableau IV.1 présente l'ensemble des données cellulaires extraites grâce à l'imagerie sans lentille. Chacune des lignes des fichiers d'acquisition contient 27 colonnes avec différentes propriétés décrivant la vie des cellules ou de leur environnement.

Colonne	Contenu
3	ID
4	Track_ID
5	Qualité
6	Position_X
7	Position_Y
8	Phase [a.u]
9	Position_T
10	N° de l'image
11	Surface [μm^2]
12	**Masse sèche [pg]**
13	Volume [μm^3]
14	Diamètre [µm]
15	Épaisseur [µm]
16	Ratio d'aspect [a.u.]
17	Longueur de l'axe principal [µm]
18	Circularité [a.u.]
19	Sphéricité [a.u.]
20	Masse sèche [pg] dans un rayon de 50 µm
21	Masse sèche [pg] dans un rayon de 100 µm
27	Vitesse [µm/min]

Table IV.1 : Contenu de chacune des lignes des acquisitions cellulaires. L'ensemble des modalités ici, hormis la masse sèche, sont présentées à titre indicatif, mais ne seront pas utilisées



(a) Vue du dessus et description de la surface et de la circularité des cellules

(b) Vue de côté et description de la sphéricité et de l'épaisseur des cellules

Figure IV.3 : Schémas vus de dessus et de côté de cellules et de certaines de leurs propriétés extraites dans les acquisitions [Bai20]

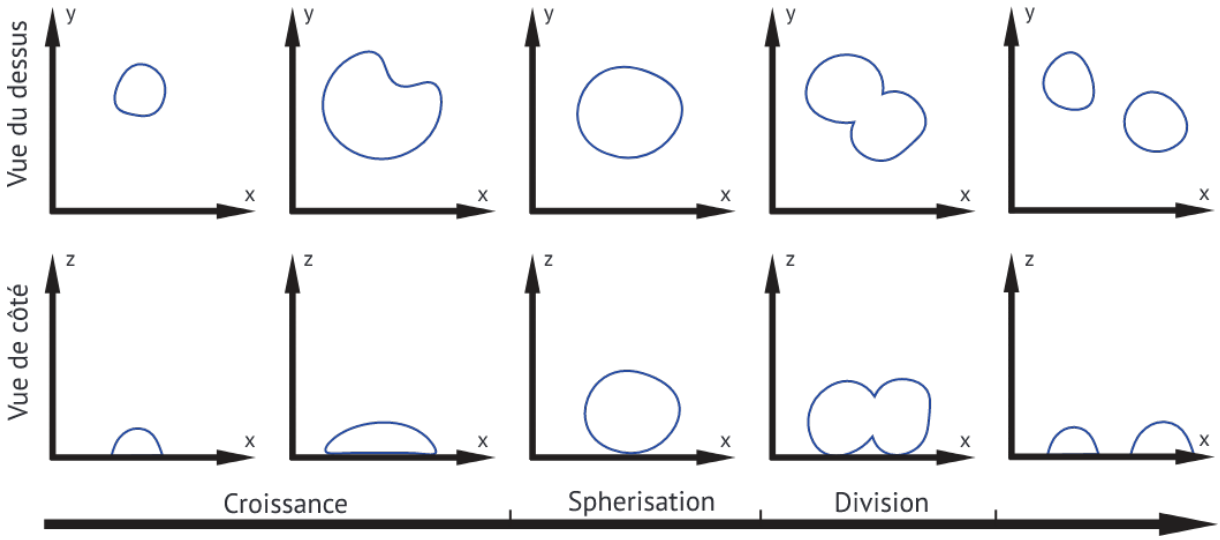


Figure IV.4 : Schéma de la division d'une cellule. Dans une première phase la cellule croit : sa surface augmente dans le plan de la boite de pétri et sa masse augmente elle aussi. Juste avant la division, la cellule se sphérise, son épaisseur augmente significativement puis se divise en 2 cellules filles [Bai20]

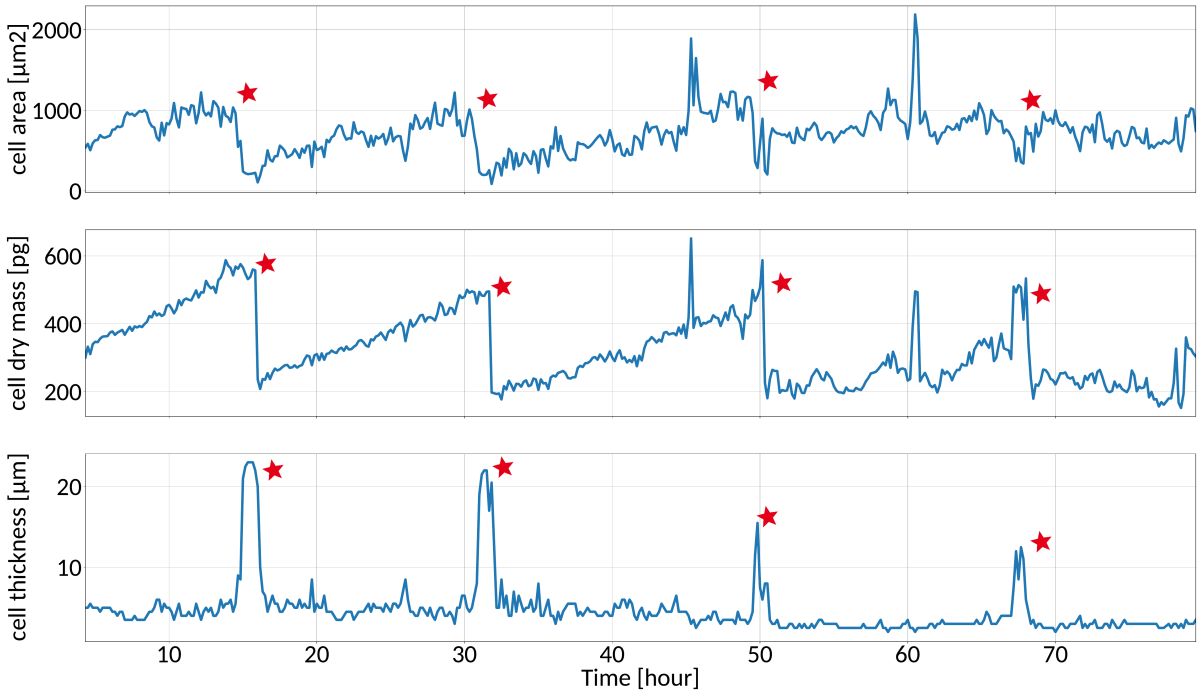


Figure IV.5 : Exemple sur une cellule de 3 modalités. De haut en bas sa surface (en μm^2), sa masse sèche (en pg), et son épaisseur (en μm). Ces observations sont réalisées sur 80 heures. On peut y observer 4 divisions cellulaires (repérées par des *) correspondant à la diminution brutale de la surface et de la masse sèche et un pic sur l'épaisseur

Les figures IV.3 et IV.4 montrent sur des schémas certaines des propriétés importantes des cellules. En particulier la figure IV.4 montre le comportement usuel d'une cellule au cours de sa vie à la fois en vue du dessus et de côté. Les cellules commencent par une phase de croissance pendant laquelle leur masse augmente. Une fois la masse *assez élevée* (il n'y a pas encore aujourd'hui de consensus sur les mécanismes exacts qui déclenchent l'étape suivante), la cellule rentre dans un processus de division. Afin de se diviser, la cellule

qui était aplatie devient sphérique, augmentant son épaisseur, mais diminuant sa surface. Le matériel génétique des futures cellules filles migre vers les extrémités de la cellule mère qui finit par se diviser symétriquement en deux. L'une de ces deux cellules est identifiée comme une nouvelle cellule dans les acquisitions (avec un nouvel `ID`) tandis que l'autre conserve l'`ID` de la mère. La figure IV.5 montre les séries temporelles de surface, masse sèche et épaisseur. On peut y voir 4 divisions cellulaires soulignées par les étoiles rouges.

B.2 Génération des jeux de données

Les acquisitions présentées dans la section précédente vont être utilisées pour 4 expériences présentées sections C.1 à C.4. Nous détaillons ici le prétraitement effectué afin de passer des données brutes aux jeux de données utilisés dans les sections suivantes. Le but sera d'appliquer le modèle StArDusTS avec une tâche prétexte de prédiction pour détecter les anomalies cellulaires, uniquement à partir de leur masse sèche.

Grâce aux acquisitions, nous avons calculé la durée moyenne d'un cycle de division qui s'élève à 17.1 ± 2.3 heures pour les cellules HeLa. Or, la tâche prétexte que nous cherchons à réaliser est une tâche de prédiction. Nous avons donc sélectionné, en collaboration avec les experts du domaine et avec précaution, la taille de la fenêtre connue pour les prédictions afin de toujours inclure une division cellulaire dans la fenêtre d'entrée. L'idée implicite derrière ce choix est de toujours pouvoir se situer au sein du cycle cellulaire en se référant à la division précédente. En effet, si l'on opte pour une fenêtre trop réduite, cela pourrait entrainer une rareté de l'événement de division cellulaire, ce qui serait alors difficile à assimiler pour les réseaux de neurones profonds. Le choix d'une fenêtre trop grande quant à lui nous forcerait à retirer les cellules dont la durée d'acquisition est inférieure à la taille de la fenêtre. Au regard de ces considérations, la taille de la fenêtre d'entrée est fixée à 20 heures (120 points) et les 10 prochaines heures du signal seront à prédire. Seules les séries temporelles de plus de 30 heures sont conservées, les autres sont retirées des jeux de données.

Les séries temporelles de masse sèche présentent une grande quantité du bruit, dû à la fois à la méthode d'acquisition et aux algorithmes de segmentation et de tracking utilisés pour les calculer. Ce n'est cependant pas la prédiction du bruit de mesure qui nous intéresse, mais plutôt la tendance du signal : la cellule va-t-elle grossir ou se diviser ? C'est pourquoi nous avons choisi de lisser la cible de la prédiction grâce à une moyenne glissante d'une heure. En effet, nous ne voulons pas pénaliser un modèle capable de prédire la tendance du signal, mais pas son bruit. Cela permet d'obtenir des métriques de prédiction plus en accord avec la tendance du signal et moins influencée par le bruit de ce dernier.

Pour chaque acquisition, 80% des séries temporelles des cellules suivies sont associées au jeu d'entrainement, 10% au jeu de validation et 10% au jeu de test, comme présenté Chapitre II section E.1. Chacun de ces jeux de données est ensuite standardisé indépendamment (chapitre II section E.2) puis est augmenté selon la procédure de fenêtrage glissant. Sauf cas particulier, le pas de la fenêtre glissante est d'un point entre deux fenêtres consécutives.

Les acquisitions a et b sont réalisées dans des instituts différents. Bien qu'elles proviennent de la même lignée cellulaire (HeLa), des différences peuvent être attendues sur ces acquisitions, c'est pourquoi elles servent à générer deux jeux de données indépendants $\mathcal{A}$ et $\mathcal{B}$. Deux versions différentes sont générées pour le jeu de données $\mathcal{B}$; $\mathcal{B}_1$ et $\mathcal{B}_{180}$ respectivement augmentés grâce à une fenêtre glissante avec un pas de 1 point (10 minutes) et 180 points (30 heures). Les validations des résultats obtenus nécessitant forcément l'intervention d'un expert, l'utilisation d'un jeu de données avec moins de fenêtres permet de minimiser la quantité de travail d'annotation manuel à réaliser par l'expert. C'est pourquoi,

dans un premier temps un jeu de données avec un pas d'augmentation de 180 points est utilisé.

De plus, deux boites de pétri avec des fibroblastes sont utilisées pour obtenir les acquisitions c et c'. La différence entre les c et c' réside dans la modification d'un gène pour les fibroblastes dans l'acquisition c'. Ces deux acquisitions sont utilisées pour générer respectivement les jeux de données $\mathcal{C}$ et $\mathcal{C}'$. Ils sont utilisés comme référence dans l'expérience section C.4. Le tableau IV.2 détaille la répartition, en nombre de cellules ainsi qu'en nombre de fenêtres de 30 heures, de ces jeux de données. On y remarque que $\mathcal{C}'$ ne sera utilisé que pendant la phase de détection d'anomalies et n'est donc composé que d'un jeu de test.

Table IV.2 : Distribution des jeux d'entrainement, de validation et de test pour les 4 jeux de données. Pour chacun, le nombre total de cellules avant prétraitement, le nombre de cellules avec une durée de vie supérieure à 30 heures et le nombre de fenêtres de 30 heures générées est indiqué.

Nom	Lignée	Lieu	Nombre de	Total	Entrainement	Validation	Test
a	HeLa	CEA Grenoble	Cellules	36396	29118	3639	3639
			Cellules >30 h	6700	5520	621	559
$\mathcal{A}$			Fenêtres 30 h	559141	440930	60179	58032
b	HeLa	Institut Marie Curie Paris	Cellules	27528	22024	2752	2752
			Cellules >30 h	7283	5859	706	718
$\mathcal{B}_1$			Fenêtres 30 h	834918	665280	86363	83275
$\mathcal{B}_{180}$			Fenêtres 30 h	9158	/	/	9158
c	Fibroblasts Wild Type	École Normale Supérieure Lyon	Cellules	28243	22596	2823	2824
			Cellules >30 h	3986	3169	404	413
$\mathcal{C}$			Fenêtres 30 h	156381	123950	16259	16172
c'	Fibroblasts Knock Out	École Normale Supérieure Lyon	Cellules	5035			5035
			Cellules >30 h	315			315
$\mathcal{C}'$			Fenêtres 30 h	26688			26688

Nous avons choisi avec ces données de nous confronter à un véritable problème de détection d'anomalies. Le choix de ce sujet applicatif vient cependant avec une contrepartie non négligeable : le fait qu'aucune annotation n'est disponible afin de valider si les cellules que nous détectons sont effectivement anormales ou non. Les cellules détectées comme anormales sont donc présentées à un expert qui doit, après notre détection, déterminer si elles sont normales ou anormales. Il n'est cependant pas possible, même pour un expert du domaine, de déterminer si une cellule est anormale uniquement à partir de sa série temporelle de masse sèche. Grâce aux `ID` des cellules, il est cependant possible de remonter à la vidéo à partir de laquelle est générée la série temporelle de masse sèche qui facilite alors l'annotation.

Cette méthode a ses défauts et ne permet notamment pas de calculer le rappel, c'est-à-dire le nombre d'anomalies manquées dans le jeu de données. C'est à cause de la contrainte de non-labellisation que nous nous sommes imposés. L'avantage est que le modèle proposé ne repose à aucun moment sur des annotations.

C Application de StArDusTS pour la détection d'anomalies cellulaires

Il est aujourd'hui difficile pour les experts du domaine d'identifier automatiquement des cellules anormales au sein d'un jeu d'acquisitions comme celui présenté section B. Nous cherchons au travers d'une méthode ne nécessitant aucune annotation manuelle à détecter des cellules anormales. En effet, la définition d'une cellule anormale est encore aujourd'hui vague et c'est pourquoi nous cherchons à bénéficier de la puissance des réseaux de neurones pour les détecter.

Nous proposons 4 expériences progressives afin d'évaluer les performances de StArDusTS pour la détection d'anomalies. L'apprentissage de représentation y est effectué grâce à une tâche prétexte de prédiction. La première expérience section C.1 valide l'efficacité de StArDusTS couplé à un détecteur à seuil sur une seule fenêtre. Dans un second temps, l'expérience section C.2 valide l'efficacité d'un second détecteur, le score d'anomalie, afin d'obtenir des résultats de détection d'anomalies plus exhaustifs et robustes. Ensuite les capacités de généralisation de la représentation apprise sont mesurées section C.3, toujours grâce au détecteur à score. Enfin l'expérience section C.4 met en évidence les limitations de la tâche prétexte de prédiction et du modèle. Pour chacune de ces sous-sections, nous présentons d'abord le protocole expérimental puis les résultats obtenus ainsi que leur analyse.

C.1 Détection d'anomalies par seuil

Le contenu de cette section a été présenté à la conférence ASPAI 2021 sous le nom *"Deep Anomaly Detection Using Self-Supervised Learning : Application to Time Series of Cellular Data"* [BMA+21]

C.1.1 Problématique

Le modèle StArDusTS proposé permet-il de détecter des anomalies cellulaires à partir de la masse sèche de cellules ?

C.1.2 Protocole

Architecture : L'architecture de réseaux de neurones choisie pour cet ensemble d'expériences est une architecture CNN 1D convolutive à 1 dimension. Ce réseau de neurones est constitué d'un encodeur contenant 3 ensembles de couches convolutives de `64` filtres de taille 3. Ces couches convolutives sont suivies de fonctions d'activation tangente hyperbolique `tanh`. Chaque ensemble de 3 couches de convolution est suivi d'une couche de `maxpooling` de facteur 2. Le décodeur est lui constitué de 3 couches denses de taille `64`, `32` et `60` pour la couche de sortie. Ces valeurs d'hyperparamètres correspondent aux meilleures valeurs obtenues grâce à l'outil présenté chapitre III section D.

Tâche prétexte : Le modèle est entrainé à prédire les 10 prochaines heures de signal (60 points) connaissant les 20 premières (120 points). Ce modèle est entrainé grâce aux jeux d'entrainement et de validation du jeu de données $\mathcal{A}$.

Détecteur d'anomalie : Dans cette expérience, le détecteur utilisé dans le modèle StArDusTS est un détecteur à seuil sur une fenêtre (présenté chapitre III section B.3.1). Si la

prédiction s'éloigne trop de la vérité terrain, la cellule est considérée comme anormale. La figure IV.6 illustre sur un exemple le principe du détecteur à seuil appliqué aux séries temporelles de masse sèche de cellules.

La valeur du seuil du détecteur est fixée de manière à considérer les cellules en dehors de l'intervalle de confiance a 97.5% comme anormale. Il est calculé grâce au jeu d'entrainement $\mathcal{A}_{\text{train}}$.

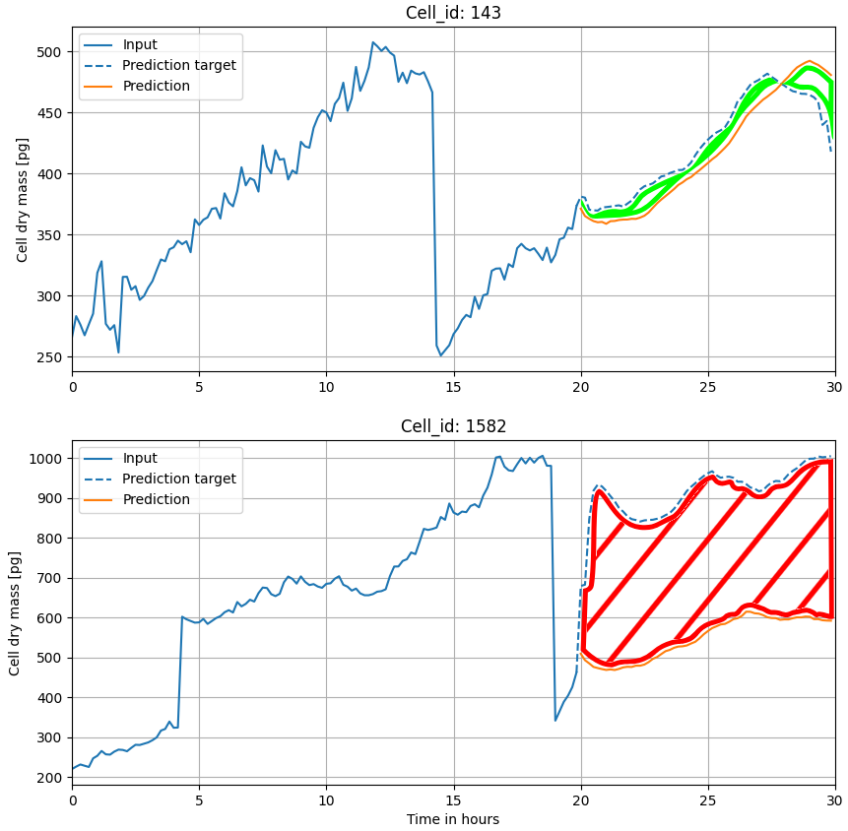


Figure IV.6 : Illustration sur un exemple normal (en haut) et anormal (en bas) du détecteur à seuil. Si l'EQM entre la prédiction (en orange) et la vérité terrain (en bleu pointillé) dépasse le seuil τ, la fenêtre est considérée comme anormale (Attention, la figure ci-dessus est une illustration, l'EQM n'est pas à proprement parler l'aire entre les deux courbes)

Jeux de données : Pour cette première expérience, le jeu de données sur lequel les anomalies sont détectées est $\mathcal{B}_{180}$ avec un pas du fenêtrage glissant pour l'augmentation de données de 30 heures (180 points). Ce choix permet d'alléger l'annotation manuelle que doit réaliser l'expert tout en permettant quand même de prouver que le modèle StArDusTS est capable de détecter des anomalies.

La totalité du modèle StArDusTS ainsi que les jeux de données utilisés sont résumés sur le schéma figure IV.7.

Table IV.3 : Résumé des données et modèle de l'expérience de détection d'anomalies avec un détecteur à seuil

Entrainement	Validation	détection d'anomalies	Modèle	Détecteur
$\mathcal{A}_{1,\text{train}}$	$\mathcal{A}_{1,\text{validation}}$	$\mathcal{B}_{180}$	α	à seuil

C.1.3 Résultats

Le seuil de détection des séries temporelles anormales calculé à $\tau = 230.87$ pg permettant de lever 208 séries temporelles anormales parmi les 9158 séries temporelles de $\mathcal{B}_{180}$. Pour rappel, le seuil est fixé de manière à détecter comme anormales les prédictions à l'extérieur de l'intervalle de confiance à $95\,\%$.

C.1.4 Analyses

Tout d'abord, cette étude a permis d'identifier 2 causes d'anomalies principales :

Anomalies biologiques : Ces anomalies pourront aussi être appelées anomalies cellulaires dans la suite du document. Comme ces cellules ont un comportement anormal, cela se retranscrit sur leur masse sèche. Elle ne peut alors pas être correctement prédite par le modèle qui détecte la cellule comme anormale. La figure IV.8 présente sur un exemple une fusion cellulaire détectée automatiquement. La cellule considérée (en bleu) fusionne avec une autre cellule (grise figure IV.8a) puis la cellule résultante se divise en 3 cellules filles (figure IV.8c) qui re-fusionnent (IV.8d).

Anomalies d'acquisition : Ces anomalies sont générées par les algorithmes en amont ayant permis d'extraire les séries temporelles à partir des vidéos de cellules. Elles sont, par abus de langage, nommées "anomalies de mesure", mais peuvent provenir à la fois de l'acquisition des images, de leur segmentation automatique ou d'un mauvais tracking des cellules une fois la segmentation réalisée. La détection automatique de ces anomalies peut contribuer à l'amélioration des algorithmes utilisés pour la génération des images à partir de figures de diffraction, mais aussi des algorithmes de segmentation et de suivi.

De plus, certaines anomalies sont difficiles à classifier, car ce sont des anomalies d'acquisition résultant d'anomalies cellulaires. Dans ce cas, la présence d'une cellule anormale rend la segmentation des cellules plus difficile pour l'algorithme utilisé pour la génération séries temporelles de masse sèche.

Enfin, parmi les anomalies qui sont détectées par le modèle, certaines cellules ont un comportement normal. Ces détections sont donc des faux positifs, pour lesquelles le modèle classifie à tort des cellules normales comme anormales.

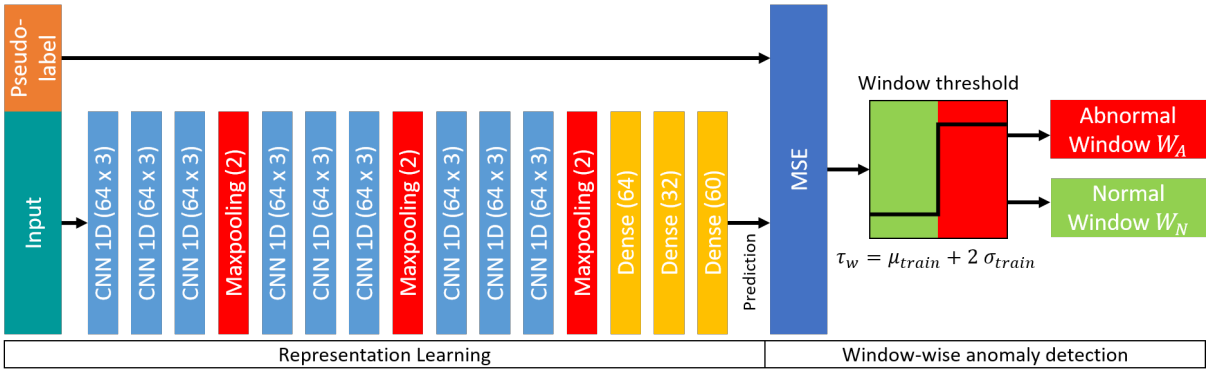


Figure IV.7 : Schéma de l'architecture et du détecteur d'anomalie à seuil simple

Remarque 9 : L'annotation, combien ça coute ?

L'annotation manuelle est obligatoirement réalisée par un expert des données cellulaires contrairement à une annotation classique pour de la vision par ordinateur où *n'importe qui* peut identifier ou segmenter le contenu d'une photographie.

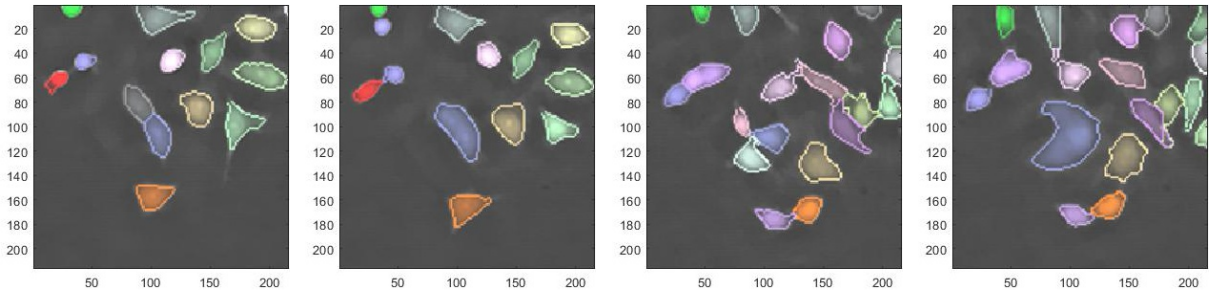
Il faut en moyenne 2 heures à l'expert pour annoter une centaine de vidéos de cellules.

L'étude présentée dans le post de blog [Ipe10] présente quelques informations sur les couts moyens de l'annotation sur le Turc mécanique d'Amazon. Ce dernier est une plateforme en ligne permettant aux entreprises de déléguer des tâches simples et répétitives à des travailleurs humains à travers le monde pour effectuer des micro-tâches moyennant rémunération. Les tâches sont réparties en petits lots appelés "HIT" (Human Intelligence Tasks), et les travailleurs réalisent ces tâches pour gagner de l'argent en fonction du nombre de HITs accomplis. Elle est en particulier utilisée pour l'annotation manuelle de données.

Les 208 séries temporelles identifiées comme anomalies par le modèle StArDusTS ont été manuellement annotées par l'expert du domaine de manière à pouvoir évaluer les performances du modèle. Ce sont $97\,\%$ de ces séries temporelles qui sont effectivement des anomalies avec 66% d'anomalies cellulaires et 31% d'anomalies d'acquisition. Seulement $3\,\%$ des fenêtres détectées comme anormales étaient en réalité normales, ce qui témoigne de l'excellente précision du modèle. Les résultats complets de l'analyse manuelle *a posteriori* effectuée par l'expert sont résumés tableau IV.4.

Table IV.4 : Résultats de l'annotation manuelle de l'expert

Anomalie	Cellulaire	de Segmentation	Faux positif
Ratio	66% (137)	31 % (65)	3% (6)
Total		**97%**	**3%**



(a) La cellule en bleu au centre de l'image se rapproche de la cellule grise ($t = 330\,\mathrm{min}$).

(b) Fusion des deux cellules en une seule ($t = 4h10$).

(c) Division de la 'super-cellule' en 3 cellules filles ($t = 18h40$).

(d) Fusion des 3 cellules filles en une super-cellule ($t = 20h$).

Figure IV.8 : Suivi de la cellule 1582 (bleu) au comportement anormal. Elle fusionne avec une autre cellule (grise figure IV.8a) puis la cellule résultante se divise en 3 cellules filles (figure IV.8c) qui fusionnent (IV.8d)

Limitations du détecteur à seuil sur une fenêtre Cette expérience permet aussi de mettre en lumière les limitations du détecteur à seuil. Lorsque nous appliquons l'expérience sur le jeu de données $\mathcal{B}_1$ avec un pas d'augmentation de données de 1 point, la série temporelle d'une cellule de 50 heures est découpée en 120 fenêtres. En supposant

que le tirage normal/anormal soit totalement aléatoire avec une probabilité de 2% d'être anormal. En modélisant alors la classification de cette série temporelle par une loi binomiale avec une probabilité $p = 0.02$ et 120 tirages, il n'y a que 8.8% de chance qu'une série de 50 heures ne soit jamais détectée comme anormale.

Cet effet est notamment visible sur l'histogramme d'EQM figure IV.9. Sur cet histogramme, chacune des valeurs d'EQM pour chaque fenêtre est représentée. En bleu les fenêtres sont normales (EQM $< \tau$) et en orange anormales (EQM $> \tau$).

Cependant, parmi ces fenêtres, certaines cellules n'ont été détectées qu'une seule fois comme anormale et d'autres pour lesquelles la métrique est supérieure au seuil de multiple fois. Par exemple dans deux cas extrêmes la série temporelle 790 a 48 fenêtres pendant lesquelles sa métrique est supérieure au seuil τ alors que la série temporelle 13834 n'a qu'une seule fenêtre pendant laquelle elle est détectée comme anormale.

Les fenêtres en vert sur la figure de droite correspondent aux cellules dont l'EQM est au moins une fois supérieure à τ qui pourraient être faussement considérées comme anormales. Ainsi bien que seulement 2.5% des fenêtres soient considérées anormales, la quasi-totalité (93.4%) des cellules ont au moins une fenêtre dont l'EQM est supérieur à τ.

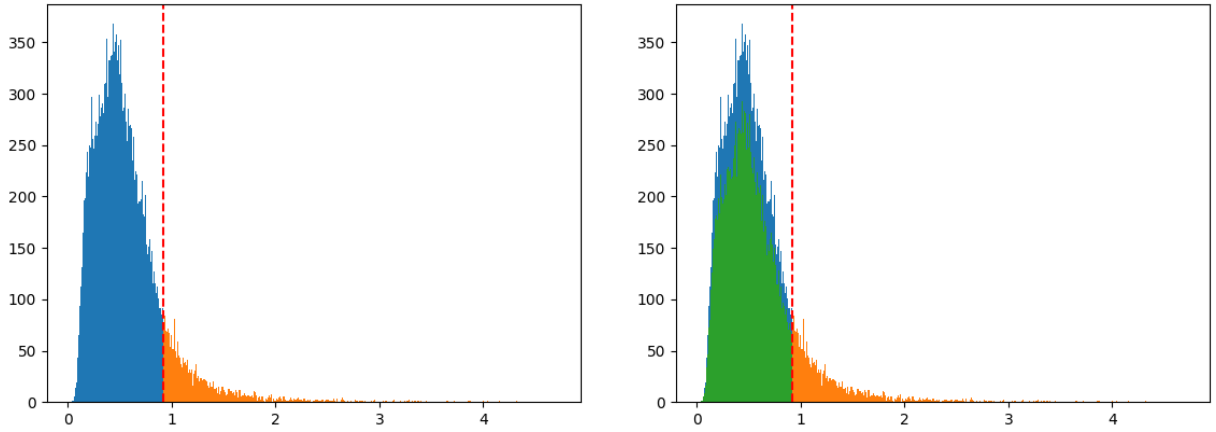


Figure IV.9 : Histogramme des EQMs de toutes les fenêtres du jeu de données. En bleu les fenêtres sont normales (EQM $< \tau$) et en orange anormales (EQM $> \tau$). Les fenêtres en vert sur la figure de droite correspondent aux séries temporelles dont le score d'anomalie est au moins une fois supérieur à τ

On ne peut donc pas considérer une unique fenêtre comme suffisante pour présenter la série temporelle complète (et donc la cellule) comme anormale.

Comme le montre cet exemple, le calcul de la métrique d'EQM ne prend pas en compte l'ensemble des fenêtres d'une même série temporelle afin de la considérer globalement. Il ne donne qu'une image locale, de l'anomalie d'une fenêtre. C'est ce constat qui nous a poussé à proposer le score d'anomalie chapitre III section B.3.2, afin d'intégrer la temporalité des données au sein du détecteur pour réaliser une détection d'anomalie plus robuste au niveau de la cellule.

C.2 Détection avec score d'anomalie

L'expérience réalisée dans la section précédente met en évidence les capacités d'apprentissage du modèle d'IA afin de détecter des séries temporelles anormales. Cependant, un jeu de données sans aucun recouvrement, B_{180}, a été utilisé. Ainsi, les séries temporelles de cellules durant 50 heures n'ont été vues qu'une seule fois par le modèle StArDusTS. Si la

cellule devient anormale durant la fin de sa vie, l'instant où la série temporelle n'est anormale peut ne pas être vu du tout par le modèle. C'est pourquoi, afin de pouvoir considérer tous les instants disponibles de la vie de la cellule, le modèle doit être appliqué sur un jeu de données avec une fenêtre glissante d'un pas de 1 point. Cela permet de prendre en compte tous les instants de la vie de la cellule dans (au moins) une fenêtre, sans exception.

Cette augmentation artificielle de la taille du jeu de données utilisé pour la détection d'anomalies introduit une nouvelle problématique : Comment compiler les différentes valeurs de métriques obtenues pour toutes les fenêtres de 30 heures d'une même cellule ? En effet, chaque fenêtre de 30 heures de la série temporelle complète d'une cellule mène au calcul d'une valeur de métrique correspondant à l'erreur entre la prédiction du réseau de neurones et la vraie valeur de la masse sèche de la cellule. Pour une cellule ayant une durée de vie de 80 heures ce sont donc 301 valeurs d'EQM qui sont calculées, une pour chaque fenêtre.

C'est pour répondre à cette problématique que nous proposons le score d'anomalie présenté section B.3.2. La détection automatique des anomalies est donc réalisée en 2 étapes :

— L'EQM de chaque fenêtre d'une cellule est calculée (Comme dans l'expérience précédente)
— Le score d'anomalie $\mathcal{S}_c$ équation (III.2) de cette cellule est calculé comme le rapport du nombre de fenêtres dont l'EQM est supérieure au seuil τ et du nombre total de fenêtres de cette cellule
— Si le score d'anomalie est supérieur à 25% la cellule est considérée comme anormale et présentée à l'expert pour annotation.

C'est donc un second bloc de détection d'anomalies, qui permet d'agréger les résultats obtenus individuellement sur chaque fenêtre grâce à la méthode de seuillage sur la prédiction en une valeur de score $\mathcal{S}_c$ pour une cellule.

🆕 **Comment visualiser les anomalies des séries temporelles ?** La figure IV.10 permet de résumer toutes les fenêtres temporelles ainsi que les valeurs de métriques associées. Alors que la figure du haut est l'ensemble de la série temporelle de masse sèche de la cellule, la figure du bas contient chacune des fenêtres de 30 heures de la série temporelle complète. La lecture de la figure du bas se fait de la gauche vers la droite, une ligne continue correspond à une fenêtre de la série temporelle de la figure du haut commençant à t_0 sur l'axe du haut et finissant à t_{end} 30 heures plus tard. La ligne noire correspond à la séparation entre les entrées du modèle et les sorties attendues (le futur à prédire). La couleur d'une courbe de masse sèche d'une fenêtre sur la courbe du bas permet de visualiser la valeur d'EQM de la fenêtre concernée. Les nuances de rouge correspondent aux fenêtres dont l'EQM est supérieure au seuil τ calculé sur le jeu d'entrainement et donc aux fenêtres *anormales* tandis que les fenêtres en nuance de bleu sont celles dont la métrique est inférieure au seuil τ et donc *normales*. Ces figures peuvent notamment servir à l'expert afin de visualiser les instants auxquels la cellule risque d'avoir un comportement anormal durant toute sa vie. Quelques exemples de ces figures sont disponibles en annexe B.

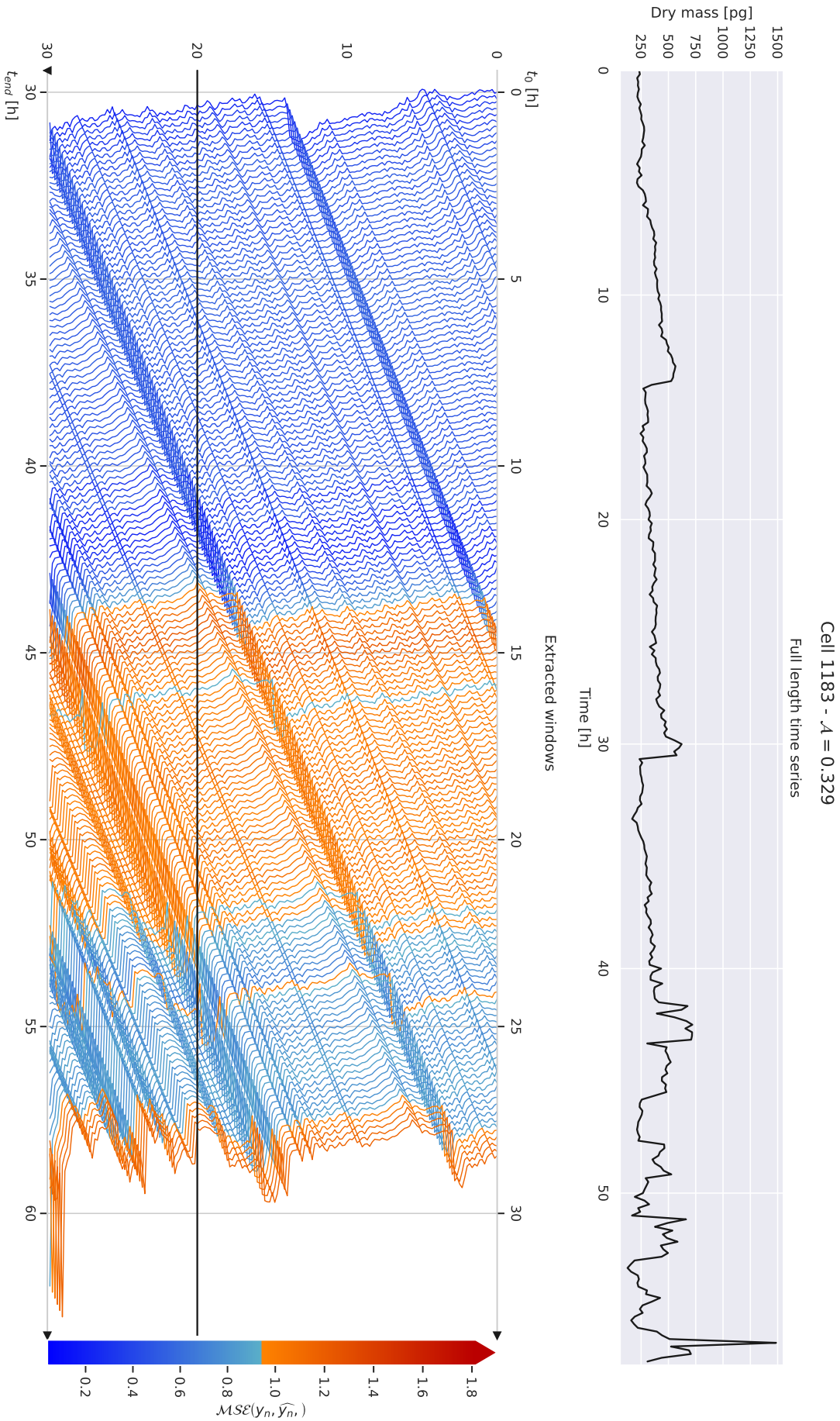


Figure IV.10 : Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule n°1183. Son score d'anomalie est de $32.9\,\%$

C.2.1 Problématique

Le but de cette expérience est de généraliser les résultats obtenus lors de l'expérience précédente section C.1 lorsque les jeux de données sont augmentés avec un pas de 1 et ainsi de valider l'efficacité du score d'anomalie. Cette version du modèle StArDusTS devrait donc être plus robuste et permettre d'être plus précis dans la détection des anomalies car les séries temporelles sont vues en entier par le modèle de détection d'anomalies.

C.2.2 Protocole

Architecture : Identique à l'expérience précédente (CNN 1D).

Tâche prétexte : Identique à l'expérience précédente (Prédiction des 10 prochaines heures du signal).

Détecteur d'anomalie : Nous présentons ici le protocole mis en place afin d'utiliser ce second détecteur, le score d'anomalie, et de mesurer ses performances de détection. Il est utilisé afin de pouvoir agréger les résultats de chaque fenêtre d'une cellule qui étaient jusqu'alors traitées indépendamment.

Jeux de données : L'expérience réalisée consiste à entrainer, valider et tester le modèle StArDusTS sur les jeux de données provenant des mêmes acquisitions, augmentés avec un pas de 10 minutes (1 point). Nous disposons à la fois des données $\mathcal{A}$ et $\mathcal{B}$ contenant les mêmes cellules HeLa avec des comportements similaires. Dans le cadre de cette expérience, un modèle α est entrainé, validé et testé sur $\mathcal{A}_1$. De manière identique, un modèle β est entrainé, validé et testé sur le jeu de données $\mathcal{B}_1$. Les modèles α et β sont tous les deux construits comme le modèle de l'expérience section C.1 avec une différence sur le détecteur d'anomalie à score d'anomalie comme le montre le schéma figure IV.11.

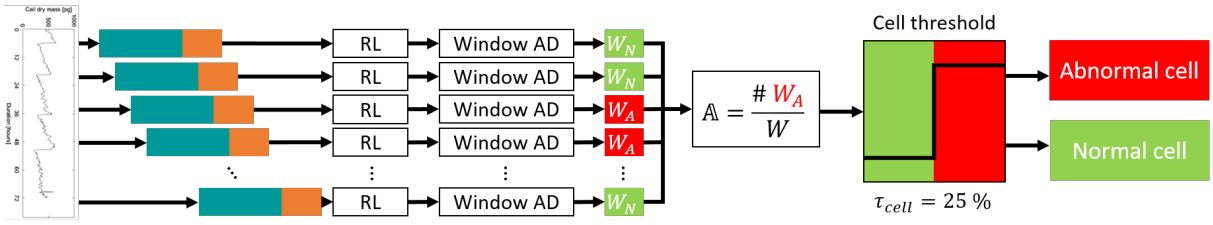


Figure IV.11 : Schéma du détecteur à score d'anomalie. Pour calculer le score d'anomalie, les EQMs de chaque fenêtre d'une série temporelle sont calculées grâce au modèle présenté figure IV.7 puis agrégées grâce au score d'anomalie (voir section B.3.2)

Table IV.5 : Résumé des données et modèles de l'expérience de détection avec score d'anomalie

Entrainement	Validation	détection d'anomalies	Modèle	Détecteur
$\mathcal{A}_{1,\text{train}}$	$\mathcal{A}_{1,\text{validation}}$	$\mathcal{A}_{1,\text{test}}$	α	à score
$\mathcal{B}_{1,\text{train}}$	$\mathcal{B}_{1,\text{validation}}$	$\mathcal{B}_{1,\text{test}}$	β	à score

C.2.3 Résultats

Le modèle proposé permet de lever 104 cellules anormales, c'est-à-dire dont le score d'anomalie $\mathcal{S}$ est supérieur à 0.25, dans l'expérience sur le jeu de données $\mathcal{A}$. À des fins de

comparaison, et pour alléger l'annotation manuelle, le même nombre (104) de cellules sont sélectionnées aléatoirement parmi les (198) cellules pour lesquelles $S > 0.25$ dans l'expérience sur le jeu de données $\mathcal{B}$ pour être annotées manuellement.

Table IV.6 : Répartition de l'annotation manuelle des anomalies de 104 cellules pour les expériences avec score d'anomalie sur les jeux de données $\mathcal{A}$ et $\mathcal{B}$

	$\mathcal{A}$	$\mathcal{B}$
Anomalies d'acquisition	49% (51)	31.1% (32)
Anomalies biologiques	36.5% (38)	38.8% (40)
Segmentation *et* biologique	2.9% (3)	3.9% (4)
Segmentation *ou* biologique	7.7% (8)	9.7% (10)
Faux positifs	3.8 % (4)	16.5% (17)
précision	**94.2%**	**83.5%**

C.2.4 Analyses

L'annotation manuelle de ces cellules a permis, comme le montre le tableau IV.6 d'identifier deux nouvelles classes d'anomalies détectées :

Les anomalies biologiques *et* d'acquisition : Une anomalie biologique peut rendre plus difficile la segmentation ou le tracking réalisé par l'algorithme en amont. L'anomalie levée par le modèle est donc à la fois une anomalie de segmentation *et* biologique.

Les anomalies biologiques *ou* d'acquisition : Dans certains cas, il n'est même pas possible pour l'expert de différencier une anomalie cellulaire d'une anomalie d'acquisition.

Parmi les 104 cellules annotées, la part d'anomalies biologiques est plus ou moins constante dans les 2 expériences avec respectivement 36.5% et 38.8% des cellules annotées. La quantité d'anomalies d'acquisition n'est pas négligeable puisqu'elle représente 49% (presque la moitié) des anomalies dans l'expérience $\mathcal{A}$ et 31.1% dans l'expérience $\mathcal{B}$. On observe cependant une grande différence entre la quantité de faux positifs présentée par le modèle α, 3.8% et celle présentée par le modèle β, 16.5%.

Par ailleurs, les figures IV.12a et IV.12b montrent sur un histogramme cumulé la répartition des classes en fonction de leur valeur de score d'anomalie. On remarque notamment que pour l'expérience $\mathcal{B}$, la majeure partie des faux positifs se situent pour les valeurs de score d'anomalie les plus faibles (inférieures à 0.5), ainsi, augmenter le seuil de score d'anomalie permettrait d'augmenter la précision globale du modèle.

L'analyse approfondie des vidéos des cellules présentant une anomalie biologique a par ailleurs mené à l'identification de 6 sous-catégories pour les anomalies biologiques :

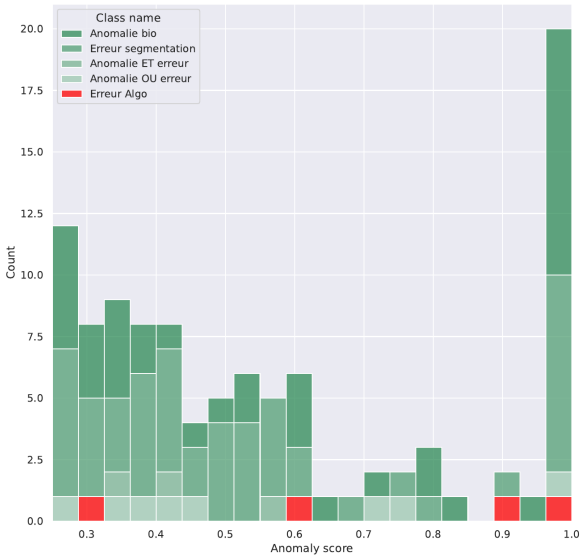
(a) Les cellules ayant une division anormale, incluant des cellules qui ne se divisent pas et stagnent sur un palier, des cellules qui se divisent de manière asymétrique ou encore des cellules qui se divisent en 3,

(b) Les cellules ayant une croissance anormale, cela peut englober une croissance trop longue, trop courte ou encore une croissance non linéaire avec des pertes de masses inexpliquées,

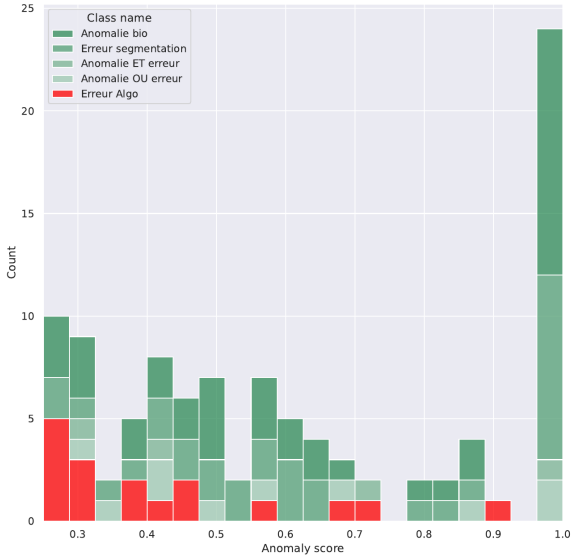
(c) Les cellules fusionnant entre elles,

(d) Les cellules anormalement grosses,

(e) Les cellules mortes,



(a) Histogramme des anomalies pour le modèle α



(b) Histogramme des anomalies pour le modèle β

Figure IV.12 : Répartition des anomalies pour l'expérience en fonction de leur score d'anomalie pour les expériences $1.\mathcal{A}$ et $1.\mathcal{B}$

(f) Les autres types d'anomalies cellulaires, tel que des problèmes de cytokinese, d'adhésion de la cellule ou de débris dans le milieu de culture.

Le tableau IV.7 détaille pour les deux jeux de données la répartition de ces anomalies biologiques. La figure IV.13 présente un exemple de chacune de ces anomalies avec à la fois les images des cellules et la série temporelle de masse sèche à partir de laquelle l'anomalie a été détectée.

Table IV.7 : Répartition de l'annotation manuelle des anomalies biologiques pour les expériences avec score d'anomalie sur les jeux de données $\mathcal{A}$ et $\mathcal{B}$

	$\mathcal{A}$ (38)	$\mathcal{B}$ (40)
(a) Division anormale	8 % (3)	15% (6)
(b) Croissance anormale	50% (19)	40% (16)
(c) Cellules fusionnant	26% (10)	5% (2)
(d) Cellules anormalement grosses	5 % (2)	20% (8)
(e) Cellules mortes	8 % (3)	20% (8)
(f) Autre	3 % (1)	0% (0)

Conclusion Nous avons montré dans deux cas distincts que le modèle StArDusTS avec score d'anomalie permet d'identifier automatiquement des anomalies, qu'elles soient cellulaires ou d'acquisition. Ces expériences ont même permis d'aller plus loin, car grâce à elles, différentes classes d'anomalies biologiques ont pu être identifiées.

Bien que les résultats du modèle β soient moins bons que ceux observés pour le modèle α, il est nécessaire de rappeler que l'objectif pour les experts du domaine n'est pas de maximiser la précision du modèle, mais bien de pouvoir détecter automatiquement, sans aucune annotation, des anomalies au sein du jeu de données. StArDusTS permet donc d'en détecter sans avoir introduit d'*a priori* humain dû à une annotation des données sur ce qui est ou non une anomalie.

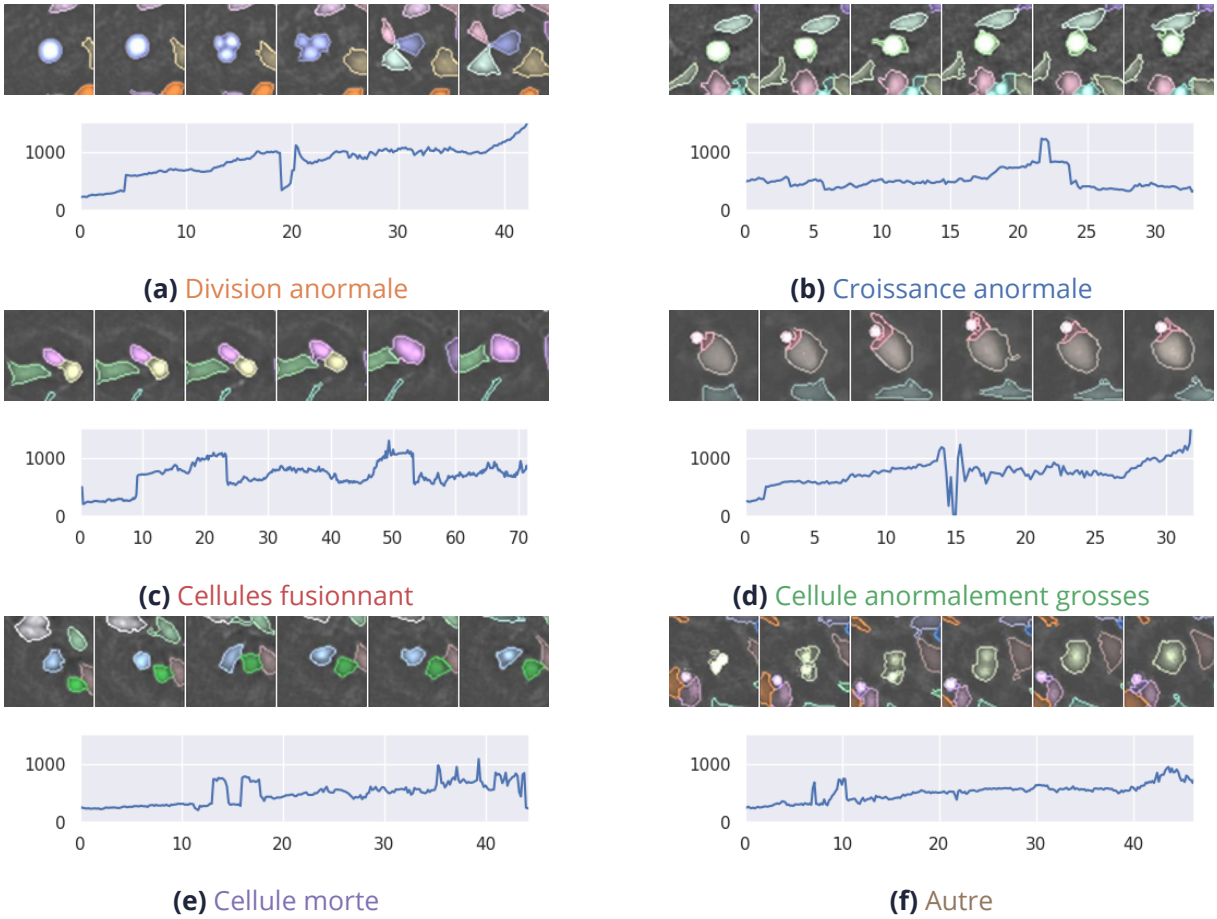


Figure IV.13 : Time-lapse de cellules Hela. Chaque image recadrée fait 100 ×100 mm². L'intervalle entre deux images est de 20 minutes. Le suivi des cellules et la segmentation cellulaire sont calculés conjointement pour obtenir cette série en accéléré. Chaque cellule présente dans l'image est identifiée avec une couleur différente. Les cellules d'intérêt sont au centre des images recadrées

C.3 Généralisation de la représentation apprise

C.3.1 Problématique

L'expérience précédente consistait à entrainer le modèle StArDusTS et à détecter les anomalies sur des cellules provenant des mêmes acquisitions. Autrement dit, le modèle α a été entrainé sur les données $\mathcal{A}_{\text{train}}$ et testé sur les cellules issues de la même culture $\mathcal{A}_{\text{test}}$ (idem pour Le modèle B et les jeux de données $\mathcal{B}_{\text{train}}$ et $\mathcal{B}_{\text{test}}$). Nous cherchons ici à mesurer si le modèle entrainé est assez générique pour être capable de détecter des anomalies provenant d'autres jeux de données.

Si le modèle est capable de détecter des anomalies cellulaires provenant de cellules sur lesquelles il n'a pas été entrainé, cela signifiera que la représentation apprise est assez robuste pour pouvoir être généralisée à d'autres acquisitions cellulaires.

C.3.2 Protocole

Architecture : Identique à l'expérience précédente (CNN 1D).

Tâche prétexte : Identique à l'expérience précédente (Prédiction des 10 prochaines heures du signal).

Détecteur d'anomalie : Identique à l'expérience précédente (Score d'anomalie).

Jeux de données : Le modèle est entrainé sur le jeu de données $\mathcal{A}$ mais testé sur le jeu de données $\mathcal{B}$. Comme dans cette expérience les jeux d'entrainement et de validation ne diffèrent pas de ceux de l'expérience précédente sur le jeu de données $\mathcal{A}$, il n'est en fait même pas nécessaire de ré-entrainer un réseau de neurones. Le modèle α de la section C.2 ayant déjà été entrainé sur le jeu de données $\mathcal{A}$ est utilisé.

Table IV.8 : Résumé des données et modèle de l'expérience de généralisation de la représentation apprise

Entrainement	Validation	détection d'anomalies	Modèle	Détecteur
$\mathcal{A}_{1,\text{train}}$	$\mathcal{A}_{1,\text{validation}}$	$\mathcal{B}_1$	α	à seuil

C.3.3 Résultats

939 cellules ont été détectées comme anormales parmi les $7\,283$ cellules du jeu de données $\mathcal{B}$. On constate que parmi ces 939 anomalies, 152 font partie des anomalies ayant été identifiées par le modèle β dans l'expérience précédente. On observe donc une consistance des modèles à identifier, au moins en partie, les mêmes cellules comme anormales bien qu'ils aient été entrainés sur des jeux de données différents.

Le tableau IV.9 montre les résultats de l'annotation manuelle de 104 cellules parmi celles identifiées comme anormales. On y observe à peu près les mêmes résultats que pour l'expérience précedente de détection d'anomalies sur les données de $\mathcal{A}_{1,test}$. Avec 42.3% des anomalies étant des anomalies de segmentation et 43.3% des anomalies cellulaires (contre respectivement 49% et 36.5% avec le modèle α sur $\mathcal{A}_{1,test}$). De plus, le tableau IV.10 montre la répartition des anomalies biologiques, on y constate les mêmes tendances que dans l'expérience section C.2 avec une majorité de cellules ayant une croissance anormale.

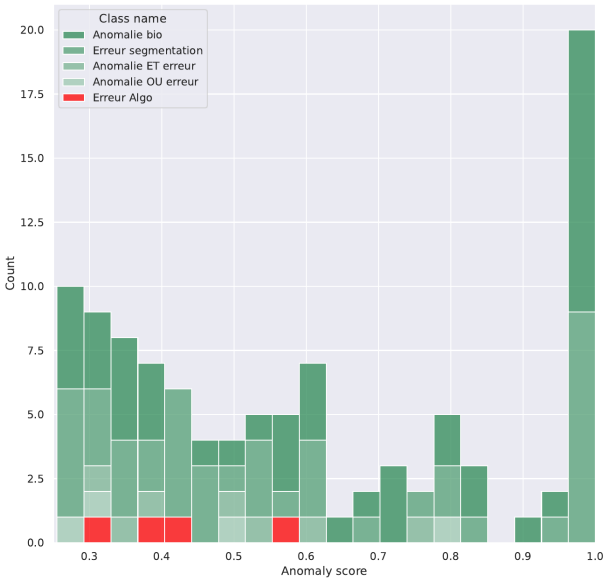


Figure IV.14 : Histogramme des scores d'anomalie en fonction de leur classe

Table IV.9 : Répartition de l'annotation manuelle des anomalies détectées

Anomalies d'acquisition	42.3 % (44)
Anomalies biologiques	43.3 % (45)
Segmentation *et* biologique	6.7 % (7)
Segmentation *ou* biologique	3.8 % (4)
Faux positifs	3.8 % (4)
Précision	**94.2%**

Table IV.10 : Classification manuelle des (45) anomalies biologiques

Division anormale	20 % (9)
Croissance anormale	49% (22)
Cellules fusionnant	9% (4)
Cellules anormalement grosse	7 % (3)
Cellules mortes	11 % (5)
Autre	4 % (2)

C.3.4 Analyses

Le modèle α semble donc plus performant que le modèle β pour détecter les anomalies du jeu de données $\mathcal{B}$ bien qu'il n'ait pas été entrainé sur ce jeu de données ci. On s'attend cependant à des performances moins bonnes du modèle α lors d'une tâche plus complexe comme la généralisation à un autre jeu de données que lors de la tâche plus simple avec une détection d'anomalies sur les mêmes acquisitions que l'entrainement. Ces résultats permettent de mieux comprendre et de mettre en perspective la plus faible précision obtenue dans le cadre de l'expérience précédente sur les données $\mathcal{B}$. Ces moins bonnes performances ne semblent pas provenir de l'entrainement du modèle (car il a été entrainé à plusieurs reprises), ni du modèle en lui-même qui ne serait pas capable d'atteindre de meilleures performances, pour preuve, le modèle $\mathcal{B}$, reposant sur la même architecture neuronale est capable de mieux détecter les anomalies. L'explication la plus plausible est donc celle de la qualité des données d'entrainement (les données $\mathcal{B}$) présentées au modèle β. Ces dernières, probablement à cause de leur diversité, soit trop importante, soit pas assez importante, n'ont pas permis un bon apprentissage de représentation du modèle. Les performances du modèle β pour la détection d'anomalies s'en retrouvent donc dégradées.

La figure IV.14 montre l'histogramme des scores d'anomalies des cellules annotées. La couleur correspond à la famille de l'anomalie. On observe ici que les cellules anormales sont parmi celles ayant un des scores d'anomalie les plus faibles.

Conclusion : L'expérience de cette section permet de démontrer les capacités de généralisation du modèle α. Il obtient en effet les mêmes résultats de détection, avec $94.2\,\%$ de précision, qu'il soit utilisé pour détecter les anomalies de $\mathcal{A}$ ou $\mathcal{B}$.

De plus, on remarque que les moins bons résultats obtenus par le modèle β pour la détection d'anomalies dans l'expérience précédente semblent provenir de la représentation apprise. L'entrainement du modèle β ayant été réalisé grâce au jeu de données $\mathcal{B}$, on peut émettre l'hypothèse que ce dernier est moins propice à l'entrainement des réseaux de neurones pour la prédiction du futur de la masse sèche d'une cellule.

C.4 Validation par un modèle et limites

Objectif

Les expériences précédentes ont nécessité une annotation manuelle de la part d'un expert. Afin d'alléger le travail d'annotation, il a donc été décidé de ne pas annoter les cellules n'étant pas détectées comme anormales par les modèles StArDusTS.

Cependant, cette méthodologie ne permet de pas de mesurer la quantité d'anomalies manquées dans le reste des cellules. C'est la notion de rappel (voir chapitre II section A). Nous proposons dans cette section de construire un jeu de données dans lequel nous

avons connaissance *a priori* des anomalies.

C.4.1 Protocole

Architecture : Identique à l'expérience précédente (CNN 1D).

Tâche prétexte : Identique à l'expérience précédente (Prédiction des 10 prochaines heures du signal).

Détecteur d'anomalie : Identique à l'expérience précédente (Score d'anomalie).

Jeux de données : Les Fibroblastes du jeu de données $\mathcal{C}'$ étant génétiquement modifiées, ces dernières ont un comportement différent des cellules du jeu de données $\mathcal{C}$. Ces cellules modifiées peuvent donc être considérées comme anormales pour les besoins de cette expérience. Ainsi, le modèle γ est entrainé sur le jeu de données $\mathcal{C}_{\text{entrainement}}$ et validé sur $\mathcal{C}_{val}$, mais la détection d'anomalies est réalisée sur la concaténation des jeux $\mathcal{C}_{test}$ et $\mathcal{C}'$. Les cellules modifiées contenues dans $\mathcal{C}'$ devraient être détectées comme anormales par le modèle.

Table IV.11 : Résumé des données et modèles de l'expérience de validation par un modèle d'anomalies

Entrainement	Validation	détection d'anomalies	Modèle	Détecteur
$\mathcal{C}_{1,\text{train}}$	$\mathcal{C}_{1,\text{validation}}$	$\mathcal{C}_{1,\text{test}} + \mathcal{C}'_1$	γ	à score

Résultats

Détection : La figure IV.15 présente les courbes Receiver Operating Characteristic (ROC) (à gauche) et PR (à droite) de détection (chapitre II section A). Chaque point correspond à une valeur de seuil τ pour l'annotation d'une série comme anormale. La croix rouge correspond au seuil τ calculé à partir du jeu d'entrainement $\mathcal{C}$.

Projections : La figure IV.16 montre sur chaque ligne une projection différente de l'espace des features au niveau du goulot d'étranglement du modèle. La première ligne réalisant une projection à l'aide de la méthode TSNE [VH08], la seconde à l'aide d'une ACP (dont les 2 premiers vecteurs représentent 17% et 6% de la variance) et la troisième à l'aide d'une ACI. Sur ces projections, les séries temporelles en bleu sont normales et celles en orange anormales. La colonne de gauche correspond à la vérité terrain, celle de droite à l'annotation grâce à une valeur de seuil $\tau = 0.92$ calculée grâce au jeu d'entrainement $\mathcal{C}_{\text{train}}$.

Analyse

Détection : On observe que les performances du modèle StArDusTS pour détecter les cellules anormales sont pertinentes, car au-dessus de la courbe orange.

De plus, le choix de la valeur de τ par rapport à l'intervalle de confiance à 95% est un choix pertinent puisqu'il fait partie des seuils les plus en haut à gauche sur la courbe ROC et les plus en haut à droite sur la courbe PR.

Nous sommes conscients des biais présents dans cette expérience, c'est pourquoi les valeurs absolues de précision et de rappel ne seront pas discutées. En effet, le jeu de données $\mathcal{C}$ (donc la vérité terrain normale) peut lui aussi contenir des cellules anormales.

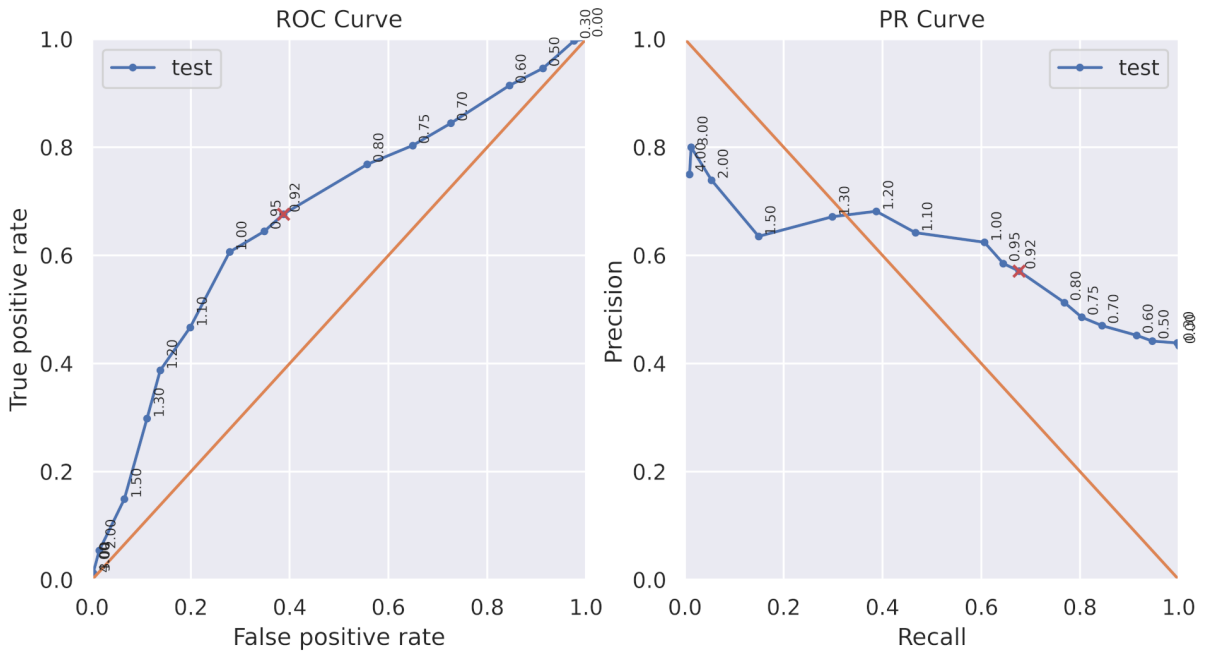


Figure IV.15 : Courbes ROC et PR pour les jeux de données $\mathcal{C}$ et $\mathcal{C}'$. Chaque point correspond à une valeur de seuil τ pour l'annotation d'une série comme anormale. La croix rouge correspond au seuil τ calculé à partir du jeu d'entrainement $\mathcal{C}$

Projections : L'analyse de la colonne de gauche montre que l'ensemble d'apprentissage de représentation composé d'une tâche prétexte de prédiction et d'une architecture de CNN 1D ne permet pas de regrouper les séries temporelles normales et anormales dans des parties distinctes de l'espace latent. En effet, on observe sur la figure que les points correspondant aux séries anormales en orange se superposent aux points des séries normales en bleu.

Bien que les méthodes de projections soient des outils puissants pour visualiser et explorer des données complexes, elles comportent des limitations. Il convient d'exercer une certaine réserve dans l'interprétation de ces résultats. Les conclusions tirées de ces projections sont des pistes de réflexion qui ne peuvent pas être considérées comme des résultats ancrés. En particulier, pourquoi notre modèle est capable de détecter des anomalies alors que les données ne sont pas séparées dans les projections des espaces latents ? Le fait de projeter les données revient en fait à enchainer 2 modèles, il n'est alors pas possible de déterminer si cette non séparabilité provient du premier ou du second projecteur. Enfin, ce n'est pas parce que les données ne sont pas séparables en dimension 2 qu'elles ne le sont pas dans l'espace d'origine.

Conclusion :

La construction d'un jeu de données dont nous disposions artificiellement des annotations a permis de valider le modèle StArDusTS utilisé dans les sections C.2 et C.3. Les deux points clés que valident cette expérience sont :

— Le choix de la valeur du seuil $\tau = \mu + 2 \cdot \sigma$ est pertinent, il correspond à l'un des meilleurs choix de seuil d'après les courbes ROC et PR figure IV.15

— Le modèle et le détecteur à seuil ne permettent pas de séparer *facilement* les séries temporelles normales de celles anormales dans l'espace latent. Il est nécessaire d'utiliser d'autres méthodes d'apprentissage de représentation afin de séparer les différentes classes du jeu de données dans l'espace latent directement lors de la

Figure IV.16 : Projection dans 3 espaces différents des features (extraites au niveau du goulot d'étranglement) des données $\mathcal{C}$ (normales) en bleu et $\mathcal{C}'$ (anormales) en orange. On constate que les données normales ne sont pas séparables des données anormales (elles se superposent). La représentation ne projette donc pas les données de classes différentes dans des endroits différents de l'espace latent

phase d'apprentissage de représentation. En particulier dans le cadre de l'apprentissage auto-supervisé, d'autres tâches prétextes peuvent être explorées.

C.5 Conclusion

Dans cette section C, quatre expériences ont été mises en place afin de confirmer sur cas pratique et applicatif les capacités du modèle StArDusTS à effectivement détecter des séries temporelles anormales.

Dans un premier temps, la version la plus simple de StArDusTS, avec un unique dé-

tecteur à seuil, a permis de vérifier la capacité de ce dernier à détecter des cellules anormales. Même grâce à ce modèle simple, 97% des séries temporelles présentées à l'expert ont été identifiées comme anormales, validant les capacités de détection d'anomalies de StArDusTS. C'est par ailleurs cette étude qui a permis de comprendre les différentes classes d'anomalies contenues dans le jeu de données.

Ensuite l'expérience section C.2 démontre les capacités de StArDusTS avec un détecteur à seuil couplé et un score d'anomalie pour la détection automatique de cellules anormales et non plus uniquement de fenêtres indépendantes. Cette section a permis à l'expert de différencier 6 sous-classes d'anomalies cellulaires.

L'expérience section C.3 montre les capacités de généralisation du modèle. Ce dernier étant entrainé sur un jeu de données, mais devant détecter les anomalies d'un autre. Ainsi, la représentation des séries temporelles apprise par le réseau de neurones est assez générale pour pouvoir être utilisée sur d'autres jeux de données de séries temporelles de cellules, même si les cellules diffèrent quelque peu du jeu d'entrainement.

Enfin la dernière expérience (section C.4) confirme les résultats applicatifs obtenus dans les deux premières sur un jeu de données artificiellement créé. L'utilisation d'acquisition de cellules génétiquement modifiées a permis de créer artificiellement un jeu de données dont nous avions la connaissance de la vérité terrain afin de pouvoir analyser les performances de détection surtout de rappel, ce qui n'était pas possible dans les expériences précédentes.

Cependant, cette expérience montre aussi que l'espace de représentation généré par la combinaison d'un réseau de neurones avec une architecture de CNN 1D et la tâche prétexte de prédiction du futur de la série temporelle ne permettent pas de séparer *facilement* les cellules normales de celles anormales. C'est pourquoi, dans l'étude de cas suivante sur les données de Mars chapitre V, une seconde tâche prétexte, l'apprentissage contrastif, est étudiée.

D Comparaison de différentes architectures

Le contenu de cette section a été accepté pour une présentation à la conférence Gretsi 2023 sous le nom *"Comparaison des capacités prédictives de réseaux de neurones, application à la masse sèche de cellules"* [BMA+23a]

Objectif

La section C a mis en avant les capacités de détection d'anomalies de modèles à base de réseaux CNN 1D. Cependant, une autre architecture de réseaux de neurones profonds aurait pu être utilisée pour l'apprentissage de la représentation utilisée pour la détection d'anomalies. Au travers de cette section, nous cherchons à déterminer l'architecture de réseau de neurones profond la plus propice pour réaliser la tâche prétexte de prédiction de séries temporelles de masse sèche de cellules. En effet, les jeux de données étant non annotés, il n'est pas possible d'évaluer automatiquement les architectures neuronales sur la tâche finale de détection d'anomalies.

Protocole

Dans cette optique, l'outil de comparaison des modèles développé chapitre III section D qui implémente la méthode d'optimisation successive des hyperparamètres présenté chapitre III section C est utilisé. Les architectures de l'état de l'art chapitre II section B sont entrainées avec les valeurs d'hyperparamètres présentés tableau IV.12. Cette expérience correspond d'ailleurs au fichier de configuration liste A.1 présenté en annexe A. Les modèles prédictifs sont entrainés sur le jeu de données $\mathcal{A}_1$.

Table IV.12 : Tableau récapitulatif de l'espace de recherche des hyperparamètres. Un hyperparamètre est optimisé avec une valeur fixée des autres hyperparamètres

Paramètres	Valeurs possibles
	Réseaux denses
$\mathcal{D}$	$32/32/32-32/64/128-64/64/64/64/64$ $64/64/64-256/128/64-512/256/128/64/32$ $128/128/128-32/32/32/32/32$
	Réseaux convolutifs à maxpooling
$\mathcal{C}$	$3-5-6-9-12-16$
$\mathcal{K}$	$3-5-8-12-16$
$\mathcal{F}$	$64-32-128-256$
$\mathcal{MP}$	$3-2-1$
$\mathcal{D}$	$64-16-32-128-256-16/16-32/32-64/64-$ $128/128-64/32-128/64-256/128-64/32/16-$ $128/64/32$
	Réseaux convolutifs à dilatation
$\mathcal{C}$	$3-5-9-12$
$\mathcal{K}$	$3-5-8-12-16$
$\mathcal{F}$	$64-128-256$
$\mathcal{DR}$	$2^i-1-4-8-16$
	LSTM
$\mathcal{L}$	$1-2-3$
$\mathcal{N}$	$64-32-128-256-512$
	Réseaux à connexions résiduelles
$\mathcal{B}$	$1-2-3-5$
$\mathcal{DR}$	$2i-2-4-8-16$

Afin de comparer les résultats des différentes architectures à une référence, nous proposons 3 bases de référence *. Ces valeurs de référence sont présentées tableau IV.13. On calcule l'EQM de ces bases de référence sur le jeu de données $\mathcal{A}_{\text{validation}}$. C'est la prédiction de la moyenne des entrées qui obtient les meilleures performances avec une EQM de $\mathbf{0.70}$.

Pour chacune des architectures présentées, les réseaux de neurones sont entrainés 10 fois afin de minimiser l'impact de l'initialisation des poids des réseaux de neurones profonds sur l'entrainement.

Les entrainements des réseaux sont réalisés sur une carte graphique `nvidia A100` grâce à la librairie `tensorflow` et l'outil de benchmark présenté chapitre III section D.

*. baselines

Cette architecture matérielle est à prendre en compte dans l'analyse absolue des temps d'entrainement et de validation.

Les architectures sont comparées selon 4 métriques :

— L'EQM minimale et moyenne sur 10 entrainements
— Le nombre de paramètres
— Le temps d'entrainement moyen
— Le temps de validation moyen

Résultats

Le tableau IV.14 présente uniquement les architectures les meilleures dans chacune des catégories mentionnées ci-dessus, mais le récapitulatif complet des expériences menées est détaillé en annexe tableau B.1.

Tout d'abord, nous constatons que, bien qu'il soit possible de différencier les familles de réseaux sur leur valeur d'EQM, ces dernières sont comprises entre 0.454 pour le meilleur de tous les réseaux (Denses) contre 0.471 pour la moins bonne famille (Avec skip connections). Cet écart de seulement 3.7% souligne des performances similaires pour toutes les architectures.

Le nombre de paramètres des réseaux est de son côté bien plus clivé. Les CNN à dilatation ne diminuant jamais la taille de l'entrée se retrouvent à avoir un nombre de paramètres dépassant le million pour un gain de performances prédictives peu significatif. En comparaison, les réseaux denses, peuvent obtenir des performances seulement 2% plus faibles pour un nombre de paramètres 127 fois plus faible. Les réseaux permettant d'obtenir le meilleur compromis nombre de paramètres/EQM sont les réseaux denses pouvant atteindre des EQMs de l'ordre de 0.454 avec seulement $56\,000$ à $76\,000$ paramètres alors qu'aucune autre famille de réseaux n'est capable de dépasser la barrière de 0.46 d'EQM.

Les LSTM et les réseaux Denses peuvent atteindre des temps d'entrainement similaires d'à peine plus de 2 minutes. Par opposition, les CNN avec ou sans skip connection peuvent atteindre des temps d'entrainement 2 à 3 fois plus longs pour des performances moindres. Il faut cependant remettre en perspective les temps d'entrainement observés dans cette étude, aucun n'ayant duré plus de dix minutes, face à des temps d'entrainement classiques dans le domaine de l'intelligence artificielle compris entre plusieurs heures et plusieurs semaines.

Concernant les temps d'exécution, les réseaux Denses sont largement plus performants que les autres architectures avec 0.83 s de moins que les LSTM en seconde position. Cet écart représente un temps d'inférence par échantillon de 88 µs et 103 µs.

Discussion et conclusion

La tâche d'intérêt pour le jeu de données considéré dans ce chapitre est la détection de cellules anormales à partir de la masse sèche des cellules. Cependant, le jeu de don-

Table IV.13 : Trois bases de références et leur valeur sur le jeu de données $\mathcal{A}_{\text{validation}}$

Référence et description	EQM
La sortie prédite est toujours 0, peu importe les données d'entrée	0.82
La dernière valeur de la fenêtre d'entrée est recopiée en sortie	1.15
La moyenne de la fenêtre d'entrée est recopiée en sortie	$\underline{\mathbf{0.70}}$

Table IV.14 : Tableau récapitulatif des métriques obtenues pour les meilleurs réseaux de neurones dans chaque famille et pour chaque métrique. Les valeurs soulignées sont les meilleures intra famille tandis que celles en **gras** sont les meilleures inter familles

Paramètres					EQM min	EQM moyenne	Nb param	Temps d'entrainement	Temps validation
$\mathcal{D}$					Perceptrons Multicouches				
256/128/64					**0.454**	**0.458 ± 0.001**	76 028	182 ± 13 s	5.94 ± 0.25 s
32/32/32					0.474	0.478 ± 0.002	**7 964**	195 ± 33 s	4.70 ± 0.04s
32/64/128					0.477	0.482 ± 0.004	22 044	**131 ± 26** s	6.70 ± 0.25 s
128/128/128					**0.454**	0.460 ± 0.002	56 252	161 ± 23 s	**4.67 ± 0.07** s
$\mathcal{C}$	$\mathcal{K}$	$\mathcal{F}$	$\mathcal{MP}$	$\mathcal{D}$	CNN-1D à Maxpooling				
9	3	64	3	128	0.460	0.470 ± 0.002	229 820	327 ± 26 s	8.46 ± 0.17 s
9	3	32	3	64	0.471	0.478 ± 0.004	59 644	291 ± 56 s	6.75 ± 0.11 s
5	3	64	3	64	0.468	0.479 ± 0.002	299 388	172 ± 7 s	5.91 ± 0.05 s
3	3	64	3	64	0.467	0.482 ± 0.004	274 684	180 ± 26 s	5.58 ± 0.04 s
$\mathcal{C}$	$\mathcal{K}$	$\mathcal{F}$	$\mathcal{DR}$		CNN-1D à taux de dilatation				
5	3	64	$2i$		0.461	0.471 ± 0.003	1 040 572	239 ± 13 s	6.50 ± 0.12 s
3	3	64	$2i$		0.466	0.476 ± 0.003	1 015 868	192 ± 36 s	5.85 ± 0.15 s
$\mathcal{L}$		$\mathcal{N}$			LSTM				
1		32			0.465	0.468 ± 0.001	16 316	191 ± 14 s	5.61 ± 0.12 s
1		256			0.478	0.483 ± 0.002	304 828	132 ± 6 s	5.50 ± 0.09 s
$\mathcal{B}$		$\mathcal{DR}$			CNN résiduels				
5		8			0.471	0.481 ± 0.003	130 044	510 ± 28	9.40 ± 0.12
1		$2i$			0.543	0.561 ± 0.005	29 180	238 ± 25	6.01 ± 0.27

nées n'étant pas annoté, on ne peut pas accéder directement aux métriques classiques d'évaluation de méthode de détection d'anomalies comme la précision ou le rappel.

La performance du modèle sur la tâche finale n'étant pas quantifiable, on cherche donc à évaluer le modèle au moins sur la tâche prétexte. L'idée principale étant que la performance du modèle sur la tâche prétexte est une image de la performance de la représentation apprise par ce dernier. Autrement dit, plus le modèle est capable de prédire le futur des cellules, plus la représentation apprise est susceptible d'être efficace pour la tâche d'intérêt.

Au travers des expériences menées dans cette section, il semblerait qu'aucune des architectures expérimentées ne soit nettement meilleure que les autres à prédire les 10 prochaines heures du signal connaissant les 20 premières, les performances tenant dans intervalle de $3.7\,\%$.

Factuellement ce sont les réseaux denses qui obtiennent les meilleurs résultats d'EQM pour des performances temps d'entrainement et d'inférences plus courts. Cependant, ces résultats restent à nuancer. En effet, la méthode d'optimisation successive a poussé l'exploration d'architectures ayant des nombres de paramètres différents. Afin d'avoir un meilleur recul pour la comparaison des résultats des différentes architectures entre elles, une étude menée à nombre de paramètres fixés serait nécessaire.

E Conclusion

E.1 Conclusion

Pour rappel ce chapitre avait pour but de répondre à la problématique applicative n°1 de la détection automatique de cellules anormales pour pouvoir les présenter ensuite à des experts pour qu'ils puissent les analyser. Nous montrons par la même occasion les capacités du modèle StArDusTS à détecter des anomalies dans des séries temporelles.

L'étude de StArDusTS a été entreprise en plusieurs étapes, visant à évaluer ses capacités de détection d'anomalies cellulaires. Dans un premier temps, une version simple de l'approche a été mise en œuvre, utilisant un unique détecteur à seuil. Les résultats de cette phase initiale ont été encourageants, avec une capacité de détection des cellules anormales atteignant un taux de $96.2\,\%$. Cette performance a permis de valider de manière convaincante les fonctionnalités d'apprentissage de représentation de StArDusTS, en particulier en ce qui concerne la détection d'anomalies dans les séries temporelles cellulaires.

Ensuite, l'expérience section C.2 a été menée pour explorer davantage les capacités de StArDusTS. Cette phase a introduit un détecteur à seuil couplé avec un mécanisme de score d'anomalie. L'objectif était de passer de la simple détection de fenêtres indépendantes à la détection automatique de cellules anormales dans leur contexte global. Les résultats de cette expérience ont été particulièrement informatifs, car ils ont permis d'identifier et de différencier six sous-classes distinctes d'anomalies cellulaires. Cela témoigne de la robustesse et de la capacité de StArDusTS à traiter des situations plus complexes et variées. Par ailleurs, nous proposons dans cette section une méthode pour visualiser et localiser les anomalies dans la série temporelle. En effet, la méthode mise en oeuvre ne permet pas de localiser l'anomalie détectée au sein de la fenêtre de 30 heures analysée (l'anomalie peut aussi bien se trouver en entrée ce qui mène à une mauvaise prédiction et donc une EQM élevée, qu'en sortie ce qui mène aussi à une EQM élevée). La visualisation proposée permet cependant aux experts d'identifier visuellement les instants, les points individuels, à partir desquels la série temporelle devient la plus anormale. Elle permet une analyse plus fine qu'une unique valeur de score d'anomalie.

L'expérience suivante, section C.3, a été conçue pour évaluer la capacité de généralisation de StArDusTS. Le modèle, entrainé sur un premier jeu de données, a ensuite été testé sur un jeu de données différent, mettant ainsi à l'épreuve sa capacité à détecter des anomalies dans un contexte différent de celui de l'entrainement. Les résultats ont montré que la représentation des séries temporelles apprise par le réseau de neurones était suffisamment générale pour être transférée avec succès vers d'autres jeux de données de cellules, même si ces dernières présentaient des variations par rapport au jeu d'entrainement initial.

L'expérience section C.4 a été menée pour consolider les découvertes et les réalisations des étapes précédentes. Pour ce faire, un jeu de données artificiel a été créé en utilisant des cellules génétiquement modifiées. Cette approche a permis de disposer d'une vérité terrain connue, permettant ainsi une évaluation plus approfondie des performances de détection, en particulier en ce qui concerne le rappel. Cette phase de validation supplémentaire a renforcé la confiance dans les capacités de StArDusTS à détecter efficacement et de manière fiable les anomalies cellulaires, tout en offrant des perspectives prometteuses pour son application dans des scénarios réels de détection d'anomalies.

Pour aller plus loin, nous nous sommes concentrés sur l'optimisation d'une architecture de réseaux de neurones capable de capturer l'information contenue dans les séries temporelles de masse sèche au travers d'une tâche prétexte de prédiction des 10 prochaines heures du signal connaissant les 20 premières. Les expériences menées section D

montrent que les architectures proposées obtiennent des résultats du même ordre de grandeur sur la tâche prétexte de prédiction. L'utilisation des réseaux CNN 1D dans la partie précédente était donc pertinente. Cependant, de simples réseaux denses auraient pu être utilisés de manière identique pour la tâche prétexte pour des durées d'entrainement et des nombres de paramètres inférieurs.

E.2 Discussion et perspectives

Les données cellulaires sont particulièrement contraignantes, car elles ne disposent pas d'annotation permettant de vérifier automatiquement l'efficacité de la totalité du modèle StArDusTS. En effet, une annotation manuelle *a posteriori* des cellules étant détectées comme anormales est nécessaire pour vérifier si celles-ci sont effectivement anormales ou non.

Remarque 10 : Concours

Un concours organisé sur le site kaggle.com intitulé *Open Problems - Multimodal Single-Cell Integration* ayant pour tâche la quantification de l'expression de gènes à partir de la séquence génomique est paru en Août 2022.

Ce genre de concours montre que l'intérêt que prêtent les différentes communautés scientifiques à l'apprentissage profond pour résoudre des problèmes complexes, en particulier dans le cadre de séries (temporelles), est en constante augmentation.

`https://www.kaggle.com/competitions/open-problems-multimodal`

Les méthodes de détection d'anomalies n'étant pas parfaitement comparables, car auraient nécessité une annotation complète du jeu de données, nous nous sommes restreints à l'utilisation d'un détecteur d'anomalie composé d'un seuil sur l'EQM de prédiction, couplé à un score d'anomalie représentant le ratio de fenêtres détectées anormales par le nombre total de fenêtres étudiées. Néanmoins, ce détecteur, associé aux capacités prédictives du réseau de neurones CNN 1D, permet de détecter ses cellules anormales avec une précision allant jusqu'à $96.2\,\%$.

L'hypothèse principale utilisée dans cette section est que la quantité de cellules normales contenue dans les jeux d'entrainement est toujours nettement supérieure à celle des cellules anormales. Grâce à cela, la représentation apprise par le réseau de neurones est une représentation adaptée pour les cellules normales et pas celles anormales ce qui permet de les détecter.

Certains points n'ont pas pu être explorés dans cette étude du fait du manque d'annotations disponibles pour évaluer la tâche dans son ensemble. C'est pourquoi, une seule tâche prétexte et un seul détecteur d'anomalies ont été utilisés. La tâche prétexte de prédiction a par ailleurs montré ses limites quant à la séparation des séries temporelles normales de celles anormales dans son espace latent.

Une comparaison relative des détecteurs d'anomalie aurait pu être effectuée. Pour cela, nous aurions pu comparer les cellules identifiées comme anormales par différents détecteurs et vérifier une cohérence au sein des multiples détections. Bien que nous ne disposions pas de la vérité terrain permettant d'affirmer si une cellule est anormale ou non, si une cellule est détectée par plusieurs détecteurs, il est plus probable qu'elle soit vraiment anormale.

CHAPITRE

V

Application : données sismiques de Mars

Table des matières

A Introduction

A.1 Contexte

Depuis l'envoi de la sonde Viking en 1975, un éventail conséquent de missions ont été entreprises en direction de la planète rouge, visant à approfondir notre compréhension de son environnement, de son passé. Diverses agences spatiales telles que la NASA, l'ESA et d'autres acteurs ont déployé rovers, orbiteurs et atterrisseurs pour sonder les mystères de Mars.

Comprendre la planète Mars est d'une importance cruciale. En particulier, Mars est la planète la plus similaire à la Terre dans notre système solaire, ce qui en fait une candidate prometteuse pour des futures colonies humaines. L'étude approfondie de Mars nous permettrait de mieux comprendre les processus géologiques et atmosphériques de cette dernière.

InSight, qui signifie *Interior Exploration using Seismic Investigations, Geodesy and Heat Transport*, est une mission conçue pour étudier la structure interne de Mars, notamment son noyau, son manteau et sa croûte [GWG+20]. La mission SEIS [LBG+19, BSB+20], abréviation de *Seismic Experiment for Interior Structure*, est une composante essentielle de la mission InSight de la NASA sur Mars. Le sismomètre SEIS est composé de trois pendules sophistiqués extrêmement sensibles qui a été déployé sur la surface de Mars par InSight en décembre 2018. Le modèle 3D de l'un de ces pendules est montré figure V.1. Son objectif principal est de détecter et d'analyser les tremblements de terre, ou plutôt les "marsquakes", sur la planète rouge. Ces tremblements de terre fournissent des informations précieuses sur la composition, la structure et l'activité tectonique de Mars. L'instrument SEIS se compose d'un sismomètre ultrasensible et d'un bouclier thermique pour le protéger des variations de température extrêmes et des vents martiens. Il a été développé par une équipe internationale de scientifiques et d'ingénieurs, avec une contribution majeure de l'agence spatiale française, le Centre National d'Études Spatiales (CNES) [LBG+19].

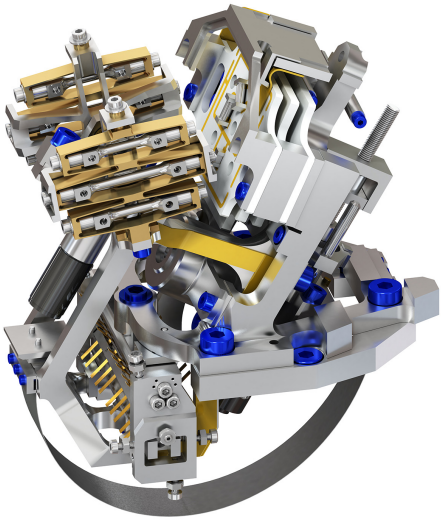


Figure V.1 : Modèle 3D du pendule VBB de SEIS (©IPGP/David Ducros [Duc18])

Les données recueillies par SEIS permettent aux scientifiques d'étudier la structure interne de Mars, y compris la taille et la composition de son noyau, ainsi que les caractéristiques de sa croûte et de son manteau [GWG+20, GLB+20, LBP+20], mais aussi son atmo-

sphère [BSN+20] et son champ magnétique [JML+20]. Ces informations sont essentielles pour comprendre la formation et l'évolution de la planète rouge, ainsi que pour comparer Mars à la Terre et aux autres corps célestes.

La mission SEIS a déjà fourni des résultats significatifs. En février 2019, le premier marsquake a été détecté, marquant la première fois qu'un tremblement de terre a été enregistré sur une autre planète que la Terre ou la Lune. Depuis lors, de nombreux autres marsquakes ont été enregistrés et analysés, contribuant à notre compréhension de Mars.

Des travaux connexes à ceux présentés dans ce manuscrit ont notamment été réalisés par [BLK+21]. Ce papier traite notamment de la reconnaissance de signaux intrinsèques aux données de Mars grâce à des méthodes d'apprentissage non supervisées avec des réseaux profonds à diffusion * [KH21, SZX+21]. Ces méthodes reposent sur l'apprentissage d'ondelettes permettant de décomposer le signal [SBP+20] et sont des versions régularisées des CNN utilisés classiquement en apprentissage profond.

A.2 Problématiques

Plusieurs problématiques ont été identifiées en accord avec les experts du domaine, ces dernières sont détaillées ci-dessous.

Problématique 1 : Détection d'évènements sismiques

La détection des évènements sismiques est aujourd'hui réalisée manuellement par des experts, sur terre. Lorsque des évènements sismiques sont identifiés au sein des données, les sismologues peuvent alors émettre une requête vers le lander afin de demander les données avec un pas d'échantillonage plus petit. Le sismomètre SEIS n'étant pas la seule mission remplie par InSight, ces données peuvent prendre jusqu'à 3 mois pour être renvoyées.
La détection automatique et sur place d'évènements sismiques permettrait un envoi direct des données sismiques en haute résolution des portions de signal contenant des événements intéressants pour l'étude de la structure interne de la planète.

Problématique 2 : Détection des glitchs et des dust devils

Hormis les événements sismiques, les données contiennent (au moins) deux autres types d'événements identifiés :

Les glitchs sont des altérations temporaires des données, notamment à cause des températures extrêmes et du rayonnement solaire subis par le lander,

Les dust devils sur Mars sont des phénomènes météorologiques similaires aux tourbillons de poussière ou aux minitornades que l'on peut observer sur Terre. Ces dust devils se traduisent dans les données par une **chute de pression**. Dans la suite du manuscrit, "dust devil" et "chute de pression" feront donc référence au même type d'événement.

Ces deux types d'événements perturbent fortement les données sismiques rendant la détection d'événements sismiques encore plus ardue. La détection automatique des glitchs et des dust devils afin de pouvoir les retirer est un axe de recherche important aujourd'hui.

*. Deep scatering networks

Problématique 3 : Débruitage du signal le jour

Les signaux produits par les capteurs de SEIS sont beaucoup plus bruités en journée que la nuit, en grande partie à cause des dust devils qui ont lieu majoritairement en journée et perturbent les acquisitions sismiques. Néanmoins, une fois le soleil couché, l'activité environnante se calme et les données sismiques sont plus faciles à analyser. Il serait donc intéressant de pouvoir débruiter les données, en particulier les données de journée qui sont très peu exploitables aujourd'hui.

Problématique 4 : reconstruction d'une modalité perdue

Le lander InSight fonctionne en totale autonomie depuis son atterrissage en 2018. Cependant, depuis avril 2021, en raison d'un problème d'empoussièrement, les capteurs de pression APSS ont été arrêtés. Il était néanmoins beaucoup plus simple de détecter les dust devils grâce au signal de pression.
Une problématique ouverte est alors la reconstruction automatique des données de pression à partir des autres capteurs disponibles sur la station.

Nous répondons dans ce chapitre aux deux premières problématiques. Nous cherchons en effet à réaliser une détection automatique des événements au sens large, que ce soient des événements sismiques, des glitchs ou des dust devils. En ce sens, ces événements sont des déviations par rapport au comportement *normal* observé, le modèle StArDusTS peut donc être utilisé pour détecter ces *anomalies*, ces événements.

A.3 Objectifs et plan

Ces problématiques applicatives vont servir de support pour la réponse à des questions méthodologiques concernant la détection d'anomalies sans annotations et en particulier les capacités du modèle StArDusTS. En effet, après le cas d'étude présenté chapitre IV, de nombreuses questions méthodologiques restent en suspens.

Dans quelle mesure le modèle StArDusTS, composé d'un CNN 1D, d'une représentation apprise grâce à une tâche prétexte de prédiction et d'un détecteur à seuil, proposé chapitre IV est un modèle capable de détecter des anomalies sur un autre jeu de données ?

Nous nous penchons dans un premier temps sur le bloc d'apprentissage de représentation de StArDusTS section C.1. Tout d'abord, la même tâche prétexte de prédiction du futur de la série temporelle est étudiée section C.1.1. Est-ce que cette tâche prétexte se transfère à tout type de jeu de données ? Ces expériences nous permettent à la fois de définir une architecture et les données à utiliser pour l'étude des données sismiques de Mars. Ensuite, une seconde tâche prétexte, d'apprentissage contrastif est étudiée section C.1.2.

Dans un second temps (section C.2), les features des deux tâches prétextes sont utilisées couplées à plusieurs détecteurs d'anomalies afin de comparer les performances couplées des représentations apprises et des détecteurs.

Contrairement aux données cellulaires du chapitre précédent, nous disposons des annotations pour pouvoir mesurer les performances des modèles du point de vue de la détection d'anomalies. Grâce aux annotations il est donc possible de mesurer la précision et le rappel des modèles à détecter des anomalies automatiquement (sans avoir à faire appel à un expert pour annoter les anomalies détectées) au moment de la détection. Par ailleurs, la disponibilité des annotations remet en cause l'utilisation de méthodes d'apprentissage auto-supervisé, nous discutons ce point chapitre VI section B.4.

B Génération des jeux de données

Les acquisitions (*i.e.* les données brutes), fournies par l'expédition SEIS ne sont pas forcément les données qui vont être utilisées en entrée d'un réseau de neurones ou d'un modèle StArDusTS. Nous présentons dans cette section dans un premier temps les acquisitions puis les prétraitements associés et les données qui vont être utilisées pour générer les jeux de données utilisés pendant l'ensemble des expériences de ce chapitre.

Données brutes Les données utilisées dans cette étude sont les données acquises du 19 juin 2019 au 19 juillet 2019. Commençons par présenter l'ensemble des modalités disponibles dans les données brutes :

Sol : La planète Mars n'a pas la même vitesse de rotation que la terre, ses journées n'ont donc pas la même durée. On nomme sol un jour solaire sur Mars. Un jour solaire est défini par le temps qui s'écoule entre deux passages de l'étoile au même point dans le ciel. Par ailleurs, le nombre de jours écoulés depuis le début de la mission est aussi exprimé en sol.

LMST : l'heure locale sur Mars, ou Local Mean Solar Time (LMST) est la manière de mesurer le temps sur Mars. Un sol dure en moyenne 24 heures, 39 minutes et 35 secondes. Une heure LMST correspond en fait à $1/24^{eme}$ de sol.

Données sismiques : Les données sismiques utilisées dans cette étude sont des données en vitesse sur 3 axes U, V, W. Ces 3 axes ne correspondent pas aux axes Z - Nord - Est que l'on peut parfois rencontrer lors du traitement de données sismiques, mais à trois axes orthogonaux positionnés selon une orientation non standard. Une simple rotation matricielle permet de réorienter ces données en ZNE, il a cependant été décidé de conserver l'orientation originale des pendules afin de traiter les données le plus brut possible. En effet, les prétraitements utilisés sur les données impactent leur contenu et sous-entendent une compréhension profonde des données en train d'être prétraitées. Dans un contexte où le contenu des données est encore peu maitrisé, en particulier lorsque l'on cherche à détecter des nouveautés, ne pas effectuer de prétraitement, c'est aussi s'assurer que l'on ne filtre pas de partie intéressante du signal. Par ailleurs, une rotation matricielle peut être effectuée par un réseau de neurones profond, dans le cas où cette rotation permettrait de faciliter l'analyse, elle pourrait être effectuée dès les premières couches d'un réseau de neurones. Elles sont échantillonnées à 20 Hz. Un premier prétraitement est effectué afin de retirer les valeurs aberrantes (supérieures à 40000 en valeur absolue).

Pression : Un capteur mesure la pression atmosphérique à une fréquence de 5 Hz. Cette modalité est particulièrement utile pour détecter les dust devils.

Élévation du soleil : C'est la mesure de l'angle entre l'horizon et le soleil. Cette modalité permet de définir si les données en train d'être observées ont lieu en journée ou de nuit, en complément du LMST.

Catalogues : C'est l'ensemble des listes répertoriant les événements ayant lieu pendant un certain laps de temps. Ces derniers ont été générés grâce à des annotations proposées par des experts du domaine. Trois catalogues sont utilisés lors de cette étude. Ils correspondent aux anomalies que l'on cherche à détecter, ils contiennent respectivement les événements sismiques, les chutes de pression et les glitchs.

Prétraitement Afin de pouvoir être utilisées, certaines données subissent des prétraitements :

— Les données sismiques ne sont que très peu prétraitées. Seules les données aberrantes, supérieures à un seuil (40000) sont mises à 0. Ces données sont des anomalies

de mesure qui ne présentent aucune difficulté à détecter ni aucun intérêt scientifique. Comme elles risquent de perturber l'entrainement des modèles en particulier à cause d'une mauvaise normalisation si ces données sont conservées, elles sont retirées.

— Les données de pression sont échantillonnées à 5Hz alors que les données sismiques sont elles acquises à une fréquence de 20 Hz. Afin d'aligner ces modalités, les données de pression sont sur-échantillonnées à une fréquence de 20Hz.

— Nous calculons à partir des données d'élévation solaire un vecteur définissant la présence ou non du soleil dans le ciel. Si l'élévation est supérieure à 0, c'est que le soleil est dans le ciel et que ces acquisitions sont réalisées de jour. La modalité Jour/Nuit contenant cette information sera notée DN*. Ce vecteur vaut `0` la journée et `1` la nuit. L'idée derrière ce vecteur est simple, le comportement des données est totalement différent entre le jour, où les données sont très bruitées et les événements difficiles à identifier et la nuit où les événements sont plus faciles à distinguer. Ce vecteur peut donc être utilisé afin de signifier la différence de ces deux régimes. C'est une manière pour nous d'encoder la temporalité des données.

Mise en forme Sur les huit modalités brutes disponibles [U, V, W, Pression, Sol, Heure, Minute, Seconde], présentées ci-dessus, nous avons choisi de nous restreindre à l'utilisation de seulement quatre d'entre elles, les données sismiques [U, V, W] et de pression [P]. Le LMST est quant à lui compressé en une seule valeur qu'est la modalité Jour/Nuit [DN]. Les cinq modalités [U, V, W, P, DN] seront donc utilisées dans les sections suivantes. Ces dernières sont illustrées figure V.2. Sur cette figure, le fond a été assombri pendant la nuit. On y constate les deux régimes en fonction du jour et de la nuit. La décroissance observée quotidiennement à la fois sur les signaux de pression et les traces sismiques coïncident avec le coucher du soleil et l'augmentation progressive de la valeur moyenne du signal correspond, elle, au lever du soleil.

L'apprentissage automatique au sens large repose sur l'acquisition de données et leur utilisation lors d'un apprentissage pour effectuer une tâche d'intérêt. On qualifie parfois de scenario la manière dont les données sont acquises et la manière dont elles vont être utilisées par le modèle. Ici, nous avons choisi de considérer un scenario plausible utilisé lors du déploiement d'un capteur intelligent : Le capteur est déployé sur son lieu de mesure, sur lequel il acquiert des données pendant une certaine durée. Les premières données acquises sont les données d'entrainement. Les données suivantes correspondent aux données de validation du modèle, puis le modèle est utilisé en inférence pour résoudre le problème qui lui a été attribué, dans un cas de laboratoire, ces données sont les données de test. Nous avons donc choisi de découper les jeux de données en jeu d'entrainement, validation et test en suivant cette logique : les 576 premières heures des acquisitions sont associées au jeu d'entrainement, les 72 heures suivantes au jeu de validation puis les 72 dernières heures au jeu de test. Ces données représentent respectivement les 80% 10% et 10% des données. [MMGR22] propose une catégorisation des différents scenarii qui peuvent être rencontrés lors du cas particulier qu'est l'apprentissage incrémental auquel les capteurs intelligents peuvent être confrontés.

Pour l'entrainement des modèles StArDusTS, des fenêtres de 30 secondes (soit 600 points) sont découpées, avec un pas du fenêtrage glissant de 5 secondes. En complément, 3 catalogues sont utilisés pour pouvoir associer un annotation à chaque fenêtre. Ces 3 catalogues correspondent aux 3 différents types d'anomalies que nous cherchons à détecter : les glitchs, les dust devils et les événements sismiques. Ainsi, chacune des fenêtres est associée à une annotation :

— `0` : Normale

*. Day/Night

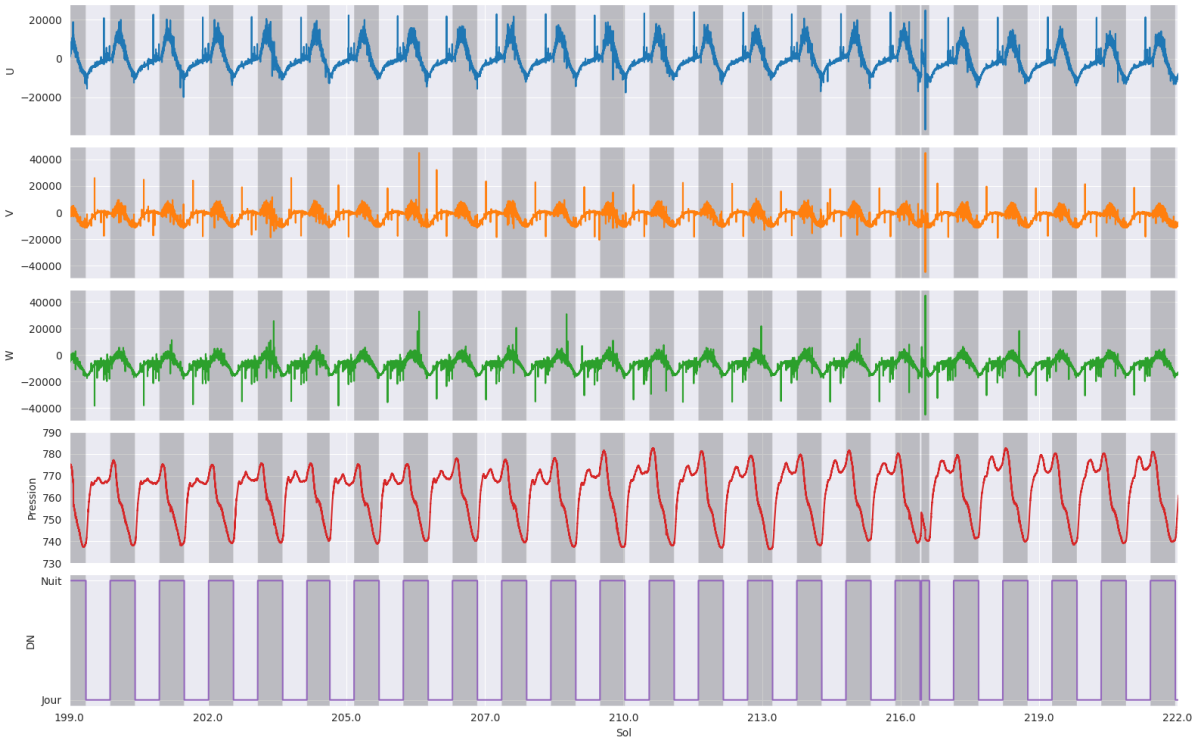


Figure V.2 : Séries temporelles complètes du jeu d'entrainement, incluant les 5 modalités U, V, W, P, DN, sur 26 sol. Le fond des courbes est rempli en gris pendant la nuit. L'artefact présent au niveau du sol 216 correspond à un trou dans les données

— 1 : Glitch
— 2 : Chute de pression
— 3 : Événement sismique

Lorsque plusieurs anomalies se superposent dans la même fenêtre, le choix a été fait, pour simplifier la classification, d'associer l'annotation d'une seule anomalie à la fenêtre. Dans ce cas, l'événement sismique est prioritaire sur les chutes de pression, eux-mêmes prioritaires sur les glitchs. Une classification avec plusieurs labels aurait pu être envisagée, néanmoins, cette dernière aurait rajouté une surcouche de complexité dans les architectures proposées. Pour une première étude, nous nous restreignons donc à un cas simple avec une seule classe associée à chaque fenêtre. Le tableau V.1 résume la répartition des annotations sur les données sismiques de Mars.

Table V.1 : Tableau récapitulatif des fenêtres dans les jeux de données sismiques de Mars

	Entrainement	Validation	Test
Normale	356 168 (90%)	44 907 (91.1%)	43 566 (88.4%)
Glitch	29 353 (7.4%)	3 587 (7.3%)	4 003 (8.1%)
Chute de pression	6 028 (1.5%)	773 (1.6%)	528 (1.1%)
Événement sismique	4 161 (1.1%)	0 (0%)	1 170 (2.4%)
# total de fenêtres de 30s	395 710 (100%)	49 267 (100%)	49 267(100%)

On remarque dans le tableau V.1 que le jeu de validation ne contient pas d'événements sismiques. Il est probable qu'en pratique, lors d'une durée d'environ 3 jours, qu'aucun événement sismique ne soit observé. Ce scénario représente donc un scenario réel qui pour-

rait arriver physiquement lorsqu'un capteur est déployé sur le terrain. Cette configuration ne permettra probablement pas d'atteindre les performances maximales du modèle pour la détection d'événements sismiques, mais elle reflète le type de scenario auquel un capteur intelligent pourrait être confronté. C'est pourquoi il a été choisi de conserver cette distribution des annotations et non pas de réordonner artificiellement les données afin d'introduire des fenêtres contenant des événements sismiques dans le jeu de validation.

De plus on constate que 5331 fenêtres contiennent un événement sismique. Dans le catalogue utilisé, sur la période d'étude, seuls 41 événements sont répertoriés, ces derniers peuvent cependant durer plusieurs dizaines de minutes. Hors, notre fenêtrage glissant a un pas de 5s, soit 25s de superposition entre deux fenêtres consécutives, ce qui explique ce grand nombre de fenêtres contenant des événements sismiques.

C Application de StArDusTS pour la détection d'anomalies dans les données sismiques de Mars

C.1 Apprentissage de représentation

Nous nous concentrons dans cette section au bloc d'apprentissage de représentation de StArDusTS. Nous étudions section C.1.1 la tâche prétexte de prédiction du futur des séries temporelles comme cela a été fait pour les séries temporelles cellulaires dans le chapitre précédent. Nous étudions ensuite une seconde tâche prétexte proposée chapitre II section D.3

C.1.1 Tâche prétexte de prédiction

Concentrons-nous dans un premier temps sur la tâche de prédiction du futur de la série temporelle. Deux points sont à définir : l'architecture du réseau de neurones ainsi que les données optimales et le choix des prétraitements à leur appliquer pour faciliter la prédiction.

Architectures Dans un premier temps, nous cherchons à appliquer le modèle StArDusTS tel qu'il a été utilisé dans le chapitre IV et notamment sa partie apprentissage de représentation. Cela implique de réutiliser des architectures avec des encodeurs CNN 1D et des décodeurs denses, entrainés sur une tâche prétexte de prédiction du futur de la série temporelle.

Dans ces expériences, nous rajoutons par ailleurs deux nouvelles variantes des architectures CNN 1D :

— Des modèles totalement convolutifs. Ces modèles présentent un encodeur composé de couches convolutionnelles à 1 dimension (comme pour l'étude précédente) et un décodeur de convolutions transposées à 1 dimension [ZTF11, DV18]. Dans l'étude précédente, la majorité des paramètres des réseaux de neurones étaient en effet concentrés dans le décodeur dense. Nous proposons ici d'étudier les capacités de réseaux de neurones avec un nombre de paramètres plus faible, toujours dans une optique d'embarquabilité, grâce à des décodeurs à convolutions 1D transposées. Ces modèles sont eux-mêmes proposés en 2 variantes :
  — La première, totalement convolutionnelle,
  — La seconde, comportant une unique couche dense au niveau du goulot d'étranglement de réseau de neurones (après l'encodeur et avant le décodeur convolutionnel). Cette couche dense peut permettre un meilleur brassage des features apprises par le réseau de neurones.

— L'utilisation d'encodeurs positionnels utilisés notamment dans [VSP+17]. Ces derniers sont ajoutés avant l'encodeur CNN 1D afin d'intégrer la notion de temporalité aux données elles-mêmes. Un encodeur positionnel ajoute des informations de position aux données avant de les envoyer dans le réseau de neurones. Ces informations de position sont généralement des vecteurs qui représentent la position relative de chaque mot dans la séquence. Ils permettent aux réseaux de neurones de capturer les dépendances séquentielles dans les données et sont assez flexibles pour s'adapter à des séquences de tailles variables (bien que cette propriété ne soit pas exploitée ici). [CLRM23] montre que l'utilisation d'encodeurs positionnels peut améliorer les résultats prédictifs d'un réseau de neurones.

Ces architectures sont entrainées à l'aide de l'outil présenté chapitre III section D. Les hyperparamètres des différentes architectures sont détaillés tableau V.2. Pour mesurer l'impact des encodeurs positionnels, les architectures totalement convolutionnelles sont testées avec et sans ces derniers. Pour toutes les architectures, le pas d'apprentissage est fixé à `0.005`, la période des couches de maxpooling est fixée à `3` couches convolutionnelles (une couche de maxpooling toutes les 3 couches convolutionnelles), le nombre de filtres par couche convolutionnelle à `64`. Deux tailles de noyaux convolutionnels sont testées : `3` et `5`. Ces hyperparamètres correspondent à ceux déjà explorés pour les CNN 1D dans le chapitre précédent et ayant obtenu les meilleures résultats.

Table V.2 : Tableau récapitulatif des hyperparamètres et des modèles entrainés à la prédiction du futur des séries temporelles de Mars

Description	Encodeur positionnel	Encodeur Conv1D	Bottleneck Dense	Décodeur Conv1D	Décodeur Dense	# conv	Taille du kernel	Taille des couches dense	Fonction d'activation
Modèle totalement convolutionnel, utilisant un encodeur ET un décodeur convolutionnel avec ou sans bottleneck dense		√		√		$3-6-9-12-15-18$	$3-5$	Sans $-64-32-16-8-2$	`relu` `tanh` `sigmoid`
Encodeur et décodeur convolutionnel + encodeur positionnel	√	√		√		$3-6-9-12-15-18$	$3-5$		`relu` `tanh` `sigmoid`
Modèle avec un encodeur convolutionnel un décodeur dense (similaire aux modèles utilisés dans le chapitre IV)		√			√	$3-6-9-12$	$3-5$	$[64,64,64]$ $[128,128,128]$ $[32,32,32]$ $[16,16,16]$ $[128,64,32]$ $[64,32,16]$	`tanh`

Quelles données prédire ? L'entrainement de réseau de neurones à prédire le futur de séries temporelles de sismologie de Mars cache en fait un double problème. En effet, plusieurs inconnues sont à déterminer dans cette section :

— À la fois, quelle architecture est le meilleure pour la prédiction du futur des séries temporelles de Mars ? Avec quelles valeurs d'hyperparamètres ? Les expériences menées

reposent sur les architectures proposées ci-dessus. Les valeurs d'hyperparamètres sont détaillées tableau V.2.

— Mais aussi quelles données utiliser et quels prétraitements supplémentaires peuvent être nécessaires afin de maximiser les performances prédictives des réseaux de neurones ? Nous explorons dans les expériences ci-dessous 8 combinaisons différentes de prétraitements et de données choisies pour la tâche prétexte de prédiction.

Afin de clarifier le cheminement scientifique suivi, nous reprenons ici la structure suivante Objectifs > Jeu de données > Résultats > Analyse, pour chacune des expériences menées. Le tableau V.3 résume l'ensemble des expériences menées sur les données et leur prétraitement.

Cette section décrit le cheminement scientifique suivi afin d'obtenir les meilleurs résultats prédictifs possibles pour pouvoir détecter les événements présents dans le signal. Une référence naïve de prédiction de la moyenne du signal d'entrée est donnée à titre de comparaison pour la performance des réseaux de neurones. Au fur et à mesure des expériences, des modalités sont rajoutées, ou les signaux sont filtrés afin d'obtenir à la fois de meilleures performances de prédiction, mais aussi rajouter de l'information qui permettra dans les sections suivantes de détecter les événements présents dans les signaux. Le tableau V.3 résume toutes les expériences menées sur les données et leur prétraitement.

Expérience 1

Objectifs : Le but de cette expérience est de reproduire les expériences menées sur les données cellulaires le plus fidèlement possible, mais aussi avec un minimum de données utilisées en entrée.
Nous entrainons donc des réseaux de neurones à prédire, du mieux possible, les signaux des données sismiques de Mars afin d'apprendre une représentation des données qui pourra ensuite être utilisée avec un détecteur. En effet, l'hypothèse 1 (page 60) présupposait que les résultats de détection d'anomalie sont proportionnels aux résultats obtenus sur la tâche prétexte.

Jeu de données : Le signal est découpé en fenêtres d'étude de 30 secondes découpées en 20 secondes en entrée des réseaux de neurones et 10 secondes à prédire (c'est-à-dire en conservant le même ratio que pour les données cellulaires). Les signaux étant échantillonnés à 20 Hz, cela représente 400 points en entrée pour 200 points en sortie par modalité. Les premières données utilisées ne contiennent que les signaux sismiques sur les 3 axes $[U, V, W]$, sans aucun prétraitement supplémentaire.

Résultats :

Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
3.82×10^{-2}	$\mathbf{3.24 \times 10^{-2}}$	3.76×10^{-2}	3.75×10^{-2}

Analyse : Les réseaux CNN avec un décodeur dense sont ceux qui permettent d'obtenir les meilleures performances prédictives. Néanmoins, ces performances sont à nuancer car le signal est qualitativement assez mal prédit, même par les meilleurs réseaux.

Expérience 2

Objectifs : La détection manuelle des dust devils est effectuée sur la modalité de pression. Peut-on rajouter cette modalité sans dégrader les performances du modèle prédictif ?
Jeu de données : Ajout de la modalité de pression P. Les données sont donc constituées des modalités $[U, V, W, P]$.

Résultats :

Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
1.21×10^{-2}	$\mathbf{7.82 \times 10^{-3}}$	1.18×10^{-2}	1.15×10^{-2}

Analyse : L'ajout de la modalité de pression ne dégrade pas les résultats de prédiction, elle semble même les améliorer. Attention cependant, la modalité de pression pèse ici pour 1/4 dans le calcul de l'EQM et est beaucoup moins bruitée que les données sismiques. Bien que les performances soient meilleures d'un point de vue de la métrique, qualitativement, ces prédictions n'offrent pas un gain de performance notable.

Expérience 3

Objectifs : Les signaux étudiés présentent une grande différence entre le jour et la nuit. La nuit, le signal n'étant pas soumis aux perturbations solaires, il est beaucoup moins bruité. L'ajout de la modalité Jour/Nuit DN permet-elle d'améliorer les résultats de prédiction ?

Jeu de données : La cinquième modalité, Jour/Nuit, est rajoutée. L'entrée est donc constituée du vecteur $[U, V, W, P, DN]$.
Le signal DN n'étant constitué que de 0 ou de 1, sa valeur permet d'apporter de l'information concernant la quantité de bruit dans les signaux. Cependant, il n'est pas intéressant comme signal à prédire à cause de sa simplicité. C'est pourquoi les signaux à prédire par le réseau de neurones en sortie restent les signaux $[U, V, W, P]$

Résultats :

Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
3.41×10^{-2}	$\mathbf{2.93 \times 10^{-2}}$	3.36×10^{-2}	3.36×10^{-2}

Analyse : L'ajout de cette modalité semble dégrader les résultats prédictifs par rapport à l'expérience précédente

Expérience 4

Objectifs : Les performances du modèle sont-elles meilleures si les données sont filtrées ?

Jeu de données : Les signaux U, V, W sont filtrés par un filtre passe bas à 5 Hz et le signal de pression P à 1 Hz (aussi par un filtre passe bas). Le signal Jour/Nuit est conservé tel quel.

Résultats :

Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
2.40×10^{-2}	$\mathbf{1.90 \times 10^{-2}}$	2.35×10^{-2}	2.33×10^{-2}

Analyse : Le filtrage des données permet de lisser le bruit très fort et donc de faciliter la tâche de prédation. L'EQM des modèles est meilleure que dans l'expérience précédente.

Expériences 5 et 6

Les tâches précédentes ont montré que la prédiction des réseaux de neurones tendait vers la prédiction de la moyenne du signal, sans pour autant y parvenir aussi bien qu'une fonction moyenne.

Objectifs : Si l'on retire la moyenne du futur du signal qui est à prédire, le modèle se concentre-t-il sur les variations du signal et améliore-t-il donc ses performances ?

Jeu de données : Pour chaque label de 10 secondes, sa moyenne lui est retirée. Ces expériences reprennent respectivement les données des expériences 3 et 4 (avec les données $[U, V, W, P, DN]$ respectivement brutes et filtrées) avec la moyenne du signal à prédire mise à 0.

Résultats :

	Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
	2.69×10^{-2}	$\mathbf{2.47 \times 10^{-2}}$	2.68×10^{-2}	2.67×10^{-2}
	1.67×10^{-2}	$\mathbf{1.44 \times 10^{-2}}$	1.66×10^{-2}	1.65×10^{-2}

Analyse : Bien que les valeurs d'EQM obtenues montrent des meilleures performances que la solution naïve de prédire toujours 0, ces résultats sont à nuancer. En effet, même lorsque l'entrée présentée est du bruit généré artificiellement, les réseaux de neurones proposent une solution environ égale à 0, et ce, quelle que soit l'entrée, ce qui n'est pas une solution satisfaisante. En effet, la prédiction de la constante 0 permet d'obtenir une meilleure valeur d'EQM. C'est pourquoi cette piste est abandonnée par la suite.

Expérience 7

Au regard de [BLK+21], le contenu fréquentiel des événements recherché est situé à des fréquences de 1 Hz ou moins.

Objectifs : Décimer le signal peut-il augmenter les performances de prédiction des réseaux de neurones ?

Jeu de données : Les signaux $[U, V, W, P, DN]$ filtrés sont décimés à une fréquence de 2 points par seconde afin de conserver le contenu fréquentiel au-dessous de 1 Hz. La dimension temporelle de 30 secondes des fenêtres est conservée, elles durent donc respectivement 40 et 20 points en entrée et en sortie.

Résultats :

Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
1.20×10^{-2}	$\mathbf{7.52 \times 10^{-3}}$	1.11×10^{-2}	1.10×10^{-2}

Analyse : On constate que les modèles ont tendance à recopier leur entrée en sortie. Bien que cette solution permette d'obtenir de bonne métrique, elles ne constituent pas de prédictions satisfaisantes.

Expérience 8

La décimation du signal peut aussi permettre d'augmenter la taille des fenêtres d'entrée et de sortie pour conserver des tailles (en nombre de points) identiques, mais des durées différentes.

Objectifs : L'utilisation de fenêtres avec une durée plus longue permise grâce à la décimation des signaux peut-elle permettre d'augmenter les performances des réseaux de neurones ?

Jeu de données : Les signaux $[U, V, W, P, DN]$ filtrés sont décimés à une fréquence de 2 points par seconde afin de conserver le contenu fréquentiel au-dessous de 1 Hz. Les longueurs des fenêtres d'entrées et de sorties sont augmentées pour atteindre respectivement 128 et 64 points soit 64 secondes en entrée et 32 secondes en sortie.

Résultats :

Référence naïve	CNN-dense	CNN-Conv1D^T	CNN-Conv1D^T + PE
1.07×10^{-2}	$\mathbf{9.70 \times 10^{-3}}$	1.14×10^{-2}	1.15×10^{-2}

Analyse : Bien que ces modèles obtiennent des scores meilleurs que la prédiction naïve, les meilleurs sont cependant des modèles recopiant le signal présenté en entrée en sortie. Encore une fois, cela se vérifie lorsque le signal présenté est du bruit généré aléatoirement. Seuls les modèles avec des décodeurs denses n'ont pas ce problème. Ces dispositifs expérimentaux ne seront pas réutilisés par la suite, que ce soit pour la détection d'anomalies (section C.2) ou la tâche prétexte d'apprentissage contrastif section C.1.2.

Le nombre de paramètres est une valeur qui reflète la complexité et l'embarquabilité du modèle. Elle n'est cependant pas discutée ici pour une raison. Nous considérons en effet que comparer la taille des modèles n'a de sens que si ces derniers parviennent à prédire le futur des séries temporelles qui leur sont présentés. Comparer la performance des réseaux de neurones en fonction de leur taille n'aurait pas de sens sachant que les performances prédictives de meilleurs réseaux de neurones sont médiocres.

Conclusion Nous avons tenté lors des expériences proposées de rajouter une architecture de réseau de neurones totalement convolutionnelle utilisant des couches de convolution transposées. Ces dernières se sont cependant montrées moins performantes que les de simples couches denses comme décodeurs. Cela peut s'expliquer par la complexité de la tâche prétexte de prédiction et du lien faible que peut avoir le signal à prédire avec le signal en entrée.

Nous avons par ailleurs considéré la temporalité des données en rajoutant des encodeurs positionnels. Les expériences ont montré que les résultats obtenus avec des encodeurs positionnels sont meilleurs que ceux obtenus uniquement avec les mêmes architectures totalement convolutionnelles sans encodeur positionnel. Cependant, le gain en performance reste minime face aux décodeurs denses.

Le meilleur réseau de neurones de l'expérience 5 présente une amélioration de la référence naïve d'environ 18% alors que l'amélioration apportée par le modèle 3 avec les

données non filtrées n'est que de 14%. C'est pourquoi, dans la suite de l'étude, les données utilisées sont les données $[U, V, W, P, DN]$ filtrées à 5 Hz pour les séries temporelles sismiques et 1 Hz pour la série temporelle de pression.

Néanmoins, il est important de préciser que les résultats prédictifs restent qualitativement mauvais. Les réseaux de neurones ne sont pas capables de bien prédire les 10 secondes futures du signal connaissant les 20 secondes passées. Or, l'hypothèse émise pour les données cellulaires (p 60) reste valable pour les données de Mars : Plus la métrique sur la tâche prétexte est bonne, plus la représentation sera efficace pour la tâche finale que l'on cherche à réaliser. Cela suppose que si le modèle n'est pas capable d'accomplir la tâche prétexte, la représentation apprise ne sera pas d'un grand intérêt pour la tâche finale que l'on cherche à accomplir. L'utilisation de réseaux de neurones plus profonds, d'autres architectures comme les LSTM ou les transformers, ou encore d'autres configurations ou prétraitements auraient pu être des pistes à creuser pour améliorer les performances prédictives obtenues. Nous avons cependant choisi d'orienter cette étude dans une autre direction qui n'avait encore pas été explorée auparavant (dans ce manuscrit) : l'utilisation d'une autre tâche prétexte pour apprendre la représentation des données.

Ces expériences menées sur la tâche prétexte de prédiction n'affirment pas qu'il soit impossible de prédire le futur du signal des données sismiques de Mars, mais montrent que les modèles utilisés doivent être choisis avec précaution. De manière plus générale, il est difficile d'affirmer que ces données ne sont pas prédictibles, mais à notre connaissance, aucun article n'a encore réussi à prédire le futur des données sismiques de Mars. Cette piste de la non-prédictibilité des données ne doit pas être exclue pour expliquer les mauvaises performances prédictives obtenues par les modèles.

Table V.3 : EQM de la meilleure architecture pour chaque famille en fonction des données utilisées et du prétraitement associé. La meilleure famille d'architecture est mise en gras dans le tableau

n°	1	2	3	4	5	6	7	8
U,V,W	√	√	√	√	√	√	√	√
P	-	√	√	√	√	√	√	√
DN	-	-	√	√	√	√	√	√
Filtré	-	-	-	√	-	√	-	-
Moyenne = 0	-	-	-	-	√	√	-	-
Décimé 2Hz	-	-	-	-	-	-	√	√
Taille E/S (s)	20/10	20/10	20/10	20/10	20/10	20/10	20/10	64/32
Taille E/S (points)	400/200	400/200	400/200	400/200	400/200	400/200	40/20	128/64
CNN-dense	$\mathbf{3.24 \times 10^{-2}}$	$\mathbf{7.82 \times 10^{-3}}$	$\mathbf{2.93 \times 10^{-2}}$	$\mathbf{1.90 \times 10^{-2}}$	$\mathbf{2.47 \times 10^{-2}}$	$\mathbf{1.44 \times 10^{-2}}$	$\mathbf{7.52 \times 10^{-3}}$	$\mathbf{9.70 \times 10^{-3}}$
CNN-Conv1D^T	3.76×10^{-2}	1.18×10^{-2}	3.36×10^{-2}	2.35×10^{-2}	2.68×10^{-2}	1.66×10^{-2}	1.11×10^{-2}	1.14×10^{-2}
CNN-Conv1D^T + PE	3.75×10^{-2}	1.15×10^{-2}	3.36×10^{-2}	2.33×10^{-2}	2.67×10^{-2}	1.65×10^{-2}	1.10×10^{-2}	1.15×10^{-2}
Solution naïve :								
Moyenne de l'entrée	3.82×10^{-2}	1.21×10^{-2}	3.41×10^{-2}	2.40×10^{-2}	-	-	1.20×10^{-2}	1.07×10^{-2}
Prédiction 0	-	-	-	-	2.69×10^{-2}	1.67×10^{-2}	-	-

C.1.2 Tâche prétexte d'apprentissage contrastif

Afin de répondre aux faibles capacités prédictives observées dans la section précédente, nous proposons d'expérimenter l'usage d'une nouvelle tâche prétexte pour l'apprentissage de représentation de StArDusTS l'apprentissage contrastif. Le principe de l'apprentissage contrastif, déjà présenté chapitre II section D.3, est le suivant. Une fenêtre d'observation fixée est transformée grâce à une ou plusieurs méthodes d'augmentation de données, ces deux versions de la même observation forment une paire positive. Le réseau de neurones utilisé projette alors chacune de ces versions de la même fenêtre dans son espace latent. Pendant l'entrainement du réseau de neurones, la fonction de cout InfoNCE équation (II.10) s'efforce de maximiser la similarité entre les éléments d'une paire positive. Parallèlement, la fenêtre originelle peut être comparée à n'importe quelle autre fenêtre du jeu de données, qui forment ensemble une paire négative, et dont la représentation doit être la plus dissimilaire possible. Le dénominateur de l'InfoNCE joue ce rôle conjoint de maximisation de la similarité entre les paires positives et minimisation de la similarité entre les paires négatives d'un batch.

Objectifs : Entrainer des réseaux CNN 1D sur une tâche prétexte contrastive et comparer les différentes méthodes d'augmentation de données sur le jeu de données de sismologie de Mars.

Architectures : Toujours dans le même objectif, l'architecture reprend les résultats obtenus lors des expériences précédentes. C'est donc une architecture à `9` couches convolutionnelles 1D, composées de `64` filtres, avec une couche de `maxpooling` toutes les 3 couches convolutionnelles. Des fonctions d'activation `tanh` sont utilisées après chaque couche convolutionnelle.

Attention, un point diffère comparé à la tâche prétexte de prédiction : la tâche prétexte d'apprentissage contrastif ne nécessite pas de décodeur pour passer de l'espace latent à une prédiction. Le réseau de neurones n'est donc constitué que d'un encodeur convolutif à une dimension. Nous proposons cependant d'effectuer des expériences afin de mesurer l'impact de la taille de l'espace latent sur la qualité de la représentation apprise. Dans cet objectif, la partie décrite ci-dessus du réseau de neurones reste identique, mais peut être couplée à une unique couche dense afin de projeter les données dans un espace latent de dimension contrôlée. Les expériences sont menées avec 4 tailles d'espace latent : `64`, `128`, `256` et sans l'ajout d'une couche dense, `4800`. Pour des raisons pratiques, une couche `flatten` est rajoutée à la fin du réseau sans projecteur dense.

C'est donc une seule architecture qui sera étudiée dans cette section avec la valeur de l'hyperparamètre de la taille de l'espace latent pouvant prendre 4 valeurs.

Remarque 11 : Calcul de la dimension de l'espace latent

Les données d'entrées sont de taille [600, 5] avec dans l'ordre le nombre d'instants de mesure et le nombre de features. La taille des données est divisée par 2 à chaque couche de `maxpooling`. Comme une couche de maxpooling est insérée toutes les 3 couches de CNN 1D.

$$\text{Taille espace latent} = \frac{\text{longueur entrée}}{2^{\text{nb maxpooling}}} \cdot \text{nb filtres}$$
$$\text{Taille espace latent} = \frac{600}{2^3} \cdot 64$$
$$\text{Taille espace latent} = \mathbf{4800}$$

Jeux de données Pour pouvoir comparer les modèles avec une tâche prétexte de prédiction avec ceux avec une tâche prétexte d'apprentissage contrastif, nous avons choisi de fixer les données utilisées. Ce sont donc celles ayant permis d'obtenir les meilleurs résultats prédictifs qui sont utilisées soit les données de l'expérience 5 : $[U, V, W, P, DN]$ filtrées à 5 Hz pour les séries temporelles sismiques et 1 Hz pour la série temporelle de pression. Ce choix ne garantit pas l'obtention des meilleures performances possibles avec la tâche prétexte d'apprentissage contrastif, mais il permet cependant de fixer la variable du choix de données afin de mieux comparer ces deux méthodes apprentissage de représentation. Les données sont donc fixées pour toutes les expériences de la section C.1.2.

Augmentations de données L'apprentissage contrastif est profondément impacté par les transformations choisies pour augmenter les données. Afin de pouvoir se comparer à [PCS22], nous proposons d'utiliser les mêmes méthodes d'augmentation de données, déjà présentées chapitre II section E.3 qui pour rappel sont :

(0) Aucune transformation
(1) Retournement gauche-droite
(2) Renversement bidirectionnel
(3) Permutations aléatoires
(4) Bruit aléatoire
(5) Blocage
(6) Rogner et redimensionner
(7) Déformation de la temporalité
(8) Lissage aléatoire
(9) *Lissage avec fenêtre glissante*

Nous entrainons les réseaux contrastifs en utilisant les augmentations une à une, afin de mesurer l'impact d'une transformation donnée sur les performances du modèle. Bien qu'il soit possible de coupler plusieurs transformations entre elles, nous choisissons donc de comparer seulement la représentation d'une donnée brute (non transformée) à sa version augmentée, les deux devant alors être proches dans l'espace latent. Sur les deux têtes du réseau contrastif l'une n'applique donc pas de transformation alors que l'autre applique une des transformations citées ci-dessus (fixée pour un entrainement).

Enfin nous ajoutons une 9^{eme} transformation qui est une version modifiée du lissage aléatoire. Au lieu de modifier chaque point en faisant une moyenne pondérée d'un paramètre λ entre ses 2 voisins (le point suivant et le point précédent), le lissage par fenêtre glissante utilise une moyenne glissante. La taille de la fenêtre est tirée aléatoirement à chaque fois entre `1` et une valeur maximale définie par l'utilisateur `max_length`. Le lissage proposé dans [PCS22] ne lisse que très faiblement nos signaux, c'est pourquoi nous proposons cette version modifiée de la méthode de lissage, qui lisse beaucoup plus les signaux.

Pour les méthodes d'augmentation de données nécessitant un choix des paramètres, les valeurs suivantes sont prises :

(4) Bruit aléatoire : Facteur d'amplitude du bruit $\lambda = 1$,

(5) Blocage : La taille du blocage est fixée à 150 points, soit 7.5 secondes. L'instant auquel commence le blocage est lui tiré aléatoirement,

(6) Rogner et redimensionner : La taille du rognage est fixée à la moitié du signal, l'instant du début du rognage est tiré aléatoirement à chaque fois.

(8) Lissage aléatoire : le paramètre λ est tiré à chaque augmentation aléatoirement dans l'intervalle $[0, 1]$,

(9) Lissage avec fenêtre glissante : taille de la fenêtre glissante = 100 points soit 5 secondes

Résultats et discussion Au total 9 transformations sont étudiées. Comme 4 tailles d'espace latent sont imposées, cela fait 36 entrainements au total. Les entrainements sont réalisés avec un `early stopping`, paramétré avec une patience de 5 époques. Un batch est composé de `64` fenêtres différentes. Artificiellement, la taille du batch est augmentée à `128` car, pour chaque fenêtre, sa version modifiée est aussi ajoutée. Les versions normales et augmentées forment les paires positives 2 à 2 et les autres échantillons du batch constituent l'ensemble des paires négatives comme dans [CKNH20].

La métrique utilisée pour comparer les modèles est la InfoNCE équation (II.10). La figure V.3 présente les valeurs d'InfoNCE de manière visuelle en fonction de la méthode d'augmentation de données et de la taille de l'espace latent. Le tableau V.4 indique les résultats complets des mêmes expériences.

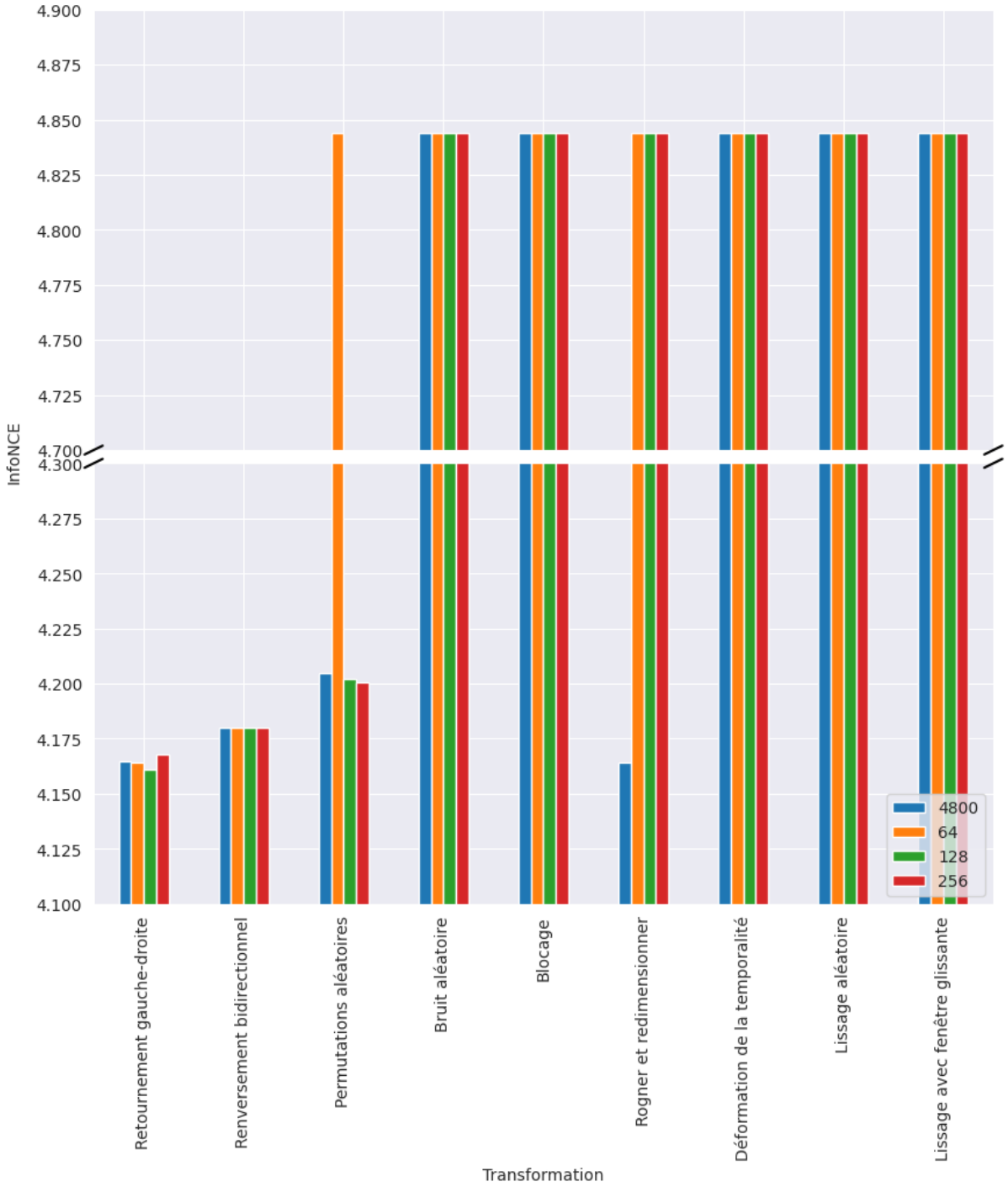


Figure V.3 : InfoNCE mesurée sur le jeu de validation en fonction de la méthode d'augmentation de données (en abscisse) et de la taille de l'espace latent (couleur des barres)

Parmi les résultats d'entrainements obtenus, nous constatons 2 régimes.

— Les modèles ne parvenant pas à apprendre une représentation des données. Ces derniers sont les modèles dont l'infoNCE plafonne à une valeur seuil de 4.8438. En observant les courbes d'apprentissage de ces réseaux, on observe que la valeur de la fonction de cout diminue lors des premiers batchs puis atteint un plafond qui ne parvient pas à être franchi. Dans ce contexte, on peut considérer que ces architectures n'ont pas réussies à être entrainées. Les raisons peuvent être multiples, mais semblent aller dans le sens des observations faites dans la section précédente : les

Table V.4 : Tableau complet des résultats d'infoNCE obtenus pour chaque transformation et chaque taille d'espace latent. La meilleure taille d'espace latent pour une transformation donnée (c.-à-d. sur une ligne) est affichée en **gras** sauf pour les modèles restant à la valeur plafond. La meilleure valeur totale est soulignée

Taille de l'espace latent	0	64	128	256
Retournement gauche-droite	4.16476	4.16404	**4.16127**	4.16805
Renversement bidirectionnel	**4.17990**	4.17994	4.17994	4.17994
Permutations aléatoires	4.20487	4.84389	4.20229	**4.20064**
Bruit aléatoire	4.84389	4.84389	4.84389	4.84389
Blocage	4.84389	4.84389	4.84389	4.84389
Rogner et redimensionner	**4.16436**	4.84389	4.84389	4.84389
Déformation de la temporalité	4.84389	4.84389	4.84389	4.84389
Lissage aléatoire	4.84389	4.84389	4.84389	4.84389
Lissage avec fenêtre glissante	4.84389	4.84389	4.84389	4.84389

données utilisées sont particulièrement bruitées et la phrase d'apprentissage de représentation n'est pas aussi facile qu'elle l'était dans le chapitre sur les données cellulaires. Les augmentations concernées par ce régime sont le bruit aléatoire, le blocage, la déformation de la temporalité, le lissage aléatoire et le lissage avec fenêtre glissante.

— Le deuxième régime concerne les transformations pour lesquelles quelque chose a été appris par le réseau de neurones et la valeur de la métrique a réussi à passer en dessous du plateau atteint par les autres transformations. Quatre méthodes d'augmentations sont concernées : le retournement gauche-droite, le renversement bidirectionnel, la permutation aléatoire et le rognage et redimensionnement. On constate que pour les deux premières transformations, les valeurs de métriques sont presque identiques, peu importe la taille de l'espace latent. Cette constance montre que les réseaux réussissent bien à projeter les données proches dans l'espace latent, peu importe sa taille et que l'encodeur est déjà assez générique pour pouvoir traiter les données, sans pour autant nécessiter la transformation non linéaire apportée par un projecteur dense en dernière couche.

On remarque deux points singuliers dans ces entrainements. Tout d'abord pour la transformation de permutations aléatoires, le réseau avec un espace latent de taille 64 n'a pas réussi à franchir le plafond. Une des pistes d'explication est la faible taille de l'espace latent qui ne permet pas de capturer toute la complexité de cette augmentation de données ainsi que celle des données elles-mêmes. Avec les degrés de liberté offerte par des espaces latents plus grands, le modèle peut alors converger. On constate par ailleurs que pour l'augmentation rognage et redimensionnement, seul l'espace latent sans projection dense réussit à tirer un apprentissage des données. Cette fois ci, une seconde explication peut être avancée. Contrairement aux modèles avec un projecteur dense, les modèles en bleu figure V.3 sont des modèles totalement convolutionnels. Nous avions par ailleurs constaté dans la section C.1.1 que les modèles totalement convolutionnels performaient peu sur la tâche prétexte de prédiction, car ils avaient tendance à prédire un signal ressemblant au signal d'entrée. En effet, l'aspect totalement convolutif de réseau de neurones ne permet pas un *mélange* des instants, contrairement aux couches denses. Ce *mélange* peut alors apporter des degrés de liberté supplémentaires, voire trop, ce qui conduit les modèles avec des projecteurs denses à ne pas bien capturer l'aspect temporel du signal et à ne pas apprendre une bonne représentation des données, et donc rester au niveau du plafond d'apprentissage.

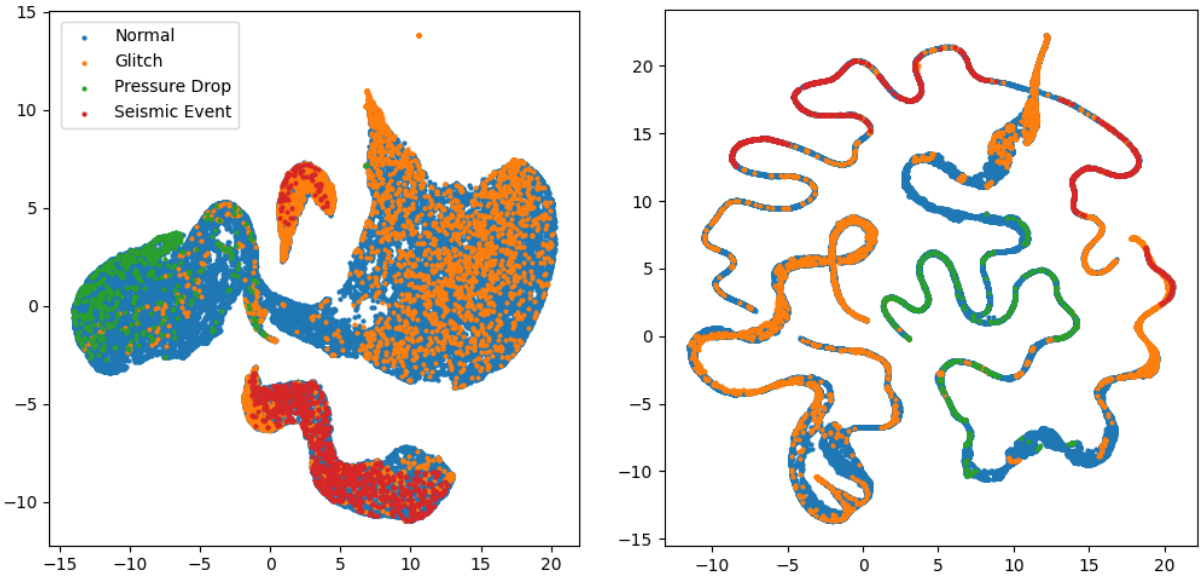
Bien que nous parlions ici de résultats intermédiaires et pas de résultats finaux sur la tâche de détection d'anomalies, il est raisonnable de supposer que les modèles n'ayant pas réussi à apprendre de représentation des données ne seront pas des extracteurs de features performants pour la détection d'anomalies. Ces résultats intermédiaires sont sensiblement différents de ceux obtenus par [PCS22]. D'un côté, on observe que les transformations (2) et (3) ont été entrainées avec succès alors que ces transformations n'apportaient pas de gain de performance dans le papier. De l'autre, les transformations (4) et (6) qui apportaient un gain de performance pour la détection d'anomalies dans [PCS22] n'ont même pas réussi à être entrainées. Cela montre qu'en fonction des architectures choisies et des données utilisées, les résultats pourraient être totalement différents et ne peuvent donc pas être généralisés.

Évaluation qualitative de la représentation apprise L'intérêt de l'apprentissage contrastif était d'organiser l'espace latent afin d'obtenir une segmentation facile de ce dernier. La figure V.4 montre un exemple de projection Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [MHM20] d'un modèle que nous avons réussi à entrainer figure V.4a et d'un qui a atteint le plafond figure V.4b. Les couleurs représentent les classes des données que ne sont affichées qu'à des fins d'interprétation.

Il est important de rappeler que la forme de la projection UMAP ne peut pas être interprétée. Néanmoins, la répartition des points dans cette représentation peut l'être. Le cas idéal que l'on aurait souhaité observer est une projection dans laquelle les différentes classes forment des regroupements bien distincts les uns des autres, ou au moins observer les anomalies séparées des données normales.

Cependant, l'une comme l'autre des projections montrent le même problème : aucune des classes d'événements n'est facilement séparable de la classe normale. Les distances et les regroupements étant conservés avec cette transformation, on peut donc affirmer que les données normales et les données anormales sont projetées dans les mêmes régions de l'espace latent, et ce malgré l'utilisation de l'apprentissage contrastif. En effet, l'objectif de l'apprentissage contrastif était d'apprendre une représentation dans laquelle les données anormales étaient isolées du reste des données sans pour autant avoir à disposition les annotations relatives aux données.

Attention cependant, comme pour le chapitre précédent, les analyses possibles grâce à cette projection sont des pistes permettant d'orienter nos réflexions. La projection des données d'un espace de dimension 128 vers un espace de dimension 2 ne permet pas de capturer l'ensemble de la complexité de la représentation. Elle permet néanmoins d'avoir une idée de la complexité de la tâche de détection d'anomalies à partir de la représentation apprise grâce à l'apprentissage contrastif. Il n'est pas impossible qu'en dimension 128, les données soient linéairement ou non séparables, c'est d'ailleurs pour cette raison que les expériences suivantes sont menées : pour confirmer ou infirmer les intuitions données par la projection UMAP sur la qualité de la représentation apprise grâce à l'apprentissage contrastif.



(a) Retournement gauche-droite **(b)** Rogner et redimensionner

Figure V.4 : Projection UMAP du jeu de test des modèles contrastifs entrainés avec les augmentations de données (1) figure V.4a et (6) figure V.4b avec un espace latent de taille 128

C.1.3 Conclusion sur l'apprentissage de représentation

Les expériences menées sur les données sismiques de Mars ont pour but de vérifier la transférabilité du modèle StArDusTS à d'autres jeux de données. C'est en sachant cela que les expériences ont été menées. C'est pourquoi, les blocs d'apprentissage de représentation et de détection d'anomalies sont entrainés et optimisés, comme dans le chapitre précédent, indépendamment.

Deux types de conclusions peuvent être tirées des expériences sur l'apprentissage de représentation, d'un côté les conclusions méthodologiques et de l'autre les conclusions applicatives.

Conclusions méthodologiques Nous avons constaté que la tâche prétexte de prédiction n'est pas toujours suffisante. En effet, en fonction des données traitées, la prédiction du futur de ces dernières peut s'avérer difficile. Des modèles d'apprentissage de représentation plus complexe comme les transformers [VSP+17] pourraient être utilisés pour espérer obtenir de meilleures performances, mais nous avons choisi de nous en tenir à des modèles légers dans une optique d'*embarquabilité* lors de cette étude. Nous avons essayé d'intégrer l'aspect temporel des données grâces aux encodeurs positionnels, mais les résultats prédictifs obtenus ne sont pas plus concluants avec que sans ces derniers.

Cependant, nous avions émis chapitre III point 1 l'hypothèse que la performance de détection d'anomalies repose sur la performance obtenue sur une tâche prétexte. C'est pourquoi nous avons proposé une seconde tâche prétexte utilisée dans la littérature (notamment en traitement d'image) : l'apprentissage contrastif. Nous avons pu observer que toutes les méthodes d'augmentation de données ne se valent pas et que deux régimes sont constatés. D'un côté certaines méthodes d'augmentation de données ne parviennent pas à passer un palier d'apprentissage. Aucun élément probant ou piste d'explication concluante n'a été identifié pour rendre compte de ce phénomène. Les résultats obtenus demeurent inhabituels, des études supplémentaires sont nécessaires pour les expliquer. De l'autre côté, les méthodes d'augmentation de données les plus prometteuses, qui sont le retour-

nement gauche-droite, le renversement bidirectionnel et la permutation aléatoire performent quasiment toutes aussi bien, peu importe la taille de l'espace latent choisi. Les réseaux convolutionnels à une dimension semblent donc adaptés pour l'apprentissage de représentation contrastif appliqué à ces séries temporelles, sans forcément nécessiter de projecteur dense.

Enfin, pour la tâche prétexte d'apprentissage contrastif, une visualisation des classes dans l'espace latent obtenu suite à l'apprentissage est proposé sous la forme d'une projection UMAP. Cette visualisation nous permet d'évaluer qualitativement la qualité de la représentation apprise. On y constate que les différentes classes du jeu de données sismique de Mars ne peuvent pas être facilement séparées comme on aurait pu l'espérer. Les expériences de détection d'anomalies menées dans la section suivante vont permette de confirmer la qualité de la représentation en l'utilisant sur la tâche d'intérêt qu'est la détection d'anomalies.

Conclusions applicatives La première question qu'il est légitime de se poser en constatant les mauvais résultats prédictifs des réseaux convolutionnels est si la complexité des séries temporelles traitées ne rend pas ces séries temporelles *non prédictibles*. Aucune méthode à notre connaissance ne permet d'affirmer qu'il est impossible quel que soit le modèle, de prédire le futur d'une série temporelle. Cependant, les séries temporelles choisies, ainsi que la taille de fenêtre choisie sont des paramètres qui peuvent rendent cette tâche plus facile ou plus difficile. Prédire la météo plusieurs semaines à l'avance est une tâche quasi impossible, il est possible de faire le parallèle en disant que le choix de cette taille de fenêtre était peut-être trop ambitieux pour les modèles sélectionnés. Pour conclure, les données de Mars sont plus complexes à prédire que les données cellulaires.

Concernant les traitements appliqués aux données, on observe grâce aux expériences menées dans cette section que le filtrage des données permet d'obtenir de meilleurs résultats prédictifs que lorsque l'on utilise les données non filtrées.

Enfin, un dernier point est à prendre en compte : on se place lors de cette étude dans un scénario particulier d'apprentissage dans lequel un des événements que l'on cherche à détecter n'est pas présent dans le jeu de validation. Ce scenario est très plausible dans le cas d'un apprentissage embarqué réalisé avec un capteur intelligent, le développement et l'utilisation de méthodes prenant en compte ces biais est une piste de développement future des travaux proposés. Le zero-shot learning [XLSA20] traite de problématiques similaires où certaines classes n'ont pas été vues en entrainement alors qu'elles peuvent apparaitre dans le jeu de test.

C.2 Détection d'anomalie

Le modèle StArDusTS proposé section B est à la fois composé d'un bloc d'apprentissage de représentation que nous venons d'étudier dans la section précédente et d'un bloc de détection d'anomalies. Nous nous attardons ici sur ce second bloc de détection d'anomalies.

Objectif :

Nous cherchons au travers des expériences menées dans cette section à déterminer les combinaisons d'apprentissage de représentation et de détecteur permettant d'obtenir les meilleures performances de détection d'anomalies.

Plan d'expérience :

Nous utilisons l'ensemble des représentations apprises, à la fois grâce à la tâche prétexte de prédiction et celle d'apprentissage contrastif et les comparer entre elles pour réaliser la détection d'anomalies. Les features extraites grâce à ces modèles sont données en entrée des 4 détecteurs présentés chapitre II section A.

Dans un premier temps, nous cherchons à isoler les événements du jeu de données, quels qu'ils soient. Autrement dit, on se rapporte à un problème de détection d'anomalies et non pas à un problème de classification à 4 classes avec les annotations présentées tableau V.1. Le label " `anormal` " est donc associé aux glitchs, aux chutes de pression et aux événements sismiques.

Le détecteur à seuil nécessite de pouvoir comparer une prédiction à une vérité terrain. Ce dernier est pertinent dans le cas de la tâche prétexte de prédiction ou la prédiction est comparée aux vraies valeurs de la série temporelle. Il est utilisé dans la même configuration que pour les expériences cellulaires, avec un seuil τ fixé sur le jeu d'entrainement de manière à classer comme anormales les prédictions pour lesquelles l'écart est en dehors de l'intervalle de confiance à 95%. Cependant, avec la tâche prétexte d'apprentissage contrastif, les données sont simplement projetées dans l'espace latent grâce au modèle d'apprentissage profond. Ce détecteur ne peut donc pas être utilisé sur la tâche prétexte d'apprentissage contrastif.

Les 3 autres types de détecteurs : les modèles OCSVM, les modèles LOF et les isolations forests sont quant à eux utilisés à la fois sur les issues de la tâche prétexte de prédiction et d'apprentissage contrastif car ces derniers génèrent un score, indépendant d'une quelconque vérité terrain.

Métrique de comparaison Dans un scénario comme les données sismiques de Mars où le jeu de données est composé à 90% d'éléments normaux, si un modèle classifie simplement tous les exemples comme étant normaux, il atteindra une exactitude de 90%, ce qui peut sembler élevé, mais n'est pas du tout utile pour détecter les anomalies. Cependant, avec l'exactitude équilibrée, un modèle qui prédit tout comme normal atteindra une exactitude équilibrée de seulement $50\,\%$, ce qui est bien plus révélateur de l'inefficacité du modèle pour détecter les anomalies.

Nous utilisons donc l'exactitude équilibrée pour comparer les performances de détection d'anomalies des différents modèles StArDusTS.

Résultats :

La figure V.5 présente de manière graphique l'ensemble des résultats obtenus pour les 4 détecteurs sur toutes les méthodes d'augmentation de données de l'apprentissage contrastif et de la prédiction. Chaque sous figure correspond à une méthode d'augmentation de données, tandis que les couleurs des points correspondent aux détecteurs. Un tableau contenant l'ensemble des résultats est disponible en annexe tableau C.1.

Analyse :

Le résultat majeur à retenir de ces expériences est le suivant : peu importe la représentation extraite, aucune d'entre elles n'est d'une quelconque utilité pour les détecteurs expérimentés puisque l'exactitude équilibrée ne dépasse jamais les 50%. Cela signifie qu'aucun modèle, peu importe le détecteur, la taille de l'espace latent ou la tâche prétexte utilisée pour l'apprentissage de représentation n'est capable de dépasser les performances d'un modèle aléatoire.

Aucun détecteur ne semble supérieur aux autres, quelle que soit la méthode d'augmentation de données considérée. Ce point nous permet de remettre en cause non pas les modèles de détection d'anomalies, mais plutôt la représentation qui a été apprise en amont.

Cette section avait pour but de pouvoir comparer les différents détecteurs proposés face aux architectures neuronales et aux tâches prétextes choisies pour l'apprentissage de représentation de StArDusTS, cependant, comme aucun détecteur n'est capable de détecter des anomalies, il n'est pas possible de les comparer sur les représentations apprises précédemment. Ces résultats confirment les pressentiments qualitatifs émis dans la section sur l'apprentissage de représentation. Comme on pouvait le constater sur la figure V.4 les différents événements ne peuvent pas être séparés facilement, peu importe le détecteur choisi. Par ailleurs, ces 4 détecteurs ayant déjà fait leurs preuves dans de nombreuses applications, ce ne sont pas les points de StArDusTS provoquant ces mauvaises détections ici. Tous ces résultats semblent plutôt remettre en cause les représentations apprises dans la section précédente. Cette discussion, ne concernant pas les détecteurs d'anomalie à proprement parler, est donc menée dans la conclusion du chapitre.

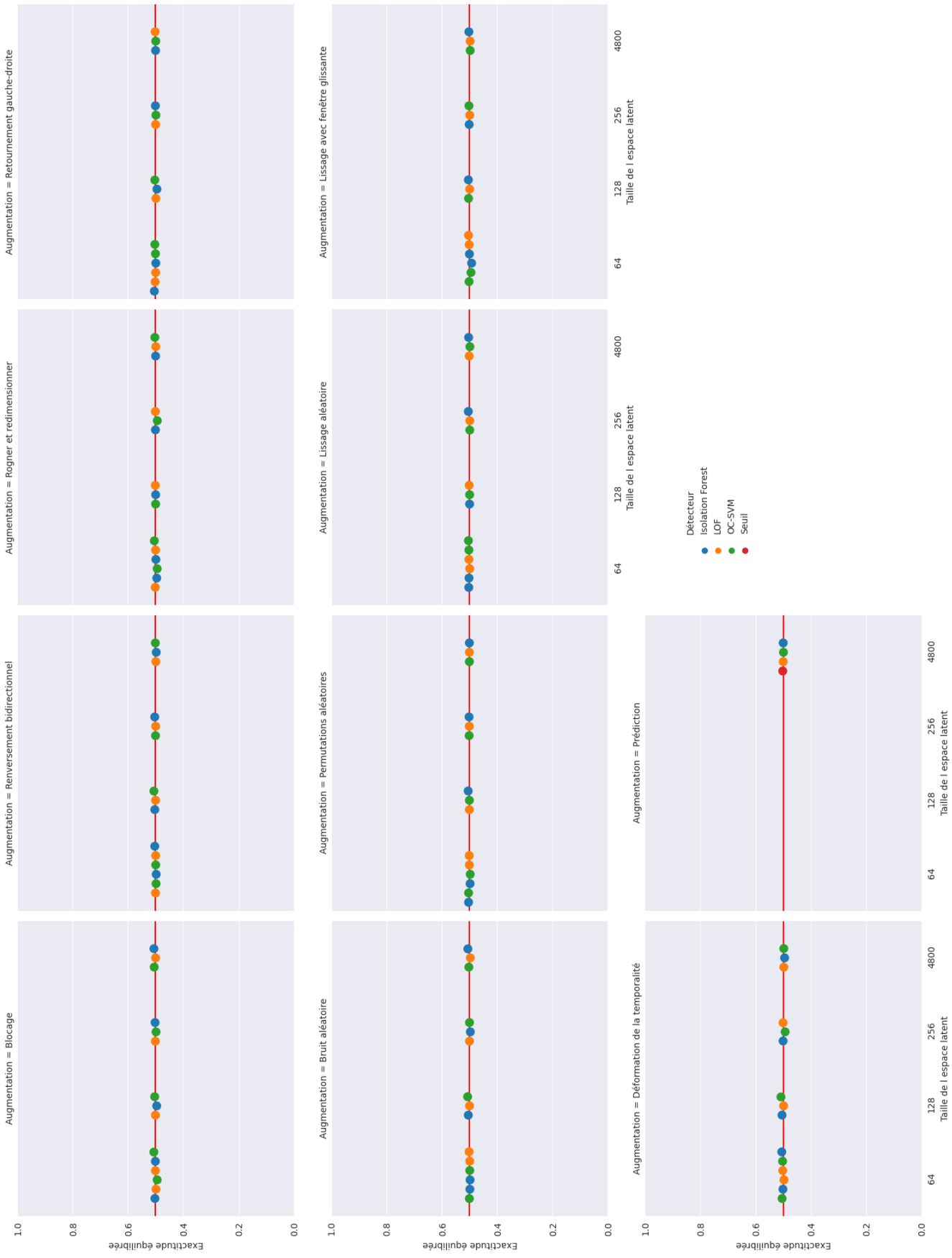


Figure V.5 : Ensemble des résultats des détecteurs sur les données de Mars. Chaque sous graphique est soit une méthode d'augmentation de données différente pour l'apprentissage contrastif soit les features extraites grâce à la tâche prétexte de prédiction pour le dernier encart. La ligne rouge est l'exactitude équilibrée d'une classification aléatoire. Aucun des modèles ne dépasse significativement les performances d'un détecteur aléatoire.

Bien que les résultats obtenus ne soient pas ceux attendus d'un point de vue applicatif, car le modèle StArDusTS n'a pas été capable, dans ces configurations, de détecter les événements que nous cherchions à détecter dans les données, nous pouvons tirer quelques conclusions méthodologiques sur le modèle StArDusTS en général et son bloc d'apprentissage de représentation en particulier.

D Conclusion

Ce chapitre traite de la détection automatique d'événements dans les signaux sismiques de Mars. Ces événements peuvent être de nature différente et prennent la forme de glitchs, de dust devils ou chutes de pression ou encore d'événements sismiques.

Dans un premier temps, nous avons cherché à évaluer l'efficacité du modèle StArDusTS, dans sa forme utilisée pour la détection d'anomalies sur les données cellulaires, sur un nouveau jeu de données qui ne possédait pas exactement les mêmes propriétés. En effet, les données sismiques de Mars étaient pour nous un bon moyen de mesurer les capacités de ce modèle en essayant d'accomplir une tâche de détection plus complexe. Les données martiennes sont plus bruitées et présentent une périodicité de l'intensité du bruit (le bruit augmente en journée et diminue la nuit) ce qui n'était pas le cas auparavant. Par ailleurs, alors que les données cellulaires étaient constituées d'une dizaine de milliers de cellules pendant un temps *réduit*, une seule grande acquisition, longue d'un mois, est découpée en segments à analyser. Enfin, le passage aux données sismiques constitue aussi une grande avancée en analysant des signaux multimodaux. Une comparaison plus détaillée des jeux de données et de leur complémentarité est détaillée section B.2.

Les faibles capacités prédictives obtenues par les réseaux de neurones nous ont poussé à expérimenter l'utilisation d'une nouvelle tâche prétexte, l'apprentissage contrastif. Nous étudions dans ce chapitre neuf méthodes différentes d'augmentations de données à une dimension afin de générer les paires positives de l'apprentissage contrastif. Toujours couplé à des architectures du réseau de neurones CNN 1D, seul un faible nombre de méthodes d'augmentation de données parviennent à franchir un plafond d'apprentissage et, en quelque sorte, à apprendre quelque chose des données pour en tirer une représentation. Cependant, déjà qualitativement, en étudiant la répartition des données des différentes classes dans l'espace latent, on se rend compte que, même pour les architectures ayant dépassé le plafond d'apprentissage, la séparation des événements des données normales n'est pas évidente.

Ce pressentiment se confirme dans la section C.2, dans laquelle 4 détecteurs d'anomalies sont comparés : le seuil, le LOF, le OCSVM et le random forest. Ces derniers prennent en entrées les features obtenues grâce aux réseaux de neurones *entrainés* précédemment et doivent détecter les données anormales, quelle que soit leur anomalie (glitch, chute de pression ou événement sismique). Les résultats sont sans appel : aucune des représentations apprise ne permet de détecter les anomalies. Tous les modèles prédisent majoritairement la classe `normale`, sans jamais prédire d'anomalies. Ce ne sont cependant pas les détecteurs qui sont à remettre en cause, mais plutôt les représentations apprises. L'apprentissage de représentation de StArDusTS (et de l'apprentissage auto-supervisé en général) repose d'un côté sur l'architecture de réseau de neurones utilisée et de l'autre sur la tâche prétexte choisie pour apprendre cette représentation. Ces deux éléments ne peuvent d'ailleurs pas être dissociés et sont interdépendants. On constate ici que les couples architecture/tâches prétextes explorés lors de cette étude ne permettent pas d'apprendre une représentation utile pour pouvoir détecter des anomalies dans le ce cas applicatif. Nous avions cependant montré dans le chapitre IV que la combinaison des réseaux de neurones CNN 1D avec une tâche prétexte de prédiction permettait de détecter des

anomalies. C'est donc l'ensemble complet du jeu de données, de la tâche prétexte et de l'architecture qui doivent être considérés lorsque l'on parle de la performance de la représentation apprise. L'utilisation d'une quantité plus importante de données est d'ailleurs une des pistes qui pourrait permettre d'améliorer les résultats obtenus.

Concernant l'apprentissage contrastif en particulier, les méthodes d'augmentation de données peuvent être remises en question. Est-ce que ces méthodes permettent de tirer plus d'informations des données ? En effet, l'hypothèse originale de l'apprentissage autosupervisé était que les données présentaient intrinsèquement des informations à propos d'elles-mêmes qui peuvent être utilisées comme des annotations pour un apprentissage supervisé. Est-ce que les méthodes d'augmentation de données représentent des transformations physiques que pourraient subir les données ? Est-ce qu'elles ne modifient pas la sémantique des données que l'on est en train d'observer ? Ces questions sont prises en compte lors de la phase d'augmentation de données utilisées en amont des entrainements. En effet, dans ce manuscrit, la seule méthode d'augmentation de données utilisées consiste à découper les fenêtres les plus longues en fenêtres plus petites grâce à un fenêtrage glissant. Néanmoins, ces considérations ne sont pas prises en compte [PCS22] pour les méthodes d'augmentation de données lorsqu'elles sont utilisées dans le cadre de l'apprentissage contrastif. Ce sont même des méthodes d'augmentation de données avec peu de sens physique, comme le retournement gauche droite ou l'inversion aléatoire des modalités, qui permettent d'obtenir les meilleurs résultats. Le développement de méthode d'augmentation de données spécifique aux données considérées est une piste d'amélioration de l'apprentissage contrastif en particulier pour les séries temporelles.

Concernant l'architecture de réseaux de neurones utilisée, l'optimisation de cette dernière pourrait aussi être un point clé de l'amélioration des résultats du modèle. L'utilisation d'architectures convolutionnelles à une dimension beaucoup plus profonde par exemple, bien que ces dernières puissent apporter d'autres problèmes notamment lors de l'entrainement du modèle. Par ailleurs, certaines architectures neuronales, plus lourdes, avaient été écartées car elles ne correspondaient pas aux critères d'embarquabilité que nous nous étions fixés. Cependant l'utilisation d'architecture comme par exemple les réseaux récurrents ou les transformers sont des pistes d'améliorations à considérer pour le bloc d'apprentissage de représentation.

Nous cherchions à comprendre, grâce à l'utilisation d'un second jeu de données, dans quelle mesure les résultats obtenus avec StArDusTS sur les données cellulaires pouvaient être généralisés à d'autres jeux de données. Nous pouvons donc affirmer que, non, le modèle StArDusTS ne se transfère pas facilement à d'autres données et nécessite d'autres ajustements afin de pouvoir peut-être fonctionner. Les résultats vont plutôt dans le sens de la remise en question de la partie apprentissage de représentation du modèle. Afin d'améliorer la représentation plusieurs pistes pourraient être explorées. Par exemple, l'utilisation d'architectures neuronales plus lourdes comme les réseaux récurrents ou les transformers, ou encore, le développement de méthodes d'augmentation de données propres aux données étudiées et qui conservent la réalité physique des signaux observés.

CHAPITRE

VI

Conclusion et perspectives

Table des matières

A Résumé des travaux

A.1 Contexte

Avec l'augmentation croissante du nombre de capteurs intelligents, nous sommes désormais capables de collecter d'énormes quantités de données en temps réel. Ces données prennent souvent la forme de séries temporelles, qui sont des enregistrements chronologiques d'observations, prises à des intervalles réguliers. Cependant, exploiter pleinement le potentiel de ces données nécessite des méthodes de traitement basées sur l'IA.

Le volume massif de données générées par les capteurs intelligents rend l'analyse manuelle inefficace, fastidieuse et souvent impossible compte tenu de l'échelle des données. Les méthodes plus traditionnelles de traitement de signal peuvent quant à elles s'avérer inefficaces au regard de la complexité des données observées. L'IA offre la possibilité d'automatiser le traitement de ces données, permettant ainsi d'obtenir rapidement des informations exploitables.

Dans cette thèse, nous nous sommes volontairement orientés vers l'analyse de séries temporelles. Aussi, les séries temporelles peuvent être extrêmement complexes, avec des modèles de variations temporelles difficiles à identifier à l'œil nu. Les méthodes d'IA, en particulier l'apprentissage automatique, sont capables de reconnaître des schémas cachés, de détecter des tendances et de faire des prédictions basées sur les données déjà observées. Dans ce contexte, la détection d'anomalies revêt une importance cruciale. Dans une masse de données en constante évolution, il est essentiel de pouvoir identifier rapidement les anomalies qui pourraient signaler des problèmes ou des défaillances potentielles sur les séries temporelles analysées.

Dans cette thèse, nous nous sommes proposés de travailler dans un cadre spécifique de l'IA et de la détection d'anomalies qui est l'apprentissage auto-supervisé. En effet, en particulier dans le contexte de données acquises en temps réel, sur le terrain, les annotations relatives aux données ne sont pas disponibles. Se pencher sur le développement de méthodes prenant en compte cette absence d'annotations est crucial pour développer des modèles performants, capables de détecter des anomalies au sein des données.

A.2 Contributions

La contribution principale de ce travail de thèse et du manuscrit est le modèle StArDusTS (pour Self-supervised Anomaly Detection on Time Series). Ce modèle est une méthodologie qui n'avait encore jamais été expérimentée pour faire de la détection d'anomalies, dans les séries temporelles, sans aucune annotation. Il est constitué de deux blocs principaux détaillés dans le chapitre III section B

— Un premier bloc est utilisé pour apprendre une représentation des séries temporelles traitées. Comme nous nous sommes imposés comme contrainte de ne pas utiliser d'annotations, nous avons choisi, en accord avec l'état de l'art, de nous tourner vers un apprentissage auto-supervisé. L'apprentissage auto-supervisé consiste à proposer des pseudo-labels, qui peuvent être extraits automatiquement des données. Ces derniers sont couplés à une tâche prétexte, c'est-à-dire, une tâche qui n'est pas la tâche d'intérêt, mais qui permet d'entrainer un réseau de neurones et d'apprendre un représentation utile des données. Dans ce manuscrit, deux tâches prétextes sont étudiées, la prédiction du futur de la série temporelle et l'apprentissage contrastif.

— Le second bloc est un bloc de détection d'anomalies. Ce bloc est utilisé sur la représentation apprise grâce au premier bloc. Nous utilisons différents détecteurs d'anomalies non supervisés de l'état de l'art et les comparons.

L'intérêt principal de StArDusTS réside dans sa capacité à apprendre une représentation des données sans avoir besoin d'annotations. L'utilisation de réseaux de neurones permet au modèle de saisir des relations complexes au sein des données, de les extraire et d'en tirer parti pour ensuite détecter les anomalies du jeu de données.

L'utilisation d'une architecture CNN 1D permet de minimiser le nombre de paramètres des modèles et d'avoir des modèles relativement légers avec environ un million de paramètres.

Sa grande modularité, entre le choix de la tâche prétexte, de l'architecture de réseau de neurones et le choix de l'algorithme de détection d'anomalies utilisé sont pour le modèle une grande force lui permettant de s'adapter à plusieurs cas applicatifs. Néanmoins, cette grande diversité possible en fait aussi une faiblesse du modèle, car il n'est pas possible, hormis par une méthode empirique, de savoir si un modèle StArDusTS complet va être efficace sur un jeu de données défini. En effet, nous avons montré qu'en fonction des données traitées, des adaptations du modèle étaient nécessaires.

Afin de pouvoir comparer les représentations, les détecteurs d'anomalies et leurs combinaisons, un outil de comparaison de modèles (voir chapitre III et section D) a été développé au cours de cette thèse. Cet outil repose sur une méthode d'optimisation successive des hyperparamètres exposée chapitre III section C.

A.3 Applications

Nous appliquons le modèle proposé à deux jeux de données distincts. La complémentarité de ces deux jeux de données est discutée section B.2.

Données cellulaires La problématique liée aux données cellulaires est la détection de cellules anormales à partir de leur série temporelle de masse sèche. Nous avons utilisé le modèle StArDusTS proposé avec pour tâche prétexte la prédiction du futur de la masse sèche. Le but de cette section était donc double : du point de vue méthodologique, prouver le fonctionnement du modèle proposé ; et du point de vue applicatif, détecter automatiquement des anomalies dans les données cellulaires, pour que les experts du domaine puissent analyser les comportements anormaux.

Dans cet objectif, un premier détecteur, basé uniquement sur la valeur d'EQM entre la prédiction et la vérité terrain est utilisé (chapitre IV section C.1). Cette détection nous a permis, avec l'aide de spécialistes du domaine, d'identifier automatiquement deux types majeurs d'anomalies dans les données : les anomalies biologiques, pour lesquelles les cellules ont un comportement anormal et les anomalies d'acquisition, qui correspondent à une anomalie dans la chaine de traitement de données en amont de notre modèle.

Dans un second temps, un détecteur permettant d'agréger les résultats individuels obtenus au niveau de chaque fenêtre d'une même cellule nous a permis de détecter des anomalies avec une précision de 96.2% (voir chapitre IV sections C.2 et C.3). L'analyse biologique réalisée par un expert des cellules détectées automatiquement comme anormales a permis d'identifier 6 sous-catégories d'anomalie cellulaire. D'un point de vue méthodologique, nous avons montré dans ces expériences que le modèle StArDusTS entrainé est assez général pour pouvoir détecter des anomalies de cellules dans un jeu de données contenant des cellules différentes de celles utilisées pendant l'entrainement. Ces capacités de généralisation sont l'un des grands points forts du modèle.

Les limitations de cette tâche prétexte de prédiction du futur de la série temporelle sont présentées dans les expériences chapitre IV section C.4. En effet, cette dernière ne permet pas de séparer dans l'espace latent les cellules anormales des cellules normales.

Enfin, une étude comparative d'architectures de réseaux de neurones pour la tâche prétexte de prédiction est exposée chapitre IV section D. Elle montre que les meilleurs réseaux de neurones pour la tâche prétexte de prédiction sont en fait les perceptrons multicouches qui obtiennent, de peu, les meilleures performances (au sens de l'EQM) des modèles testés.

Données sismiques de Mars Comme pour les expériences cellulaires, les expériences menées sur les données sismiques de Mars avaient un double objectif. Nous cherchions dans un premier temps à confronter le modèle StArDusTS à un nouveau jeu de données. Le modèle est-il assez générique pour pouvoir être adapté, tel quel, à un autre jeu de données ? Sur le plan applicatif, la détection automatique des différents événements identifiés dans le signal que sont les glitchs, les chutes de pression et les événements sismiques, particulièrement dans le fort bruit des jours martiens, est d'une importance cruciale.

Pour répondre aux deux questions simultanément, nous cherchons dans un premier temps à prédire au mieux possible le futur des données martiennes. La représentation apprise grâce à la tâche prétexte de prédiction pourra ensuite être utilisée couplée à un détecteur d'anomalies. Différentes valeurs d'hyperparamètres pour les architectures de CNN 1D ainsi que différentes combinaisons des données et plusieurs méthodes de prétraitements sur les données (chapitre V section C.1.1) sont explorées afin de trouver la meilleure façon de prédire ces données. Comparativement, c'est un réseau CNN 1D avec 9 couches convolutionnelles et un décodeur dense qui obtient les meilleures performances prédictives. Les données finalement sélectionnées grâce à cette étude sont les données sismiques U, V, W filtrées à $5\,\mathrm{Hz}$, couplées à la modalité de pression P filtrées à $1\,\mathrm{Hz}$ ainsi qu'un vecteur Jour/Nuit DN soit les données $[U, V, W, P, DN]$. Cependant, les réseaux de neurones ne permettent pas de *bien* prédire le futur des séries temporelles.

C'est pourquoi nous proposons d'explorer les méthodes d'apprentissage de représentation grâce à une seconde tâche prétexte, l'apprentissage contrastif (chapitre V section C.1.2). Neuf méthodes d'augmentation de données sont comparées afin de pouvoir choisir la meilleure. Cependant, la plupart d'entre elles ne permettent pas une phase d'apprentissage correcte des réseaux de neurones et restent bloquées à un palier d'apprentissage. Une étude complémentaire nous permet d'affirmer que, comme pour la tâche prétexte de prédiction du futur de la série temporelle dans l'étude sur les données cellulaires, les différentes classes du jeu de données de Mars ne sont pas séparées dans l'espace latent, même avec la tâche prétexte d'apprentissage contrastif. Cependant, nous avions quand même réussi à détecter des anomalies avec StArDusTS avec cette constatation sur l'espace latent.

C'est pourquoi, les représentations apprises grâce à ces deux tâches prétextes sont comparées avec 4 détecteurs d'anomalies non supervisés (chapitre V section C.2) : à seuil, OCSVM, LOF et Isolation Forest. Aucun d'entre eux ne permet de détecter les anomalies dans les données sismiques de Mars. Nous avons montré dans cette étude qu'afin d'améliorer les résultats de détection, la piste principale à suivre est celle de l'amélioration de la phase d'apprentissage de représentation du modèle.

Cette étude sur le jeu de données de Mars met en avant les faiblesses du modèle StArDusTS qui ne peut pas être transféré facilement à des jeux de données plus complexes.

B Discussions et perspectives

Nous concluons ce document en abordant les sujets qui nous semblent importants au regard des expériences menées au cours de cette thèse, mais qui n'ont pas pu être abordées au cours de ce manuscrit.

B.1 Un travail au sein d'un laboratoire

Le travail présenté dans ce manuscrit s'inscrit dans la thématique de recherche du laboratoire d'accueil nommé le LIIM pour Laboratoire d'Intelligence Intégrée Multicapteurs. Son but est de rapprocher l'intelligence et les traitements des données au plus proche des capteurs pour plusieurs raisons :

— Afin de diminuer la latence des traitements. Pour pouvoir obtenir des traitements toujours plus rapides, notamment dans des contextes fortement contraints comme la voiture autonome, dans lequel des calculs toujours plus rapides et des temps de latence toujours plus faibles sont exigés pour des raisons de sécurité. Il est donc crucial de réfléchir à des modèles, légers, pouvant être portés sur des plateformes embarquées. Bien que cette problématique n'ait pas été directement traitée, elle est restée en ligne de fond de cette thèse et a orienté certains choix techniques comme celui de l'utilisation d'architectures CNN 1D. Le développement de méthodes de détection d'anomalies et l'implémentation du modèle StArDusTS sur une cible embarquée est en cours de réflexion au sein du laboratoire. Les principaux développements devraient aujourd'hui être la recherche d'une application concrète pour la détection d'anomalies dans les séries temporelles. L'utilisation de modèle quantifié est par exemple une piste déjà étudiée au sein du laboratoire pour accélérer les temps d'inférence et diminuer l'empreinte mémoire des modèles utilisés. L'application de ces méthodes pourrait permettre d'embarquer le modèle StArDusTS.

— Pour se rapprocher d'un modèle de capteur intelligent totalement autonome. En particulier la thématique de l'apprentissage continu ou encore l'apprentissage tout au long de la vie. Ces deux champs de recherche considèrent de nombreux scenarii d'utilisation de capteurs intelligents ou l'ensemble des données qui pourront être observée ne sont pas connus lors de l'entrainement du modèle. Le développement de modèles permettant de faire face à l'oubli catastrophique [MC89, MSPT21] est alors primordial. Des solutions pour répondre à cette problématique sont développées au sein du laboratoire [Sol21, MSR+21, MMGR22], cependant ces modèles reposent sur une hypothèse forte. Ils nécessitent en effet de savoir qu'une nouvelle classe est en train d'être observée. La détection de nouveauté, qui est un des autres noms de la détection d'anomalies, devient alors une tâche indispensable pour détecter des observations qui n'ont pas encore été apprises. Pour pousser plus loin le modèle StArDusTS, le développement d'un modèle d'apprentissage continu, incluant un détecteur d'anomalies auto-supervisé pour choisir automatiquement les nouveaux exemples des nouvelles classes à apprendre est un sujet de recherche ambitieux. Le problème actuel auquel nous serions confrontés est l'apprentissage de représentation qui devrait être réalisé tout au long de la vie du modèle, sans oublier les classes précédentes apprises, ainsi que la mise à jour au fur et à mesure des classes *connues* par l'algorithme de détection d'anomalies afin de ne plus les détecter comme anormales une fois que ces dernières ont été apprises.

— Liée à ces deux thématiques, la question de la confiance que l'on peut apporter aux résultats proposés par les réseaux de neurones est une problématique ouverte au sein du laboratoire. En effet, la faible explicabilité des réseaux de neurones, dont le fonctionnement est souvent celui d'une boite noire, remet en question l'utilisation de l'IA pour des domaines d'application critiques comme la santé par exemple. Dans quelle mesure peut-on faire confiance au résultat produit par une IA si on ne peut pas l'expliquer ? Cette thèse s'inscrit dans cette problématique grâce à son aspect détection d'anomalies. Si une anomalie est détectée (au niveau de la donnée, aussi bien qu'au niveau de la classification), la confiance que l'on peut mettre dans la classification d'une IA est plus faible que dans un cas où les données et la classification semblent

normales. Ces travaux ne sont cependant qu'une manière préliminaire d'approcher la problématique d'explicabilité des modèles d'IA afin d'avertir en cas de comportement déviant du modèle ou des données. Cette thématique est aujourd'hui le sujet d'une thèse au sein du laboratoire.

B.2 Jeux de données

Volonté de travailler avec des jeux de données réels

Dans ce manuscrit, aucune expérience méthodologique ne s'est appuyée sur des données simulées. Ce choix présente plusieurs avantages importants :

Réalisme : Les données du monde réel reflètent la complexité et la variabilité de situations réelles. Elles sont générées par des processus naturels ou humains et incluent des facteurs imprévisibles, des bruits et des erreurs. Tester des modèles sur de telles données permet d'évaluer leurs capacités à gérer la réalité et à généraliser à des situations inconnues, ce qui est particulièrement pertinent dans un contexte de détection d'anomalies ou de détection de nouveautées. Les données du monde réel offrent une variété d'exemples pertinents qui aident le modèle à développer une meilleure capacité de généralisation.

En effet, l'intelligence artificielle est une science des données qui ne saurait exister sans lesdites données. Autrement dit, le développement de méthodes à base d'intelligence artificielle n'a de sens que si aucune autre méthode plus simple ne permet de résoudre les problèmes posés. C'est donc une grande chance que d'avoir pu travailler avec des données réelles, à résoudre des problématiques du monde réel. Le développement des méthodes en étroite liaison avec les experts des données a permis d'ancrer les problématiques méthodologiques abordées dans cette thèse dans la réalité des données. Ces développements prenant en considération les questions méthodologiques aussi bien qu'applicatives conjointement ont alors aussi bien permis d'ouvrir la voie à des avancées significatives pour le domaine de l'intelligence artificielle que pour les domaines spécifiques des données.

Biais de simulation : Les données simulées peuvent contenir des biais, des simplifications ou des hypothèses qui ne reflètent pas fidèlement la réalité. Ces biais de simulation peuvent affecter la performance du modèle lorsqu'il est utilisé dans des situations réelles, car il n'aura pas été exposé à une variété suffisante de situations. De plus, les réseaux de neurones sont particulièrement efficaces pour trouver des solutions triviales au sein des données, voir Remarque B.2. Attention cependant, comme le montre cette remarque, des biais peuvent aussi être présents dans les données réelles, néanmoins, le fait de travailler avec des données simulées peut rajouter une quantité supplémentaire de biais, propres à la simulation.

Remarque 12 : Détection automatique de tank ?

Pour illustrer la notion de biais dans les données, une étude en particulier est souvent utilisée comme illustration. Dans cette étude, les chercheurs ont collecté de nombreuses images de tanks et de paysages pour entrainer le réseau de neurones à reconnaître les différences entre les deux.

Après avoir investi beaucoup de temps et de ressources dans le projet, le réseau de neurones aurait été prêt pour les tests sur le terrain. Cependant, lors des essais, le système s'est révélé être un échec complet. Il a été incapable de détecter les tanks dans un environnement réel.

Après enquête, les chercheurs auraient découvert la raison du problème : toutes les images de tanks utilisées pour l'entrainement avaient été prises en intérieur, avec une faible luminosité, tandis que les images sans tanks étaient prises en extérieur. En réalité, le réseau de neurones avait simplement appris à distinguer les différences de luminosité entre les images en extérieur et en intérieur plutôt que de reconnaître les tanks eux-mêmes.

Ces expériences mettent en lumière un problème réel dans l'apprentissage automatique : celui de la distribution des données d'entrainement. Si les données utilisées pour entrainer un modèle ne représentent pas fidèlement les scénarios réels auxquels le modèle sera confronté, il peut produire des résultats inattendus et non désirés. C'est pourquoi il est crucial d'avoir des ensembles de données de qualité et représentatifs.

Néanmoins, [Bra11] montre que cette histoire n'a jamais eu lieu, et qu'il s'agit en réalité d'une légende urbaine. Pire, il affirme pour plusieurs raisons que cette histoire n'aurait pas pu avoir lieu. Les chercheurs et professeurs continuent cependant à la raconter pour son côté didactique et sa morale, ce manuscrit ne faisant pas exception à la règle.

Validation empirique : Tester un modèle sur des données du monde réel fournit une validation empirique de son efficacité. Cela permet d'évaluer objectivement les performances et les capacités du modèle dans des scénarios pratiques. Tester ces modèles sur des données réelles est essentiel pour garantir qu'ils répondent efficacement aux besoins des utilisateurs finaux.

C'est l'utilisation de données réelles qui a motivé au cours de cette thèse le développement de méthodes et de modèles ne nécessitant pas d'annotations et en particulier de l'apprentissage auto-supervisé. Néanmoins, l'utilisation de données réelles vient aussi avec des contraintes dont nous avons conscience et auxquelles nous avons été confrontés au cours de cette thèse :

Maitrise du jeu de données : Lorsque l'on utilise des données réelles, il peut être difficile de contrôler tous les aspects de l'environnement, contrairement aux données simulées. Cela peut rendre plus complexe l'évaluation et la comparaison des modèles. En particulier les données réelles peuvent être sujettes à des biais qui peuvent affecter la performance des modèles. Ces biais sont alors difficiles à identifier tant ils sont spécifiques aux données observées. De plus, contrairement aux données simulées, ils ne peuvent pas être contrôlés.

Données sans annotation : L'un des points clés d'un projet de recherche méthodologique en IA reste l'évaluation et la comparaison des différents modèles. Cependant, l'utilisation de jeux de données réels ne permet pas toujours d'avoir à disposition des annotations associées aux données. Ces annotations sont pourtant cruciales pour pouvoir évaluer les modèles. Il est donc nécessaire de trouver des alternatives. Dans notre cas, nous avons choisi d'évaluer les modèles sur leur tâche prétexte, mais des

alternatives sont formalisées dans le domaine de l'open set recognition. Elles sont discutées section B.5. Par ailleurs, ces données peuvent ne pas encore être bien comprises. Elles peuvent mener à des étiquetages qui sont ambigus comme ceux auxquels nous avons été confrontés dans le jeu de données cellulaires (c'est aussi l'un des points qui rend l'utilisation de tels jeux de données intéressante).

Éthique : Bien que nous n'y soyons pas confrontés dans ce manuscrit, l'utilisation de données réelles peut soulever des questions éthiques, en particulier pour la protection de la vie privée des personnes. Ces sujets émergent doucement au sein de la communauté, la plupart des conférences et des journaux demandent aux auteurs de se pencher sur la question et la formation à l'éthique de la recherche est aujourd'hui obligatoire pour tous les doctorants.

Complémentarité des jeux de données

Le choix des jeux de données utilisés pendant cette thèse repose d'un côté sur des convictions personnelles, de vouloir travailler avec des données issues de milieux scientifiques liés au domaine de la santé ou de l'environnement. Ces jeux de données n'ont cependant pas étés choisis au hasard, voici les points sur lesquels ils se complètent :

Taille des données Les données cellulaires sont composées d'une dizaine de milliers de cellules d'une durée de vie assez courte de l'ordre de 80 heures au maximum. Les données sismiques de Mars, elles, sont composées d'une seule série temporelle qui s'étend sur une durée d'un mois, quasiment sans aucune interruption.

Nombre de modalités Les données cellulaires contiennent une unique modalité qui est la masse sèche des cellules alors que les données sismiques de Mars contiennent les 3 modalités d'accélérométrie, 1 de pression et une de jour/nuit qui encode deux rythmes différents. L'utilisation de plus de modalités sous-entend que plus d'information est disponible pour le réseau de neurones, mais il peut alors être plus difficile à entrainer.

Différents types d'anomalies Les deux jeux de données se rejoignent sur ce point. Parmi les anomalies à détecter, plusieurs sous classes d'anomalies peuvent être distinguées. Que l'on parle des anomalies cellulaires face aux anomalies d'acquisition pour les données cellulaires ou des glitchs face aux chutes de pression et aux événements sismiques pour les données martiennes.

Embarquabilité de l'algorithme développé Les données cellulaires sont acquises en laboratoire, elles ont donc une contrainte de stockage et de puissance de calcul assez faible. A contrario, la distance entre la terre et le robot posé sur la surface de Mars illustre bien la nécessité de développer des algorithmes de taille réduite pour pouvoir être exécutés, même sur une autre planète, mais aussi assez robustes pour pouvoir fonctionner en totale autonomie.

Disponibilité des annotations Un point clé qui oppose les deux jeux de données est qu'aucune anomalie n'était connue dans le jeu de données cellulaires avant de réaliser les expériences du chapitre IV et de présenter les anomalies détectées à un expert pour qu'il les analyse. Néanmoins, l'annotation *a posteriori* des séries temporelles anormales étaient particulièrement fastidieuse, c'est pourquoi nous nous sommes tournés vers un second jeu de données pour lequel des annotations étaient déjà disponibles. Ces derniers nous permettent donc de valider la pertinence des modèles entrainés.

B.3 L'IA et le traitement de séries temporelles

Au centre des thématiques de recherche sur l'IA, on retrouve aujourd'hui deux thématiques principales :

— Le TALN conçus pour analyser, interpréter et générer du texte, leur permettant de *comprendre* les intentions des utilisateurs, de répondre à des questions, de traduire des langues et d'effectuer des tâches linguistiques complexes.
— La vision par ordinateur discipline de l'IA axée sur l'analyse, l'interprétation et la compréhension du contenu visuel. Les systèmes de traitement d'image répondent à des problématiques techniques telles que la reconnaissance d'objets, la segmentation, la détection et la classification d'éléments dans les images. Ils permettent aux machines de "voir" et de comprendre les informations visuelles, ouvrant ainsi la voie à des applications variées comme l'interprétation sémantique d'images, la conduite autonome ou la surveillance.

Bien que les séries temporelles soient très intéressantes à étudier, comme cela a été prouvé dans ce manuscrit, on remarque que ces dernières sont beaucoup moins médiatisées, à la fois auprès du grand public, mais aussi au sein de la communauté de l'apprentissage profond. On peut expliquer cet engouement moins fort autour du traitement de séries temporelles grâce à plusieurs axes :

— Les séries temporelles sont souvent plus complexes à traiter que d'autres types de données, car elles dépendent du temps et peuvent présenter des motifs difficiles à interpréter. Les modèles et algorithmes utilisés pour les séries temporelles sont généralement plus spécialisés et moins accessibles au grand public que ceux utilisés dans le TALN ou le traitement d'image. Elles sont par ailleurs développées dans des domaines très spécialisés.
— Contrairement à certaines applications de NLP ou de traitement d'image qui peuvent produire des résultats visuellement impressionnants ou susciter des réactions émotionnelles, les analyses de séries temporelles et les prédictions peuvent sembler moins spectaculaires aux yeux du grand public et/ou des investisseurs.
— Les domaines applicatifs liés aux séries temporelles sont beaucoup plus spécialisés que ce qui peut être observé en traitement d'image. La spécialisation de ces données ne facilite pas la généralisation de résultats obtenus dans un domaine à un autre domaine. C'est d'ailleurs un résultat observé au cours de ce manuscrit. Ainsi, l'utilisation d'un nouveau jeu de données peut aussi être synonyme du développement de nouveaux modèles.
— Cette diversité des domaines applicatifs provoque un effet de bord indésiré qui est qu'il n'existe pas d'introduction facile au traitement de séries temporelles à la manière du jeu de données MNIST [Den12] pour le traitement d'image, qui permet de prendre en main les méthodes de l'état de l'art pour des ingénieurs ou des chercheurs qui voudraient utiliser des modèles d'apprentissage automatique.

Malgré ces facteurs, il est important de souligner que les séries temporelles continuent de jouer un rôle crucial dans de nombreux domaines et qu'elles sont étudiées activement par la communauté scientifique et l'industrie. Les progrès dans la modélisation et la prédiction des séries temporelles ont des implications significatives pour les entreprises et la prise de décision dans de nombreux secteurs.

B.4 Pour aller plus loin

Les résultats obtenus, notamment sur le jeu de données sismiques de Mars ne sont pas satisfaisants du point de vue de la tâche de détection d'anomalies. Nous proposons ici quelques pistes d'amélioration du modèle qui pourraient permettre d'améliorer les résultats obtenus.

D'autres architectures Nous nous sommes restreints à l'utilisation d'architectures principalement convolutionnelles à une dimension. Ces architectures présentaient pour nous l'avantage notable d'être assez légères pour pouvoir, un jour être déployées sur des cibles embarquées. Néanmoins, si la métrique d'intérêt n'est plus uniquement la portabilité de l'architecture mais uniquement sa performance, d'autres architectures, plus lourdes, peuvent être envisagées pour apprendre une représentation des données.

En particulier, les architectures reposant sur des mécanismes auto-attentifs, aussi appelés transformers [VSP+17] n'ont pas été utilisées dans ce manuscrit, mais sont une piste prometteuse pour le traitement de série temporelle lorsque les ressources de calcul ne sont pas limitées.

Utilisation des annotations Nous avons choisi de développer des méthodes ne nécessitant pas d'annotations pour pouvoir être utilisées. Néanmoins, au cours des études menées dans ce manuscrit, nous nous sommes rendus compte que ces annotations sont nécessaires afin de valider les modèles proposés d'un point de vue méthodologique.

Cependant, l'état de l'art montre que les méthodes d'IA n'atteignent jamais les performances obtenues grâce aux annotations. Ainsi, si le but d'une expérience n'est pas, comme nous l'avons fait, de valider l'utilisation d'un modèle, mais de maximiser les performances de ce dernier, utiliser des annotations dans un apprentissage supervisé, ou du moins en partie supervisé, semble la piste à suivre.

C'est par exemple la piste choisie par [PCS22] qui utilise l'apprentissage auto-supervisé afin d'apprendre une représentation des données dont ils disposent, mais entrainent un classificateur de manière supervisée à partir de la représentation apprise. Ce type d'apprentissage pourrait alors être qualifié de semi-supervisé (bien que ce terme soit utilisé pour décrire de nombreux types d'expériences).

L'utilisation de l'apprentissage auto-supervisé comme un proxy pour apprendre une représentation des données et améliorer les performances d'un classificateur supervisé est une piste intéressante pour améliorer les résultats de détection d'anomalies dans les données sismiques de Mars.

B.5 Comment évaluer une représentation ?

L'apprentissage en environnement ouvert, également connu sous le nom d'open set recognition en anglais, est un problème dans le domaine de la vision par ordinateur et plus largement dans les applications de traitement par IA sur systèmes embarqués. Il s'agit d'une extension du problème de classification traditionnel, où les modèles d'apprentissage sont généralement formés pour classer les données en un certain nombre de catégories prédéfinies.

Dans un problème de reconnaissance de classe standard, le modèle est entrainé à prédire une classe pour une observation parmi un nombre fini de classes connues. Cependant, en pratique, il est courant de rencontrer des données qui ne correspondent à aucune des classes apprises lors de l'entrainement, aussi appelées nouveautés. Le problème de l'apprentissage en environnement ouvert consiste donc à développer des modèles capables

de reconnaître les exemples des classes connues tout en étant capables de détecter et de rejeter les exemples des classes inconnues.

L'apprentissage en environnement ouvert présente des défis importants, notamment la détection de nouveautés, qui doit pouvoir être réalisée grâce à des méthodes ne nécessitant pas d'annotation. C'est la formalisation des méthodes de validation pour la détection d'anomalies sans annotations [Goi16, MCZS15] utilisées dans ce domaine qui peuvent nous intéresser pour répondre à la question suivante : Est-il possible d'évaluer une représentation, sans connaitre les annotations associées pour pouvoir l'évaluer sur la tâche finale que l'on cherche à accomplir ?

En particulier, on distingue d'un côté les méthodes de validation de modèles externes, c'est-à-dire qui utilisent des informations provenant d'un expert, d'une mesure annexe ou d'un modèle physique, pour quantifier les performances d'un modèle. Nous ne pouvons pas utiliser ces dernières, car nous n'avons pas à disposition d'annotations. De l'autre, les méthodes internes permettent de comparer les modèles sans pour autant nécessiter de disposer des annotations. Elles se basent sur des interprétations analytiques de la segmentation entre les données normales et anormales. Ces méthodes internes peuvent encore être décomposées en 3 sous-catégories :

Ensemblistes : elles tentent de trouver le modèle qui donne le meilleur consensus parmi un grand ensemble de modèles entrainés [ZCS14, RA15, LSL17, LTFO20, Kle99, DMS+20],

Géométriques : elles évaluent la qualité du modèle de détection d'anomalies en analysant la séparabilité des anomalies dans l'espace d'entrée des données [MCZS15, NNN16, MCSZ20, BMHP21, Goi22],

Post-tâches : elles évaluent la cohérence d'une détection apprise de manière non supervisée en reproduisant la même expérience avec un apprentissage supervisé. Elles doivent alors obtenir le même résultat pour être performantes [NNN+15, KKSZ11, DNN16].

L'utilisation de telles méthodes d'évaluation des représentations couplée au bloc d'apprentissage de représentation de StArDusTS pourrait s'avérer bénéfique pour la représentation apprise en mesurant quantitativement sa qualité et/ou en comparant différents groupes d'hyperparamètres de modèles StArDusTS et donc pourrait permettre d'augmenter les performances de détection d'anomalies des modèles.

Des développements méthodologiques, mais pas que... Nous avons proposé dans cette section deux axes de développement méthodologique pour améliorer le modèle existant et ainsi proposer une méthode toujours plus robuste, capable de *mieux* détecter les anomalies d'un jeu de données.

Néanmoins, le modèle StArDusTS a été développé pour détecter des anomalies et répondre à des problématiques applicatives. La *simple* utilisation du modèle est une des pistes qui serait très intéressante. En particulier, des discussions ont émergé avec les experts des données biologiques concernant leur besoin de mieux comprendre des lignées cellulaires données. L'utilisation, telle quelle, du modèle dont nous avons démontré le fonctionnement, pour répondre à leurs problématiques applicatives nécessiterait cependant le conseil d'un expert en science des données capable d'aiguiller les biologistes et comme lors de cette thèse créer du savoir ensemble.

En somme, le développement de nouvelles relations afin de mettre en avant le modèle proposé pour pouvoir explorer de nouvelles thématiques de recherche, hors du domaine de l'IA, auxquelles StArDusTS peut répondre est un des axes de poursuite des recherches menées dans cette thèse.

Bibliographie

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow : Large-scale machine learning on heterogeneous systems, 2015.

[AAK+18] Osama Abdeljaber, Onur Avci, Mustafa Serkan Kiranyaz, Boualem Boashash, Henry Sodano, and Daniel J. Inman. 1-D CNNs for structural damage detection : Verification on a structural health monitoring benchmark data. *Neurocomputing*, 275:1308–1317, January 2018.

[AAKI17] Onur Avci, Osama Abdeljaber, Serkan Kiranyaz, and Daniel Inman. Structural Damage Detection in Real Time : Implementation of 1D Convolutional Neural Networks for SHM Applications. In Christopher Niezrecki, editor, *Structural Health Monitoring & Damage Detection, Volume 7*, Conference Proceedings of the Society for Experimental Mechanics Series, pages 49–54, Cham, 2017. Springer International Publishing.

[AASM21] Omar Alghushairy, Raed Alsini, Terence Soule, and Xiaogang Ma. A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams. *Big Data and Cognitive Computing*, 5(1):1, March 2021.

[ABA+17] Jesse K. Adams, Vivek Boominathan, Benjamin W. Avants, Daniel G. Vercosa, Fan Ye, Richard G. Baraniuk, Jacob T. Robinson, and Ashok Veeraraghavan. Single-frame 3D fluorescence microscopy with ultraminiature lensless FlatScope. *Science Advances*, 3(12):e1701548, December 2017.

[ADL+76] Don L. Anderson, Frederick K. Duennebier, Gary V. Latham, M. Fafi Toksöz, Robert L. Kovach, Tony C. D. Knight, Andrew R. Lazarewicz, Wayne F. Miller, Yosio Nakamura, and George Sutton. The Viking Seismic Experiment. *Science*, 194(4271):1318–1321, December 1976.

[AM21] Haider Al-Tahan and Yalda Mohsenzadeh. CLAR : Contrastive Learning of Auditory Representations. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 2530–2538. PMLR, March 2021.

[AMKF18] Joshua Aslan, Kieren Mayers, Jonathan G. Koomey, and Chris France. Electricity Intensity of Internet Data Transmission : Untangling the Estimates. *Journal of Industrial Ecology*, 22(4):785–798, 2018.

[AMV+17] C. Allier, S. Morel, R. Vincent, L. Ghenim, F. Navarro, M. Menneteau, T. Bordy, L. Hervé, O. Cioni, X. Gidrol, Y. Usson, and J.-M. Dinten. Imaging of dense cell cultures by multiwavelength lens-free video microscopy. *Cytometry Part A*, 91(5):433–442, 2017.

[ANR74] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, C-23(1):90–93, January 1974.

[ARS+20] Jean-Baptiste Alayrac, Adria Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-Supervised MultiModal Versatile Networks. *Advances in Neural Information Processing Systems*, 33:25–37, 2020.

[AS19] Abhijeet Awasthi and Sunita Sarawagi. Continual learning with neural networks : A review. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CODS-COMAD '19, pages 362–365, New York, NY, USA, 2019. Association for Computing Machinery.

[ASP94] P.J. Angeline, G.M. Saunders, and J.B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, January 1994.

[AW10] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010.

[AYW+22] Jesse K. Adams, Dong Yan, Jimin Wu, Vivek Boominathan, Sibo Gao, Alex V. Rodriguez, Soonyoung Kim, Jennifer Carns, Rebecca Richards-Kortum, Caleb Kemere, Ashok Veeraraghavan, and Jacob T. Robinson. In vivo lensless microscopy via a phase mask generating diffraction patterns with high-contrast contours. *Nature Biomedical Engineering*, 6(5):617–628, May 2022.

[Bai20] Romain Bailly. AI processing of time series for smart sensors. Technical report, Grenoble INP Phelma, Rapport de stage, Grenoble INP Phelma, 2020.

[Bal96] William H. Baltosser. Biostatistical Analysis, 3rd ed. *Ecology*, 77(7):2266–2268, October 1996.

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(null):281–305, February 2012.

[BBBK11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[BCE+23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence : Early experiments with GPT-4, April 2023.

[Bel66] Richard Bellman. Dynamic programming. *Science (New York, N.Y.)*, 153(3731):34–37, 1966.

[BGNR17] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing Neural Network Architectures using Reinforcement Learning, March 2017.

[BGRN17] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating Neural Architecture Search using Performance Prediction, November 2017.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, July 1992. Association for Computing Machinery.

[BK97] Apostolos N. Burnetas and Michael N. Katehakis. Optimal Adaptive Policies for Markov Decision Processes. *Mathematics of Operations Research*, 22(1):222–255, February 1997.

[BKK18] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271 [cs]*, April 2018.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF : Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104, May 2000.

[BLB+13] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software : Experiences from the scikit-learn project. In *ECML PKDD Workshop : Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[BLK+21] Salma Barkaoui, Philippe Lognonné, Taichi Kawamura, Éléonore Stutzmann, Léonard Seydoux, Maarten V. de Hoop, Randall Balestriero, John-Robert Scholz, Grégory Sainton, Matthieu Plasman, Savas Ceylan, John Clinton, Aymeric Spiga, Rudolf Widmer-Schnidrig, Francesco Civilini, and W. Bruce Banerdt. Anatomy of Continuous Mars SEIS and Pressure Data from Unsupervised Learning. *Bulletin of the Seismological Society of America*, 111(6):2964–2981, November 2021.

[BM01] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction : Applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 245–250, New York, NY, USA, August 2001. Association for Computing Machinery.

[BMA+21] Romain Bailly, Marielle Malfante, Cédric Allier, Lamya Ghenim, and Jérôme Mars. Deep anomaly detection using self-supervised learning : Application to time series of cellular data. In *ASPAI 2021 - 3rd International Conference on Advances in Signal Processing and Artificial Intelligence*, November 2021.

[BMA+23a] Romain Bailly, Marielle Malfante, Cédric Allier, Lamya Ghenim, and Jérôme Mars. Comparaison des capacités prédictives de réseaux de neurones, application à la masse sèche de cellules. In *XXIXème Colloque Francophone de Traitement Du Signal et Des Images (GRETSI 2023)*, page 1, 2023.

[BMA+23b] Romain Bailly, Marielle Malfante, Cedric Allier, Chiara Paviolo, Lamya Ghenim, Kiran Padmanabhan, Sabine Bardin, and Jérôme Mars. StArDusTS, a self-supervised model for anomaly detection on time series. Application to the detection of abnormal behaviors in cell dry mass time series. *Scientific Reports*, 2023.

[BMHP21] Ali Bagherinia, Behrooz Minaei-Bidgoli, Mehdi Hosseinzadeh, and Hamid Parvin. Reliability-based fuzzy clustering ensemble. *Fuzzy Sets and Systems*, 413:1–28, June 2021.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[Bra11] Gwern Branwen. The Neural Net Tank Urban Legend. https://gwern.net/tank, September 2011.

[BSB+20] W. Bruce Banerdt, Suzanne E. Smrekar, Don Banfield, Domenico Giardini, Matthew Golombek, Catherine L. Johnson, Philippe Lognonné, Aymeric Spiga, Tilman Spohn, Clément Perrin, Simon C. Stähler, Daniele Antonangeli, Sami Asmar, Caroline Beghein, Neil Bowles, Ebru Bozdag, Peter Chi, Ulrich Christensen, John Clinton, Gareth S. Collins, Ingrid Daubar, Véronique Dehant, Mélanie Drilleau, Matthew Fillingim, William Folkner, Raphaël F. Garcia, Jim Garvin, John Grant, Matthias Grott, Jerzy Grygorczuk, Troy Hudson, Jessica C. E. Irving, Günter Kargl, Taichi Kawamura, Sharon Kedar, Scott King, Brigitte Knapmeyer-Endrun, Martin Knapmeyer, Mark Lemmon, Ralph Lorenz, Justin N. Maki, Ludovic Margerin, Scott M. McLennan, Chloe Michaut, David Mimoun, Anna Mittelholz, Antoine Mocquet, Paul Morgan, Nils T. Mueller, Naomi Murdoch, Seiichi Nagihara, Claire Newman, Francis Nimmo, Mark Panning, W. Thomas Pike, Ana-Catalina Plesa, Sébastien Rodriguez, Jose Antonio Rodriguez-Manfredi, Christopher T. Russell, Nicholas Schmerr, Matt Siegler, Sabine Stanley, Eléanore Stutzmann, Nicholas Teanby, Jeroen Tromp, Martin van Driel, Nicholas Warner, Renee Weber, and Mark Wieczorek. Initial results from the InSight mission on Mars. *Nature Geoscience*, 13(3):183–189, March 2020.

[BSN+20] Don Banfield, Aymeric Spiga, Claire Newman, François Forget, Mark Lemmon, Ralph Lorenz, Naomi Murdoch, Daniel Viudez-Moreiras, Jorge Pla-Garcia, Raphaël F. Garcia, Philippe Lognonné, Özgür Karatekin, Clément Perrin, Léo Martire, Nicholas Teanby, Bart Van Hove, Justin N. Maki, Balthasar Kenda, Nils T. Mueller, Sébastien Rodriguez, Taichi Kawamura, John B. McClean, Alexander E. Stott, Constantinos Charalambous, Ehouarn Millour, Catherine L. Johnson, Anna Mittelholz, Anni Määttänen, Stephen R. Lewis, John Clinton, Simon C. Stähler, Savas Ceylan, Domenico Giardini, Tristram Warren, William T. Pike, Ingrid Daubar, Matthew Golombek, Lucie Rolland, Rudolf Widmer-Schnidrig, David Mimoun, Éric Beucler, Alice Jacob, Antoine Lucas, Mariah Baker, Véronique Ansan, Kenneth Hurst, Luis Mora-Sotomayor, Sara Navarro, Josefina Torres, Alain Lepinette, Antonio Molina, Mercedes Marin-Jimenez, Javier

Gomez-Elvira, Veronica Peinado, Jose-Antonio Rodriguez-Manfredi, Brian T. Carcich, Stephen Sackett, Christopher T. Russell, Tilman Spohn, Suzanne E. Smrekar, and W. Bruce Banerdt. The atmosphere of Mars as observed by InSight. *Nature Geoscience*, 13(3):190–198, March 2020.

[BYC13] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search : Hyperparameter optimization in hundreds of dimensions for vision architectures. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, June 2013. PMLR.

[CB21] Mayank Arya Chandra and S. S. Bedi. Survey on SVM and their application in imageclassification. *International Journal of Information Technology*, 13(5):1–11, October 2021.

[CBC+20] Ramona Corman, Willem Boutu, Anna Campalans, Pablo Radicella, Joana Duarte, Maria Kholodtsova, Laure Bally-Cuif, Nicolas Dray, Fabrice Harms, Guillaume Dovillaire, Samuel Bucourt, and Hamed Merdji. Lensless microscopy platform for single cell and tissue visualization. *Biomedical Optics Express*, 11(5):2806–2817, May 2020.

[CCC16] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-Scale Convolutional Neural Networks for Time Series Classification. *arXiv:1603.06995 [cs]*, May 2016.

[CCG+22] Savas Ceylan, John F. Clinton, Domenico Giardini, Simon C. Stähler, Anna Horleston, Taichi Kawamura, Maren Böse, Constantinos Charalambous, Nikolaj L. Dahmen, Martin van Driel, Cecilia Durán, Fabian Euchner, Amir Khan, Doyeon Kim, Matthieu Plasman, John-Robert Scholz, Géraldine Zenhäusern, Eric Beucler, Raphaël F. Garcia, Sharon Kedar, Martin Knapmeyer, Philippe Lognonné, Mark P. Panning, Clément Perrin, William T. Pike, Alexander E. Stott, and William B. Banerdt. The marsquake catalogue from InSight, sols 0–1011. *Physics of the Earth and Planetary Interiors*, 333:106943, December 2022.

[CCv+21] John F. Clinton, Savas Ceylan, Martin van Driel, Domenico Giardini, Simon C. Stähler, Maren Böse, Constantinos Charalambous, Nikolaj L. Dahmen, Anna Horleston, Taichi Kawamura, Amir Khan, Guenolé Orhand-Mainsant, John-Robert Scholz, Fabian Euchner, William B. Banerdt, Philippe Lognonné, Don Banfield, Eric Beucler, Raphaël F. Garcia, Sharon Kedar, Mark P. Panning, Clement Perrin, William T. Pike, Suzanne E. Smrekar, Aymeric Spiga, and Alexander E. Stott. The Marsquake catalogue from InSight, sols 0–478. *Physics of the Earth and Planetary Interiors*, 310:106595, January 2021.

[CGRL20] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification : Applications, challenges and trends. *Neurocomputing*, 408:189–215, September 2020.

[CGS16] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net : Accelerating Learning via Knowledge Transfer, April 2016.

[Che18] Guillaume Chevalier. English : The Long Short-Term Memory (LSTM) cell can process data sequentially and keep its hidden state through time., May 2018.

[Chu92] C. K. Chui. *An Introduction to Wavelets*. Academic Press, January 1992.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, July 2020.

[CLRM23] Paul Compagnon, Aurore Lomet, Marina Reyboz, and Martial Mermillod. Domestic Hot Water Forecasting for Individual Housing with Deep Learning. In Irena Koprinska, Paolo Mignone, Riccardo Guidotti, Szymon Jaroszewicz, Holger Fröning, Francesco Gullo, Pedro M. Ferreira, Damian Roqueiro, Gaia Ceddia, Slawomir Nowaczyk, João Gama, Rita Ribeiro, Ricard Gavaldà, Elio Masciari, Zbigniew Ras, Ettore Ritacco, Francesca Naretto, Andreas Theissler, Przemyslaw Biecek, Wouter Verbeke, Gregor Schiele, Franz Pernkopf, Michaela Blott, Ilaria Bordino, Ivan Luciano Danesi, Giovanni Ponti, Lorenzo Severini, Annalisa Appice, Giuseppina Andresini, Ibéria Medeiros, Guilherme Graça, Lee Cooper, Naghmeh Ghazaleh, Jonas Richiardi, Diego Saldana, Konstantinos Sechidis, Arif Canakoglu, Sara Pido, Pietro Pinoli, Albert Bifet, and Sepideh Pashami, editors, *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Communications in Computer and Information Science, pages 223–235, Cham, 2023. Springer Nature Switzerland.

[CMS+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 213–229, Cham, 2020. Springer International Publishing.

[CNI23] CNIL. Modèle (IA). https://www.cnil.fr/fr/definition/modele-ia, 2023.

[Com94] Pierre Comon. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314, April 1994.

[CSd+14] Nicolas Chenouard, Ihor Smal, Fabrice de Chaumont, Martin Maška, Ivo F. Sbalzarini, Yuanhao Gong, Janick Cardinale, Craig Carthel, Stefano Coraluppi, Mark Winter, Andrew R. Cohen, William J. Godinez, Karl Rohr, Yannis Kalaidzidis, Liang Liang, James Duncan, Hongying Shen, Yingke Xu, Klas E. G. Magnusson, Joakim Jaldén, Helen M. Blau, Perrine Paul-Gilloteaux, Philippe Roudot, Charles Kervrann, François Waharte, Jean-Yves Tinevez, Spencer L. Shorte, Joost Willemse, Katherine Celler, Gilles P. van Wezel, Han-Wei Dan, Yuh-Show Tsai, Carlos Ortiz de Solórzano, Jean-Christophe Olivo-Marin, and Erik Meijering. Objective comparison of particle tracking methods. *Nature Methods*, 11(3):281–289, March 2014.

[CSN+20] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On Empirical Comparisons of Optimizers for Deep Learning, June 2020.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, December 1989.

[CZH19] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS : Direct Neural Architecture Search on Target Task and Hardware, February 2019.

[DBK+19] H. A. Dau, A. Bagnall, K. Kamgar, C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, November 2019.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale, June 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Den12] Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012.

[Des17] Travis Desell. Large scale evolution of convolutional neural networks using volunteer computing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 127–128, New York, NY, USA, July 2017. Association for Computing Machinery.

[DGE16] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised Visual Representation Learning by Context Prediction. *arXiv:1505.05192 [cs]*, January 2016.

[DKLM05] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics*. Springer Texts in Statistics. Springer, London, 2005.

[DLYH19] S. Du, T. Li, Y. Yang, and S. Horng. Deep Air Quality Forecasting Using Hybrid Deep Learning Framework. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019.

[DMS+20] Sunny Duan, Loic Matthey, Andre Saraiva, Nicholas Watters, Christopher P. Burgess, Alexander Lerchner, and Irina Higgins. Unsupervised Model Selection for Variational Disentangled Representation Learning, February 2020.

[DNN16] Phai Vu Dinh, Thanh Nguyen Nguyen, and Quang Uy Nguyen. An empirical study of anomaly detection in online games. In *2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, pages 171–176, September 2016.

[DSH15] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, June 2015.

[Duc18] David Ducros. Accueil - SEIS / Mars InSight. https://www.seis-insight.eu/fr/, 2018.

[DV18] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, January 2018.

[É23] Le Robert Équipe éditoriale. anomalie - Définitions, synonymes, conjugaison, exemples | Dico en ligne Le Robert. https://dictionnaire.lerobert.com/definition/anomalie, 2023.

[EIK19] Levent Eren, Turker Ince, and Serkan Kiranyaz. A Generic Intelligent Bearing Fault Diagnosis System Using Compact Adaptive 1D CNN Classifier. *Journal of Signal Processing Systems*, 91(2):179–189, February 2019.

[EMH17] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple And Efficient Architecture Search for Convolutional Neural Networks, November 2017.

[EMH19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search : A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[FD+22] Jean Baptiste Joseph Fourier, Gaston Darboux, et al. *Théorie Analytique de La Chaleur*, volume 504. Didot Paris, 1822.

[FFW+18] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Data augmentation using synthetic data for time series classification with deep residual networks. *arXiv:1808.02455 [cs]*, August 2018.

[FPD+17] Germain Forestier, Francois Petitjean, Hoang Anh Dau, Geoffrey I. Webb, and Eamonn Keogh. Generating Synthetic Time Series to Augment Sparse Datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 865–870, New Orleans, LA, November 2017. IEEE.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GG05] Cyril Goutte and Eric Gaussier. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 345–359, Berlin, Heidelberg, 2005. Springer.

[GLB+20] D. Giardini, P. Lognonné, W. B. Banerdt, W. T. Pike, U. Christensen, S. Ceylan, J. F. Clinton, M. van Driel, S. C. Stähler, M. Böse, R. F. Garcia, A. Khan, M. Panning, C. Perrin, D. Banfield, E. Beucler, C. Charalambous, F. Euchner, A. Horleston, A. Jacob, T. Kawamura, S. Kedar, G. Mainsant, J.-R. Scholz, S. E. Smrekar, A. Spiga, C. Agard, D. Antonangeli, S. Barkaoui, E. Barrett, P. Combes, V. Conejero, I. Daubar, M. Drilleau, C. Ferrier, T. Gabsi, T. Gudkova, K. Hurst, F. Karakostas, S. King, M. Knapmeyer, B. Knapmeyer-Endrun, R. Llorca-Cejudo, A. Lucas, L. Luno, L. Margerin, J. B. McClean, D. Mimoun, N. Murdoch, F. Nimmo, M. Nonon, C. Pardo, A. Rivoldini, J. A. Rodriguez Manfredi, H. Samuel, M. Schimmel, A. E. Stott, E. Stutzmann, N. Teanby, T. Warren, R. C. Weber, M. Wieczorek, and C. Yana. The seismicity of Mars. *Nature Geoscience*, 13(3):205–212, March 2020.

[GMT16] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, September 2016.

[Goi16] Nicolas Goix. How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms ?, July 2016.

[Goi22] Nicolas Goix. EMMV_benchmarks, December 2022.

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR 2018*, Vancouver, Canada, April 2018.

[GWG+20] M. Golombek, N. H. Warner, J. A. Grant, E. Hauber, V. Ansan, C. M. Weitz, N. Williams, C. Charalambous, S. A. Wilson, A. DeMott, M. Kopp, H. Lethcoe-Wilson, L. Berger, R. Hausmann, E. Marteau, C. Vrettos, A. Trussell, W. Folkner, S. Le Maistre, N. Mueller, M. Grott, T. Spohn, S. Piqueux, E. Millour, F. Forget, I. Daubar, N. Murdoch, P. Lognonné, C. Perrin, S. Rodriguez, W. T. Pike, T. Parker, J. Maki, H. Abarca, R. Deen, J. Hall, P. Andres, N. Ruoff, F. Calef, S. Smrekar, M. M. Baker, M. Banks, A. Spiga, D. Banfield, J. Garvin, C. E. Newman, and W. B. Banerdt. Geology of the InSight landing site on Mars. *Nature Communications*, 11(1):1014, February 2020.

[GZM+20] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020.

[HHA+] Sandro Hawke, Ivan Herman, Phil Archer, Eric Prud'Hommeaux, and Ivan Herman. W3C Semantic Web Activity Homepage - Www.W3.Org/.

[HHL11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 507–523, Berlin, Heidelberg, 2011. Springer.

[Hin12] Geoffrey E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks : Tricks of the Trade : Second Edition*, Lecture Notes in Computer Science, pages 599–619. Springer, Berlin, Heidelberg, 2012.

[HKV19] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning : Methods, Systems, Challenges*. Springer Nature, 2019.

[Ho95] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, August 1995.

[Ho98] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998.

[HRRP20] Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing Change : Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, December 2020.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, NY, 2009.

[HTJS22] Chen Huang, Walter Talbott, Navdeep Jaitly, and Josh Susskind. Efficient Representation Learning via Adaptive Context Pooling, July 2022.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[IKE+16] Turker Ince, Serkan Kiranyaz, Levent Eren, Murat Askar, and Moncef Gabbouj. Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks. *IEEE Transactions on Industrial Electronics*, 63(11):7067–7075, November 2016.

[Ipe10] Panos Ipeirotis. The New Demographics of Mechanical Turk, 2010.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift, March 2015.

[ISO] Commité ISO. JPEG - JPEG 1 - Jpeg.Org/.

[JBZ+21] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A Survey on Contrastive Self-Supervised Learning. *Technologies*, 9(1):2, March 2021.

[JF18] Simon Jenni and Paolo Favaro. Self-Supervised Feature Learning by Learning to Spot Artifacts. *arXiv:1806.05024 [cs]*, June 2018.

[JML+20] Catherine L. Johnson, Anna Mittelholz, Benoit Langlais, Christopher T. Russell, Véronique Ansan, Don Banfield, Peter J. Chi, Matthew O. Fillingim, Francois Forget, Heidi Fuqua Haviland, Matthew Golombek, Steve Joy, Philippe Lognonné, Xinping Liu, Chloé Michaut, Lu Pan, Cathy Quantin-Nataf, Aymeric Spiga, Sabine Stanley, Shea N. Thorne, Mark A. Wieczorek, Yanan Yu, Suzanne E. Smrekar, and William B. Banerdt. Crustal and time-varying magnetic fields at the InSight landing site on Mars. *Nature Geoscience*, 13(3):199–204, March 2020.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[KBGA23] Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. GPT-4 Passes the Bar Exam, March 2023.

[KH21] Taekyung Ki and Youngmi Hur. Deep Scattering Network with Max-pooling. In *2021 Data Compression Conference (DCC)*, pages 348–348, March 2021.

[KIG17] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Personalized Monitoring and Advance Warning System for Cardiac Arrhythmias. *Scientific Reports*, 7(1):1–8, August 2017.

[KKLK19] Asif Khan, Dae-Kwan Ko, Soo Chul Lim, and Heung Soo Kim. Structural vibration-based classification and prediction of delamination in smart composite laminates using deep learning neural network. *Composites Part B : Engineering*, 161:586–594, March 2019.

[KKSZ11] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. Interpreting and Unifying Outlier Scores. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 13–24. Society for Industrial and Applied Mathematics, April 2011.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.

[Kri09] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Technical report no.4., University of Toronto*, page 60, 2009.

[kro68] L. kronecker. Ueber bilineare Formen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1868(68):273–285, January 1868.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[Kus64] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1):97–106, March 1964.

[KW21] Martijn Koot and Fons Wijnhoven. Usage impact on data center electricity needs : A system dynamic forecasting model. *Applied Energy*, 291:116798, June 2021.

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, December 1989.

[LBE15] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning, October 2015.

[LBG+19] P. Lognonné, W. B. Banerdt, D. Giardini, W. T. Pike, U. Christensen, P. Laudet, S. de Raucourt, P. Zweifel, S. Calcutt, M. Bierwirth, K. J. Hurst, F. Ijpelaan, J. W. Umland, R. Llorca-Cejudo, S. A. Larson, R. F. Garcia, S. Kedar, B. Knapmeyer-Endrun, D. Mimoun, A. Mocquet, M. P. Panning, R. C. Weber, A. Sylvestre-Baron, G. Pont, N. Verdier, L. Kerjean, L. J. Facto, V. Gharakanian, J. E. Feldman, T. L. Hoffman, D. B. Klein, K. Klein, N. P. Onufer, J. Paredes-Garcia, M. P. Petkov, J. R. Willis, S. E. Smrekar, M. Drilleau, T. Gabsi, T. Nebut, O. Robert, S. Tillier, C. Moreau, M. Parise, G. Aveni, S. Ben Charef, Y. Bennour, T. Camus, P. A. Dandonneau, C. Desfoux, B. Lecomte, O. Pot, P. Revuz, D. Mance, J. tenPierick, N. E. Bowles, C. Charalambous, A. K. Delahunty, J. Hurley, R. Irshad, Huafeng Liu, A. G. Mukherjee, I. M. Standley, A. E. Stott, J. Temple, T. Warren, M. Eberhardt, A. Kramer, W. Kühne, E.-P. Miettinen, M. Monecke, C. Aicardi, M. André, J. Baroukh, A. Borrien, A. Bouisset, P. Boutte, K. Brethomé, C. Brysbaert, T. Carlier, M. Deleuze, J. M. Desmarres, D. Dilhan, C. Doucet, D. Faye, N. Faye-Refalo, R. Gonzalez, C. Imbert, C. Larigauderie, E. Locatelli, L. Luno, J.-R. Meyer, F. Mialhe, J. M. Mouret, M. Nonon, Y. Pahn, A. Paillet, P. Pasquier, G. Perez, R. Perez, L. Perrin, B. Pouilloux, A. Rosak, I. Savin de Larclause, J. Sicre, M. Sodki, N. Toulemont, B. Vella, C. Yana, F. Alibay, O. M. Avalos, M. A. Balzer, P. Bhandari, E. Blanco, B. D. Bone, J. C. Bousman, P. Bruneau, F. J. Calef, R. J. Calvet, S. A. D'Agostino, G. de los Santos, R. G. Deen, R. W. Denise, J. Ervin, N. W. Ferraro, H. E. Gengl, F. Grinblat, D. Hernandez, M. Hetzel, M. E. Johnson, L. Khachikyan, J. Y. Lin, S. M. Madzunkov, S. L. Marshall, I. G. Mikellides, E. A. Miller, W. Raff, J. E. Singer, C. M. Sunday, J. F. Villalvazo, M. C. Wallace, D. Banfield, J. A. Rodriguez-Manfredi, C. T. Russell, A. Trebi-Ollennu, J. N. Maki, E. Beucler, M. Böse, C. Bonjour, J. L. Berenguer, S. Ceylan, J. Clinton, V. Conejero, I. Daubar, V. Dehant, P. Delage, F. Euchner, I. Estève, L. Fayon, L. Ferraioli, C. L. Johnson, J. Gagnepain-Beyneix, M. Golombek, A. Khan, T. Kawamura, B. Kenda, P. Labrot, N. Murdoch, C. Pardo, C. Perrin, L. Pou, A. Sauron, D. Savoie, S. Stähler,

E. Stutzmann, N. A. Teanby, J. Tromp, M. van Driel, M. Wieczorek, R. Widmer-Schnidrig, and J. Wookey. SEIS : Insight's Seismic Experiment for Internal Structure of Mars. *Space Science Reviews*, 215(1):12, January 2019.

[LBOM98] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. Efficient BackProp. In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks : Tricks of the Trade*, Lecture Notes in Computer Science, pages 9–50. Springer, Berlin, Heidelberg, 1998.

[LBP+20] P. Lognonné, W. B. Banerdt, W. T. Pike, D. Giardini, U. Christensen, R. F. Garcia, T. Kawamura, S. Kedar, B. Knapmeyer-Endrun, L. Margerin, F. Nimmo, M. Panning, B. Tauzin, J.-R. Scholz, D. Antonangeli, S. Barkaoui, E. Beucler, F. Bissig, N. Brinkman, M. Calvet, S. Ceylan, C. Charalambous, P. Davis, M. van Driel, M. Drilleau, L. Fayon, R. Joshi, B. Kenda, A. Khan, M. Knapmeyer, V. Lekic, J. McClean, D. Mimoun, N. Murdoch, L. Pan, C. Perrin, B. Pinot, L. Pou, S. Menina, S. Rodriguez, C. Schmelzbach, N. Schmerr, D. Sollberger, A. Spiga, S. Stähler, A. Stott, E. Stutzmann, S. Tharimena, R. Widmer-Schnidrig, F. Andersson, V. Ansan, C. Beghein, M. Böse, E. Bozdag, J. Clinton, I. Daubar, P. Delage, N. Fuji, M. Golombek, M. Grott, A. Horleston, K. Hurst, J. Irving, A. Jacob, J. Knollenberg, S. Krasner, C. Krause, R. Lorenz, C. Michaut, R. Myhill, T. Nissen-Meyer, J. ten Pierick, A.-C. Plesa, C. Quantin-Nataf, J. Robertsson, L. Rochas, M. Schimmel, S. Smrekar, T. Spohn, N. Teanby, J. Tromp, J. Vallade, N. Verdier, C. Vrettos, R. Weber, D. Banfield, E. Barrett, M. Bierwirth, S. Calcutt, N. Compaire, C. L. Johnson, D. Mance, F. Euchner, L. Kerjean, G. Mainsant, A. Mocquet, J. A. Rodriguez Manfredi, G. Pont, P. Laudet, T. Nebut, S. de Raucourt, O. Robert, C. T. Russell, A. Sylvestre-Baron, S. Tillier, T. Warren, M. Wieczorek, C. Yana, and P. Zweifel. Constraints on the shallow elastic and anelastic structure of Mars from InSight seismic data. *Nature Geoscience*, 13(3):213–220, March 2020.

[LEC+07] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 473–480, New York, NY, USA, June 2007. Association for Computing Machinery.

[LGW+19] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. PyWavelets : A Python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, April 2019.

[LHS20] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. Contrastive Representation Learning : A Framework and Review. *IEEE Access*, 8:193907–193934, 2020.

[LHSY17] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, October 2017.

[LMS17] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning Representations for Automatic Colorization. *arXiv:1603.06668 [cs]*, August 2017.

[LS22] Kevin Lee and Shubho Sengupta. Introducing the AI Research SuperCluster — Meta's cutting-edge AI supercomputer for AI research. https://ai.meta.com/blog/ai-rsc/, 2022.

[LSL17] Christina Lioma, Jakob Grue Simonsen, and Birger Larsen. Evaluation Measures for Relevance and Credibility in Ranked Lists, August 2017.

[LSV+18] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *ArXiv*, abs/1711.00436, 2018.

[LTFO20] Zinan Lin, Kiran Koshy Thekumparampil, Giulia Fanti, and Sewoong Oh. InfoGAN-CR and ModelCentrality : Self-supervised Model Training and Selection for Disentangling GANs, August 2020.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, December 2008.

[LTZ12] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):3:1–3:39, March 2012.

[LXZ15] Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things : A survey. *Information Systems Frontiers*, 17(2):243–259, April 2015.

[LZZW17] Dan Li, Jianxin Zhang, Qiang Zhang, and Xiaopeng Wei. Classification of ECG signals based on 1D convolution neural network. In *2017 IEEE 19th International Conference on E-Health Networking, Applications and Services (Healthcom)*, pages 1–6, October 2017.

[MC89] Michael McCloskey and Neal J. Cohen. Catastrophic Interference in Connectionist Networks : The Sequential Learning Problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, January 1989.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013.

[MCSZ20] Henrique O. Marques, Ricardo J. G. B. Campello, Jörg Sander, and Arthur Zimek. Internal evaluation of unsupervised outlier detection. *ACM Transactions on Knowledge Discovery from Data*, 14(4):1–42, August 2020.

[MCZS15] Henrique O. Marques, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. On the internal evaluation of unsupervised outlier detection. In *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, pages 1–12, La Jolla California, June 2015. ACM.

[Mey92] Yves Meyer. *Wavelets and Operators : Volume 1*. Cambridge University Press, 1992.

[MHM20] Leland McInnes, John Healy, and James Melville. UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, September 2020.

[MHWR91] L. Marquez, T. Hill, R. Worthley, and W. Remus. Neural network models as an alternative to regression. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, volume iv, pages 129–135 vol.4, January 1991.

[MMGR22] Marion Mainsant, Martial Mermillod, Christelle Godin, and Marina Reyboz. A study of the Dream Net model robustness across continual learning scenarios. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 824–833, November 2022.

[MML+13] Onur Mudanyali, Euan McLeod, Wei Luo, Alon Greenbaum, Ahmet F. Coskun, Yves Hennequin, Cédric P. Allier, and Aydogan Ozcan. Wide-field optical detection of nanoparticles using on-Chip microscopy and self-assembled nanolenses. *Nature Photonics*, 7(3):247–254, March 2013.

[MP21] Juliano Morimoto and Fleur Ponton. Virtual reality in biology : Could we become virtual naturalists ? *Evolution : Education and Outreach*, 14(1):7, May 2021.

[MPI20] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. Edge Machine Learning for AI-Enabled IoT Devices : A Review. *Sensors*, 20(9):2533, January 2020.

[MSPT21] Jesus Lovon Melgarejo, Laure Soulier, Karen Pinel-Sauvagnat, and Lynda Tamine. Oubli Catastrophique et Modèles Neuronaux de Recherche d'Information. In *17ème conférence francophone en Recherche d'Information et Application (CORIA 2021)*, page 1, April 2021.

[MSR+21] Marion Mainsant, Miguel Solinas, Marina Reyboz, Christelle Godin, and Martial Mermillod. Dream Net : A privacy preserving continual leaming model for face emotion recognition. In *2021 9th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 01–08, September 2021.

[MTH89] Geoffrey F Miller, Peter M Todd, and Shailesh U Hegde. Designing neural networks using genetic algorithms. In *ICGA*, volume 89, pages 379–384, 1989.

[MZH16] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and Learn : Unsupervised Learning Using Temporal Order Verification. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 527–544, Cham, 2016. Springer International Publishing.

[NF17] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. *arXiv:1603.09246 [cs]*, August 2017.

[NNN+15] Trung Thanh Nguyen, Anh Tuan Nguyen, Tuan Anh Ha Nguyen, Ly Thi Vu, Quang Uy Nguyen, and Long Dao Hai. Unsupervised Anomaly Detection in Online Game. In *Proceedings of the Sixth International Symposium on Information and Communication Technology*, pages 4–10, Hue City Viet Nam, December 2015. ACM.

[NNN16] Van Huy Nguyen, Thanh Trung Nguyen, and Uy Quang Nguyen. An evaluation method for unsupervised anomaly detection algorithms. *Journal of Computer Science and Cybernetics*, 32(3):259–272, 2016.

[NPJ+22] Thang L. Nguyen, Soorya Pradeep, Robert L. Judson-Torres, Jason Reed, Michael A. Teitell, and Thomas A. Zangle. Quantitative Phase Imaging : Recent Advances and Expanding Potential in Biomedicine. *ACS Nano*, 16(8):11516–11544, August 2022.

[NS94] Laurent Najman and Michel Schmitt. Watershed of a continuous function. *Signal Processing*, 38(1):99–112, July 1994.

[Ope23] OpenAI. GPT-4 Technical Report, March 2023.

[PC16] Parlement européen and Conseil de l'Union européenne. Règlement (UE) 2016/679 du Parlement européen et du Conseil du 27 avril 2016 relatif à la

protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données, et abrogeant la directive 95/46/CE (règlement général sur la protection des données) (Texte présentant de l'intérêt pour l'EEE), April 2016.

[PCS22] Johannes Pöppelbaum, Gavneet Singh Chadha, and Andreas Schwung. Contrastive learning based self-supervised time-series analysis. *Applied Soft Computing*, 117:108397, March 2022.

[Pea01] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[PKD+16] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders : Feature Learning by Inpainting. *arXiv:1604.07379 [cs]*, November 2016.

[Pos20] Seda Postalcıoğlu. Performance Analysis of Different Optimizers for Deep Learning-Based Image Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 34(02):2051003, February 2020.

[PPL+08] Gabriel Popescu, YoungKeun Park, Niyom Lue, Catherine Best-Popescu, Lauren Deflores, Ramachandra R Dasari, Michael S Feld, and Kamran Badizadegan. Optical imaging of cell mass and growth dynamics. *American Journal of Physiology-Cell Physiology*, 295(2):C538–C544, 2008.

[Pre98] Lutz Prechelt. Early Stopping - But When? In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks : Tricks of the Trade*, Lecture Notes in Computer Science, pages 55–69. Springer, Berlin, Heidelberg, 1998.

[RA15] Shebuti Rayana and Leman Akoglu. Less is More : Building Selective Anomaly Ensembles, January 2015.

[RAHL19a] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Aging evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, volume 2, page 2, 2019.

[RAHL19b] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized Evolution for Image Classifier Architecture Search, February 2019.

[RATC18] Cory A. Rieth, Ben D. Amsel, Randy Tran, and Maia B. Cook. Issues and Advances in Anomaly Detection Evaluation for Joint Human-Automated Systems. In Jessie Chen, editor, *Advances in Human Factors in Robots and Unmanned Systems*, Advances in Intelligent Systems and Computing, pages 52–63, Cham, 2018. Springer International Publishing.

[RBL+07] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning : Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 759–766, Corvalis, Oregon, USA, June 2007. Association for Computing Machinery.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[REC15] Karen Rose, Scott Eldridge, and Lyman Chapin. The internet of things : An overview. *The internet society (ISOC)*, 80:1–50, 2015.

[Rin08] Markus Ringnér. What is principal component analysis ? *Nature Biotechnology*, 26(3):303–304, March 2008.

[RM05] Lior Rokach and Oded Maimon. Decision Trees. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 165–192. Springer US, Boston, MA, 2005.

[RMN09] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 873–880, New York, NY, USA, June 2009. Association for Computing Machinery.

[RMS+17] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2902–2911, Sydney, NSW, Australia, August 2017. JMLR.org.

[RPG+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation, February 2021.

[S+91] Donald F Specht et al. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[Say19] Ignasi Sayol. Edge Computing : The technological trends' engine., December 2019.

[SBP+20] Léonard Seydoux, Randall Balestriero, Piero Poli, Maarten de Hoop, Michel Campillo, and Richard Baraniuk. Clustering earthquake signals and background noises in continuous seismic data with unsupervised deep learning. *Nature Communications*, 11(1):3972, August 2020.

[SDG09] Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*, 15(2):185–212, April 2009.

[SG22] Muhammad Shafiq and Zhaoquan Gu. Deep Residual Learning for Image Recognition : A Survey. *Applied Sciences*, 12(18):8972, January 2022.

[SG23] Raghubir Singh and Sukhpal Singh Gill. Edge AI : A survey. *Internet of Things and Cyber-Physical Systems*, 3:71–92, January 2023.

[SKKS10] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting : No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1015–1022, Madison, WI, USA, June 2010. Omnipress.

[Skl10] Rebecca Skloot. *The Immortal Life of Henrietta Lacks*. Picador, pan macmillan edition, 2010.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[SM02] Kenneth O. Stanley and Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, June 2002.

[Sol21] Miguel Angel Solinas. *Dual Memory System to Overcome Catastrophic Forgetting*. PhD thesis, Université Grenoble Alpes [2020-....], December 2021.

[Spe04] C. Spearman. "General Intelligence," Objectively Determined and Measured. *The American Journal of Psychology*, 15(2):201–292, 1904.

[SPS+01] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7):1443–1471, July 2001.

[SSA14] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-Thaw Bayesian Optimization, June 2014.

[SSN17] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 497–504, New York, NY, USA, July 2017. Association for Computing Machinery.

[SSW88] P. K Sahoo, S Soltani, and A. K. C Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233–260, February 1988.

[Ste35] W. Stephenson. Technique of Factor Analysis. *Nature*, 136(3434):297–297, August 1935.

[SVI+15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567 [cs]*, December 2015.

[SWD+20] John-Robert Scholz, Rudolf Widmer-Schnidrig, Paul Davis, Philippe Lognonné, Baptiste Pinot, Raphaël F. Garcia, Kenneth Hurst, Laurent Pou, Francis Nimmo, Salma Barkaoui, Sébastien de Raucourt, Brigitte Knapmeyer-Endrun, Martin Knapmeyer, Guénolé Orhand-Mainsant, Nicolas Compaire, Arthur Cuvier, Éric Beucler, Mickaël Bonnin, Rakshit Joshi, Grégory Sainton, Eléonore Stutzmann, Martin Schimmel, Anna Horleston, Maren Böse, Savas Ceylan, John Clinton, Martin van Driel, Taichi Kawamura, Amir Khan, Simon C. Stähler, Domenico Giardini, Constantinos Charalambous, Alexander E. Stott, William T. Pike, Ulrich R. Christensen, and W. Bruce Banerdt. Detection, Analysis, and Removal of Glitches From InSight's Seismic Data From Mars. *Earth and Space Science*, 7(11):e2020EA001317, 2020.

[SWS+99] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support Vector Method for Novelty Detection. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[SZ15] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[SZX+21] Jun Shi, Yanan Zhao, Wei Xiang, Vishal Monga, Xiaoping Liu, and Ran Tao. Deep Scattering Network With Fractional Wavelet Transform. *IEEE Transactions on Signal Processing*, 69:4740–4757, 2021.

[TLI+23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA : Open and Efficient Foundation Language Models, February 2023.

[UPP+17] Terry T. Um, Franz M. J. Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 216–220, Glasgow UK, November 2017. ACM.

[vDZ+16] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet : A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0 : Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.

[VH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using T-SNE. *Journal of machine learning research*, 9(11), 2008.

[vKE+16] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional Image Generation with PixelCNN Decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

[vLV19] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding, January 2019.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wai89] Alex Waibel. Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Computation*, 1(1):39–46, March 1989.

[Wes78] Joan S. Weszka. A survey of threshold selection techniques. *Computer Graphics and Image Processing*, 7(2):259–265, April 1978.

[WKMJ22] Kristoffer Wickstrøm, Michael Kampffmeyer, Karl Øyvind Mikalsen, and Robert Jenssen. Mixing up contrastive learning : Self-supervised representation learning for time series. *Pattern Recognition Letters*, 155:54–61, March 2022.

[WO18] Yichen Wu and Aydogan Ozcan. Lensless digital holographic microscopy and its applications in biomedicine and environmental monitoring. *Methods*, 136:4–16, March 2018.

[WR78] Joan S. Weszka and Azriel Rosenfeld. Threshold Evaluation Techniques. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(8):622–629, August 1978.

[XLD15] Z. Xu, S. Li, and W. Deng. Learning temporal features using LSTM-CNN architecture for face anti-spoofing. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 141–145, November 2015.

[XLSA20] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-Shot Learning – A Comprehensive Evaluation of the Good, the Bad and the Ugly, September 2020.

[YRC07] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On Early Stopping in Gradient Descent Learning. *Constructive Approximation*, 26(2):289–315, August 2007.

[YSHZ19] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A Review of Recurrent Neural Networks : LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270, July 2019.

[YWD+22] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. TS2Vec : Towards Universal Representation of Time Series, February 2022.

[YWX+18] Su Yi, Hao Wang, Wenqian Xue, Xiaojing Fan, Lefei Wang, Jun Tian, and Ryuichi Matsukura. Interference Source Identification for IEEE 802.15.4 wireless Sensor Networks Using Deep Learning. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, September 2018.

[YZC22] Xinyu Yang, Zhenguo Zhang, and Rongyi Cui. TimeCLR : A self-supervised contrastive learning framework for univariate time series representation. *Knowledge-Based Systems*, 245:108606, June 2022.

[ZCS14] Arthur Zimek, Ricardo J.G.B. Campello, and Jörg Sander. Ensembles for unsupervised outlier detection : Challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):11–22, March 2014.

[ZL17] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning, February 2017.

[ZLP+18] Wei Zhang, Chuanhao Li, Gaoliang Peng, Yuanhang Chen, and Zhujun Zhang. A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mechanical Systems and Signal Processing*, 100:439–453, February 2018.

[ZT14] Thomas A. Zangle and Michael A. Teitell. Live-cell mass profiling : An emerging approach in quantitative biophysics. *Nature Methods*, 11(12):1221–1228, December 2014.

[ZTF11] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

[ZVSL18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

[ZYW+18] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical Block-Wise Neural Network Architecture Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2423–2432, 2018.

ANNEXE

A

Annexe : Documentation technique de l'outil algorithmique de comparaison

Table des matières

A Apprentissage de représentation

A.1 Apprentissage profond

Les modèles d'apprentissage profond entrainés dans le cadre de cette étude sont principalement entrainés selon un apprentissage auto-supervisé. Ces modèles nécessitent le choix d'une tâche prétexte (chapitre II section B) et d'une architecture de réseaux de neurones (chapitre II section D.2). Avant cela, la méthode implémentée pour optimiser les hyperparamètres des différentes architectures est présentées chapitre III section C.

A.1.1 Tâche prétexte et `dataLoader`

Le `dataLoader` a deux utilisations principales. C'est l'outil qui permet de charger les données, mais aussi de les préparer pour le modèle. Ainsi, il doit pouvoir préparer les données et les séparer en entrée/sortie pour que modèle achève la tache prétexte qui lui incombe.

Paramètres

- `loadpath (str)` chemin d'accès vers les données à charger
- `upscale_delta (int)` Taille du pas pour la fenêtre glissante utilisée pour l'augmentation des données.
- `input_length (int, list[int])` taille (en nombre de point) de l'entrée
- `output_length (int, list[int])` taille (en nombre de point) de la sortie

Des paramètres globaux peuvent aussi être passés à l'ensemble des `dataLoader` via le dictionnaire `"globalDataLoader"`.

Tâches prétextes Cinq `dataLoader` sont implémentés sous la forme de cinq tâches prétextes :

- La reconstruction du signal (l'entrée et la sortie sont identiques) au travers de la `class AutoEncoder()`,
- Le débruitage, via la `class DenoisingAutoEncoder()` qui hérite de la classe précédente et rajoute les paramètres `mutation`, `mean` et `std` qui définissent respectivement le facteur de mutation, la moyenne et l'écart type du bruit gaussien utilisé,
- La prédiction du futur `class FuturePrediction()`,
- La prédiction du passé `class PastPrediction()`,
- La prédiction du contexte `class ContextPrediction()`, l'instant à partir duquel une partie des données sont occultées est défini par le paramètre `i0`

A.1.2 Architectures implémentées

Dans cette section les architectures de réseaux de neurones implémentées et les paramètres qui peuvent être modifiés sont présentés. Les architectures sont basées sur la librairie `Tensorflow` [AAB+15] et son module `keras`.

Toutes les architectures disposent des paramètres (avec leur valeur par défaut) :

- `loss='MSE'` : Fonction de cout à utiliser pour l'apprentissage des poids du réseau,
- `metric='MSE'` : Métrique finale à utiliser pour comparer les architectures,
- `optimizer='ADAM'` : Optimisateur à utiliser,

— `learning_rate=0.001` : Taux d'apprentissage à chaque actualisation des poids

Les paramètres `input_shape` et `label_size` qui définissent la forme de l'entrée et de la sortie sont passés automatiquement depuis le `dataLoader` .

Perceptrons Multicouches La classe `class DenseNet()` présentée chapitre II section B.2 chapitre II section B.2 a pour paramètres :

— `dense_size (list[int])` Liste contenant les tailles de chaque couche `dense`

— `dropout=0 (float)` Taux de Droupout pour les couches denses du décodeur

— `activation='relu' (str)` Fonction d'activation à utiliser au sein du réseau

— `last_activation='linear' (str)` Fonction d'activation à utiliser pour la couche de sortie

Réseaux convolutionnels à 1 dimension La classe `class CNN1D()` présentée chapitre II section B.3 a pour paramètres :

— `nb_conv (int)` Nombre de couches convolutionnelles de l'architecture.

— `nb_filter (int|list)` Nombre de filtres par couches convolutionnelles. Si c'est un `int` , le nombre de filtres est identique pour chaque couche. Si c'est une liste il est défini différemment à chaque couche convolutionnelle par l'élément de la liste correspondant.

— `kernel_size (int|list)` Taille de kernel pour les couches convolutionnelles. Si c'est un `int` , le taille des kernel est identique pour chaque couche. Si c'est une liste il est défini différemment à chaque couche convolutionnelle par l'élément de la liste correspondant.

— `dense_size (int|list)` Tailles de couches denses du décodeur. Une unique couche Dense si c'est un (int), plusieurs si c'est une liste.

— `dilation_rate=1 (int|list|"2i")` Taux de dilatation par couche. Si c'est un `int` , le taux de dilatation est identique pour chaque couche. Si c'est une liste il est défini différemment à chaque couche convolutionnelle par l'élément de la liste correspondant. Enfin la valeur `"2i"` correspond à une valeur de taux de dilatation qui croit exponentiellement avec le nombre de couches comme proposé dans [vKE+16]. Le taux de dilatation vaut 2^i avec i l'indice de la couche convolutionnelle.

— `strides=1 (int)` Pas des convolutions

— `maxpooling_period=0 (int)` Période à laquelle placer une couche de `maxpooling` . Pour une valeur de 3, une couche de `maxpooling` est placée toute les 3 couches convolutionnelles.

— `pool_size=2 (int)` Taille des couches de maxpooling,

`dropout=0 (float)` Taux de Droupout pour les couches denses du décodeur

— `activation='relu' (str)` Fonction d'activation à utiliser pour les couches convolutionnelles

— `dense_activation='relu' (str)` Fonction d'activation à utiliser pour les couches denses

— `last_activation='linear' (str)` Fonction d'activation à utiliser pour la couche de sortie

LSTM L'achitecture `class LSTM()` présentée chapitre II section B.4 a pour paramètres :

- `nb_lstm_layers (int|list[int])` Nombre de couches LSTM.
- `lstm_units (int|list[int])` Nombre d'unités LSTM au sein d'une couche.
- `dense_size (int|list[int])` Tailles de couches denses du décodeur. Une unique couche Dense si c'est un (int), plusieurs si c'est une liste.
- `dropout=0 (float)` Taux de Droupout pour les couches denses du décodeur
- `lstm_activation='tanh' (str)` Fonction d'activation à utiliser pour les couches lstm
- `dense_activation='tanh' (str)` Fonction d'activation à utiliser pour les couches dense
- `last_activation='linear' (str)` Fonction d'activation à utiliser pour la couche de sortie

Réseaux avec connexions résiduelles (ResNet) L'architecture avec connexions résiduelles `class Resnet()` est présentée chapitre II section B.5. Elle est basée sur une architecture de CNN 1D avec des connexions résiduelles entre les blocs ResNet.

- `nb_resnet_blocks (int)` nombre de blocs ResNet à enchainer dans l'architecture.
- `nb_filter (int|list)` Nombre de filtres par couches convolutionnelles.
- `kernel_size (int|list)` Taille de kernel pour les couches convolutionnelles.
- `dilation_rate=1 (int|list|str)` Taux de dilatation par couche. Si c'est un `int`, le taux de dilatation est identique pour chaque couche. Si c'est une liste il est défini différemment à chaque couche convolutionnelle par l'élément de la liste correspondant.
- `dense_size (int|list)` Tailles de couches denses du décodeur. Une unique couche Dense si c'est un (int), plusieurs si c'est une liste.
- `dropout = 0.25` Dropout utilisé au sein des blocs ResNet
- `dense_activation='relu' (str)` Fonction d'activation à utiliser pour les couches dense
- `last_activation='linear' (str)` Fonction d'activation à utiliser pour la couche de sortie

Réseaux en U (Unet) L'architecture en U `class Unet()` est basée sur des couches convolutionnelles et des connexions résiduelles entre les différents niveaux d'encodeurs et de décodeurs. Ses paramètres sont

- `nb_encoder_blocks` Nombre de blocs d'encodeurs. Cette variable définit aussi le nombre de blocs de Décodeurs puisqu'il y en a autant que d'encodeurs.
- `nb_filter (int|list)` Nombre de filtres par couches convolutionnelles.
- `kernel_size (int|list)` Taille de kernel pour les couches convolutionnelles.
- `latent_space_size (int)` Taille de l'espace latent au milieu de l'architecture en U.
- `nb_convs = 2` Nombre de couches convolutionnelles des blocs encodeurs.
- `pool_size = 2` Taille du maxpooling des couches encodeurs
- `activation = 'relu'` Fonction d'activation des couches convolutionnelles
- `last_activation = 'linear'` Fonction d'activation à utiliser pour la couche de sortie

Transformers L'architecture `class Transformers()` présentée chapitre II section B.6 est composée dans notre cas d'un encodeur auto-attentif et d'un décodeur dense :

- `nb_encoder_blocks (int)` Nombre de blocs encodeurs.

— `d_model (int)` Dimension du modèle dans un bloc encodeur.

— `nb_heads (int)` nombre de têtes auto-attentives par bloc encodeur.

— `ff_dim (int)` Dimension de la couche Dense de la couche des blocs encodeurs.

— `dropout (float)` Taux de dropout des blocs encodeurs.

— `dense_size (int|list)` Tailles de couches denses du décodeur. Une unique couche Dense si c'est un (int), plusieurs si c'est une liste.

— `last_activation = 'linear'` Fonction d'activation à utiliser pour la couche de sortie

A.2 Représentations non basées sur l'apprentissage profond

La classe `class Wavelet()` implémente la décomposition en ondelette (chapitre II section D.4). Le paramètre `wavelet (str)` permet de définir la base d'ondelettes à utiliser pour la décomposition et `mode (str)` est la méthode de padding à utiliser. Plus d'informations sont disponibles dans la documentation de la librairie `PyWavelets` [LGW+19].

L'ensemble de méthodes présentées chapitre II section D.4 sont implémentées sous la forme des classes ci-dessous qui sont des `wrapper` des fonctions implémentées dans les librairies `Scikit-learn` [BLB+13] et `SciPy` [VGO+20].

Toutes les classes ci-dessous ont un attribut `n_components` définissant la taille de la base dans laquelle doivent être projetées les données. Dans certains cas, cet argument ne restreint pas l'espace de projection, mais seulement les `n_components` sont renvoyés à l'utilisateur.

— `class FactorAnalysis()` Analyse factorielle (chapitre II section D.4)

— `class Fourier()` Transformée de Fourier discrète (chapitre II section D.4)

— `class DCT()` Transformée en Cosinus Discrète (chapitre II section D.4)

— `class PCA()` Analyse en composante principale (chapitre II section D.4)

— `class ICA()` Analyse en composante indépendante (chapitre II section D.4)

— `class KernelPCA()` Analyse en composante principale à noyau (chapitre II section D.4). Le paramètre `kernel='linear' (str)` permet de définir le noyau à utiliser.

— `class RandomProjection()` Projection aléatoire (chapitre II section D.4). Le paramètre `kind` permet de définir le type de bruit (`'sparse'` ou `'gaussian'`) à utiliser pour la génération des vecteurs aléatoires.

B Détection d'anomalie

Les méthodes de détection d'anomalies implémentées dans l'outil sont celles présentées chapitre II section A.

La classe `class AnomalyDetection()` est la classe mère dont hérite tous les détecteurs. Elle implémente les méthodes `fit(y_true, y_pred)` et `transform` qui permettent respectivement d'entrainer le détecteur sur données et d'obtenir un vecteur correspondant aux annotations prédits des données. Hormis la classe `Threshold()` toutes les classes ci-dessous sont des `wrapper` des classes de détection d'anomalies correspondantes implémentées dans `Scikit-learn` [BLB+13].

B.1 Détecteur à seuil

Le détecteur à seuil est implémenté dans l'outil au travers de la classe `class Threshold()`. Plusieurs réglages sont possibles pour ce détecteur. Les fenêtres sont ordonnées selon leur valeur de métrique calculée selon la valeur de l'attribut `metric="mse"`. Le type de l'attribut `thresh` définit le comportement du détecteur. Si c'est un `int`, les `thresh` fenêtres avec la métrique la plus élevée sont considérées comme anormales. Si c'est un `float` ce sont les `thresh` premiers pourcents qui sont anormales. Enfin, `thresh` prend pour valeur la chaine de caractère `'2-sigma'` le seuil est calculé dans la méthode fit comme la moyenne plus deux fois l'écart type du jeu utilisé dans la méthode `fit(y_true, y_pred)`

B.2 Facteur Local d'Anomalie

La classe `class LocalOutlierFactor()` est une surcouche de `sklearn.neighbors.LocalOutlierFactor()` [BLB+13]. Ce détecteur repose principalement sur le nombre de voisins à considérer `n_neighbors` pour mesurer la densité de l'espace de représentation autour d'un point.

Paramètres :

- `n_neighbors: int default=20` : Number of neighbors to use by default for kneighbors queries. If $n_neighbors is larger than the number of samples provided, all samples will be used.$
- `algorithm: {'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'` : Algorithm used to compute the nearest neighbors :
  - `'ball_tree'` will use `BallTree`
  - `'kd_tree'` will use `KDTree`
  - `'brute'` will use a brute-force search.
  - `'auto'` will attempt to decide the most appropriate algorithm based on the values passed to fit method.

  `leaf_size: int, default=30`
  Leaf size passed to `BallTree` or `class KDTree`. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.
- `metric: str or callable, default='minkowski'` : Metric used for the distance computation.
- `p: int, default=2` : Parameter for the Minkowski metric from. When $p = 1$, this is equivalent to using `manhattan_distance` (l_1), and `euclidean_distance` (l_2) for $p = 2$. For arbitrary p, `minkowski_distance` (l_p) is used.
- `metric_params: dict, default=None` : Additional keyword arguments for the metric function.
- `contamination: 'auto' or float, default='auto'` : Define the proportion of anomalies to be found within the dataset.
- `novelty: bool, default=False` : By default, LocalOutlierFactor is only meant to be used for outlier detection (novelty=False). Set novelty to True if you want to use LocalOutlierFactor for novelty detection. In this case be aware that you should only use predict, `decision_function` and `score_samples` on new unseen data and not on the training set.
- `n_jobs: int, default=None` : The number of parallel jobs to run for neighbors search.

B.3 SVM à une classe

La classe `class OneClassSVM()` est une surcouche de `sklearn.svm.OneClassSVM()` [BLB+13]. Les OC-SVM cherchent à isoler les données normales du reste de l'espace latent par une hypersphère. La fonction noyau à utiliser pour le calcul de cette hypershpère est défini par le paramètre `kernel` .

Paramètres :

- `kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'` : Specifies the kernel type to be used in the algorithm.
- `degree: int, default=3` : Degree of the polynomial kernel function (`'poly'`). Ignored by all other kernels.
- `gamma: {'scale', 'auto'} or float, default='scale'` : Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
  - if `gamma='scale'` (default) is passed then it uses `1 / (n_features * X.var())` as value of gamma,
  - if `'auto'` , uses $\frac{1}{n_{\text{features}}}$.
- `coef0: float, default=0.0` : Independent term in kernel function. It is only significant in `'poly'` and ' `sigmoid'` .
- `tol: float, default=1e-3` : Tolerance for stopping criterion.
- `nu: float, default=0.5` : An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Should be in the interval $(0, 1]$.
- `shrinking: bool, default=True` : Whether to use the shrinking heuristic.
- `cache_size: float, default=200` : Specify the size of the kernel cache (in MB).
- `max_iter: int, default=-1` : Hard limit on iterations within solver, or -1 for no limit.

B.4 Isolation Forest

est une surcouche de `sklearn.ensemble.IsolationForest()` [BLB+13]. Ses paramètres principaux sont `n_estimator` le nombre d'arbres composants la forêt d'estimateurs. `max_samples` et `max_features` le nombre respectifs d'échantillons et de features à utiliser pour l'entrainement d'un unique arbre.

Paramètres :

- `n_estimators: int, default=100` : The number of base estimators in the ensemble.
- `max_samples: "auto", int or float, default="auto"` : The number of samples to draw from `X` to train each base estimator.
  - If `int` , then draw `max_samples` samples.
  - If `float` , then draw `max_samples * X.shape[0]` samples.
  - If `"auto"` , then `max_samples=min(256, n_samples)` .
- `contamination: 'auto' or float, default='auto'` : Define the proportion of anomalies to be found within the dataset.
- `max_features: int or float, default=1.0` : The number of features to draw from X to train each base estimator.

  - If int, then draw `max_features` features.
  - If float, then draw `max_features * X.shape[1]` features.

- `bootstrap: bool, default=False` : If `True` , individual trees are fit on random subsets of the training data sampled with replacement. If False, sampling without replacement is performed.
- `n_jobs: int, default=None` : The number of jobs to run in parallel for both `fit` and `predict` .
- `random_state: int, RandomState instance or None, default=None` : Controls the pseudo-randomness of the selection of the feature and split values for each branching step and each tree in the forest.
- `warm_start: bool, default=False` : When set to `True` , reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest.

Les méthodes implémentées ci-dessus ne nécessitent pas de label afin de créer un classificateur normal/anormal des séries temporelles. Néanmoins, dans le cas d'une étude disposant d'annotation de séries temporelles anormales, d'autres classificateurs pourraient être implémentés.

C Un exemple d'utilisation

L'ensemble des paramètres de configurations sont passés au script principal via un fichier de configuration `config.json` . Chaque paramètre y est nommé entre doubles quotes et la ou les valeurs qu'il peut prendre sont explicitées ensuite `{"param1" = [1,2,3], "param2" = 4}`

Le liste A.1 est le fichier de configuration utilisé pour réaliser les expériences de la chapitre IV section D

Les hyperparamètres globaux utiles pour toutes les architectures comme le nombre d'époques sont définis par la variable `"hyperparameter"` .

Les paramètres concernant tous les `dataLoader` comme le chemin d'accès des données ou le pas de la fenêtre glissante d'augmentation de données sont définis dans `"dataLoader"` . Dans le cas de cette expérience, ceux ci auraient pu être définis dans l'unique `"dataloader"` qu'est `"futurePrediction"` .

Enfin, le paramètre `"architecture"` permet de lister les différentes architectures à optimiser. Comme les perceptrons multicouches `"dense"` , les CNN-1D à maxpooling `"cnn-maxpool"` ou a taux de dilatation `"cnn-dilation"` etc...

On remarque par exemple que pour les CNN 1D à dilatation, le réseau dense à utiliser comme décodeur est celui qui a eu les meilleurs résultats pour les CNN 1D à maxpooling `"dense_size":["best-cnn-maxpool"]` (ligne 40).

Listing A.1 – Fichier de configuration de l'expérience de la chapitre IV section D

```
{"hyperparameter": {
    "epochs": 30,
    "verbose": 1
},
"globalDataloader": {
    "loadpath":"~/CELL_DATA/WORKING/Data_A/",
    "upscale_delta":1,
    "batch_size":64
},
"dataLoader": {
    "futurePrediction":{
        "input_length": [120],
        "label_length": [60]
    }
},
"architecture": {
    "denseNet": {
        "dense_size": [
            [32, 32, 32], [64, 64, 64], [128, 128, 128],
            [32, 32, 32, 32, 32], [64, 64, 64, 64, 64],
            [512,256,128,64,32],[256,128,64],[32,64,128]
        ]
    },
    "cnn-maxpool":{
        "nb_conv":[3, 5, 9, 12, 16, 32],
        "kernel_size": [3, 5, 8, 12, 16],
        "maxpooling_period": [3,2,1],
        "nb_filter": [64, 128, 256],
        "dense_size":[[64], [32], [16], [128], [256],
            [16,16], [32,32], [64,64],
            [128,128], [16,16,16], [32,32,32], [64,64,64],
            [256, 128], [128,64], [64,32], [32,16], [128, 64, 32], [64,32,16]
        ]
    },
    "cnn-dilation":{
        "nb_conv":[3, 5, 9, 12, 16, 32],
        "kernel_size": [5, 3, 8, 12, 16],
        "dilation_rate":["2i",1, 2, 4, 8, 16],
        "nb_filter": [64, 128, 256],
        "dense_size":["best-cnn-maxpool"]
    },
    "lstm" : {
        "nb_lstm_layers" : [1,2,3],
        "lstm_units" : [64, 32, 128, 256, 512],
        "dense_size" : ["best-cnn-maxpool"]
    },
    "resnet": {
        "nb_resnet_blocks" :  [3, 5, 10, 12, 16, 24, 32, 50],
        "dilation_rate" : ["best-cnn-dilation", "2i", 2, 4, 8, 16],
        "kernel_size" : ["best-cnn-dilation"],
        "nb_filter" : ["best-cnn-dilation"],
        "dense_size" : ["best-cnn-maxpool"]
}}}
```

ANNEXE

B

Annexes à l'application sur les données cellulaires

Table des matières

A Figures avancées pour l'analyse des séries temporelles de cellules anormales

B Tableau complet des résultats de l'outil algorithmique pour la comparaison d'architectures de réseaux de neurones pour la tâche prétexte de prédiction

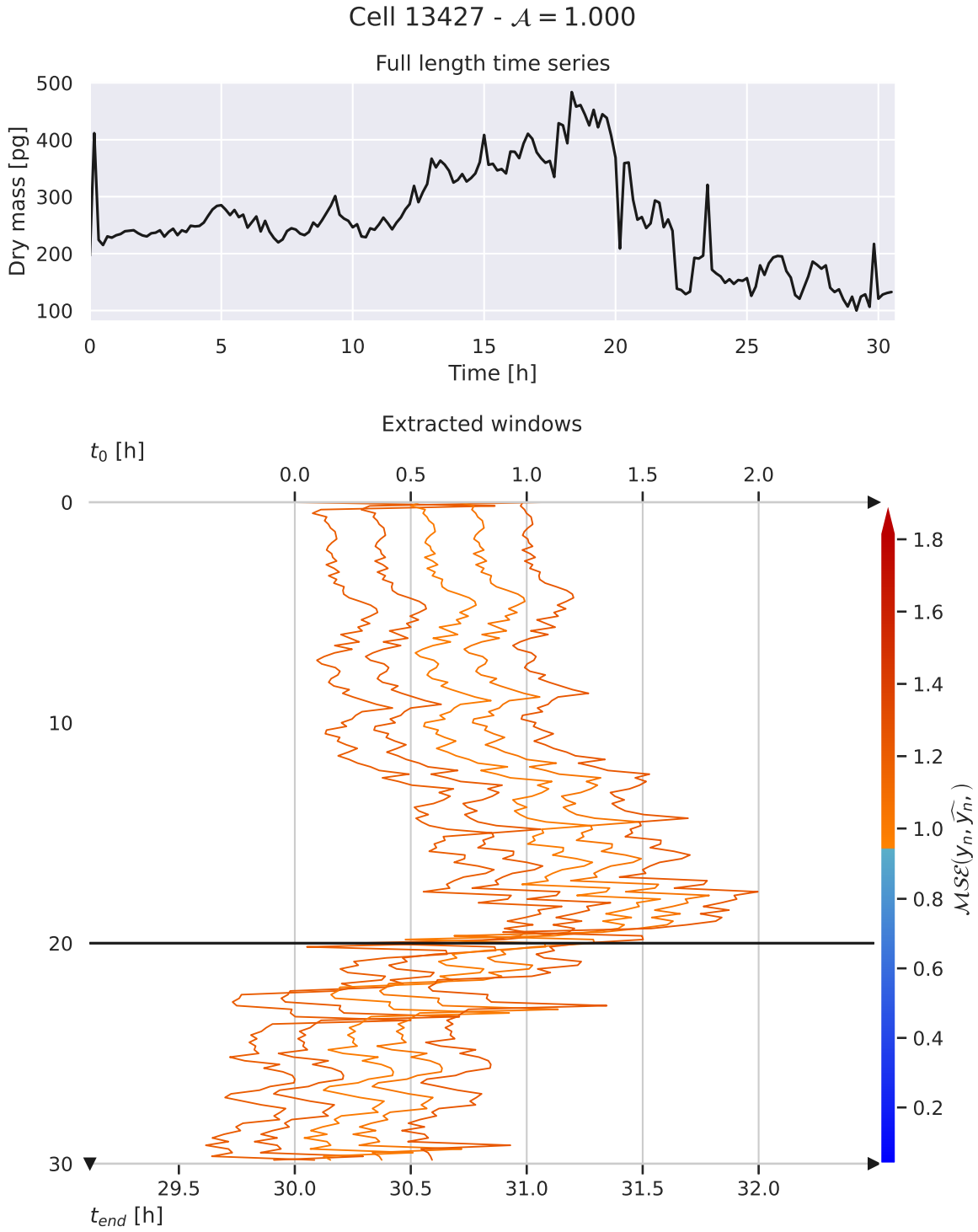
Table B.1 : Tableau complet des résultats d'expériences de différentes architectures pour la tâche prétexte de prédiction pour le jeu de données de masse sèche de cellules

Paramètres					EQM min	EQM moyenne	Nb param	Durée entrainement	Durée validation
$\mathcal{D}$					Réseaux denses				
32/32/32					0.474	0.478 ± 0.001	7964	195 ± 33 s	4.70 ± 0.04 s
64/64/64					0.460	0.464 ± 0.001	19964	158 ± 20 s	4.68 ± 0.05 s
128/128/128					0.454	0.459 ± 0.002	56252	161 ± 23 s	4.67 ± 0.07 s
32/32/32/32/32					0.482	0.487 ± 0.014	10076	281 ± 289 s	4.96 ± 0.01 s
64/64/64/64/64					0.470	0.482 ± 0.003	28284	177 ± 25 s	5.02 ± 0.10 s
256/128/64					0.454	0.458 ± 0.001	76028	182 ± 13 s	5.94 ± 0.25 s
512/256/128/64/32					0.461	0.471 ± 0.004	238492	174 ± 26 s	5.31 ± 0.21 s
32/64/128					0.477	0.482 ± 0.003	22044	131 ± 26 s	6.70 ± 0.25 s
$\mathcal{C}$	$\mathcal{K}$	$\mathcal{F}$	$\mathcal{MP}$	$\mathcal{D}$	CNN-1D à Maxpooling				
3	3	64	3	64	0.467	0.481 ± 0.003	274684	180 ± 26 s	5.58 ± 0.04 s
5	3	64	3	64	0.468	0.478 ± 0.002	299388	172 ± 7 s	5.91 ± 0.05 s
6	3	64	3	64	0.468	0.475 ± 0.002	188860	210 ± 16 s	6.27 ± 0.08 s
9	3	64	3	64	0.461	0.470 ± 0.002	164476	296 ± 21 s	7.06 ± 0.04 s
12	3	64	3	64	0.516	0.525 ± 0.002	168764	358 ± 29 s	7.83 ± 0.08 s
16	3	64	3	64	0.585	0.609 ± 0.024	201788	451 ± 140 s	10.23 ± 0.04 s
9	5	64	3	64	0.471	0.478 ± 0.002	230140	344 ± 31 s	8.18 ± 0.32 s
9	8	64	3	64	0.489	0.498 ± 0.005	328636	369 ± 120 s	7.66 ± 0.13 s
9	12	64	3	64	0.511	0.530 ± 0.012	459964	490 ± 63 s	7.94 ± 0.06 s
9	16	64	3	64	0.545	0.550 ± 0.002	591292	686 ± 154 s	8.16 ± 0.12 s
9	3	32	3	64	0.471	0.478 ± 0.004	59644	291 ± 56 s	6.75 ± 0.11 s
9	3	128	3	64	0.478	0.608 ± 0.085	521596	401 ± 51 s	8.48 ± 0.15 s
9	3	256	3	64	0.501	0.622 ± 0.081	1825660	488 ± 96 s	8.32 ± 0.11 s
9	3	64	2	64	0.515	0.523 ± 0.002	131708	341 ± 35 s	8.35 ± 0.08 s
9	3	64	3	16	0.469	0.479 ± 0.002	115468	365 ± 30 s	8.36 ± 0.22 s
9	3	64	3	32	0.466	0.472 ± 0.002	131804	352 ± 17 s	8.23 ± 0.10 s
9	3	64	3	128	0.460	0.469 ± 0.002	229820	327 ± 26 s	8.46 ± 0.17 s
9	3	64	3	256	0.461	0.471 ± 0.003	360508	323 ± 23 s	8.43 ± 0.24 s
9	3	64	3	16/16	0.469	0.475 ± 0.002	115740	321 ± 39 s	8.46 ± 0.21 s
9	3	64	3	32/32	0.461	0.471 ± 0.002	132860	381 ± 32 s	8.48 ± 0.09 s
9	3	64	3	64/64	0.468	0.475 ± 0.001	168636	321 ± 33 s	8.07 ± 0.12 s
9	3	64	3	128/128	0.466	0.475 ± 0.002	246332	316 ± 34 s	8.05 ± 0.16 s
9	3	64	3	64/32	0.473	0.475 ± 0.001	164636	377 ± 62 s	8.28 ± 0.12 s
9	3	64	3	128/64	0.465	0.470 ± 0.002	234236	276 ± 21 s	8.46 ± 0.43 s

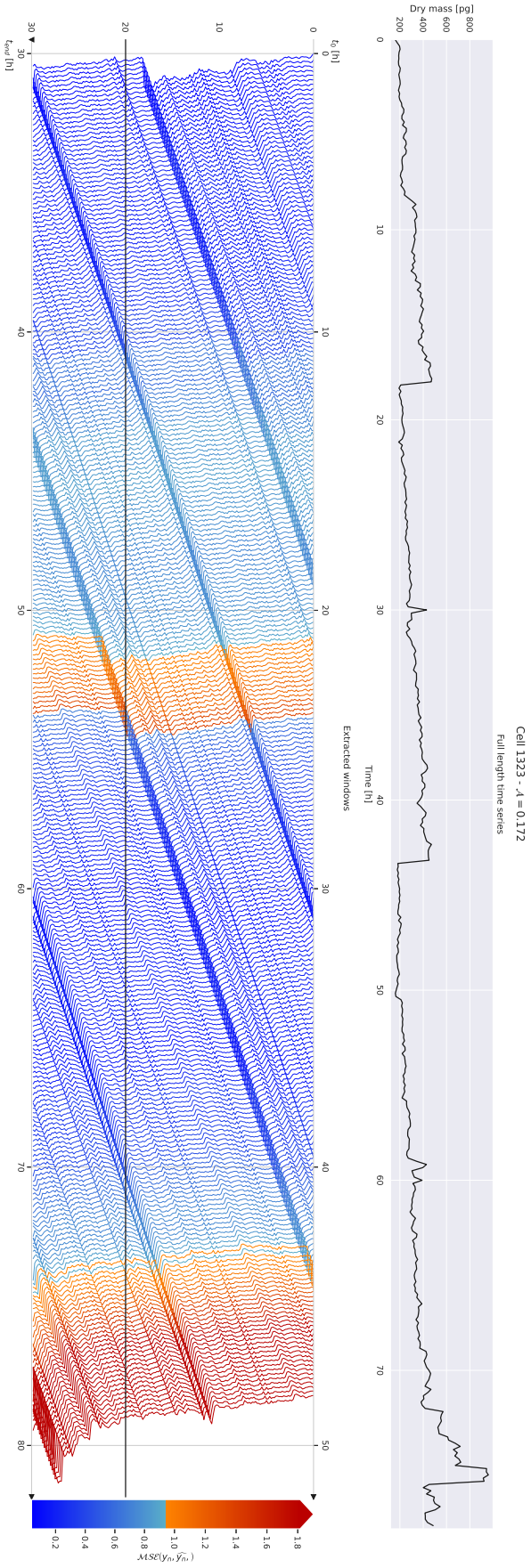
Suite à la page suivante

Suite de la page précédente

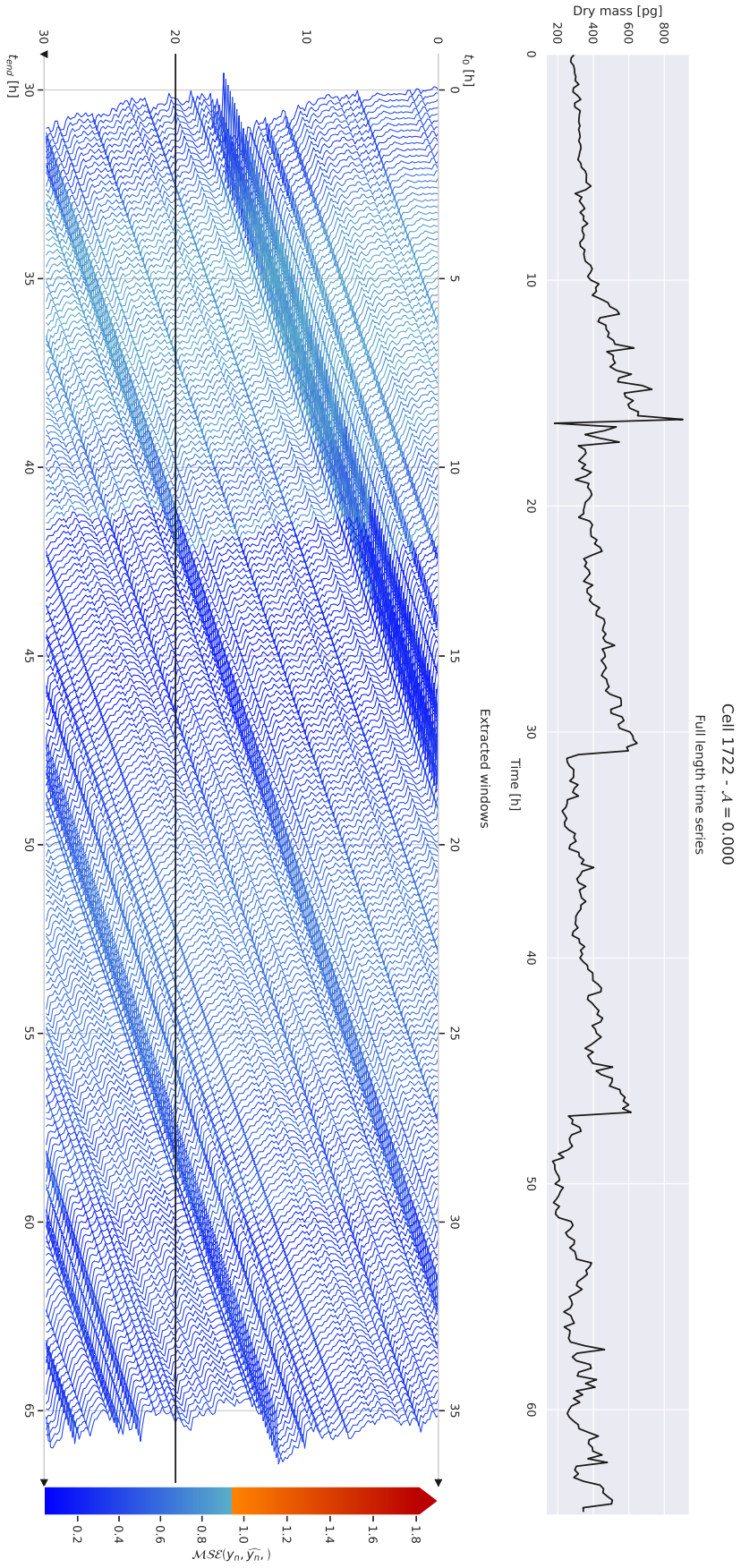
9	3	64	3	256/128	0.466	0.474 ± 0.002	385 724	316 ± 28 s	8.08 ± 0.15 s
9	3	64	3	64/32/16	0.466	0.473 ± 0.002	164 204	302 ± 16 s	8.49 ± 0.15 s
9	3	64	3	128/64/32	0.473	0.473 ± 0.000	234 396	337 ± 0 s	8.29 ± 0.00 s
$\mathcal{C}$	$\mathcal{K}$	$\mathcal{F}$	$\mathcal{DR}$		CNN-1D à taux de dilatation				
3	3	64	$2i$		0.466	0.475 ± 0.003	1 015 868	192 ± 36 s	5.85 ± 0.15 s
5	3	64	$2i$		0.461	0.470 ± 0.002	1 040 572	239 ± 13 s	6.50 ± 0.12 s
9	3	64	$2i$		0.459	0.472 ± 0.003	1 089 980	362 ± 47 s	7.77 ± 0.11 s
12	3	64	$2i$		0.465	0.476 ± 0.002	1 127 036	626 ± 80 s	11.39 ± 0.23 s
9	5	64	$2i$		0.466	0.476 ± 0.003	1 155 644	385 ± 65 s	8.42 ± 0.23 s
9	8	64	$2i$		0.484	0.498 ± 0.003	1 254 140	272 ± 15 s	8.39 ± 0.15 s
9	12	64	$2i$		0.500	0.513 ± 0.003	1 385 468	263 ± 6 s	8.57 ± 0.23 s
9	16	64	$2i$		0.505	0.518 ± 0.004	1 516 796	311 ± 12 s	9.36 ± 0.38 s
9	3	128	$2i$		0.487	0.633 ± 0.090	2 368 700	356 ± 51 s	8.93 ± 0.34 s
9	3	256	$2i$		0.598	0.812 ± 0.052	5 515 964	581 ± 61 s	10.36 ± 0.22 s
9	3	64	1		0.480	0.492 ± 0.004	1 089 980	256 ± 15 s	7.98 ± 0.16 s
9	3	64	2		0.477	0.494 ± 0.005	1 089 980	248 ± 10 s	7.78 ± 0.28 s
9	3	64	4		0.476	0.487 ± 0.004	1 089 980	298 ± 17 s	8.56 ± 0.17 s
9	3	64	8		0.476	0.490 ± 0.003	1 089 980	276 ± 15 s	8.25 ± 0.03 s
9	3	64	16		0.471	0.481 ± 0.004	1 089 980	307 ± 25 s	7.46 ± 0.09 s
$\mathcal{L}$		$\mathcal{N}$			LSTM				
1		64			0.468	0.471 ± 0.000	32 956	174 ± 24 s	5.58 ± 0.38 s
2		64			0.471	0.478 ± 0.002	65 980	173 ± 5 s	6.05 ± 0.38 s
3		64			0.473	0.482 ± 0.002	99 004	205 ± 11 s	6.52 ± 0.12 s
1		32			0.465	0.467 ± 0.000	16 316	191 ± 14 s	5.61 ± 0.12 s
1		128			0.473	0.477 ± 0.001	90 812	142 ± 3 s	5.54 ± 0.10 s
1		256			0.478	0.483 ± 0.001	304 828	132 ± 6 s	5.50 ± 0.09 s
1		512			0.482	0.485 ± 0.003	1 126 076	145 ± 3 s	5.71 ± 0.20 s
$\mathcal{B}$		$\mathcal{DR}$			RESNET				
1		$2i$			0.543	0.561 ± 0.004	29 180	238 ± 25	6.01 ± 0.27
2		$2i$			0.510	0.524 ± 0.010	54 396	260 ± 29	6.59 ± 0.33
3		$2i$			0.499	0.514 ± 0.003	79 612	304 ± 31	7.50 ± 0.08
5		$2i$			0.496	0.508 ± 0.004	130 044	488 ± 59	8.45 ± 0.02
5		2			0.499	0.516 ± 0.005	130 044	568 ± 51	9.35 ± 0.12
5		4			0.494	0.493 ± 0.000	130 044	354 ± 0	8.41 ± 0.00
5		8			0.471	0.480 ± 0.003	130 044	510 ± 28	9.40 ± 0.12
5		16			0.473	0.481 ± 0.001	130 044	584 ± 52	8.57 ± 0.09



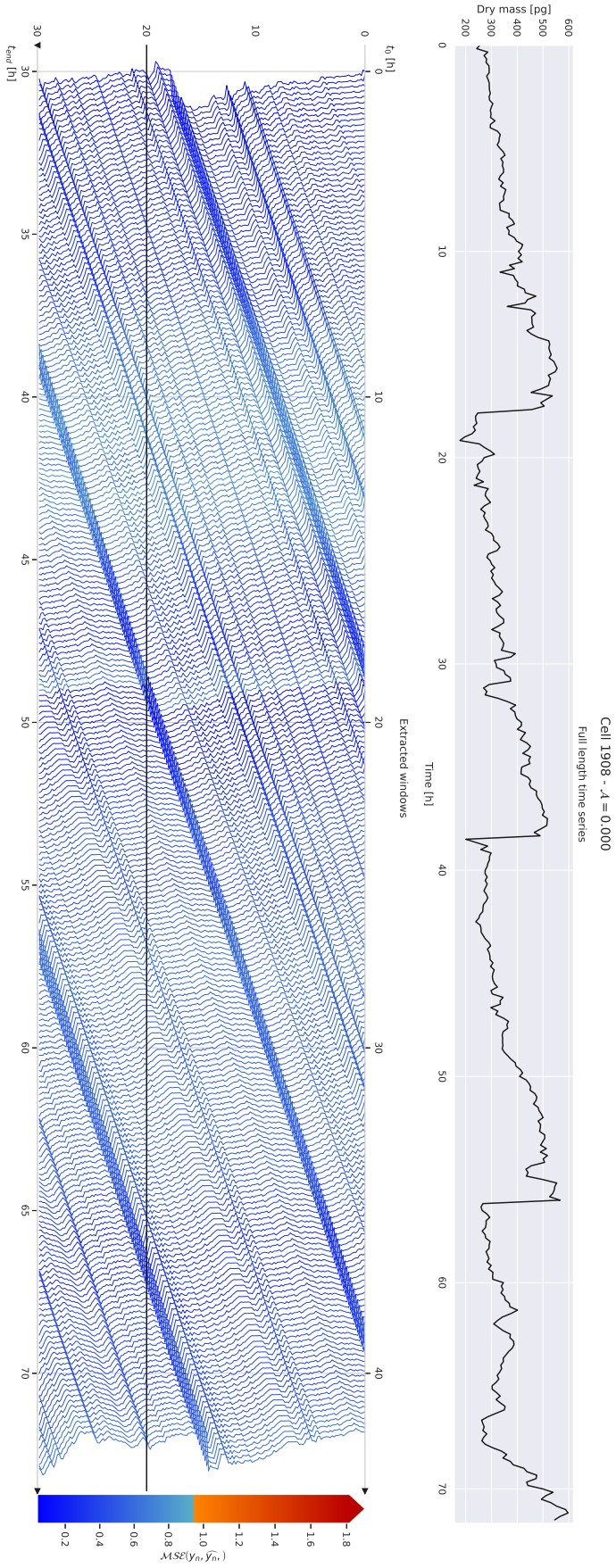
(a) Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule 13427



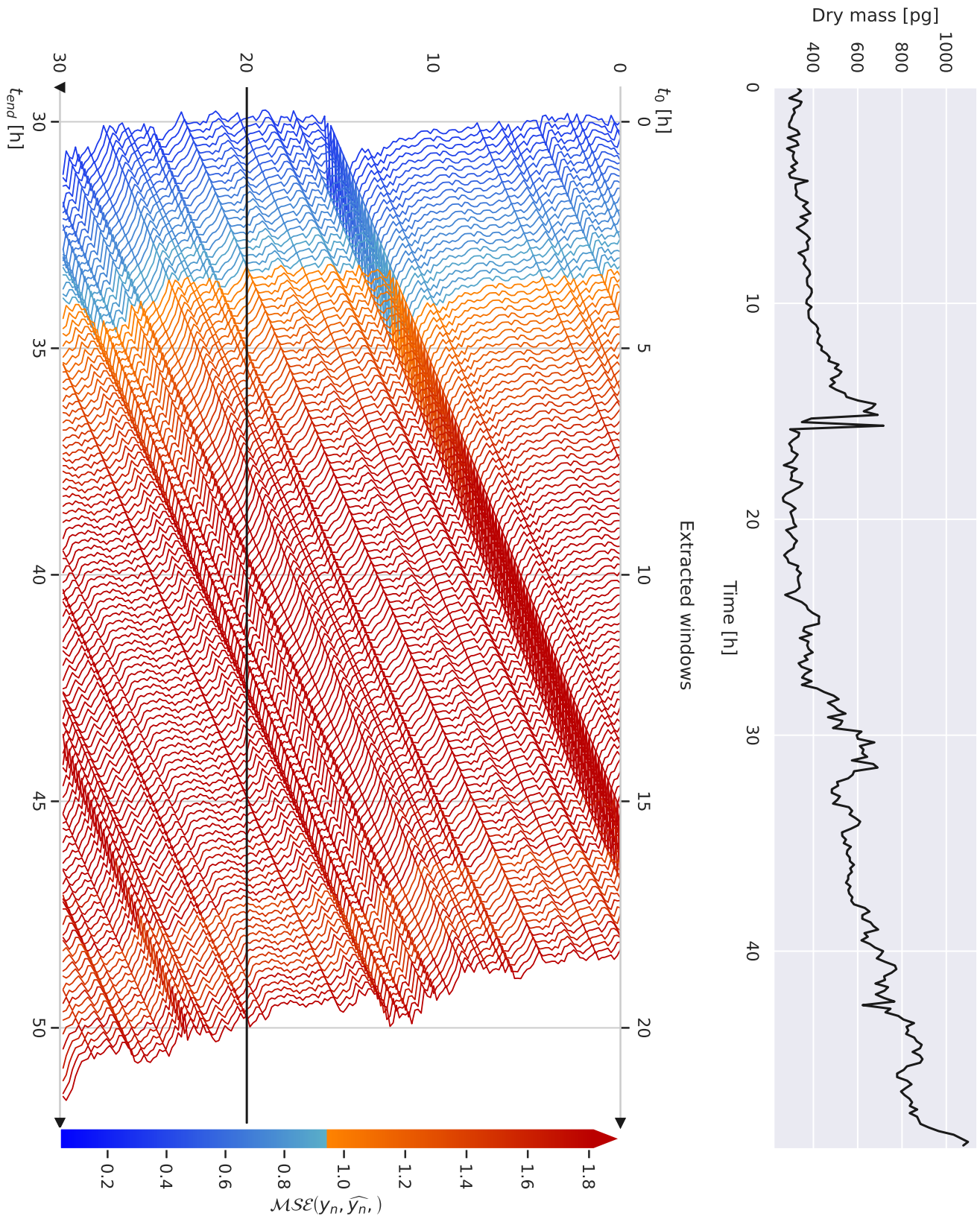
(b) Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule 1323



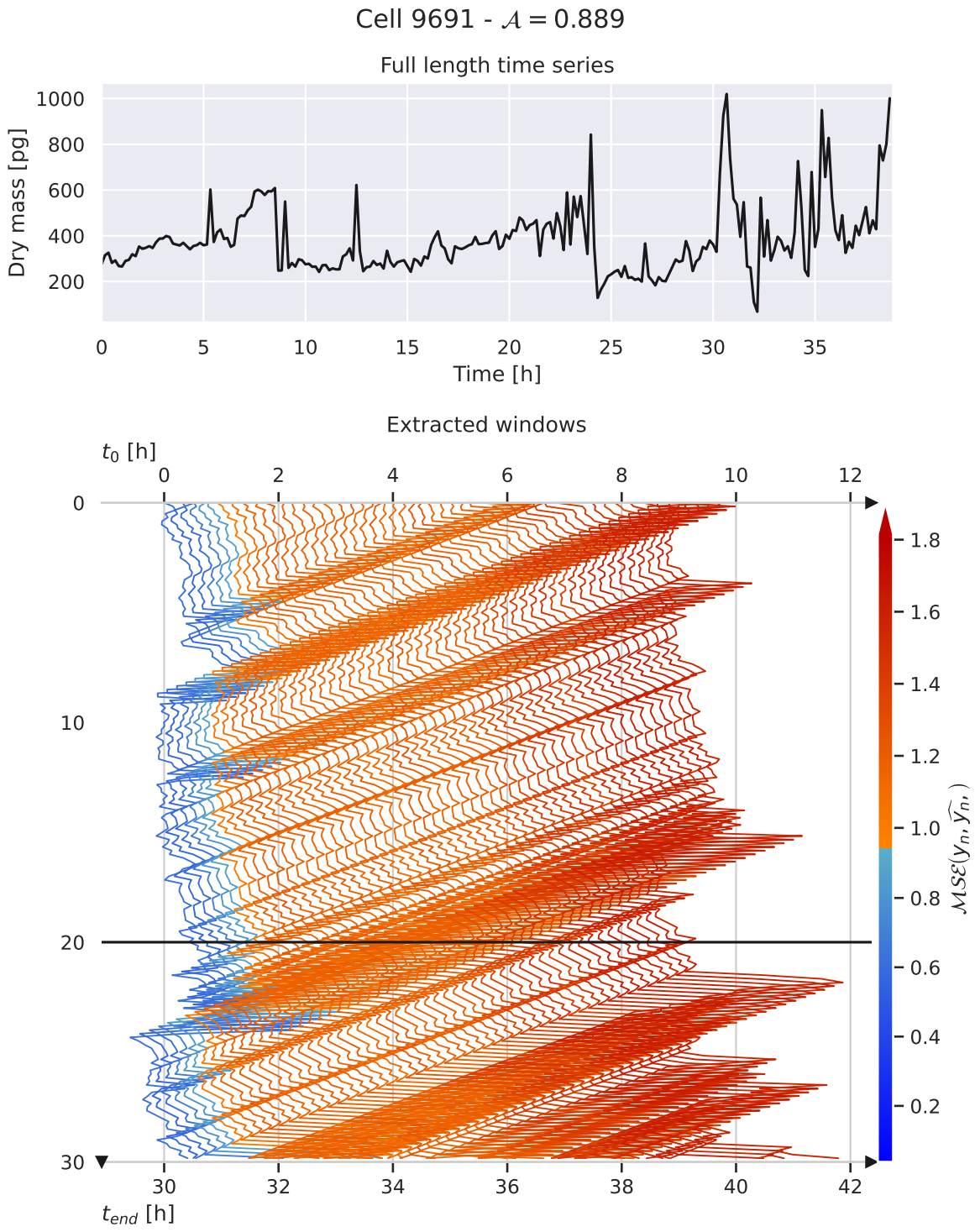
(c) Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule 1722



(d) Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule 1908



(e) Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule 2467



(f) Détection d'anomalies sur toutes les fenêtres de la série temporelle de la cellule 9691

Figure B.1 : Six exemples de détection d'anomalies sur toutes les fenêtres.

ANNEXE

C

Annexes à l'application sur les données sismiques de Mars

Table C.1 : Tableau récapitulatif des métriques obtenues sur tous les modèles d
d'anomalies pour toutes les représentations apprises

Rang	Augmentation	Features	Détecteur	Exactitude équilibrée
24	Blocage	64	Isolation Forest	0.502498
95	Bruit aléatoire	64	Isolation Forest	0.498187
5	Déformation de la temporalité	64	Isolation Forest	0.505152
26	Lissage aléatoire	64	Isolation Forest	0.502189
111	Lissage avec fenêtre glissante	64	Isolation Forest	0.491938
19	Permutations aléatoires	64	Isolation Forest	0.502963
18	Renversement bidirectionnel	64	Isolation Forest	0.502991
76	Retournement gauche-droite	64	Isolation Forest	0.499172
81	Rogner et redimensionner	64	Isolation Forest	0.498908
105	Blocage	128	Isolation Forest	0.495874
10	Bruit aléatoire	128	Isolation Forest	0.503962
9	Déformation de la temporalité	128	Isolation Forest	0.504180
75	Lissage aléatoire	128	Isolation Forest	0.499249
12	Lissage avec fenêtre glissante	128	Isolation Forest	0.503686
8	Permutations aléatoires	128	Isolation Forest	0.504346
22	Renversement bidirectionnel	128	Isolation Forest	0.502731
106	Retournement gauche-droite	128	Isolation Forest	0.495164
68	Rogner et redimensionner	128	Isolation Forest	0.499705
29	Blocage	256	Isolation Forest	0.501820
104	Bruit aléatoire	256	Isolation Forest	0.496514
50	Déformation de la temporalité	256	Isolation Forest	0.500376
11	Lissage aléatoire	256	Isolation Forest	0.503813
42	Lissage avec fenêtre glissante	256	Isolation Forest	0.500702
35	Permutations aléatoires	256	Isolation Forest	0.501113
16	Renversement bidirectionnel	256	Isolation Forest	0.503118
43	Retournement gauche-droite	256	Isolation Forest	0.500678
45	Rogner et redimensionner	256	Isolation Forest	0.500513
3	Blocage	4800	Isolation Forest	0.506017
4	Bruit aléatoire	4800	Isolation Forest	0.505550
107	Déformation de la temporalité	4800	Isolation Forest	0.494875
21	Lissage aléatoire	4800	Isolation Forest	0.502733

Suite page suivante

Rang	Augmentation	Features	Détecteur	Exactitude équilibrée
28	Lissage avec fenêtre glissante	4800	Isolation Forest	0.501878
57	Permutations aléatoires	4800	Isolation Forest	0.500051
63	Prediciton	4800	Isolation Forest	0.499897
98	Renversement bidirectionnel	4800	Isolation Forest	0.497655
55	Retournement gauche-droite	4800	Isolation Forest	0.500096
64	Rogner et redimensionner	4800	Isolation Forest	0.499897
87	Blocage	64	LOF	0.498641
82	Bruit aléatoire	64	LOF	0.498888
101	Déformation de la temporalité	64	LOF	0.497083
84	Lissage aléatoire	64	LOF	0.498785
15	Lissage avec fenêtre glissante	64	LOF	0.503202
46	Permutations aléatoires	64	LOF	0.500479
53	Renversement bidirectionnel	64	LOF	0.500143
69	Retournement gauche-droite	64	LOF	0.499665
33	Rogner et redimensionner	64	LOF	0.501225
47	Blocage	128	LOF	0.500479
71	Bruit aléatoire	128	LOF	0.499585
92	Déformation de la temporalité	128	LOF	0.498405
37	Lissage aléatoire	128	LOF	0.500935
80	Lissage avec fenêtre glissante	128	LOF	0.499010
56	Permutations aléatoires	128	LOF	0.500091
54	Renversement bidirectionnel	128	LOF	0.500105
83	Retournement gauche-droite	128	LOF	0.498857
36	Rogner et redimensionner	128	LOF	0.501054
38	Blocage	256	LOF	0.500826
73	Bruit aléatoire	256	LOF	0.499463
48	Déformation de la temporalité	256	LOF	0.500462
90	Lissage aléatoire	256	LOF	0.498442
86	Lissage avec fenêtre glissante	256	LOF	0.498660
51	Permutations aléatoires	256	LOF	0.500270
66	Renversement bidirectionnel	256	LOF	0.499825
61	Retournement gauche-droite	256	LOF	0.499931
41	Rogner et redimensionner	256	LOF	0.500767
62	Blocage	4800	LOF	0.499930
103	Bruit aléatoire	4800	LOF	0.496585
97	Déformation de la temporalité	4800	LOF	0.497794
44	Lissage aléatoire	4800	LOF	0.500651
100	Lissage avec fenêtre glissante	4800	LOF	0.497148
59	Permutations aléatoires	4800	LOF	0.500000
65	Prediciton	4800	LOF	0.499893
77	Renversement bidirectionnel	4800	LOF	0.499152
32	Retournement gauche-droite	4800	LOF	0.501552
74	Rogner et redimensionner	4800	LOF	0.499292
108	Blocage	64	OC-SVM	0.494362
89	Bruit aléatoire	64	OC-SVM	0.498626
25	Déformation de la temporalité	64	OC-SVM	0.502206
13	Lissage aléatoire	64	OC-SVM	0.503575
34	Lissage avec fenêtre glissante	64	OC-SVM	0.501186

Suite page suivante

Rang	Augmentation	Features	Détecteur	Exactitude équilibrée
102	Permutations aléatoires	64	OC-SVM	0.497003
93	Renversement bidirectionnel	64	OC-SVM	0.498313
49	Retournement gauche-droite	64	OC-SVM	0.500415
7	Rogner et redimensionner	64	OC-SVM	0.504578
14	Blocage	128	OC-SVM	0.503238
1	Bruit aléatoire	128	OC-SVM	0.506468
0	Déformation de la temporalité	128	OC-SVM	0.508079
78	Lissage aléatoire	128	OC-SVM	0.499123
17	Lissage avec fenêtre glissante	128	OC-SVM	0.503060
60	Permutations aléatoires	128	OC-SVM	0.500000
2	Renversement bidirectionnel	128	OC-SVM	0.506184
20	Retournement gauche-droite	128	OC-SVM	0.502759
67	Rogner et redimensionner	128	OC-SVM	0.499771
91	Blocage	256	OC-SVM	0.498416
70	Bruit aléatoire	256	OC-SVM	0.499643
110	Déformation de la temporalité	256	OC-SVM	0.492909
88	Lissage aléatoire	256	OC-SVM	0.498627
31	Lissage avec fenêtre glissante	256	OC-SVM	0.501603
40	Permutations aléatoires	256	OC-SVM	0.500767
52	Renversement bidirectionnel	256	OC-SVM	0.500238
85	Retournement gauche-droite	256	OC-SVM	0.498781
109	Rogner et redimensionner	256	OC-SVM	0.493864
6	Blocage	4800	OC-SVM	0.504985
27	Bruit aléatoire	4800	OC-SVM	0.501932
96	Déformation de la temporalité	4800	OC-SVM	0.497844
94	Lissage aléatoire	4800	OC-SVM	0.498187
99	Lissage avec fenêtre glissante	4800	OC-SVM	0.497257
58	Permutations aléatoires	4800	OC-SVM	0.500000
79	Prediciton	4800	OC-SVM	0.499020
39	Renversement bidirectionnel	4800	OC-SVM	0.500825
72	Retournement gauche-droite	4800	OC-SVM	0.499476
23	Rogner et redimensionner	4800	OC-SVM	0.502587
30	Prediciton	4800	Seuil	0.501612

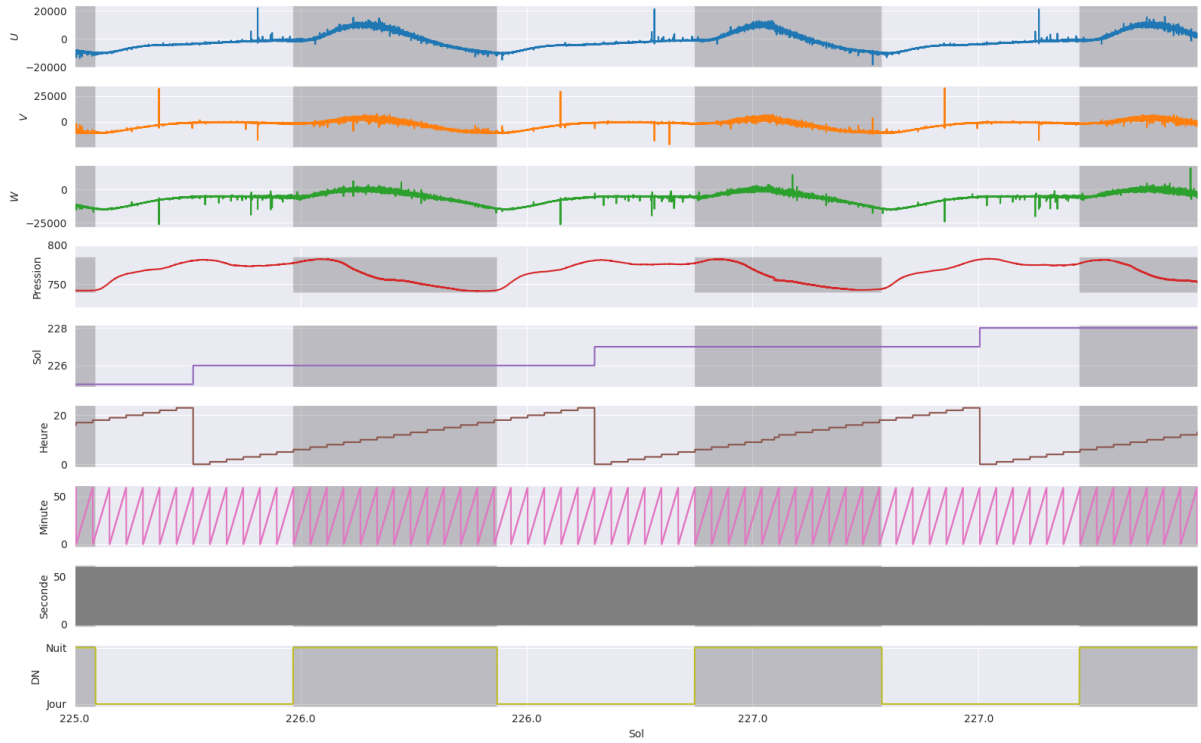


Figure C.1 : Jeu de données de test avec les 9 modalités disponibles une fois le jeu de données créé. Dans l'ordre de haut en bas, [U, V, W, Pression, Sol, Heure, Minute, Secondes, Jour/Nuit]

Résumé

Les capteurs intelligents sont devenus omniprésents dans notre société connectée, générant des quantités massives de données temporelles. La détection d'anomalies dans ces séries temporelles est essentielle pour garantir la fiabilité, la sécurité et les performances optimales des systèmes associés. Cependant, les méthodes traditionnelles de détection d'anomalies reposent souvent sur des techniques de supervision qui nécessitent des étiquettes d'anomalies préalablement annotées, ce qui peut être couteux et difficile à obtenir dans des scénarios réels.

Dans cette thèse, nous explorons l'utilisation de l'apprentissage auto-supervisé pour la détection d'anomalies dans les séries temporelles, en particulier dans le contexte des capteurs intelligents. L'apprentissage auto-supervisé permet d'exploiter les structures inhérentes aux données non étiquetées, en se basant sur des caractéristiques internes des données elles-mêmes. Nous proposons une méthodologie complète, nommée StArDusTS pour la détection automatique d'anomalies dans les séries temporelles grâce à des réseaux de neurones.

Nous évaluons notre approche sur plusieurs ensembles de jeux de données réels. Les cas applicatifs traités dans cette thèse couvrent deux domaines distincts, mais complémentaires. Tout d'abord, dans le contexte de la microscopie sans lentille, nous nous concentrons sur la détection de cellules anormales à partir de leur masse sèche. De plus, le second cadre applicatif concerne la détection d'événements sismiques sur Mars, où des capteurs sont déployés pour surveiller l'activité sismique. Ces deux cas applicatifs démontrent la polyvalence de l'apprentissage auto-supervisé pour la détection d'anomalies dans les séries temporelles.

Abstract

Smart sensors have become ubiquitous in our connected society, generating massive amounts of time series data. Detecting anomalies in these time series is essential to ensure the reliability, safety and optimum performance of the associated systems. However, traditional anomaly detection methods often rely on supervisory techniques that require previously annotated anomaly labels, which can be costly and difficult to obtain in real-life scenarios.

In this thesis, we explore the use of self-supervised learning for anomaly detection in time series, particularly in the context of smart sensors. Self-supervised learning makes it possible to exploit the structures inherent in unlabeled data, based on internal features of the data itself. We propose a comprehensive methodology, named StArDusTS for the automatic detection of anomalies in time series using neural networks.

We evaluate our approach on several real datasets. The application cases addressed in this thesis cover two distinct but complementary domains. Firstly, in the context of lens-free microscopy, we focus on the detection of abnormal cells based on their dry mass. In addition, the second application framework concerns the detection of seismic events on Mars, where sensors are deployed to monitor seismic activity. These two application cases demonstrate the versatility of self-supervised learning for the detection of anomalies in time series.