Repérez le piratage : Systèmes de détection d'intrusion pour réseaux avioniques en utilisant l'apprentissage automatique

Spot the hack: Intrusion detection systems for avionics networks using machine learning

Fehmi Jaafar[1], Florent Avellaneda[2], Omer Nguena Timo[3], Mario Beaulieu[4], David Landry[5]

[1] Département d'informatique et de mathématique, Université du Québec à Chicoutimi, Québec, Canada, fehmi.jaafar@uqac.ca
[2] Département d'informatique, Université du Québec à Montréal, Québec, Canada, avellaneda.florent@uqam.ca
[3] Département d'informatique et d'ingénierie, Université du Québec en Outaouais, Québec, Canada, omer.nguena-timo@uqo.ca
[4] Centre de Recherche Informatique de Montréal, Québec, Canada, mario.beaulieu@crim.ca
[5] Centre de Recherche Informatique de Montréal, Québec, Canada, david.landry@crim.ca

RÉSUMÉ. MIL-STD-1553B est une norme définissant un ensemble d'exigences qui couvrent tous les aspects d'un bus de données, des aspects mécaniques aux aspects électriques et fonctionnels. Le bus visait à interconnecter via un seul support les sous-systèmes avioniques. Plusieurs services et sous-traitants militaires ont adopté MIL-STD-1553 comme bus de données avionique en raison de son assurance d'intégrité des données. Cependant, de nouveaux travaux de recherche ont montré un ensemble de vulnérabilités de sécurité de ce bus. Cet article présente la création de Système de Détection d'Intrusions (SDI) efficace pour les réseaux avioniques et les technologies de bus utilisés dans l'industrie spatiale et aérospatiale.

ABSTRACT. The MIL-STD-1553B is a standard defining a set of requirements which cover all aspects of a serial multiplex data bus, from mechanical to electrical and functional aspects. The bus aimed to interconnect avionics subsystems via a single medium. Several military services and contractors adopted MIL-STD-1553 as an avionics data bus due to its data integrity insurance. However, new research work has demonstrated a set of security vulnerabilities for this bus. This paper presents the creation of effective intrusion detection systems (IDS) for avionics networks and bus technologies used within the aerospace and space industry.

MOTS-CLÉS. MIL-STD-1553, Système de Détection d'Intrusions, Réseaux Avioniques, Technologies de bus, LogBert.

KEYWORDS. MIL-STD-1553, Intrusion detection systems, Avionics networks, Bus technologies, LogBert.

Introduction

La norme militaire de bus MIL-STD-1553 est utilisé couramment dans des aéronefs et d'autres équipement d'utilisation critique pour échanger des renseignements avioniques, comme l'altitude, la position et la vitesse. Récemment, des recherches ont cerné des cyber attaque possibles susceptibles de permettre l'altération des données dans le bus MIL-STD-1553. L'apprentissage machine, et l'intelligence artificielle au sens plus large, permet entre autres de mieux reproduire le comportement des équipements et des systèmes, afin d'assurer une compréhension contextualisée des anomalies et la détection plus efficace des tentatives d'intrusions. Les Systèmes de Détection d'Intrusions (SDI) peuvent donc être plus efficaces lorsqu'ils sont infusés d'intelligence artificielle.

Dans cet article, nous présentons une solution qui utilise l'apprentissage machine et l'intelligence artificielle pour détecter ces cyber attaques. Notre solution consiste à construire un modèle/une représentation de comportements normaux dans un réseau et d'utiliser ce modèle pour la détection des anomalies causées par des intrusions dans un bus MIL-STD-1553. Nous avons construit un

simulateur pour générer les données et nous nous sommes intéressés à la détection des anomalies qui modifient la logique du fonctionnement du bus MIL-STD-1553.

Afin de traiter le cas de modèles très volumineux, nous étudions dans la section 2 l'approche d'inférence de modèles par réseau de neurones. En se contentant de modèles opaques et approximant le modèle à inférer, les réseaux de neurones permettent par nature d'inférer des modèles complexes pouvant contenir plusieurs millions d'états. Nous avons développé un simulateur pour générer des données de simulation selon la norme militaire MIL-STD-1553 contenant à la fois des messages normaux et anormaux. Nous avons concentré nos expérimentations sur les modèles de réseaux de neurones de type transformeur. Notre étude empirique montre que, bien que ces modèles par nature ne permettent pas d'obtenir des précisions de 100/100, elle nous permet d'obtenir des précisions très prometteuses à partir d'une seule séquence d'apprentissage.

Le reste de cet article se présente comme suit. Dans la section 1., nous présentons brièvement la norme MIL-STD-1553, les intrusions et nous effectuons une mini revue de littérature de la détection d'intrusions. Dans la section 2., nous présentons les trois principales activités de notre approche de détection des intrusions. Le simulateur que nous avons développé pour générer des données est présenté dans la section 3. La section 4. présente un cas d'étude que nous avons construit à des fins d'expérimentation. Nous présentons ainsi les résultats expérimentaux notre approche. La conclusion de cette article introduit également nos travaux futurs.

1. État de l'art

1.1. *Étude de la norme*

MIL-STD-1553 est un protocole de communication qui définit les caractéristiques mécaniques, électriques, et fonctionnelles d'un bus de communication en série. Ce protocole est utilisé entre autres en avionique militaire et spatiale pour permettre le transfert de données entre des composants ou sous systèmes d'un système avionique. Dans le cadre de nos travaux sur la détection d'intrusions, nous nous intéressons en particulier aux caractéristiques fonctionnelles de la norme.

En effet, le protocole MIL-STD-1553 spécifie un réseau de communication qui utilise un bus commun pour connecter tous les périphériques et contrôle les communications entre ces périphériques suivant un ordonnancement déterministe strict. Pour accroître la fiabilité de ce protocole, plusieurs bus peuvent être ajoutés afin d'assurer la redondance des messages qui circulent entre les périphériques.

Un réseau qui utilise une adoption du protocole MIL-STD-1553 est identifié comme étant un réseau MIL-STD-1553. Chaque appareil participant au réseau MIL-STD-1553 est identifié comme étant des terminaux distants (Remote Terminal ou RT) ou un Bus Contrôleur (BC). Le réseau MIL-STD-1553 attribue une adresse unique entre 1 et 31 à chaque RT participant. Un contrôleur de bus (Bus Contrôleur ou BC) assure les communications entre les RT selon un ordonnancement déterministe strict et un protocole de style Commande/Réponse.

Selon ce protocole, les messages communiqués sont spécifiés comme une série de mots de 20 bits (16 bits + 3 bits de synchronisation + 1 bit de parité) commençant par un mot de commande qui est utilisé par le BC pour émettre une commande à un RT [LSR]. Ce protocole accepte une commande de transmission de données (le RT concerné enverra des données à un autre RT), une commande de réception de données (le RT concerné recevra des données d'un autre RT), ou une autre commande d'une liste de commandes spécifiées. Le mot de commande comprend un champ d'adressage pour indiquer le RT concerné. Ce champ permet d'adresser le mot à un terminal en particulier en utilisant cinq bits qui permettent 32 adresses, cependant l'adresse « 11111 » est réservée pour contacter l'ensemble des terminaux (« broadcast »), c'est pour cela qu'il ne peut y avoir que 31

terminaux connectés au bus. Le mot de commande peut être des données ou des modes codes. Ces derniers permettent de contrôler les terminaux. Par exemple, le BC peut envoyer un mode code à un terminal pour qu'il lui ordonne de se synchroniser (reset de l'horloge interne et/ou lancement d'une séquence de synchronisation) et de répondre par un mot de statut. Comme un autre exemple, BC peut envoyer un mot de commande de réception à un terminal pour l'informer qu'il va commencer à recevoir des données.

Le RT doit répondre aux commandes provenant du BC en utilisant le mot d'état. Ce mot spécifique peut indiquer l'état du RT ou des informations d'erreur à l'aide de divers indicateurs. Pour reconnaître le RT qui partage son état, le mot d'état contient un champ d'adressage.

Ce protocole code les messages communiqués en utilisant le schéma de codage Manchester qui est une modulation par décalage de phase binaire (BPSK) qui assure la synchronisation d'horloge en utilisant une violation du code Manchester sur les premiers 3 bits du mot envoyé.

Le protocole MIL-STD-1553 contient un certain nombre d'exigences strictes de synchronisation que le BC doit suivre :

a. Temps de réponse : lors de la transmission d'un mot de commande à un RT, le BC s'attend à recevoir une réponse dans les 4 à 12 µs. Une réponse reçue entre 12 et 14 µs générera une erreur de « réponse tardive ». Après 14 µs, le BC expirera et générera une erreur « pas de réponse » si une réponse du RT n'a pas été reçue.

b. Mots de données multiples : les mots de données dans un message doivent être transmis consécutivement et en continu sans aucun intervalle de temps entre eux.

c. Écart entre les messages : le BC doit maintenir un intervalle de 4 µs après la fin de chaque message avant de commencer à transmettre le message suivant dans son programme.

Pour planifier l'envoie des mots aux différents RT en respectant les temps de réponse spécifiés par le protocole, le BC groupe les mots dans des fenêtres de temps :

1. Les mots sont d'abord regroupés dans des fenêtres de temps mineurs.

2. Les fenêtres de temps mineures sont ensuite regroupées en fenêtres de temps majeures.

3. Chaque fenêtre de temps majeure se voit alors assigner une fréquence pour transmettre les mots un à la fois.

4. Certains événements peuvent également générer du trafic apériodique sur le bus. Le BC gérera le trafic apériodique en l'ajoutant à la fin du prochain message mineur dans le programme qui a de l'espace disponible.

1.2. *Cyberattaque, intrusions et vulnérabilités*

Le protocole MIL-STD-1553 a été conçu dans les années 70 pour garantir un niveau élevé de tolérance aux pannes, tandis que moins d'attention a été accordée à la sécurité [De +21]. Des études récentes déjà abordées, l'impact des cyberattaques réussies sur les véhicules aérospatiaux qui mettent en œuvre MIL-STD-1553 [De +21] [Sta+19]. Dans cette section, nous présentons une analyse de sécurité de MIL- STD-1553.

Une cyberattaque peut cibler tous les composants du bus MIL-STD-1553 et en particulier le BC ou le RT qui implémentent les fonctionnalités d'un système avionique. Elle s'opère par l'intrusion d'artefacts malveillants dans les composants du réseau MIL-STD-1553. Il peut s'agir de l'intrusion

de malware, de code malveillant dans un logiciel, l'intrusion de données incorrectes ou l'intrusion de composants électroniques dans le but d'altérer des caractéristiques physiques du bus.

Une intrusion se manifeste par une modification de comportements des composants attaqués, ce qui impacte également le comportement du système avionique. D'une part, une cyberattaque peut modifier les règles temporelles du protocole comme le cas d'une attaque par Déni de service (DoS) qui menace la disponibilité de la bande communication (les communications entre terminaux deviennent bloquées par l'injection aléatoire de messages qui provoqueront des collisions sur le BUS.)

D'autre part, une cyberattaque peut altérer l'intégrité des données comme le cas d'une attaque d'usurpation d'identité. Dans le contexte de la sécurité des réseaux, une attaque par usurpation d'identité est une situation dans laquelle une personne ou un programme malveillant s'identifie avec succès comme un autre en falsifiant des données, pour obtenir un avantage illégitime.

Dans un réseau MIL-STD-1553, l'attaque par usurpation d'identité consiste à modifier l'adresse source dans l'en-tête des mots de données pour faire croire au RT récepteur ou au BC que le mot de données provient d'une source fiable, comme un autre RT sur le réseau, et l'accepter. Récemment, De Santo et al. [De +21] ont présentée dans étude sur les vulnérabilités du protocole MIL-STD-1553 et ont déployé un outil logiciel capable d'effectuer des cyberattaques d'usurpation d'identité. En effet, leur outil montre qu'un composant non autorisé avec un accès physique au bus 1553 peut détecter un temps d'inactivité sur le bus (c'est-à-dire lorsque ni le BC, ni les RT ne transmettent des données) et lancent un message échange dans cet intervalle. Tout autre composant supposera que la communication sur le bus a été initiée par le BC légitime. Par conséquent, le composant non autorisé pourrait fournir aux terminaux des informations malveillantes. Cette attaque peut être le résultat d'une intrusion dans le réseau pour injecter de code malveillant ou de données incorrectes dans les mots de données à l'aide de l'une des techniques suivantes :

1. Injection et manipulation de code : Cela fait référence à la capacité d'injecter ou de manipuler le code des composants du système afin d'effectuer l'attaque. Cela inclut le programme codé dans l'émetteur-récepteur, ainsi que le système d'exploitation ou le logiciel d'un sous-système. Étape a : Un code malveillant peut-être injecté au cours des principales phases du cycle de vie du composant (développement, chaîne d'approvisionnement, déploiement et maintenance). Étape b : Un attaquant peut injecter du code programmé pour fonctionner dans un contexte spécifique et peut identifier lorsqu'il se trouve dans l'environnement réel, échappant ainsi aux outils de détection.

2. Injection de fausses données et violation de l'intégrité des données : l'injection de données fait référence à de fausses données fournies par des capteurs (les systèmes de positionnement global (GPS), systèmes de détection et de télémétrie radio (RADAR), etc.) ou des appareils non-autorisés qui communiquent avec le réseau.

3. Falsification physique : une personne malveillante qui a un accès physique au système peut apporter des modifications subtiles au système, comme l'ajout de capacités de calcul à un coupleur, la manipulation du câblage ou la mise à la terre du coupleur. De telles modifications peuvent ne pas changer le comportement du composant de manière significative, mais créent un certain type d'effet secondaire (comme un rayonnement électromagnétique amplifié), qui peut également passer inaperçu si le système n'est pas spécifiquement testé pour ces effets secondaires spécifiques.

1.3. *Revue de la littérature*

Dans cette section, nous présentons les approches présentées pour faire face aux vulnérabilités du protocole MIL-STD-1553.

Mohan et al. [Moh+13] ont développé une architecture de détection d'intrusions basée sur l'analyse de temps de communication appelée Architecture de Système Simplexe Sécurisé (S3A). S3A repose sur le concept de systèmes de contrôle en temps réel ayant un comportement d'exécution déterministe. Ce concept suppose que le système consiste en un ensemble de tâches périodiques en temps réel avec des contraintes de temps et de délai gérées par un contrôleur en temps réel. L'architecture proposée par Mohan et al. [Moh+13] surveille l'unité centrale de traitement (CPU) pour identifier toute variation significative dans le temps de communication. Toute variation supérieure à une valeur de seuil prédéfinie de 0,6 µs est signalée comme étant une éventuelle intrusion.

Yoon et al. [Yoo+13] ont développé une approche qui utilise l'apprentissage statistique pour effectuer la détection d'intrusions sur des systèmes embarqués multicœurs en temps réel. L'approche proposée, SecureCore crée un profil des délais d'exécution qu'il compare ensuite aux informations de synchronisation surveillées au moment de l'exécution.

Loisier [LSR] a proposé une approche pour détecter les intrusions potentielles sur le MILSTD-1553 en examinant le trafic et les profils du trafic durant des périodes sélectionnées aléatoirement et en les comparant au profil de base. Un pourcentage de différence entre ces profils déclenchera une alerte de détection d'anomalies indiquant une intrusion potentielle.

He et al. [He+19] ont présenté plusieurs approches de mise en œuvre de simulateurs pour tester la sécurité des systèmes électroniques intégrés et présenter une mise en œuvre spécifique d'un simulateur de bus 1553. En outre, les auteurs ont discuté des scénarios d'intégrité et d'attaque DoS.

Stan et al. [Sta+17] ont suggéré d'améliorer la sécurité du réseau MIL-STD-1553 en intégrant un système de détection d'intrusions (IDS) basé sur l'apprentissage automatique. L'IDS se compose de deux modules : (1) un module d'authentification de terminal distant qui détecte les composants connectés illégalement et les transferts de données, et (2) un module de détection d'anomalies qui détecte les anomalies dans le fonctionnement du réseau. Comme une preuve de concept, les auteurs ont évalué le système de détection d'intrusions avec des données synthétiques de simulations d'attaques d'usurpation d'identité et des attaques DoS. Les résultats des expériences ont montré que leur système peut faire la distinction entre les messages anormaux et légitimes dans un réseau MIL-STD-1553.

Finalement, les modèles à états-transitions et les automates ont été grandement utilisés pour modéliser les protocoles de communication. Des travaux comme [Cho+10] apprennent des modèles formels de protocoles et inspirent nos contributions. Nous discutons plus en détail de ces travaux dans les prochaines sections.

2. La détection des intrusions

Compte tenu du fait que les intrusions se manifestent par un changement de comportements des composants du réseau et donc un changement des fonctionnalités du système, notre approche consiste à identifier des comportements anormaux.

Afin d'identifier les comportements anormaux, nous allons équiper le réseau d'un composant « oracle » capable d'observer toutes les communications qui s'effectuent dans le réseau, les comparer avec les communications attendues et générer une alerte lorsque les communications observées sont différentes de celles attendues.

Nous équipons donc le composant « oracle » avec une connaissance du comportement normal du réseau, c'est-à-dire, des communications attendues. Nous représentons le comportement normal du réseau avec un modèle. Nous construisons automatiquement le modèle à partir des observations

normales préalablement faites sur un réseau non corrompu, c'est-à-dire, qui n'a fait l'objet d'aucune intrusion.

Dans le cadre de ce travail, nous réalisons trois activités : (A1) nous collectons des observations normales d'un réseau, (A2) nous construisons le modèle du comportement normal du réseau à l'aide des approches d'inférence de modèles et d'apprentissage automatique, et (A3) nous présentons comment le composant « oracle » utilise le modèle pour détecter des intrusions. Les études dans [Cho+10] démontrent la viabilité d'une telle approche et la nécessité de prendre en compte les spécificités du système pour adapter les techniques d'apprentissage et permettre le passage à l'échelle.

3. Inférence de réseaux de neurones

Dans cette section, nous présentons une analyse des différents modèles d'apprentissage profonds que nous pouvons utiliser avec les données du simulateur.

3.1. *Analyse et simulation des données*

Nous avons développé un simulateur en langage Python qui génère les données qui circulent dans un bus MIL-STD-1553 pour la gestion des fonctions principales d'un train d'atterrissage. Ces fonctions incluent les manœuvres de roulage, de décollage, et d'atterrissage. Notre simulateur contient un contrôleur de bus (Bus Controller ou BC) et divers terminaux distants (Remote Terminal ou RT). Le BC assure les communications normales entre les RT selon un ordonnancement déterministe strict. Les terminaux distants simulent divers appareils embarqués tels qu'un train d'atterrissage, un altimètre, etc. En addition, les communications entre les composants du simulateur sont regroupées dans des fenêtres de temps mineures et des fenêtres de temps majeures. Selon le protocole MIL-STD-1553, cette spécification permet la transmission de messages cycliques et de messages acycliques dans le bus.

Nous avons simulé un scénario d'utilisation normal du bus (les fonctions principales d'un train d'atterrissage), ainsi qu'un scénario d'attaque par injection. Dans ce scénario, un RT hostile utilise des moments où le bus est inutilisé pour affecter le fonctionnement de l'appareil. Les messages injectés sont conformes au protocole MIL-STD-1553, mais créent un comportement anormal du train d'atterrissage, empêchant sont abaissement.

Dans la figure 1, nous présentons un exemple de données générées par notre simulateur. Ces données spécifient des séries temporelles univariées, puisque l'ensemble de données est composé d'une seule série d'observations pour chaque pas de temps avec un ordre temporel. Les données de séries temporelles multivariées désignent les données pour lesquelles il y a plus d'une observation pour chaque pas de temps.

Un modèle est nécessaire pour apprendre de la série d'observations passées pour prédire la valeur suivante de la séquence.

```
2021-05-21 10:46:04.722702 STATUS RT0 1000
2021-05-21 10:46:04.772408 CMD RT1 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:04.772728 STATUS RT1 1
2021-05-21 10:46:05.023109 CMD RT2 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:05.023541 STATUS RT2 167
2021-05-21 10:46:05.073299 CMD RT3 RECV SUB_DAT1 N2 -152
2021-05-21 10:46:05.073649 STATUS RT3 None
2021-05-21 10:46:05.223773 CMD RT2 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:05.224002 STATUS RT2 155
2021-05-21 10:46:05.273866 CMD RT3 RECV SUB_DAT1 N2 -737
2021-05-21 10:46:05.274044 STATUS RT3 None
2021-05-21 10:46:05.424281 CMD RT2 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:05.424370 STATUS RT2 148
2021-05-21 10:46:05.474434 CMD RT3 RECV SUB_DAT1 N2 256
2021-05-21 10:46:05.474655 STATUS RT3 None
2021-05-21 10:46:05.625023 CMD RT2 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:05.625270 STATUS RT2 74
2021-05-21 10:46:05.675096 CMD RT3 RECV SUB_DAT1 N2 -618
2021-05-21 10:46:05.675279 STATUS RT3 None
2021-05-21 10:46:05.725254 CMD RT0 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:05.725538 STATUS RT0 1000
2021-05-21 10:46:05.925999 CMD RT2 TRNSMT SUB_DAT1 N1 None
2021-05-21 10:46:05.926240 STATUS RT2 87
2021-05-21 10:46:05.976041 CMD RT3 RECV SUB_DAT1 N2 548
```

Figure 1. *Exemple de log produit par le simulateur*

3.2. *Modèles d'apprentissage profond*

Notre analyse des données a montré que le simulateur du protocole MIL-STD-1553 génère une série de données indexées dans l'ordre logique et temporel. Traditionnellement, l'inférence de ce type de données (les séries temporelles) a été dominée par des méthodes linéaires comme ARIMA (Autoregressive Integrated Moving Average) parce qu'elles sont simples et efficaces [MSR16].

En revanche, l'utilisation de ces méthodes classiques dans notre projet n'est pas adéquate puisqu'il est connu que ces méthodes souffrent de certaines limitations [Ste07], tel que la nécessité d'utiliser des données complètes (les données manquantes ou corrompues ne sont généralement pas prises en charge) et qui ont une tendance constante dans le temps. Cela est inadéquat avec la spécification du protocole MIL-STD-1553 qui permet l'insertion des données selon des fréquences asynchrones. En plus, nous insérons dans les données du simulateur des données qui reflètent une cyberattaque qui ont une tendance anormale.

En effet, les réseaux de neurones d'apprentissage profond sont capables d'apprendre automatiquement des relations complexes entre des entrées aux sorties et prennent en charge plusieurs entrées et sorties. Les réseaux de neurones que nous envisageons utiliser dans ce projet sont le perceptron multi-couches (Multilayer Perceptron ou MLP), le réseau de neurones à convolutions (Convolutionnal Neural Network ou CNN), le réseau de neurones récurrent (Recurrent Neural Network ou RNN) et le réseau transformeur (Tranformer Network).

Multilayer Perceptron (MLP)

Un MLP est structuré en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie selon un mode de propagation directe (feedforward).

Ce type de réseau de neurones peut être utile pour les séries temporelles en utilisant un réseau de neurones avec une couche cachée et en augmentant le nombre de neurones dans la couche, ou en augmentant le nombre de couches cachées.

Le modèle MLP apprendra une fonction qui projette une séquence d'observations passées vers une observation de sortie. En tant que tel, la séquence d'observations doit être transformée en plusieurs exemples dont le modèle peut apprendre.

L'un des principaux défis des réseaux de neurones MLP est la configuration des hyperparamètres suivants :

1. La profondeur du MLP (quantité de couches cachées du réseau) 2.La largeur du MLP (nombre de nœuds dans chaque couche) 3.La longueur de la fenêtre

4.Le taux d'apprentissage 5.La fonction d'optimisation

Ces paramètres peuvent être optimisés à l'aide de la recherche aléatoire ou de techniques plus avancées.

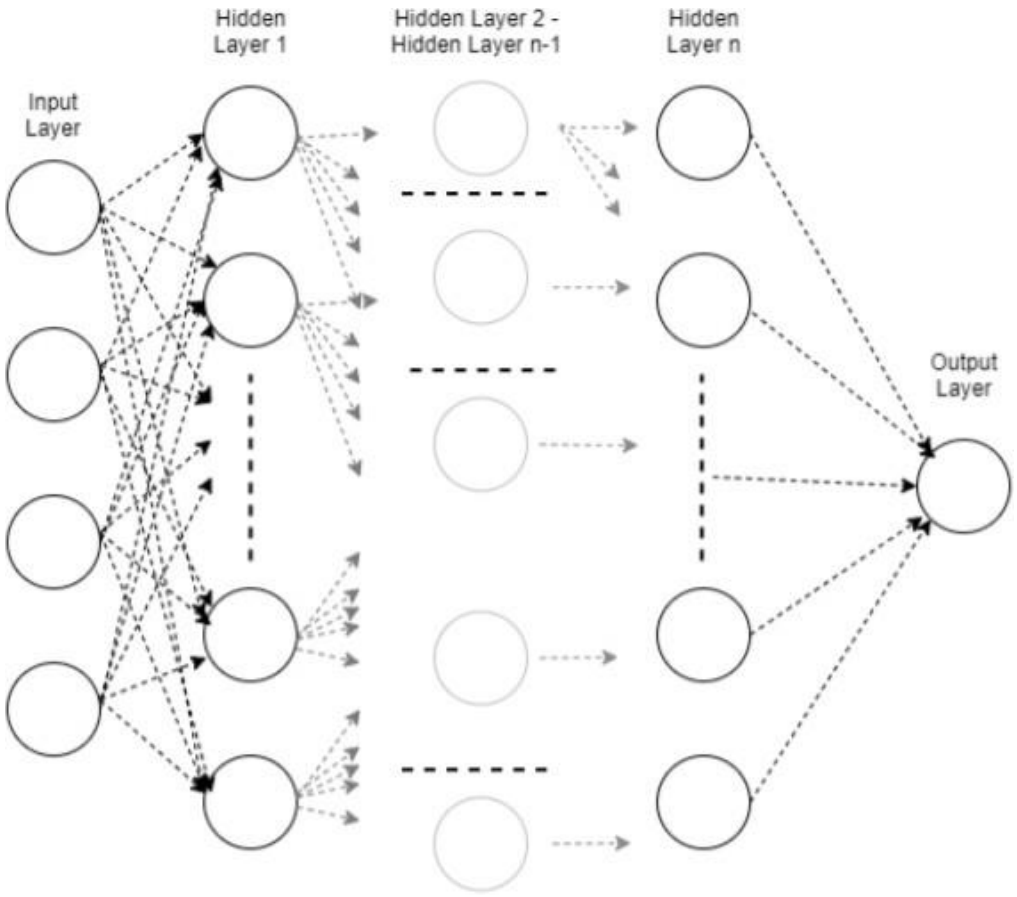

Figure 2. *Multilayer Perceptron (MLP) [BW20]*

Plusieurs chercheurs ont appliqué les réseaux de neurones MLP pour détecter des anomalies qui peuvent révéler des cyberattaques. Par exemple, Teoh et al. [Teo+18] ont montré l'efficacité du MLP à identifier des cyberattaques en analysant un ensemble de données volumineux contenant des informations d'attaques malveillantes. Plusieurs autres recherches [Gni20 ; WYO17] ont présenté des évaluations de la performance des réseaux de neurones MLP par rapport à d'autres modèles d'apprentissage profond. Ces études montrent la variation des performances d'identification des anomalies selon les données analysées.

Convolutional Neural Networks (CNN)

Les réseaux de neurones convolutifs ou CNN sont un type de réseau de neurones conçu pour gérer efficacement les données d'images. Ils se sont avérés efficaces sur des problèmes de vision par ordinateur complexes, à la fois en obtenant des résultats de pointe sur des tâches telles que la classification d'images et en fournissant un composant dans les modèles hybrides pour des problèmes entièrement nouveaux tels que la localisation d'objets, le sous-titrage d'images, etc.

Les réseaux de neurones convolutifs opèrent directement sur des données brutes, telles que des valeurs de pixels bruts. Le modèle apprend ensuite à extraire automatiquement les caractéristiques des données brutes qui sont directement utiles pour le problème traité.

C'est ce qu'on appelle l'apprentissage de la représentation. En utilisant un réseau de neurone convolutif, les caractéristiques sont extraites indépendamment de la façon dont elles se produisent dans les données, ce que l'on appelle la transformation ou invariance de distorsion.

Les réseaux de neurones convolutifs combinent trois idées architecturales pour assurer un certain degré d'invariance de décalage et de distorsion : l'utilisation d'un champ réceptif local, la spécification des poids partagés (ou réplication de poids) et, parfois, le sous-échantillonnage spatial ou temporel.

La capacité des CNN à apprendre et à extraire automatiquement des fonctionnalités à partir de données d'entrée brutes peut être appliquée aux problèmes de prédiction de séries temporelles. En effet, une séquence d'observations peut être traitée comme une image unidimensionnelle qu'un modèle CNN peut l'utiliser durant l'apprentissage. Cette capacité des réseaux de neurones convolutifs s'est avérée très efficace sur la classification des séries temporelles pour des tâches telles que la détection automatique des comportements [Xu+19] ou la détection des anomalies dans les réseaux informatiques [Hu+19].

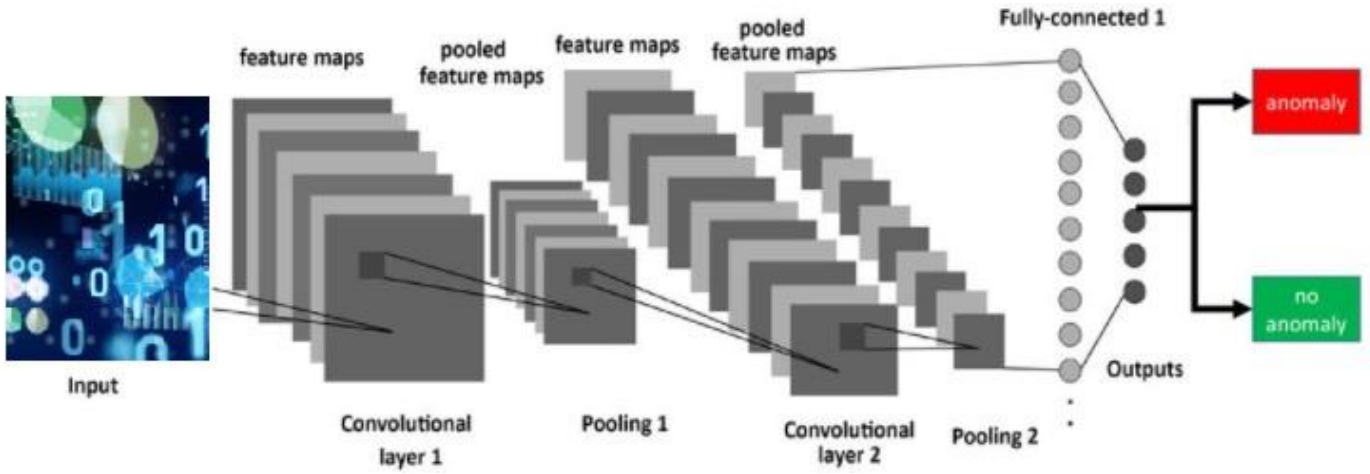

Figure 3. *Convolutional Neural Networks (CNN) [GL18]*

Recurrent Neural Networks (LSTM)

Un réseau de neurones récurrent comme le réseau de mémoire à long court terme ou LSTM est un type de réseau neuronal qui ajoute la gestion de l'ordre entre les observations lors de l'apprentissage des données d'entrée constituées de séquences d'observations. Cette notion qui n'est pas offerte par les MLP ou les CNN permet au réseau d'apprendre une fonction de mappage pour les entrées au fil du temps vers une sortie au lieu de mapper les entrées aux sorties uniquement.

Cette capacité des LSTM a montré son efficacité dans des problèmes complexes de traitement du langage naturel tels que la traduction automatique neuronale [Hua+18] où le modèle doit apprendre

les interrelations complexes entre les mots à la fois dans une langue donnée et entre les langues en traduisant d'une langue à une autre. Cette capacité peut être utilisée dans la détection des anomalies dans les séries temporelles [Sin17].

Outre les avantages généraux liés à l'utilisation de réseaux de neurones pour la prévision de séries temporelles, les réseaux de neurones récurrents peuvent également apprendre automatiquement la dépendance temporelle à partir des données [TL18]. Le contexte le plus pertinent des observations d'entrée par rapport à la sortie attendue est appris et peut changer dynamiquement. En effet, le LSTM apprend à la fois un mappage des entrées aux sorties et apprend quel contexte de la séquence d'entrée est utile pour le mappage, et peut changer dynamiquement ce contexte si nécessaire.

En raison de cette capacité à apprendre des corrélations à long terme dans une séquence, les réseaux LSTM évitent le besoin d'une fenêtre temporelle prédéfinie et sont capables de modéliser avec précision des séquences multivariées complexes [STN19].

Le modèle LSTM apprendra une fonction qui mappe une séquence d'observations passées comme entrée d'une observation de sortie. En tant que telle, la séquence d'observations doit être transformée en de multiples exemples dont le LSTM peut apprendre.

Contrairement à un CNN qui lit l'ensemble du vecteur d'entrée, le modèle LSTM lit un pas de temps de la séquence à la fois et construit une représentation d'état interne qui peut être utilisée comme contexte appris pour faire une prédiction.

Plusieurs couches LSTM cachées peuvent être empilées les unes sur les autres dans ce que l'on appelle un modèle LSTM empilé. Une couche LSTM nécessite une entrée tridimensionnelle et les LSTM par défaut produiront une sortie bidimensionnelle comme une interprétation à partir de la fin de la séquence.

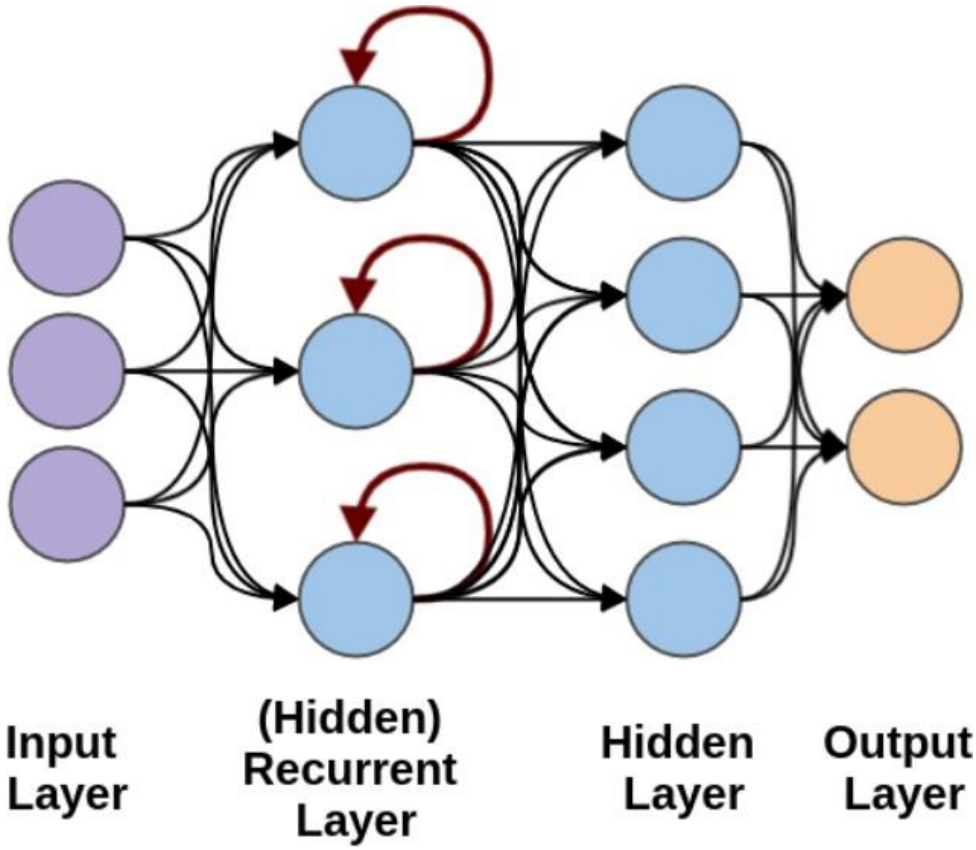

Figure 4. *Recurrent Neural Networks (LSTM) [Wal19]*

Bien que les réseaux de neurones MLP et CNN sont capables d'apprendre des relations arbitraires, l'ajout explicite de la prise en charge des séquences d'entrée dans les RNN offre une efficacité et de meilleures performances pour apprendre automatiquement les dépendances temporelles à la fois dans la séquence d'entrée et de la séquence de sortie. Ces réseaux peuvent également être combinés. En effet, les modèles hybrides combinent efficacement les diverses capacités de ces réseaux. Par exemple nous notons l'utilisation de modèles hybrides comme les CNN-LSTM [Gao+19] et les ConvLSTM [LLG17] qui cherchent à exploiter les capacités des trois types de modèles pour optimiser la détection des anomalies.

Traditionnellement, de nombreuses recherches ont été consacrées à l'utilisation des MLP pour la détection des anomalies de séries temporelles. Néanmoins, les résultats les plus prometteurs dans l'application des méthodes d'apprentissage profond à la détection des anomalies des séries temporelles est l'utilisation des CNN [Kwo+18], des LSTM [Cha+19] et des modèles hybrides [KKK+19].

Transformer Neural Network (BERT)

Un transformeur est un modèle d'apprentissage profond qui adopte le mécanisme de l'attention, pesant l'influence de différentes parties des données d'entrée. Il est principalement utilisé dans le domaine du traitement du langage naturel (NLP). À l'instar des RNN, les transformeurs sont conçus pour gérer des données d'entrée séquentielles, telles que le langage naturel, pour des tâches telles que la traduction et la synthèse de texte. Cependant, contrairement aux RNN, les transformeurs ne nécessitent pas que les données séquentielles soient traitées dans l'ordre.

En effet, le transformeur ne traite pas les données de manière ordonnée. Au lieu de cela, il traite toute la séquence de données et utilise des mécanismes d'attention pour apprendre les dépendances dans la séquence. Par conséquent, les modèles basés sur le transformeur ont le potentiel de modéliser des dynamiques complexes de données de séries chronologiques qui sont difficiles pour les modèles de séquence.

Le modèle transformeur basé sur le concept du codeur-décodeur pour mettre en oeuvre le mécanisme d'attention a été introduit en 2017 par Vaswani et al. [Vas+17]. L'un des encodeurs lit l'entrée de texte et le décodeur produit une prédiction pour la tâche. En 2018, Google a publié BERT (Bidirectional Encoder Representations from Transformers) [Dev+18] qui utilise un mécanisme d'attention qui apprend les relations contextuelles entre les mots (ou sous-mots) d'un texte. Contrairement aux modèles directionnels, qui lisent l'entrée de texte de manière séquentielle (de gauche à droite ou de droite à gauche), l'encodeur de BERT lit la séquence entière de mots à la fois. Par conséquent, il est considéré comme bidirectionnel. Cette caractéristique permet au modèle d'apprendre le contexte d'un mot en fonction de l'ensemble de son environnement (à gauche et à droite du mot).

Guo et al. [GYW21] ont proposé LogBERT, un framework pour la détection des anomalies basé sur BERT. LogBERT apprend les modèles de séquences de journaux normaux d'un système informatique pour détecter les anomalies lorsque les modèles sous-jacents s'écartent des séquences de journaux normales. Les résultats expérimentaux sur trois ensembles de données montrent que LogBERT surpasse les approches traditionnelles pour la détection des anomalies.

Dang et al. [Dan+20] ont proposé une nouvelle méthode basée sur BERT pour détecter les anomalies dans les données de surveillance. Contrairement aux propositions récentes, cette méthode utilise une petite quantité de données historiques étiquetées. Les résultats de la simulation démontrent que cette méthode n'a besoin que d'une petite quantité de données étiquetées pour entraîner le modèle afin de détecter correctement les anomalies.

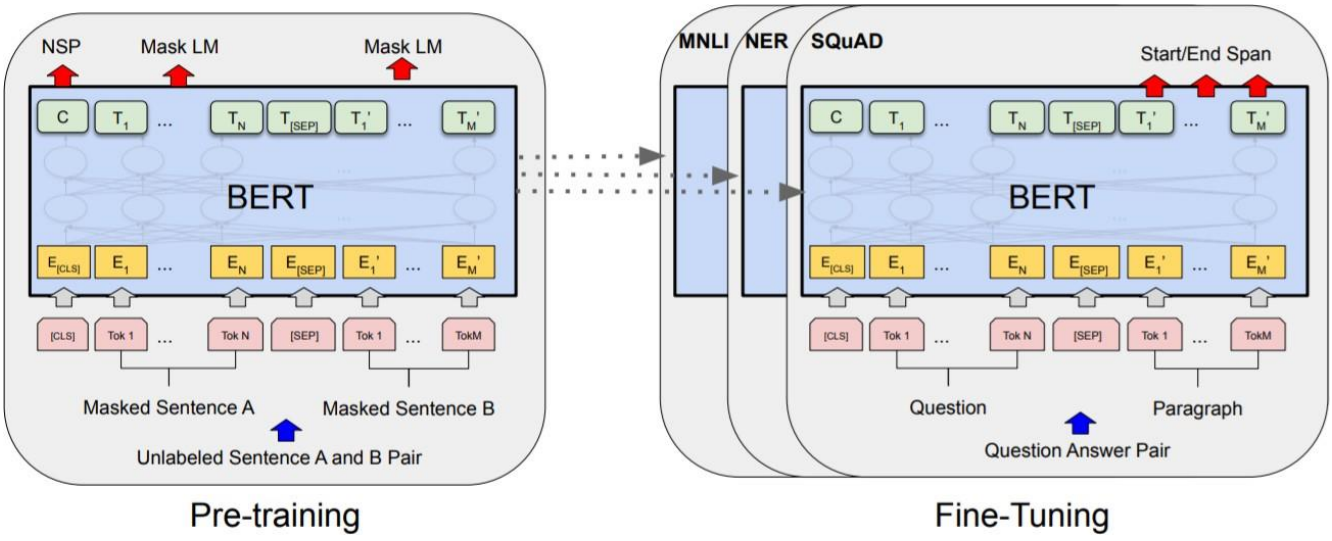

Figure 5. *Transformer Neural Network (BERT) [Dev+18]*

4. Expérimentations

4.1. *Expérimentations sur les réseaux de neurones*

Suivant notre évaluation des architectures applicables à la détection d'anomalies dans le bus de communication MIL-STD-1553, nous avons concentré nos expérimentations sur les modèles de type transformeur. Ces derniers nous ont semblé particulièrement prometteurs pour la tâche étant donné leur capacité accrue à prendre en considération le contexte lors de l'inférence. En effet, puisque les systèmes avioniques agissent en temps réel et fonctionnent de façon très prévisible, les modèles de type transformeur pourront constituer un modèle très précis du fonctionnement dans un contexte d'utilisation prévisible afin de reconnaître les activités anormales. Cela pourrait permettre par exemple, la détection d'anomalies dans les réseaux avioniques comme des sauts irréalistes dans la position GPS ou des communications inconsistantes provenant de différents capteurs. Cette section décrit notre approche expérimentale pour évaluer le potentiel des architectures de type transformeur pour la détection d'anomalies sur le bus MIL- STD-1553

Implémentation du modèle

Nous avons mené nos expérimentations en réutilisant l'architecture LogBERT[GYW21]. LogBERT est un modèle de type transformeur pouvant détecter les anomalies dans les journaux d'exécution (log) de systèmes informatiques. Cette problématique nous a semblé assez similaire au problème de détection d'anomalies dans le bus MIL-STD-1553 pour justifier une investigation approfondie. Le code source de LogBERT étant librement disponible, nous avons pu rapidement itérer sur la problématique, dans l'esprit du programme IDEeS.

LogBERT est une architecture de type transformeur entraînée à prédire des entrées masquées de journaux. Étant donné une entrée manquante dans un journal, elle cherche à prédire quelles entrées pourraient plausiblement trouver à la place de l'entrée manquante. Un tel modèle peut être utilisé pour la détection d'anomalies de la façon suivante : pour chaque entrée d'un journal, demander au modèle sa prédiction des entrées les plus probables. Si l'entrée que l'on trouve dans le journal et les sorties du modèle sont fortement différentes, alors nous sommes possiblement en présence d'une anomalie.

Le modèle choisi se distingue d'approches précédentes basées sur des RNN par sa meilleure capacité à capturer le contexte. Son mécanisme d'attention, issu de son architecture de transformeur, lui permet de combiner des entrées de journaux plus distantes dans le temps pour tirer des conclusions. Cette qualité de conception a largement fait ses preuves dans le domaine du traitement

des langues naturelles, et nous postulons qu'il pourra produire des résultats prometteurs à la détection d'anomalies dans le bus MIL- STD-1553.

Pour nos expérimentations, nous avons utilisé l'implémentation de LogBERT (dans la figure 6)[1]. Quelques modifications du code source ont été nécessaires pour adapter ce modèle à nos données de simulation. Étant donné que les séquences de messages du journal d'exécution sont non structurées, un analyseur de journaux d'exécution (*Drain* [He+17]) est appliqué comme prétraitement afin de les structurer et en extraire des gabarits, c'est-à-dire des modèles de chaînes de caractères. À chacun des gabarits, on associe une clé unique (voir figure 7). L'analyseur a produit ainsi un ensemble de 252 clés qui serviront de vocabulaire pour LogBERT. Par la suite, on définit une séquence S de messages du journal d'exécution comme un ensemble ordonné des clés k_t en fonction de la position t : $S = \{k_1, k_2, ...k_T\}$. L'objectif sera donc de prédire si une nouvelle séquence S est anormale par rapport à l'ensemble d'entraînement

$$D = \{S^j\}_{j=1}^{N}$$

qui ne comprend que des séquences normales.

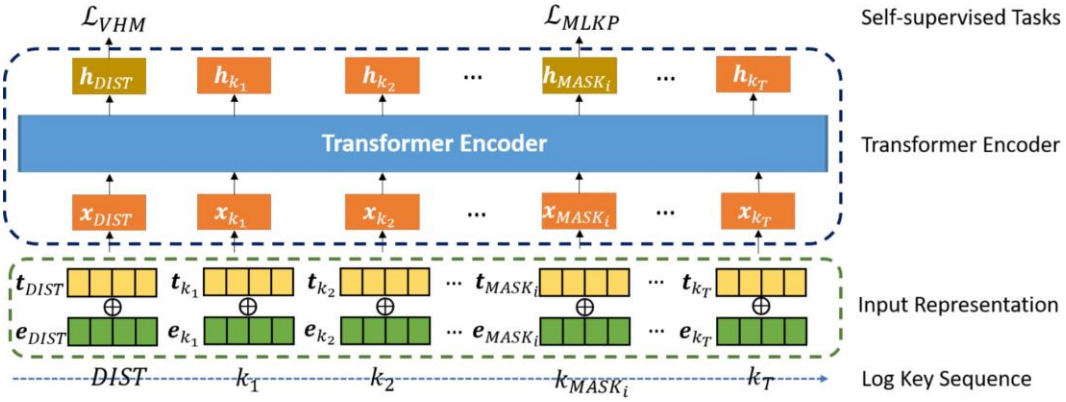

Figure 6. *Aperçu de logBERT [GYW21]*

Figure 7. *Aperçu du prétraitement des messages du journal d'exécution*

Jeux de données

Pour l'entraînement, nous avons généré 40 minutes de simulation contenant à la fois des messages normaux et anormaux. Les attaques par injection sont insérées de façon aléatoire pendant l'exécution du système, afin d'éviter du surapprentissage. La simulation inclut **47108 lignes dont 21 correspondent à des tentatives de cyberattaques**, soit 0.04% des messages. Ce qui constitue un très faible pourcentage d'anomalies si on le compare aux ensembles de données BGL et HDFS utilisés pour les tests de LogBert qui contiennent respectivement 2.5% et 7.3% d'anomalies.

[1]https ://github.com/HelenGuohx/logbert

# de messages normaux	# de messages anormaux	Total
47087	21	47108

Tableau 4.1. *Distribution des messages de la simulation*

La création des séquences de clés S est effectuée en appliquant un découpage du journal d'exécution à l'aide d'une fenêtre glissante temporelle de taille ΔT et de pas δ. Les messages qui se sont produits dans la même fenêtre glissante sont regroupés en tant que séquence de clés. Ainsi, les clés peuvent se dupliquer dans plusieurs fenêtres glissantes en raison du chevauchement si $\delta < \Delta T$. Ces séquences sont par la suite divisées en deux ensembles : 1) les séquences normales et 2) les séquences avec des messages anormaux. L'apprentissage du modèle est effectué à partir de 75% des séquences normales, tandis qu'au moment du test, le modèle est appliqué sur la sortie contenant des messages anormaux et les 25% de messages normaux restants.

Étant donné la petite taille du journal d'exécution simulé, la taille de la fenêtre temporelle ΔT a été fixée à 60 secondes et le pas d'échantillonnage δ à 1 seconde. Ce qui nous a fourni 2304 instances de séquences de longueur moyenne de 1190 clés. À cause de la répartition aléatoire des messages anormaux et du chevauchement dû au pas d'échantillonnage, les 21 messages anormaux du départ se sont répartis sur 958 séquences anormales qui seront utilisées durant la phase de test. Considérant les tailles respectives d'une séquence, de la fenêtre temporelle ΔT et du pas d'échantillonnage, un message anormal donné apparaîtra avec un décalage moyen de 20 messages entre deux séquences consécutives. Il est possible que deux messages anormaux ou plus soient intégrés à une séquence, ce qui aurait comme effet de réduire le nombre de séquences anormales. Cela implique que les messages anormaux seront entourés d'un contexte différent d'une fois à l'autre. Cette observation supporte notre choix d'utiliser des réseaux de neurones de type transformeur pour permettre au modèle d'apprendre le contexte d'un message en fonction de l'ensemble de son environnement

Ensuite, il nous reste 1346 séquences normales qui seront séparées en deux ensembles dans un rapport 3 :1, donc 1009 séquences normales pour l'apprentissage, et 337 séquences normales pour le test.

# séquences normales pour l'apprentissage	# séquences normales pour le test	# séquences anormales pour le test	Total
1009	337	958	2304

Tableau 4.2. *Distribution des séquences S pour la phase d'apprentissage et de test*

4.2. *Apprentissage*

Afin de capturer les informations contextuelles bidirectionnelles des séquences du journal d'exécution, LogBERT est entraîné à prédire les clés du journal d'exécution masquées dans les séquences normales du journal. LogBERT prend en entrée les séquences avec des masques, où un ratio de 50% des clés est remplacé aléatoirement par un token *MASK*. L'objectif de l'apprentissage est de prédire avec précision les clés masquées aléatoirement du journal. Le but est donc de faire en sorte que LogBERT encode la connaissance préalable des séquences normales du journal.

La représentation des données d'entrée pour l'apprentissage suit le formaliste de BERT. Au début d'une séquence $S = \{k^1, k^2, ...k^T\}$ on ajoute un token spécial *DIST* comme première clé. LogBERT encode

chaque clé k^t

comme la somme de deux embeddings : l'embedding de la clé t_j

t

et un embedding de

position e_j (voir figure 6).

t

L'apprentissage est effectué en minimisant la fonction de coût de l'entropie croisée pour la prédiction des clés masquées par rapport aux vraies clés. La taille du mini-lot d'apprentissage a été fixée à 32 items et l'algorithme d'optimisation utilisé est l'optimiseur ADAM avec ses paramètres par défaut. Nous avons effectués deux apprentissages avec LogBert : le premier initialise LogBERT avec quatre couches de transformeur avec des couches multi-attention à quatre entrées, le second nous avons augmenté le nombre d'entrées des couches multi-attention à huit.

La figure 8 montre l'évolution du score du modèle sur le jeu d'entraînement à travers les époques.

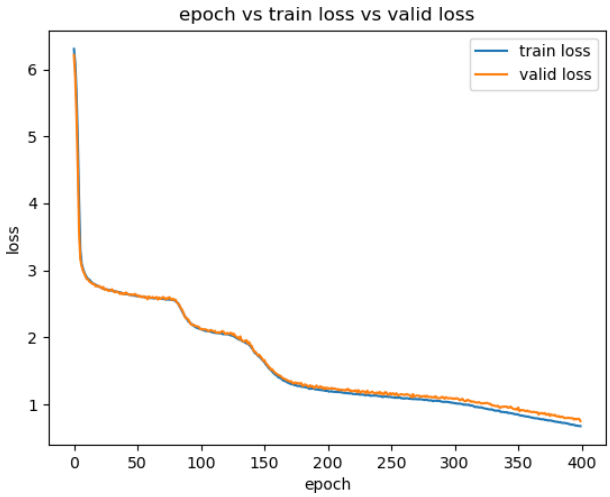

Figure 8. *Valeur de la Fonction de coût (training et validation) en fonction des itérations*

Résultats

Les tableaux 4.3 et 4.4 montrent les performances du modèle LogBERT lors de la détection d'anomalies dans notre système simulé. Contrairement aux systèmes basés sur l'inférence d'automates, notre approche basée sur les réseaux de neurones ne fait pas de supposition forte sur le système modélisé. Par conséquent, on peut supposer que des modèles similaires à LogBERT ont une meilleure chance de pouvoir s'appliquer à des systèmes réels.

Dans ce contexte, le modèle LogBERT affiche des performances prometteuses lors de la détection d'anomalies de notre jeu de données simulées. Le modèle à quatre entrées pour les couches multi-attention (voir tableau 4.3) a un rappel égal à 100% mais une précision inférieure, indiquant une légère tendance à signaler de faux positifs. Généralement, un taux de faux positifs trop élevé compromet l'utilité du système de cybersécurité, car il accroît le risque de voir les alertes ignorées par les opérateurs, ce qui ne semble pas être notre cas comme le montre les résultats. Tandis que le second modèle (voir tableau 4.3) est plus précis, mais laisse passer plus des détections.

Notre étude expérimentale avec des données simulées d'un réseau qui utilise la norme militaire de bus MIL-STD-1553 indique qu'en utilisant des modèles d'entraînement autosupervisées, LogBERT peut modéliser avec succès les séquences de journaux normales et identifier davantage les séquences anormales avec une grande précision. En effet, les solutions proposées dans notre étude peuvent être adaptées à d'autres systèmes ou protocoles aérospatiaux de même nature pour surveiller et analyser le trafic du bus pendant qu'il fonctionne et de détecter les anomalies tout en minimisant les taux de faux résultats positifs.

Vrais positifs	Vrais négatifs	Faux positifs	Faux négatifs
958	234	103	0
Precision	**Recall**	**F1 Score**	
90.29%	100%	94.9%	

Tableau 4.3. *Résultat de LogBERT avec quatre entrées (heads) pour la couche multi-attention*

Vrais positifs	Vrais négatifs	Faux positifs	Faux négatifs
916	331	6	42
Precision	**Recall**	**F1 Score**	
99.35%	95.62%	97.45%	

Tableau 4.4. *Résultat de LogBERT avec huit entrées (heads) pour la couche multi-attention*

Conclusion

Dans cet article, nous avons proposé une méthode à base de réseaux de neurones pour inférer des modèles approximant le système à inférer. Après l'étude de différentes options architecturales, nous avons utilisé LogBERT pour la détection d'anomalies sur un bus MIL-STD-1553 simulé. LogBERT a montré des performances prometteuses sur les traces d'exécutions qui lui ont été fournies. De plus, comme elle ne fait aucune supposition sur la nature du système simulé, LogBERT présente un bon potentiel de passage à l'échelle vers des systèmes réels. Par conséquent, une extension naturelle de nos travaux est d'étudier l'applicabilité de LogBERT (ou autre architecture de type transformeur) sur des données provenant d'un système MIL-STD-1553 réel.

Nous souhaitons poursuivre les efforts de recherche et de développement relativement aux solutions scientifiques élaborées dans la présente composante. Par exemple, nous souhaitons améliorer LogBERT en utilisant des données d'apprentissage provenant d'un système MIL-STD-1553 réel, de sorte qu'elles atteignent un niveau de maturité supérieur.

Bibliographie

[BW20] BRAEI M., WAGNER, S., "Anomaly detection in univariate time-series : A survey on the state-of-the-art". In : *arXiv preprint arXiv :2004.00433* (2020).

[Cha+19] CHAWLA A., et al., "Bidirectional LSTM autoencoder for sequence based anomaly detection in cyber security." In: *International Journal of Simulation–Systems, Science & Technology* (2019).

[Cho+10] CHO C.W., et al. "Inference and analysis of formal models of botnet command and control protocols". In : *Proceedings of the 17th ACM Conference on Computer and Com- munications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. Sous la dir. d'Ehab AL-SHAER, Angelos D. KEROMYTIS et Vitaly SHMATIKOV. ACM, 2010, p. 426- 439. DOI : 10.1145/1866307.1866355. URL : https://doi.org/10.1145/1866307.1866355.

[Dan+20] DANG W., et al., "Time Series Anomaly Detection Based on Language Model". In : *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*. 2020, p. 544-547.

[De +21] DE SANTO D., et al., "Exploiting the MIL-STD-1553 avionic data bus with an active cyber device". In : *Computers & Security* 100 (2021), p. 102097.

[Dev+18] DEVLIN J., *et al.*, "Bert : Pre-training of deep bidirectional transformers for language understanding". In : *arXiv preprint arXiv :1810.04805* (2018).

[Gao+19] GAO J., et al., "Abnormal gait recognition algorithm based on lstm-cnn fusion network". In : *IEEE Access* 7 (2019), p. 163180-163190.

[GL18] GOOSSENS A., Van LOO, J.P., "Applying Neural Networks to Anomaly De- tection". In : *Data Innovation Summit*. 2018.

[Gni20] GNIEWKOWSKI M., "An overview of DoS and DDoS attack detection techniques". In: *International Conference on Dependability and Complex Systems*. Springer. 2020, p. 233-241.

[GYW21] GUO S., YUAN S., WU X., *LogBERT : Log Anomaly Detection via BERT*.

2021. arXiv : 2103.04475[cs.CR].

[He+17] HE P., et al., "Drain : An Online Log Parsing Approach with Fixed Depth Tree". In : *2017 IEEE International Conference on Web Services (ICWS)*. 2017, p. 33-40. DOI : 10. 1109/ICWS.2017.13.

[He+19] HE D., et al., "Simulation design for security testing of integrated electronic systems". In: *IEEE Network* 34.1 (2019), p. 159-165.

[Hu+19] Hu W., et al., "Network data analysis and anomaly detection using CNN technique for industrial control systems security". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE. 2019, p. 593-597.

[Hua+18] HUANG X., et al., "A LSTM-based bidirectional translation model for optimizing rare words and terminologies". In: *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE. 2018, p. 185-189.

[KKK+19] KHAN M.A.,, KARIM M., KIM Y., et al., "A scalable and hybrid in- trusion detection system based on the convolutional-LSTM network". In : *Symmetry* 11.4 (2019), p. 583.

[Kwo+18] KWON D., et al., "An empirical study on network anomaly detection using convo- lutional neural networks". In : *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, p. 1595-1598.

[LLG17] LUO W., LIU W., GAO, S. "Remembering history with convolutional lstm for anomaly detection". In: *2017 IEEE International Conference on Multimedia andExpo (ICME)*. IEEE. 2017, p. 439-444.

[LSR] LOSIER C., SMITH R., ROBERGE, V.. "DESIGN OF A TIME-BASED INTRUSION DETECTION ALGORITHM FOR THE MIL-STD-1553". In : ().

[Moh+13] MOHAN S., et al., "S3A : Secure system simplex architecture for enhanced security and robustness of cyber-physical systems". In: *Proceedings of the 2nd ACM internatio- nal conference on High confidence networked systems*. 2013, p. 65-74.

[MSR16] MAHALAKSHMI G., SRIDEVI S., RAJARAM S., "A survey on forecasting of time series data". In : *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*. IEEE. 2016, p. 1-8.

[Sin17] SINGH A., *Anomaly detection for temporal data using long short-term memory (lstm)*. 2017.

[Sta+17] STAN O., et al. "Protecting military avionics platforms from attacks on mil-std-1553 com- munication bus". In : *arXiv preprint arXiv :1707.05032* (2017).

[Sta+19] STAN O., et al., "Intrusion detection system for the mil-std-1553 communication bus". In : *IEEE Transactions on Aerospace and Electronic Systems* 56.4 (2019), p. 3010-3027.

[Ste07] STEVENSON S., "A comparison of the forecasting ability of ARIMA models". In : *Journal of Property Investment & Finance* (2007).

[STN19] SIAMI-NAMINI S., TAVAKOLI N., Siami -NAMIN A., "The performance of LSTM and BiLSTM in forecasting time series". In : *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, p. 3285-3292.

[Teo+18] TEOH T.T., et al., "Anomaly detection in cyber security attacks on networks using MLP deep learning". In : *2018 International Conference on Smart Computing and Electronic Enter- prise (ICSCEE)*. IEEE. 2018, p. 1-5.

[TL18] TAO F., LIU G., "Advanced LSTM : A study about better time dependency modeling in emotion recognition". In : *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, p. 2906-2910.

[Vas+17] VASWANI A., et al. "Attention is all you need". In : *arXiv preprint arXiv :1706.03762* (2017).

[Wal19] WALTERS, A.G. "Classify Sentences via a Recurrent Neural Network (LSTM)". 2019.

[WYO17] WANG Z., YAN W., OATES T., "Time series classification from scratch with deep neural networks : A strong baseline". In : *2017 International joint conference on neural networks (IJCNN)*. IEEE. 2017, p. 1578-1585.

[Xu+19] XU Y. et al., “Dual-channel CNN for efficient abnormal behavior identification through crowd feature engineering”. In : *Machine Vision and Applications* 30.5 (2019), p. 945-958.

[Yoo+13] YOON M.K. et al., “SecureCore : A multicore-based intrusion detection architecture for real-time embedded systems”. In : *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. 2013, p. 21-32.