



Titre: TRIPT-IDS: Triplet Loss Pre-Trained Transformer for Avionic Intrusion
Title: Detection System

Auteur: Jean-Simon Marrocco
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Marrocco, J.-S. (2023). TRIPT-IDS: Triplet Loss Pre-Trained Transformer for Avionic
Citation: Intrusion Detection System [Mémoire de maîtrise, Polytechnique Montréal].
PolyPublie. <https://publications.polymtl.ca/56994/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/56994/>
PolyPublie URL:

**Directeurs de
recherche:** Gabriela Nicolescu, & Amal Zouaq
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**TRIPT-IDS: Triplet Loss Pre-trained Transformer for Avionic Intrusion
Detection System**

JEAN-SIMON MARROCCO

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Décembre 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**TRIPT-IDS: Triplet Loss Pre-trained Transformer for Avionic Intrusion
Detection System**

présenté par **Jean-Simon MARROCCO**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Martine BELLAÏCHE, présidente

Gabriela NICOLESCU, membre et directrice de recherche

Amal ZOUAQ, membre et codirectrice de recherche

Adel ABUSITTA, membre

DEDICATION

*To my family and friends,
To my teachers, my mentors and my work colleagues,
Thank you for all your support!*

ACKNOWLEDGEMENTS

I extend my heartfelt gratitude to my research director, Prof. Gabriela Nicolescu, and my co-director, Prof. Amal Zouaq, for their unwavering support. I would also like to express my sincere thanks to Jimmy, Nuno, Mikaela, Jean-Yves, and Felipe for the enjoyable times spent in the laboratory and their valuable support during the execution of my research project.

RÉSUMÉ

L'évolution des technologies de pointe et leur intégration transparente dans les infrastructures critiques, en particulier dans le secteur de l'avionique, ont repoussé les limites de ce qui peut être accompli. L'incorporation de systèmes interconnectés et de communication sans fil a entraîné un changement de paradigme dans les avions modernes, et cette tendance devrait perdurer à l'avenir. Malheureusement, ces avancées remarquables ont également introduit des vulnérabilités qui étaient autrefois considérées comme impossibles. Notamment, des protocoles spécialisés tels que le MIL-STD-1553 ont été établis à une époque antérieure à la cybersécurité, les rendant mal équipés pour résister aux menaces cybernétiques modernes.

Le paysage actuel de la cybersécurité dans le domaine de l'avionique repose fortement sur des mécanismes de sécurité enracinés dans des protocoles désuets principalement conçus pour assurer la fiabilité et la détermination de la communication, complétée par l'incorporation de mesures d'obscurcissement. Bien que ces mesures de protection contribuent à renforcer la sécurité, elles ne fournissent pas une sécurité absolue.

Compte tenu du cadre réglementaire actuel et de la complexité inhérente à la rétro-ingénierie des systèmes avioniques existants et futurs, les efforts de recherche en cours sont fortement concentrés sur le développement de solutions pouvant être intégrées de manière transparente en tant qu'extensions, telles que les systèmes de détection d'intrusion (de l'anglais Intrusion Detection System (IDS)). Cette approche vise à résoudre rapidement les vulnérabilités inhérentes à ces systèmes. À l'heure actuelle, les IDS avioniques de pointe utilisent des modules multifacettes pour identifier diverses menaces, en employant principalement des approches basées sur les réseaux neuronaux récurrents (de l'anglais Recurrent Neural Network (RNN)) et les autoencodeurs (AE). Certains de ces IDS sont entraînés exclusivement sur des données sans aucune cyberattaque, mais ils sont limités dans la détection d'un sous-ensemble d'attaques. À l'inverse, les IDS entraînés sur des données étiquetées ne sont souvent pas testés contre une combinaison de vecteurs d'attaque et données de vol précédemment inconnus, et elles peuvent ne pas fonctionner de manière adéquate dans des scénarios à avec d'exemples (de l'anglais few-shot), où seule une petite quantité de données malveillantes est disponible lors de l'entraînement de notre IDS.

L'objectif de cette recherche est de proposer un IDS avionique basé sur la détection d'anomalies d'anomalies qui tire parti d'une architecture basée sur les Transformers et l'apprentissage par transfert. Nous présentons TRIPT-IDS, qui signifie "Triplet loss Pre-trained Transformer Intrusion Detection System" (Système de détection d'intrusion basé sur un modèle Trans-

former pré-entraîné à l’aide d’une fonction de coût par triplet). Cette approche innovante vise à améliorer les performances contre des attaques précédemment inconnues sur le protocole avionique MIL-STD-1553 et à améliorer la généralisation lors de l’entraînement sur un ensemble limité de données d’attaque en apprenant un espace latent significatif au cours d’une phase de pré-entraînement.

Notre méthodologie commence par exploiter à la fois des ensembles de données nouvellement générés et disponibles publiquement à partir du logiciel OD1NF1ST, une interface MIL-STD-1553 à source ouverte utilisée pour générer des données de vol réalistes avec dix types d’attaques synthétiques injectées. Pour améliorer les performances de notre IDS, nous avons élaboré et validé des schémas d’encodage novateurs adaptés au protocole MIL-STD-1553. Ces schémas d’encodage permettent la conservation de méta-informations et s’alignent bien avec la capacité du modèle Transformer à utiliser des balises pour une extraction complète des informations sémantiques et d’arrangement de paquets des séquences.

De plus, nous introduisons une nouvelle méthodologie d’évaluation pour les IDS avioniques qui tient compte de scénarios plus réalistes, notamment la circulation et les attaques précédemment inconnues, les scénarios de classification avec un ensemble limité de données et les expériences avec des attaques catégoriques inconnues. TRIPT-IDS est soumis à une analyse comparative directe avec l’IDS proposé en conjonction avec OD1NF1ST et une variante de notre architecture qui ne bénéficie d’aucune étape de pré-entraînement.

Le premier ensemble d’expériences a testé notre IDS entraîné sur les dix MIL-STD-1553 types d’attaques. Fondamentalement, notre architecture TRIPT-IDS a présenté des résultats supérieurs par rapport à l’approche Bi-LSTM d’OD1NF1ST, avec une augmentation moyenne de 3,30% de l’indice F1. TRIPT-IDS a démontré une compétence remarquable dans le scénario de classification à faible tirage, où l’ensemble d’apprentissages ne contenait qu’environ 0,16% de données malveillantes. Il a surpassé notre variante non pré-entraînée de 16,9% en indice F1, atteignant un score de 94,5%. En revanche, le modèle Bi-LSTM n’a été en mesure de détecter aucune attaque. Cette capacité peut considérablement réduire l’effort et les coûts informatiques associés à la collecte de données, car seul un nombre limité de points de données est nécessaire.

Le deuxième ensemble d’expériences évalue TRIPT-IDS dans plusieurs scénarios d’attaques inconnues. L’objectif de ces expériences est de déterminer dans quelle mesure notre approche peut généraliser les caractéristiques apprises pour détecter de nouvelles attaques ou des attaques inconnues. Cela est particulièrement crucial pour l’IDS avionique, car les véritables menaces cybernétiques n’auront probablement pas été vues lors de l’entraînement. Une fois de plus, TRIPT-IDS surpasse les autres IDSs, notamment dans notre expérience impliquant

des attaques catégorielles non vues. Dans cette expérience, nous nous entraînons sur un petit ensemble d'attaques connues par catégorie et utilisons cette connaissance pour détecter de nouvelles attaques employant des tactiques similaires. Notamment, l'IDS OD1NF1ST s'est révélée incapable d'apprendre adéquatement la détection d'intrusion dans un tel scénario contraignant. Notre approche l'a surpassé par une moyenne de $4,17\times$ en F1-Score par catégorie.

En conclusion, notre recherche a démontré le potentiel des IDS basés sur les Transformers pour les systèmes avioniques, ainsi que l'amélioration des performances résultant d'une tâche de pré-entraînement visant à améliorer la représentation de l'espace latent du trafic. Dans le cadre de travaux futurs, TRIPT-IDS devra être intégré dans une structure pour une application en temps réel sur du matériel avionique et validé sur davantage de protocoles. De plus, diverses tâches de pré-entraînement pourraient être explorées pour capturer des informations sémantiques alternatives, ce qui pourrait améliorer les performances de TRIPT-IDS sur des attaques inconnues plus sophistiquées.

ABSTRACT

The rapid evolution of cutting-edge technologies and their seamless integration into critical infrastructure, particularly within the avionics sector, has pushed the boundaries of what can be achieved. The incorporation of interconnected systems and wireless communication has brought about a transformative paradigm shift in modern aircraft, and this trend is poised to persist into the future. Regrettably, these remarkable advancements have also introduced vulnerabilities that were once considered impossible. Notably, specialized protocols like MIL-STD-1553 were established in a pre-cybersecurity era, rendering them ill-equipped to withstand modern cyber threats.

The current landscape of cybersecurity in the avionics domain heavily relies on safety mechanisms rooted in antiquated protocols primarily designed to ensure communication reliability and determinism, complemented by the incorporation of obfuscation measures. While these protective measures do contribute to enhancing security to a certain extent, they do not provide absolute security.

Given the current regulatory framework and the complex nature of retrofitting both existing and future avionics systems, ongoing research endeavours are heavily concentrated on developing solutions that can be seamlessly integrated as extensions such as Intrusion Detection System (IDS). This approach aims to promptly address the inherent vulnerabilities associated with these systems. Currently, state-of-the-art avionic IDS employ multifaceted modules to identify various threats, primarily employing Recurrent Neural Network-based and Auto-Encoder-based approaches. Some of these IDSs are trained exclusively on benign data but are limited in detecting subset of attacks. Conversely, those trained on labelled data are often untested against a combination of previously unseen attack vectors and flight data, and they may not perform adequately in few-shot scenarios, where only a small volume of malicious data is available during training.

The objective of this research is to propose an avionic anomaly-based IDS that leverages a Transformer-based architecture and transfer learning. We introduce TRIPT-IDS, which stands for Triplet loss Pre-trained Transformer Intrusion Detection System. This innovative approach aims to enhance performance against previously unseen attacks on the MIL-STD-1553 avionic protocol and improve generalization when trained on a limited set of attack data by learning a meaningful latent space during a pretraining phase.

Our methodology commences by leveraging both newly generated and publicly available datasets from the OD1NF1ST software, an open-source MIL-STD-1553 interface used to

generate realistic flight data with ten injected synthetic attack types. To enhance the performance IDS, we have devised and validated novel encoding schemes tailored for the MIL-STD-1553 protocol. These encoding schemes permit the retention of additional meta-information and align well with the Transformer model’s capacity to utilize tags for comprehensive extraction of sequence’s semantic and packet arrangement information.

Additionally, we introduce a new evaluation methodology for avionics IDS that accounts for more realistic scenarios, encompassing previously unseen traffic and attacks, few-shot classification scenarios, and categorical unseen attack experiments. TRIPT-IDS is subjected to direct comparative analysis against the IDS proposed in conjunction with OD1NF1ST and a variant of our architecture that does not undergo any pre-training steps.

The first set of experiments tested our IDS trained on all ten MIL-STD-1553 attack types. Essentially, our TRIPT-IDS architecture exhibited superior results compared to the OD1NF1ST Bi-LSTM baseline, achieving an increase of 3.30% in F1-Score. TRIPT-IDS demonstrated remarkable competence in the few-shot classification scenario, where the training set contained only approximately 0.16% of malicious data. It outperformed our unpre-trained variant by 16.9% in F1-Score, achieving a 94.5%. In contrast, the Bi-LSTM baseline was unable to detect any attacks. This capability can substantially reduce the effort and computational costs associated with data collection, as only a limited number of data points are required.

The second set of experiments evaluates TRIPT-IDS on multiple unseen attack scenarios. The goal of these experiments is to determine how well our approach can generalize learned characteristics to detect new or unseen attacks. This is especially crucial for an avionic IDS, as real cyber threat will most likely not have been seen during training. Once again, TRIPT-IDS outperforms the other IDSs, particularly in our novel experiment involving categorical unseen attacks. In this experiment, we train on a small set of known attacks per category and utilize this knowledge to detect new attacks that employ similar tactics. Notably, the OD1NF1ST IDS proved unable to learn adequately intrusion detection in such scenario. Our approach outperformed it by an average of $4.17\times$ in F1-Score per category.

In conclusion, our research has demonstrated the potential of Transformer-based IDS for avionic systems, as well as the increased performance resulting from a pre-training task aimed at improving traffic embedding space representation. In future work, TRIPT-IDS will need to be integrated into a framework for real-time application on real aircraft hardware and validated on more avionic protocols. Additionally, various pre-training tasks could be explored to capture alternative semantic information, which may improve TRIPT-IDS’s performance on more sophisticated unseen attacks.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	viii
TABLE OF CONTENTS	x
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF SYMBOLS AND ACRONYMS	xvi
LIST OF APPENDICES	xviii
CHAPTER 1 INTRODUCTION	1
1.1 Context	1
1.2 Problem Statement	1
1.3 Research Challenges	3
1.4 Research Objectives	4
1.5 Contributions	5
1.6 Thesis Outline	6
CHAPTER 2 BASIC CONCEPTS	7
2.1 Avionic Systems	7
2.2 Avionic Communication Protocol	8
2.2.1 ARINC 429	8
2.2.2 Avionics Full-Duplex Switched Ethernet	9
2.2.3 Avionics Standard Communication version D (ASCB-D)	12
2.2.4 MIL-STD-1553	13
2.3 Deep Learning Architecture for Intrusion Detection	17
2.3.1 Auto-Encoder	18
2.3.2 Recurrent Neural Network (RNN)	19

2.3.3	Transformer	19
2.4	Transfer learning	21
2.5	Cybersecurity	22
2.5.1	Intrusion detection system	22
2.5.2	Cyberattack technique	23
CHAPTER 3 LITERATURE REVIEW		25
3.1	Deep Learning-based Intrusion Detection System	25
3.1.1	Comparison of Machine Learning and Deep Learning techniques . . .	25
3.1.2	Recurrent Neural Network based IDS	25
3.1.3	Transformer-based IDS	26
3.2	Intrusion Detection System for avionics system	28
3.2.1	ARINC 429 IDS	28
3.2.2	Ethernet-based Protocol IDS	29
3.2.3	MIL-STD-1553 IDS	29
3.3	Alternative countermeasures against avionic cyber-attacks	35
3.3.1	Encryption	36
3.3.2	Ontology	36
3.3.3	Zero Trust Model	37
3.4	Summary Of the State-of-the-Art	39
CHAPTER 4 METHODOLOGY		42
4.1	Data Source	42
4.1.1	ODINF1ST software	42
4.1.2	Adversary model	43
4.1.3	Data Generation	46
4.1.4	Sequence Encoder	48
4.2	TRIPT-IDS Architecture	52
4.2.1	TRIPT Module	53
4.2.2	Pre-training: Triplet Loss Metric-Learning	54
4.2.3	Fine-tuning: Binary Intrusion Detection	57
CHAPTER 5 EXPERIMENTS AND RESULTS		60
5.1	Environment Setup	60
5.2	Experimental Setup	60
5.3	Evaluation Method	61
5.4	Hyperparameters tuning	63

5.5	Experiments on seen attacks scenario	64
5.5.1	Evaluating sequence encoding strategies	64
5.5.2	Fully fine-tune experiment	64
5.5.3	Few-shot classification experiment	69
5.6	Experiments on unseen attacks scenario	71
5.6.1	Individual unseen attack cross-evaluation	71
5.6.2	Categorical unseen attack	73
5.6.3	Few-shot classification on categorical unseen attack	75
CHAPTER 6	CONCLUSION	77
6.1	Summary of Works	77
6.2	Limitations	78
6.3	Future Research	78
REFERENCES		80
APPENDICES		88

LIST OF TABLES

Table 3.1	Comparison of state-of-the-art avionic IDS and TRIPT-IDS	41
Table 4.1	Summary of the OD1NF1ST generated attacks	46
Table 4.2	Word distribution and maximum attack sequence length in datasets .	47
Table 4.3	Distribution of attack sequences in datasets following the generation process with $wSize = 5$	48
Table 4.4	Distribution of attack sequences in datasets following the generation process with $wSize = 20$	48
Table 4.5	Attributes of each word type	49
Table 5.1	Train and test set distribution for both $wSize=5$ and $wSize=20$. . .	61
Table 5.2	TRIPT-IDS hyperparameters	63
Table 5.3	Encoding strategies comparison with the TRIPT-IDS architecture and $wSize=5$	64
Table 5.4	Performance and runtime comparison of TRIPT-IDS on seen attacks scenario	65
Table 5.5	Performance comparison of TRIPT-IDS on few-shot classification at- tacks scenario $wSize=20$	70
Table 5.6	Performance comparison of TRIPT-IDS on unseen attack cross-evaluation with $wSize=20$	72
Table 5.7	Attack category distribution in both train and test set $wSize=20$. . .	73
Table 5.8	Performance comparison of TRIPT-IDS on unseen categorical attacks scenario with $wSize=20$	74
Table 5.9	Performance comparison of TRIPT-IDS on unseen categorical attacks scenario with $wSize=5$	75
Table 5.10	Performance comparison of TRIPT-IDS on few-shot classification on unseen categorical attacks scenario with $wSize=20$	75
Table A.1	Small Transformer hyperparameters	88
Table A.2	Performance and runtime comparison of Small-TRIPT-IDS on seen attacks scenario	88

LIST OF FIGURES

Figure 2.1	Commercial aircraft domains. [1]	7
Figure 2.2	Aviation systems and technology infrastructure for communication. [1]	8
Figure 2.3	An ARINC 429 layout with just one transmitting Line-Replaceable Unit (LRU) and up to multiple recipients. [2]	9
Figure 2.4	The data format of an ARINC 429 word [2]	9
Figure 2.5	Illustration of Virtual Links over an Avionics Full-Duplex Switched Ethernet (AFDX) Network. [3]	10
Figure 2.6	The arrangement of an AFDX frame resembles a standard Ethernet frame and can therefore be transported via COTS Ethernet hardware. [2]	11
Figure 2.7	ASCB-D achitecture [4]	12
Figure 2.8	MIL-STD-1553 architecture and its primary components.	13
Figure 2.9	MIL-STD-1553 format for Command, Data and Status words [5] . . .	15
Figure 2.10	MIL-STD-1553 communication schemes [5]	16
Figure 2.11	A basic Deep Neural Network architecture.	17
Figure 2.12	A basic Auto-Encoder architecture.	18
Figure 2.13	A basic Recurrent Neural Network architecture formed with memory cells.	19
Figure 2.14	Transformer architecture [6]	20
Figure 3.1	<i>Stan et al.</i> IDS process [7]	31
Figure 3.2	TLP-IDS process [8]	32
Figure 3.3	AnoMili architecture [9]	33
Figure 3.4	AnoMili’s anomaly explanation engine architecture [9]	34
Figure 3.5	OD1NF1ST IDS architecture [10]	35
Figure 3.6	The “Swiss cheese” model of human error causation [11]	38
Figure 3.7	GPS chain of trust [12]	38
Figure 4.1	Overview of integrated pipeline to process OD1NF1ST data. [10] . . .	43
Figure 4.2	Comparison of three sequences encoding schemes $wSize = 5$. The <i>Original</i> scheme is the one introduced alongside the OD1NF1ST software. The <i>Tagging</i> scheme incorporates tags into the sequence to aid the Transformer model’s focus during learning. The <i>Vector</i> scheme generates a general word vector encompassing all attributes of different word types.	50

Figure 4.3	Overview of the TRIPT-IDS architecture	52
Figure 4.4	Overview of the TRIPT Module architecture	54
Figure 4.5	Overview of the Triplet Loss Metric-Learning pre-training task setup, where C_c^a , C_c^p , C_c^n are the class logits outputted by the TRIPT Module for the anchor, positive and negative input respectively.	56
Figure 4.6	The error produced by the Triplet Margin Loss function is designed to encourage the model to minimize the distance between an <i>anchor</i> and a <i>positive</i> sample of the same class, while concurrently maximizing the distance between the <i>anchor</i> and a <i>negative</i> sample of a different class.	57
Figure 4.7	Overview of the Binary Intrusion Detection fine-tuning task setup, where C_c is the class logits outputted by the TRIPT Module.	59
Figure 5.1	OD1NFIST IDS fully fine-tune experiment results: (a) TPR and FNR for each sequence type; (b) Output embedding of each sequence by type	66
Figure 5.2	TransformerEncoder-IDS fully fine-tune experiment results: (a) TPR and FNR for each sequence type; (b) Output embedding of each se- quence by type	67
Figure 5.3	TRIPT-IDS fully fine-tune experiment results: (a) TPR and FNR for each sequence type; (b) Output embedding of each sequence by type	68
Figure 5.4	True Positive Rate (TPR) and False Negative Rate (FNR) for each attack type during few-shot training experiment for TRIPT-IDS . . .	70
Figure 5.5	TPR and FNR for each unseen attack type during separate training experiment for TRIPT-IDS	72

LIST OF SYMBOLS AND ACRONYMS

AFDX	Avionics Full-Duplex Switched Ethernet
AISD	Airline Information Services Domain
ACD	Aircraft Control Domain
AOC	Air Operational Control
ATC	Air Traffic Control
AUC	Area Under the ROC Curve
AI	Artificial Intelligence
ANN	Artificial Neural Network
AE	Auto-Encoder
ASCB-D	Avionics Standard Communication version D
BC	Bus Controller
BM	Bus Manager
CAN	Controller Area Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DL	Deep Learning
DoS	Denial of Service
DDoS	Distributed Denial of Service
DoD	United States Department of Defence
EFB	Electronic Flight Bag
FAA	Federal Aviation Administration
FAR	False Alarm Rate
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
GAN	Generative Adversarial Network
GAO	Government Accountability Office
HIDS	Host-based Intrusion Detection System
ICS	Industrial Control System
IDS	Intrusion Detection System
IMA	Integrated Modular Avionics
ML	Machine Learning

MITM	Man-in-the-Middle
MSFS	Microsoft Flight Simulator 2020
MLP	Multi-Layer Perceptron
NIDS	Network-based Intrusion Detection System
LRU	Line-Replaceable Unit
LSTM	Long Short-Term Memory
PIESD	Passenger Information & Entertainment Services Domain
PODD	Passenger Owned Devices Domain
RT	Remote Terminal
RNN	Recurrent Neural Network
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TNR	True Negative Rate
UDP	User Datagram Protocol
VL	Virtual Link

LIST OF APPENDICES

Appendix A	Small hyperparameters and experiment results	88
------------	--	----

CHAPTER 1 INTRODUCTION

This chapter introduces the overall motivations and objectives behind the field of avionic cybersecurity. We explore the challenges and context that have inspired this thesis. This is followed by a brief overview of the document’s organization.

1.1 Context

Intrusion detection is a crucial task for ensuring the security and reliability of critical infrastructures, such as power grids, transportation systems and aviation networks. These infrastructures are increasingly exposed to cyberattacks and physical faults that can compromise their normal operation and cause severe consequences. Therefore, developing effective methods to identify and isolate anomalies in real time is a challenge for researchers and practitioners [13]. Recent incidents related to aircraft and the automobile domain [14], have revealed that there is insufficient protection in place to mitigate the risks posed by potential attackers. Based on a research done by Trellix for the Q3 2022 [15], in the United States alone there is been a 100% increase of ransomware campaigns against the transportation and shipping sector (includes aviation) and globally it is the most impacted sector following telecommunications. As technology progress, critical infrastructure industries are becoming a more attractive target as the cyber threats and malicious actors continue to grow. As of today, the aerospace sector lags behind in implementing key practices necessary for conducting risk-based cybersecurity oversight programs. Although the motivation and resources needed to carry out a successful cyberattack on aircraft makes them highly unlikely, compared to common cyberthreats, a single instance could cause catastrophic consequences. Thus, institution like the Government Accountability Office (GAO) emphasize the importance for government agencies like the Federal Aviation Administration (FAA) to solves its cybersecurity oversight for airplanes [16].

1.2 Problem Statement

In recent years, the avionics industry has undergone an exponential evolution powered by the progress achieved by the implementation of integrated modular avionics and Ethernet-based protocols. This facilitated real-time data links, weight reduction and the integration of various applications. Consequently, this progress opened the door to new attack vectors that were previously infeasible. Previous studies [1, 16] have documented the potential vulnerabilities,

threat actors, adversarial capabilities, vulnerabilities, operational impact, and more. Their analysis has revealed that adversaries could gain access to the target system through various means, including airport network, airline network, third-party networks, insider attacks, human errors, public-facing applications, supply chain compromise, and targeting internet-accessible devices. Adversaries can exploit overlapping connection via external and internal network access points allowing them to spread between different components. After obtaining initial access, the adversaries can establish communication channels to command and control the compromised assets by abusing communication methods. Evasion tactics are also possible in aircraft systems. This can be done by displaying false information on various panels (e.g., cockpit control panels, Electronic Flight Bag (EFB) applications and crew terminal panels), suppressing alarms, blocking auxiliary communications, interfering with sensor data and spoofing or blocking reporting messages [1]. The objective of an attack varies depending on the threat actor, ranging from disrupting operations and corporate espionage to endangering passengers' safety. More research has highlighted various security incidents and threats that have happened in recent years, providing evidence of vulnerabilities not only in avionic systems but also across the entire aerospace sector [14].

Addressing the problem of the lack of cybersecurity in critical infrastructure requires an in-depth knowledge of the infrastructure ecosystem, industry-specific policies, specifications, and regulations. When focusing on the aircraft, four distinct branches can be defined: Aircraft as a System of Systems, Aircraft as a Node of a Network, Aircraft interaction within its environment and policies regarding the Aviation Ecosystem [17]. For the branch Aircraft as a System of Systems—where the aircraft is considered as a box of system—the threat vectors revolve around the internal networks and avionics systems [17]. To achieve resiliency against these threats, it is possible to draw inspiration from other industries, such as telecommunications, where significant progress has been made. However, implementing solutions in the avionics industry is challenging due to the large number of regulations and the presence of diverse outsourced onboard computers, communication protocol and legacy systems in current and near-future aircraft. Therefore, only strategies that do not necessitate changes to the current infrastructure are feasible to implement. Intrusion Detection System (IDS) are security technologies that watch over computer networks or system for malicious activity or policy violation. These systems can easily be installed on top of systems and networks without impacting the current behaviour. When developing such systems in the avionics ecosystems, three main components can be explored. Firstly, human-IDS interfaces play a critical role in effectively communicating important information to the flight crew without becoming a nuisance that increases the workload [17]. Trustworthiness modules focus on mitigating low-level attack vectors, ensuring secure installation of legitimate hardware, and

facilitating the development, deployment, and operation of software systems [17]. Lastly, predictive analytic solutions utilize prior knowledge to assess the likelihood of an attack [17].

Given the complexity of avionics systems and the increasing sophistication of cyber threats, Artificial Intelligence (AI)/Machine Learning (ML) methodologies have emerged as central components of many solutions. More especially, Deep Learning (DL), a subset of ML methods that employs multilayered Artificial Neural Networks (ANNs), has demonstrated superior performance compared to standard ML approaches. DL methods excel in modelling complex logic, making them well suited to address avionics security challenges [18, 19].

Accurately detecting novel or zero-day attacks is a crucial for real-life Network-based Intrusion Detection System (NIDS) that aims to secure critical infrastructure. However, due to the scarcity of high-quality datasets and the lack of real cyberattack examples in actual aircraft, it becomes necessary to develop solutions that are robust against unseen attacks. With that in mind, we introduce a novel approach for training predictive analytic type IDS models, coupled with a new architecture called TRIPT-IDS. This anomaly-based IDS harnesses the transfer-learning capability of a Transformer-based model through a triplet loss metric pre-training task with the goal to learn a more representative embedding space.

By leveraging a pre-trained embedding representation of network traffic that highlights certain key characteristics, a downstream task can prioritize discerning malicious traffic from benign ones. By introducing transfer-learning capability into IDS solutions, we open the door to solutions that would have expendable capabilities, enhancing their effectiveness in detecting previously unseen attacks.

1.3 Research Challenges

Aerospace cybersecurity is a brand-new field of research and with it comes numerous challenges. In the section, we will expand on the specific challenges that have notable impact.

The most prevalent challenge faced by every researcher in this domain is the limited accessibility to avionics systems and associated information. This constraint arises due to proprietary nature of the systems, and the reluctance of manufacturers and operators to share critical data. Their stance is justified by concerns over security and safety. While it is true that exposing vulnerabilities to the general public may potentially incite malicious actors, it is also preventing substantial advancements in the field [1, 16]. In our perspective, their strategies of relying solely on security through obfuscation, while effective in deterring potential malicious actors, offers diminishing returns in the long run.

Acquiring real flight data presents a significant challenge primarily due to the extensive

internal processes within aviation companies and the rigorous security compliance measures. While these barriers are essential for upholding a high level of security, they have considerably hindered progress in aviation research. Consequently, researchers have often resorted to using synthetic data and off-the-shelf testbeds to validate their solutions. However, synthetic datasets are constrained by expert assumptions and the inherent randomness introduced by researchers during their creation. This limitation implies that some IDS solutions may not exhibit robustness against scenarios that fall outside the scope of the defined equations used in the simulation [20].

The complexity of avionics systems and the associated cost and resource constraints constitute another significant challenge that influenced the direction of our study. In short, effectively developing and implementing cybersecurity solutions for avionics can be expensive and require substantial resources [16]. The validation of feasibility for integrating new systems necessitates a realistic testbed. However, hardware required for such testbeds are costly, as they are mainly intended for use in actual aircraft. Additionally, the design and creation of a testbed can be a time-consuming task that may be treated as a separate project. In these situations, conducting a study in parallel with the implementation and procurement of a testbed can pose challenges for any group in validating their findings through practical experimentation.

1.4 Research Objectives

Considering the aforementioned challenges, we have framed our research question as follows:

Can the performance of an anomaly-based Intrusion Detection System against previously unseen attacks on the MIL-STD-1553 avionic protocol be improved by incorporating semantic representations into the inputs and leveraging a pre-training phase aimed at learning a meaningful latent space?

In this research, the primary objective is to investigate the effectiveness of a specific approach in addressing the challenges associated with detecting new intrusion in avionics systems. Our proposed solution, called Triplet loss Pre-trained Transformer Intrusion Detection System (TRIPT-IDS), harnesses the transfer-learning capability of a Transformer-based model. In particular, we draw inspiration from the work done on embedding enriching [21–25], and integrated the task of metric learning using the triplet loss [26] function with an improved input encoding scheme. To train and evaluate our solution, we utilize the publicly available *L1NNA Dataset* [27] that comes with OD1NF1ST [10], an open-source MIL-STD-1553 interface to generate realistic flight data with injected synthetic attacks. By utilizing this

software, we generate more flight data with synthetic attacks injected to use alongside the available dataset, essentially doubling the amount of usable data. This addition enables us to conduct more extensive experiments and evaluate the performance of our TRIPT-IDS approach more accurately.

In order to achieve our primary goal, we have established the following sub-objectives:

1. Generate additional flight data for the MIL-STD-1553 protocol using the OD1NF1ST software, enabling more complex experiments;
2. Engineer and validate new encoding schemes for the MIL-STD-1553 protocol to improve performance with the new architecture;
3. Develop a Transformer-based IDS for avionic systems, drawing inspiration from its successful application in telecommunications IDS research.
4. Design a novel approach for training IDS that includes a pre-training phase followed by a downstream task of intrusion detection;
5. Validate our novel solution against a state-of-the-art approach on a new evaluation methodology for avionics IDS that accounts for more realistic scenarios, involving unseen traffic and attacks, few-shot classification, and categorical unseen attack experiments;

1.5 Contributions

Following the completion of the objectives mentioned above, we successfully addressed our research questions and presented three subsequent contributions.

Proposing a semantically enhanced MIL-STD-1553 packet sequence encoding.

In this work, we present two new approaches to encode MIL-STD-1553 packet sequences that improve our model’s performance by 8.65% on average across our evaluation metrics. The first approach introduces tagging to the originally proposed method, while the second fully innovates the process by representing each packet of a sequence with a vector representation that preserves the distinction between packet attributes.

Assembling an array of realistic intrusion detection experiments. We evaluate our approach on five different experiments, each aiming to challenge an IDS in various ways. Particularly, three of them are introduced for the first time by us in the context of avionic intrusion detection research. We utilize two datasets, one publicly available for training and a second that we generated with the same tool for testing. This ensures that the IDS is tested

under different flight conditions for all experiments. The first newly introduced experiment tests the IDS in a few-shot setting, where only 0.16% of the training set consists of malicious traffic. The second aims to measure the ability of an IDS to learn categorical patterns from only a few types of attacks in order to detect a larger set of previously unseen types. The last is a combination of the previous two, pushing the IDS to its limits in an extremely unfavourable training setting.

Pre-training an IDS with triplet loss. To the best of our knowledge, we are the first to use triplet loss as part of the pre-training of an IDS. We compared the performance of our Transformer-based architecture with and without a triplet loss pre-training phase and have successfully demonstrated an increase in performance. We achieved an average increase of 13.97% terms of precision, recall, F1-Score, Area Under the ROC Curve and False Alarm Rate (FAR) for all evaluation experiments, notably a 47.77% increase in the F1-Score for the more demanding experiments of few-shot classification on categorical unseen attacks.

1.6 Thesis Outline

The structure of the thesis is as follows: In Chapter 2 we review avionics systems, avionics protocols, relevant DL architectures, and cybersecurity concepts. Chapter 3, provides an overview of the state-of-the-art in NIDS, with a section focusing on avionic systems, and discusses alternative countermeasures against cyberattacks on avionic systems. Chapter 4, offers a detailed exploration of our data sources, an enhanced encoding scheme for MIL-STD-1553 data, and the proposed model architecture, along with its distinct training stages. In Chapter 5, we describe the different experiments conducted and present the obtained results. Lastly, Chapter 6 offers concluding remarks and outlines directions for future work.

CHAPTER 2 BASIC CONCEPTS

This chapter focuses on basic concepts crucial for comprehending the key subjects discussed in the upcoming literature review and proposed solutions. We begin with a exploration of avionic systems, followed by an in-depth examination of pertinent network communication protocols. We go over DL architectures that are used in state-of-the-art solutions. Finally, this chapter concludes by presenting essential cybersecurity concepts and techniques.

2.1 Avionic Systems

The evolution of avionics and information systems in aircraft over the last decade has been marked by significant advancements. Yet, the architecture of these systems still includes a combination of new and legacy technologies, partly due to the lengthy approval processes for new innovations in the aviation industry. As the number of flights increases daily [28], there is a growing need to reduce fuel costs by reducing the weight of the aircraft which can be achieved by improving network infrastructure. This drive for efficiency has been supported by the development of numerous software systems on Integrated Modular Avionics (IMA) capable of supporting numerous applications. Additionally, the transition to Ethernet protocols and networks has played a crucial role in enhancing the performance and capabilities for commercial and cargo aircraft. [1]

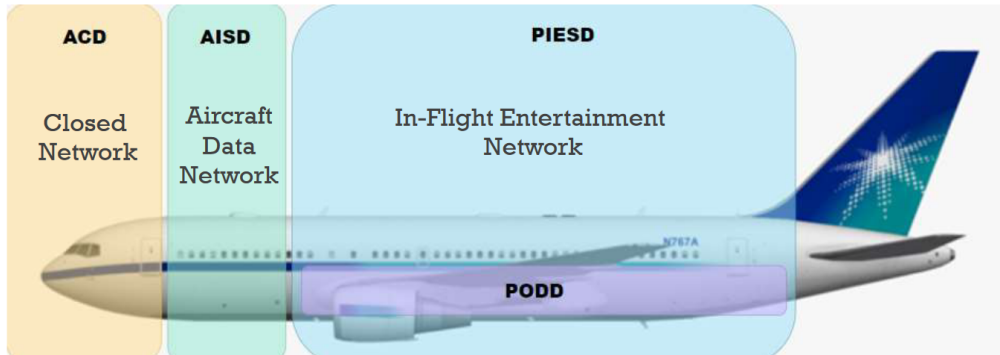


Figure 2.1 Commercial aircraft domains. [1]

Aircraft systems play diverse roles with varying levels of importance and sensitivity, which differ based on the category of aircraft, such as commercial, cargo, and military. Figure 2.1 shows the four domains in which these systems are categorized in a commercial setting: the Aircraft Control Domain (ACD), Airline Information Services Domain (AISD), Passen-

ger Information & Entertainment Services Domain (PIESD) and Passenger Owned Devices Domain (PODD).

Our IDS focuses on the ACD, which comprises critical systems and network responsible for the aircraft's essential operation. Consequently, the ACD demands the most strict security requirements, encompassing systems like the Air Operational Control (AOC) and the Air Traffic Control (ATC). Within the ACD, there are two key subdomains: *Cabin-Core*, which is necessary to operate the cabin's other components, and the *Flight and Embedded Control System*, which enable pilots to control the aircraft from the flight deck. Figure 2.2 display several input/output sources that facilitate communication with on-ground services. Some of these communication sources also serve the passenger domains; hence, the separation between domains is not absolute [1].

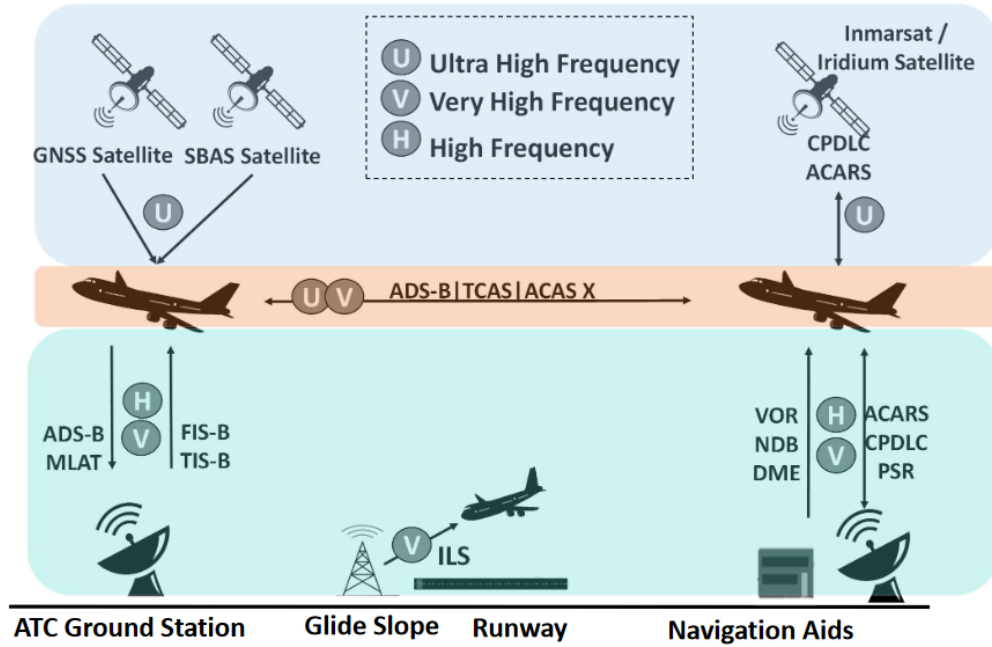


Figure 2.2 Aviation systems and technology infrastructure for communication. [1]

2.2 Avionic Communication Protocol

2.2.1 ARINC 429

ARINC 429 is one of the earliest standards specifically targeted at the avionics applications and continues to be used today in small aircraft and with legacy hardware [29]. In ARINC 429, Line-Replaceable Units (LRUs) are directly interconnected through a serial that is twisted shielded and pair-based, establishing connection between peers. Unlike modern

protocols, ARINC 429 is a signalling standard. Figure 2.3 demonstrates that only one LRU can transmit data on a given line, while multiple recipients LRUs can listen to the transmitted data. Communication on this protocol is unidirectional, thus if a receiver LRU needs to respond, it requires its own dedicated communication line to transmit back to the previous transmitting LRU. This characteristic of the protocol highlights the potential weight and complexity it can introduce to an aircraft system. [2]

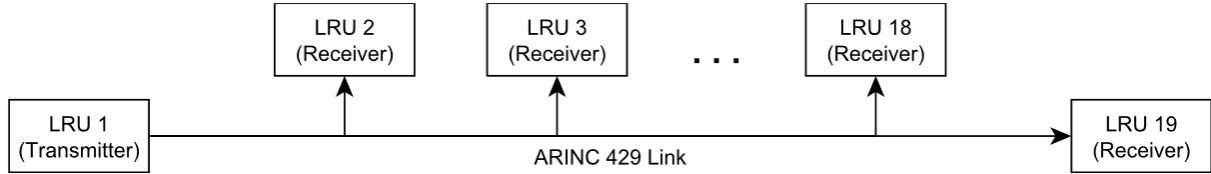


Figure 2.3 An ARINC 429 layout with just one transmitting LRU and up to multiple recipients. [2]

ARINC 429 is a non-byte-aligned protocol that prioritizes low latency. In fact, its maximum word size of 32 bits plays a central role in achieving safety, resilience and reliability within the protocol. The small size and unidirectional nature of the traffic result in minimal latency, reducing processing delays and ensuring precise timing. This characteristic makes ARINC 429 a leading avionic protocol for responsiveness. Figure 2.4 presents the structure of an ARINC 429 frame. The frame header encompasses a label that carries encoding and decoding format information. These label digits are predefined within the standard and can assume various meanings depending on the specific LRUs involved in data transmission and reception. Following the label, bits 8-9 define the source and destination, while the next 19 bits contain the data payload. The SSM (Sign/Status Matrix) bits define the type of data, and the final bit serves as a parity bit. [2]

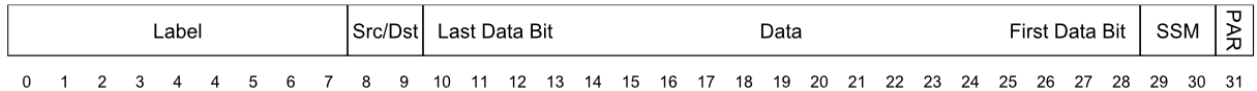


Figure 2.4 The data format of an ARINC 429 word [2]

2.2.2 Avionics Full-Duplex Switched Ethernet

Avionics Full-Duplex Switched Ethernet (AFDX), also referred as ARINC 664, is an avionics communications protocol that combines Ethernet technology with proven security and availability features. While Ethernet has been widely used outside of the aerospace industry, it

lacks essential features required for high availability and reliability needed in aerospace [3]. The main drawback of using Ethernet directly in avionics stems from its non-determinism of Ethernet. Specifically, when sending data in such networks, collisions can occur causing all stations involved to resend the data afterwards, which can lead to a blocked bus. To resolve this issue, the concepts of *Virtual Links (VLs)* and *Switches* were introduced in AFDX [2].

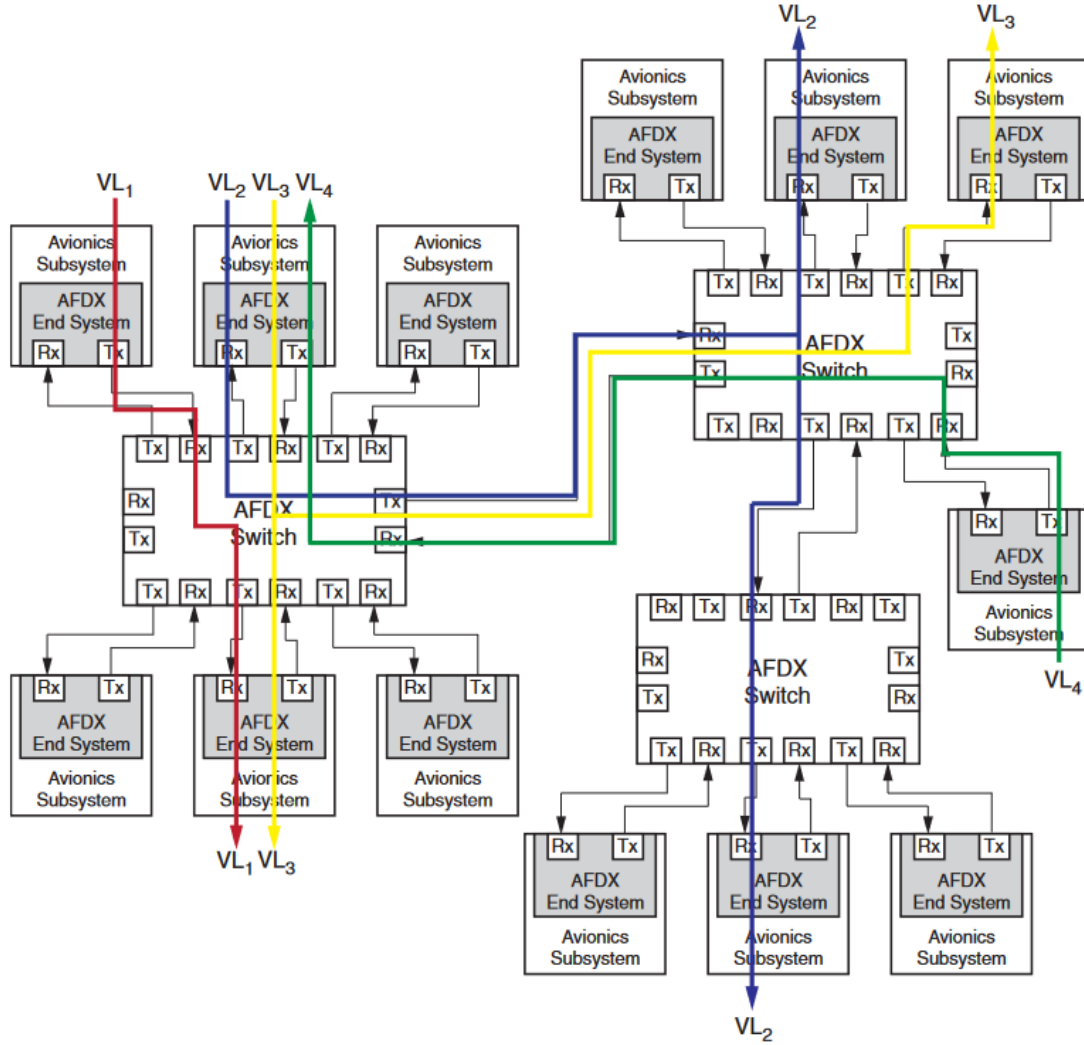


Figure 2.5 Illustration of Virtual Links over an AFDX Network. [3]

Virtual Links are point-to-point connections which differ from the physical existence of ARINC 429. Instead VLs are logical links. These VLs operate unidirectional, allowing only one *subsystem* to transmit data using a *Virtual Link identifier (VLID)*, replacing MAC-addresses found in Ethernet. *Subsystems* use communication ports to assume different roles in multiple VLs. Additionally, many recipients can also be part of a VL. In order to support the high availability required in this environment, AFDX incorporates redundancy by having two *end*

systems for every *subsystems*. Subsequently, duplicate frames are transmitted, with the first arriving to the destination ahead of the next duplicate frame, which must be detected and discarded. This process is facilitated by a frame sequence number, that is analyzed by the Redundancy Manager(RM). [3]

AFDX *switches* are specialized hardware required to enforce VL rules that make determinism possible. Switches are responsible for enforcing latency, bandwidth constraints, and providing a reliable, fixed configuration. The VLs are isolated from each other via the switches that perform passthrough frame scheduling based on the VLID [2,3].

Following the trend of echoing the Internet world, Figure 2.6 shows how AFDX adheres to the Open Systems Interconnection (OSI) layer model. Contrary to what is normally expected, some functionalities (e.g., packet loss handling and packet reordering) are implemented in layers 2 and 3 of the network structure. Because transmission control and integrity are already provided at these two lower layers, UDP is chosen as the default protocol in AFDX instead of the more common TCP [2].

				Type (2)	IP Header (20 bytes)	UDP Header (8 bytes)	L3 Payload (>17 bytes)			
Preamble (7 bytes)	Start Frame Delimiter (1 byte)	AFDX Destination Address (6 bytes)	AFDX Source Address (6 bytes)	Layer 2 Payload				Sequence Number (1 byte)	FrameCheck Sequence (4 bytes)	Inter Frame Gap (12 bytes)

Figure 2.6 The arrangement of an AFDX frame resembles a standard Ethernet frame and can therefore be transported via COTS Ethernet hardware. [2]

Although no public datasets or specific examples of cyberattacks on the AFDX protocol have been disclosed, some studies have revealed vulnerabilities in the protocol's design. One notable concern pertains to the redundancy management and integrity checking concept, which can lead to frame dropping or babbling resets if a network error occurs [30]. While these vulnerabilities are not directly linked to cyber threats, it is important to note that a malicious actor who gains access to the system only needs to modify one bite of a frame to initiate an Denial of Service (DoS) attack. The AFDX protocol does not validate the content of the payloads. Consequently, a corrupt subsystem could transfer invalid data that would be processed by the receiver systems without their knowledge.

2.2.3 Avionics Standard Communication version D (ASCB-D)

ASCB-D is an Airbus propriety communication protocol for private and civil aircraft [4]. The motivation behind ASCB-D was to develop a fault-tolerant with real-time communication protocol for their Primus Epic avionics suit. Like AFDX, ASCB-D is based on Ethernet and shown in Figure 2.7 to use two buses for each side of an airplane with a backup bus for each. The protocol is TDMA-based (Time-division multiple access) and it is designed to guarantee that all four buses remain deterministic and synchronized at the data transmission rates [31].

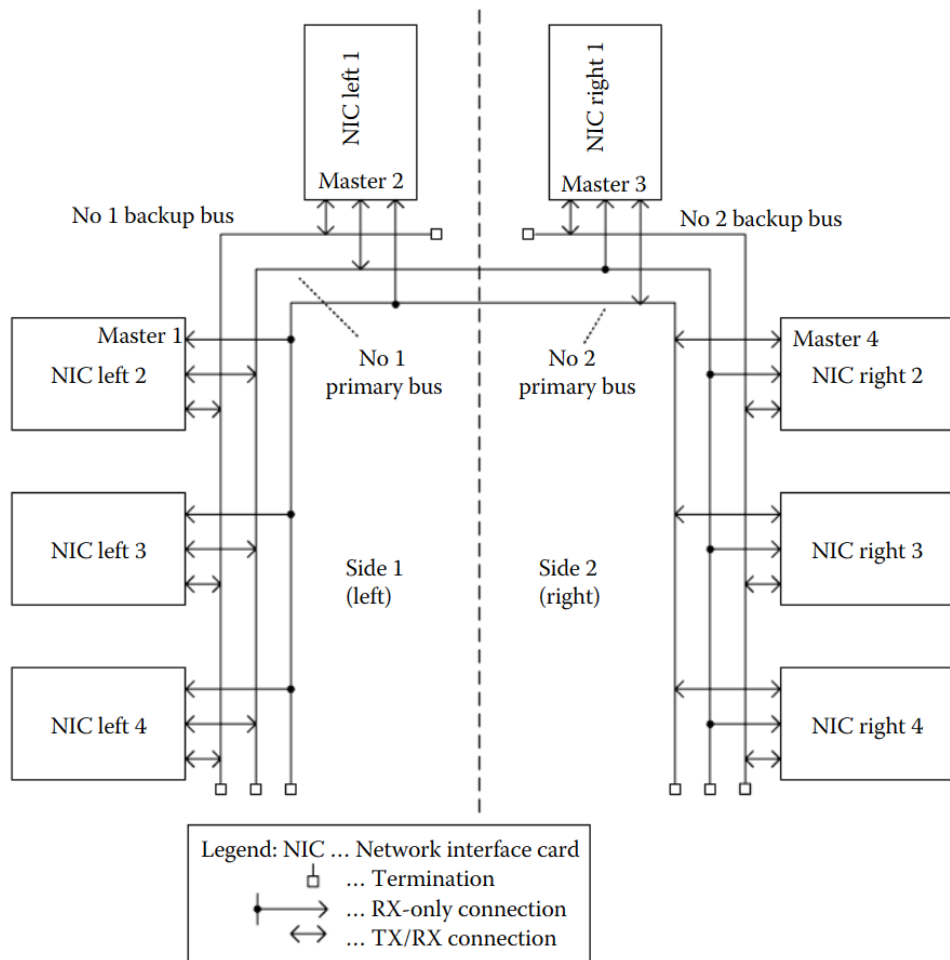


Figure 2.7 ASCB-D achitecture [4]

2.2.4 MIL-STD-1553

The MIL-STD-1553 is a military-oriented standard that was initially introduced by the United States Department of Defence (DoD) in 1968. Originally intended for military avionic applications, it has found broader use across armed forces, spacecraft and even some commercial avionic systems. Over the years, MIL-STD-1553 has become the networking backbone in embedded avionics system and has been utilized extensively for many decades. Experts anticipate that it will continue to be employed in near-future military vehicles' internal communication [32].

Communication bus

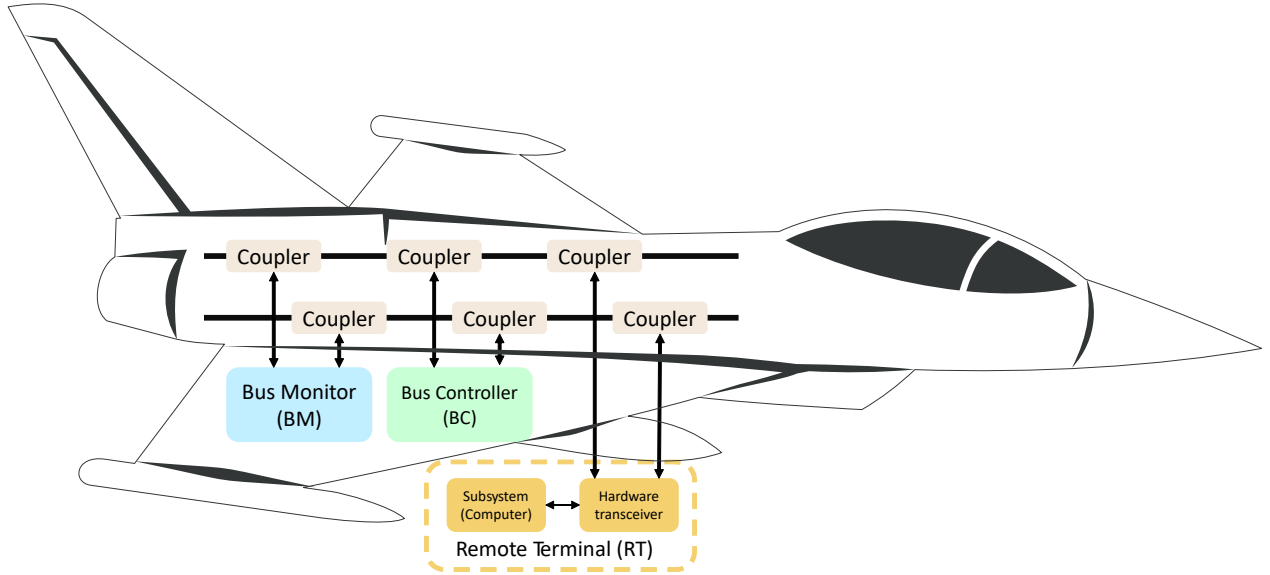


Figure 2.8 MIL-STD-1553 architecture and its primary components.

Figure 2.8 illustrate the basic arrangement of the MIL-STD-1553 communication bus. The bus consists of several components, but the communications are handled by three main ones [5,32]:

Bus Controller (BC): This component is responsible for managing all data flow on the bus. It uses a command/response method and is the only device that can initiate a data transmission. To ensure redundancy and backup functionality, multiple BC can be present on the network, though only one can be active at any given time. The BC employs predefined time constraints to detect the presence of errors and trigger recovery procedures when necessary.

Remote Terminal (RT): RTs can transfer data to and from computing subsystems and sensors. It follows the directive of the bus controller to know when they can send words through the bus.

Bus Manager (BM): The BM monitors, collects and records all messages and data from the bus depending on the storage capability. It is an essential component for developing and integrating an IDS in the MIL-STD-1553 architecture, as it enables collecting data for training purposes and adding intrusion detection capabilities to the system.

The protocol support double and triple redundancy by coupling each device to multiple bus. Although improving the reliability of the system, the redundancy does nothing for the security. All transmissions are not encrypted and are visible to every device linked to the bus. [5]

Word types

The MIL-STD-1553 protocol utilizes frames—commonly referred to as a "word" in the literature—as the basic unit for transmitting messages. There are three types of words: command, status and data word. Each word consists of a 16-bit data field, a 3-bit sync pattern, and a 1-bit parity, resulting in a 20-bit word format. These words are transmitted sequentially in a given message, and there is a minimum gap of $4\mu s$ between consecutive words. However, in practice, the gap between each word is often found larger, it can extend up to $1ms$ for some older BCs. This extended gap provides enough time for a compromised component to inject fake messages into the communication stream without raising suspicion. [32]

Command Word: The commands are classified into three categories: transmit commands, and mode code commands. Command words define the message format to be transmitted and can only be sent by a BC to the RTs. As shown in Figure 2.9, it is composed of an address field to indicate the RT being commanded, a transmit/receive bit to signal the direction of the data flow, and a sub-address/mode code field to specify the type of data desired or the mode of operation. [32]

Status Word: Status are sent at the start and at the end of a data transmission to the BC. They serve the purpose of acknowledging and validating commands, as well as confirming the successful transfer of data [32]. In Figure 2.9, the bits used to represent specific statuses are illustrated.

Data Word: This word type has the simple task of carrying the information part of a message between RTs and the BC or between RTs [32]. The payload consists of 16 bits that contain the information sent through the bus between two devices.

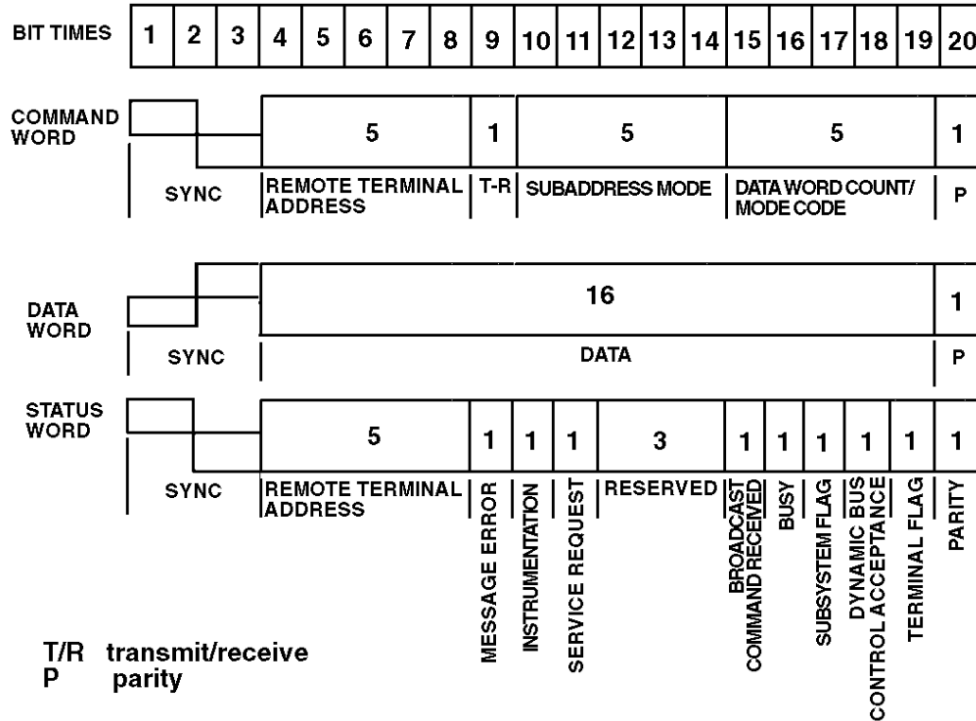


Figure 2.9 MIL-STD-1553 format for Command, Data and Status words [5]

Communication schemes

Figure 2.10 illustrates four different communication schemes in which transmitted messages can be classified into [32]. All of these protocols are initiated by the BC.

RT-BC: The BC sends a transmit word to specify which RT will be sending the data. Next, the RT in question response with a status word followed by the data words and ending by a another status word.

BC-RT: This communication mode facilitates the communication where the BC will send data to a RT listening. The process begins with the BC instructing the RT in question to prepare for the reception of data with a receive command. The receive command is followed by specifying the number of incoming data words. Once the RT receives the command and completes reading the data words, it must respond promptly

by transmitting a status word, ensuring it reaches the BC no later than $12\mu s$. Any delay beyond this time frame will lead the BC to assume an error occurred.

RT-RT: Allows data transfer from one RT to another. Just like the previous protocols, the BC will initiate the communication. This time, it sends a transmit and receive word on the bus indicating which RT will be transmitting and which one will listen. The transmitter RT will send a status word before sending the data words and the receiver RT will send a status word at the end of the protocol.

Broadcast: Broadcast communication scheme can be divided into two types. In the first one, the BC initiates the broadcast by sending a receive word to all devices, followed by the data words. Alternatively, in the second type, the BC sends a transmit word to a target RT that will then broadcast data on the bus.

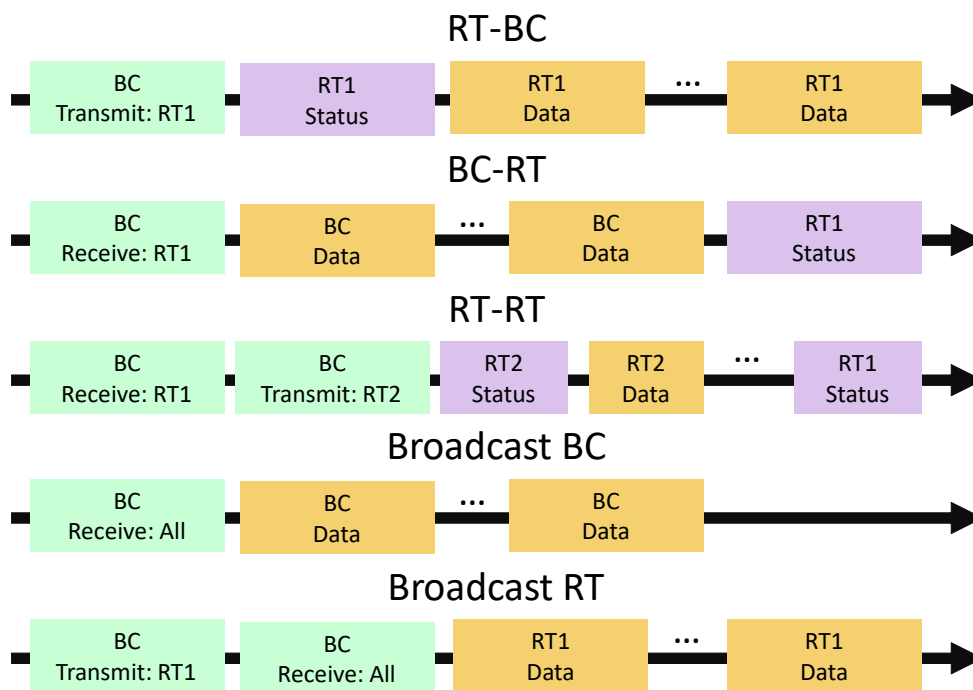


Figure 2.10 MIL-STD-1553 communication schemes [5]

2.3 Deep Learning Architecture for Intrusion Detection

Intrusion detection is a highly complex task that requires a deep understanding of the underlying data to be successful. DL is a specialized subset of ML that employ ANNs at its core. These ANNs perform various non-linear data transformations and learn to recognize useful patterns in the data, making them highly versatile, powerful, and scalable for intrusion detection. [18]

ANN are built with Multi-Layer Perceptron (MLP) and are called Deep Neural Network (DNN) when they have more than two hidden layers. Figure 2.11 illustrates the basic structure of a DNN. We can observe the *perceptrons* found at the roots of all ANN. They are units that compute a weighted sum of their inputs and then apply an *activation function* to generate an output. The learning process in ANN stems from the principle of *backpropagation*. After the network produces an output from given inputs, an error is calculated using an *error function* (also called *loss function*). This error is then propagated backward through the network, measuring the contribution of each connection to the error. The final step involves slightly adjusting the weights to minimize the error, thus refining the network's performance. [33]

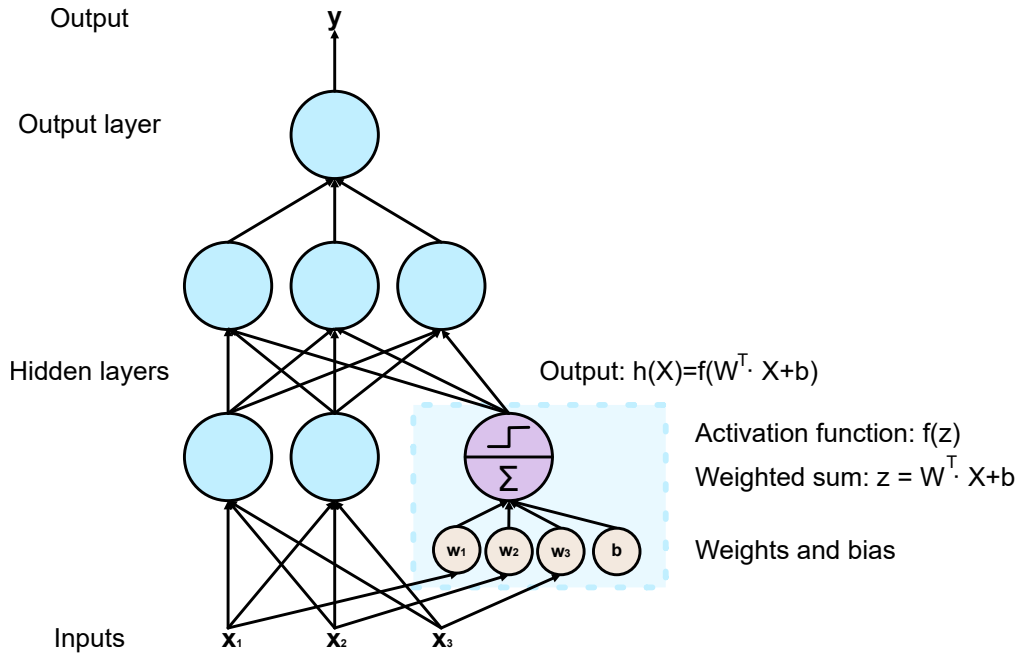


Figure 2.11 A basic Deep Neural Network architecture.

2.3.1 Auto-Encoder

Auto-Encoder (AE) are feed-forward ANNs that possess the capability to learn efficient representations of input data. Figure 2.12 shows the architecture of an AE, which consists of two ANNs— acting as the encoder and decoder respectively. The encoder’s output represents a lower-dimensional representation of the input data, which the decoder then attempts to reconstruct to its original state.

This process is carried out in an unsupervised manner, meaning the training set used for training the AE does not require labelled data. Instead, the AE learns from the raw input data itself, finding patterns and features within the data that enable it to create a compact and meaningful representation. This ability to automatically learn data representations makes AEs valuable in various applications, including data compression, anomaly detection, and feature learning [33].

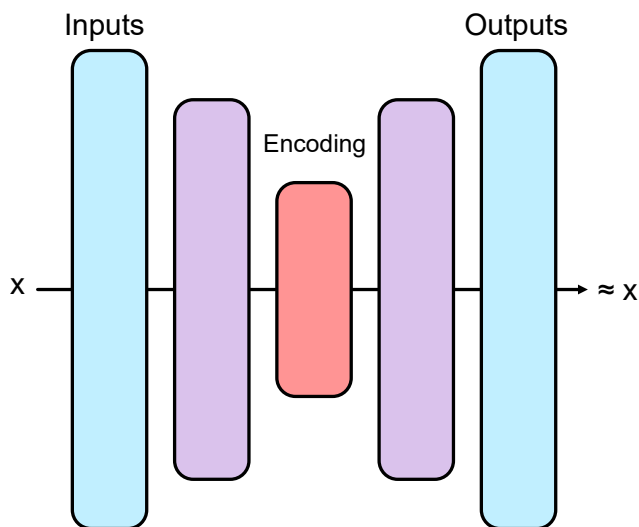


Figure 2.12 A basic Auto-Encoder architecture.

2.3.2 Recurrent Neural Network (RNN)

RNNs are a type of ANNs that possess a unique ability to analyze time series data, enabling them to work on sequences of arbitrary lengths. Figure 2.13 showcases how, at each time step t , a *recurrent neuron* receives two main inputs: the current entry x_t and the output of the previous time step, y_{t-1} . This recurrent structure allows RNN models to maintain and utilize information across different time steps, making them well-suited for sequential data analysis. [33]

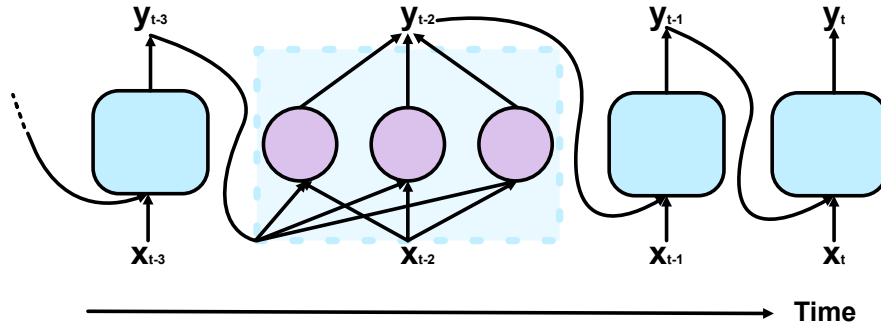


Figure 2.13 A basic Recurrent Neural Network architecture formed with memory cells.

Modern RNN models often employ more sophisticated *memory cells*, such as the Long Short-Term Memory (LSTM) cell. As the name suggests, LSTM cells benefit from long and short-term memory capabilities, which aid in faster convergence and the detection of long-term dependencies in data. This is particularly important in long RNN sequences, as traditional RNNs might struggle to retain such information effectively. [33]

RNNs find extensive application in Natural Language Processing (NLP) tasks, including automatic translation, speech-to-text conversion, and sentiment analysis. The ability to handle sequential data effectively makes them a valuable tool in understanding network traffic data.

2.3.3 Transformer

Both RNN and AE architectures face the fundamental constraint of sequential computation, making it challenging for them to effectively learn dependencies between distant positions in a sequence. To address this issue, the Transformer architecture was introduced by Google [6], leveraging self-attention.

Figure 2.14 demonstrates how the Transformer models follow an encoder-decoder structure. At each time step, the input, which comprises the entire sequence, is transformed into a

low-level representation. The decoder then utilizes this representation to generate an output sequence for each element of the input sequence.

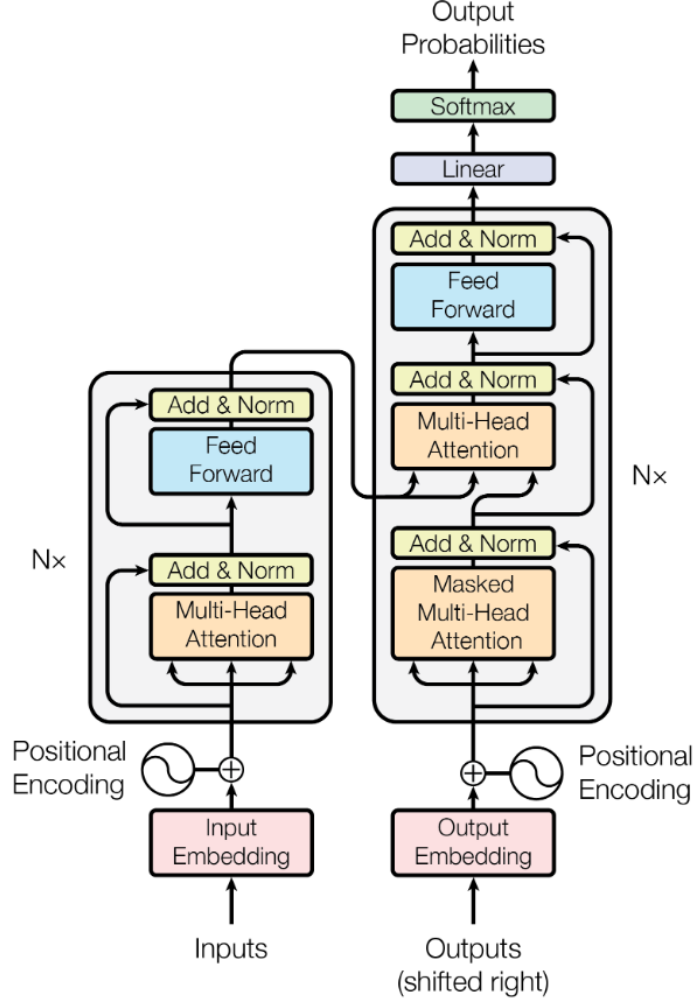


Figure 2.14 Transformer architecture [6]

The concept of *attention* in the Transformer architecture involves the use of three matrices: Q (queries), K (keys), and V (values). These matrices are used to compute the attention scores. The attention mechanism is expressed using the *Scaled Dot-Product Attention* equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.1)$$

where d_k represents the dimension of both the queries and keys.

The Transformer architecture introduces the concept of *Multi-head Attention*. With multi-head attention, the queries, keys, and values are linearly projected multiple h times using

different learned linear projections. These projected values are then concatenated together and projected once more, resulting in the final output values. This approach allows the model to attend to different parts of the input sequence in a more flexible and expressive manner, capturing complex patterns and dependencies in the data effectively.

To leverage the order of the sequence without relying on recurrence or convolution, the Transformer architecture utilizes *positional encodings*. These positional encodings provide additional information about the position of each element in the sequence and are added to the sequence embeddings. The inventors of the Transformer propose using sine and cosine functions to create the positional encoding, which are defined as follows:

$$PE_{(\text{pos}, 2i)} = \sin(\text{pos}/10000^{2i/d_{\text{model}}}) \quad (2.2)$$

$$PE_{(\text{pos}, 2i+1)} = \cos(\text{pos}/10000^{2i/d_{\text{model}}}) \quad (2.3)$$

where pos is the position, i is the dimension and d_{model} the dimension of the input embedding vector.

The Transformer architecture has shown remarkable success and has become the backbone of many state-of-the-art models in various NLP tasks. Recently, groundbreaking models like BERT [34], ELECTRA [35], and GPT [36] have leveraged pre-trained Transformers, enabling efficient transfer learning for fine-tuning on specific downstream tasks.

2.4 Transfer learning

Transfer learning refers to the practice of reusing layers from a previously trained neural network that has been exposed to a combination of various data sources and/or learning tasks [37]. It not only reduces the amount of training necessary but also speeds up the process [33]. The use of pre-trained models has revolutionized multiple fields, such as NLP [34–36] and computer vision [26, 38, 39]. These referred methods have pushed the idea of transfer learning further. They use multiple pre-training tasks aiming to capture different valuable information from one or many data sources that can then be picked up by a downstream model trained on a more niche task.

In IDS research the potential of this strategy has yet to be fully explored. To the best of our knowledge, our proposed solution, TRIPT-IDS, is the first to explore the application of pre-training a Transformer-based model specifically for intrusion detection.

2.5 Cybersecurity

2.5.1 Intrusion detection system

With the continuous surge in data utilization within computers and over networks, traditional techniques for distinguishing normal behaviour from abnormal behaviour have become challenging, if not impossible, for security experts to employ. IDS serve as software or hardware tools designed to detect and report cyber threats to mitigate their impact. The fundamental role of an IDS involves the monitoring of data traffic, which consequently allows for its classification into host-based or network-based categories [19]. *Host-based Intrusion Detection System (HIDS)* centers its attention on the monitoring and analysis of log files and inbound/outbound traffic of host computers [40]. In contrast, *Network-based Intrusion Detection System (NIDS)* is designed to detect cyber threats by inspecting network traffic using a sniffer [19, 40]. The strategies adopted by these IDSs can then be categorized as signature-based or anomaly-based [18]. *Signature-based IDS* relies on prior knowledge of attack scenarios stored in a database as signatures, enabling the mapping of events to specific patterns. However, this strategy's reliance on a predefined pattern database renders it incapable of learning new attack types [18, 40]. *Anomaly-based IDS*, on the other hand, delves into the system's normal behaviour and flags any deviations from it as anomalies or intrusions. This approach, however, is susceptible to a higher rate of False Positive Rate (FPR) since it might categorize infrequent activities as anomalies [18, 40].

Given the diverse hardware components in an aircraft, often from varying manufacturers, implementing a unique HIDS for each component would be impractical. Thus, our research focuses on NIDS implementation. While a signature-based IDS may offer fast response times and clear comprehension of cyber threats, it necessitates prior knowledge of attack patterns. As mentioned in Section 1.3, the field of avionic cybersecurity is relatively new, with no recorded or shared instances of cyber attacks on real avionics networks. Consequently, an anomaly-based IDS offers heightened robustness and adaptability, catering to previously unseen attacks (zero-day attacks) or attack patterns that deviate from known laboratory scenarios.

2.5.2 Cyberattack technique

By its definition, cybersecurity aims to ensure three main properties: integrity, confidentiality, and availability [10]. A threat represents a potential violation of any of these properties. In this subsection, we introduce cyberattack techniques that are pertinent to the upcoming state-of-the-art studies. Some threat models incorporate cyberattacks derived from at least one of these techniques:

Spoofing: These techniques exploit the faults and vulnerabilities in the authentication logic of the network by deceiving others about its identity [41]. Some concrete spoofing attacks on aircraft systems involve manipulating GPS signals, leading military drones astray and causing them to perform various undesired actions [42]. Another type of spoofing attack targets fake ADS-B messages. ADS-B broadcasts beacons containing the aircraft’s identity, position, speed, heading, and other data points. These broadcasts are used in real-time to alert nearby aircraft and ground control for air traffic management, air traffic alerts, and collision avoidance. Several studies have demonstrated the feasibility of creating phantom aircraft and impersonating other legitimate aircraft with equipment costing as low as 2,000 USD, which is publicly available to carry out both spoofing and denial-of-service attacks [43].

Fabrication: Similar to spoofing, the fabrication technique involves injecting realistic fabricated messages into the network, which are then processed by the systems. In contrast to spoofing techniques where malicious actors impersonate and replace legitimate actors, the fabrication technique entails sending fabricated messages alongside authentic ones to induce confusion [8].

Fuzzing: To expose and exploit vulnerabilities in a protocol, an attacker will randomly inject numerous messages with arbitrary data. Fuzzing can be viewed as a form of brute-forcing the application in order to cause it to crash [44].

Replay: One straightforward method for inflicting varying levels of damage is to employ message replay. In essence, malicious actors can use the replay technique to retransmit messages that were previously sent, thereby transmitting contextual and temporally relevant messages that have the potential to induce sufficient confusion, ultimately resulting in data leakage [8].

Denial of Service (DoS): As the name suggests, denial-of-service techniques aim to prevent users from accessing the resources of a device, network, or system [41]. They are among the most common, as they can be executed easily by utilizing low-level

jamming techniques to exhaust the BUS resources. If the threat actor gains physical access to the bus, the implementation can be as straightforward as flooding the bus with standard messages, introducing interference, noise, and inducing errors in components decoding [32]. The potential consequences of such attacks can be catastrophic; critical systems such as the wings, motor, weapons (in the case of military aircraft), and crucial sensors may be completely cut off from their intended functionalities.

Physical Tampering: These techniques encompasses situations where malicious actors gain physical access to deployed components and tamper with them [7]. Physical tampering can occur intentionally, but it can also happen inadvertently due to human error, which might be a result of social engineering techniques. Insider assistance can be exploited to bypass various security measures and is often the easiest entry point for attackers [1]. Tampering can also happen through compromising the supply chain. The simple fact that millions of components are purchased from various levels of sub-suppliers is often overlooked. It is virtually impossible to track such complex supply chains to detect potential threats, encompassing both hardware and software components [14]. A notable example of an oversight happened in 2010 when the U.S. Navy purchased 59,000 counterfeit electronic equipment and other military components from foreign sources [14].

Eavesdropping: These techniques primarily focus on exploiting vulnerabilities related to confidentiality properties, which involves the theft of information transmitted via a network. One rudimentary yet known eavesdropping attack on avionic systems is achieved through ADS-B messages, which allows tracking of any aircraft equipped with ADS-B transponders [14]. Such information can be easily misused for malicious tracking of important figures or preparation of sabotage attacks.

In Subsection 4.1.2, we elaborate on our adversary model, going over the different cyberattacks that are found in our work. In essence, these attacks exploit software and informational vulnerabilities rather than hardware ones and use a variety of spoofing, fabrication, and DoS techniques.

CHAPTER 3 LITERATURE REVIEW

In this chapter, we provide an overview of studies that we consider essential for understanding the current state of IDS techniques, both in a general context and within the realm of avionic applications. We commence by presenting studies that have introduced DL-based IDS, focusing on relevant techniques and breakthroughs. Following that, we review studies that have proposed IDS solutions tailored specifically for avionics protocols. Finally, we conclude the chapter by discussing alternative countermeasures that warrant attention and provide a general summary of the state-of-the-art.

3.1 Deep Learning-based Intrusion Detection System

3.1.1 Comparison of Machine Learning and Deep Learning techniques

Before moving forward with DL techniques, it is important to consider why ML techniques, which have a simpler structure, are easier to interpret, and require less execution time, have been put aside. Classic ML approaches include decision trees, linear regression, Naive Bayes, and Support Vector Machine (SVM) [18]. However, the biggest limitations of these approaches lie in the need for human expertise and interaction, as well as the requirement for recurrent retraining to maintain effectiveness [18]. Additionally, these strategies often require extensive data processing and labelling, making them unsuitable for large-scale problems. Consequently, many ML-based IDS systems suffer from a high FPR and low detection rate [18]. Several studies have demonstrated that DL techniques can facilitate deeper analysis and faster identification of anomalies, promising better performance against low-frequency attacks, which are often the real challenges in real-world scenarios [45, 46].

3.1.2 Recurrent Neural Network based IDS

Due to the sequential nature of network traffic, it is not surprising that RNN techniques have been at the core of innovation in addressing IDS problems. By utilizing their internal memory, Recurrent Neural Networks can effectively model temporal correlations of security attacks and malicious behaviours [18].

The work in [44] introduces a LSTM-based approach for intrusion detection in modern automobiles, specifically focusing on the Controller Area Network (CAN) protocol utilized within them. Just like aircraft, automobiles lack basic security features like authentication and authorization mechanics, leaving them susceptible to various attacks. The research team

produced their own CAN bus attack datasets from an actual car, injecting three different types of attacks: DoS, Fuzzing, and Spoofing. This approach brings the training and validation process closer to real-world scenarios and actual vehicle communication architectures. The use of RNN-based approaches proved to be quite efficient at detecting attack patterns within the sequential nature of the protocol. While displaying promising performance, this IDS solution does have some limitations. Notably, it was not tested on unknown attacks, which in a real-world scenario are most likely the types of attack encountered, since new methods are continually evolving in the realm of cyber threats. Another constraint of the LSTM model is the significant computational resources required for both training and inference. This impacts its training time as well as its response time, which is crucial in critical infrastructure.

A significant challenge in anomaly detection is the trade-off between achieving a low FAR and a high True Positive Rate (TPR). Anomaly-based IDS systems often contend with higher FAR due to the difficulty in accurately generalizing network traffic patterns [18]. In essence, an immense volume of data would be necessary to comprehensively represent the entire network traffic space. To enhance performance in this regard, [47] proposed a two-stage hybrid framework. This approach employs AE for optimal feature extraction, followed by Bi-LSTM, an architecture that employs two LSTM models, one for the forward pass and the other for the backward pass of the sequence, for anomaly detection. The AE learns a low-dimensional representation of the incoming traffic, which is then input to the Bi-LSTM classifier. Reducing the input dimensions enables the IDS to improve its ability to generalize the traffic, leading to enhanced accuracy, TPR, and FAR. However, due to the bidirectional pass of the Bi-LSTM architecture, this solution suffers from increased training and inference times. The proposed framework was trained and tested exclusively on the NSL-KDD dataset [48], and its performance in real-world scenarios with unseen attacks remains unexplored.

3.1.3 Transformer-based IDS

Since its disruptive introduction in 2017 [6], the Transformer model has been pivotal in driving virtually all significant advancements in the field of NLP [34–36]. While its primary applications have included tasks such as translation, summarization, sentiment analysis, text generation, and conversational chatbots, its performance in anomaly detection within text has demonstrated superiority over traditional methods [49]. A key element contributing to the success of models such as BERT [34] and GPT [36] is the Transformer’s capacity to learn through self-supervised task—a learning approach similar to supervised learning that

employs the input itself as the target rather than utilizing fixed labels. Additionally, the Transformer’s ability to facilitate knowledge sharing among models through transfer learning has been crucial to its accomplishments and versatility [34, 36].

In the last two years, Transformer-based IDS solutions have begun to emerge, aiming to address ongoing issues in the field. [50] focused on mitigating the lack of interpretability in existing ML and DL models, which makes it difficult to comprehend and isolate anomalies. They introduced the Network Transformer (NeT), a model that combines graph representation for enhanced interpretability with a Transformer model to transform network traffic into a latent representation. For binary anomaly detection, the NeT model employs one of three algorithms: Local Outlier Factor (LOF), One-Class SVM (OCSVM), and AEs—with AEs producing the lowest FPR. The training of their models was carried out without labels, utilizing the self-supervised learning task of predicting the next packet in a given sequence for the Transformer. The three prediction algorithms are unsupervised and thus do not necessitate labels either. NeT demonstrated its effectiveness in anomaly detection within an Industrial Control Systems (ICSs) network. The data for their evaluation was provided by the Idaho National Laboratory and encompassed two attack types (Flood Attack and Scan Attack) and two anomalies (failed authentication and setting changes).

To steer away from the prevailing trend of employing deeper and computationally expensive DL models, [51] propose a simpler approach utilizing the Transformer architecture in the manner of a AE with a Softmax layer at the end for anomaly classification. Essentially, their approach shares similarities with [47], with their AE Bi-LSTM-based IDS. However this time, with the transformer-based feature extraction, dimensionality reduction, and self-attention mechanisms. Their proposed solution, named RTIDS, was evaluated using both the CICIDS2017 [52] and CIC-DDoS2019 [53] datasets. RTIDS managed to outperform classic ML and DL algorithms (such as SVM, RNN, LSTM, and DNN), exhibiting significant improvements in F1-Score metrics.

A notable limitation commonly observed in NIDS research is the lack of consideration for real-world scenarios, where emerging unknown attacks frequently occur. This concern is highlighted in [54], where the performance degradation experienced by previous solutions when trained on the CIC-IDS2012 [55] dataset but tested on the more recent CIC-IDS2017 dataset is illustrated, revealing the discrepancy between training and real-world conditions. To address this disparity, they propose an innovative approach that leverages the multi-head self-attention mechanism (MHSA). Their methodology involves classifying attack types using a flow-embedding technique, which is constructed using a byte encoder akin to the Transformer’s encoder component. Furthermore, a packet and flow encoder, incorporating additive

attention, is employed. The study focuses on detecting new attacks that exhibit specific characteristics shared with known attacks used during training. However, they acknowledge the challenge of detecting unseen attacks, which bear minimal resemblance to previously encountered attacks during training. Remarkably, the MHSA model surpasses even the performance of the best ensemble detector, showcasing its superiority in various attack types. The dearth of high-quality datasets in the realm of cybersecurity exacerbates the demand for an IDS solution capable of robust generalization. This necessity is particularly pronounced in cases where attack examples are overly simplistic, artificially introduced, or even contrived without any real-world occurrences.

3.2 Intrusion Detection System for avionics system

Within the domain of aircraft as a System of Systems where the aircraft is considered as a box of system, intrusion detection techniques can be broadly categorized into three distinct areas. The first involves Flight Data anomaly detection approaches [56], which entail the analysis of data collected post-flight and are typically employed subsequent to an incident occurrence. Analog-based solutions [7, 9, 57] constitute another category, aiming to acquire knowledge of hardware fingerprints to discern malicious devices. These solutions often function as integral components within comprehensive IDS systems. The third and pivotal area revolves around real-time network traffic data [7–10, 58] for intrusion detection, serving as the central focus of our research.

3.2.1 ARINC 429 IDS

The absence of source authentication in ARINC 429 exposes significant vulnerabilities to its security [29]. Although gaining access to a civil aircraft is no straightforward task, potential risks such as supply chain compromises and human/engineering errors can introduce rogue devices into the aircraft. These devices can transmit malicious data without undergoing verification processes.

In this work, [57] introduces an IDS based on the concept of hardware fingerprinting. This technique involves creating hardware fingerprints using a set of analog features, which are then utilized in an anomaly detection framework. The system can identify a malicious hardware component by detecting deviations in its hardware fingerprints from those of the legitimate hardware. Notably, this approach enables real-time detection, even before the aircraft has departed from the airport, as any modifications to the hardware trigger distinct electric signal alterations, resulting in changes to the fingerprints. While this solution boasts a

remarkable achievement of near-zero FAR and rapid detection, it does require an initial setup and calibration process for new hardware components and environmental contexts. As a result, scaling this approach would necessitate significant efforts, limiting its use.

3.2.2 Ethernet-based Protocol IDS

Given the heightened connectivity present in the latest generation of aircraft, predominantly facilitated by Ethernet-based protocols, the imperative for a robust NIDS becomes increasingly pressing. [31] introduced a IDS framework for the ASCB-D protocol. They generated flow-based data utilizing the same tools employed to create the datasets: CIC-IDS2012, CIC-IDS2017, and CIC-DDoS2019. Subsequently, they subjected their framework to testing against three distinct types of attacks: portscan attack, Distributed Denial of Service (DDoS) attack, and a combined portscan attack followed by a DDoS attack. While their research demonstrated the viability of a IDS designed for Ethernet-based avionics, it should be noted that the ML-based algorithms employed were of a rudimentary nature. These algorithms have already been proven to be outperformed by other strategies within standard Ethernet environments [18, 45].

3.2.3 MIL-STD-1553 IDS

The focus of IDS research has predominantly gravitated towards the military protocol MIL-STD-1553. This inclination can be attributed to various factors, including its designation as a military protocol. Notably, this domain has been further accentuated by the availability of publicly accessible datasets [10, 59] tailored specifically for IDS tasks. These datasets play a pivotal role by enabling research teams with limited resources to engage in the exploration of IDS challenges. Furthermore, these datasets serve as a benchmark for evaluating the performance of different solutions, offering a standardized means of comparison.

In line with this, the Department of National Defence and the Canadian Armed Forces (DND/CAF) have initiated a call to action. Although currently closed, the primary objective of this initiative is to seek innovative and effective IDS solutions for avionics networks and other bus technologies pertinent to the aerospace and space sectors [60]. This call has gathered significant response, resulting in the submission of ten projects from diverse participants, ranging from private enterprises to academic institutions. Notably, these projects have collectively received an average of \$500,000 in funding. Additionally, this collaboration between government agencies and private industry underscores the potential for fruitful outcomes in addressing complex challenges within the avionics domain.

[7] were among the first to address the security vulnerabilities of MIL-STD-1553 and propose a ML-based IDS, which partly contributed to motivating the government of Canada’s call to action [60]. Their IDS comprises two modules: a RT authentication module and a sequence-based anomaly detection module. Figure 3.1 shows the two-step process in which the RT authentication module identifies unauthorized data transfers:

- **Legitimate Behaviour Filter:** Utilizes hardware fingerprints from a known set to create clusters of legitimate signal patterns. An alert is triggered if an incoming signal does not fit within any legitimate cluster.
- **RT Classifier:** Using the data from the previous step, a classifier is trained to predict which RT sent the signal and compares it to the RT address specified in the message. This step allows the identification of impersonators.

The sequence-based anomaly detection module focuses on validating whether a message follows the predefined bus behaviours. A Markov chain model serves as the foundation for the detection mechanism. Their solution was evaluated on a custom testbed composed of off-the-shelf hardware and exhibited high detection rates based on real-world hardware and system training. While the ability to integrate the IDS into existing systems without topology changes is emphasized as a significant advantage, the necessity to retrain the IDS when adding, updating, or removing components present challenges for ensuring the integrity of newly added components.

Following the publication of a dataset of RT spoofing attacks on MIL-STD-1553 traffic by [59], two subsequent studies have incorporated it into their evaluation processes. [46] focused on evaluating the performance of various ML algorithms in a binary anomaly detection task. The assessed ML algorithms included ANN, Bi-LSTM, One-Class SVM, Isolation Forest, Minimum Covariance Determinant (MCD), Local Outlier Factor, and AWS XGBoost. Among these, only their One-Class SVM performed well. The Isolation Forest, Minimum Covariance Determinant, and Local Outlier Factor exhibited varying degrees of performance, correctly classifying some normal samples while struggling with malicious ones. The ANN and AWS XGBoost detected all anomalies but consistently triggered large numbers of false alarms, indicating a lack of learning. Although the Bi-LSTM failed to detect any malicious traffic, it did not produce false alarms. The suboptimal performance of these algorithms can be partially attributed to the scarcity of anomalous instances in this dataset.

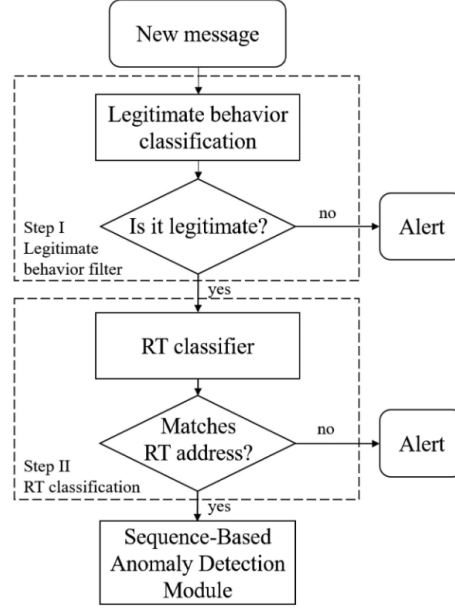


Figure 3.1 *Stan et al.* IDS process [7]

Building upon the work of [7], [8] proposed a two-layer IDS named TLP-IDS, based on time, sequence logic, and historical data. Figure 3.2 display the order in which the incoming traffic goes through each layer. The first layer operates in a similar fashion to the sequence-based step of [7] IDS, utilizing a Markov decision process (MDP) derived from Reinforcement Learning (RL) to learn logical relationships between sequences. The second layer focuses on the numeric data recorded by the BM. The authors employed an online sequence extreme learning machine (OS-ELM) combined with the Weibull distribution function for prediction and detection. The decision to use online learning algorithms allows the IDS to adapt and improve over time during its usage. TLP-IDS demonstrated improved performance compared to [7] IDS, yet it still faced challenges with spoofing attacks, exhibiting an average FAR of 5.1, which is relatively high for critical infrastructure. This highlights limitations in handling adversarial samples generated by sophisticated attackers.

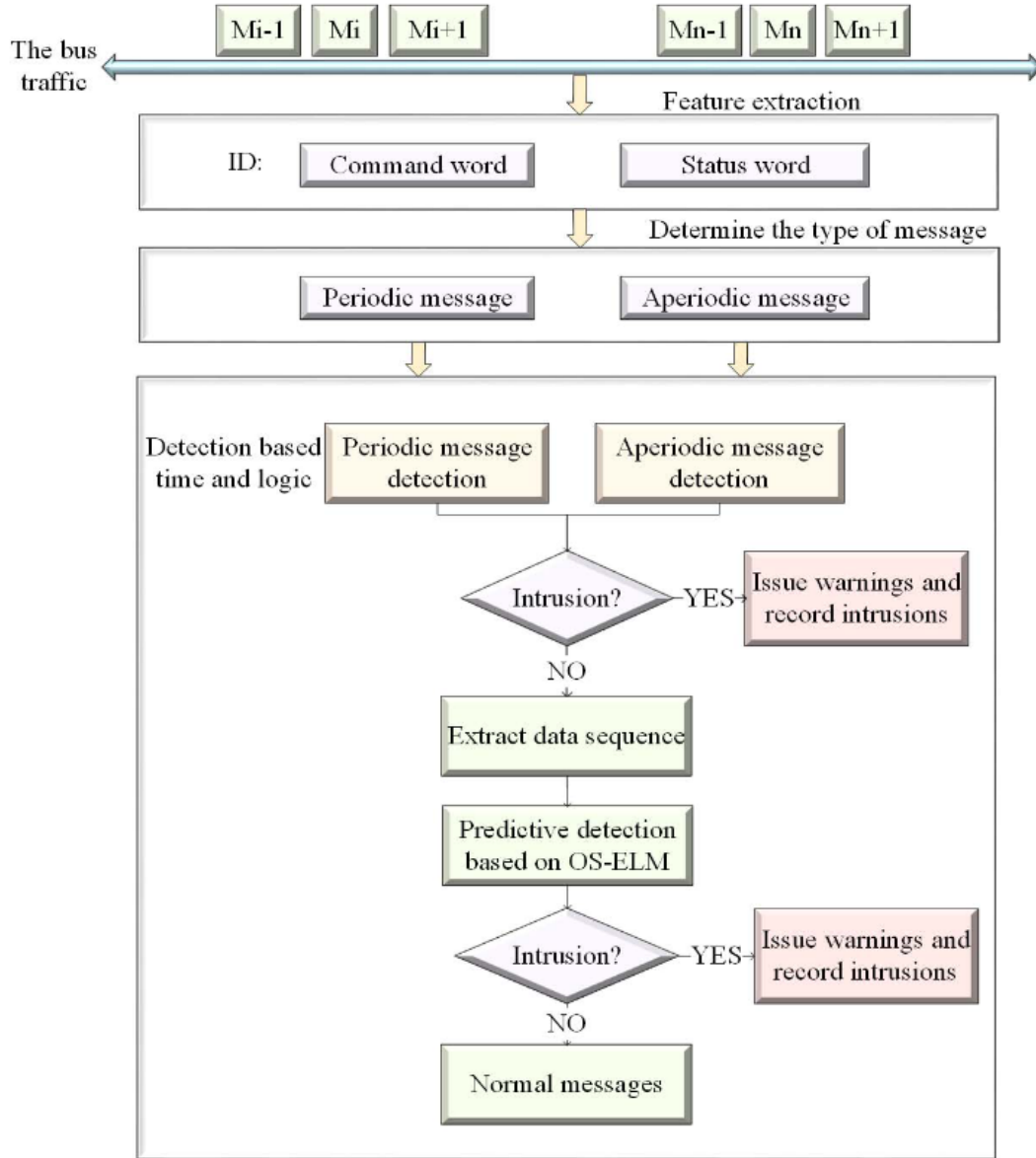


Figure 3.2 TLP-IDS process [8]

With the inherent complexity of avionics systems, an IDS striving to provide comprehensive protection against diverse threat models must extend beyond traffic data analysis. It should also encompass the integrity and legitimacy assessment of both hardware and software components. Furthermore, effective communication with users, such as pilots or ground personnel, is imperative to convey critical information requiring immediate action to avert potential incidents. [9] directly tackle these challenges with their IDS consisting of four modules, referred to as AnoMili. Figure 3.3 illustrates the architecture and sequence of each of these modules. The first module employs an AE to analyze voltage signals captured from

the bus, enabling the detection of unauthorized devices even if they are passive or silent. The second module incorporates a Device Fingerprinting Mechanism utilizing a Convolutional Neural Network (CNN)-based classifier, effectively countering spoofing attacks. As its third module, AnoMili integrates a Context-Based Anomaly Detection Mechanism that employs unsupervised LSTM AE, trained solely to reconstruct benign traffic and produce an abnormal score equivalent to a reconstruction error.

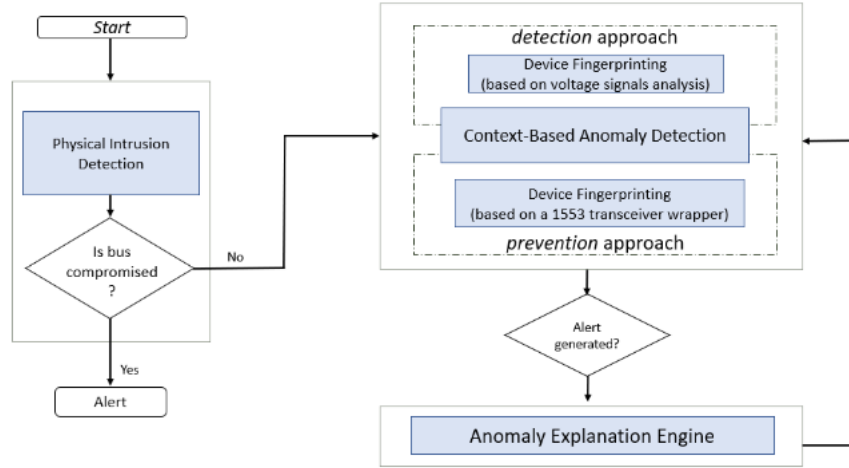


Figure 3.3 AnoMili architecture [9]

Figure 3.4 depicts the architecture of the Anomaly Explanation Engine. This engine automatically generates real-time explanations for detected anomalies, empowering users to discern the attacker's intent and undertake appropriate actions. This is achieved through three distinct modules:

- **Most Influential Feature (MIF) Calculator:** Identifies the feature contributing the most to the abnormal score. This involves supervised training of a decision tree that forms a Boolean expression elucidating the relationship between the input feature and the final decision. The SHAP TreeExplainer method is then employed to calculate the MIF.
- **Device Identifier:** Leveraging device fingerprinting, this module identifies the compromised device and determines the sender of the anomalous message.
- **Neural Machine 1553-Message Translator (1553-MNT):** This module employs a translation model to convert aircraft operations into human-readable language easily understood by the cabin crew.

While the lack of transparency and elaboration regarding the dataset generated on their custom testbed hinders comprehensive performance validation of the approach, it is evident that AnoMili showcases versatility and the ability to bridge the gap between human understanding and IDS operations.

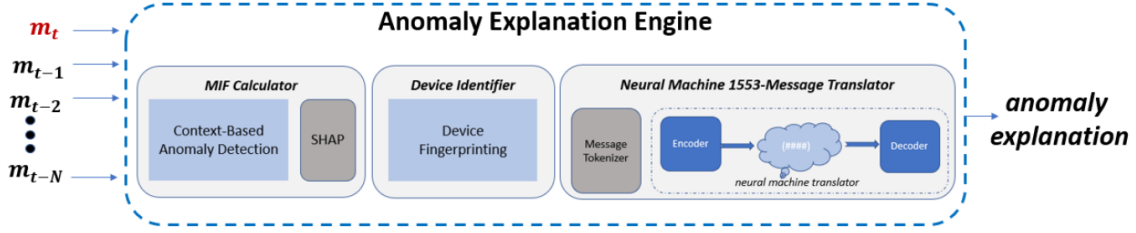


Figure 3.4 AnoMili’s anomaly explanation engine architecture [9]

The scarcity of available datasets stands as a primary limiting factor in avionic cybersecurity research. This scarcity can be attributed to the critical and sensitive nature of the domain, leading commercial leaders to retain resources for private use, thereby restricting shared information. Consequently, numerous research labs resort to constructing their own testbeds using off-the-shelf hardware to generate synthetic intrusion data. Regrettably, certain research efforts choose not to release their datasets to the public, impeding comparison and standardization efforts. In response to this challenge, [10] introduce an open-source simulator software named OD1NF1ST. This software streamlines the collection of long-term, high-resolution data for IDS development related to the MIL-STD-1553 protocol. The simulation system harmoniously interfaces in real time with an IDS and facilitates the injection of ten distinct types of attacks. In addition to the software, the authors propose a DL-based IDS architecture. Figure 3.5 summarized the architecture: (1) Word Embedding and Convolution, where observed words are transformed into sub-tokens and mapped onto embedding vectors. A 1D Convolution operation with Softmax activation assigns probability values to each token. (2) Top-N Selection identifies the most pertinent tokens based on their probability scores. (3) Bi-LSTM, a two-layer model, exploits contextual insights from preceding and succeeding tokens to make final predictions regarding the presence of anomalies or attacks. Notably, a substantial limitation of their approach surfaces when the IDS encounters unseen attacks, rendering specific attack types entirely undetectable.

In our research, we harnessed this software to generate our own datasets, in conjunction with their publicly accessible dataset. Furthermore, the authors have presented the architecture of the IDS solution alongside the OD1NF1ST software. We employ their solution as a baseline for comparison with our approach. In Section 4.1, we go deeper into the data generation

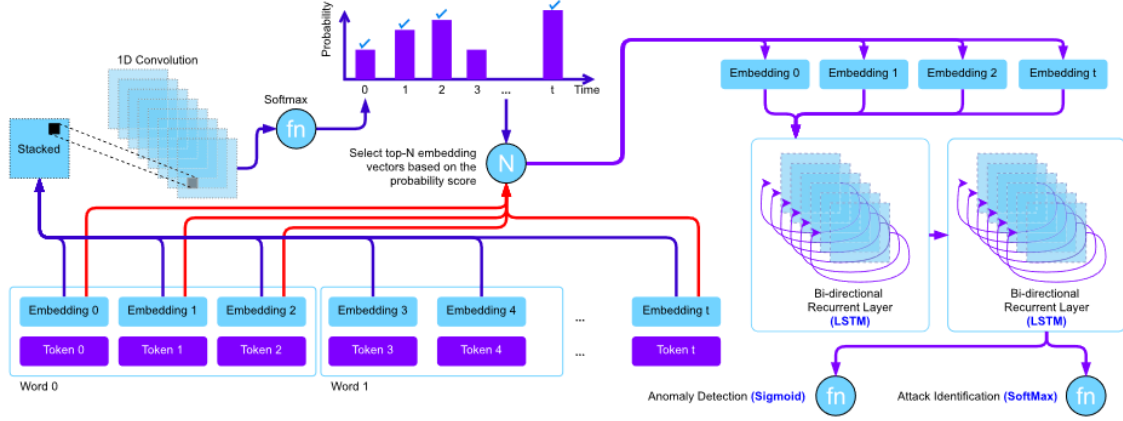


Figure 3.5 OD1NF1ST IDS architecture [10]

process, examining data distribution, and elaborating on each attack type.

[58] present AdaptIDS, an adaptive IDS that capitalizes on advanced DL techniques, specifically LSTM. The authors formulate two distinct stacked ensembles: a homogeneous ensemble and a heterogeneous ensemble, employing the elucidated base models as foundational components. The homogeneous ensemble integrates two LSTM dense base models, while the heterogeneous ensemble encompasses a dense LSTM model and a shallow LSTM model. These ensemble models are trained as meta-learn models, learning the optimal manner to amalgamate predictions from the underlying models for enhanced accuracy. Though AdaptIDS appears to surpass the approach in [10], direct comparison between the methods was unfeasible due to the unavailability of the source code for AdaptIDS. Nevertheless, [58] contribute valuable insights into the performance and efficacy of their proposed AdaptIDS system. It is crucial to acknowledge that the trajectory towards deeper and larger models might entail challenges when integrating AdaptIDS into aircraft systems, considering the intrinsic limitations and constraints of such systems.

3.3 Alternative countermeasures against avionic cyber-attacks

This section presents alternative or complementary approaches to IDSs for strengthening the cybersecurity of aircraft. Since the underlying end goal is to achieve a fully secured aerospace, multiple measures need to be put in place as no one system can cover all threats. We did not work on these measures in our research as our focus was on improving avionic IDSs. However, we believe that understanding these approaches is valuable for future integration work.

3.3.1 Encryption

An issue repeatedly highlighted in the cybersecurity analysis of avionics systems is the absence of authentication in communication, leaving room for malicious actors to exploit this vulnerability [1, 14, 16]. As a result, various studies have delved into this very issue.

[61] introduced a pioneering pairing-free ADS-B authentication batch verification scheme within an ID-based framework. This novel approach aims to address the shortcomings of current ADS-B authentication schemes, which often exhibit inefficiencies in terms of computational cost and communication overhead. The authors solely provide theoretical proof of the scheme’s security against chosen message and identity attacks within the random oracle model, along with evidence of its improved computational and communication efficiency.

Additionally, [62] introduced the Avionics Trusted Domain Chain (ATDChain), a blockchain-based trusted authentication model. This approach leverages the Interplanetary File System (IPFS) to store avionics equipment information and generate unique hash values as identity credentials for communication. Ethereum, a decentralized blockchain, is utilized for system registration and certificate creation through smart contracts. Through the incorporation of blockchain and cryptographic methods, this solution ensures the traceability, security, and immutability of communication records. However, the real-world implementation presents challenges, such as interoperability with existing avionics systems.

Despite the heightened data security that encryption offers, its integration into current avionics systems remains intricate. Implementation could require a comprehensive overhaul of communication protocols for each aircraft, potentially introducing complexities that need to be managed.

3.3.2 Ontology

Ontologies have been developed as tools to accompany IDS implementations, functioning as a language for recognition, reporting, and correlation of computer attacks. Consequently, an ontology serves as a formal description of a particular domain of knowledge and facilitate semantic interoperability and integration within organizations across various domains [63]. Often, ontologies fulfill the role of domain experts, bridging gaps between different domains and enabling IDS alerts to be comprehensible to non-expert users [64].

[1] propose an extensive taxonomy for avionics cybersecurity, approached from an adversary’s perspective. They classify techniques into relevant categories based on the lifecycle stages of adversarial attacks. This taxonomy, which partly extends from MITRE ATT&CK’s taxonomies, forms the basis for developing such a tool. Building upon this, they present an

ontology that defines the entities through which various threats and actors involved in different attacks can be analyzed. Moreover, other studies have introduced their own ontologies for avionics systems, which can prove beneficial in IDS solutions. Examples include the Avionics Analytics Ontology (AAO) [65], which employs semantic uncertainty to enhance decision-making capabilities, and ATMONTTO [66], developed by NASA, which serves as a general ontology for the avionic domain.

While substantial progress has been achieved in the realm of ontologies for avionic cybersecurity, persistent challenges remain. A common issue across many ontologies is their lack of compatibility with each other, often leading to duplication and inefficiencies [63].

3.3.3 Zero Trust Model

The concept of zero-trust networks has been present since the introduction of modern networks in the late 90s and was notably reintroduced by John Kindervag in 2010 [67]. This framework redefines how network architectures should be constructed, centered on the principle of never trusting and always verifying. In theory, as a system grows in complexity and more actors and devices gain access to it, the likelihood of detecting breaches increases. CCX Technologies published a white paper addressing the requirements for implementing a zero-trust model in avionics systems [12]. They illustrate how the zero-trust approach not only pertains to users and data within the network but also extends to every connected node that interacts with an avionic component throughout its lifecycle. Figure 3.7 provides an in-depth depiction of the extent to which the chain of trust must extend for a single component. While a fully realized zero-trust network is an admirable goal, especially for critical infrastructures like avionics systems, achieving this objective requires substantial effort. It entails the establishment of trust levels for every user, device, and network within a system, along with the implementation of a framework to grant or deny access. It remains crucial to acknowledge that even if the probability of a successful attack is exceedingly low, a tragic event could occur if all the right conditions align. Figure 3.6 depicts how this scenario is often illustrated using the "Swiss cheese" model.

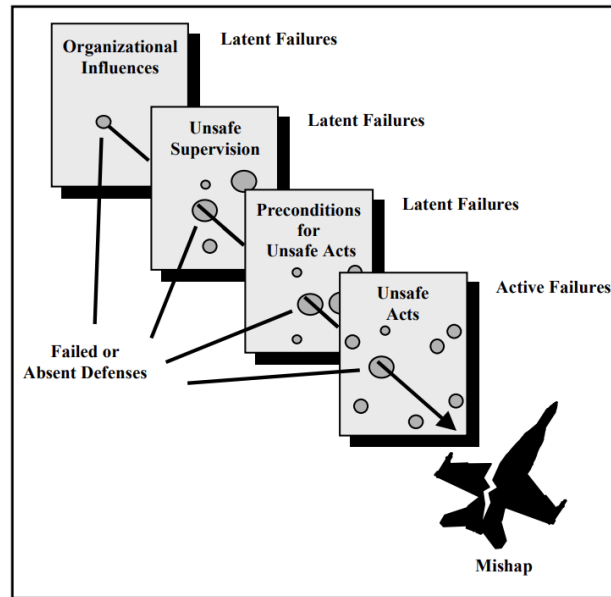


Figure 3.6 The “Swiss cheese” model of human error causation [11]

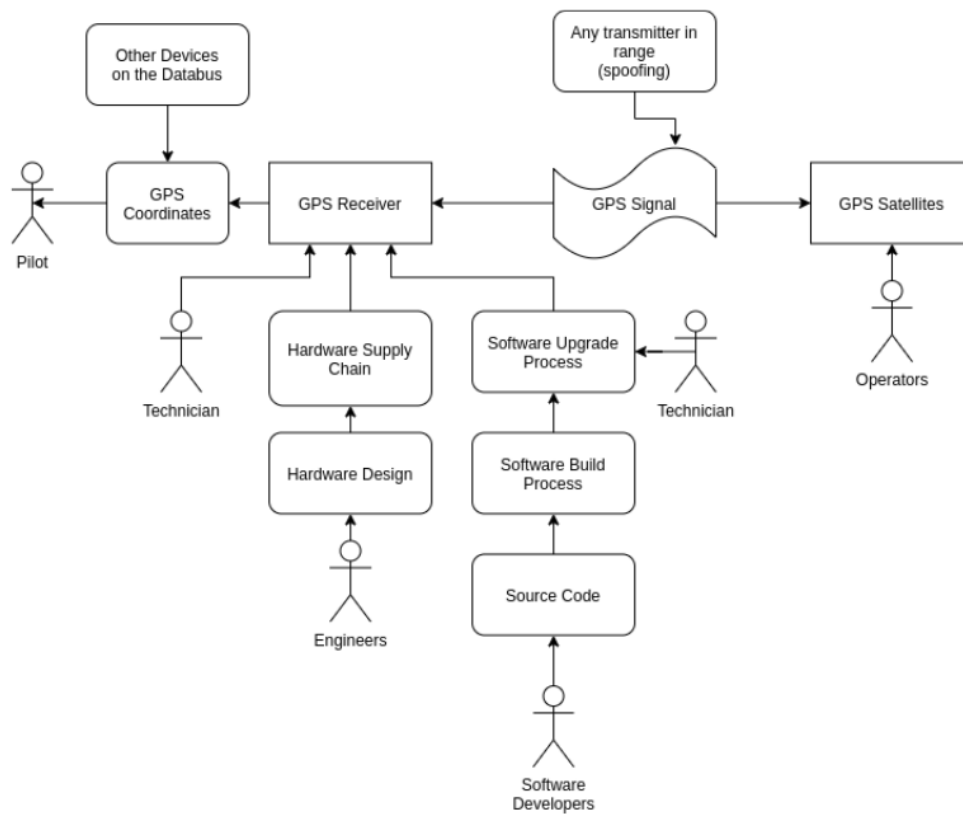


Figure 3.7 GPS chain of trust [12]

3.4 Summary Of the State-of-the-Art

The current state of avionic cybersecurity, heavily relies on safety mechanisms built upon outdated protocols, primarily aimed at ensuring communication reliability and determinism, along with incorporating obfuscation measures. While these safeguards do enhance security to some extent, their core objective is not to guarantee it [16].

Given the relatively recent integration of cybersecurity measures into the avionics sector, several solutions have been proposed to fortify security on multiple fronts. These range from the implementation of encryption schemes for communication, safeguarding data exchanges when aircraft interact with their environment [61,62], to enforcing legitimacy proofs for every node in the avionics chain [12]. The augmentation of safety barriers constitutes one element of the overall strategy. IDSs can be compared to surveillance cameras, vigilantly observing their environment and triggering alerts in case any intruders breach the defences. Nevertheless, the challenge of establishing effective human-machine interfaces that allow seamless communication to crew members who may not possess expertise in cybersecurity persists. This is where ontologies can contribute by bridging the gap in understanding. The formulation of policies and regulations pertaining to the aviation cybersecurity ecosystem is another aspect requiring attention. The development of standardized practices and their enforcement remains vital in ensuring the integrity and security of aviation systems. The full realization of a comprehensive and robust cybersecurity framework within the avionics sector remains a multifaceted challenge that encompasses technological, operational, and regulatory dimensions.

Looking deeper into the state-of-the-art of avionics IDSs, we present a comparison in Table 3.1 among the most pertinent solutions alongside our proposed TRIPT-IDS. Notably, a significant observation surfaces—the vast majority of IDSs are tailored for the MIL-STD-1553 protocol. This prevalence can be attributed to the protocol’s documented vulnerabilities [32], the incentive by government entities [60], and the availability of publicly accessible IDS datasets [27,59], facilitating rapid implementation and validation. Another noteworthy observation pertains to the diversity of data sources employed. Many of these approaches leverage multiple modules to detect various attacks using distinct data sources. Analog data is harnessed by [7,9,57] to discern hardware fingerprints and identify rogue devices within the network. Feature data (e.g., timing, word count, address, etc.) is utilized by [7–10,46,58] to capture the protocol’s logic, distinguishing between anomalous and benign traffic. Content data is analyzed by [7,10,31,58] to verify the legitimacy of transmitted data words, susceptible to manipulation by sophisticated adversaries. In alignment with [10,58], our approach focuses on features and content-based data due to shared dataset.

Where our approach charts new territories is within the algorithm itself and the ensuing training and testing experiments, particularly on realistic scenarios. [7–9, 57] adopt an unsupervised training methodology, exclusively utilizing benign traffic for training. While this strategy appears promising in theory, in practice, extensive training might be essential to robustly learn and generalize normal traffic behaviours to effectively counter unforeseen flight scenarios. [10, 58] adopt a supervised approach, focusing on differentiating between normal traffic and multiple attack types. In their experiments, they cross-validate their model on independently unseen attacks (trained on all other types). Additionally, [58] evaluates AdaptIDS on unobserved flight data during testing. Nonetheless, a comprehensive test combining unseen attacks and flight data is lacking, which would more closely resemble real-world scenarios.

Our proposed TRIPT-IDS prioritizes the learning of an embedding space that clusters abnormal sequences together and normal sequences together with the least possible overlap, essentially yielding a more resilient IDS against previously unseen attacks. We tested our approach on unobserved flight data and subsequently performed cross-evaluation on independently unseen attacks. Additionally, we introduced an innovative approach, testing performance on multiple unseen attacks alongside few-shot scenarios. This showcases the flexibility and generalization capacity of our model. We also contribute to the broader field of IDS research with our incorporation of transfer learning with Transformer-based IDS.

Table 3.1 Comparison of state-of-the-art avionic IDS and TRIPT-IDS

IDS	Protocol	Threat Type	Data Source	Algorithm	Unseen train & test scenarios
[57]	ARINC 429	Tampering	Analog	Local Outlier Factor	Trained on benign physical data only
[31]	ASCB-D	DDoS PortScan	Content	Logistic Regression SVM Random Forest ANN	×
[7]	MIL-STD-1553	Spoofing DoS Tampering	Analog Feature Content	K-means Markov chain	Trained on benign physical data only
[46]	MIL-STD-1553	Spoofing	Feature	ANN Bi-LSTM One-Class SVM Isolation Forest MCD Local Outlier Factor AWS XGBoost	×
TLP-IDS [8]	MIL-STD-1553	Spoofing Fabrication Replay Tampering DoS	Feature Content	Markov decision process OS-ELM	Trained on benign data only
AnoMili [9]	MIL-STD-1553	Spoofing Tampering	Analog Feature	AE CNN LSTM AE	Trained on benign data only
OD1NF1ST [10]	MIL-STD-1553	DoS Spoofing Fabrication	Feature Content	1D Convolution Bi-LSTM	Cross-validation on independently unseen attacks
AdaptIDS [58]	MIL-STD-1553	DoS Spoofing Fabrication	Feature Content	Stacked Dense Bi-LSTM	Unobserved flight data Cross-validation on independently unseen attacks
TRIPT-IDS	MIL-STD-1553	DoS Spoofing Fabrication	Feature Content	Pre-trained Transformer encoder	Combined unobserved flight data and cross-evaluation on independently unseen attacks Combined unobserved flight data and multiple unseen attacks

CHAPTER 4 METHODOLOGY

In this chapter, we expand upon our proposed solution. We commence by elaborating on the data sources, detailing the approach employed for generating and encoding the MIL-STD-1553 data. Afterwards, we introduce the TRIPT-IDS architecture, along with its various training phases.

4.1 Data Source

4.1.1 OD1NF1ST software

In our research, we employed the open-source software OD1NF1ST [68], introduced by [10], to use realistic flight data embedded with synthetic attacks on the MIL-STD-1553 protocol that is procured by it. The OD1NF1ST software stands out for its adeptness in simulating cyberattack scenarios, offering ten distinct vulnerabilities within MIL-STD-1553. These scenarios encompass three facets of system security: confidentiality, integrity, and availability. Notably, due to the absence of authentication mechanisms in the MIL-STD-1553, an intruder who infiltrates the network can transmit words to the bus, impersonating any device. This surpasses approaches such as [7, 9, 46], which cover a limited range of intrusion scenarios, often focused on specific subsets of security paradigms.

The simulator’s architecture is comprised of three primary components: the transmission layer, responsible for emulating raw data transfer; the event system, governing device responses based on transmitted information; and the Interactive Display package, facilitating real-time simulation. The physical layer employs User Datagram Protocol (UDP) for emulating data transmission, while the message layer utilizes a software-based proxy manager to simulate message transmission.

The software employs a pipeline depicted in Figure 4.1 to generate data, either in real time or offline with pre-packaged data for subsequent use. Although our intention was to utilize the real-time functionality of the software, a lack of comprehensive documentation and non-intuitive integration hindered our efforts in this regard. Further work is necessary to address these challenges. Nonetheless, we were able to use their pre-packaged dataset [27] and generate our own data through the offline pipeline. The pipeline’s components involve collecting flight data using Microsoft Flight Simulator 2020 (MSFS) ¹ and the SimConnect

¹Microsoft Flight Simulator 2020. <https://www.flightsimulator.com>.

(SC) Python package ². The simulator interfaces with MSFS's data, mimicking cyclic communication scenarios akin to those seen in military aircraft. This interaction yields JSON files containing word data, payloads, timestamps, and attack labels.

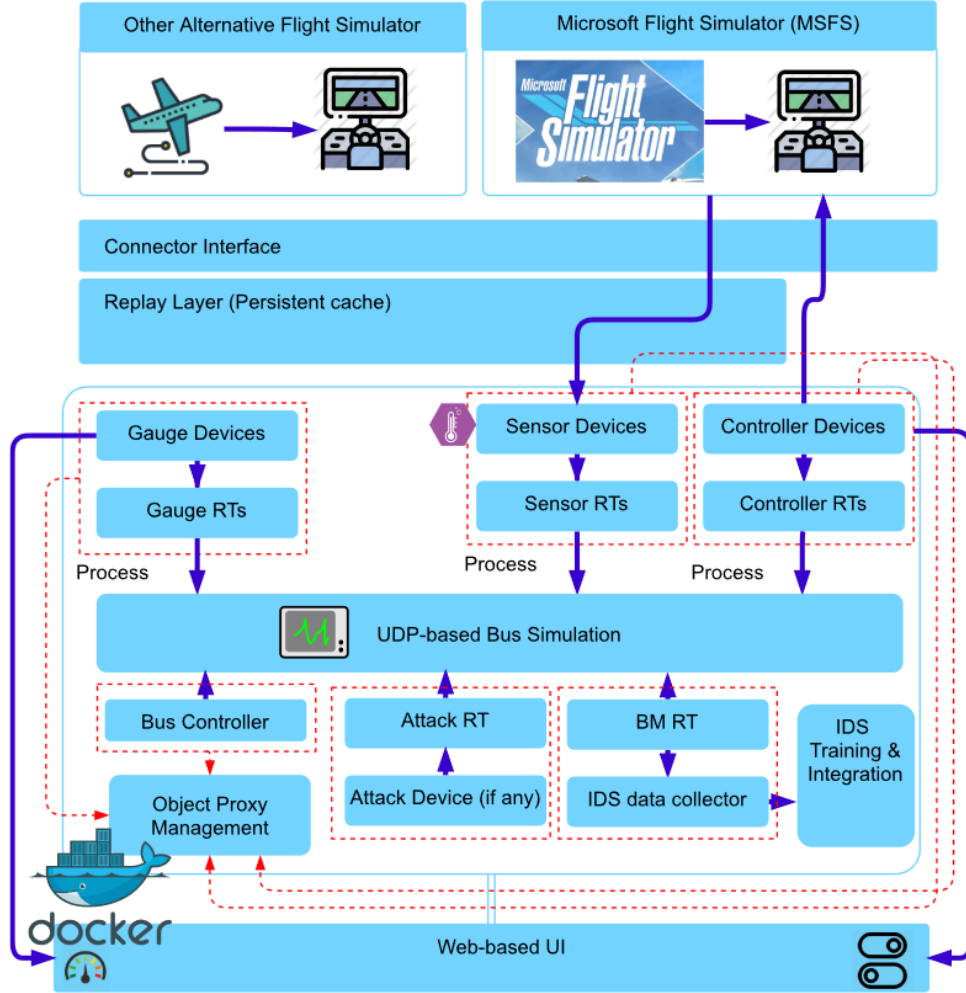


Figure 4.1 Overview of integrated pipeline to process OD1NF1ST data. [10]

4.1.2 Adversary model

As our study utilizes the same data source as [10], we adopted their threat assessment framework for our research. This decision not only enables direct comparisons but also addresses a gap in the cybersecurity research field, which the OD1NF1ST team aims to bridge.

Given that the OD1NF1ST software primarily focuses on the MIL-STD-1553 protocol, which is a part of a highly secure ecosystem, it operates under the assumption that the hardware

²SimConnect API. <https://pypi.org/project/SimConnect/>.

devices on the bus are legitimate. Compromising these devices would require physical access to a military base or the supply chain itself. Consequently, the simulated attacks within OD1NF1ST predominantly exploit software and informational vulnerabilities rather than hardware tampering. The software generates ten attack types, all leveraging the absence of authentication in the MIL-STD-1553 protocol as a catalyst.

The simulated attack types are as follows:

- **Random Word Generation:** A simple Denial of Service attack that disrupts or collides with legitimate messages by spamming the bus with random words or interfering with specific words. This attack can either target a specific RT or all RTs on the bus.
- **Status Word Manipulation:** The attacker sends fake status to the BC to induce device idleness, potentially leading to bus freeze or Man-in-the-Middle (MITM) attacks. This attack can be achieved by targeting either receive or transmit commands.
- **Man-in-the-Middle (MITM):** The attacker poses as a BC, transmitting fake commands to legitimate RTs to initiate unauthorized data transfers.
- **Command Invalidation:** Aims to force an RT into idle mode by sending a fake command word right after a benign one.
- **TX Shutdown:** Involves sending a shutdown command word to disable a target RT, potentially with reactivation via status word response.
- **Desynchronization:** Exploits an RT's clock reset upon receiving a resync word, causing message collisions and bus performance degradation.
- **Data Trashing:** Aims to make an RT ignore data words after a state change by inserting a malicious command word during a normal data transfer.
- **Data Word Corruption:** Injects crafted data into a legitimate transmission to process corrupted words by the devices.

To identify indicators of attacks within the MIL-STD-1553 protocol, two main categories are employed: message timing and message contents. However, the use of timing as an indicator introduces challenges. Different avionics platforms exhibit unique configurations tailored to their specific networks, with variations in harmonic rate groups and polling rates. This diversity hampers the universal use of timing as an indicator. Additionally, manually configured communication cycles in many MIL-STD-1553 networks result in substantial variations, further complicating timing analysis [10].

Conversely, [10] outline three types of message content indicators to discern attack scenarios:

- **Interrupted Bus Communications:** Covers message collisions or freezes caused by attacks such as random word generation, status word manipulation, and poorly executed MITM attacks. Detection relies on identifying recurring messages or the absence of words on the bus.
- **Intercepted Protocol:** Encompasses attacks aiming to insert data into existing protocols without disrupting MIL-STD-1553's operation. Detection mechanisms involve pattern anomaly recognition or scrutinizing content for modifications.
- **Interrupted RT Functionality:** Targets specific RTs to impair their functionality without impacting the entire bus. Detection relies on distinguishing legitimately from malicious commands based on device addresses and transmitted commands.

By utilizing the classification of attack types and indicators described by [10], we categorized the attack types into four overarching categories based on how the attack is executed:

1. **Collision-based DoS:** Involves attacks that flood the network with a large number of random bits, leading to collisions and disrupting benign traffic.
2. **Status Word Manipulation:** Attacks that utilize status words to interrupt normal system operations.
3. **Malicious Command:** Focuses on injecting malicious commands or impersonating the BC to compromise system security.
4. **Word Generation:** Encompasses sophisticated attacks that inject realistic corrupt words to target RTs or the BC, causing them to process intrusive data unknowingly.

Table 4.1 illustrates the categorization of each attack type. It is worth noting that the *Data Word Corruption* attack type is placed in its own category since it is the only attack that aims to inject realistic data words. While both the *MITM* and *Data Trashing* attacks share a similar attack indicator with *Data Word Corruption*, they are performed by masquerading as a BC, whereas the *Data Word Corruption* attack takes on the persona of a RT.

Table 4.1 Summary of the OD1NF1ST generated attacks

Attack Category	Attack Type	Protocol Exploited	Attack Indicators
Collision-based DoS	Collision Attack Against The Bus	None	Interrupt Bus
	Collision Attack Against Target RT	None	Interrupt Bus
Status Word Manipulation	Fake Status Reply to Receive Command	RT-BC, BC-RT, RT-RT	Interrupt Bus
	Fake Status Reply to Transmit Command	RT-BC, BC-RT, RT-RT	Interrupt Bus
Malicious Command	Command Invalidation	RT-BC, RT-RT	Interrupt RT
	TX Shutdown	All	Interrupt RT
	Desynchronization	All	Interrupt RT
	Data Trashing	BC-RT, RT-RT	Intercept Protocol
	Man-in-the-Middle	All	Intercept Protocol
Word Generation	Data Word Corruption	RT-BC, RT-RT	Intercept Protocol

4.1.3 Data Generation

To contribute to cross-study validation, we train and test our approach using data generated by the OD1NF1ST software. To make our study more realistic, we use two separate datasets for training and testing purposes. This reflects the fact that real life flight scenario may not match the one found during training. To push the idea further, we also conduct experiments where the types of attacks in the train and test set are different.

L1NNA Dataset. [27] This dataset is based on data collected from two flights: 1) Frankfurt (FRA) to Toronto (YYZ), and; 2) Montreal (YUL) to Toronto (YYZ). These flights were recorded using the aerospace vehicle Cessna Citation CJ4 plane. The dataset comes divided into Take-off & Landing and Cruise sets. This allows researchers to test approaches on different scenarios or unseen traffic. In our case, we have recombined them since we generated our own dataset for this purpose. **Our Dataset.** We use a similar setup as the one used for the L1NNA's to generate our dataset. We recorded three different flights in MSFS: 1) Montreal (YUL) to Mirabel (YMX); 2) Montreal (YUL) to Toronto (YYZ); 3) Montreal (YUL) to New York (JFK).

Table 4.2 displays the distribution of each attack type's occurrence in both datasets. The frequency of attacks may vary depending on their nature. For instance, TX Shutdown, Desynchronization, and Command Invalidation attacks exhibit a lower distribution due to their reliance on sending specific command words. In contrast, Collision-based DoS attacks have among the highest distribution, as their strategy involves flooding the network with noise. The column "maximum attack sequence length" indicates the longest consecutive sequence length of attack words for each type. This information can be valuable for selecting an adequately sized context window to provide to our IDS model. A quick observation reveals that the datasets are imbalanced in favour of benign traffic. Some studies have employed the Synthetic Minority Over-sampling Technique (SMOTE) to address the imbalance issue

in their IDS datasets [51]. However, we align ourselves with the viewpoint of the AdaptIDS team, emphasizing the significance of preserving this imbalance to mirror real-world scenarios [58].

Table 4.2 Word distribution and maximum attack sequence length in datasets

Label	Attack	LINNA Dataset		Our Dataset	
		Word count	Max. atck. seq. len.	Word count	Max. atck. seq. len.
1	Collision Attack Against The Bus	20783	15	16503	10
2	Collision Attack Against Target RT	7996	12	5846	13
3	Data Thrashing	651	1	256	1
4	Man-in-the-Middle	900	2	565	2
5	TX Shutdown	103	1	28	1
6	Fake Status Reply to Receive Command	2335	2	2122	2
7	Fake Status Reply to Transmit Command	3119	2	4072	2
8	Desynchronization	446	2	241	2
9	Data Word Corruption	2930	29	3806	17
10	Command Invalidation	43	1	9	1
Total Malicious		39306		33448	
Total Benign		1900963		1787448	
Presence		2.03%		1.84%	

Sequence Generation

The OD1NF1ST software’s offline pipeline generates JSON files, each representing recording of up to 5 minutes of flight data. These recordings may or may not contain malicious words. The default sliding window size ($wSize$) implemented in OD1NF1ST’s offline pipeline is 5, which is close to the mean of the maximum sequence length. This window size is used to generate sequences that serve as inputs. These sequences are assigned two labels by OD1NF1ST: one to indicate whether they contain any attack words and, if so, another label specifying the specific attack type. It is important to note that each JSON file recording introduced by OD1NF1ST includes only one type of attack. A restriction imposed in the implementation of the software. The sliding window, also called a context window, allows a sequence-based model to focus its analysis on a more contained sequence of inputs. If the window size $wSize$ is too large, learning long-range dependencies becomes challenging, and if too small, useful relations can be missed [6]. The term "sliding" implies that the sequence vectors are created by selecting, for a timestamp t , the next $wSize$ words. In our case, the value t is incremented by 1 to generate the next vector until we reach the end of the recorded data. This results in a set of vectors, each representing a chronological sequence of words. During the hyperparameter tuning phase (explained in Section 5.4), we found that a $wSize$ of 20 provided better performance for our architecture.

Table 4.3 and Table 4.4 present the new distribution resulting from the sequence generation process. Due to the labelling of sequences as malicious if they contain any attack words, it

is expected to observe an increase in the total number of malicious instances, accompanied by a reduction in benign instances.

Table 4.3 Distribution of attack sequences in datasets following the generation process with $wSize = 5$

Label	Attack	L1NNA Dataset	Our Dataset
1	Collision Attack Against The Bus	67006	62490
2	Collision Attack Against Target RT	33228	26632
3	Data Thrashing	3221	1276
4	Man-in-the-Middle	3037	1855
5	TX Shutdown	515	140
6	Fake Status Reply to Receive Command	11645	10543
7	Fake Status Reply to Transmit Command	14989	19970
8	Desynchronization	1478	824
9	Data Word Corruption	5982	11712
10	Command Invalidation	215	44
Total Malicious		141316	135 486
Total Benign		1794941	1 684 130
Presence		7.87%	7.46%

Table 4.4 Distribution of attack sequences in datasets following the generation process with $wSize = 20$

Label	Attack	L1NNA Dataset	Our Dataset
1	Collision Attack Against The Bus	84813	92755
2	Collision Attack Against Target RT	51774	44529
3	Data Thrashing	12780	5076
4	Man-in-the-Middle	9809	5924
5	TX Shutdown	2060	560
6	Fake Status Reply to Receive Command	42811	37465
7	Fake Status Reply to Transmit Command	48334	59022
8	Desynchronization	4772	2591
9	Data Word Corruption	8648	19034
10	Command Invalidation	860	164
Total Malicious		266661	267120
Total Benign		1654551	1547696
Presence		13.72%	14.72%

4.1.4 Sequence Encoder

Before feeding the data into our model, we encode the traffic words into interpretable sequences. Referring to the three word types of MIL-STD-1553 mentioned in Section 2.2.4,

each type utilizes 16 bits for transmitting essential information. Table 4.5 presents the attributes captured from each word type. The command word’s attributes specify the target system and the command’s mode, while the status word employs various bits to represent different states. The data word uses the full 16 bits for transferring data, equivalent to 2 bytes.

In their studies, [10] introduced an original encoding scheme alongside OD1NF1ST, which was also adopted by [58]. This scheme concatenates the attributes of words into a single vector, maintaining the order they appear in the sequence. Padding is introduced to ensure uniform vector sizes. Although this scheme yielded promising results in their research, we observed that it omits certain metadata that could be valuable. The naive concatenation of attributes leads to a loss of information regarding the word types and the individual meanings of each attribute. As indicated in Table 4.1, certain attacks exploit specific communication protocols. Given that these attacks rely on particular word type sequences, the order in which these word types appear becomes crucial information.

Table 4.5 Attributes of each word type

Word Type	Attributes
Command	Destination address Remote Terminal Destination sub address Word Count/Mode Code
Data	Data
Status	Address Message error bit Instrumentation bit Service request bit Reserved bits Broadcast received bit Busy bit Subsystem flag bit Dynamic bus control bit Terminal flag bit Parity bit

In this study, we propose two new methods of encoding MIL-STD-1553 communications that preserve the overall meaning of the word type and their attributes. Figure. 4.2 illustrates how each encoding works using a basic example, starting by the *original* encoding followed by our proposed schemes *tagging* and *vector* respectively.

Original	[6 0 2 28 5 0 0 0 0 0 0 0 1 1 200 193 5 1 2 28 182 66 ... 0]
Tagging	[257 6 0 2 28 259 5 0 0 0 0 0 0 0 1 1 259 200 193 257 5 1 2 28 259 182 66 ... 256 260] 256 = Padding 257 = Command 258 = Status 259 = Data 260 = Class
Vector	[[0 0 6 0 2 28 0 0 0 0 0 0 0 0 0 0], [0 0 0 0 0 0 5 0 0 0 0 0 0 0 1 1], [200 193 0 0 0 0 0 0 0 0 0 0 0 0 0 0], [0 0 5 1 2 28 0 0 0 0 0 0 0 0 0 0], [182 66 0 0 0 0 0 0 0 0 0 0 0 0 0 0], [$w_c^0 w_c^1 w_c^2 w_c^3 w_c^4 w_c^5 w_c^6 w_c^7 w_c^8 w_c^9 w_c^{10} w_c^{11} w_c^{12} w_c^{13} w_c^{14} w_c^{15} w_c^{16}$] ← Class tag]

Figure 4.2 Comparison of three sequences encoding schemes $wSize = 5$. The *Original* scheme is the one introduced alongside the OD1NF1ST software. The *Tagging* scheme incorporates tags into the sequence to aid the Transformer model’s focus during learning. The *Vector* scheme generates a general word vector encompassing all attributes of different word types.

Tagging Encoder

The Tagging Encoder approach draws inspiration from the BERT methodology, emphasizing the inclusion of specialized tags within sequences [34]. Tagging an input involves introducing distinct tokens into the input sequence before presenting it to the model. These unique tokens serve diverse purposes and enhance the model’s efficacy in executing tasks.

Word Separation: The original encoder processes a sequence of attributes rather than individual words. By incorporating tagging, we can segregate distinct attributes within the sequence. This separation facilitates the model’s comprehension of boundaries between different segments of the sequence, thus averting confusion between words. We introduce four distinct separation tags within the concatenated vector generated by the original methodology. Given that attribute values are derived from bytes, the maximal attribute value remains 255. Consequently, our tags are assigned values ranging from 256 upwards. Before concatenating the content of each word, we append a tag indicative of the word’s type. In lieu of employing 0 as the padding value, we employ a unique

tag to differentiate between padding and a status word comprised entirely of zeros.

Classification Token: We append a *class* tag, which holds significance for tasks such as classification. This token directs attention towards it and encapsulates the semantic essence of the complete input sequence.

The diverse tag values are depicted in Figure 4.2. Although this innovative encoding scheme imparts additional meta-information to the encoding, it retains a critical challenge when presented to the model. Ordinarily, in the context of NLP, unique input embeddings are assigned to words/tokens. Consequently, each word possesses a distinctive spatial representation. However, in both the original and tagging encoding schemes, an attribute value does not inherently maintain its uniqueness across various word types. To illustrate, the attribute of a *Destination Address* in a command word might exhibit the same value as the content of a data word, thereby sharing an identical input embedding. This phenomenon can potentially confuse the model or, at the very least, complicate the learning process.

Vector Encoder

To address the challenge of attribute value uniqueness, we propose an alternative encoding scheme. Instead of representing traffic as sequences of word attributes, we transform the traffic into sequences of words composed of a generic vector representation. Each word is characterized by a vector of size 17:

$$[x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}] \quad (4.1)$$

Values within x_0 to x_1 are allocated for data word attributes, x_2 to x_5 pertain to command attributes, and x_6 to x_{17} correspond to status attributes. Through the vector encoding strategy, every instance of a word possesses a distinctive representation. Implicitly, Figure 4.2 shows how the word type order is encapsulated since attributes pertinent to the word type are populated, while the remainder stays at zero. Similar to the Tagging Encoder, we add a *class* tag either at the start or the end of the sequence to encapsulate the semantic essence of the entire input sequence. In this encoder, it is manifested as a vector of learned parameters. An additional benefit of this encoding approach is the preservation of a constant sequence length based on word count during the transition from an attribute-based sequence to a word-based one. Consequently, the necessity for padding is eliminated. This not only results in more concise sequences but also streamlines model processing.

We evaluated each strategy in Section 5.5.1 and demonstrated how these approaches enhance

the efficacy of our solution. Nevertheless, further enhancements can be applied. For instance, within the tagging scheme, there exists an opportunity to map each attribute to a distinct token range, thereby ensuring its uniqueness. Additionally, there is ample room for research aimed at devising an encoding scheme that facilitates cross protocol translation.

4.2 TRIPT-IDS Architecture

We introduce an innovative approach to enhancing the training of a Transformer-based IDS, thereby improving their generalization and resiliency. This approach draws inspiration from techniques employed by renowned NLP models, such as Google’s BERT [34], ELECTRA [35], and OpenAI’s GPT [36]. These models leverage transfer learning principles, whereby a model is initially pre-trained on certain tasks and subsequently fine-tuned on specific downstream tasks. While these models are often trained across a diverse range of tasks, our research focuses on a singular pre-training task.

The foundation of our approach lies in the transfer learning aspect, where during the fine-tuning step the model is initialized with pre-trained parameters. As a result, both the pre-training and fine-tuning phases share a unified architecture, which we refer to as the TRIPT Module for the sake of simplicity. In the subsequent subsections, we delve into the architecture of the TRIPT Module, elucidate the pre-training task, and outline the fine-tuning task.

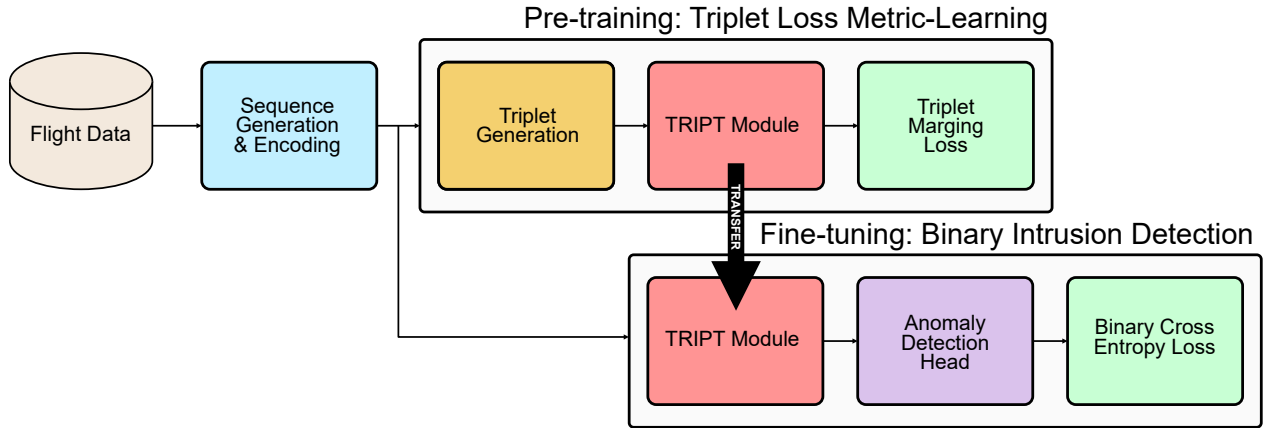


Figure 4.3 Overview of the TRIPT-IDS architecture

4.2.1 TRIPT Module

The module commences by processing a sequence that has been encoded using one of the three schemes detailed in Section 4.1.4. Each element of an input sequence is directed through an embedding layer to be transformed into a vector that is more compatible with the Transformer architecture. The output of the embedding layer is summed with a position embedding, computed as outlined in Section 2.3.3. This addition gives the Transformer model the capacity to discern the positional context of each element within the sequence. Throughout the conceptualization and testing phases, we determined that achieving better results involved inserting the *class tag* at the sequence’s beginning and excluding it from the embedding layer. Following the incorporation of the position embedding, we arrive at the input embedding, which is then propagated through the Transformer.

Similar to the BERT model [34], we employ the encoder component of the Transformer architecture [6]. Unlike the architectures discussed in these studies, our Transformer encoder is of a relatively small size. Following the hyperparameter tuning phase (explained in Section 5.4), our Transformer encoder is composed of 4 layers (L), 1024 hidden units (H) and 8 attention heads (A) compared to the Transformer in [6] with (L=6, H=1024, A=16) and BERT [34] with (L=24, H=1024, A=16). The encoder yields a hidden vector for each element in the sequence.

Since we incorporate a *class tag* in both of our novel encoding schemes, the Transformer is primed to encapsulate the semantic essence of the complete input sequence within the hidden vector generated for the *class tag*. Consequently, we only transmit this hidden vector through a dense layer comprising an MLP architecture with *LeakyReLU* as the activation function. This configuration produces the class logits C_c , in other words, the lower-dimensional representation of the sequence’s type— normal or abnormal in our case.

Figure 4.4 shows the architecture of the TRIPT module, where the encoding for the i -th element of a sequence as W_i , with c index referring to value for the *class tag*. The subsequent input embeddings are denoted as E_i , the hidden vectors as T_i .

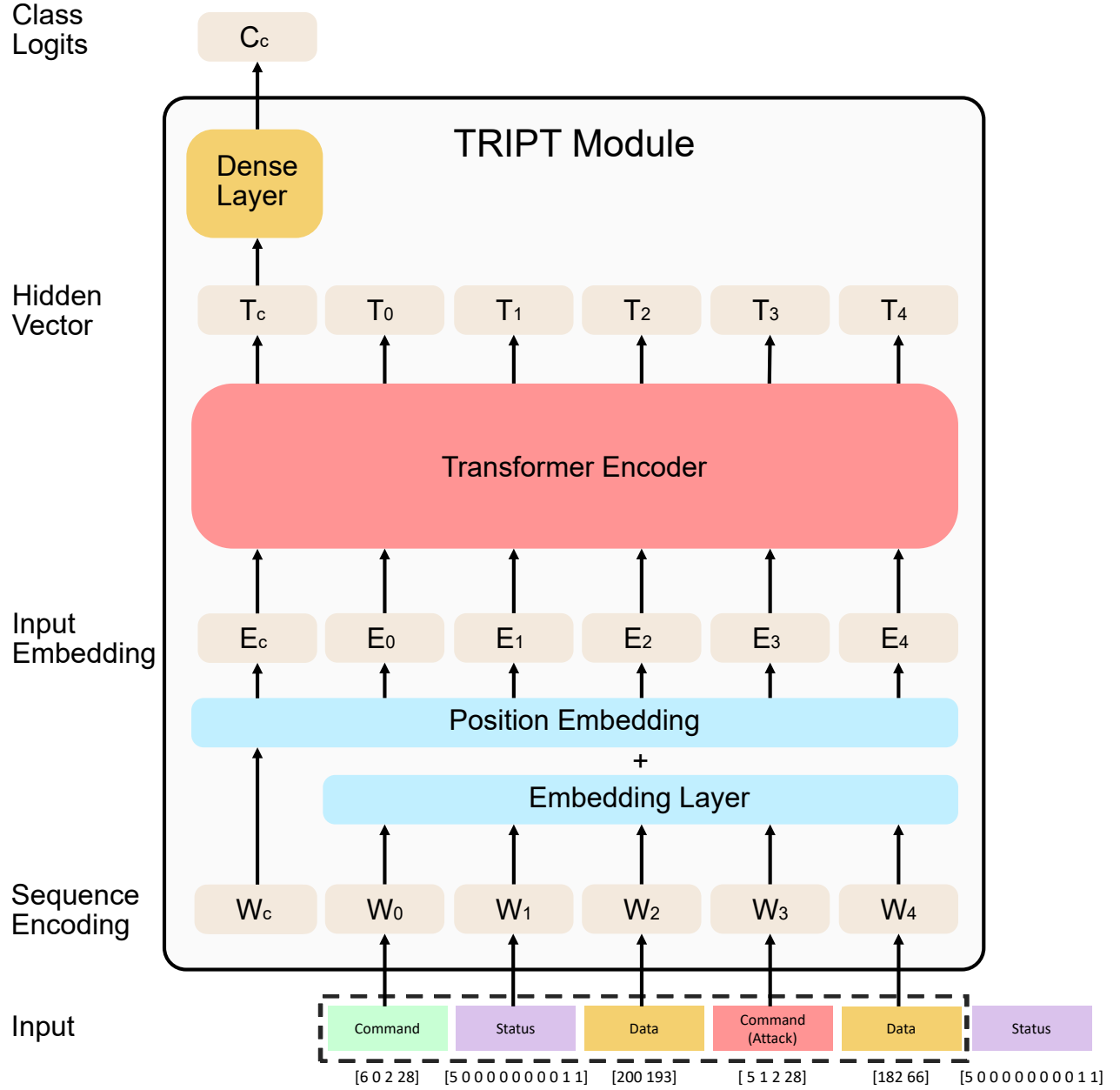


Figure 4.4 Overview of the TRIPT Module architecture

4.2.2 Pre-training: Triplet Loss Metric-Learning

The utilization of pre-trained embeddings has become synonymous with modern state-of-the-art NLP solutions. Extensive research has demonstrated that these embeddings capture valuable semantic and syntactic information [69], consequently propelling overall performance. However, as per the best of our knowledge, the adoption of a similar pre-trained traffic embedding remains unexplored within the realm of ML-based IDS. As a result, we concentrate

our efforts on a specific pre-training task aimed at acquiring intricate embeddings for both normal and abnormal network traffic.

A model trained for intrusion detection faces three main challenges to perform well: recognizing the context of traffic sequences (generalization), identifying common traffic sequences among incoming data (clustering), and determining the category of a given traffic sequence (detection). The pre-training task we chose tackles the generalization and clustering of the problem.

Inspired by the work done on embedding enrichment [24], we use an approach based on the concept of Metric Learning, a technique aimed at acquiring a distance function capable of accurately gauging the similarity or dissimilarity between pairs of samples [25, 70]. In essence, our strategy employs Triplet Networks. Figure 4.5 shows the network consisting of three branches, each sharing the same set of parameters. The model is trained by simultaneously presenting a positive and a negative example to the corresponding anchor, utilizing labeled data. In our context, if the anchor corresponds to a benign sequence, the positive example would be another benign sequence (not necessarily related), and the negative example would correspond to a malicious sequence. To facilitate optimal learning and yield superior outcomes, we employ the Triplet Margin Loss in conjunction with a well-considered approach to triplet generation.

Triplet Margin Loss

The loss function for the triplet approach is based on the idea of maximizing the distance between the anchor (x_i^a) and the negative (x_i^n) sequence, while minimizing the distance between the anchor and the positive (x_i^p) sequence. Figure 4.6 visually illustrates the outcome of a learning step aimed at minimizing the loss function's output. The distance calculation employs the Euclidean norm of the embedding $f(x) \in \mathbb{R}^d$ for every input sequence x with a d -dimensional. The objective is to satisfy:

$$\|f(x_i^a) - f(x_i^p)\|_2 + \text{margin} < \|f(x_i^a) - f(x_i^n)\|_2, \quad (4.2)$$

where a *margin* equal to 1 is used to ensure the minimum separation between the positive and negative pairs.

The loss that is being minimized is then,

$$\mathcal{L}1 = \max\{d(x_i^a, x_i^p) - d(x_i^a, x_i^n) + \text{margin}, 0\}, \forall (x_i^a, x_i^p, x_i^n) \in \mathcal{T} \quad (4.3)$$

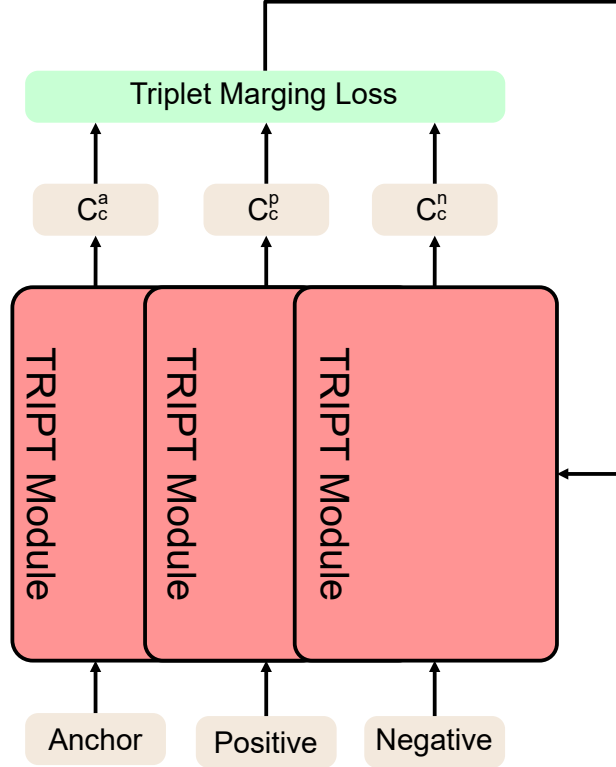


Figure 4.5 Overview of the Triplet Loss Metric-Learning pre-training task setup, where C_c^a , C_c^p , C_c^n are the class logits outputted by the TRIPT Module for the anchor, positive and negative input respectively.

where $d(x_i, y_i) = \|x_i - y_i\|_2$ and $\mathcal{T}$ is the set of all generated triplets. The next subsection talks about how triplets that violate the constraint in Eq. (4.2) are more effective for a fast convergence and why a good selection strategy is necessary.

Triplet Generation

The construction of triplets stands as a pivotal step in both training and optimizing model performance. A naive approach of generating all possible triplets or employing fixed triplets from the start would inevitably result in a multitude of ineffective triplets—those that contribute little to the learning process. Thus, it is imperative to dynamically select active triplets capable of enhancing the model’s capabilities.

Similarly to [26], we craft a unique triplet for each anchor traffic sequence x_i^a . For the *hard-positive* word sequence x_i^p , we choose the sequence with the maximum distance from the anchor within the embedding space: $\arg \max_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$. Subsequently, we opt for a *semi-hard negative* word sequence x_i^n instead of a *hard-negative* given that it has been shown

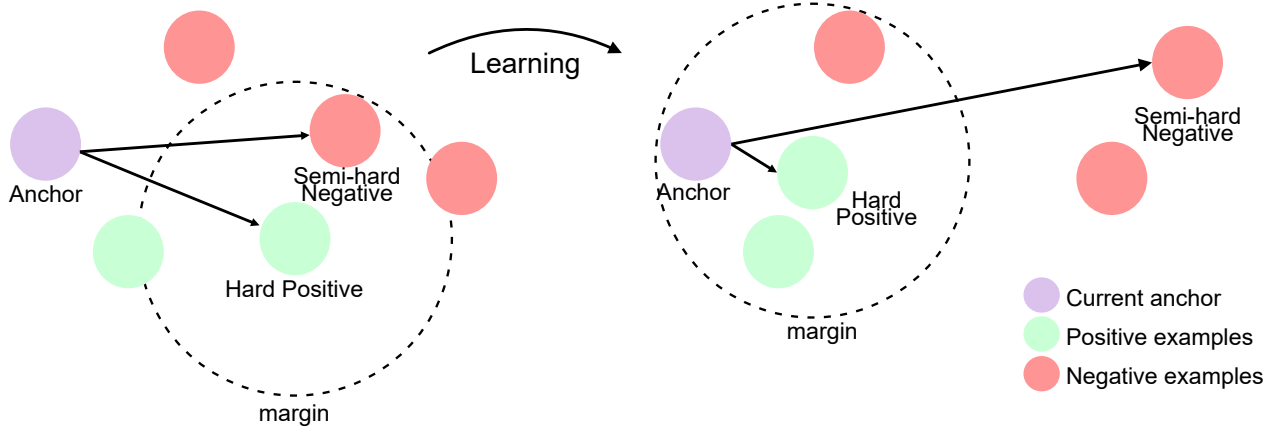


Figure 4.6 The error produced by the **Triplet Margin Loss** function is designed to encourage the model to minimize the distance between an *anchor* and a *positive* sample of the same class, while concurrently maximizing the distance between the *anchor* and a *negative* sample of a different class.

to sometime lead to bad local minima early on in training [26]. The *semi-hard negative* is positioned farther from the anchor than the *hard-positive* sequence, and is still posing a challenge, as the squared distance falls below the *margin* of the *hard-positive* :

$$d(x_i^a, x_i^p)^2 < d(x_i^a, x_i^n) < d(x_i^a, x_i^p)^2 + \text{margin} \quad (4.4)$$

This method prevents early training convergence to suboptimal local minima.

Due to the infeasibility of computing all hard and semi-hard positive and negative pairs across the entire training set [26], as well as the potential for outliers skewing these pairings, we opt to select triplets within each mini-batch. Ensuring a meaningful anchor-positive distance representation requires that each traffic type is sufficiently represented in every mini-batch. To achieve this balance, we employ weighted sampling during mini-batch formation, thus ensuring comprehensive representation across all attack types and mitigating the risk of training solely on benign-positive triplets.

4.2.3 Fine-tuning: Binary Intrusion Detection

The downstream task employed for fine-tuning our model involves binary intrusion detection, meaning the classification of network traffic into either benign ($y_i = 0$) or malicious ($y_i = 1$) categories. We opt for binary classification as we aim to be robustness against unseen attacks that might not be considered by a multi-class classification approach. As the field of avionics systems cybersecurity advances, with a more comprehensive understanding of potential at-

tack categories, there could be merit in exploring a multi-category classification downstream task. Contrary to a multi-class classification, which focuses on predicting the specific type of attack, a multi-category classification approach would be geared towards grouping novel, unseen attacks into predefined attack categories. For instance, these categories might include Collision-based DoS and Status Word Manipulation. The goal here is to effectively cluster new, previously unencountered attack instances into established attack categories, contributing to a more comprehensive understanding of the evolving threat landscape. This could be particularly advantageous when introducing an explanatory engine, which could leverage classification information to elucidate the origins and strategies of detected attacks.

Figure 4.7 displays the architecture of the downstream model mirroring the pre-trained model, except for the inclusion of an additional classifier layer. This anomaly detection head constitutes a straightforward linear transformation followed by a sigmoid function. The transfer learning is achieved by initializing the weights of the TRIPT Module with the learned weights from the pre-training phase.

For the loss function, Binary Cross Entropy is employed:

$$\mathcal{L}_2 = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (4.5)$$

Here, y_i represents the expected result, $\hat{y}_i$ is the predicted output, and N denotes the number of sequences in a mini-batch. The mini-batch utilized during the fine-tuning phase is not weighted as it was in the pre-training stage. This distinction arises due to the triplet sampling strategy in pre-training, which involves replacement to ensure equitable representation of attack instances with low prevalence in the dataset. However, within the context of intrusion detection, such uniform representation is not necessary. Our objective here is to train the model using a distribution of data that better simulates real-world conditions.

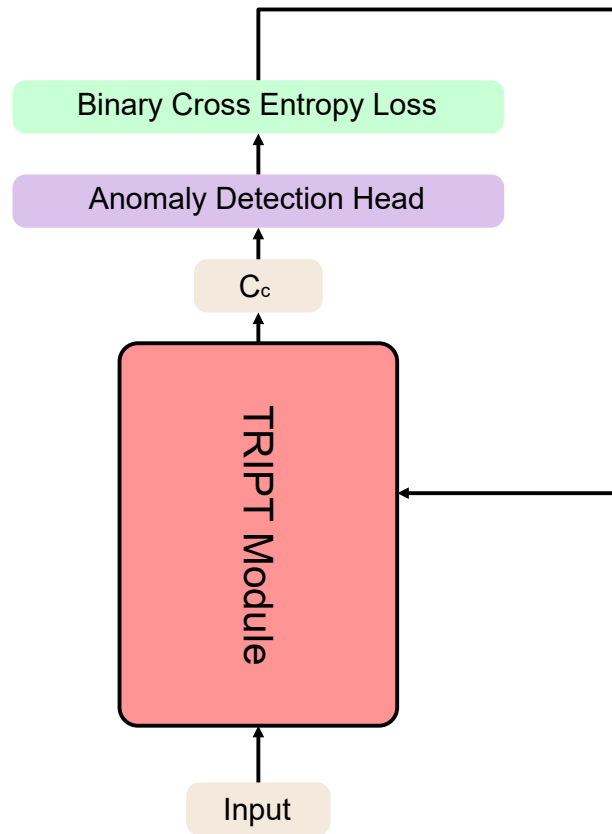


Figure 4.7 Overview of the Binary Intrusion Detection fine-tuning task setup, where C_c is the class logits outputted by the TRIPT Module.

CHAPTER 5 EXPERIMENTS AND RESULTS

In this chapter, we start by providing an overview of our environment and experiment setup, along with our evaluation methodology. Subsequently, we present each experimental scenario and engage in a discussion of the results. The experimental scenarios are categorized into those that involve testing against all previously seen attacks and those that exclusively incorporate previously unseen attacks.

5.1 Environment Setup

We implemented our solution and conducted experiments using Python, along with common libraries such as PyTorch¹ for DL techniques. Our experiments were conducted on a Linux machine running Fedora Linux 36. The hardware configuration includes an 8-core 11th Gen Intel(R) Core(TM) i7-11700F processor clocked at 2.50GHz, 64 GB of RAM, and an NVIDIA GeForce RTX 3090 Ti with 24 GB of VRAM.

5.2 Experimental Setup

Following the data generation process described in Section 4.1.3, we generated two sets of sequences using a *wSize* of 5 and 20. The smaller *wSize* comes from [10], while the larger *wSize* was established during the hyperparameters tuning (see Section 5.4). For training, we utilized the L1NNA Dataset [27], and for testing, we used our newly generated dataset. This separation allowed us to always evaluate on previously unobserved flight traffic, resulting in a more realistic assessment. The distribution of each set is shown in Table 5.1. To mitigate the influence of seed bias, we conducted the training and testing on five different seeds.

As previously mentioned, we employed the OD1NF1ST IDS [10] as our baseline for comparing our solution. While the L1NNA Dataset does come from the team behind OD1NF1ST, unfortunately, it is unclear if it is the same data used in [10] since it is not referenced in their work. Additionally, the lack of comprehensive data distribution and statistics prevent us from validating if it is indeed the same data. Lastly, some of the experiments conducted by [10] utilized previously observed flight data during testing, a practice we aim to avoid. For these reasons, we decided to directly run the OD1NF1ST IDS’s code publicly available in the same repository [68] as the software. This has enabled us to conduct direct comparisons using our experiments setup and dataset.

¹PyTorch. <https://pytorch.org/>

To assess the impact of transfer learning, we conducted experiments using a TransformerEncoder-IDS as well. This model shares the same architecture as the one used in TRIPT-IDS’s binary intrusion detection task, as illustrated in Figure 4.7, but without being pre-trained on any task.

Table 5.1 Train and test set distribution for both wSize=5 and wSize=20

Label	Attack	Train set (L1NNA Dataset)		Test set (Our Dataset)	
		Window size			
		5	20	5	20
1	Collision Attack Against The Bus	67006	84813	62490	92755
2	Collision Attack Against Target RT	33228	51774	26632	44529
3	Data Thrashing	3221	12780	1276	5076
4	Man-in-the-Middle	3037	9809	1855	5924
5	TX Shutdown	515	2060	140	560
6	Fake Status Reply to Receive Command	11645	42811	10543	37465
7	Fake Status Reply to Transmit Command	14989	48334	19970	59022
8	Desynchronization	1478	4772	824	2591
9	Data Word Corruption	5982	8648	11712	19034
10	Command Invalidation	215	860	44	164
Total Malicious		141316	266661	135486	267120
Total Benign		1794941	1654551	1684130	1547696
Total		1936257	1921212	1819616	1814816

5.3 Evaluation Method

We evaluate the performance of our solution in various scenarios using the confusion matrix, which allows us to measure five critical metrics:

True Positive (TP): Represent the malicious sequences that were accurately detected as intrusions.

False Positive (FP): Benign sequences that were incorrectly detected as intrusions.

True Negative (TN): Correspond to the benign sequences that were correctly not flagged as intrusions.

False Negative (FN): Denote the malicious sequences that went undetected as intrusions.

These values enable the calculation of the following metrics:

Precision: Precision measures the fraction of correctly detected intrusions among all the intrusion predictions:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5.1)$$

Recall: Recall, also known as the TPR, represents the fraction of correctly detected intrusions out of the total actual intrusions:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (5.2)$$

F1-Score: The F1-Score is the harmonic mean of precision and recall. A high F1-Score indicates that the model can accurately identify intrusions while minimizing false alarms:

$$\text{F1-Score} = 2 \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right) \quad (5.3)$$

False Alarm Rate (FAR): FAR (or False Positive Rate (FPR)) indicates the proportion of falsely detected intrusions out of all the predicted intrusions. It is essential to monitor this metric, especially for a critical infrastructure IDS, as excessive false alarms can lead to increased workload for users and introduce higher risks:

$$\text{FAR} = \frac{FP}{FP + TN} \quad (5.4)$$

Area Under the ROC Curve (AUC): AUC measures the two-dimensional area under the receiver operating characteristic curve (ROC). The ROC curve plots the TPR (Recall) against the FAR. AUC quantifies the separability of the model and its ability to distinguish between benign and malicious traffic.

By considering and combining these five metrics, we can effectively assess the efficiency of our IDS in identifying malicious sequences. Given that both datasets are skewed toward benign traffic, traditional metrics such as accuracy become less informative. For instance, a solution that triggers no alarms (i.e., detects no malicious traffic) would still achieve a high accuracy rate, around 92% in our case, which does not provide meaningful insights into performance.

In addition to these metrics, we consider *Runtime*, representing the time taken for the IDS to process the entire test set. Efficient runtime is crucial for detecting cyberthreats in critical infrastructure. It is important to note that our script for running experiments was not optimized for speed performance, and suboptimal code design might impact runtime. To ensure valid comparisons, we use the same script to test each model, only swapping the model that was trained and tested.

Table 5.2 TRIPT-IDS hyperparameters

Categorie	Hyperparameters	Values
General	Epoch	20
	Mini-batch size	512
	Window size (wSize)	20
Optimizer	Algorithm	Adam
	Learning rate (lr)	1e-3
	Momentum	0.9
	Lr decays step size	1
	Lr decays gamma	0.99
Triplet Loss	Margin	1
Model	Input embedding size	16
	Number of attention head	8
	Number of sub-encoder-layers	4
	Transformer feedforward network dimension	1024
	Intrusion detection head hidden size	16
	Intrusion detection head number of layers	1

5.4 Hyperparameters tuning

Table 5.2 present all optimal hyperparameters for TRIPT-IDS. We obtained these hyperparameters by conducted experiments within the context of the seen attacks scenario as outlined in Section 5.5. We leveraged the Scikit Learn cross-validated grid-search function to systematically explore and identify the most optimal values for various hyperparameters. For the grid-search, we randomly sampled 80% of the train set for training and 20% for validating. In the case of the TransformerEncoder-IDS, the grid-search recommended a smaller architecture for the Transformer encoder module, as detailed in Annex A. For the sake of efficient experimentation and to comply with the memory constraints of our environment, we deliberately set the number of epochs and the mini-batch size at values that would ensure relatively short experiment duration. To ensure fairness and comparability across all solutions, we maintained uniformity in the general hyperparameters across the board.

5.5 Experiments on seen attacks scenario

We conducted the first set of experiments on a seen attacks scenario. This means the solutions were tested on all 10 attack types that were seen in some way during training. This set of experiments allows us to showcase and compare how well a solution is able to learn attack patterns and identify them among unseen traffic data.

5.5.1 Evaluating sequence encoding strategies

Table 5.3 Encoding strategies comparison with the TRIPT-IDS architecture and wSize=5

Encoding Scheme	Precision	Recall	F1	FAR	AUC	Runtime(s)
Original	87.0	89.3	88.2	1.10	94.1	31.49
Tag	86.7	94.9	90.6	1.20	96.9	32.75
Vector	97.4	98.6	98.0	0.20	99.2	13.37

Table 5.3 presents the resulting performance metrics from an evaluation of the three sequences encoding schemes introduced in Section 4.1.4 using the TRIPT-IDS architecture. Comparing the performance of the *Original Encoder* with the *Tag Encoder*, we observed a significant improvement when incorporating tags for context information and focusing on learning in a Transformer-based approach. Among the three schemes, the *Vector Encoder* emerged as the most effective, surpassing the *Original Encoder* on average by 8.65% in terms of precision, recall, F1-score, AUC, and achieving a lower FAR. This underscores the significance of treating MIL-STD-1553 packet attributes as a single entity rather than as separate tokens concatenated in a sequence. For all subsequent experiments, we utilized the *Vector Encoder* for sequence encoding in Transformer-based solutions. However, it is worth noting that we refrained from using the *Vector Encoder* with the OD1NF1ST model, as their model was specifically designed to work with the *Original Encoder*.

5.5.2 Fully fine-tune experiment

For the fully fine-tune experiment, we conducted a comparative analysis of each model’s performance in detecting the 10 known intrusions observed during training against previously unseen flight scenarios. Each model was evaluated using two context window sizes: 5 and 20, allowing us to assess the impact of a larger context window on performance. Upon reviewing the results presented in Table 5.4, several key observations can be made. Firstly, there is a noticeable improvement in performance when transitioning from the Bi-LSTM approach to

Table 5.4 Performance and runtime comparison of TRIPT-IDS on seen attacks scenario

Model	wSize	Precision	Recall	F1	FAR	AUC	Runtime(s)
OD1NFIST	5	92.0	77.3	84.0	0.50	88.4	13.83
	20	92.7	97.1	94.8	1.30	97.9	21.96
TransformerEncoder-IDS	5	96.1	97.7	96.9	0.30	98.7	11.38
	20	98.5	97.6	98.0	0.30	98.7	11.87
TRIPT-IDS	5	97.4	98.6	98.0	0.20	99.2	13.37
	20	97.5	98.6	98.1	0.43	99.1	19.56

Transformer-based models. Furthermore, a significant performance boost is observed for the OD1NFIST model when using the larger context window size, although the performance gain is less pronounced for both Transformer-based models. Comparing OD1NFIST’s results with our TRIPT-IDS trained with the original encoding, our solution performs better in terms of recall, F1-Score and AUC. Regarding the influence of the pre-training steps, we note a slight performance increase with the TRIPT-IDS model, especially with a smaller context window.

Figures 5.1, 5.2, and 5.3 present visualizations of the embedding space on the test set for each of the solutions using the t-SNE algorithm. Additionally, they depict the TPR and False Negative Rate (FNR) for each traffic type. All three IDSs exhibit overall good TPR for each sequence type. However, the OD1NFIST approach appears to struggle with the TX Shutdown attack type, which affects its overall performance. This could be explained by the few examples of TX shutdown attacks in the data and the fact that detecting this attack can be challenging, as it requires the IDS to differentiate abnormal shutdown commands from normal ones. Regarding the embedding visualizations, both Transformer-based approaches demonstrate an impressive ability to cluster attack traffic effectively. The difference between TransformerEncoder-IDS and TRIPT-IDS can be explained by the objective of the TRIPT-IDS’s pre-training task. The triplet loss metric learning encourages the model to cluster all types of malicious traffic together. Consequently, this creates an embedding space where all attack types are closely grouped. It is interesting to note how well TransformerEncoder-IDS is able to learn attack patterns, resulting in separate clusters for each attack type without explicit receiving any information about the attack type during training. In contrast, the OD1NFIST approach is trained for both binary classification and multi-class classification.

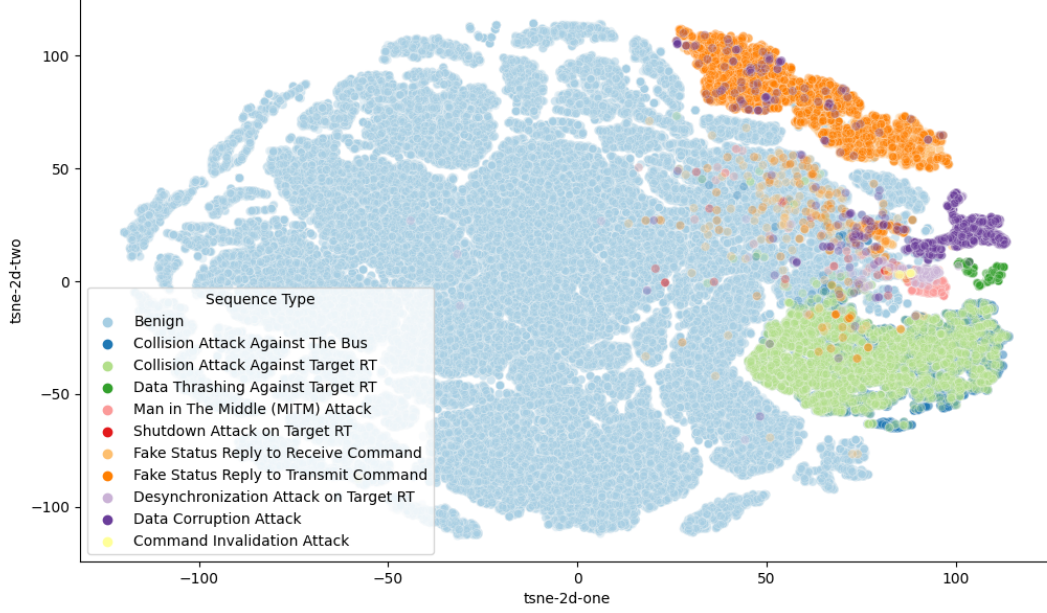
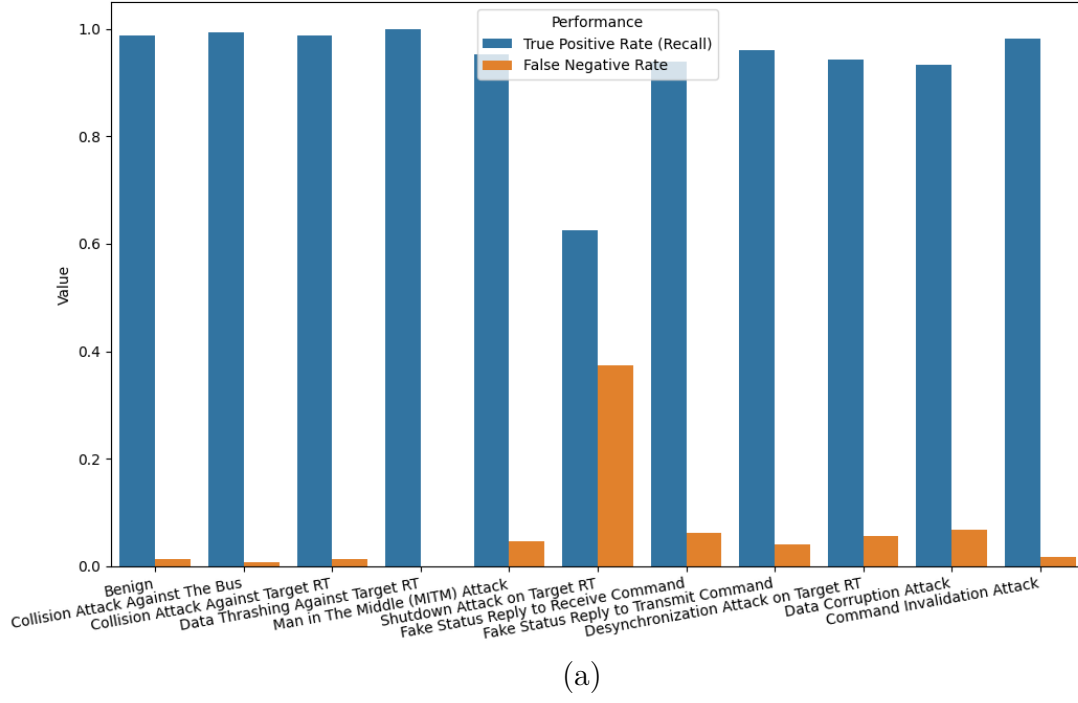


Figure 5.1 OD1NFIST IDS fully fine-tune experiment results: (a) TPR and FNR for each sequence type; (b) Output embedding of each sequence by type

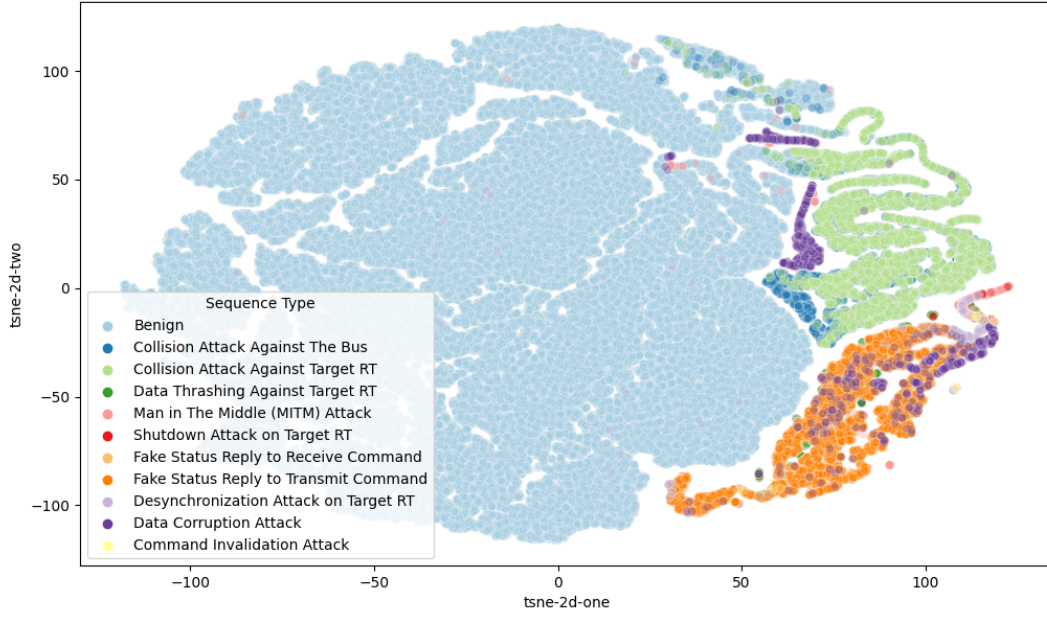
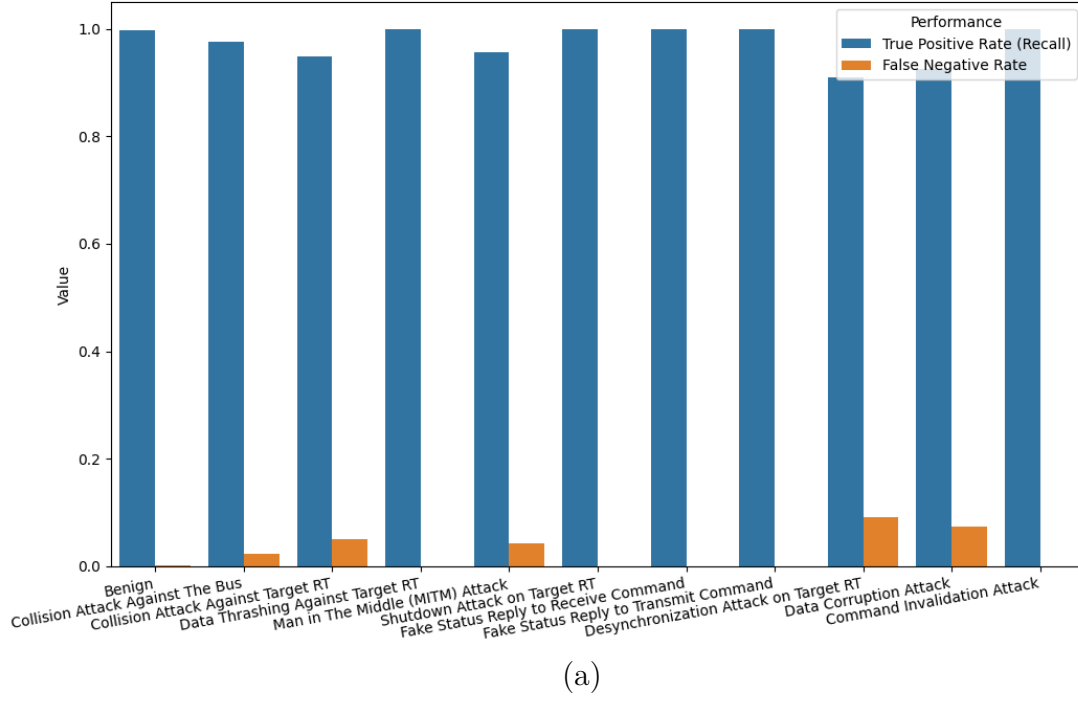


Figure 5.2 TransformerEncoder-IDS fully fine-tune experiment results: (a) TPR and FNR for each sequence type; (b) Output embedding of each sequence by type

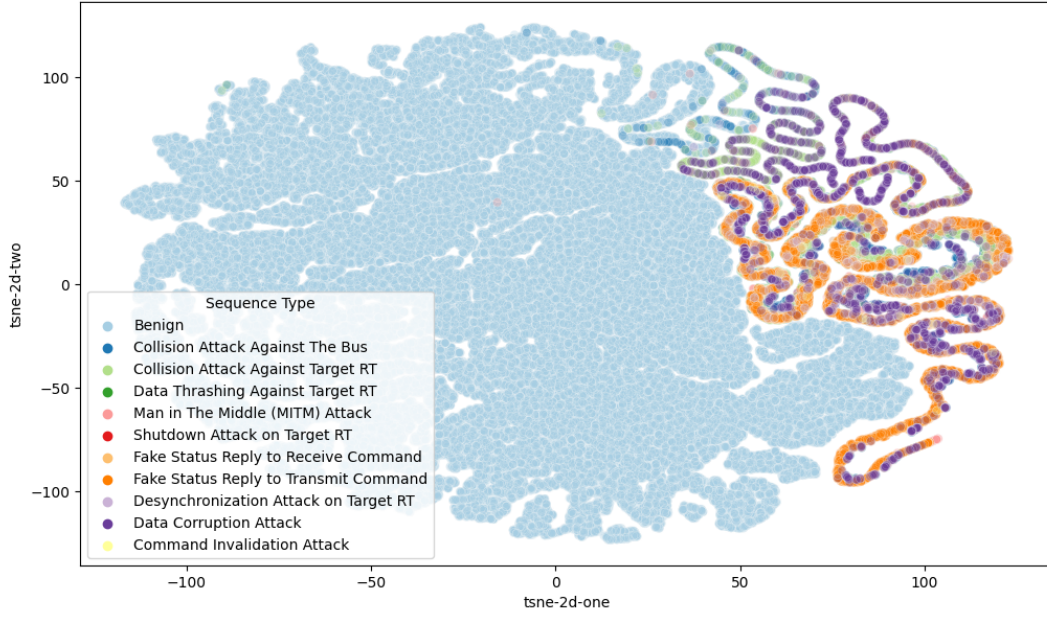
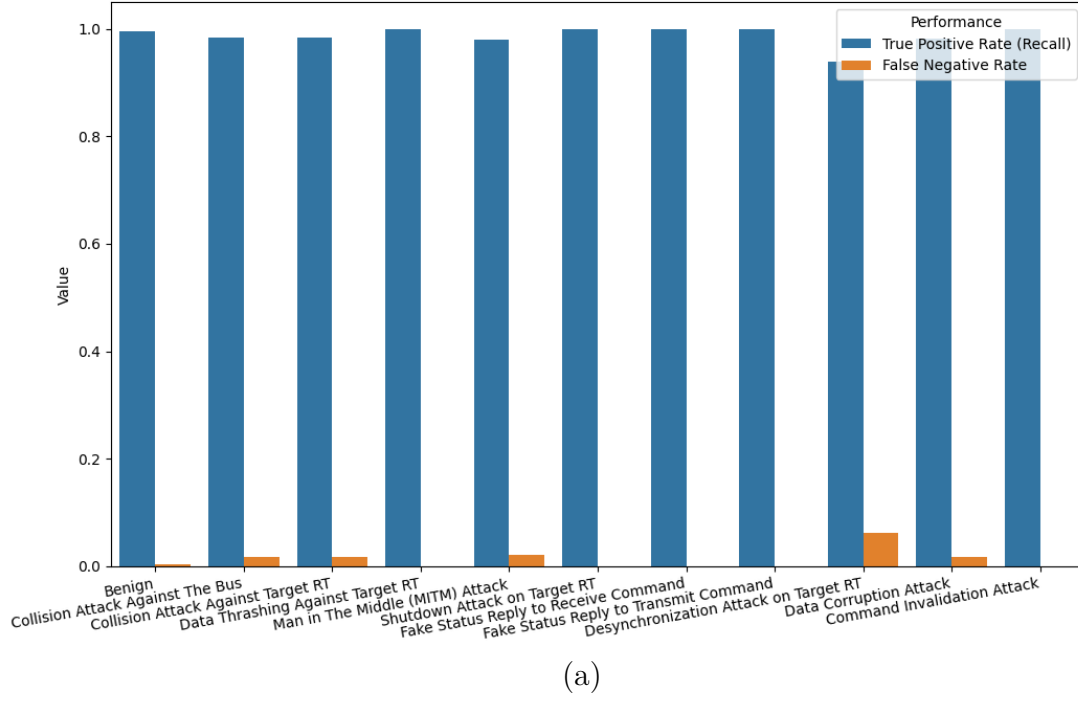


Figure 5.3 TRIPT-IDS fully fine-tune experiment results: (a) TPR and FNR for each sequence type; (b) Output embedding of each sequence by type

5.5.3 Few-shot classification experiment

Few-shot learning, specifically few-shot classification, involves a model’s attempt to classify data into an unknown number of categories, with only a few examples per category during training [71]. The underlying motivation is to mimic the way humans learn, as humans can recognize someone after seeing just a few pictures of them, for example. The theoretical goal is to enable a model to learn rare cases even when only a few examples are available. Successfully achieving this can substantially reduce the effort and computational costs associated with data collection, as only a limited number of data points would be required.

It can be argued that intrusion detection tasks in avionics cybersecurity can be framed as few-shot classification problems. This is primarily due to the imbalanced nature of the datasets and the absence of real-world recorded attacks. In practical terms, this means that if a new real-world recorded attack occurs, there may not be a large number of examples available for that specific attack type. Therefore, having a few-shot enabled IDS is valuable, especially when an updated version of the IDS needs to be deployed before the attack can be reproduced multiple times in a controlled testbed setting.

Triplet Networks can be classified as few-shot-based approaches [71]. This classification arises from the additional prior knowledge about similarity, both positive and negative examples, that is provided during training. This additional information aids the model in learning with just a few data points. This stands in contrast to traditional supervised ML approaches, where the only prior knowledge is the label.

To assess the few-shot classification capabilities of TRIPT-IDS, we conducted an experiment similar to the one described in Section 5.5.3. However, in this experiment, we arbitrarily chosen to randomly sampled only 1% of each attack type for the training set. This sampling guaranty only a few examples per attack type, while keeping the underlying distribution of each attack type the same.

This resulted in a total of 2675 malicious sequences compared to the unchanged 1654551 benign sequences, corresponding to a mere 0.16% presence of malicious data in the training set. Table 5.5 presents the performance of each solution in this few-shot scenario.

Immediately, we observed that the OD1NF1ST model exhibited an inability to learn to detect any attacks. In other words, the Bi-LSTM based IDS is unable to perceive any significant pattern from the small set of attack examples, which results into mistakenly classifying all traffic as benign. While less dramatic, the TransformerEncoder-IDS experienced a significant drop in performance. In contrast, TRIPT-IDS maintained relatively high performance, with only a modest average drop of 5% for Recall, F1-Score, and AUC. An interesting observation

Table 5.5 Performance comparison of TRIPT-IDS on few-shot classification attacks scenario wSize=20

Model	Precision	Recall	F1	FAR	AUC
OD1NF1ST	00.0	00.0	00.0	0.00	50.0
TransformerEncoder-IDS	99.7	63.6	77.6	0.03	81.8
TRIPT-IDS	99.9	90.1	94.5	8.85e-5	95.0

is the low FAR, which is close to zero. This can be attributed to the fact that, given the limited number of attack examples during training, the IDS is less prone to trigger false alarms. Figure 5.4 further illustrates that TRIPT-IDS is effective in detecting all attack types, with the exption of the MITM type. Our hypothesis is that the nature of the attack is quite challenging to learn when very few examples are provided. The MITM attack involves sending realistic command words, posing as a BC, and making two Remote RTs start transmitting unwanted data, in the lack of training data, can be seen as normal behaviour. It would be interesting to study if this limitation persists with a larger pool of legitimate flight data.

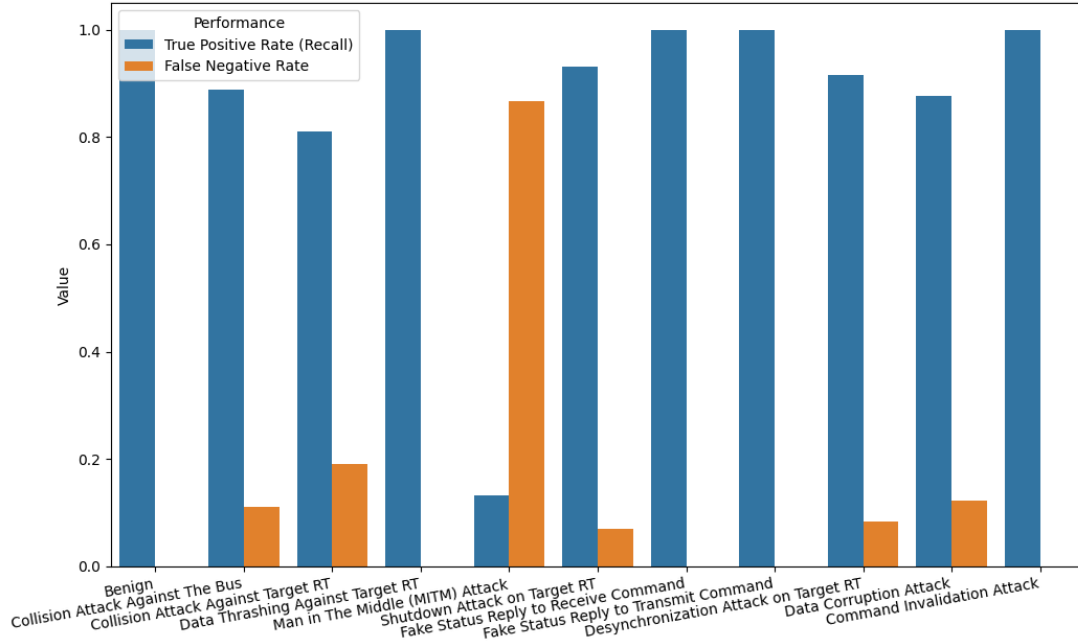


Figure 5.4 TPR and FNR for each attack type during few-shot training experiment for TRIPT-IDS

5.6 Experiments on unseen attacks scenario

With the objective of evaluating the performance of our solution as closely as possible to a realistic scenario, we conducted three experiments to assess its robustness against unseen attacks and traffic. The first experiment involved individual cross-evaluation on unseen attacks to showcase the IDS’s performance on each attack type it was not trained on. The second experiment aimed to demonstrate the system’s performance in detecting new attacks within each attack category introduced in Table 4.1, which exhibit specific characteristics similar to the smaller set of known attacks used during training. The last experiment was conducted to validate the system’s performance under highly challenging conditions, where only a restricted subset of an already small pool of malicious traffic patterns was available during the training phase.

5.6.1 Individual unseen attack cross-evaluation

In this specific experiment, we trained the model on nine types of attacks and then tested it on the remaining one, repeating this process for each attack type. The results of these experiments can be observed in Table 5.6. OD1NF1ST performs well on collision attacks and sending fake status to receive command attacks. However, we immediately see a significant drop in performance for all the other attacks. This essentially means that this Bi-LSTM-based model struggles to detect previously unseen attacks alongside unobserved flight data. This could be related to a lack of sufficient generalization of the seen attacks’ characteristics.

On the other hand, both of our Transformer-based IDS performed significantly better, showing the superior generalization capability of this architecture over the baseline’s. As for the impact of pre-training, we observe an improvement across the attack types, however, there’s still room for improvement. Figure 5.5 shows a better visualization of TRIPT-IDS’s performance. There is a noticeable drop in performance for both Transformer-based IDS when it comes to attacks with unique patterns, such as TX Shutdown. Additionally, the models exhibit an inability to detect attacks that aim to initiate unauthorized transmissions and transmit specific and unique malicious data, as seen in the cases of MITM attack and Data Corruption attack.

We hypothesize that since our solution exhibits a more cautious behaviour (i.e., it favours a low FAR), and these attacks impersonate and imitate realistic behaviours, our IDS may opt not to trigger potential false alarms. A possible workaround or method for improving our approach could involve employing an additional or alternative pre-training task, similar to masked language modelling as found in models like BERT [34], ELECTRA [35], and

Table 5.6 Performance comparison of TRIPT-IDS on unseen attack cross-evaluation with wSize=20

Unseen attack	Model					
	ODINFI1ST		TransformerEncoder-IDS		TRIPT-IDS	
	TPR	FNR	TPR	FNR	TPR	FNR
Collision Attack Against The Bus	99.2	0.80	89.7	10.3	98.5	1.50
Collision Attack Against Target RT	95.2	4.80	82.4	17.6	83.6	16.4
Data Thrashing	11.1	88.9	100.0	0.00	100.0	0.00
Man-in-the-Middle	11.4	88.6	00.0	100.0	0.20	99.8
TX Shutdown	0.00	100.0	53.4	46.6	56.8	43.2
Fake Status Reply to Receive Command	81.3	18.7	100.0	0.00	89.6	10.4
Fake Status Reply to Transmit Command	0.00	100.0	100.0	0.00	100.0	0.00
Desynchronization	0.00	100.0	86.1	13.9	90.1	9.90
Data Word Corruption	30.6	69.4	30.4	69.6	30.7	69.3
Command Invalidation	0.00	100.0	100.0	0.00	100.0	0.00

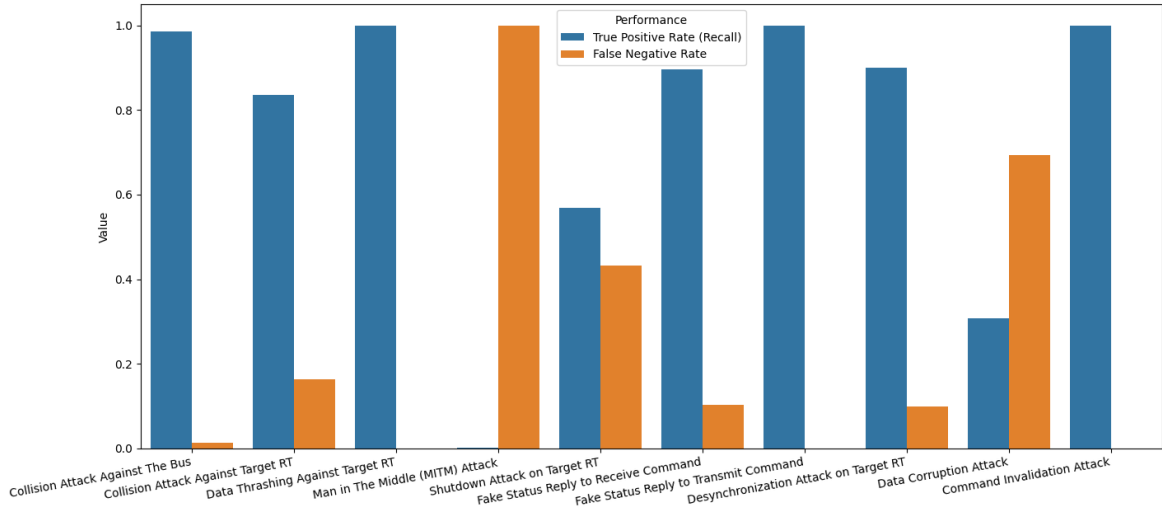


Figure 5.5 TPR and FNR for each unseen attack type during separate training experiment for TRIPT-IDS

DATE [49].

These approaches utilize self-supervised training tasks, which means they use the input data itself as a label and learn to recognize and recreate plausible sequences. By employing such self-supervised training tasks, an IDS could potentially gain a better understanding of the semantics of benign traffic. This improved understanding could enhance its ability to identify previously unknown MITM attacks and similar threats.

5.6.2 Categorical unseen attack

The second set of evaluations aims to demonstrate the model’s capability to learn specific characteristics of a small set of known attacks and utilize this knowledge to detect new attacks that employ similar tactics. We outlined in Table 5.7 the new distribution of the train and test set. Four attack types have been selected for training the model, each representing a different attack category. Subsequently, we conducted tests on the remaining six attack types. We conducted separate tests for each attack category ones the model is trained. This approach allows us to compare the performance of each model on every category and avoids bias towards attack types with a higher presence in the dataset. Nevertheless, we conducted tests on all unseen attack types to demonstrate the overall performance of an IDS.

Table 5.7 Attack category distribution in both train and test set wSize=20

Attack Category	Attack	Train set	Test set
Denial-of-Service	Collision Attack Against The Bus	84813	-
	Collision Attack Against Target RT	-	44529
Status Word Manipulation	Fake Status Reply to Receive Command	42811	-
	Fake Status Reply to Transmit Command	-	59022
Malicious Command	Data Thrashing	-	5076
	Man-in-the-Middle	9809	-
	TX Shutdown	-	560
	Desynchronization	-	2591
	Command Invalidation	-	164
Word Generation	Data Word Corruption	8648	-
Total Malicious		146081	111942
Total Benign		1654551	1547696
Presence		8.11%	6.74%
Total Benign		18000632	1659638

Upon analyzing the results presented in Table 5.8, It is evident that both TransformerEncoder-IDS and TRIPT-IDS performed relatively well, achieving F1-Scores of 96.0% and 94.9%, respectively, on all combined categories. In general, TRIPT-IDS displayed slightly better performance than TransformerEncoder-IDS in both the DoS and Status Word Manipulation categories. However, TRIPT-IDS exhibited greater robustness in detecting attacks within the Malicious Command category, which includes attack types with more unique and sophisticated exploits. In contrast, the OD1NF1ST LSTM-based approach faced challenges when trained on a limited number of attacks and then tested on unknown attacks. This approach showed notable limitations, particularly in its inability to detect both unseen Status Word Manipulation and Malicious Command attacks.

Table 5.8 Performance comparison of TRIPT-IDS on unseen categorical attacks scenario with wSize=20

Model	Attack Category	Precision	Recall	F1	FAR	AUC
OD1NF1ST	DoS	97.3	95.1	96.2	0.70	97.2
	Status Word Manipulation	91.8	20.7	33.8	0.70	60.0
	Malicious Command	33.4	6.60	1.10	0.70	52.9
	Combined	98.1	49.2	65.6	0.70	74.3
TransformerEncoder-IDS	DoS	98.9	81.2	89.2	0.30	90.5
	Status Word Manipulation	99.4	99.9	99.7	0.2	99.9
	Malicious Command	94.0	75.2	83.6	0.30	87.5
	Combined	99.6	90.7	94.9	0.30	95.2
TRIPT-IDS	DoS	98.8	83.7	90.7	0.30	91.7
	Status Word Manipulation	99.3	99.9	99.7	0.3	99.9
	Malicious Command	95.2	88.1	91.5	0.20	94.0
	Combined	99.7	92.6	96.0	0.20	96.2

However, the OD1NF1ST performs better than our TRIPT-IDS for the DoS category. As shown in Table 5.9, when running the experiment with a smaller window size of 5, we observed an increase in performance for the DoS category across all models. The smaller window size also improved the performance of the OD1NF1ST model in the Status Word Manipulation category. For both Transformer-based approaches, the smaller context window slightly decreased the performance in detecting both unseen Status Word Manipulation and Malicious Command attacks. This is because these two categories involve more sophisticated attacks than those found in the DoS category, and a larger context window helps detect unseen attacks.

Table 5.9 Performance comparison of TRIPT-IDS on unseen categorical attacks scenario with wSize=5

Model	Attack Category	Precision	Recall	F1	FAR	AUC
OD1NFIST	DoS	97.3	99.5	99.4	0.10	99.7
	Status Word Manipulation	99.3	98.9	98.9	0.10	99.2
	Malicious Command	46.0	4.0	8.70	0.10	52.4
	Combined	99.7	94.6	97.1	0.10	97.3
TransformerEncoder-IDS	DoS	97.0	98.4	97.7	0.50	98.9
	Status Word Manipulation	96.2	99.9	98.1	0.5	99.8
	Malicious Command	67.9	69.4	68.6	0.40	84.5
	Combined	98.4	97.7	98.0	0.50	98.6
TRIPT-IDS	DoS	99.0	98.2	98.6	0.20	99.0
	Status Word Manipulation	98.7	99.9	99.3	0.2	99.9
	Malicious Command	87.5	73.4	79.9	0.10	86.6
	Combined	99.4	97.8	98.6	0.20	98.8

5.6.3 Few-shot classification on categorical unseen attack

In this evaluation, we combined the few-shot classification scenario with the scenario involving previously unseen categorical attacks. This experiment allows us to assess the IDSs performance in an extreme case where only a limited set of an already small subset of malicious traffic patterns is available during training. We observe in Table 5.10 that both the OD1NF1ST and TransformerEncoder IDSs lost their capability to detect previously unseen attacks. The OD1NF1ST IDS did not trigger any alarms on the test set.

Table 5.10 Performance comparison of TRIPT-IDS on few-shot classification on unseen categorical attacks scenario with wSize=20

Model	Attack Category	Precision	Recall	F1	FAR	AUC
OD1NFIST	DoS	0.00	0.00	0.00	0.00	50.0
	Status Word Manipulation	0.00	0.00	0.00	0.00	50.0
	Malicious Command	0.00	0.00	0.00	0.00	50.0
	Combined	0.00	0.00	0.00	0.00	50.0
TransformerEncoder-IDS	DoS	98.3	32.8	49.2	0.20	66.3
	Status Word Manipulation	1.70	0.00	00.0	0.10	49.9
	Malicious Command	0.00	0.00	0.00	0.20	49.5
	Combined	98.6	13.1	23.1	0.10	56.5
TRIPT-IDS	DoS	98.8	99.9	80.8	1.87e-4	90.4
	Status Word Manipulation	99.9	99.9	99.9	2.97e-4	99.9
	Malicious Command	91.2	6.30	11.8	2.07e-4	53.1
	Combined	99.9	85.3	92.1	2.58e-4	92.6

In contrast, TRIPT-IDS demonstrated effectiveness in detecting attacks from both the DoS

and Status Word Manipulation categories. However, it faced challenges in successfully detecting attacks from the Malicious Command category. This outcome is not entirely surprising, as the Malicious Command category encompasses various attack types, each exploiting different command word vulnerabilities. This arrangement already makes it a challenging feat to detect accurately, but the additional constraint of the 1% sampling rates further exacerbates the difficulty. When examining the sequence distribution in Table 5.7, we observe that we encountered 8391 previously unseen Malicious Command attacks during testing. Compared the 1% sampling we now get only 98 Malicious Command instances (1% of MITM sequences) encountered during training. This result may highlight a potential limitation of our approach. However, we hypothesize that once trained on all ten attack types, TRIPT-IDS will exhibit greater robustness against newly unseen Malicious Command attacks compared to existing LSTM-based approaches.

CHAPTER 6 CONCLUSION

This chapter provides a summary of our research and emphasizes our contributions. Additionally, we elaborate the limitations of our approach and outline potential directions for future research.

6.1 Summary of Works

The emergence of novel technologies and their integration in critical infrastructure, including avionics, has pushed the boundaries of what is possible in the field. The adoption of interconnected systems and wireless connectivity has revolutionized aircraft of today and will continue to do so in the future. Unfortunately, these advancements introduce threats that were previously deemed impossible. The prevalence of plausible cyberattacks is increasing, even in systems such as the automobile and avionics that were previously considered closed. This highlights the importance of conducting research to identify and develop solutions that can effectively protect against and mitigate this growing issue.

We introduce TRIPT-IDS, a novel intrusion detection approach that leverages a Transformer-based architecture with a triplet loss pre-training task. The pre-training tasks use a metric-learning-based approach with the objective to learn an embedding space capable to accurately represent the similarity and dissimilarity between benign and malicious traffic. To put it differently, our model is first trained on a task that does not focus on intrusion detection but rather on learning the semantic relations of the traffic. Then, through transfer learning, we finally fine-tune the model on intrusion detection with the prior knowledge of the network’s semantics.

We demonstrate the gain in performance of our approach with a array of experience all intended to test the performance on realistic scenarios. While our approach did outperform the other IDS in the fully fine-tune experiment, it is in the few-shot classification and categorical unseen attack experiment that our approach truly outperformed the rest. This shows TRIPT-IDS’s ability to learn complex patterns with only a few data points and generalize to detect unseen attacks with similar behaviours. In the context of a critical infrastructure with little or no real-world examples of cyber attacks, this ability can be a game changer.

6.2 Limitations

This work introduces a novel IDS approach for the MIL-STD-1553 avionic protocol, enhancing its resilience and versatility through a Transformer-based architecture and transfer learning. However, due to the challenges described in Section 1.3 and other factors, we identified two limitations of our approach.

Firstly, the initial objective was to experiment with real AFDX traffic data in real time on a sophisticated testbed. However, we encountered concerns due to non-disclosure agreements with partners and delays in purchasing the test bench caused by COVID-19. As a result, we had to resort to an alternative option. We ended up utilizing the newly developed OD1NF1ST software [68] to generate and use MIL-STD-1553 data. While this option allowed us to test and validate our solution, it also introduced limitations. Firstly, due to time constraints, we were unable to thoroughly validate the quality of the data generated by the software. Secondly, we were unable to utilize the presumed real-time functionality of the software and only tested on offline data. Although this option brought limitations, it also enabled us to directly validate our solution against a state-of-the-art IDS.

A second limitation of our approach is in detecting certain unseen attack types: MITM and Data Word Corruption. Both of these attack types aim to transmit malicious data words as masking as plausible normal traffic. Throughout our experiments, our Transformer-based architecture (pre-trained or not) displayed a low FAR, indicating that our IDS is vigilant and less prone to triggering false alarms. This means it has difficulty with more sophisticated attacks that can mask themselves as benign traffic. A potential solution could involve generating more flight data, as we are currently training on a relatively small dataset, and possibly improving the pre-training task.

6.3 Future Research

We anticipate that this work will serve as a foundational step towards the development of a new generation of IDS systems that harness the transfer learning capabilities of the Transformer model, similar to its success in the NLP field. Our study has demonstrated that by employing a specific pre-training task with a Transformer-based architecture, we can significantly enhance the performance of an IDS, particularly in the context of previously unseen attacks and few-shot classification scenarios. In the future, we can continue drawing inspiration from the NLP field by introducing self-supervised tasks, akin to those in the DATE model [49], to train on unlabelled data and further improve the model’s ability to learn benign traffic patterns. Similar to BERT [34], this new IDS could be pre-trained on

multiple tasks to capture various semantic and sequence arrangement information.

As a second direction, our work can be integrated and tested within the newly acquired testbed of our laboratory. To enable this integration, we propose establishing a framework similar to AnoMili [9]. This framework would need to support real-time data transfer to the TRIPT-IDS, implement a threshold mechanism to trigger alarms only when the IDS detects a minimum continuous flow of malicious traffic, and include an explanation engine responsible for conveying critical information to the flight crew, enabling them to make informed decisions. The importance of this last module cannot be overstated, as a simple alert would be inadequate and could potentially overwhelm pilots who are already managing multiple flight-related alerts.

A final direction for future research could expand our work to encompass other network protocols. The avionic ecosystem regroups numerous different protocols, and adapting and testing TRIPT-IDS on these protocols could enhance its versatility and integration capabilities. Furthermore, similar investigations could be conducted in the realm of telecommunication protocols and other critical infrastructure sectors, such as the automobile industry. The concept of transfer learning could be explored even further. Our current approach is limited to a single protocol, but it would be intriguing to explore the possibility of using a IDS model pre-trained on multiple protocols and then fine-tuning it for a new protocol. This approach could be particularly compelling for Ethernet-based protocols like AFDX and ASCB-D, or the CAN Bus protocol, which is utilized in both avionic and automobile systems.

REFERENCES

- [1] E. Habler, R. Bitton, and A. Shabtai, “Evaluating the security of aircraft systems.” [Online]. Available: <http://arxiv.org/abs/2209.04028>
- [2] C. M. Fuchs, “The evolution of avionics networks from ARINC 429 to AFDX,” *Seminar Aerospace Networks SS2012 Chair for Network Architectures and Services Faculty of Informatics, Technical University of Munich*, p. 13, 2012. [Online]. Available: https://www.researchgate.net/profile/Christian-Fuchs-17/publication/282108687_The_Evolution_of_Avionics_Networks_From_ARINC_429_to_AFDX/links/5c95fabda6fdccd46033ec9c/The-Evolution-of-Avionics-Networks-From-ARINC-429-to-AFDX.pdf
- [3] I. Land and J. Elliott, “Architecting ARINC 664, part 7 (AFDX) solutions,” p. 25. [Online]. Available: <https://usermanual.wiki/Document/xapp1130.1154455528.pdf>
- [4] M. Paulitsch, *ASCB: Avionics Standard Communications Bus*, 01 2015, pp. 295–301. [Online]. Available: <https://www.taylorfrancis.com/chapters/edit/10.1201/b17365-46/ascb-avionics-standard-communications-bus-michael-paulitsch>
- [5] D. D. Corporation, *MIL-STD-1553 DESIGNER’S GUIDE*. Data Device Corporation, 1998. [Online]. Available: www.ddc-web.com/1553DesignersGuide
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.
- [7] O. Stan, A. Cohen, Y. Elovici, and A. Shabtai, “Intrusion detection system for the MIL-STD-1553 communication bus,” vol. 56, no. 4, pp. 3010–3027. [Online]. Available: <https://ieeexplore.ieee.org/document/8946705/>
- [8] X. Liu, D. He, Y. Gao, S. Zhu, and S. Chan, “TLP-IDS: A two-layer intrusion detection system for integrated electronic systems,” in *2020 International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, pp. 205–214. [Online]. Available: <https://ieeexplore.ieee.org/document/9252065/>
- [9] E. Levy, N. Maman, A. Shabtai, and Y. Elovici, “AnoMili: Spoofing prevention and explainable anomaly detection for the 1553 military avionic bus,” number: arXiv:2202.06870. [Online]. Available: <http://arxiv.org/abs/2202.06870>

- [10] M. M. Wrana, M. Elsayed, K. Lounis, Z. Mansour, S. Ding, and M. Zulkernine, "OD1NF1ST: True Skip Intrusion Detection and Avionics Network Cyber-attack Simulation," *ACM Transactions on Cyber-Physical Systems*, vol. 6, no. 4, pp. 1–27, Oct. 2022. [Online]. Available: <https://dl.acm.org/doi/10.1145/3551893>
- [11] D. A. Wiegmann and S. A. Shappell, "Human error perspectives in aviation," *The International Journal of Aviation Psychology*, vol. 11, no. 4, p. 341–357, 2001.
- [12] C. Eidsness, "Securing avionics with a zero trust model." [Online]. Available: <https://ccxtechnologies.com/systemx-avioincs-ids/>
- [13] V. R. Palleti, S. Adepu, V. K. Mishra, and A. Mathur, "Cascading effects of cyber-attacks on interconnected critical infrastructure," vol. 4, no. 1, p. 8. [Online]. Available: <https://cybersecurity.springeropen.com/articles/10.1186/s42400-021-00071-z>
- [14] M. Wolf, M. Minzlaff, and M. Moser, "Information Technology Security Threats to Modern e-Enabled Aircraft: A Cautionary Note," *Journal of Aerospace Information Systems*, vol. 11, no. 7, pp. 447–457, Jul. 2014. [Online]. Available: <https://arc.aiaa.org/doi/10.2514/1.I010156>
- [15] Trellix, "The Threat Report, Fall 2022," 2022. [Online]. Available: <https://www.trellix.com/en-us/assets/threat-reports/trellix-arc-threat-report-fall-2022.pdf>
- [16] U. S. G. A. Office, "Aviation cybersecurity: Faa should fully implement key practices to strengthen its oversight of avionics risks," *GAO-21-86*, Oct 2020. [Online]. Available: <https://www.gao.gov/products/gao-21-86>
- [17] A. B. Garcia, R. F. Babiceanu, and R. Seker, "Artificial Intelligence and Machine Learning Approaches For Aviation Cybersecurity: An Overview," in *2021 Integrated Communications Navigation and Surveillance Conference (ICNS)*. Dulles, VA, USA: IEEE, Apr. 2021, pp. 1–8. [Online]. Available: <https://ieeexplore.ieee.org/document/9441594/>
- [18] S.-W. Lee, H. Mohammed sidqi, M. Mohammadi, S. Rashidi, A. M. Rahmani, M. Masdari, and M. Hosseinzadeh, "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review," *Journal of Network and Computer Applications*, vol. 187, p. 103111.
- [19] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501.

- [20] A. Lališ, V. Socha, P. Křemen, P. Vittek, L. Socha, and J. Kraus, “Generating synthetic aviation safety data to resample or establish new datasets,” *Safety Science*, vol. 106, pp. 154–161, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925753516302260>
- [21] C.-O. Favreau, “Learning to rank with bert for argument quality,” Master’s thesis, Polytechnique Montréal, April 2022. [Online]. Available: <https://publications.polymtl.ca/10296/>
- [22] F. Gosselin and A. Zouaq, “Sorbet: A siamese network for ontology embeddings using a distance-based regression loss and bert,” in *The Semantic Web – ISWC 2023*, T. R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng, and J. Li, Eds. Cham: Springer Nature Switzerland, 2023, pp. 561–578.
- [23] S. Durocher, “Entity typing with natural language inference for fine-grained named entity recognition,” Master’s thesis, Polytechnique Montréal, May 2023. [Online]. Available: <https://publications.polymtl.ca/53392/>
- [24] D. Piché, “Génération de base de connaissance à partir de données hétérogènes dans le monde culturel,” Master’s thesis, Polytechnique Montréal, May 2022. [Online]. Available: <https://publications.polymtl.ca/10327/>
- [25] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” 2019.
- [26] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 815–823, arXiv:1503.03832 [cs]. [Online]. Available: <http://arxiv.org/abs/1503.03832>
- [27] “L1nna datasets-archive.” [Online]. Available: <https://github.com/L1NNA/datasets-archive>
- [28] S. R. Department, “Airline industry worldwide - number of flights 2023,” 2023. [Online]. Available: <https://www.statista.com/statistics/564769/airline-industry-number-of-flights/>
- [29] C. R. Spitzer, *ARINC Specification 429 Mark 33 Digital Information Transfer System*. CRC Press, 2006. [Online]. Available: <https://www.taylorfrancis.com/chapters/mono/10.1201/9781315222240-10/arinc-specification-429-mark-33-digital-information-transfer-system-cary-spitzer>

- [30] M. Anand, S. Dajani-Brown, S. Vestal, and I. Lee, “Formal modeling and analysis of AFDX frame management design,” in *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC’06)*. [Online]. Available: <https://ieeexplore.ieee.org/document/1630505>
- [31] M. H. Naeem, I. H. Abbasi, and M. Mohsin, “An autonomous intrusion detection system for ethernet-based avionics communication bus,” in *2021 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9659587/>
- [32] K. Lounis, Z. Mansour, M. Wrana, M. A. Elsayed, S. H. H. Ding, and M. Zulkernine, “A review and analysis of attack vectors on MIL-STD-1553 communication bus,” vol. 58, no. 6, pp. 5586–5606. [Online]. Available: <https://ieeexplore.ieee.org/document/9780616/>
- [33] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O’Reilly Media, 2017.
- [34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.
- [35] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,” Mar. 2020, arXiv:2003.10555 [cs]. [Online]. Available: <http://arxiv.org/abs/2003.10555>
- [36] T. S. Alec Radford, Karthik Narasimhan and I. Sutskever, “Improving language understanding by generative pre-training,” 2018. [Online]. Available: <https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf>
- [37] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, M. Palmer, R. Hwa, and S. Riedel, Eds. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 670–680. [Online]. Available: <https://aclanthology.org/D17-1070>
- [38] K. Gopalakrishnan, S. K. Khaitan, A. N. Choudhary, and A. Agrawal, “Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection,” *Construction and Building Materials*, vol.

- 157, pp. 322–330, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:116738023>
- [39] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, “Meta-transfer learning for few-shot learning,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 403–412, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:54448258>
- [40] T. T. Nguyen and V. J. Reddi, “Deep reinforcement learning for cyber security,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 3779–3795.
- [41] D. De Santo, C. Malavenda, S. Romano, and C. Vecchio, “Exploiting the MIL-STD-1553 avionic data bus with an active cyber device,” vol. 100, p. 102097. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0167404820303709>
- [42] D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, “Drone hack: Spoofing attack demonstration on a civilian unmanned aerial vehicle,” Aug 2012. [Online]. Available: <https://www.gpsworld.com/drone-hack/>
- [43] A. Costin and A. Francillon, “Ghost in the air(traffic): On insecurity of ads-b protocol and practical attacks on ads-b devices,” in *BlackHat USA*, 2012. [Online]. Available: https://www.researchgate.net/publication/267557712_Ghost_in_the_AirTraffic_On_insecurity_of_ADS-B_protocol_and_practical_attacks_on_ADS-B_devices
- [44] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, “LSTM-based intrusion detection system for in-vehicle can bus communications,” vol. 8, pp. 185 489–185 502. [Online]. Available: <https://ieeexplore.ieee.org/document/9216166/>
- [45] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.
- [46] F. Onodueze and D. Josyula, “Anomaly detection on MIL-STD-1553 dataset using machine learning algorithms,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, pp. 592–598. [Online]. Available: <https://ieeexplore.ieee.org/document/9343073/>
- [47] E. Mushtaq, A. Zameer, M. Umer, and A. A. Abbasi, “A two-stage intrusion detection system with auto-encoder and lstms,” *Applied Soft Computing*, vol. 121, p. 108768, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1568494622001983>

- [48] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.
- [49] A. Manolache, F. Brad, and E. Burceanu, "DATE: Detecting Anomalies in Text via Self-Supervision of Transformers," Apr. 2021, arXiv:2104.05591 [cs]. [Online]. Available: <http://arxiv.org/abs/2104.05591>
- [50] D. L. Marino, C. S. Wickramasinghe, C. Rieger, and M. Manic, "Self-Supervised and Interpretable Anomaly Detection using Network Transformers," Feb. 2022, arXiv:2202.12997 [cs]. [Online]. Available: <http://arxiv.org/abs/2202.12997>
- [51] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System," *IEEE Access*, vol. 10, pp. 64 375–64 387, 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/9794665/>
- [52] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. [Online]. Available: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116>
- [53] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8.
- [54] S. Seo, S. Han, J. Park, S. Shim, H.-E. Ryu, B. Cho, and S. Lee, "Hunt for Unseen Intrusion: Multi-Head Self-Attention Neural Detector," *IEEE Access*, vol. 9, pp. 129 635–129 647, 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9539167/>
- [55] A. Shiravi, H. Shiravi, M. Tavallaei, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers Security*, vol. 31, no. 3, pp. 357–374, 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404811001672>
- [56] M. Memarzadeh, B. Matthews, and T. Templin, "Multiclass anomaly detection in flight data using semi-supervised explainable deep learning model," vol. 19, no. 2, pp. 83–97. [Online]. Available: <https://arc.aiaa.org/doi/10.2514/1.I010959>

- [57] N. G. Markevich and A. Wool, “Hardware fingerprinting for the ARINC 429 avionic bus.”
- [58] M. A. Elsayed, M. Wrana, Z. Mansour, K. Lounis, S. H. H. Ding, and M. Zulkernine, “AdaptIDS: Adaptive intrusion detection for mission-critical aerospace vehicles,” pp. 1–15. [Online]. Available: <https://ieeexplore.ieee.org/document/9928421/>
- [59] R. Yahalom, D. Barishev, A. Steren, Y. Nameri, M. Roytman, A. Porgador, and Y. Elovici, “Datasets of RT spoofing attacks on MIL-STD-1553 communication traffic,” vol. 23, p. 103863. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2352340919302148>
- [60] G. of Canada, “Spot the hack: Intrusion detection systems for avionics networks and bus technologies,” Mar 2023. [Online]. Available: <https://www.canada.ca/en/department-national-defence/programs/defence-ideas/element/competitive-projects/challenges/spot-hack-intrusion-detection-avionics-networks-technologies.html>
- [61] G. Thumbur, N. Gayathri, P. V. Reddy, M. Z. U. Rahman, and A. Lay-Ekuakille, “Efficient pairing-free identity-based ads-b authentication scheme with batch verification,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 5, pp. 2473–2486, 2019.
- [62] Z. Wu and J. Liu, “Blockchain-based trusted avionics authentication for secure ground-to-air communication,” in *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, 2022, pp. 1–9.
- [63] L. Obrst, B. Ashpole, W. Ceusters, M. Mani, S. Ray, and B. Smith, *The Evaluation of Ontologies*. Springer, New York, NY, 2006-08-01 00:08:00 2006. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=822618
- [64] J. Undercoffer, A. Joshi, and J. Pinkston, “Modeling computer attacks: An ontology for intrusion detection,” in *Recent Advances in Intrusion Detection*, G. Vigna, C. Kruegel, and E. Jonsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 113–135.
- [65] C. C. Insaurralde, P. C. G. Costa, E. Blasch, and K. Sampigethaya, “Uncertainty considerations for ontological decision-making support in avionics analytics,” in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, 2018, pp. 1–9.
- [66] R. M. Keller, “The nasa air traffic management ontology.” [Online]. Available: <https://data.nasa.gov/ontologies/atmonto/index.html>

- [67] J. Kindervag, “Build security into your network’s dna: The zero trust network architecture,” Nov 2010. [Online]. Available: <https://www.forrester.com/report/Build-Security-Into-Your-Networks-DNA-The-Zero-Trust-Network-Architecture/RES57047>
- [68] M. Wrana, “Od1nf1st.” [Online]. Available: <https://github.com/MichaelWrana/OD1NF1ST>
- [69] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” 2013.
- [70] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *Journal of Machine Learning Research*, vol. 10, no. 9, pp. 207–244, 2009. [Online]. Available: <http://jmlr.org/papers/v10/weinberger09a.html>
- [71] J. Wei, C. Huang, S. Vosoughi, Y. Cheng, and S. Xu, “Few-shot text classification with triplet networks, data augmentation, and curriculum learning.” [Online]. Available: <http://arxiv.org/abs/2103.07552>

APPENDIX A SMALL HYPERPARAMETERS AND EXPERIMENT RESULTS

Table A.1 Small Transformer hyperparameters

Categorie	Hyperparameters	Values
Model	Input embedding size	32
	Number of attention head	4
	Number of sub-encoder-layers	2
	Transformer feedforward network dimension	128
	Intrusion detection head hidden size	32-16
	Intrusion detection head number of layers	2

Table A.2 Performance and runtime comparison of Small-TRIPT-IDS on seen attacks scenario

Model	Precision	Recall	F1	FAR	AUC	Runtime(s)
TransformerEncoder-IDS	98.5	97.6	98.0	0.30	98.7	11.87
Small-TRIPT-IDS	98.2	96.3	97.2	0.30	98.0	11.87