Detection of contextual anomalies in air traffic data using neural network models

Antoine Chevrot

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UNIVERSITÉ DE
FRANCHE-COMTÉ

THÈSE DE DOCTORAT

DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

ANTOINE CHEVROT

Detection of Contextual Anomalies in Air Traffic Data Using Neural Network Models

Détection d'anomalies contextuelles dans les données de trafic aérien à l'aide de réseaux de neurones

Thèse présentée et soutenue à Besançon, le 13 mai 2022

Composition du Jury :

MARTINOVIC IVAN	Professeur à l' Université d' Oxford	Rapporteur
MICHIARDI PIETRO	Professeur à Eurecom	Rapporteur
GOTLIEB ARNAUD	Directeur de recherche à SIMULA LAB	Examinateur
KOUCHNARENKO OLGA	Professeur à l' Université de Franche-Comté	Examinatrice et Présidente
OLIVE XAVIER	Chercheur à l' ONERA, Toulouse	Examinateur
LEGEARD BRUNO	Professeur à l' Université de Franche-Comté	Directeur
VERNOTTE ALEXANDRE	Post-doctorant à l' Université de Franche-Comté	Co-directeur

N° X X X

ACKNOWLEDGEMENTS

First of all, I want to thank Ivan Martinovic and Pietro Michiardi for accepting their role of reviewer. Having these two persons reading through my work as part of their jury duty is a great honor. My thanks also go to Xavier Olive and Arnaud Gotlieb for both their role of examiner in the jury, but also their help throughout my thesis. I also thank Olga Kouchnarenko for accepting her role in the jury.

Then, I would like to thank my two thesis supervisor, Bruno Legeard and Alexandre Vernotte for their guidance during the past 3 years and a half. They managed to keep me focused during this whole period which is a feat by itself and they overall made my Ph.D. very enjoyable to pursue. I partly owe them my motivation to keep working in the academic area and I am very grateful for the trust they have placed in me.

My Ph.D. was also the occasion to meet awesome individuals which made the days at the DISC go too quickly. I want to thank Fabrice and Fabien as it was a real treat to be their next-door neighbour. I also want to thank the two (ex) Ph.D. student Jean-Philippe and Henri for the great moment we had together. I had also the "chance" to share some long debugging session with Pierre Bernabé. I do not know if I am thankful for this particularly but I am most certainly glad to have worked with him. Finally, these years at the DISC allowed me to work with a great and good soldier of the Java order. A man with an infinite kindness, unless it is monday. Thank you Aymeric !

I want to thank my family for always being supportive and of course some of my friends: Kevin whose size is inversely proportional to the friendship I have for him; Hugo for being Hugo; my friends in Lyon for the hikes, the shelter and all the good moments I had with them for the past 10 years and finally the ones that I forgot to mention.

I could not finish this section without thanking Marta, the polish lady who invaded my life 6 years ago. She has now become my partner in crime and the person who I am certainly thankful the most for helping me through the hardship of writing a Ph.D. thesis.

CONTENTS

ACRONYMS

- **ACAS:** Airborne Collision Avoidance System.
- **ADS-B:** Automatic Dependent Surveillance-Broadcast.
- **AE:** Auto-Encoder.
- **AI:** Artificial Intelligence.
- **AIS:** Automatic Identification System.
- **ARIMA:** AutoRegressive Integrated Moving Average.
- **ATC:** Air Traffic Control.
- **ATM:** Air Traffic Management.
- **BDS:** Comm-B Data Selector.
- **CAE:** Contextual Auto-Encoder.
- **CL:** Climb.
- **CR:** Cruise.
- **CRC:** Cyclic Redundancy Check.
- **DBSCAN:** Density-Based Spatial Clustering of Applications with Noise.
- **DE:** Descent.
- **DSL:** Domain Specific Language.
- **EHS:** Enhanced Surveillance.

- **ELS:** Elementary Surveillance.
- **FAA:** Federal Aviation Administration.
- **FDIA:** False Data Injection Attack.
- **GNSS:** Global Navigation Satellite System.
- **GPS:** Global Positions System.
- **GR:** Ground.
- **GRU:** Gated Recurrent Unit.
- **GS:** Ground Speed.
- **ICAO:** International Civil Aviation Organisation.
- **LOF:** Local Outlier Factor.
- **LSTM:** Long Short-Term Memory.
- **MAE:** Mean Absolute Error.
- **ML:** Machine Learning.
- **MLAT:** Multilateration.
- **Mode-S:** Mode Selective.
- **MSE:** Mean Squared Error.
- **PSR:** Primary Surveillance Radar.
- **RNN:** Recurrent Neural Network.
- **SSR:** Secondary Surveillance Radar.
- **TCAS:** Traffic Collision Avoidance System.

- **VAE:** Variational Auto-Encoder.
- **VHF:** Very High Frequency.
- **VTS:** Vessel Traffic Services.

1

INTRODUCTION

One can measure a society's development by how advanced its transport infrastructure is. With the globalization setting of the 21st century, freight and people's mobility has become compulsory for a country to flourish. Not only that, but transport systems also insure opportunities such as access to employment, education, culture and is directly connected to the quality of life of a population.

In the past twenty years, transport has been revolutionized: online bookstore companies becoming leaders in delivering electronics and everyday-use objects or start-up companies competing with taxi companies modifying how people get their rides are only few out of many examples of the deep changes the sector has witnessed.

One mode of transport that really blossomed in this context is the air transport. With all the technological advances on powered flights brought by the Second World War as well as the advent of digital electronics producing great innovations in flight instrumentation, aircraft rapidly became a very popular mean of transport in civilian mobility.

This popularity led to a whole set of new challenges. For instance, with the appearance of low-cost air companies and the access to cheap flights for any man or woman in the street, the air traffic has seen an unprecedented rise of aircraft in the air, leading to congestion problems. To face these unseen challenges, the Air Traffic Control (ATC) had to innovate to create new ways to better oversee their traffic.

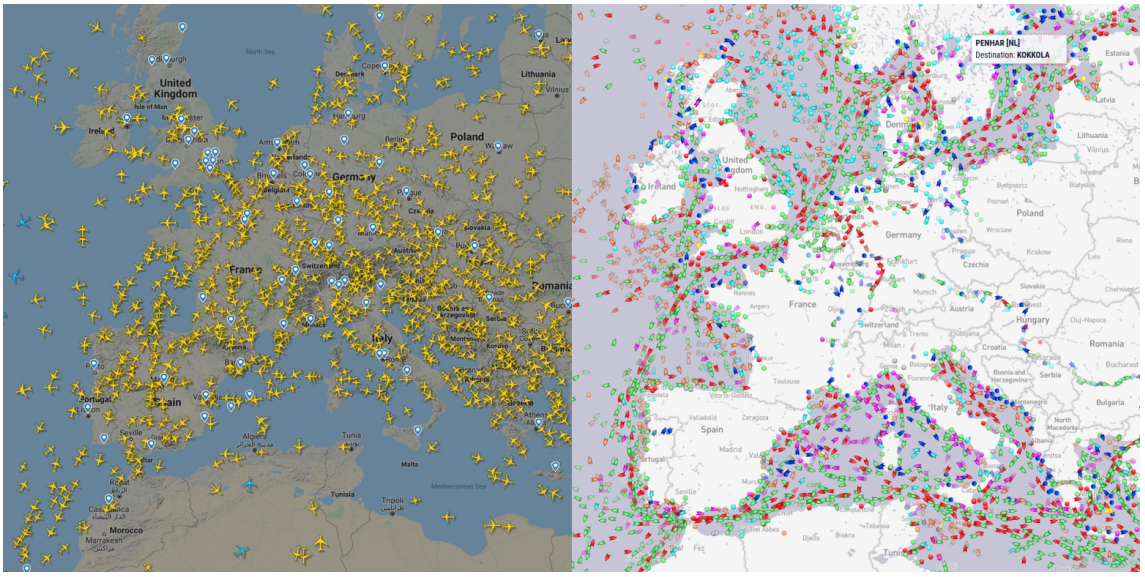


Figure 1.1: Situation of both air and maritime traffic in western Europe on the 15th of September 2021. Courtesy of FlightRadar24 and Marine Traffic.

This introductory chapter aims at giving an overall presentation of the context in which this dissertation takes place as well as the objective of this thesis. The first section presents the technologies used in the air surveillance domain and their use leading to numerous cybersecurity issues. The second section presents how these security failures can be treated, and in particular how Machine Learning technologies can be used for this purpose. Lastly, the final section introduces the different research questions originating from this first analysis that we tried to tackle in the context of this thesis.

1.1/ ADS-B Protocol: A New Paradigm

The original surveillance data in air traffic control come from primary surveillance radars (PSR). This kind of radars work with an antenna that rotates between 5 to 12 rounds per minute, emitting a pulse of radio waves. When reaching any flying object, the wave is then reflected and some of it is returned to the antenna. The antenna is then able to give the range and the bearing of the target in respect of its position by calculating the time difference between the emission and the reception. Due to their design, PSR antennas do not openly transmit their data and

come with a handful of limitations like a zone of the airspace above the antenna not being surveyed (called the cone of silence) or the impossibility to precisely determine an aircraft position without knowing its altitude (level). Due to these limitations, secondary surveillance radars (SSR) have been introduced to use as co-location.

The SSR uses similar types of an antenna as the PSR by transmitting a pulse from a rotating antenna. The main difference is the presence of a transponder in the aircraft receiving this pulse. The transponder, upon receiving the pulse sends back a reply containing only a code if the transponder is in Mode A, but most of the time this code is combined with level information (mode C) or additional information like the aircraft identification, position information etc. (mode S for Select). The information received by the antenna both depends on the interrogation sent and the transponder capacities.

The Mode S being the newest but also the most comprehensive of all transponder downlink interrogation formats, efforts were made to extend its capacity to allow what is called squitters. Squitters are periodic burst transmissions that are self-generated — not sent to answer an interrogation from SSR systems — and are used for two surveillance techniques, namely ACAS and ADS-B. ACAS is the Airborne Collision Avoidance System which is squitter information received by other aircraft to avoid eventual collisions. ADS-B or Automatic Dependent Surveillance-Broadcast uses an extended squitter to send a series of messages broadcasting the aircraft's identification and its GPS position but also navigation status. These data are sent encoded and can be received by anyone decoding them in the vicinity of the transmission. In addition, compared to the PSR or SSR, the ADS-B transmissions do not need massive and expensive antennas to be picked up.

This new technology unlocked a whole new level of access to surveillance data. On one hand, the data being broadcast without being encrypted allows anyone with the right equipment to receive data from an aircraft. On the other hand, the said equipment is cheap compared to other surveillance systems, leading to an increasing number of people having their own antenna to monitor the air.

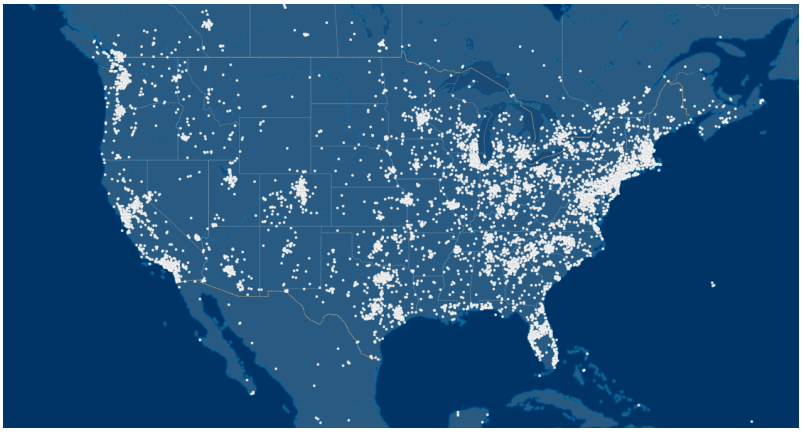


Figure 1.2: Each dot represents a seeder to the FlightAware ADS-B database

However, this accessibility is a double-edged sword. This kind of open-access surveillance data allows anyone to track almost any commercial flight currently travelling, which causes an obvious threat to the security of the observed area. Not only this, but anyone with the right equipment and right knowledge can also simulate and emit fake surveillance data, which is then picked up by traffic controllers. This can be as benign as figures drawn in the sky (see Figure 1.3) but could also lead to major disturbances.

The progressive shift from independent and non-cooperative technologies (PSR/SSR) to dependent and cooperative technologies (ADS-B) has created a strong reliance on external entities (aircraft, GNSS) to estimate the aircraft state. This reliance, along with the introduction of air-to-ground data links via Mode S A/C/S and the broadcast nature of ADS-B, has brought alarming cybersecurity issues. Extensive research can be found in the literature that discuss these issues (Schäfer et al., 2013; Zhang et al., 2017; Strohmeier et al., 2017), stressing that the introduction of ADS-B has enabled a class of attacks referred to as False Data Injection Attacks (FDIAs). FDIAs are a type of attacks that consists in an attacker penetrating the communication system of ADS-B to either create, modify or delete messages sent by aircraft and received by antennas. This could lead to airport receiving false reports of flight crashes, false emergency reports, flooding etc. Still, performing FDIAs on surveillance communications like the ATC requires a deep understanding of the system, its protocol(s) and its logic, to covertly alter

the surveillance flow. These attacks are much more complex to carry out than e.g., jamming, because the logic of the communication flow must be preserved and the falsified data must appear probable. Nonetheless, one can sense the potential for disaster if one of these fake operations were to be executed successfully. It is of the utmost importance that no real threat can be carried out to such a critical infrastructure with human lives on the line.

Of course, other protocols are used in combination with the ADS-B and the radar technologies to secure the traffic. For instance, the voice exchanges between controllers and pilots or the Aircraft Communication Addressing and Reporting System. Nonetheless, the ADS-B stays crucial in the ways of determining an aircraft position thanks to its precision and its possibilities in a crowded area, especially during departure and arrival phases. For this reason, there is a strong need to secure this protocol. However, due to the properties of the protocol, the solutions aiming at strengthening it tend to be costly and often end up modifying how the protocol itself works (Strohmeier et al., 2017). For these reasons, efforts are made to improve the analysis capacities of the ADS-B protocol but differentiating real-life situations from attacks is a challenge for ATC. This need for analysis improvement is one of the motivations for this thesis and the exploration of the state of the art presents different approaches to examine the ADS-B surveillance protocol in order to improve its security.

Figure 1.3: A typical anomalous emission of ADS-B data resulting in a plane drawing pattern

1.2/ How To Improve ADS-B Security?

Many work on securing the ADS-B protocol using different technologies already exist with different degrees of feasibility. First, multilateration techniques or MLAT can be used to determine an aircraft position based on measures of time of arrivals (TOAs) of radio feeds. Each ADS-B message is time-stamped and broadcast by aircraft. If several radars with synchronized clocks and known positions received the same ADS-B feed, then it is possible to calculate the position of the aircraft based on the differences of TOAs.

Using the physical layer information like the strength of a signal, it is also possible to detect intrusions (Strohmeier et al., 2015a) or to use physical phenomena like the Doppler effect (Schäfer et al., 2016) to verify the En-Route positions.

On another level, several solutions were proposed for encrypting ADS-B. Based on the identity of an aircraft (Baek et al., 2017) or through authentication (Cook, 2015) using Public Key Infrastructure (PKI) to try and secure ADS-B but it either suppresses the open characteristics of ADS-B or requires a change in the protocol itself.

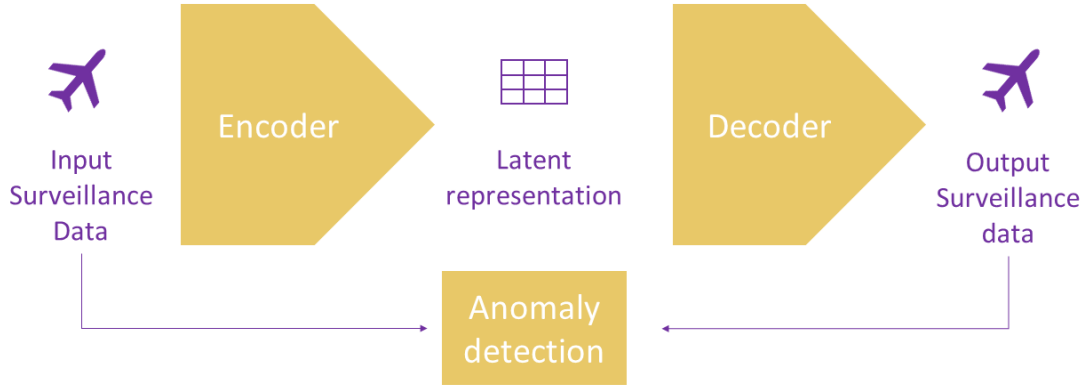


Figure 1.4: Basic architecture of a classic autoencoder for anomaly detection

Among these different existing solutions, some are based on Machine Learning (ML) anomaly detection models. These models already find applications in many

different domains like power systems (Wang et al., 2018) or sensor networks (Malhotra et al., 2016) and have been found quite popular in recent years. One downside of these models is their need for consequent data availability to achieve meaningful results. It is indeed critical for ML researchers to have access to reliable and genuine data sources to train their models. Thankfully, for ADS-B data, the OpenSky Network (Schäfer et al., 2014) and other websites gathering ADS-B data from seeders 1.2 allow an easy accessibility of surveillance data from almost anywhere on the globe.

ML models can be decomposed into 2 different categories of approach: supervised and unsupervised. The supervised ones use labelled data to train models to then categorise the data depending on said labels. The name supervised comes from the fact that these models learn what they are told as if they were children taught by a professor. To schematise, if during training, someone gives an image to a model and labels it as an apple, the model will learn it that way, even if a pear is on the image due to mislabelling.

On the contrary, unsupervised methods give unlabelled data to a model and it tries to separate these data into categories, or clusters. It will not know what is an apple or what is a pear, but it will be able to differentiate them by identifying their characteristic features.

Both approaches could be used as anomaly detection methods. However, the access to genuine data and the lack of anomalous ones in comparison favours unsupervised approaches as they will try to comprehend the features of regular flight during training and then use this knowledge to separate normal behaviours from suspicious ones. Comparatively, supervised models would have way too much data labelled as normal behaviour compared to anomalies and it would have trouble to properly separate them. This phenomena is called class imbalance.

One particular architecture of unsupervised ML called auto-encoder (AE) (Figure 1.4) is often used for anomaly detection and can be found in the literature in many different forms. These models use a first network called the encoder which encodes the input data into a latent representation which is then decoded by a

second network called the decoder. The discrepancies between the input data and the output ones are then used to detect anomalies in the original data. They can be coupled with Recurrent Neural Networks - RNN - to address the temporal nature of the data (Malhotra et al., 2016). Shown to be quite effective, they have already been used in the past to detect different types of anomalies in the ADS-B protocol like en-route trajectory anomalies (Olive and Basora, 2019) or spoofing attempts (Ying et al., 2019). However, there is no existing work giving a clear answer as to how efficient ML models are to detect FDIAs. Work like Habler and Shabtai (2018) only give partial answers to this question on very limited anomaly scenarios.

1.3/ Thesis Objective and Research Questions

From the different existing work using unsupervised architectures to detect anomalies in ADS-B data, none clearly takes into account the context from which data are issued to make decisions. Simply put, the context of a data is a label or background information that provides a broader understanding of an event, a person, or in our case, an ADS-B message. The motivation behind the use of contextual data is that data issued from FDIAs, depending on the context, could be regarded as regular data. For instance, a sudden important drop in altitude is perfectly normal in the context of a descending phase, while it is quite abnormal in the context of an ascending or cruising phase.

Creating an ML architecture using context to detect anomalies can be worth exploring and has led to the following thesis research objective:

Investigate the benefits of adding contextual awareness to unsupervised ML models to better detect FDIAs in air traffic surveillance data.

This research objective led to 3 separate Research Questions (RQ) expressed below:

RQ1: To what extent unsupervised ML algorithms can detect False Data

Injection Attacks in ADS-B data?

Before even introducing the effect of the context on the detection of FDIAs, there is a need to assess the capacities of ML models at detecting anomalous scenarios in ADS-B. ADS-B data are peculiar and fall under the multivariate time-series category. There are already several existing models that are available in the literature to detect these kind of anomalies (Audibert et al., 2020) and some that are already applied to ADS-B data (Habler and Shabtai, 2018) but none clearly state efficiency on lifelike FDIAs on ADS-B data. To answer this research question, experiments are done using different existing auto-encoder architectures on a testing dataset creating our data pipeline. The different results are presented in Chapter 7.

RQ2: Is using the context of a data helpful in detecting anomalies?

ADS-B anomalies often fall under the contextual anomaly category as explained in Chapter 3. Training models without taking into account the different context of the data can lead to a model being trained on all regular data that will only detect outliers in terms of feature aberrations but will omit the anomalies tied to a given context like a hijacking. To avoid that and to answer **RQ2**, one of the main contributions of this thesis is a new model of auto-encoder using data context of ADS-B messages called the Contextual Auto-Encoder (CAE) (Chevrot et al., 2022) that specialises each of its decoders in a given context. Evaluation of the CAE are also presented in Chapter 7 where it is compared to other AE models in order to show the relevance of the approach.

RQ3: Is the CAE a field-specific approach or can it be extended to other transport domains?

From its very definition, a context is field-dependant. Any kind of context defined in the ATC domain will not be applicable in any other domain. This means that for each domain, one needs to define their own context and use the CAE model using these contexts. Therefore, it is only normal to question the efficiency of the CAE model on other domains with other contexts. To show its genericity, in Chapter 8, the CAE model is tested in a different domain, namely the VTS or

Vessel Traffic Services. This domain uses the Automatic Identification System or AIS, a communication protocol very similar to ADS-B with comparable security issues. However, the number of contexts identified for the VTS domain is bigger than for the ATC. For this reason, the CAE needs modifications not to get a too high number of decoders. These improvements are also presented in Chapter 8.

1.4/ Structure of the Thesis

Overall, this thesis dissertation is composed of three main connected topics: the data, the models, and the evaluations, as shown in the flowchart 1.5. The Data module represents surveillance data, and more particularly air traffic data. The Models module represents ML models with their link to data needed for training. The Evaluation module is the final stone to the architecture representing the decisions taken by models given different data. The different modules presented in this flowchart are the backbone of this thesis and the content order of the dissertation is based on it.

The first part presents the background and the related work of the thesis. Chapter 2 gives an overall presentation of the functioning of the ATC and its different surveillance systems. Focus is made on the ADS-B protocol and its potential security caveats. Chapter 3 presents all the related work in terms of FDI-A, how to detect anomalies in ADS-B data and how one can use ML techniques to detect anomalies in time-series. At the end of the first part, Chapter 4 introduces all the different existing tools to receive, aggregate and manipulate ADS-B data for the purpose of training an ML model. It also includes a presentation of FDI-T, a tool that enables the generation of FDIA scenarios.

The second part presents the different contributions and is composed of two different chapters. Chapter 5 presents the main contribution of this thesis: the contextual auto-encoder or CAE. It is introduced as a new mean to take advantage of the benefits of the auto-encoder architecture while taking into account the context of the input data by using several decoders instead of a single one. The second

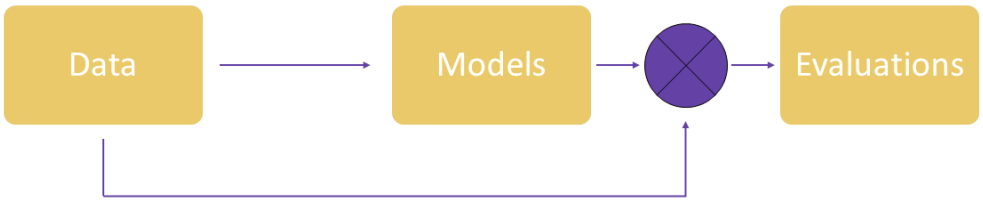


Figure 1.5: Flowchart of the different part of this thesis

chapter of this part, Chapter 6, presents the data architecture created to process the data on top of the already existing work. It includes outlier detection and different pre-treatment to create a training and testing dataset for our model.

Lastly, the third part presents the different experiments using the CAE model. Chapter 7 exposes the different experimental results of different anomaly detection models using a dataset created by the data processing pipeline presented in Chapter 6, including the CAE. The first two research questions are answered in this chapter.

Then, Chapter 8 answers to the last research question by using an improved version of the CAE on the maritime domain. The improvement is the use of an affinity score between the different contexts to group them together, resulting in a smaller model. Only early experiments are presented, however, as it is still a work in progress.

I

Background and Related Work

2

Air Traffic Control

The main domain of application of this thesis is the Air Traffic Control (ATC) and more specifically the Automatic Dependent Surveillance-Broadcast (ADS-B) protocol, its usage, its vulnerabilities and security countermeasures. The first chapter of this first part aims to give the reader a basic understanding of the ATC by giving an overview of the different surveillance systems used by Air Traffic controllers (ATCo) to monitor the air traffic. The already existing technologies and how these work together are developed in its different sections, from primary surveillance radars to the newer Mode-S protocol. This chapter also gives a presentation of the ADS-B protocol and more precisely the type of data found in it and its specifications, how to decode it and the potential vulnerabilities in terms of digital security such a system is exposed to.

2.1/ Overall Description

Airspace surveillance is a critical and complex process whose goal is to detect, localize, and identify all active aircraft in the sky at a given time (Nolan, 2011). Air Traffic Control or ATC uses airspace surveillance to assure the security of the different aircraft by ensuring, for instance, a proper distance between them. This distance is set to be at least 3 nautical miles next to an airport but over 50 nautical miles when aircraft are cruising at sea, where surveillance systems are minimal.

Surveillance systems like radars detect aircraft, identifying their position and their

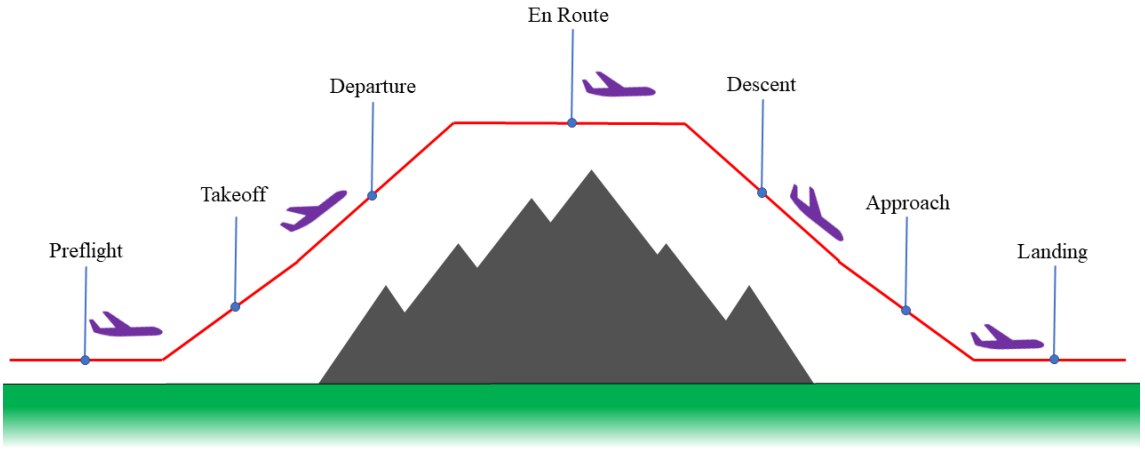


Figure 2.1: The profile of a typical flight.

trajectory. This information is then sent to ATC systems, helping the controllers to properly guide pilots. In zones with low-density traffic, aircraft are usually followed by the controller through textual or voice reports given by the different pilots. This method is unfortunately impossible in high-density zones like around international airports, thus requiring automatic facilities to help the controllers in their task.

With more than $160,000$ flights per day over the world, the task of ATC to keep aircraft from colliding is no easy task and requires good coordination and knowledge. To ensure this, the airspace is divided by the International Civil Aviation Organization (ICAO) into Flight Information Regions (FIR), each under the responsibility of a given country having the operational control within it. Each FIR is then managed by an area control centre (ACC) and, depending on the country, these regions can be subdivided into smaller zones based, for instance, on the proximity of an airport using terminal control (TMC). An aircraft in a specific zone division is thus managed by the authority of the area, giving information and orders to the pilot. When the aircraft crosses the border between two divisions, the previous controller *passes it* to the controller of the new area.

A commercial airline usually crosses several areas during its flight and will be managed differently depending on the phase it is in. The different phases, de-

scribed in the Figure 2.1 can be presented as follow:

Preflight: Pilots in a commercial flight fly using mostly their instrument (Instrument Fly Rules, or IFR) to allow them to fly in most kinds of weather (e.g. above the clouds where it is hard to apprehend the altitude). Pilots using IFR have to submit a flight plan at least a half-hour before the departure. This flight plan includes flight information like the ICAO of the aircraft (a unique aircraft code), the airline name, the type of aircraft, its equipment, the intended speed and cruising altitude, and the route of the flight including departure airport and arrival airport.

Takeoff: Once the aircraft is checked and the flight plan is submitted to the local ATCo, it is then processed to give clearance to the pilot along with information for their incoming flight. The aircraft is then ready to leave its gate to head to the runway and take off. During this time, the ground controller ensures that the aircraft takes the right taxiway as well as the clearance of the aircraft's path on the ground. Finally, when the plane is ready to leave the ground, a local controller checks the airspace around the airport to give final orders for take-off.

Departure: Once departed, the pilot of the aircraft turns on its transponder. It is used to detect incoming radar transmissions as well as broadcasting an amplified and encoded radio signal including different flight data like the flight number, altitude etc. A departure controller using primary radars controls the different flights in the area of the airport (e.g. the TRACON or Terminal radar approach control in the USA). This controller gives an indication to the recently departed aircraft to lead it towards ascent corridors and its cruising phase.

En route: Once the plane leaves the vicinity of its departure airport and enters its En Route phase, it is usually monitored by FIR controllers. The FIR controllers monitor the airspace at both high (above $24,000$ feet) and low altitudes. They provide information about the weather and the traffic to the pilot as well as information about surrounding aircraft. They will follow the aircraft until it passes to another FIR. This is all done using different kinds of information, most of it coming from radars. Radars are usually coupled with other technologies for areas they cannot

reach like over some mountain ranges or out at sea. Most of these technologies use satellite communication to talk with air-traffic control. High-frequency radio can also be used, but is not as reliable.

Descent: The FIR controller directs the aircraft reaching the vicinity of the arrival airport (usually around 250km from it). He directs all the planes flying towards their destination to move from high altitudes to low altitudes and merges all the descending flights into a single line towards the arrival runways. Depending on the traffic and eventual congestions, the flight might be put into a holding pattern, which is a route around the airport where the aircraft wait till the runway can handle the arrival. The FIR controller keeps monitoring the flight till it executes its approach manoeuvres.

Approach: Once the aircraft gets clearance to land, it is handled by an approach controller that directs the pilot to adjust the course of the aircraft with the designated runway. Once aligned with the runway, the local controller from the airport tower takes control for the final landing.

Landing: The local controller uses binoculars and surface radar to check clearance of the runways. Once they make sure everything is safe, the clearance is given for the final landing. The plane is then directed to a taxiway and the pilot is then given the frequency for the ground controller, watching runways and taxiways to direct the aircraft to its appropriate terminal gate.

As seen in the different flight phases, one can sense the importance of the different surveillance systems used and how important it is for controllers and pilots to have precise information. We will now take a deeper look at the different technologies used by ATC.

2.2/ SURVEILLANCE PROTOCOLS

In this section, the different main surveillance systems used by ATC are presented with an emphasis on Mode S protocol and particularly ADS-B, which is the focus

of the remainder of this dissertation.

2.2.1/ PRIMARY RADAR

Primary Surveillance Radars (PSR) are the types of radars one usually pictures when thinking about air surveillance. Even though the theory behind their functioning is rather complex, their principle is intuitive. Radars emit an electromagnetic (EM) pulse and then measure the time it takes to travel a round-trip from them to an aircraft to determine the aircraft's position relative to the radar (Skolnik, 2008). This is possible for 3 different reasons:

- Electromagnetic waves reflect on conductive surfaces meaning that any pulse received back by a radar usually results from it being reflected by the hull of an aircraft.
- The speed of an EM wave is constant in the air and is roughly equal to the speed of light ($300,000$ km per second). Thanks to this property, the distance between the radar and the aircraft can be directly induced by calculating the time an EM pulse took to travel this said distance.
- EM waves travel in a straight line in a constant environment and will only curve when changing said environment. This property, combined with efficient radars sending pulses in a precise direction, helps to determine the azimuths and the altitude of the detected aircraft.

Using these different properties of EM waves, a PSR radar uses a rotative antenna emitting waves in all directions and determines the position of any flying object in its vicinity thanks to their echo. This independent approach offers several advantages. First and foremost, this is a **non-cooperative** approach, meaning the aircraft is passive and does not require onboard devices to work. The other main advantage of PSR is the integrity of the data received as it is very complicated for an adversary entity to temper the electromagnetic waves sent or received by PSR antennas.

Classical radars, or PSR, were created during the Second World War in order to detect hostile aircraft flying in a restricted area and obviously needed a non-cooperative approach to detect them. These technologies held out well in a civilian context to monitor with limited air traffic. However, PSR technologies suffer from several limitations. Indeed, aircraft are not the only objects that can reflect EM waves sent by PSR antennas. Buildings, mountain ranges, thicker clouds or even windmills (Sergey et al., 2008) can create noise and disturb the functioning of PSR (such reflections are called clutter). It also requires a strong enough signal to be able to be received back from an echo, resulting in expensive energy consumption. Moreover, PSR does not allow the identification of aircraft and even though it could determine the altitude of a flying plane, it requires expensive and precise radar that are not often used in a civilian context.

Finally, PSR, by its very way of functioning, has trouble distinguishing targets being at the same slant range but at different levels, which often results in echos overlapping. This problem is actually amplified with the advent of air transportation and an ever-growing number of aircraft in the sky. To face the new set of challenges crowded air traffic brings, the ATC rolled out the Secondary Surveillance Radars (SSR) to assist PSR.

2.2.2/ Secondary Radar

The Secondary Surveillance Radar technologies or SSR (Trim, 1990) embody a switch of surveillance paradigm compared to PSR. Instead of using echoes from the aircraft's hull to scan the air, the SSR introduces communication between the antenna and the aircraft, implying the necessity for a device receiving and transmitting signals in every aircraft. This means that SSR technologies **depend** on the aircraft capabilities of answering to communications. To do so, SSR radars broadcast interrogations that are received by an onboard transponder (transmitting responder) in any aircraft in range. On reception, the transponder emits a reply containing information about its flight. The interrogation format, (also called

uplink format) only consists of two pulses separated by a certain time which determines the mode of interrogation. For simplicity, we will only consider the Mode A (8 µs between the two pulses) and the Mode C (21 µs between the two pulses). Other modes exist but are mainly used in a military context, which is out of scope in this dissertation. The Mode A is the equivalent of the interrogation *Who are you?* and the Mode C is the equivalent of *What is your altitude?*. The interrogation is then answered by the transponder using the reply format (also called downlink format). The identification code is coded on 8 bits, providing 4096 different possibilities entered by the pilot. This code is usually provided by the ATC of the area and can be bound to changes during the flight. Special codes like 7700 for emergencies or 7500 for hijacking are international standards and should be avoided in regular conditions.

Unfortunately, it quickly became difficult to assign a unique code to an aircraft within an area with only the 4096 allowed by the Mode A. Having two aircraft with the same code would lead to safety issues, which, added to other problems like asynchronous interference named FRUIT (for False Replies Un-synchronised In Time) or overlapping replies (also called garbling) led the ATC to develop more advanced communication protocols to acquire even more information from a large number of aircraft.

2.2.3/ MODE S

Created in the 70's and rolled out in the 90's, the Mode S is one of the main sources of information the ATC has at its disposal today. It uses selective unique interrogation (S stands for Select) and provides an addressing capability of 24-bits, which is enough to give a unique registration code for each aircraft. In addition, it improves the data integrity by adding a parity check at the end of each message and also improves the precision of some information like the altitude, passing from a 100 feet minimum increment to 25 by changing the way it is encoded.

Downlink Format	Use
DF0	Short air to air surveillance (ACAS)
DF4	Surveillance (roll call), altitude reply
DF5	Surveillance (roll call), identify reply
DF11	Mode S only All-call reply
DF16	Long air to air surveillance (ACAS)
DF17	Extended Squitter
DF18	Extended squitter, supplementary
DF19	Military Extended squitter
DF20	Comm-B, altitude reply
DF21	Comm-B, identify reply
DF24	Comm-D Extended Length Message (ELM)

Table 2.1: The different downlink format for Mode S transmissions

There are currently 8 different types of uplink formats UF the Mode S is able to handle. As for the downlink format DF, 11 can be sent by a capable aircraft and are presented in the Table 2.1. This difference is due to the extended squitters sent through the DF17, the DF18 or DF19 which are designed to be broadcast automatically without prior interrogation. The main use for these extended squitters is the Automatic Dependent Surveillance-Broadcast protocol, also known as ADS-B.

2.2.4/ ADS-B

The Automatic Dependent Surveillance-Broadcast protocol or ADS-B is a surveillance technology used in addition to the primary and secondary radars to improve the monitoring of airspace with heavy traffic or limited access. It was created in the 90's to use the advances in GPS (Global Positioning System) technologies and more particularly the GNSS (global navigation satellite system) to obtain accurate information of an aircraft. It is, as of today, compulsory to be equipped with an ADS-B transponder to comply with most airspace worldwide.

The ADS-B protocol uses a specific downlink format of the previously presented Mode S called extended squitter that enables aircraft to **A**utomatically broadcast periodically different data gathered from the onboard systems and could not work

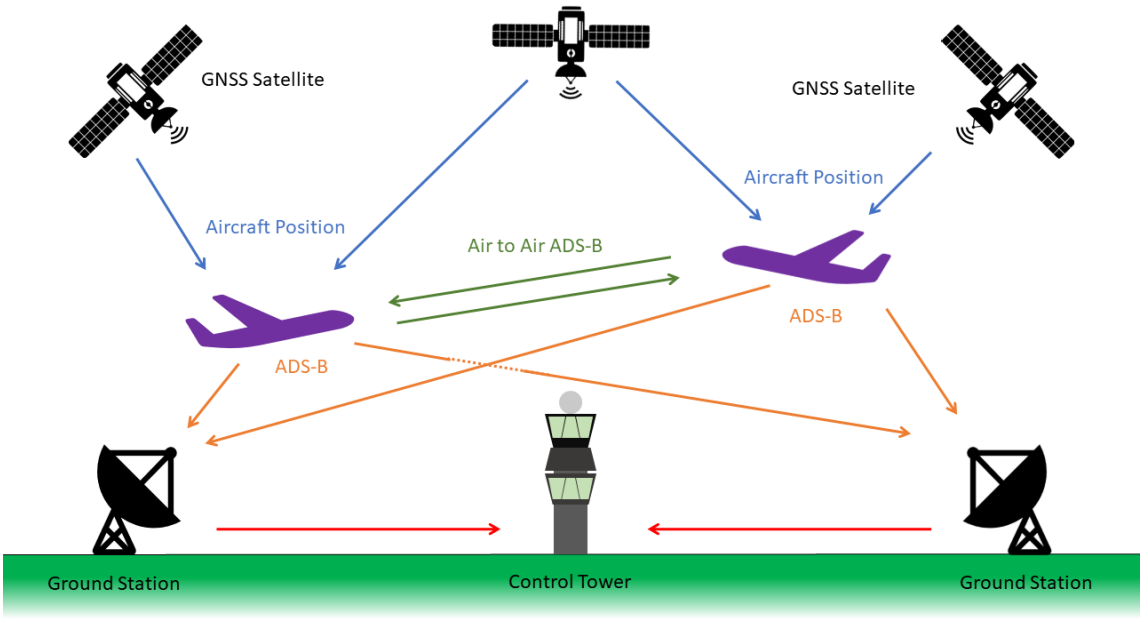


Figure 2.2: The overall mode of operation of the ADS-B protocol

without them (hence the **D**ependant). Its structure is always the same and is represented in Figure 2.3. It is decomposed into 5 different components that can be described as follows:

DF stands for Downlink Format. These 5 bits are used by all the different Mode S types of messages.

CA stands for **CA**pability. This 3-bits part defines the level of the transponder which emitted the message. There are 4 different levels of transponders with increasing receiving and emitting capabilities. For instance, the European Airspace requires all aircraft flying in IFR to be equipped with a level 4 transponder.

ICAO: These 24 bits are used to identify the aircraft which sent the message. It is called the ICAO address because it is allocated by ICAO or the region the aircraft is registered in. This code is often used as a string of 6 characters representing the hexadecimal conversion of the 24 bits. For instance, all flights registered in France have their ICAO code starting with $0011 - 10$, so French flights hexadecimal identifier always starts with 38, 39, $3A$ or $3B$. This code is not supposed to change over the lifespan of the aircraft except for some specific demands for

Figure 2.3: The structure of an ADS-B message.

privacy reasons.

Message Data: This is the core of the message and where the information gathered from the different onboard instruments are located. It is itself decomposed into different parts. The first 5 bits are called the type code (TC) and determine what kind of information is included in the message. As seen on the Table 2.2, depending on the TC, the composition of an ADS-B message will change from position information like GNSS position to identification of the aircraft, including, for instance, its callsign.

Parity Check: or Parity Information is used as an error detection code. It uses a cyclic redundancy code based on a polynomial calculated on both the sending and receiving end to make sure the message is not corrupted.

Type Code	Information
1-4	Identification and Category Message
5-8	Surface Position Message
9-18	Airborne Position Message
19	Airborne Velocity Message
20-22	Airborne position Message
23-27	Reserved
28	Aircraft status
29	Target state and status information
31	Aircraft operation status

Table 2.2: The different Type Code dictating the information contained in an ADS-B message

To summarize, the ADS-B protocol embodies the shift from independent and non-cooperative surveillance technologies, historically used for aircraft surveillance, to dependent and cooperative technologies. It means, for instance, that not only ground stations with antennas positioned at the right angle and direction can re-

ceive position information. Aircraft can now receive squitters from other aircraft, which notably improves cockpit situational awareness as well as collision avoidance.

Another advantage of the ADS-B protocol is that, as we have seen in this short introduction of the protocol, its messages always follow the same structure. This property makes the automation of its decoding and processing easier than the other surveillance technologies (Sun, 2021).

The major drawback of the technology lies in its lack of encryption and authentication, which is discussed in the following section.

2.3/ ADS-B CYBERSECURITY ISSUES

This section presents the different security vulnerabilities a protocol like ADS-B could suffer from and how a malicious person could use them to create undesirable situations.

2.3.1/ ADS-B VULNERABILITIES

The shift from independent and non-cooperative technologies (PSR/SSR (Skolnik, 2008)) to dependent and cooperative technologies (ADS-B) has created a strong reliance on external entities (aircraft, GNSS) to estimate aircraft state. This reliance, added to the fact that none of the aforementioned data link based surveillance technologies (SSR's Mode A/C/S or ADS-B) feature any kind of encryption or authentication, has brought alarming cybersecurity issues. Considering the attacker has the necessary equipment, they can perform three malicious basic operations using the vulnerabilities of the protocol:

(i) Message injection which consists of emitting non-legitimate but well-formed ADS-B messages. This is possible by the lack of authentication in the ADS-B protocol preventing any logical verification.

(ii) Message deletion which consists of physically deleting targeted legitimate messages using destructive or constructive interference. It should be noted that message deletion may not be mistaken for jamming, as jamming blocks all communications whereas message deletion drops selected messages only.

(iii) Message modification which consists of modifying targeted legitimate messages. This modification can be done using different techniques like overshadowing, bit-flipping or combinations of message deletion and message injection. A successful message modification has to respect the syntax of the protocol expected by the control systems. Otherwise, it will often be considered as a message deletion as the message will often simply be ignored by the receptors.

In 2010, a report (Administration, 2010) was published by the Federal Aviation Administration (FAA) on many different aspects of the ADS-B cybersecurity. From confidential analysis, the report assures that the use of the ADS-B protocol does not expose an aircraft more than it was already exposed before its creation and thus, it makes the ADS-B protocol a key component of the increased capacity and safety of the air transportation system. It also claims that there is a low likelihood of malicious exploitation of unencrypted data links. In their analysis, McCallie et al. (2011) disagree with this conclusion by presenting security vulnerabilities associated with the ADS-B implementation.

The lack of encryption of the ADS-B protocol leads to an absence of **confidentiality**. While this makes the process of encoding and decoding data quicker and assures the availability of the ADS-B data to any aircraft or controller that would need it, it also makes most of the air traffic data available to anyone, whether they are simple aircraft aficionados, ill-intended persons or criminal organisations. Furthermore, there is no authentication process from the emitter in ADS-B. The main consequence of this lack of security is the possibility to simulate an aircraft's emissions. This is an issue for secondary radars as the position of the aircraft is calculated based on the physical properties of the signal (its bearing etc), compli-

cating message injections. As for ADS-B, the position and elevation of the aircraft are fully calculated onboard and sent through messages, which allows omnidirectional antennas listening to 1090 MHz communications to receive ADS-B but ease the emission of false data, allowing **authenticity** usurpation and at the same time jeopardizing its **integrity**.

Hence, the ADS-B protocol violates the three key characteristics of the internet security: confidentiality, integrity and authenticity. Also called the CIA triad (Samonas and Coss, 2014), these characteristics can act as a benchmark on how well an information system is secured. These violations automatically lead to vulnerabilities, giving further credit to McCallie et al. (2011)'s analysis.

2.3.2/ Attack Taxonomy of ADS-B Protocol

The three previously presented techniques (Message injection, deletion and modification) allow the execution of several attack scenarios (Schäfer et al., 2013) that can be categorized in the following taxonomy:

- **Ghost Aircraft Injection.** Creation of a non-existing aircraft by broadcasting fake ADS-B messages on the communication channel.
- **Ghost Aircraft Flooding.** This attack is similar to the first one but consists of injecting multiple aircraft simultaneously with the goal of saturating the air situation and thus generating a denial of service of the controller's surveillance system.
- **Virtual Trajectory Modification.** Using either message modification or a combination of message injection and deletion, the objective is to modify the trajectory of an aircraft, for instance, to simulate an emergency scenario.
- **False Alarm Attack.** Modification of the messages of an aircraft in order to indicate a fake alarm. A typical example would be modifying the squawk code to 7500, indicating the aircraft has been hijacked.

- **Aircraft Disappearance.** Deletion of all messages emitted by an aircraft can lead to the failure of collision avoidance systems and ground sensors confusion. It could also force the aircraft under attack to land for safety checks.
- **Aircraft Spoofing.** Spoofing of the ICAO number of an aircraft through message deletion and injection. This could allow an enemy aircraft to pass for a friendly one and reduce causes for alarm when picked up by PSR.

2.4/ SUMMARY

In this chapter, we introduced the main functioning parts of the Air Traffic Control along with its main surveillance technologies including both primary and secondary surveillance radar systems. Using the Mode S and GNSS technologies, the ADS-B protocol enables ground controllers to have a more precise and complete monitoring of the airspace, especially in congested areas. This protocol, however, is a double-edged sword. Its simplicity and access made it vulnerable to spoofing with all the risks it implies for the security of the air traffic. In the two following chapters of this part, the different work carried out to try and improve the ADS-B integrity as well as the different efforts made to acquire and manipulate data will be presented.

3

Related work

This chapter introduces the different core notions in terms of anomalies, attacks and how to detect them as well as the different published work revolving around them. The first section of this chapter introduces the notion of FDIA. Then the second section focuses on the different solutions to detect anomalies specifically used in the ATC domain. Finally, the last section focuses on ML-based techniques on anomaly detection, their different applications and how anomaly detection methods have been applied to the ADS-B protocol so far.

3.1/ False Data Injection Attacks

FDIAs were initially introduced in the domain of wireless sensor networks. A wireless sensor network is composed of a set of nodes (i.e. sensors) that send data reports to one or several ground stations. Ground stations process the reports to reach a consensus about the current state of the monitored system. A typical scenario (Ma, 2008) consists of an attacker who first penetrates the sensor network, usually by compromising one or several nodes, and thereafter injects false data reports to be delivered to the base stations. This can lead to the production of false alarms, the waste of valuable network resources, or even physical damage. Active research regarding FDIAs has been conducted in the power sector, mainly against smart grid state estimators (Dan and Sandberg, 2010). It shows that these attacks may lead to power blackouts but can also disrupt electricity

markets (Xie et al., 2010), despite several integrity checks.

After the apparition of the term FDIA in the sensor network domains, it appeared in different articles centred on how to improve the resilience against these attacks (Zhou and Chakrabarty, 2006; Ozdemir, 2006; Zhu et al., 2007). However, the term was solely used for this domain and no proper definition was given until 2011. Liu et al. (2011) formally introduces the term of FDIA in the smart grids and greatly helps to democratize the term in other fields.

FDIAs also exist in the domain of air traffic surveillance. Since surveillance relies on the information provided by aircraft's transponders to ground stations, aircraft transponders are equivalent to nodes from a wireless network, and ground stations are equivalent to base stations. Although in the ATC domain, there is no real effort needed to penetrate the sensor network, as all communications are unauthenticated and in cleartext.

Still, performing FDIAs on surveillance communications requires a deep understanding of the system, its protocol(s) and its logic, to covertly alter the surveillance flow. These attacks are much more complex to achieve than, e.g., jamming because the logic of the communication flow must be preserved and the falsified data must appear probable.

3.2/ Security Solutions for ADS-B Protocol

Many work on securing the ADS-B protocol using different technologies already exist with different degrees of feasibility. These solutions include the encryption of the data, the analysis of the physical layer, multilateration techniques, and finally ML algorithms.

3.2.1/ MULTILATERATION

The multilateration (MLAT) is a technique used in both military and civil air services to monitor traffic. MLAT determines an aircraft position based on measures of time of arrivals (TOAs) of radio feeds received by at least four different radars. If several radars with synchronized clocks and known positions received the same ADS-B feed, then it is possible to calculate the position of the aircraft based on the differences of TOAs. MLAT can be used to detect ADS-B anomalies (Monteiro et al., 2015) and has the advantage to be very accurate. Recent work from Zhao et al. (2020) also uses MLAT to improve the ADS-B protocol accuracy as well as increase the robustness of the surveillance systems.

Thus, FDIA signals emitted from an attacker next to an airport would be directly discarded by the MLAT due to incoherent TOAs. The main downside to this approach is the necessity of precise time-stamps (usually nano-seconds) which can be hard to obtain due to weather situations or mountain ranges as well as the price induced by having several radars instead of a single one. Finally, Fute et al. (2019) show experimentally that FDIAs can also be created to attack multilateration systems assuming an organized attacker with several devices to emit fake ADS-B proving that MLAT can have ultimately similar issues as ADS-B w.r.t. FDIAs.

3.2.2/ ENCRYPTION

On another level, several solutions were proposed for encrypting ADS-B. Finke et al. (2013) discuss different encryption schemes and highlights the difficulties associated with implementing security protections for the ADS-B environment using encryption. Based on the identity of aircraft, Baek et al. (2017) describe a confidentiality framework to encrypt the ADS-B. Similarly, Cook (2015) uses Public Key Infrastructure (PKI) to try and secure ADS-B but it either suppresses the open characteristics of ADS-B or requires a change in the protocol itself. Further discussion and analysis can be found in a survey by Strohmeier et al. (2015b),

with, for instance, modifications of the current mechanism of the ADS-B protocol by adding a new type of message broadcasting the public key of a given aircraft. Overall, the main constraints for encryption solutions are that it would often mean a modification of the ADS-B protocol which would be rather hard to implement worldwide given it took 20 years to roll out the current version.

3.2.3/ PHYSICAL LAYER

Regarding the use of the physical layer information, Strohmeier et al. (2015a) create an intrusion detection system based on the strength of the signals they received from 2 different sensors. Similarly, Schäfer et al. (2016) use the Doppler shift measurements to verify the En-Route positions of aircraft over time. Using the clocking system of the Mode S sensors, Leonardi (2019) manages to obtain similar results to multilateration systems regarding on-board anomalies without the hassle of having at least 4 different sensors. Yang et al. (2019) uses ML methods such as Gradient Boosting or Support Vector Machine to successfully flag anomalies on the PHY-layer features of ADS-B. Another way of using the physical layer is to check the velocity and the position of aircraft by using the coherence time of electromagnetic waves (Ghose and Lazos, 2015). The main issue of the detection of anomalies on the physical layer is that it goes against the paradigm ADS-B embodies. Indeed, these techniques would require additional instruments to analyse it, negating the main advantage of the ADS-B protocol being its cheaper implementation compared to PSR techniques as well as its fully cooperative approach.

3.2.4/ MACHINE LEARNING TECHNIQUES

Compared to the other kind of solutions presented so far, ML techniques are most of the time data-driven and applied directly to the data carried by the ADS-B protocol, also called logical layer, trying to find abnormalities in the time-series received by air traffic controllers, which could be due to FDIAs. This has the

advantage of not requiring additional radars, nor modifying the protocol. Most of the ML models can be used offline or online and can be quite adaptive depending on the kind of attacks one wants to detect.

Anomaly detection is a wide field that many domains are subject to. For this reason, many solutions exist as of today and the next section gives an overview before presenting more precisely what has already been developed regarding the ADS-B protocol.

3.3/ DETECTING ANOMALIES IN TIME-SERIES

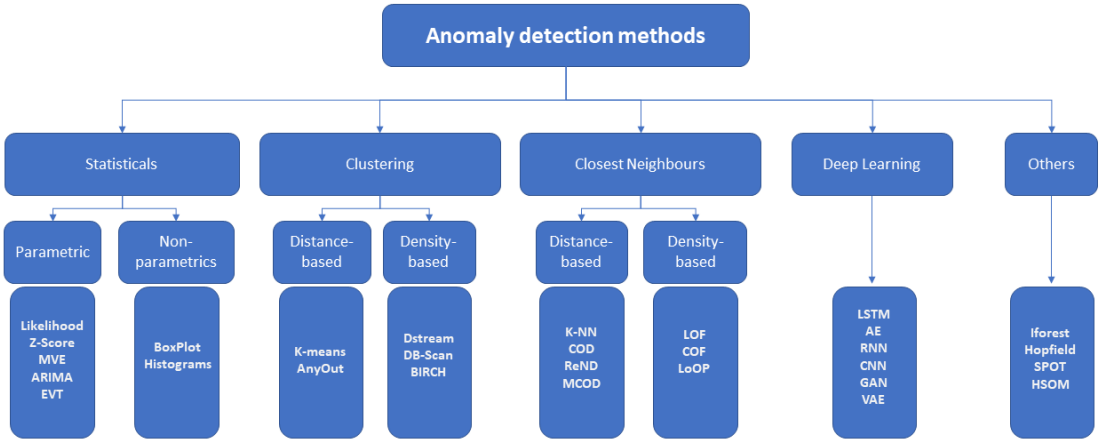


Figure 3.1: A collection of anomaly detection techniques

As we can see in Figure 3.1, the subject of anomaly detection using statistical or ML techniques is a subject that can be found in many application domains and takes many forms. In 2009, Chandola et al. (2009) published a survey presenting different research on anomaly detection, both generic and applied to specific fields. In this survey, they separate anomalies into different categories, each with specific properties. The difference between those categories are mainly based on their nature: individual outliers among a set of data, for instance, are called point

anomalies, while data being abnormal depending on a specific context are called contextual anomalies. The third type of anomaly called collective anomalies is a group of data considered abnormal while individuals inside this group might be considered normal. In the case of anomaly detection in ADS-B data, the first type is often due to encoding errors and can be considered as outlier data to clean up before model training. Contextual anomalies are usually the main anomaly type concerning time-series. Indeed, an example of contextual anomaly is an anomaly detected because it breaks a time seasonality or because the evolution of a feature over time is not going in the expected direction. These anomalies would not be detected as point anomalies because their features' values do not stray away from normality but they are anomalies nonetheless.

Many traditional anomaly detection methods are not fit to detect these types of anomalies in time-series. For instance, most distance-based or clustering methods like DBSCAN (Ester et al., 1996), or IMS (Iverson et al., 2012) are most effective to detect individual outliers in the data but disregard contextual anomalies. Additionally, they are usually suffering from the curse of dimensionality, which is troublesome when working on big time-series datasets. Similarly, ensemble methods like Isolation Forests (Liu et al., 2012), though sometimes coupled with sliding windows (Ding and Fei, 2013) to manage time-dependencies, are also falling behind other techniques to properly detect contextual abnormalities.

To detect contextual anomalies in time-series, one approach would be to transform them first into point anomalies and then use one of the previously cited techniques. For instance, in cell phone fraud detection, Fawcett and Provost (1996) use cell phone usage records to create rules which are used to contextualize data. Once contextualized, the data are then processed by a linear threshold unit to determine if they are fraudulent or not. Similarly, Ma and Perkins (2003) transform time-series into a feature space which can then be processed by a one-class SVM to detect anomalies.

Another way to tackle contextual anomalies is to utilize the structure of the time-series itself. To do so, during its training, a model is only given normal data, with

a regular pattern given a context. Once trained, this model is expected to behave improperly when given abnormal data, hence raising an alert. For instance, statistical models such as ARIMA (Bianco et al., 2001) can be trained to predict future values of a given sequence. ARIMA computes predictions based on a given test sequence and compares it against the real value to raise an alarm or not. Zhao et al. (2017) uses a similar statistical approach to make short-term state forecasting on power micro-grids to detect FDIAs. Other efforts like this one can actually be found to predict FDIAs in smart grids whether using residual functions coupled with a threshold (Ameli et al., 2018) or using deep neural networks (Yu et al., 2018).

3.3.1/ Reconstruction-based Methods

Reconstruction-based methods also use the structure of the time-series itself. As described by Pimentel et al. (2014), reconstruction methods use a lower-dimensional representation of the original data which helps models to separate normal representations from anomalous ones. They are separated into two different categories: subspace-based models and neural networks. While subspace models like PCA can be used to detect anomalies (Dutta et al., 2007), most of the time it is used to reduce the dimensions of a given problem to then use other methods like clustering (Jarry et al., 2020). On the other hand, neural networks and more precisely auto-encoders are widely applied methods to detect anomalies, including anomalies in time-series.

Auto-encoders (AE) are neural network models that have the same number of input and output neurons, with a smaller hidden layer that compresses the original data. As shown in Figure 3.2, it is composed of two different parts: an encoder and a decoder. The encoder takes the input data window and creates a latent representation that embeds the most important features of the original data. Usually, the latent representation is made to be smaller than the original input to force the encoder to retain only the important information, in the same way as a file

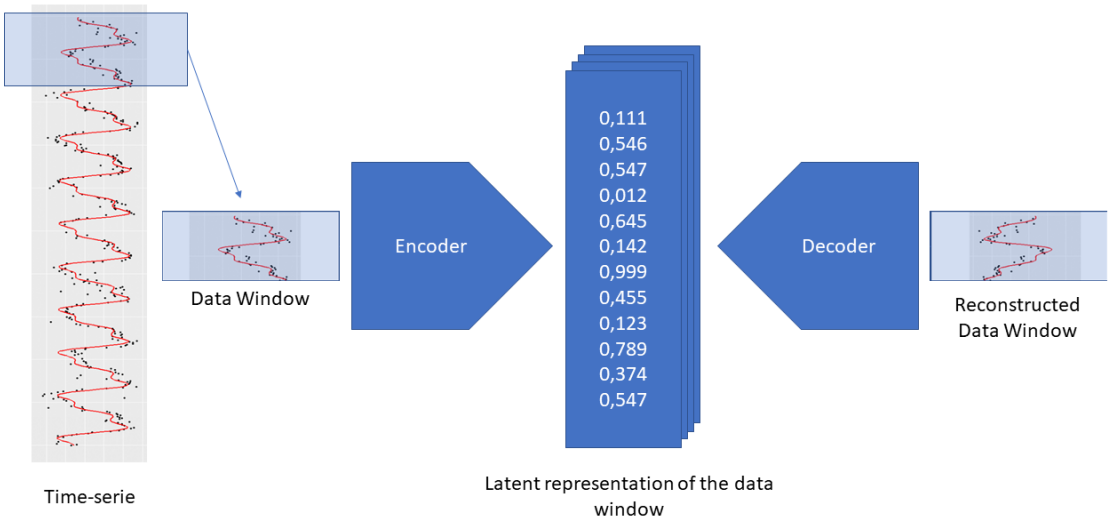


Figure 3.2: Architecture of a classic auto-encoder

archiver would do it.

The decoder, on the other hand, is trained to take this representation and reconstruct the original data back, the same way an archiver would take the compressed data to uncompress it. This type of architecture has a wide range of use like feature extraction (Masci et al., 2011), reinforcement learning (Lange and Riedmiller, 2010), or anomaly detection (Kundu et al., 2020). For anomaly detection, the assumption is that anomalies are troublesome to reconstruct once in a latent representation, leading to a margin between the original and the reconstructed data. This margin is calculated using a reconstruction loss, that is used to train the model through gradient descent techniques.

AEs using regular perceptrons are very basic but this architecture can easily be improved by using other units or by changing the nature of the latent representation, and many new models are based on it. Often coupled with Recurrent Neural Network (RNN) like LSTM (Long-Short-Term-Memory) (Hochreiter and Schmidhuber, 1997) or GRU (Gated Recurrent Unit) (Cho et al., 2014) , auto-encoders (Malhotra et al., 2016) have shown good accuracy to detect coarse anomalies (Habler and Shabtai, 2018), or more specific behaviours (Olive et al.,

2018; Olive and Basora, 2019). Li et al. (2019) use LSTM-based AEs in a generative adversarial network (GAN) as a mean to avoid an anomaly threshold selection but miss proper metrics like recall or precision to correctly prove the efficiency of this method. More recently, Audibert et al. (2020) also created a GAN architecture by using two adversely trained AE architectures sharing the same encoder. The results obtained by this approach are at least as good as the VAE-LSTM approach of Su et al. (2019) on public multivariate time-series datasets.

Indeed, VAE – variational auto-encoder, a stochastic variation of the auto-encoder – are shown to be useful for detecting anomalies in time-series like in the work of Park et al. (2018). Unlike a traditional auto-encoder, which maps the input onto a latent vector, a VAE maps the input data into the parameters of a probability distribution, such as the mean and variance of a Gaussian. Further explanations about this type of model are given in Chapter 7.

3.3.2/ Machine Learning and ADS-B

As stated by Janakiraman and Nielsen (2016), the challenges of detecting anomalies in ADS-B messages are mainly due to the high dimensionality, the time-dependencies and the multivariate nature of the data. Most of the traditional methods of detecting anomalies do not tackle properly all these challenges except for the more recent neural network / deep-learning methods (Chalapathy and Chawla, 2019). The same conclusion in their survey presenting anomaly solutions in aviation applications, Basora et al. (2019) place a great emphasis on AEs to help secure ADS-B protocol. Indeed, even though several efforts toward securing ADS-B using unsupervised ML techniques can be found – Li et al. (2020) uses a hidden Markov model to predict hidden states of the ADS-B protocol and uses them to analyze the deviations during attacks to detect them – their results do not show any advantages compared to other ML solutions like Recurrent Neural Network (RNN) based models in terms of accuracy or false positive rate (FPR).

Habler and Shabtai (2018) are the first to use neural networks to try and secure ADS-B communication by implementing a regular AE with LSTM units to detect hand-crafted anomalies. These first experiments done on several worldwide flights gathered from online sources showed good accuracies on anomalies like velocity drift, trajectory replacement or added random noise. While promising, these first experiments showed only good results on obvious anomalies with a FPR of around 5%, which is quite high for an anomaly detection model.

Akerman et al. (2019) use similar LSTM-AE along with convolutional networks to provide images of the traffic and the anomalies to improve the user experience of such a solution. Luo et al. (2021) use an LSTM-VAE model coupled to a support vector data description (SVDD) model to automatically generate its anomaly threshold showing good results on similar coarse anomalies introduced by Habler and Shabtai (2018). However, as pointed by Su et al. (2019), simply coupling LSTM and VAE together ignores the temporal dependence for the stochastic variables. It also assumes a Gaussian distribution of the z-space of the ADS-B data which can lead to mediocre results depending on the given data.

3.4/ SUMMARY

The different solutions presented to secure ADS-B have shown limited results on FDIAs with some degrees of realism and the literature, while showing interesting models to detect anomalies in ADS-B, lacks proper evaluation on more subtle FDIAs. Experiments presented in Chapter 7 show that these models actually do not perform very well on anomalies where the context of the data is important to be taken into account.

Compared with these presented auto-encoder models, the approach developed in this thesis is a deterministic uneven auto-encoder using a single encoder to create a latent representation of the ADS-B data linked to several decoders, each getting different data chosen thanks to a contextual feature. This architecture, called Contextual Auto-Encoder (CAE) stems from Yook et al. (2020) work on

separating the sound received by speakers placed differently and to the best of our knowledge, was never used in the anomaly detection field. As a result, the latent space created from the single encoder represents the ADS-B data well while the different specialized decoders capture the information in a given context, addressing the variability of the time-series over a certain period of time, resulting in better detection. But before diving into the presentation of this model, the next chapter presents the different existing tools used during this thesis to obtain ADS-B data, enabling the creation of a sane environment to train and evaluate ML models.

4

ADS-B DATA ACQUISITION AND MANIPULATION

In any ML application, the most critical part is the data. Before starting any kind of ML architecture, one needs to assure they have access to a reliable source of data. Of course, the reliability of a data source can be defined in different ways depending on the application but it can generally be broken down into three main factors: its accuracy, its integrity and its accessibility.

An accurate data source ensures that its data are overall correct, consistent and precise. This helps the data scientist to have data in a uniform format properly displaying the situation from where they were taken from. The integrity of a data source, on the other hand, makes sure those data can be trusted, ensuring the data's accuracy was not fabricated and that they remained objective throughout the processing phase. Finally, the constant accessibility of the data is another critical part to have for a data source, especially for research purposes. It helps the reproduction of results by other researchers which would not be possible if experiments were done on a private access database. Secondly, in the case of data bounded to time, long term access to a data source can help show seasonalities in the data and can improve their overall comprehension.

In this chapter, we introduce some of the existing work done in terms of data acquisition and data curation concerning the ADS-B protocol we used during this thesis. Following the architecture in Figure 4.1, first is introduced the data acqui-

sition using the Opensky Network. Then, the different existing Python libraries are presented along with the different ADS-B format used. Our data processing and the model are presented in the next part of the thesis. This chapter focuses on the sources and tools used during this thesis. Most of the choices for the sources were made based on the three criteria developed above.

4.1/ DATA ACQUISITION

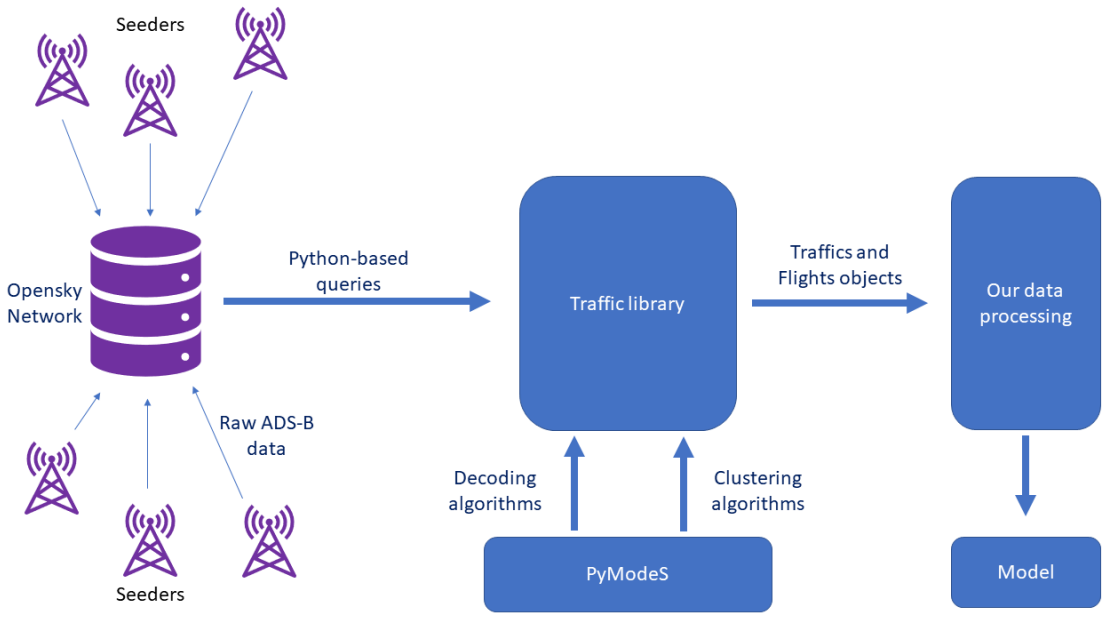


Figure 4.1: Architecture of the data processing

Acquiring ADS-B data is made possible thanks to several factors. With the number of daily flights all across the world's airspace and the broadcast and unencrypted nature of the protocol, anyone with a small antenna linked to a relatively cheap receptor[1] can have access to a reliable source of ADS-B data. However, these data would only come from the vicinity of the antenna which is usually within a 200 kilometres radius around it. Depending on the experiment and the area, this can be plenty of data from many unique aircraft. But in other cases, a more diversified source of data can be needed. To obtain such data, seeded on-

[1] https://shop.jetvision.de/radarcape_en

line databases exists and can be queried to get historical ADS-B data. Following the architecture presented in Figure 4.1, the first step is the gathering of the data through these seeding-fed databases.

4.1.1/ SEEDED DATABASE

The easy access to ADS-B data for anyone with the right equipment was the main incentive that led to the creation of different organisations using a multitude of sensors to monitor the air worldwide. These sensors are connected to the Internet and maintained by volunteers, industrial supporters, and academic/governmental organizations who act as seeders for a common historical database.

FlightRadar24 [2], ADSBExchange [3] or the Opensky-Network [4] are a few examples of these organisations using seeders to compile ADS-B data together.

Taking care of petabytes of data can be cumbersome, especially when the data come from many different sources. Problems related to redundancy, faulty antennas or decoding abnormalities, are a few examples that need to be dealt with when maintaining such a data load. Furthermore, Opensky-Network not only keeps all the ADS-B messages but also generalize their approach to most of the mode S down-link formats, including all-call and roll-call replies.

For this thesis, the main data source used is the Opensky-Network. They provide free access to their historical data to academic researchers and have many seeders across the world, helping them to have desirable coverage. Their database store their data into different tables and different formats with some degrees of data-processing done on it. In the following subsection, the different formats of ADS-B used during the thesis are presented as well as how they can be accessed.

[2] https://www.flightradar24.com/
[3] https://www.adsbexchange.com/
[4] https://opensky-network.org/

4.1.2/ THE OPENSKY-NETWORK

The Opensky-Network (Schäfer et al., 2014) database uses an Impala architecture that can be accessed using SQL-like queries. The data is separated in different tables, each hosting a different type of Mode S data. In particular, for ADS-B data, there are three tables for the data directly received by the network (identification, velocity and position) and one table compiling them together with a pre-processing done by Opensky called the state-vector[5]. For future references, the three tables containing the original messages are called **raw tables**, and the data itself is named **raw data**.

The state-vector table contains all the main information one could find in the different ADS-B messages like the bearing of the flight, its altitude, its speed, its identification etc. Additional to the concatenation of the data, some additional processing is done on the state vectors. For instance, a down-sampling is done to reduce the number of total messages to only have one every other second. In addition, some treatment is done to get rid of the aberration data due to faulty antennas, errors in decoding, missing messages etc. All these pre-processing treatments are done under the hood by the Opensky-Network API.

4.2/ THE DATA PROCESSING

Access to data is easy nowadays and this is especially true when it comes to the thousands of aircraft flying in the air every day. Compiled in a database like the Opensky-Network, the data itself does not provide much value to anyone. However, information extracted out of these petabytes of data is valuable and can provide good insight into an air situation. This section presents the existing different libraries and algorithms used as baseline for our own pre-processing framework.

[5]There is another table with another level of abstraction called flight_data but for simplicity, we only refers to these processed data tables as a state-vector table.

4.2.1/ TRAFFIC

The process of extracting information is hard and the traffic library is a great help when it comes to ADS-B data. Made by Olive (2019) and openly available on his Github page[6], the Traffic library is a Python framework helping users with the data-processing of ADS-B data. It comes with many methods that could be rather tedious to rewrite and with a whole data architecture based on Panda's dataframes using objects called traffics and flights.

Adding to this, it brings an API to the Opensky-Network that allows querying the database using Python functions instead of SQL queries, making easier the creation of a whole Python-based pipeline as presented in Chapter 6.

4.2.2/ TRAJECTORY CLUSTERING

Data from an ADS-B source is chaotic. Whether it is from a single antenna or from a raw table from the Opensky-Network, the data need a heavy treatment before being in a state to be usable by any ML model. As we already introduced in the previous chapters, different types of messages translate into different data in each message of the time-series. Adding to this, ADS-B data are not regular, meaning that the time between messages may vary as opposed to data coming from, eg., a sensor in a power grid.

The algorithm for data curation to transform the raw data into a more regular time-series is straightforward but can be costly in time when dealing with a large amount of data.

From these continuous ADS-B time-series datasets, the next goal of the data processing is flight clustering. The flight clustering takes a time-series of several flights of an aircraft and split it into different flights according to their ICAO code. This is needed because of some later processing done on the data that require to have only data from a given flight without interference from other flights. This

[6] https://github.com/xoolive/traffic

could be done easily by following a simple 5 step method:

1. Group the data into aircraft groups by ICAO address
2. Sort these groups by ascending time-stamp
3. Calculate the time difference between each message
4. Find an average of this time difference to calculate a separation threshold
5. Extract flights from each aircraft group using this threshold

This method is easily implemented but time-consuming and inefficient. Added to the already long pre-processing to create a usable dataset, this would lead to an unusable pipeline to create training data for ML model.

Another solution is to use a clustering method that would take a flight group from the first step of the previously presented method and cluster each of the different flights in its own group. Many different unsupervised clustering algorithms exist and have their own pros and cons. Sun et al. (2017) proposed the use of the DBSCAN algorithm (Ester et al., 1996), mainly for its handling of an unknown number of clusters.

DBSCAN (which stands for density-based spatial clustering of applications with noise) uses the density of the data to create clusters. It uses two main parameters:

- **eps**: It is the distance that specifies the neighbourhoods. Two data points are considered neighbours if they are within this distance. Usually, the Euclidian distance is used as it is the simplest to calculate provided that all the different features are within the same range.
- **minPts**: This is the minimum number of data-point needed to define a cluster

Using these two parameters, the algorithm defines core data-points which are data-points that have more than **minPts** other data-points in their neighbourhood

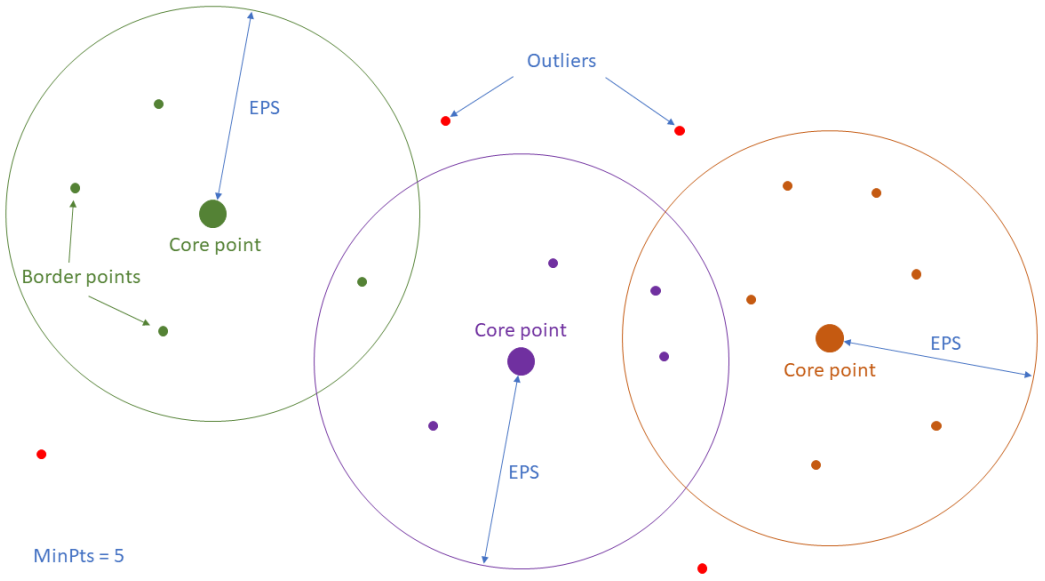


Figure 4.2: Simple example of the application of the DBSCAN algorithm

including themselves. These cores can be considered as the centre of a density-based cluster. All other data points in the vicinity of these cores (i.e. within **eps** distance from a core) are called border points and are included in the closest core cluster. All the other data that do not fit in the two previous categories of data points are considered outliers by the algorithm. The Figure 4.2 shows the different categories of data points used by DBSCAN.

To apply DBSCAN on a set of flights and more precisely to each group of aircraft defined earlier, only the time-stamp is considered to calculate the distance between data-points used to determine the neighbours as using all the features would cause problems due to the differences between aircraft. While this would be a good idea to directly try and eliminate outliers at the same step as clustering the flight, it would be harder to find an eps and a minPts that fit all the flights and all the different types of aircraft. For this reason, only the outliers from the time-stamp feature perspective are treated during this phase.

4.3/ FDIA AND TESTING SCENARIO GENERATION

This chapter has been focused on acquiring regular data to train ML models. To test and validate an approach, one also need a reliable source of data including anomalies that can be used to evaluate the accuracy of a model. While some websites and pages survey many anomalies that happened in the past years[7], this could hardly be transformed into a reliable source of testing material. To overcome this problem, we proposed (Chevrot et al., 2020) a framework that generates synthetic anomalous data covering the whole spectrum of seen and anticipated attack scenarios, as compiled in the already presented taxonomy Strohmeier et al. (2015b). This includes vehicle spoofing and disappearance, but also ghost vehicle injection, vehicle flooding, and virtual trajectory modification. The framework is built upon an existing FDIA testing framework called FDI-T (Cretin et al., 2018), designed for surveillance systems testing. Several extensions were brought to FDI-T in order to turn it into an efficient and intuitive tool for training and testing ML models.

4.3.1/ FDI-T : FALSE DATA INJECTION TESTING

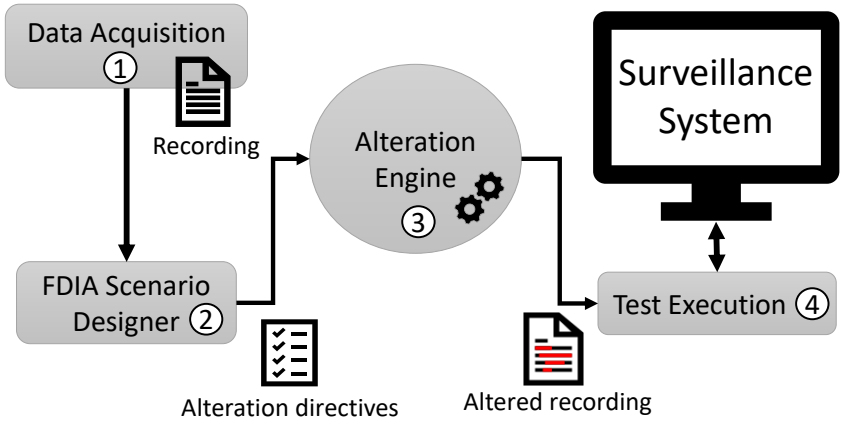


Figure 4.3: FDI-T Framework Architecture

FDI-T[8] is a testing framework that allows domain experts to design FDIA scenar-

[7] https://www.flightradar24.com/blog/category/special-event/

[8] FDI-T is conjointly developed by Smartesting (https://www.smartesting.com) and Kereval

ios by altering (creating, modifying and deleting) recorded legitimate surveillance messages in a fruitful, scalable and productive manner. The altered recordings are then played back (with respect to time requirements) onto real surveillance systems, to simulate an attacker tampering with the surveillance communication flow. The approach mainly aims to evaluate the system resilience against potential security and safety anomalies, and more precisely to FDIAs.

As depicted in Fig. 4.3, the architecture of the FDI-T framework is composed of five components:

① **Data acquisition**. This component collects legitimate surveillance messages (in Beast or SBS formats for ADS-B) obtained either from the Internet or a Mode S receiver. Data take the form of a recording, i.e. a sequence of surveillance messages ordered by reception time.

② **FDIA Scenario Design**. The domain expert defines FDIA scenarios to be applied on a recording obtained via the data acquisition component. FDIA scenarios have various parameters, such as a time window, a list of targeted aircraft, triggering conditions, etc. Once designed, FDIA scenarios are translated into a set of alteration directives, which is the output of the component (an alteration directive is a small modification of the initial recording, usually doable by hand). Scenario design is specified via a Domain Specific Language (DSL).

③ **Alteration Engine**. This component takes as input a set of original sub-recordings, a set of alteration directives, and a correspondence matrix that defines which alteration scenario should be applied on which given sub-recording. It then produces altered sub-recordings in the system input format.

④ **Execution Engine**. The obtained altered ATC sub-recordings are fed to the Surveillance system as if it was receiving live surveillance messages.

(https://www.kereval.com/)

4.3.2/ FDI-T FOR THE TRAINING AND TESTING OF ANOMALY DETECTION MODELS

Although FDI-T was initially designed for surveillance systems testing, it also enables the generation of datasets for training and testing ML-based anomaly detection systems.

Described below are the four features brought to the existing testing framework in order to turn it into an efficient synthetic data generation tool, capable of generating hundreds of thousands of data samples with very little effort.

Labelling. Storing alteration information (which messages were tampered with, what properties were altered) is achieved by directly appending an alteration bitmask at the end of each surveillance message. Each bit of this bitmask is associated with a message field: if the field content has been modified by FDI-T, its corresponding bit is flipped to 1, and 0 otherwise. For example, given an 8 fields message, and only the latitude and longitude (fields no. 3 & 4 in the message) were tempered with, the bitmask 00110000 is appended to the original message.

Massification capabilities. While test engineers usually seek a sharp test suite that satisfies all test requirements with as few test cases as possible, data scientists need the consequent amount of data in order to properly train and test models. FDI-T in its original state allowed users to create one altered recording at a time, and constituting a dataset was realistically unfeasible in these conditions. Thus, massification capabilities to the DSL were developed: users may first define variables containing lists of values, and thereafter reference the variables inside a scenario instead of a value. The framework creates as many single-valued scenarios as there are values in the variable. In the case of a scenario with several references to variables, a single-valued scenario shall be created per combination of variables values. An example of a massified false alarm scenario is presented below:

let *$start*** = {**0,25,40,112**}, **let *$end* = {**120,215,230**},

alter plane from *$start* seconds until *$end* seconds with_values SQUAWK = 7700

It contains two variables, ***$start*** and ***$end*** (containing 4 and 3 values respectively), which are used for defining the alteration start and end time. The framework, therefore, generates $4 * 3 = 12$ single-valued scenarios leading to 12 altered recordings, each having a different start and end time.

Recording-Agnostic scenarios. Initially, the test engineer had to have some knowledge about the recording to be altered.

Again, this is not acceptable in an ML model training/testing context, and two additions to the DSL were needed to fix the issue. First, relevant information (about the recording and vehicle) is gathered and stored as constant in the DSL. Then, it was made possible to use value offsets when altering vehicle properties and defining waypoints. This way, initial values can be unknown by the user. In addition, it makes such scenarios applicable to different vehicles without any change required. The aforementioned additions are illustrated below:

alter plane at 0.5 * **REC_DURATION seconds with_waypoints [**
(>> 0.5,~) **with_altitude** 2000 **at** 0.2 * **ALT_TW**, (>> 0.5,~) **with_altitude** 2000 **at** 0.8 * **ALT_TW]**

The scenario contains two constants representing the recording duration and the alteration time window. Simple arithmetic operations may involve these constants, e.g., 0.5 * **REC_DURATION** equals to half the recording duration. Both waypoints have their coordinates defined using offsets: >> 0.5 represents an 0.5 latitude offset while ~ means preserving the initial longitude. Time of passage is defined as a ratio of the alteration time window **ALT_TW**.

Batch generation. Making FDIA scenarios recording-agnostic means they can be applied sequentially on a set of recordings without human intervention. A batch generation function was added to FDI-T as it could initially only process one recording at a time. It is now possible to supply FDI-T with a set of recordings, and the framework iteratively applies FDIA scenarios on each of the recordings. This

function, combined with the massification capabilities of the DSL, truly enables data scientists to obtain large albeit rich synthetic data sets with few efforts.

4.4/ SUMMARY

The purpose of this chapter was to give a swift presentation of the main tools and libraries used during the major part of the thesis. It can be seen as a preamble to chapter 6 which presents the different contributions done regarding data preprocessing.

Thanks to the different existing tools presented in this chapter, from the raw data extracted in the Opensky-database using traffic, we can extract meaningful information usable in different algorithms like DBSCAN. These information can then be used in tools like FDI-T to create testing scenarios. All the work done on FDI-T, improving its original functionalities, to enable the creation of testing dataset allowing the evaluation of our model was mostly a part of the thesis of Aymeric Cretin (Cretin, 2021).

This ends the first part of this dissertation. The next part presents the main contributions of this thesis starting with a new auto-encoder model called the CAE.

II

Modelisation and Data Architecture

5

CAE: Contextual Auto-Encoder

To detect FDIAs in ADS-B time-series, the context in which the data are issued is primordial. Indeed, a long drop in altitude is perfectly normal in the context of a descending phase, while it is quite abnormal in the context of an ascending or cruising phase. These types of anomalies are called contextual anomalies. As discussed in Chapter 3, while neural network models like auto-encoders are well suited to detect anomalies in time-series, they are usually trained with as many training examples as possible, disregarding the context of the input data. This leads to a generalized model that does not manage to recognise an anomalous scenario as long as its data are not away from the distribution of normal data.

In this chapter, the Contextual Auto-Encoder (CAE), a novel type of anomaly detection auto-encoder architecture is introduced as a new means to take advantage of the benefits of the auto-encoder architecture while taking into account the context of the input data by using several decoders instead of a single one.

5.1/ Motivation

As developed in Chapter 3, the task of detecting abnormal ADS-B messages falls into the category of anomaly detection, and more particularly, in multivariate time-series. A time-series can be defined as successive observations which are usually collected at equal-spaced timestamps. A multivariate time-series $\mathbf{x}$ of length $\mathbf{N}$ is defined as $\mathbf{x} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$, where an observation $\mathbf{x_t} \in \mathbf{x}$ is an M-

dimensional vector at time $t(t \leq N)$, *i.e.* $\mathbf{x_t} = [x_t^1, x_t^2, \ldots, x_t^M]$ such that $\mathbf{x} \in \mathbb{R}^{M \times N}$. The dimension M represents the number of features in an observation $\mathbf{x_t}$. In the domain of anomaly detection in time-series, the goal is to find out if an observation $\mathbf{x_t}$ is anomalous or not.

However, time windows are usually preferred to single observations in order to get a better understanding of the evolution of the data over time. A time window $\mathbf{W_{t-T:t}} \in \mathbb{R}^{M \times (T+1)}$ is a set of $T + 1$ observations $\{\mathbf{x_{t-T}}, \mathbf{x_{t-T+1}}, \ldots, \mathbf{x_t}\}$ from time $t - T$ to t. The goal is then to determine if a particular time window holds anomalous observations or not. Auto-encoder models, like most ML models, train on datasets full of these windows when dealing with time-series. The observations they are based on are usually similar, meaning they hold the same features and these features come respectively from similar sources. Mingling different features or sources too different from each other during training can jeopardise a model due to the data distribution being too uneven.

Nonetheless, having the same features and the same sources does not necessarily translate into uniform data. This statement is especially true for time-series which can variate quite heavily with time. A regular auto-encoder, even if fitted with LSTM units taking into account time dependency, will suffer from variations inside its training data. These variations, among other reasons, can be due to a change of context in the data. The context for a given data observation can be defined as the situation within which it has been created. This situation can either be described by the other immediate surrounding observations, by other external data called meta-data or can simply be apprehended through the knowledge of the data expert. For instance, household energy consumption data are heavily influenced by their context. Not only a given observation is dependent on its predecessors, but it is also dependent on meta-data such as the time of the day, the external temperature on a given day (Fumo and Rafe Biswas, 2015), the weekday, the month (Franses, 1991) etc. These meta-data come often in the form of categorical features, making it difficult to directly include them as features of the observed time-series.

Thus, for the presented approach, every time windows $\mathbf{W_{t-T:t}}$ are associated with a contextual feature $\mathbf{C_{t-T:t}}$ to address these discrepancies. The goal of this feature is to mark differences between time windows whether it is time-wise, nature wise etc. This can be seen as a static feature that is used by the model in the likes of Miebs et al. (2020) but which is not a part of the training per se. In the context of this thesis, the flight phases from which ADS-B time windows are used as a contextual feature.

Using a single auto-encoder for all the contexts can lead to a sub-optimal model trained on slightly different distributions. While it can have a good reconstruction loss, it will however ignore the contextual differences in the data and this is critical when used for anomaly detection. Indeed, contextual anomalies are data that could be considered normal in one given context but completely abnormal in another one. If an auto-encoder is trained by disregarding the context of its data, then a contextual anomaly could not possibly be detected. One approach to tackle the context of the data would be to *divide-and-conquer* the problem, consequently creating one model for each given context. This would work for time-series with limited contextual variations but it would require too much time and space to train these models when the number of contexts increases.

In the next section, we present the CAE, a model separating the different contexts during the training, without the need of having several different models trained.

5.2/ MODEL ARCHITECTURE

The CAE can be described as a single auto-encoder with several decoders, all sharing the same encoder, and each specialized in one context. This architecture by itself has a range of different uses, just like any other classical auto-encoder. The literature shows the usage of auto-encoder architecture to denoise images (Lu et al., 2013), for language translation (Li et al., 2015) and even the more recent transformer architecture are all based on an encoder-decoder structure (Vaswani et al., 2017). In the case of anomaly detection, an auto-encoder takes advantage

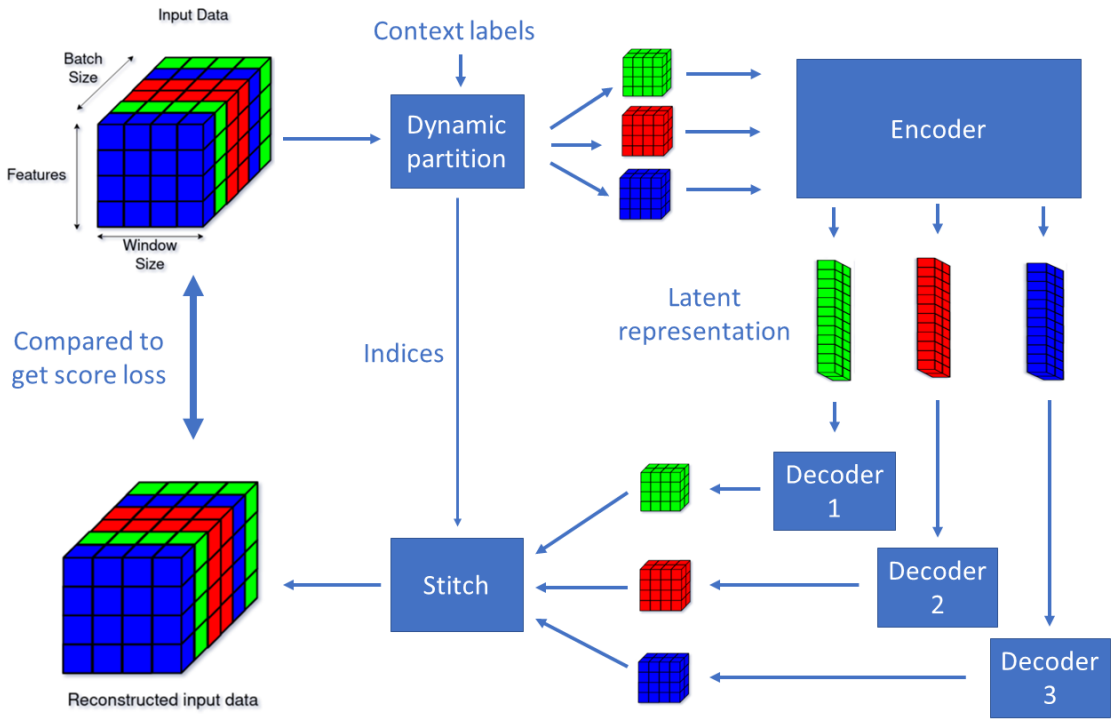


Figure 5.1: The architecture of the CAE model

of the error in the reconstruction of the input to determine its nature. This section explains the different parts of the model that can be found in Figure 5.1.

5.2.1/ The Input and its Context Partitions

The input dataset is constituted of batches of time windows. Each of the batches has a vector of contexts meta-data associated to them, linking every time window inside these batches to the context they are issued.

Through the use of the meta-data, each window of observation is arranged in sub-datasets, one for each context. Therefore, a tensor of windows W is transformed into w_i partition tensors. This is accomplished by using the dynamic partitioning functionality from Tensorflow [1] that allows the partition of data into tensors using the labels as shown in Figure 5.2.

[1] https://www.tensorflow.org/api_docs/python/tf/dynamic_stitch

This function also yields order indices that can be used later on by the stitch function. Once the reconstruction of the different decoders is done, the stitching, which is just the reverse computation explained on Figure 5.2, will merge together the outputs in the order given by the indices to eventually compare this reconstruction to the original data.

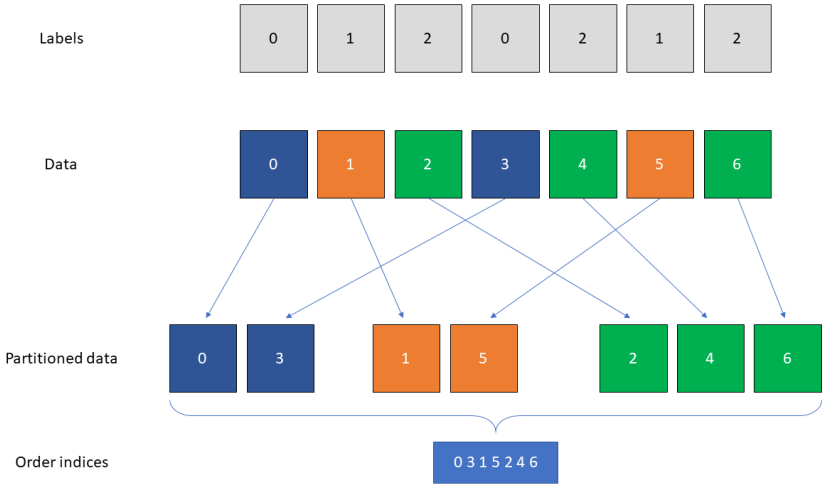


Figure 5.2: An example of how data can be partitioned from labels

5.2.2/ THE ENCODER

The encoder is the first layer of the model. It is unique and transforms all of the original input into its encoded counterpart, also called latent representation. The encoder can be seen as a simple transition function $\phi : \mathcal{X} \rightarrow \mathcal{L}$ where $\mathcal{X}$ is the input space and $\mathcal{L}$ is the latent space. In its simplest form, the encoder function ϕ can be a simple perceptron unit with an activation function. In the case of time-series analysis, one can use recurrent neural networks (RNN) to address the time dependencies of the data. The main problem of classic RNNs is their struggle to learn the long-term dependencies in a sequence because of the gradient vanishing during learning which can be detrimental for long time windows (e.g. ADS-B time windows can be up to 60 seconds long and the model must remember what the state of the aircraft was in this time span). Alternatives to RNNs are LSTMs and more recently the GRU, which do not suffer from the vanishing gradient prob-

lem thanks to a system of gating units. In most cases, these variants perform equally, and while GRU can have fewer parameters on a smaller dataset, LSTMs having a separate update gate and forget gate can be more effective on longer sequences than the GRU. To add up additional information to the latent representation of the ADS-B time windows, a bidirectional mechanism can also be added in the encoder layer in order to use both close past and future to encode the data.

5.2.3/ The Latent Representation

The latent representation $h \in \mathcal{L}$ is the encoded version of the input of the encoder. By itself, the latent space does not have any meaning for a human, and can be considered as a black box representation just like an archived file could be. However, this vector captures the most important features of the input, including the time dependencies thanks to gated units like LSTMs or GRUs. In other words, with a properly trained encoder, h could be used by any model afterwards as if it was the original observation, provided that the said model knows how to extract information from this latent representation.

The dimension of this vector is primordial and is closely related to the type and number of units chosen in the encoder layer. If the latent representation has fewer features than its input counterpart, then the encoder compresses the information to keep only the most important ones. In this case, the encoder can be considered as a dimensionality reduction model. On the other hand, if the latent space is bigger than the input, then while there are still possibilities of learning important representing features of the data, the risk of creating an identity function out of the encoder appears, rendering it useless. Regularisation methods like dropout (Goodfellow et al., 2016) or sparsity of the encoder (Ng et al., 2011) can help avoid this problem.

In the case of windowed data, the time-dependency can be seen as a feature that needs to be addressed by the latent space. This directly affects the size of the latent representation, effectively increasing it when compared to the original number

Algorithm 1: Pseudo-code of the Contextual Auto-Encoder

CAE (W, C)
  inputs : A window of input $W = x_0, \ldots, x_n$; an array of context labels $C = c_1, \ldots, c_n$
  output: A reconstructed window $\hat{W}$; A reconstruction loss array L
  $w_0, \ldots, w_n \leftarrow \texttt{Split}(W, C)$
  $h_0, \ldots, h_n \leftarrow \texttt{Encoder}(w_0, \ldots, w_n)$

  foreach context $i \leftarrow 0$ **to** n **do**
    $\hat{w}_i \leftarrow \texttt{Decoder}_i(h_i)$

  $\hat{W} \leftarrow \texttt{Combine}(\hat{w}_0, \ldots, \hat{w}_n)$
  $L \leftarrow \texttt{MSE}(W, \hat{W})$
  return $\hat{W}, L$

of input. In other words, a latent space with more features than the original data can still be perceived as a compressed representation due to the time-dependent nature of the input. More precise dimension figures are showcased in Section 7 during the experimentation of the CAE.

5.2.4/ The Contextual Decoders

The decoding part of the CAE is what makes it different from a regular auto-encoder. While the encoder encodes all the input data regardless of their context, the decoding part is carried out by several decoders, each decoder being represented by a transition function $\psi_i : \mathcal{L}_i \rightarrow \hat{\mathcal{X}}_i$, where $\mathcal{L}_i$ is the latent representation of the input from the sub-dataset $\mathcal{X}_i$ and $\hat{\mathcal{X}}_i$ is its reconstructed counterpart. Like the encoder transition ψ, each decoder transition function ψ_i can be composed of any kind of neural network unit, from shallow perceptron to deep multi-layered RNNs. Each decoder is independent of another and has its own training depending on the loss of its attributed training data and therefore, could be composed of different elements, depending on the needs for each contextual distribution.

5.3/ Loss Calculation

The main architecture is now presented and as seen in the algorithm1, the loss L is one of the two outputs of the CAE. The loss is the deviation between the input $\mathbf{W}$ and the reconstructed tensor $\hat{\mathbf{W}}$.

The function most commonly used for training auto-encoders is the Mean Squared Error or MSE. This function calculates the mean value of the squared deviations of the model's predictions from the original true values. Given an input window $\mathbf{W}$ of $\mathbf{n}$ observations $\mathbf{x}$ and its reconstructed counterpart $\hat{\mathbf{W}} = \hat{\mathbf{x}}_\mathbf{0} \dots \hat{\mathbf{x}}_\mathbf{n}$, the MSE between these two windows can be calculated as:

$$\texttt{MSE}(\mathbf{W}, \hat{\mathbf{W}}) = \frac{1}{n} \sum_{i=0}^{n} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 \tag{5.1}$$

This loss is also called L2 loss. L2 stands for Least Square Errors and comes from the fact that the error $(\mathbf{x}_i - \hat{\mathbf{x}}_i)$ is squared compared to L1 losses where the absolute value of the error is used instead. An example of L1 loss is the Mean Absolute error computed as followed:

$$\texttt{MAE}(\mathbf{W}, \hat{\mathbf{W}}) = \frac{1}{n} \sum_{i=0}^{n} |\mathbf{x}_i - \hat{\mathbf{x}}_i| \tag{5.2}$$

Overall, the MSE is preferred to the MAE as the deviation is squared, which can avoid loss values going too small, preventing the gradient descent. The only times where MAE can be better than the MSE is when the dataset has many outliers, squaring their value will increase even more their values, throwing off the model's training. This is not a desired behaviour during the training but is valuable during testing to better detect anomalies. Since we make sure there are no outliers in the training dataset beforehand, the MSE is a good candidate for the model.

Other loss functions exist and could be used instead of the presented Mean Squared Error. For instance, Habler and Shabtai (2018) use the Cosine Similarity between their input and output window to calculate the anomaly score of

an ADS-B window. This choice does not have any proven downside to the MSE loss nor has any advantages. For the CAE experiments, the MSE performed the best. The global MSE loss of the model is calculated between input from the input space $\mathcal{X}$ and output from the output space $\hat{\mathcal{X}}$. This loss is used for the gradient descent to train the encoder part of the model. As for the decoders, each decoder is trained on the MSE loss between their own $\mathcal{X}_i$ and $\hat{\mathcal{X}}_i$ spaces.

5.4/ Anomaly Determination

As presented so far in this chapter, for each input of the CAE, a reconstructed output is given by one of the decoders. The deviation of this reconstruction from the original input is called the loss associated with the said input during training but is usually called error during evaluation. A low reconstruction loss means the input fits in the learnt distribution of the decoder it went into. On the other hand, a high reconstruction loss is due to the decoder not being able to properly reconstruct the input data. If the decoder was properly trained and the input data indeed originated from the context the decoder was trained for, then the likely reason for a high reconstruction loss is that the input data originated from abnormal behaviour. The task left to do is then to take a decision on the nature of the original data based on its reconstructed representation output by the CAE.

This section discusses how we can properly differentiate normal data from anomalies using the CAE's error score and the different methods chosen for the experimentation done in this dissertation.

5.4.1/ Anomaly Detection Methods Using CAE's Output

In Section 3 is introduced different ways of using the auto-encoder architecture to detect anomalies, each having its own perks regarding the data they are dealing with. For the CAE, the main particularity is the presence of a different decoder for each given context. Each decoder being trained on their own subset, their

output distribution can quite substantially differ from each other. This affects the way the model can detect anomalies as one method could perform very well on some decoders but very poorly on others at the same time.

In the case of very similar distributions or applications where false positives are not an issue, having the same anomaly detection method for all the decoders is likely to be the best solution as it will not involve any additional calculation. Otherwise, there is a need to associate every decoder to its own anomaly detection system. Complex and time-consuming solutions like the SVDD used by Luo et al. (2021) to calculate thresholds over the output error of the auto-encoder are only viable if the application has a limited number of contexts since this number is directly correlated to the number of decoders of the model.

In the remainder of this section, the chosen method of calculation of a threshold over a given error score distribution is developed. Other solutions like the use of the latent representation of the CAE to train other anomaly models are discussed later on in Section 7.4 as potential improvements of the currently presented model.

5.4.2/ THE 3-SIGMA DEVIATION RULE

To handle the problems related to having different decoders, working directly with their own output error score offers several advantages. First, to refer back to the anomaly taxonomy of Chandola et al. (2009), using the reconstruction loss from the CAE and detecting anomalies based on it can be seen as using the CAE model as a model transforming a contextual anomaly problem into one of the point categories. Indeed, its input is a multivariate time-series and its output is a univariate error score with no time dependency. This means we then can rely on the literature at our disposal on this family of problems. Please note that for future references, after training, we refer to the reconstruction loss as the anomaly score of a given window W. Using the equation 5.1, we simply define the anomaly score given by the CAE as:

$$\texttt{Anomaly}(\mathbf{W}) = \texttt{MSE}(\mathbf{W}, \hat{\mathbf{W}}) \tag{5.3}$$

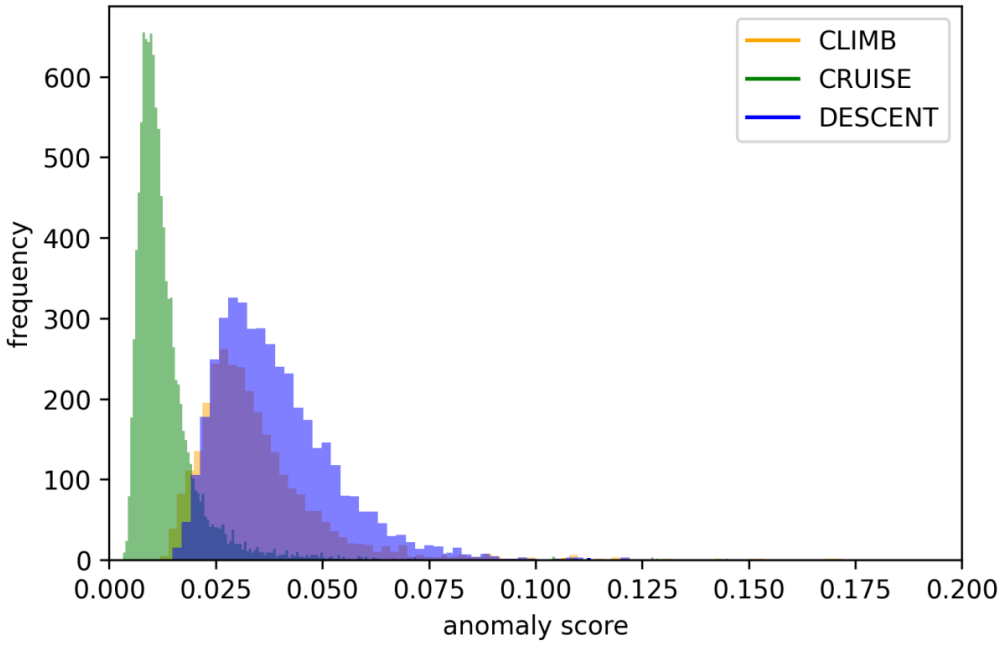


Figure 5.3: The anomaly score's distributions of the training data for each phase

Among the different solutions for point anomaly detection in univariate time-series, in this dissertation, we favoured the 3-sigma deviation rule, a simple and efficient method to determine a threshold on a given normal distribution. Also called the empirical rule or the $68-95-99.7$ rule, it is a statistical rule stating that 99.7% of the data of a normal distribution fall within three standard deviations, noted σ, away from its mean, noted μ. We can use this property to calculate a threshold between normal data and anomalous data on a given anomaly score distribution. This method has the advantage to be rather cost-efficient as only the mean and the standard deviation of the training distribution need to be calculated to determine a threshold for each decoder. Once these thresholds are calculated, one only needs to compare the anomaly score given by a decoder to the corresponding threshold to determine a decision. It is also quite fitting as it is rather simple to store different thresholds for each decoder, making it convenient to use along with the CAE.

However, as we can observe in Figure 5.3, the anomaly score distributions of

the different decoders are following a log-normal distribution more than a regular normal distribution. As shown in Figure 5.4, we can transform our distribution into a normal distribution to calculate our threshold using the three-sigma rule. The green plot is the original distribution following a log-normal distribution. Once the natural logarithm function is applied to it, the red plot is obtained, following a Normal distribution. From this normal distribution, a threshold can be calculated using the 3-sigma rule deviation. Then, we can apply the exponential function to this threshold to obtain the final threshold usable on our distribution and use it seamlessly.

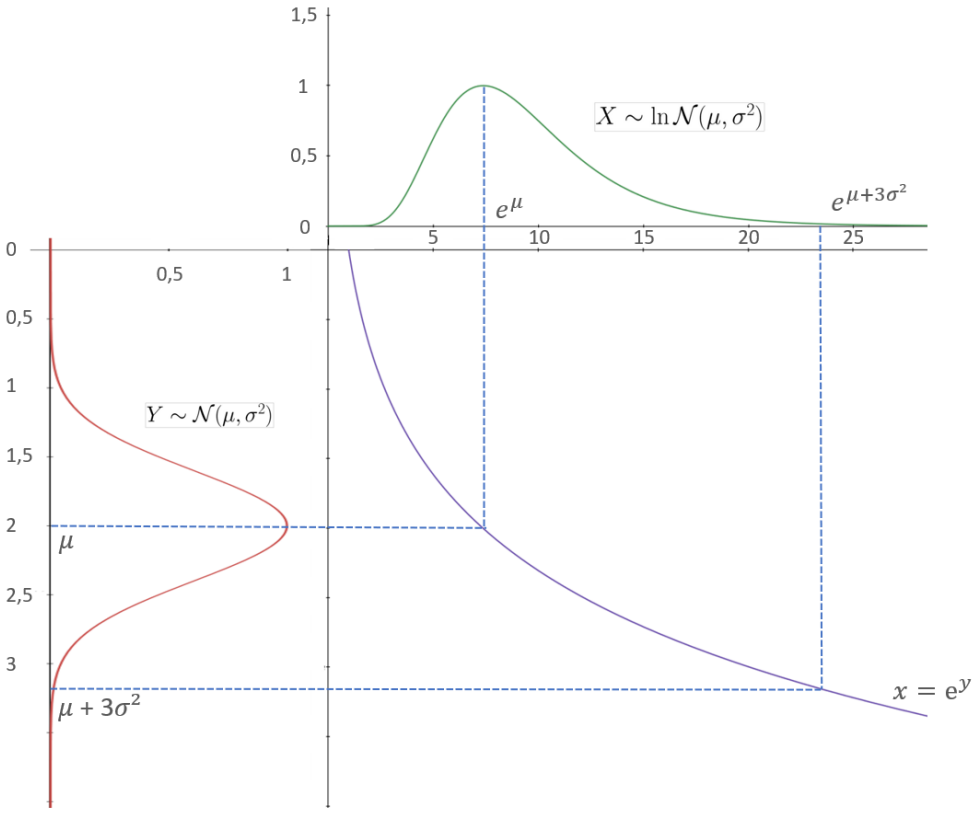


Figure 5.4: shows the equivalence between a log-normal distribution and a normal distribution thanks to the exponential/natural logarithm function.

5.5/ SUMMARY

In this chapter, we introduced the main contribution of this thesis, the Contextual Auto-Encoder architecture. The novelty of this architecture is its awareness to

the context of the data, allowing for the detection of contextual anomalies. Compared to a classical AE that encodes and decodes data regardless of their origin, the CAE separates the different data into different decoders, one for each of the context identified while keeping a single encoder.

Combined with a method to calculate an anomaly threshold, the CAE is a good candidate to detect anomalies in multivariate time-series. For the experiments of this thesis, a 3-sigma deviation rule was chosen for the threshold selection as it is an efficient way of finding the outliers of a statistical distribution.

In order to prove the efficiency of this method on ADS-B data, the next step is to create an evaluation set-up to try and compare this model against other anomaly detection models. The following chapter presents the whole data architecture created around the CAE during the thesis. It ties together the model with the data gathered through the Opensky network to create a safe environment for evaluations.

6

ADS-B DATA PRE-PROCESSING

The aim of this dissertation is not only to present a new type of auto-encoder model able to detect anomalies in ADS-B time-series, but also more generally an architecture able to use a raw ADS-B feed as input using said model. In fact, an ML model like the one presented in the previous chapter is only as good as the architecture revolving around it. The importance of the quality of the data going in the model has already been discussed and it highlights the necessity of having a good data structure prior to the ML model. This is the reason why a hefty part of the thesis was dedicated to the understanding and the processing of the ADS-B data in order to properly experiment on the detection of anomalies in them.

This chapter, based on the work presented in chapter 4, presents the full data architecture, from raw ADS-B data to a dataset of time-series usable by the CAE model for both training and evaluation. While the presentation of the model in the previous chapter tries to be domain agnostic, this data pre-processing part, while giving some general insights about ML structure, is centred on the ADS-B data.

6.1/ ML PROJECT STRUCTURE

Before diving into the ADS-B pipeline implemented during this thesis, let us take a step back and have a more generic approach to what an ML project looks like. Figure 6.1 presents the hierarchy of a whole ML project from data collection to deep-learning-based model training and testing.

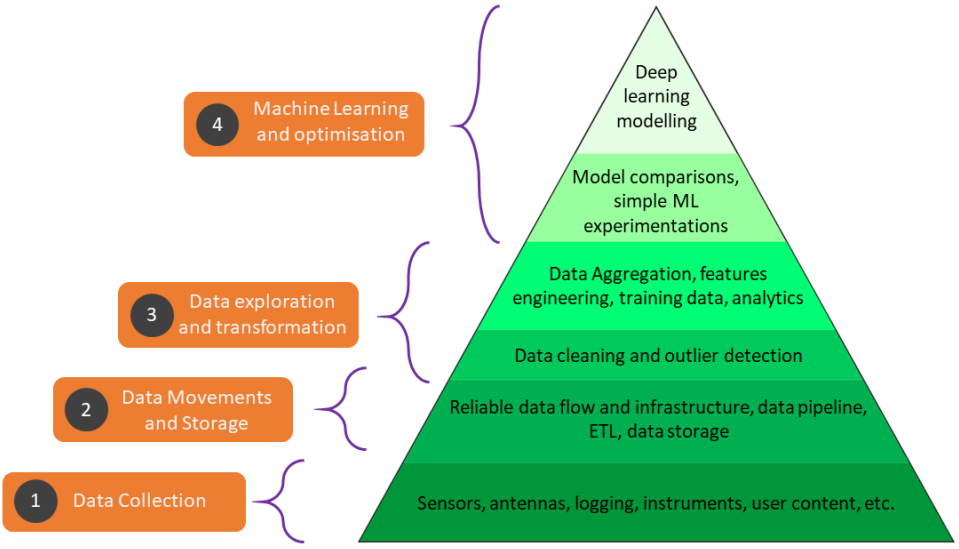


Figure 6.1: The structure of an ML project.

This pyramid can be decomposed into 4 different main parts, each including one or more layers of the pyramid:

1. **Data Collection**. The first step for any ML project is to ensure the accessibility to data. The amount of data needed to train a model depends on its type but in general, deep learning models do require more than other classical ML models, as they usually underperform when having limited access to data (Barbedo, 2018). The data collection step may not be done by the scientist itself, but can come from the user's data or from a range of different data collectors like sensors, antennas ...

2. **Data Movements and Storage** The second stage is focused on the data recuperation and its storage. A good source of data can only be valued if its data can be accessed in a timely manner with as few constraints as possible. For this reason, an efficient data pipeline working along with a database to store the collected data enables the conservation of a historical set of data that is quite beneficial when working with time-series.

3. **Data exploration and transformation** Now that the data is collected and

properly stored, the actual data pre-processing can start. This part is arguably the most important in a project. Indeed, stored data is mostly useless by itself. For instance, gigabytes of ADS-B messages received from many different flights in random orders do not have much significance and do not give any insights about the traffic situation. To gain knowledge out of these data, exploration and diverse transformation need to be applied to them. Whether it is clustering, sorting, outlier cleaning etc., these manipulations dictates what can be done from this information.

This layer is also the moment where the first ideas for the chosen ML model are drafted. Thanks to the analysis of the data and the transformation of erratic data into engineered features, the additional insights on the problem allows for choosing the fittest model for it.

4. **ML and optimisation** The *last part* of an ML project is the design, experimentation and comparisons of different ML algorithms. While this last part is certainly considered as being the front window of an ML project, it is only efficient if the base layers are solid. The great results of deep learning models are mainly due to recent access to huge databases as well as pipelines and machines able to handle them.

This generic architecture can be used in many different domains by different ML models. This shows how a whole ML project relies on the 3 first layers and then the ML / Deep Learning layer comes to the top and evolves depending on the task that needs to be done on the data. The particularity of the ML architecture and its different models is how the top layer of the data pyramid is switchable with other top layers, eventually coming from other domain's pyramids. This particularity, called transfer learning is one of the main reasons why the artificial intelligence research domain has grown so quickly in the past years. Chapter 8 tries to prove this property of the CAE by experimenting on the maritime surveillance domain.

6.2/ The Overall Architecture of our Project

Figure 6.2 presents the general architecture implemented during this thesis. Though some of it was already presented in Chapter 4, this chapter aims to present a more detailed view on the ADS-B data pipeline, highlighting our contributions.

Most of the said contributions have been regrouped in a Github library called Scifly[1] along with the different models used during the evaluation.

Most of the data preprocessing is based on back and forth calls between traffic and scifly. This is due to the baseline data types offered by traffic that intervenes with the processing all along the pipeline.

The rest of this chapter presents the different parts of the Figure 6.2 by following the order of the different layers of the data processing presented in the previous section.

6.3/ ADS-B Data Acquisition and Formating

The data acquisition is the first part of the ML project and might be the most crucial one. In Chapter 4, we described how ADS-B data can be gathered from antenna feeds that anyone could get at their home. However, a more convenient choice is to use a historical database archiving ADS-B data from feeders in every part of the world like the Opensky Network. Not only does it take care of the data acquisition, but it also manages the data storage using an Impala database. On top of that, the traffic library (Olive, 2019) greatly simplifies the data acquisition from this database. At the beginning of this thesis, however, the traffic library — which was still a recent addition to the ADS-B data processing world — did not have the capabilities to query the raw data tables and mainly relied on the processed state_vectors table. While this could have been usable to train ML models,

[1] https://github.com/Wirden/scifly

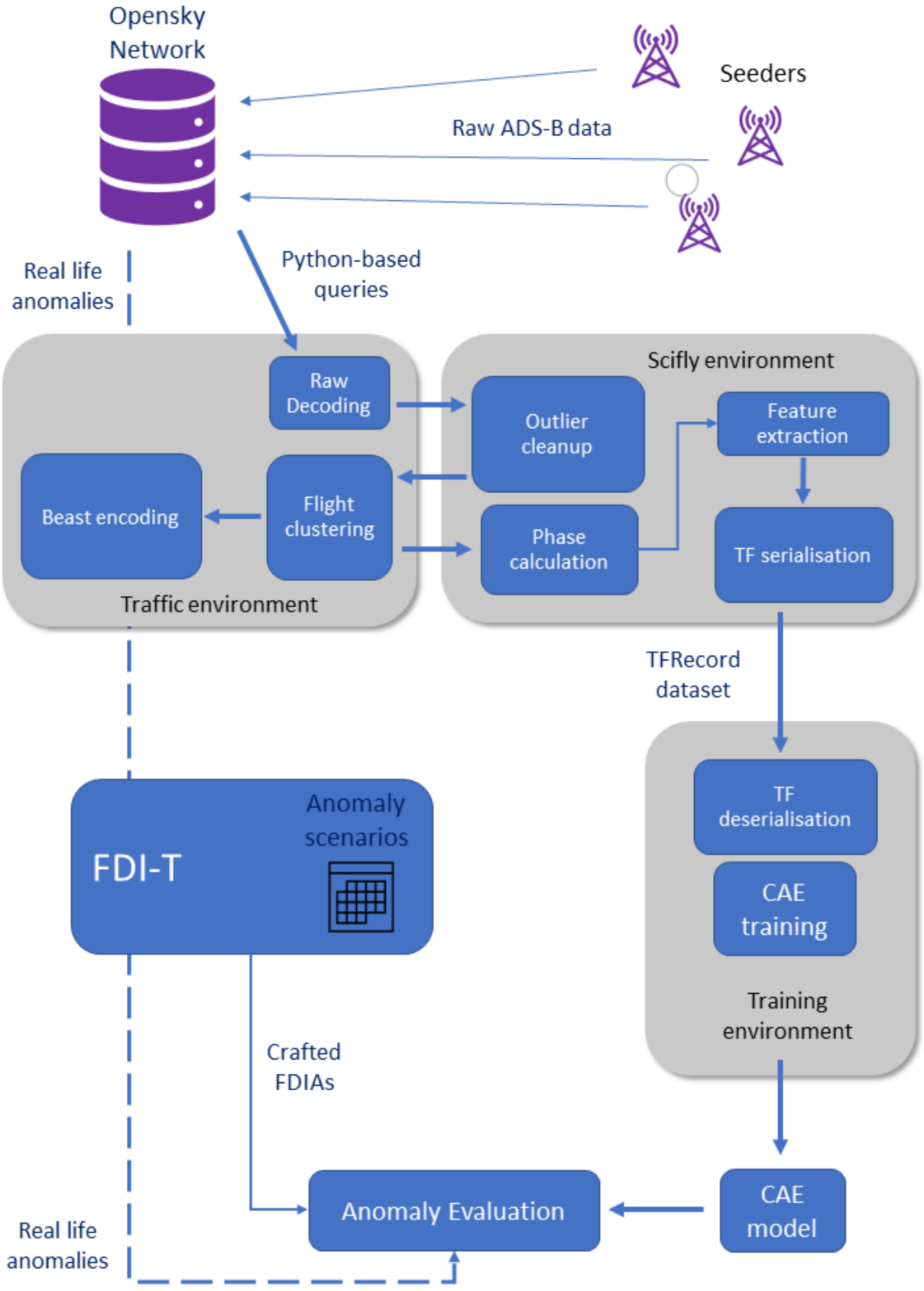


Figure 6.2: The overall architecture is presented in this dissertation.

not having full control on the data pre-processing like the outlier management or the resampling is not ideal in a research context. This is why one of the early work of this thesis was to try and access the raw ADS-B data tables using the Traffic query API. These efforts were included in the traffic library through a software contribution [2]. This allowed the use of raw data directly using the traffic dataframes, easing our own data cleaning on raw data.

Another format task is the transformation of the raw data into formats readable by FDI-T, the FDIA framework like the BEAST format.

6.3.1/ THE BEAST FORMAT AND FDI-T

The BEAST format is a binary format originally created for multilateration (MLAT) technologies. Indeed, MLAT needs very precise time-stamps to be able to precisely triangulate the position of an aircraft given the time of reception of one message from different antennas. The BEAST format is simply a raw ADS-B message, preceded by a @ and a 12 bytes time-stamp. This is a convenient way to converse with the FDI-T module which needs both time-stamp and the raw ADS-B message to work on. Having this part in our pipeline also ensures that data directly comes from Mode S decoders like the Dump1090[3] can already be used seamlessly. The encoding into this format is now included in the traffic library thanks to the collaboration above-mentioned.

SBS Format:

MSG,4,5,211,4CA2D6,10057,2008/11/28,14:53:49.986,2008/11/28,14:58:51.153,,,408.3,146.4,,,64,,,,,
MSG,8,5,211,4CA2D6,10057,2008/11/28,14:53:50.391,2008/11/28,14:58:51.153,,,,,,,,,,0
MSG,4,5,211,4CA2D6,10057,2008/11/28,14:53:50.391,2008/11/28,14:58:51.153,,,408.3,146.4,,,64,,,,,
MSG,3,5,211,4CA2D6,10057,2008/11/28,14:53:50.594,2008/11/28,14:58:51.153,,37000,,,51.45735,-1.02826,,,0,0,0,0

Beast Format:

@016CE3671AA8A800199A8BB80030A8000628F400
@016CE3671BA8D4840D6202CC371C32CE0576098

Figure 6.3: Difference between SBS and BEAST formats

[2] https://github.com/xoolive/traffic/pull/68
[3] https://www.rtl-sdr.com/tag/dump1090/

Another format also output by the Dump1090 decoder is the SBS BaseStation format (difference between BEAST and SBS can be seen in Figure 6.3. This format is the original format read by FDI-T and has the advantage to be easily readable by a human. It is a CSV format including most of the information that can be found in a raw ADS-B message. It includes a human-readable time-stamp with a date and has different types depending on the information included in the message.

To summarize, through a contribution to the traffic library to allow the retrieving of raw data as well as the transformation of ADS-B data into different formats, we now have a functioning interface to receive data directly issued from antennas. The next steps are to clean these data and transform them into datasets readable by ML models.

6.4/ DATA CLEANING

Having an operational data source, with an accessible and reliable historical database makes up the 2 first layers of the pyramid presented in the Figure 6.1. This is done by the combination of the Opensky Network and the Traffic library. The next step is the processing of the raw ADS-B data and in particular, the extraction of flights using clustering, the outlier detection and removal as well as the identification of the different phases of the flight for the purpose of using them as contextual features with the CAE.

6.4.1/ DATA CLUSTERING AND OUTLIER DETECTION

To use the flight clustering presented in4.2.2, the input data need to have limited outliers. Indeed, DBSCAN using the proximity of its neighbours to determine if a data point is in a cluster or not eventually takes care of many outliers. However, more often than not, as shown in Figure 6.4, errors from an antenna can come in chunks and these types of errors throw off DBSCAN which then consider these

chunks as separated flights, resulting in flights cut in half.

To prevent that, we added a treatment of outlier detection and deletion prior to the data clustering in order to get rid of flights with too many aberrations in them. This treatment could not be done by classical outlier detection algorithms like the Local Outlier Factor (LOF) (Breunig et al., 2000) or other types of unsupervised treatments because of how they detect outliers. Indeed, just like DBSCAN, these algorithms use a proximity factor with their neighbours and they tend to cluster abnormal chunks together and don't consider them as anomalous.

This treatment uses the proximity property of ADS-B messages to check the distance between close messages and further messages. The algorithm 2 explains the treatment done on every message. The main idea of this algorithm is to check the distance between a given message and its successor as well as the hundredth message after it. To do so the Vincenty distance Vincenty (1975) — a formulae used to calculate the distance between two points on a spheroid — is calculated between the $ieth$ and the $ieth + 1$ message (line 2) and the $ieth$ and the $ieth + 100$ message (line 3). Then on line 5, the messages with a distance higher than the set thresholds are filtered out.

Commercial flights having a quite regular ground speed, these distances are most

Figure 6.4: Data discrepancy from a flight between Madrid and Moscow

Algorithm 2: Pseudo-code for the outlier filtering

inputs: M: An array of ADS-B messages from the same ICAO ;
ϕ_f, ϕ_c: thresholds to consider a message as outlier

for $i = 1$ **to** $M.length$ **do**
    $M[i][dist_{close}] \leftarrow \texttt{Vincenty}(M_{coord}[i],\ M_{coord}[i+1])$
    $M[i][dist_{far}] \leftarrow \texttt{Vincenty}(M_{coord}[i],\ M_{coord}[i+100])$

$M \leftarrow M[M[dist_{close}] < \phi_c\ \&\ M[dist_{far}] < \phi_f]$

of the time the same. To take advantage of this, we set a threshold on both of those distances based on a pre-calculated average and filter out the message above these thresholds. Having two distances calculated ensure we filter single outliers but also the chunks of anomalous data.

6.4.2/ Phase Identification

The main contribution of this study is the use of the contextualisation of the data to detect anomalies (the context of a data point is defined in chapter 3) in ADS-B data. One of the obvious and easier contexts to grasp for the ADS-B data is the phase of a flight and it is one that can be quite easily identified.

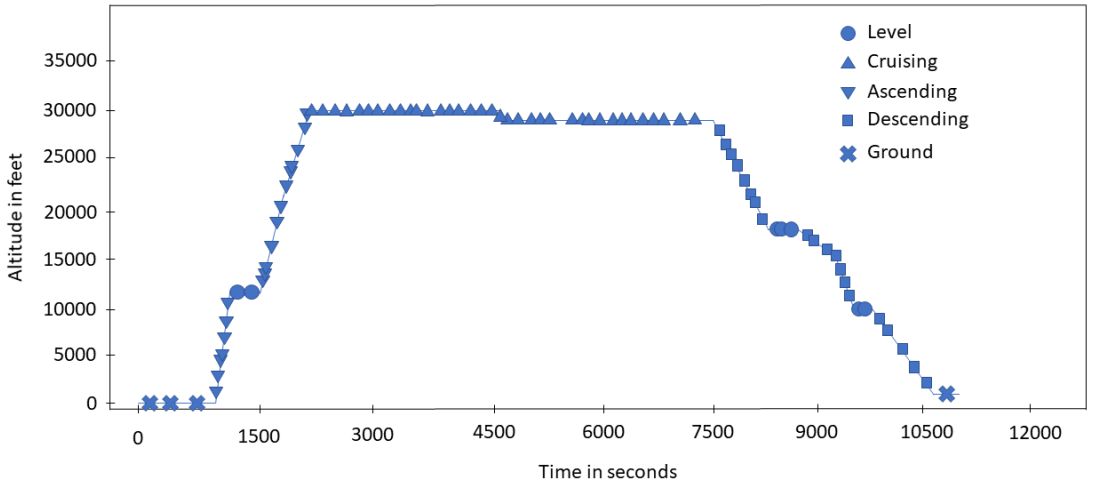


Figure 6.5: Example of phase identification

Thanks to the clustering process, we are provided with a dataset full of continuous individual flights. To identify the different phases of a given flight, one could use an unsupervised algorithm as the one mentioned before. However, the Euclidian distance between data points in the same flights are very small, so it would be rather tedious for a classical clustering method to properly discriminate a data in a CR phase from a close one in a DE phase. In addition, just like for the flight clustering, differences in-flight behaviours and aircraft types necessarily result in a difference between the training data and the testing data.

Instead, Sun et al. (2017) propose an approach based on a fuzzy logic identification method. The fuzzy sets theory (Goguen, 1973) is used to model logical reasoning with vague statements like ”the temperature is average, the air pressure is low and the air humidity is high so it must be raining”. To do so, a fuzzy set assigns a degree of membership to any elements in the dataset helping the taking of the decision. Transforming any input into a fuzzy membership function is called fuzzifying the data. In the case of ADS-B, the membership function are applied to the different features as followed:

- velocity: High, Medium and Low
- altitude: Ground, low, and high
- vertical rate: Zero, Positive, and negative

Sun et al. (2017) uses Gaussian functions as well as Z-shaped and S-shaped membership functions to fuzzify the different input data. Now with a set of rules, we can compute the fuzzy output function that is then used in the last step called defuzzification, giving the most likely flight phase the input data originated.

It is to be noted that this process of phase identification is done on time windows and not on isolated data points. The mean of the velocity, altitude and vertical rate is used for the process of fuzzification. In doing so, we limit the potential errors due to isolated outliers and we ensure a smoother phase identification.

In the Figure 6.5, we can see an example of the phase identification used on a flight.

6.4.3/ IMPROVED PHASE IDENTIFICATION

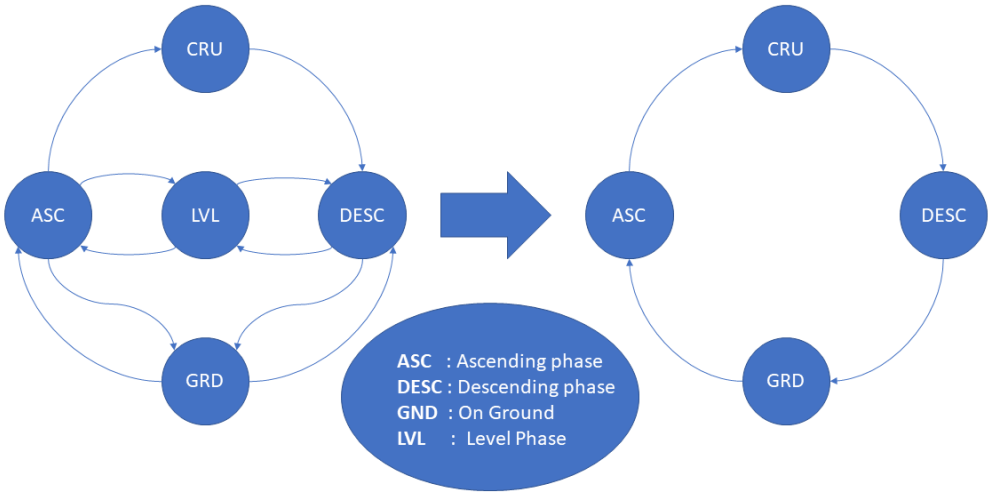


Figure 6.6: The different phases identified by Junzi Sun and the one used in our experiments

The main issue with using fuzzy logic to determine the different phases of the flights is that it is entirely determined by the data of the flight. This leads to a phenomenon where a crash scenario could be mistaken with a descent phase. As a result, the data issued from the crash would be processed by the decoder associated with the descending phase instead of the one associated with the cruising phase as seen on the left side of Figure 6.7.

To try and mitigate this phenomenon, we can take advantage of the physiognomy of a flight. In normal conditions, the ascending and descending phase is not supposed to be found outside the vicinity of the departure and arrival airport respectively. With this information, the improved phase identification adds a layer of decision on top of the fuzzification algorithm that finds the maximum Vincenty distance (over the cleaned-up training dataset) between the departure airport po-

sition and the transition between ascending and cruising phase. The same is done for the arrival airport and the descending phase. These distances are used to define the vicinity of the airports and all the training and evaluation data are processed to prevent having data labelled with the ascending or descending phase outside of these vicinities.

This approach is inexpensive in terms of calculation and helps have the right data in the right decoders. However, it has one main inconvenience: as seen in Figure 6.7, while the right plot shows that most of the crash data is labelled as cruising data, the end, which entered the vicinity of the airport, is labelled as a descent. It did not have much impact in the experiments as it concerned only a couple of data points.

Lastly, as shown in Figure 6.6, we limited the different experiments to 3 flying phases, neglecting the Level phase. This choice is motivated by the fact that this phase is very unpredictable and very dependant on the situation. However, for further experiments, exploring the results of a decoder trained with solely the Level data could be interesting as these phases are often synonyms of congested airspace, leading to a decoder specialized in these types of situations.

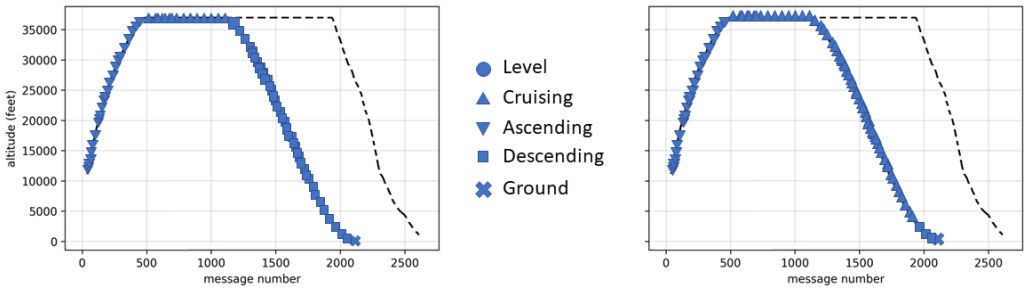


Figure 6.7: Differences in phase identification between the original algorithm (left) and our approach (right) on a crash scenario

6.5/ Dataset Creation

The last part of the chapter presents the different elements allowing the creation of a dataset fit to be used for training and evaluating the CAE, including the different

features extracted from the ADS-B data and the data format used to feed the model.

6.5.1/ FEATURES

From the data gathered through the data architecture presented in this chapter, only some features of the ADS-B messages are kept and fed to the model. This subsection aims to present them and how they are pre-treated to be used in the CAE model.

- **Altitude** in feet given by the airborne position messages.
- **Consecutive Delta** in kilometers. This is the Vincenty distance between two consecutive messages calculated from the latitudes and longitudes. This distance is bound to change from 2 main factors. The first one is the change of speed of the aircraft and the second one is the absence of messages picked up by the OpenSky Network. The third reason would be errors in decoding or from sensor malfunctions but most are filtered out from the data cleaning processing explained above.
- **Tracking Delta**. Difference between the tracking received through ADS-B and the ideal tracking calculated from the position of the aircraft and the position of the arrival airport.
- **Vertical Rate** in feet/mn. Represents the aircraft's vertical speed – the positive or negative rate of altitude change with respect to time.
- **Groundspeed** in knots. Represents the speed over the ground.

It is worth noting that some base features of ADS-B like the tracking, the latitude and the longitude are not directly used in the dataset. Concerning the tracking, the feature being a cyclic feature in degree, experiments were made using the sine and cosine component to avoid the discontinuity implied by having a heading

varying between 0 and 360 when 0 and 360 are de facto the same angle. Unfortunately, having two features for the heading instead of one doubled its impact on the model and created some unbalance hence the choice for the heading delta feature presented earlier.

In a similar fashion, the latitude and longitude also being cyclical data were turned into the consecutive delta feature. Another reason for this change is the will to make the model area-agnostic which would have been impossible with the coordinates as is as features. This improves the accuracy of the models on data they have not seen during their training, as shown by Fried and Last (2021).

6.5.2/ Data Windowing

As presented in chapter 5, the input of the CAE is in the form of time windows. As shown in Figure 6.8, creating windows of data using the slicing window technique adds an extra set of hyperparameters to choose to train models: the stride and the size of the time-windows. The values of these parameters are further discussed in Section 7.2.

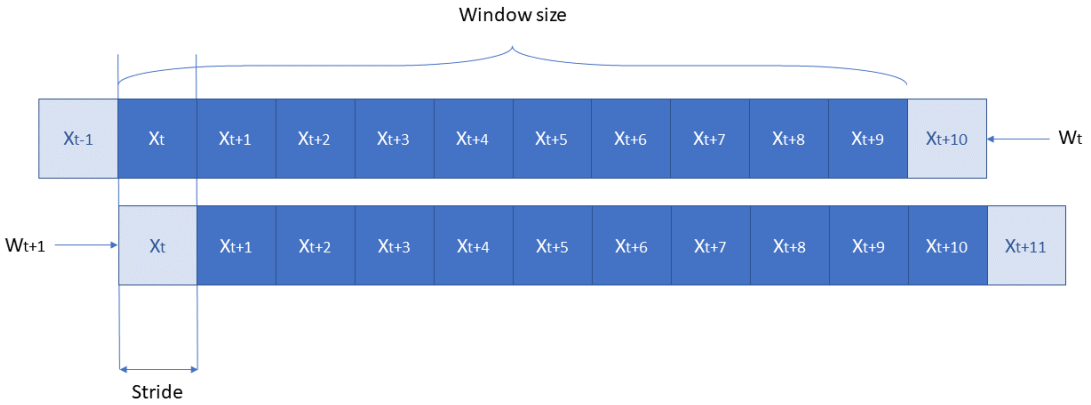


Figure 6.8: Example of time windowing with a window size of 10 and a stride of 1.

Another question raised by the windowing of the data at hand is the phase linked to each window. Due to the categorical nature of the phase label, it is not possible to just take the average of a window to labels it. Instead, the choice was made to

take the most present flight phase in a given window as the flight of said window.

6.5.3/ TFRecord Format

Now that the whole data process has been described, the only detail left to explain is how the model is fed with this data. The ML framework used during this thesis is the Python framework Tensorflow and this framework allows the use of a serialized format called TFRecord. This format is a simple format for storing a sequence of binary records. It uses the protocol buffers [4] to serialise the data and work in a similar fashion to a markup language. Using this TFRecord has two main advantages:

- The main advantage is that it keeps the hierarchy of the data. This is important when dealing with ADS-B time-series to know which data belong to which flight and which time frame. This avoids the costly sorting that would be needed before the training.

- This format allows the interleaving of different TFRecord dataset files to avoid a bottleneck linked to input/output limitations on a single file thanks to parallelisation. Coupled to a model training using a GPU, the CPU can solely focus on the reading of the file and the transformation of the dataset into batches for the GPU to train on, leading to efficient training.

The main issue with this format is the extended size of the dataset when in TFRecord formats compared to other compressed formats. This can be a problem when using a hefty dataset but this was not an issue for the experiments of the thesis.

[4] https://developers.google.com/protocol-buffers/

6.6/ SUMMARY

The data pipeline is a necessary part of any ML model. The different choices explained in this chapter were mostly taken during the span of the thesis while discovering the topic and might not be optimal in terms of the state of the art data acquisition and treatment but works well for the experimentation of the CAE.

This pipeline is the second main contribution of this thesis. While tools already existed to gather and process ADS-B data, no proper end-to-end pipeline existed for the creation of datasets for anomaly detection experiments. The objective of the work presented in this chapter was to create said pipeline by adding different missing parts:

- Different format conversions enabling the use of raw data. It also allowed for a direct link between ADS-B sources like Opensky and our FDIA testing framework FDI-T. This led to a software contribution to the traffic library.
- A method based on the calculation of Vincenty distances between related ADS-B messages enabling the detection and the filtering of outlier messages. While this outlier elimination method was not necessary with the processed data of Opensky, it ensures cleaner data on raw stream coming directly from ADS-B antennas.
- The phase of the flight being the context used for the CAE experiments, this is primordial to have a phase identification algorithm as accurate as possible. The improved phase identification presented in this chapter uses the fuzzy logic introduced by Sun et al. (2017) and takes into account the distance from the departure and arrival airport to improve its accuracy in case of an FDIA attack.

This concludes the second part of this dissertation presenting the new type of auto-encoder model (CAE) for the detection of ADS-B anomalies, as well as all the software processing and developed components for data acquisition and prepa-

ration for machine learning. The next and last part presents the experimentation done using the CAE and the presented data pipeline.

III

EXPERIMENTS

7

Experiments on Air Traffic Data

This third and last part of this dissertation is dedicated to the different experiments done using the CAE model. In chapter 7, the evaluation of the CAE on ADS-B data is presented along with other models. Then, in chapter 8, improvements of the CAE model are presented by modifying how the decoders are specialised. This new implementation of the model is then showcased using Automatic Surveillance System (AIS) data from the maritime domain.

In this chapter, the evaluation of the CAE model is presented in the air traffic domain. Using the data pipeline presented previously, we constituted a training dataset to train a CAE with 3 decoders, one for each phase. Then, we created an evaluation dataset using real-life anomalies, as well as crafted ones using attack scenarios in FDI-T, to compare our model to a baseline constituted of other unsupervised ML models.

7.1/ Data Specifications

Before presenting the different evaluations made with the CAE in the rest of the chapter, this section introduces the different data extracted using the data pipeline presented in chapter 6 to train and evaluate the different models.

7.1.1/ Training Data Distribution

Despite the good coverage of the Opensky Network as well as the outliers management present in the data pipeline, using international flights as training data for ML models often lead to unstable data due to different factors:

1. International aircraft often fly over maritime zones where the ADS-B coverage is poor if not nonexistent. This lead to cut flights, often for a long period of time which have a tendency to disturb the flight clustering process which then considers the separated parts of one flight as different unique flights.

2. Some zones of the world, even though they are covered by sensors have higher error rates than others. This can be due to an error in the set-up of the receiver, a faulty antenna or simply due to geographical constraints like mountain ranges. The latter problem is usually solved using sensor redundancy but this is not always possible.

3. Depending on the region, the regulations for the commercial flight callsigns are different. This creates inconsistencies when trying to check on flights for a given air company or to track results afterwards. This does not have any impact on the training of the model but simplifies the analysis.

For these different reasons, the training dataset is exclusively focused on internal European flights. This choice is mainly motivated by the excellent land coverage of the Opensky Network in this area, thus avoiding factor 2. Factor 1 is not an issue as there are no big ocean in this area and the regulation of the callsign is well followed by the countries of the European Union, making the last factor irrelevant.

It is to be noted that, as explained in chapter 6, the data pipeline could directly process data from any ATC system issuing ADS-B data. In this case, the different presented factors could be limited or nonexistent and the CAE could be trained using data from these sources seamlessly.

The different flight routes used for the training can be visualized in Table 7.1. It compiles together 15 flight routes for a total of 1008 flights. The dataset is composed of both long and short flights, as well as flights travelling in different directions to ensure data diversity.

Departure airport	Arrival airport	Number of flights	Duration (hours)
Athens (LGAV)	London (EGGW)	56	3.6
Berlin (EDDB)	Kiev (UKBB)	33	1.6
Budapest (LHBP)	Dublin (EIDW)	43	2.8
Frankfurt (EDDF)	Lisbon (LPPT)	68	2.5
Hamburg (EDDH)	Barcelona (LEBL)	29	2.0
Kiev (UKBB)	Paris (LFPG)	83	3.3
London (EGGW)	Milan (LIMC)	46	1.6
Madrid (LEMD)	Moscow (UUEE)	59	4.2
Malaga (LEMG)	Frankfurt (EDDF)	81	2.9
Manchester (EGCC)	Istambul (LTFJ)	75	3.8
Munich (EDDM)	Lisbon (LPPT)	68	3.3
Paris (LFPG)	Oslo (ENGM)	34	1.9
Stockholm (ESSA)	Barcelona (LEBL)	25	3.2
Vienna (LOWW)	Copenhagen (EKCH)	83	1.3
Zurich (LSZH)	London (EGLL)	225	1.2
Hamburg (EDHI)	Hawarden (EGNR)	45	1.5
London (EGLL)	Moscow (UUEE)	184	4.0

Table 7.1: Flights used for the training of the different models presented. 15 flight routes data taken from September to December 2020 for training and 2 flight routes taken in January 2021 for validation.

7.1.2/ Evaluation Data

The evaluation dataset is composed of two different subsets: an FDIA set and a baseline set. The FDIA set is composed of data modified using FDI-T. These data simulates emissions that could be resulting from an FDIA including different Trajectory modification scenarios an attacker could create. All the anomalies contained in this set are applied on all the flights found in the training set but from January 2021 only.

On the other hand, the baseline dataset includes real-life data with no modifica-

tion done by FDI-T. It includes normal data from other parts of the world to see if there is any bias to train a model on a single region of the world and also includes a recent anomaly caught through ADS-B. This anomaly is not an FDIA, so it technically does not belong in the scope of this research, however, its presence in the dataset enables us to monitor the tool bias eventually induced by FDI-T.

Here is a detailed list of what the evaluation database is composted of :

① **FDIAs**

Gradual Drift (DRIFT) – The gradual drift is an attack that consists of simulating an altitude drift or a velocity drift, either positive or negative. The altitude or velocity messages on the attacked time windows are all raised/lowered by an increasing/decreasing multiple of an arbitrarily chosen n feet. So, if the first message is lowered by 2 feet compared to the original message, the second is then lowered by 4, etc. The Figure 7.1 shows a velocity drift used during the evaluation.

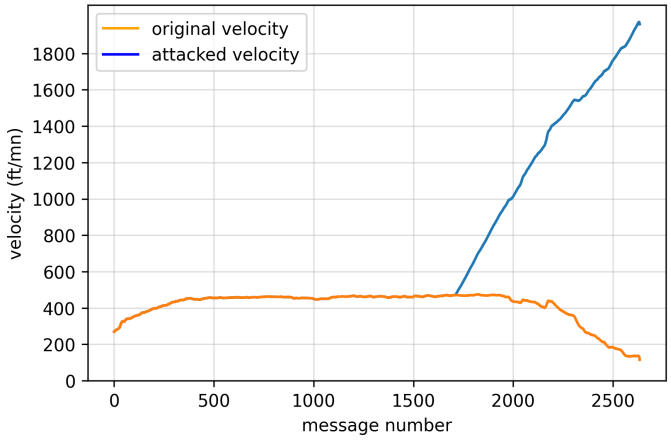


Figure 7.1: Velocity drift attack

Made-up Crashes (CRASH) – Using FDI-T, crash scenarios can be created combining an altitude drift, a negative vertical rate, and a reduction of groundspeed. This reduction of groundspeed is not to be mistaken with a reduction of airspeed. Indeed, a stalling aircraft has a groundspeed plummeting while its airspeed increases. The signal is then stopped once the aircraft reaches the ground (for these experiments, ground level is set at 300 feet above see level). Figure 7.2 shows some of the features modified during a CRASH attack.

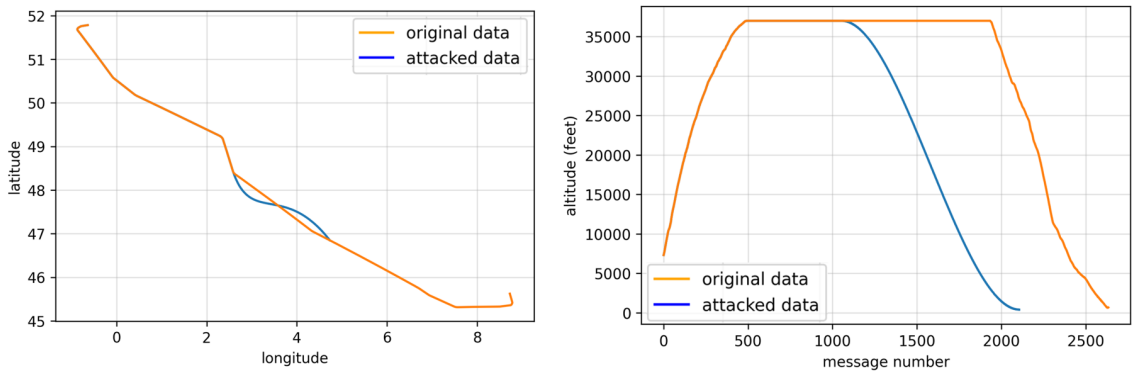


Figure 7.2: Crash attack. Latitude/longitude on the left and the altitude on the right. Other features like vertical speed or track are also modified realistically.

Constant position offset (OFFSET) – This attack, when toggled takes the real data of a flight and adds an offset of 1 in both the latitude and the longitude (see Figure 7.3). This offset represents a distance of around 132 kilometres between the original and the anomalous trajectory.

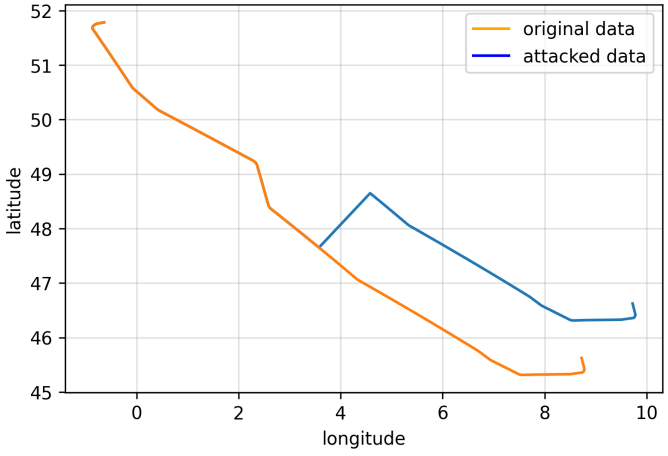


Figure 7.3: Constant position offset attack

② Baseline

World Data (WORLD) – As the training dataset is exclusively composed of data from European flights, including regular data from other parts of the world in the testing dataset allows for checking the genericity of the approach. This part of the dataset does not include any anomaly and as such, any alarm raised by models on these data will be false positives. It includes flights from the European

airspace, American airspace – e.g. Dallas to Louisville – or Australian airspace – e.g. Canberra to Perth –.

Ryanair Hijack (HJK) – Constituted of the Ryanair flight 4978 from Athens to Vilnius which was forcibly diverted to Minsk after entering the Belarus airspace on the 23rd of May 2021[1]. It is to be noted that the emergency was turned on by the crew 2 minutes after the flight started to change its course as seen on Figure 7.4. For the evaluation, the labels have been set to 1 from the beginning of the emergency till the landing. This is not an FDIA as it is actually a real-life abnormal scenario but detecting this anomaly is important to prove that the model is not biased into detecting only anomalies created by FDI-T.

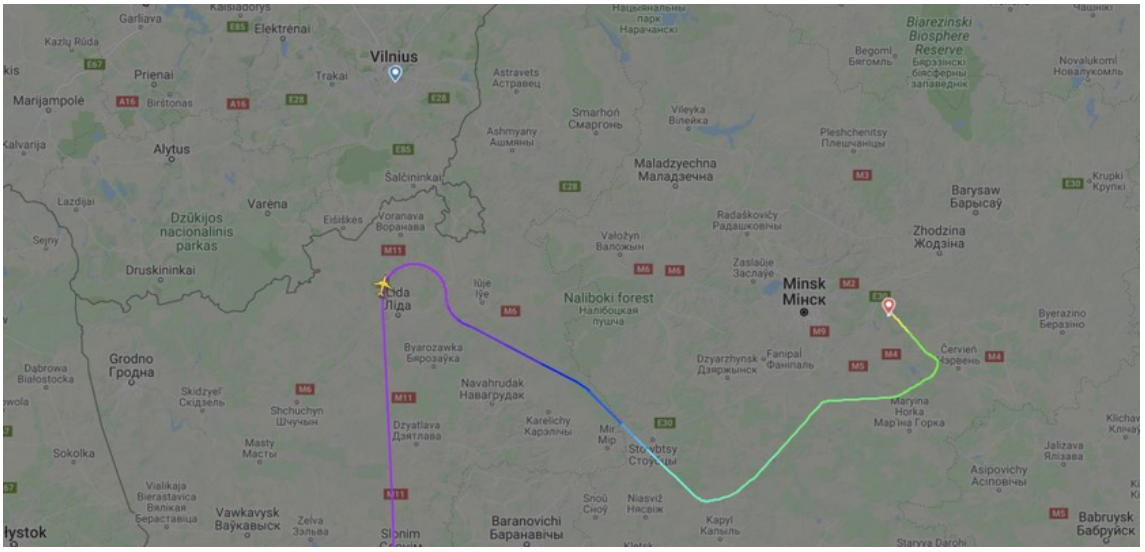


Figure 7.4: Flight hijacking initially going to Vilnius, landing in Minsk

7.2/ TRAINING SPECIFICATIONS

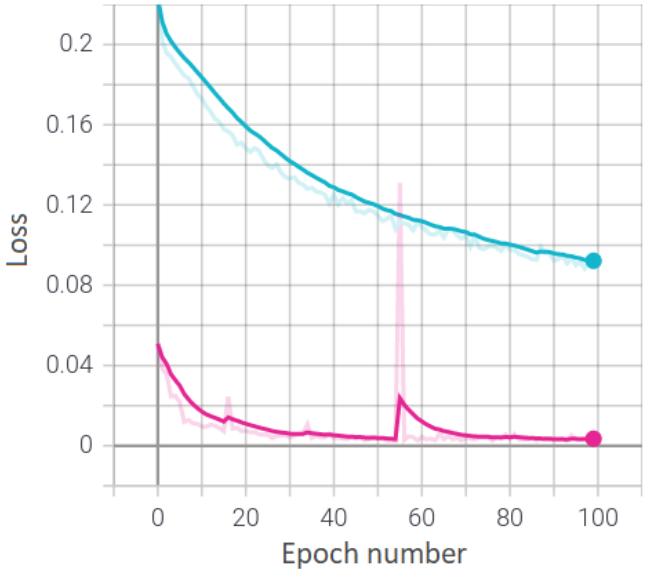
Once the training and evaluation datasets are constructed, the next step is to prepare an environment to train the model, as well as choosing the different fitting hyperparameters. This section presents all the leftover aspects of preparation to evaluate the CAE model including the training environment, the hyperparameters,

[1] https://www.flightradar24.com/blog/ryanair-flight-4978-to-vilnius-forcibly-diverted-to-minsk/

the different evaluation metrics and finally the models trained alongside the CAE to serve as baselines.

7.2.1/ TRAINING ENVIRONMENT

The training environment is defined by the hardware and the software used during the training. Concerning the software, we used Tensorflow as we already showed in Chapter 6. The training dataset we described in the previous section represents 336 Mb of data separated into 15 tfrecord files. The interleaving of the different tfrecord files allowed the training of our models without using more than 6 Gb of memory. For the hardware part, we used the supercalculator of the Mésocentre de Calcul de Franche-Comté[2]. We trained on a calculation node hosting a Tesla V100 performing at 7.8 TeraFLOPS. The training for the CAE on this node on average was taking around 26 minutes per epoch for the CAE. In Figure 7.5, the training loss and the validation loss over the first 100 epochs of training is shown. One can notice a difference in loss between the training set and the validation set. It is due to a few leftover outliers in the training set which make the average training loss much higher than its validation counterpart.



[2]http://meso.univ-fcomte.fr/

Figure 7.5: The training loss per epoch. In blue is the training loss, in pink the validation loss

7.2.2/ CAE'S HYPERPARAMETERS

To train any Neural Network model, one needs to find different values of parameters to optimize the training of the said model: its learning rate, its number of layers, units, different kinds of parameters for the optimizers etc. Each of these parameters have an impact on how well the model trains and a poor choice often leads to sub-optimal results. These parameters, also called hyperparameters can be chosen by different means:

- **Grid Search**. Using common knowledge on the model in training and insights we can have on the data, it is sometimes possible to narrow down the number of possible "best" hyperparameters to approach the best model possible. If from this preliminary examination, the combinatorial explosion of searching through all the possible parameters has been limited, then it is possible to give a dictionary of all the possible values that every parameter can take to an algorithm that trains the model for a limited number of epochs and choose the combination of parameters that yields the best loss out of all the possibilities.

- **Random Search**. Sometimes the number of parameters is too important for a grid search and finding the optimal set of parameters would take way too much time. Instead, we can use a Monte-Carlo method that consists of transforming every parameter into uniform distributions and each iteration randomly samples from each of these distributions to have a set of parameters used to train the model. This method does not necessarily yield the optimal solution but often returns a relatively good one in a much quicker fashion than a grid search.

- **Other methods**. Finding the best set of hyperparameters falls under the domain of optimisation problems and as such, has a wide range of algorithms

that exists in the literature to treat it (Bergstra et al., 2011). For instance, a known one used to find an optimized set of hyperparameters is called the Bayesian optimiser (Snoek et al., 2012). The goal of the Bayesian optimisation is to minimize a function $f(x)$ bounded to a set of parameters $\mathcal{X}$ (e.g., here, the function would be the loss of the model and the set of parameters, the hyperparameters). To do so, it constructs a probability model of $f(x)$ that gradually improves after some evaluation of $f(x)$ on randomly chosen points. Iterations after iterations, the algorithm then exploits this model to choose values of $\mathcal{X}$ to next evaluate $f(x)$, while being non-deterministic. The main advantage of this approach compared to Grid Search or Random Search is that every iteration uses the results of the previous one to try and approach the optimal solution. This kind of optimiser has been trialled for the training of the CAE but unfortunately, the model was not big enough to justify the extra calculation a Bayesian optimisation adds.

There is also an *ad-hoc Random Search*, which would describe many of the first experiments done on a model. It is a regular random search, but instead of having an algorithm picking randomly the values, the parameters are chosen at the discretion of the researcher, based on its empirical knowledge and previous experience.

The hyperparameters for the CAE were first found using a grid search using previous results found on regular auto-encoders. They then were refined through trial and errors. For the final iteration of the CAE of which the results can be found in the following section, windows of 30 messages have been used with a stride of 1. The batch size is 256. The number of units in the encoder's BiLSTM is 32 which is then flattened to feed a Dense layer reducing the dimension to 10, the chosen latent space size. The different decoders each embed a single LSTM layer with 32 units.

7.2.3/ METRICS

For the evaluation, having metrics that enable the comparison of performance between different models is necessary. Just like hyperparameters, the choice of metrics is important and often depends on the task evaluated. For instance, in a reconstruction task like data compression or denoising, the loss of the model, representing the distance between the original input and its reconstructed counterpart, can be a first approximation of how well a model compares to another. However, for an anomaly detection task, this loss is not enough and must be coupled to a threshold to determine if a given input is an anomaly or not.

One metric often used in ML is accuracy. The accuracy is the proportion of correct classifications. The accuracy is originally a classification metric but we can actually consider anomaly detection as a 2-class classification problem. However, when the different classes are unbalanced, which is often the case in anomaly detection problems, the accuracy yields misleading results.

For this reason, predictive analytics give other metrics to properly evaluate the quality of an anomaly detection model. These metrics are based on the different values found in a confusion matrix as seen in Figure 7.6: the true positives (TP), the true negatives (TN), the false positive (FP), and the false negatives (FN).

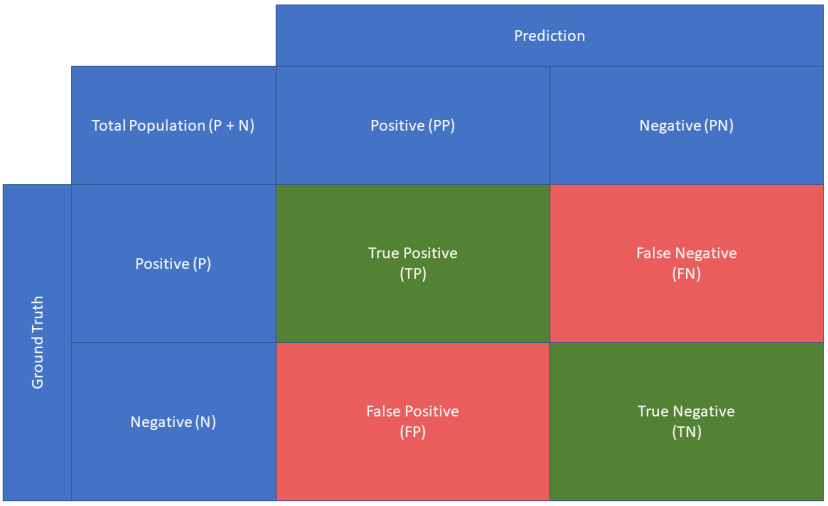


Figure 7.6: The confusion matrix from where the different metrics stems from.

The different metrics used during this evaluation are:

- **Recall**. Also called sensitivity, the recall represents the proportion of anomalies detected. Also referred to as the true positive rate, it tells if the model was sensible enough to detect anomalies that were actual anomalies. It is defined as:

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7.1}$$

- **False Positive Rate (FPR)**. The FPR, also called the false alarm rate, is the proportion of false-positive predictions over the total negatives. In some domains like the air surveillance, the FPR is almost as important as the accuracy for having too many false alarms can be more disturbing than the real FDIAs for a controller. The FPR is calculated as:

$$FPR = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{7.2}$$

- **F1-Score**. Before defining the F1-Score, we need to define the precision. The precision is the proportion of identified anomalies being true anomalies. The F1 score or F-Score uses the recall and the precision to determine a score of how efficient a model is at detecting anomalies. However, just like the accuracy, this score can be subjected to bias and would not be as useful alone. It is defined as:

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{7.3}$$

- **Accuracy**. The accuracy, as already defined is the proportion of correctly predicted observation. It is defined as:

$$Acc = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{7.4}$$

For the evaluation made during this thesis, these presented metrics were good approximations of how well the different models were performing. For future ex-

periments, the use of the Phi coefficient or of the ROC curve to better tweak the different thresholds (Powers, 2020) could be considered.

7.2.4/ BASELINE MODELS PRESENTATION

To show the overall performance of the CAE, it is compared with 3 other unsupervised approaches for anomaly detection of false data injected in ADS-B time-series: a regular Isolation Forest (Liu et al., 2008) model, an LSTM-auto-encoder (Habler and Shabtai, 2018), and a VAE-SVDD (Luo et al., 2021). Before diving into the different results obtained during this evaluation, here is an introduction to these different baseline models.

LSTM-AE is a sequence to sequence model based on an encoder-decoder reconstruction used by Habler and Shabtai (2018) to detect false data in ADS-B time-series. Both encoder and decoders are composed of LSTM units that take care of the time-dependency of the data. This model has shown decent results on coarse anomalies and has the advantage to be rather simple to implement and fast to train.

The different hyperparameters were chosen using previous experiments on this model and are similar to the ones used for the CAE. The threshold was calculated based on the empirical rule too.

VAE-SVDD (Luo et al., 2021) is a variational auto-encoder (VAE) coupled with a support vector data description model (SVDD) to automatically determine its threshold. A VAE is a deep Bayesian model which represents an input $\mathbf{x_t}$ to a latent representation $\mathbf{z_t}$ with a reduced dimension, and then reconstructs $\mathbf{x_t}$ by $\mathbf{z_t}$. The main difference with a regular auto-encoder is that the latent variable $\mathbf{z_t}$ is sampled from a probability distribution, such as a Gaussian distribution with the mean and the standard variation being outputs of the encoder network.

For the VAE-SVDD, the method to choose the anomaly thresholds is already given in the paper and the different hyperparameters are chosen according to the author's remarks.

Isolation Forest (Liu et al., 2008) is an anomaly detection algorithm using an ensemble of isolation trees to differentiate normal data from anomalies. It has the advantage of being fast, light-weight and can be quickly implemented. It is however not well equipped to tackle the time dependency of the data.

7.3/ RESULTS

This section details the experimental results made to determine the pros and cons of the different approaches and which one performed the best overall on this dataset. All the implementations are accessible on the Scifly[3] Github repository. This is made as an effort to improve the replicability of the presented evaluation as well as propose a non-exhaustive, upgradable baseline dataset for future models in the growing field of anomaly detection in ADS-B data.

Table 7.2 shows the accuracy, the recall, the FPR and the F1 score on the different datasets for each model. Overall, these experimental results demonstrate the superiority of the CAE compared with the state-of-the-art approaches in FDIA detection in ADS-B. Indeed the F1 score on the Total Dataset is more than 20% over the second-best performing model (not considering the IForest for the reasons explained below). It is to be noted that the WORLD dataset does not have any true positives nor false negatives which automatically set the Recall to Nan (division by zero) and the F1 to 0. Next, we analyze the performance of the different methods in detail.

7.3.1/ BASELINE RESULTS

- **LSTM-AE:** This simple deterministic model well manages to capture the ADS-B normal behaviour in its latent space showing very low FPR using a 3-sigma threshold as well as decent results on the Velocity Drift dataset.

[3] https://github.com/Wirden/scifly

Evaluation Dataset	Metrics	LSTM-AE	IForest	VAE-SVDD	CAE
World Data	Accuracy	0.994	0.687	0.899	0.989
	Recall	NaN	NaN	NaN	NaN
	FPR	0.006	0.313	0.101	0.011
	F1 score	0	0	0	0
Ryanair Hijack	Accuracy	0.946	0.890	0.722	0.847
	Recall	0.637	1	0.227	0.301
	FPR	0.001	0.129	0.231	0.017
	F1 score	**0.778**	0.729	0.152	**0.439**
Velocity drift	Accuracy	0.933	0.944	0.949	0.961
	Recall	0.809	0.957	0.930	0.912
	FPR	0.001	0.063	0.043	0.012
	F1 score	0.886	**0.937**	0.926	**0.939**
Constant position offset	Accuracy	0.519	0.709	0.541	0.526
	Recall	0.033	0.491	0.077	0.053
	FPR	0.001	0.073	0.046	0.004
	F1 score	0.060	**0.598**	0.107	**0.097**
Made-up Crash	Accuracy	0.506	0.919	0.710	0.962
	Recall	0.003	0.922	0.426	0.929
	FPR	0.001	0.084	0.037	0.004
	F1 score	0.005	**0.925**	0.573	**0.955**
Total	Accuracy	0.780	0.830	0.764	0.857
	Recall	0.371	0.843	0.415	0.549
	FPR	0.002	0.132	0.092	0.010
	F1 score	0.544	**0.797**	0.440	**0.738**

Table 7.2: Comparison of the different models evaluated

Its very low F1 score on the Made-up Crash dataset can be explained by the data resembling a regular descent trajectory which leads the decoder to reconstruct the data as-is. Lowering the threshold to a 2-sigma helps raise the F1 score but results in an FPR being too high for anomaly detection.

- **VAE-SVDD:** The stochastic nature of the VAE could explain the better results compared to the regular LSTM-AE on the Made-up Crash and Velocity drift dataset. Luo et al. (2021) combine LSTM and VAE by replacing the feed-forward network in a VAE to GRU but do not include information from $\mathbf{z_t - 1}$ into $\mathbf{z_t}$ in the likes of Su et al. (2019). That might explain the issues the VAE-SVDD has to properly represent the distributions of the input data, leading to high FPR compared to the other methods. All in all,

the VAE-SVDD, while performing well on coarse anomalies like the velocity drift and to some extent the Made-up Crash thanks to its stochasticity, fails to reconstruct properly ADS-B data leading to high FPR on new data and mediocre results overall. This could be explained by the limitation of having a Gaussian qnet being too trivial to properly reconstruct ADS-B information coming from other parts of the world, negating the advantages of having such an architecture.

- **IForest:** The model yields good results when compared to the other models, which is explained by the evaluation dataset being based on the same flights as the training data but one month later. The IForest manages to flag anomalies on flights it has already seen or in the vicinity of these said flights – for instance, the Ryanair Hijack – without any trade-off except its FPR. Indeed, the FPR is on average almost ten times higher than the CAE's which makes it hard to use as a reliable anomaly detector. It would even be completely pointless on flights in part of the world it did not see during its training. On the contrary, training it on the data from different parts of the world would add too much complexity for good results.

7.3.2/ CAE RESULTS AGAINST THE BASELINE

Compared to the LSTM-AE with a single decoder, the CAE, thanks to its specialized decoders, manages to discriminate anomalous situations like crashes from regular descent operations while keeping a very similar low FPR overall. However, from the metrics alone, the CAE seems to be under-performing compared to the LSTM-AE for the hijack scenario. This can be explained by looking at Figure 7.7 which compares the anomaly score over the message windows for both models. One can observe that for the CAE, the anomaly is set off before the actual emergency due to its delay with the diversion of the flight. It explains the FPR being much higher than the other models and displays the reactivity of the CAE in

such circumstances. On the other hand, the low recall is due to the score going back to a normal value after some time which means the CAE does not label the end of the flight as abnormal from its ADS-B data.
Compared to the VAE-SVDD, the CAE performs better on all datasets except on the constant position offset where all models perform poorly due to the scenario's very nature. Indeed, the small offset added to the latitude and longitude is not enough to trigger alarms leading to extremely low F1 scores. This attack can only be detected by the LSTM based models when the values actually change. Finally, the IForest model, despite being cost-effective and accurate on the few flights it has seen during its training, is not as dependable as the LSTM-AE or the CAE due to its high FPR, limiting its usage in real-life applications. In conclusion, while the CAE does not well perform on the position offset FDIA, it has the best accuracy on normal data and the best scores on other FDIAs. It is also the only model that manages to detect the real-life anomalous scenario as soon as it started, leading to an alarm raised before the change of flight status.

This comparison of the CAE model against other ML models for anomaly detection allows us to answer the second research question that stems from the research objective of this thesis: **RQ2: Is using the context of a data helpful in detecting anomalies?** The different experiments show an overall improvement compared to other AE approaches. Indeed, in scenarios where the context is primordial like the crash scenario, the CAE is the only AE model with both a low FPR and a high F1 score. The other AE models which trained on all data mistook the beginning of the crash for a descent phase. Similarly, the Ryanair anomaly is detected the quickest by the CAE, as a sudden change in heading during a cruising phase is unusual.

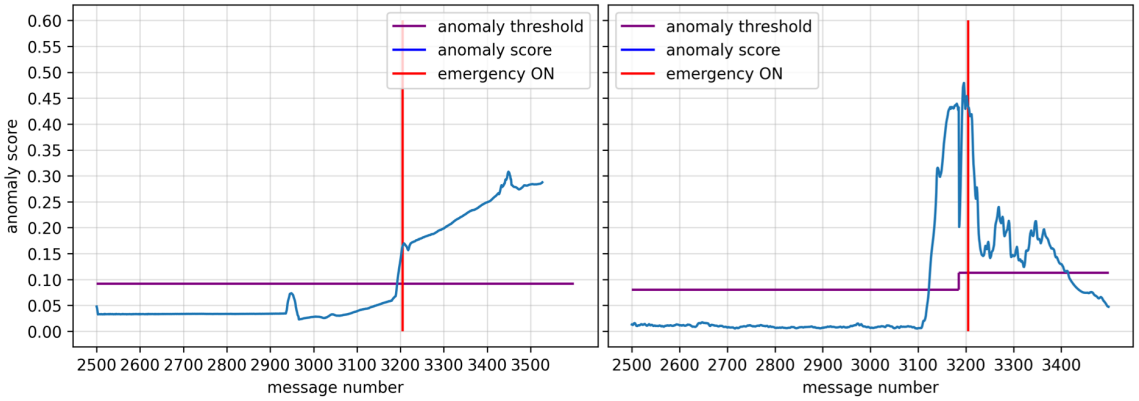


Figure 7.7: Anomaly score for the LSTM-AE on the left and the CAE on the right for the Ryanair hijack flight.

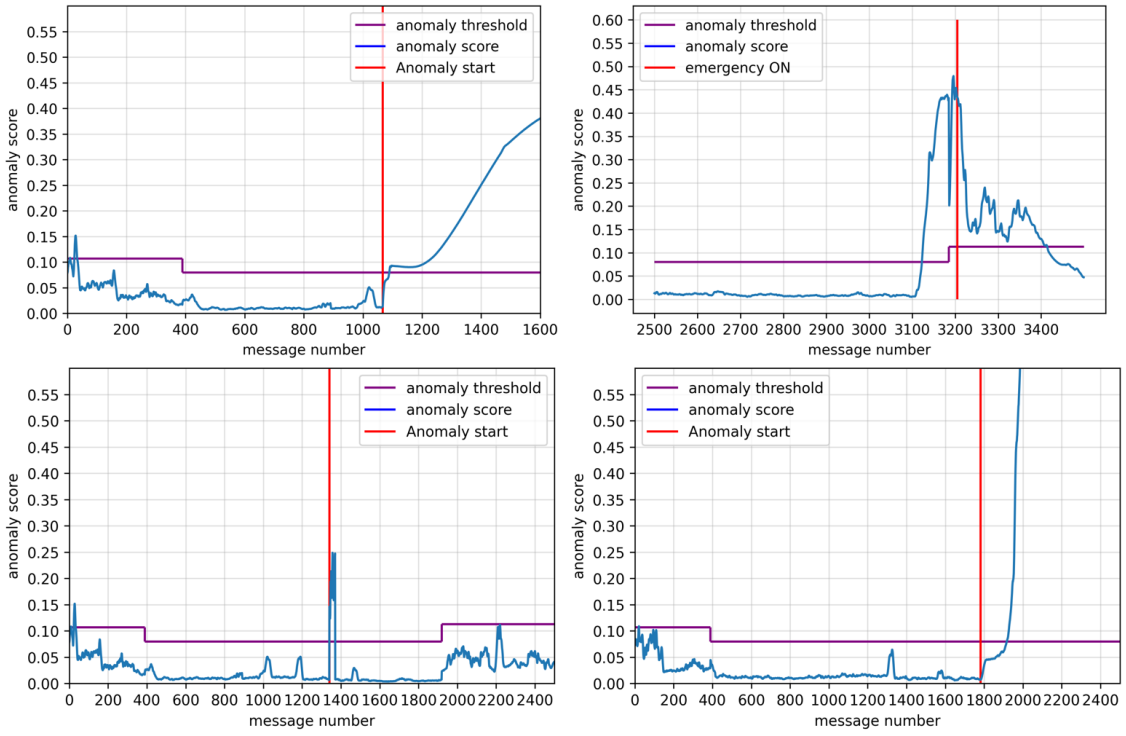


Figure 7.8: CAE anomaly scores for a flight taken randomly from the different evaluation datasets. The top-left figure is from the CRASH dataset, the top-right is from the Ryanair hijack, the bottom-left is from the constant position offset and the bottom-right is from the velocity drift dataset

7.4/ Discussion

The CAE model shows good results on the chosen evaluation dataset compared to other ADS-B anomaly detection models. This section first covers a discussion on the presented results with regards to the **RQ1** introduced in Chapter 1. Then different caveats and limitations are presented.

7.4.1/ Usability of unsupervised ML models for detecting FDIAs

The first assumption made for the usability of ML models to detect anomalies is the authenticity of the data used during the training. If data sources like sensors or the Opensky-Network were to be attacked, the models trained from these corrupted sources would not be able to detect ADS-B anomalies properly. For instance, if a corrupted source of ADS-B set the cruising altitude to 45000 of the commercial flights instead of the regular altitude, a model trained on these data would raise many false alarm on real non-corrupted data. To avoid that, implementing safeguards ensuring the data follow the ATC standards are advisable.

Once we made sure this was not the case thanks to safeguards ensuring , the experiments give us enough insights to answer the research question **RQ1: To what extent unsupervised ML algorithms can detect False Data Injection Attacks in ADS-B data?** The answer to this question can be decomposed into two parts:

1. Results on different attack scenarios showed that unsupervised ML models, especially the CAE with its context awareness, were fit to detect FDIAs in ADS-B messages. However, an ML model trained for this task is not able to make the difference between a real anomalous situation and an FDIA. Indeed, as seen in the results, the Ryanair hijack was flagged regardless of it being an FDIA or not. While this proves that models are not detecting anomalies because of any tool bias induced by FDI-T, it also shows that

FDIAs and real-life anomalies share the same property of being extraordinary situations, not seen by ML models during their training. For this reason, unsupervised models are suitable for anomaly detection in the ATC domain but, as of now, cannot differentiate between anomaly subclasses, aka real-life scenarios vs FDIAs.

2. While conclusive on most of the scenarios presented, some FDIAs can be difficult for ML models to be detected, in the likes of the Offset scenario. Due to their nature, flight trajectories, while being overall linear over the same routes, can have inconsistencies, mainly due to fluctuating weather or congestion problems. This results in ADS-B time-series to have a tolerance margin when used to train ML models. This means that all attacks made within this margin likely end up not being detected if the attacker carefully conforms to the ADS-B protocol and to the flight plan. The question of the severity of such attacks is however out of the scope of this dissertation.

7.4.2/ CAVEATS, LIMITATIONS AND CONSIDERATIONS

Through these experiments, we answered to the two first research questions of this thesis. Here are some additional reflections that can be added:

- The CAE in its current state does not support online learning and therefore cannot be updated to the latest ADS-B data. However, all the data used in the evaluation dataset are from 2021 while the training data were from 2020 showing no significant differences between them. This result only has two explanations: either the data does not change significantly enough over time to make a difference or the model is robust enough not to be disturbed by small changes. Only future data can give proper insight to answer this. In addition, the low FPR on the world dataset shows that the model is area-agnostic thanks to the features and the data-processing used for the data. This avoids the training of different models for specific regions.

- However, it is also the model where the anomaly disappear once the main change in the track is over. This can be explained by the switch to the DESCENT decoder which is less sensitive to changes in track due to the flight activity when approaching the arrival airport including congestion management and level flight. These results could be improved by adding other decoders taking level phase data or congestion management data.

- One of the downsides of the creation of realistic scenarios through a framework like FDI-T is the introduction of a tool bias that could lead to the detection of anomalies being eased. While this would question a supervised approach being trained using said data, for the unsupervised approach, it only shows that models are able to detect these abnormal scenarios. If coupled with a few real-life examples of anomaly situations, it only constitutes content to prove the robustness of the models.

- Having an FPR higher than zero can be a problem in air traffic management as it would trigger unnecessary measures to take care of false alarms. Unfortunately, it is not an easy task to create a model sensitive enough to detect all kinds of attacks without ever having false positives. In Figure 7.8, the false positives observed barely exceed the threshold while the anomaly scores like the one on the Ryanair hijack almost reach five times the threshold value. Adding other soft thresholds – e.g. four or five sigma rule – to determine the gravity of the attack could help discard the false alarms in most cases and on the other hand could raise emergencies if the anomaly score would go too high, disregarding entirely the rest of the flight. This strategy would help in the case of anomaly spikes like in the constant position offset, which would otherwise go undetected.

7.5/ SUMMARY

In this chapter, we presented the different evaluations made using different models found in the literature to detect anomalies in ADS-B messages. We also presented the results of our own approach to compare it against the baseline. The contributions of this chapter can be summarised as followed:

- The creation of a training dataset using the pipeline of Chapter 6 composed of different flight data from across Europe.
- The creation of a testing dataset also using the pipeline of Chapter 6 composed of different FDIA anomalies created using FDI-T as well as real-life anomalies.
- The implementation of different baseline models found in the literature compiled in our scifly repository[4]. It also include the implementation of our own anomaly detection model, the CAE.

Using these contributions, results showed that FDIA can indeed be detected by ML models. Not only that, but adding contextual awareness has proven useful at detecting more realistic anomalies. Even though the CAE, like the other ML models, manage to detect anomalies, it failed at properly discriminating real anomalies from crafted ones through ADS-B data alone. While this limits the possibility of using such models as FDIAs detectors only, this does not prevent these models from being used as anomaly detection methods, including FDIAs.

The next and last chapter of this dissertation presents early experiments using the CAE to determine whether it can be extended to other transport domains or not.

[4] https://github.com/Wirden/scifly

8

Validation of the Approach Using Maritime Data

To evaluate the genericity of our approach as well as a possible extension to the CAE model, additional experiments were done by the end of this thesis. The maritime domain was chosen because we had opportunities to have access to data thanks to a partnership as well as its similarities to the air traffic domain.

This last chapter gives some insights into the maritime domain and the use of a similar protocol to ADS-B. It will highlight some specifics of the domain that led the experiments to an extension of the CAE model. Finally, early experiments[1] done with Pierre Bernabé, a Ph.D. student from our research team at UBFC, and working at Simula Research Laboratory, are presented.

8.1/ Maritime traffic surveillance

Maritime traffic surveillance is a broad term that includes a numerous variety of missions. The origin of this diversity is directly linked to the diversity of ships' activities at sea and to the freedom offered by international water status. Of course, said activities are directly linked to the geographical area they evolve in. For in-

[1]This work was supported by the Norwegian Research Council (RCN) TSAR project under contract 287893 and benefited from the experimental infrastructure to explore exascale calculus (eX3), which RCN funds under contract 270053. Satellite AIS data used for model development and testing has been made available courteously by its owner, the Norwegian Coastal Administration (Kystverket).

stance, in France, we can identify several categories of missions:

- Assistance and rescue missions include the surveillance of distress signals, surveillance of the weather forecast, or the patrolling around beaches.
- Traffic and fishing regulation: this includes the regulation of the fishing zones, the surveillance of the tonnage and the content of ship's holds, the control of heavy traffic around busy areas like harbours or even the prevention of polluting activities.
- Territory surveillance: this consists of the surveillance and identification of vessels sailing in restricted areas and the apprehension of wrongdoers.

These missions not only vary depending on the country but also from one activity to another. Someone affected by the safety and rescue of tourists will have a very different everyday routine from a carrier pilot's.

8.1.1/ THE AUTOMATIC IDENTIFICATION SYSTEM

Like the ADS-B protocol, the Automatic Identification System (AIS) appeared in the years 1990 and has been broadly used since then. It was initially created to offer to vessels and surveillance officers an estimation of the traffic surrounding them at close range. The goal was to improve traffic flow by predicting the course of other ships, hence avoiding potential collisions. Since 2004, an AIS transponder is compulsory in vessels with a tonnage over 300 tons or embarking more than 12 passengers.

The AIS protocol uses the GNSS to determine the position of the vessels and uses the VHF frequencies to exchange information about their speed and the traffic surrounding them. The broadcast of the information is periodical and they are not encrypted nor authenticated. Different classes of AIS exist depending on the size of the vessel and determine how often they broadcast their information through AIS.

Initially, the AIS transponders were solely placed inside vessels and on the coasts, however, this configuration did not allow the collection of data in high seas because of the limited range of the VHF radio due to the curvature of the earth. In 2008, the introduction of satellites equipped with AIS receptors (S-AIS) allowed worldwide coverage making the surveillance of most maritime areas using the AIS protocol possible. Thanks to this technology upgrade, many new applications using the AIS protocol appeared (e.g the protection of fishing zones (Mazzarella et al., 2014) or the collision prevention between vessels and cetaceans (Wiley et al., 2011)).

8.1.2/ THE AIS VULNERABILITIES

Similar to the ADS-B protocol, officials do not communicate much about vulnerabilities and the overall security of the AIS protocol. Even though the International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA) showed concerns about loopholes in the protocol [2,3], it is harder to find equivalent conclusions from the International Maritime Organisation (IMO). One can nonetheless find researches on security analysis (Riveiro et al., 2018) that highlight the flows of the protocol's security, failing to prevent the manipulation of its messages.

The lack of encryption, the use of known and accessible radio frequencies, and the lack of authentication preventing any source verification are the main factors for any malicious individual to create, delete or modify AIS data.

Without surprise, we find similar attack patterns in the AIS protocol to the ones found in ADS-B. Some attacks have already been proven feasible through the AIS architecture (Balduzzi et al., 2014).

Here is a taxonomy of the different attacks possible found in the literature (Balduzzi et al., 2014; Goudosis and Katsikas, 2018):

[2]http://www.iainav.org/News/nws0456-ais-vulnerability.pdf
[3]https://www.navcen.uscg.gov/pdf/IALA_Guideline_1082_An_Overview_of_AIS.pdf

- **AIS-SART Spoofing** This is used to create fake emergency messages to lure rescue teams in a given area.
- **Fake weather forecast** Authorities may use the AIS protocol to broadcast weather forecasts. This attack tries to usurp the sender's identity to send fake weather reports.
- **Availability disruption** This attack's goal is to disrupt the AIS communication of a given vessel. There are three ways to do so. The first one is to simply turn off the AIS transponder, often done by fishing vessels to hide their real location (this attack is also referred to as AIS Shutdown). The second is called frequency hopping and consists of stealing the identity of an authority to force a vessel to change its emission frequency. Lastly, the straightforward flooding creates a deny of service attack.
- **Collision Prevention Assist Spoofing** The system of collision prevention on ships is based on AIS messages emitted by surrounding vessels to avoid sailing into them. This attack aims to create fake vessels in the close vicinity of others to set off their collision prevention assist alarm.
- **Ship Spoofing** This attack aims to create a fake vessel by using the identity of another one, often in other parts of the world.
- **Aid to Navigation spoofing** Some floating aids to navigation like buoys, beacons, or lights are equipped with AIS for the vessels to confirm their position even before seeing it visually. This attack aims to emit fake messages to disrupt the manoeuvers of vessels in the area.
- **AIS Hijacking** This type of attack uses strong signal emissions to override messages sent by vessels. This aims to modify the content of the AIS messages.

These different attacks can be treated like FDIAs and as such, the different work exposed in Chapter 3 for the ADS-B protocol are also applicable in the maritime domain and its AIS protocol. For this reason, the AIS protocol is a great use case

for our model to evaluate its genericity. The main concerns now before experimenting are the availability of the data and the choice of a contextual feature.

8.1.3/ DATA ACQUISITION

Creating a large-scale dataset of raw data like in ADS-B is a challenge in AIS. The data are mostly collected by the coastal base stations and satellites operated by companies and coastal guards. Considered as critical information, the states are often reluctant to open this data freely. When the AIS data are available, they have been pre-processed and cleaned. However, to create a model that can be generalized to every part of the world, using raw data is necessary. We can cite two organizations that work to make AIS data available. The first one is ”Global Fishing Watch” which tries to contact the governments directly to access their data, but the raw data they use in their research is not public. The second one is ”AIS hub”. They aim to become a raw AIS data sharing center where everyone can get memberships to share the data they collect on their own AIS receiving station. Although the project is interesting, the coasts' coverage is still limited, and data from satellites are necessary to cover the non-coastal areas.

As part of a Norwegian research project, our partner Simula Research Laboratory has obtained access to raw satellite AIS data from Statsat, a governmentally owned company that defines, develops and sources space infrastructure for Norwegian public purpose, which recently sent satellites to space to collect AIS data.[4]. This data is collected by four satellites, but they do not cover the whole earth simultaneously. Moreover, the AIS protocol through satellites (Skauen, 2019) has a limited capacity in dense areas such as the EU, China, or the US coast. This is due to the large areas covered by satellites, highly increasing the number of messages they get every second, leading to saturation. To limit this problem, we integrate data from ground stations from Norwegian coastal guards to this dataset that does not have this limitation thanks to their smaller coverage.

[4]https://directory.eoportal.org/web/eoportal/satellite-missions/content/-/article/norsat-1

8.1.4/ VESSEL TYPE: THE CONTEXTUAL FEATURE

Similar to the ADS-B protocol, the AIS is subject to contextual anomalies and attacks. The most blatant examples are vessels trying to conceal their real activity. One can notice that, compared to ADS-B where the data feed was disrupted by a third party, in AIS, the anomalies often come from the source itself. For instance, fishing ships hiding their real intention or cargo ships passing for smaller vessels. For this reason, the CAE model, by specializing its decoders in different vessel activities, would be a good candidate to detect fraudulent activities in AIS data. For that, the AIS protocol already provides a contextual feature called the Navigational Status (NS). This status defines the activity the vessel is currently doing and is set manually by the crew. It thus differs from the flight phase used in the ATC as it is not entirely bound to the data provided by AIS. Another difference is the diversity of vessels found in the dataset. For ADS-B, we made sure that only commercial flights were used for the training and the evaluation, avoiding erratic flights from amateurs. This is not as easily done in AIS because of the vessel types not being as well-defined as in ATC.

There are 16 different NS in AIS currently, with only the 9 first used worldwide. The other 7 are used only regionally or are reserved. These 9 NS are each tied to a number between 0 and 8:

0. **Under way using engine**. A vessel is considered to be underway when it meets the following criteria:
   - is not aground
   - is not at anchor
   - was not attached to a dock, the shore, or any other stationary object.

   This navigational status message refers to machinery vessels in motion.

1. **At anchor**. A vessel is at anchor when it is held in position by an anchor on the bottom of a body of water, thus preventing a vessel from drifting away from its desired position (e.g. waiting for a berth, heavy weather, receiving

fuel oil, loading, and unloading cargo, for maintenance purposes). The ”at anchor” state begins when the anchor hatches firmly hit the seabed and the ship is held in a certain position. While the vessel is considered to be ”underway” as soon as the anchor is weighed or towed on the seabed. The vessel is not fixed at a dock, which is called moored.

2. **Not under command**. The term “not under command” means a vessel that through some exceptional circumstance is unable to manoeuver and is therefore unable to keep out of the way of another vessel.

3. **Restricted Manoeuvrability**. Manoeuvring characteristics include turning, yaw-checking, course-keeping, and stopping abilities of the vessel. The term ”restricted manoeuvrability” means the vessel is unable to keep out of the way of another vessel. It also includes:

   – A vessel engaged in laying, servicing, or picking up a navigational mark, submarine cable, or pipeline.

   – A vessel engaged in dredging, surveying, or underwater operations.

   – A vessel engaged in replenishment or transferring persons, provisions, or cargo while underway.

   – A vessel engaged in the launching or recovery of aircraft.

   – A vessel engaged in mine clearance operations.

   – A vessel engaged in a towing operation that severely restricts the towing vessel and her tow in their ability to deviate from their course.

4. **Constrained by draught**. A power-driven vessel that is severely restricted in the ability to deviate from the course it is following. This is due to the draught in relation to the available depth and width of the navigable water.

5. **Moored**. Securing a vessel at a pier or elsewhere by several lines or cables to limit the movement.

- Multi-Buoy Moorings (MBM), conventional buoy moorings – A facility whereby a tanker is usually moored by a combination of the ship anchors forward and mooring buoys aft and held on a fixed heading.
- Single Point Mooring (SPM) – A facility whereby the tanker is secured by the bow to a single buoy or structure and is free to swing with the prevailing wind and current. Three types are common: Catenary Anchor Leg Mooring, Single Anchor Leg Mooring, and Turret Mooring.

6. **Aground**. A vessel that ran aground onto or on a shore, reef, or the bottom of a body of water.

7. **Engaged in Fishing**. The term ”engaged in fishing” means any vessel fishing with nets, lines, trawls, or other fishing apparatus.

8. **Under way sailing**. A vessel is considered to be underway when it meets the following criteria:

   - It is not aground
   - It is not at anchor
   - It was not attached to a dock, the shore, or any other stationary object.

   This navigational status message refers to all ships using wind power.

8.2/ THE AFFINITY AUTO-ENCODER

While using the contextual nature of time-series has shown conclusive results on ADS-B using the flight phase, it is not that easily applicable in the maritime domain due to the higher number of context from the status. Indeed, for the ADS-B experiment, only 3 decoders were needed to take care of the different contexts offered by the flight phases but the status would require 8 different ones to achieve an adapted architecture. One could easily find an application domain where the context has many different classes, leading to a CAE architecture way too uneven

in terms of encoder-decoder number, increasing the training time and the overall size of the model.

This problem needs to be tackled to claim any genericity of the CAE approach. As a way to try and improve our model as well as address the mentioned problem, a pre-training process was added to the CAE base model to calculate an affinity score between the different statuses. This section details how this affinity is calculated and how it can be used to reduce the number of decoders in the CAE when the number of contexts increases.

8.2.1/ CLASS AFFINITY

Based on the architecture presented by Fifty et al. (2021) on the calculation of task affinities for images, the new architecture tries to preemptively reduce the number of classes by calculating an affinity score between them. Classes with high affinity will be processed together in the same decoder in the CAE architecture. To do so, we use a simple auto-encoder model that will, during each step of its training, calculate the inter-gain between each status. The gain $\mathcal{G}_{C_iC_j}$ of a class C_i over the class C_j is defined as:

$$\mathcal{G}_{C_iC_j} = (1 - \mathcal{L}_{C_iC_j}/\mathcal{L}_{C_j})/lr \tag{8.1}$$

with $\mathcal{L}_{C_j}$ being the `MSE` loss between the input data of class j and its reconstructed counterpart, $\mathcal{L}_{C_iC_j}$ is also the `MSE` loss between the input data of class j and its reconstructed counterpart but with the model using the weights derived from the gradient descent on the loss $\mathcal{L}_{C_i}$. lr is the learning rate.

The higher the inter-gain is between two classes, the higher their affinity. Through the monitoring of the different inter-gain during the training of the auto-encoder, classes can be merged together and be considered as one for the training of a CAE model afterward.

Algorithm 3 shows a training step of the class affinity pre-training approach re-

turning the loss of the given step as well as the inter-gain between the different classes. It is to be noted that the calculation of the different gains does not intervene in the model itself as it is only updated using the weights calculated with the overall loss.

Algorithm 3: Algorithm of Affinity Auto-Encoder

Data: $(T, C, S) \in \mathbb{R}^{b \times w \times f} \times \mathbb{N}^b \times \mathbb{N}^b$
/* `T: 3-dimensional array containing the time-series data` */
/* `C: Array containing the context information of the data` */
/* `S: Sample-weights used for the loss calculation` */
Result: Model's loss $\mathcal{L}$, matrix G of inter-gains g for each category C

1 $\mathcal{R}_T \leftarrow \texttt{AE}(T, C)$
2 $\mathcal{L} \leftarrow \texttt{MSE}(T, \mathcal{R}_T, S)$
3 $\mathcal{L}_C \leftarrow (\mathcal{L}_{C_1}, \mathcal{L}_{C_2}, \ldots, \mathcal{L}_{C_n}) \leftarrow \texttt{MSE}_C(T, \mathcal{R}_T, S)$
4 $\mathcal{L}_{C_\sigma} \leftarrow \sum_{i=0}^{n} \mathcal{L}_{C_i}$
5 $\mathcal{L}_P \leftarrow (\mathcal{L}_C, \mathcal{L}_{C_\sigma})$
6 $\mathcal{G} \leftarrow []$
7 **foreach** $\mathcal{L}_{P_i} \in \mathcal{L}_P$ **do**
8 $W_{Pi} \leftarrow \texttt{get\_weights}(\mathcal{L}_{P_i}, \texttt{AE}, O)$ `// Gradient Descent`
9 $\mathcal{R}_{TP_i} \leftarrow \texttt{AE}(T, P, W_{P_i})$ `// Reconstruction using updated weights`
10 $\mathcal{L}_{P_iC} \leftarrow (\mathcal{L}_{P_iC_1}, \mathcal{L}_{P_iC_2}, \ldots, \mathcal{L}_{P_iC_n}, \mathcal{L}_{P_iC_\sigma}) \leftarrow \texttt{MSE}_C(T, \mathcal{R}_{TP_i}, S)$
11 $\mathcal{G}_{C_i} \leftarrow []$
12 **foreach** $(\mathcal{L}_{P_iC_j}, \mathcal{L}_{C_j}) \in (\mathcal{L}_{P_iC}, \mathcal{L}_C)$ **do**
13 $\mathcal{G}_{P_iC_j} \leftarrow (1 - \mathcal{L}_{P_iC_j}/\mathcal{L}_{C_j})/lr$ `// lr : learning rate`
14 $\mathcal{G}_{P_i}.append(\mathcal{G}_{P_iC_j})$
15 $\mathcal{G}.append(\mathcal{G}_{P_i})$
16 return $\mathcal{L}, \mathcal{G}$

The different parts of this algorithm can be detailed as followed:

- **line** $1-6$. Using a classic AE architecture, the total loss of a given step $\mathcal{L}_{C_\sigma}$ is calculated and then decomposed into n $\mathcal{L}_{C_n}$, one for each class.
- **line** $7-11$. Looping over each obtained loss, a gradient descent calculates the weights associated with the current loss $\mathcal{L}_{P_i}$. These weights are then used to calculate a new loss $\mathcal{L}_{P_iC}$, which represents the loss of a step calculated on all of the data, but with the model trained solely on the class associated with the current loss $\mathcal{L}_{P_i}$. This loss is composed of n $\mathcal{L}_{P_iC_n}$, again one for each class

- **line** $12-14$. For each $\mathcal{L}_{P_iC_n}$, the gain $\mathcal{G}_{C_iC_j}$ is calculated using both the loss $\mathcal{L}_{P_iC_n}$ and the global loss $\mathcal{L}_{C_j}$ which represents the loss of all of the data were used during this step instead of only the data of the class i. This gain represents how well the model is able to reconstruct the data from a class while it was trained on the current step using another class' data.

The evolution of the inter-gains over the different steps and epochs can then be used to determine the affinity between the different classes. Classes with high affinities mean that decoders trained with these classes merged together are likely to perform as well as separate decoders for each class. This helps decrease the number of decoders in the CAE model, decreasing its training time as well as its size, without a too significant loss in performance. This architecture is named the grouped CAE (G-CAE) for the remainder of this chapter.

8.3/ EXPERIMENTS

In this section, the preliminary experiments using the CAE in the maritime domain are presented. This study aims at specialising the different decoders in one or more NS in order to detect fraudulent fishing activities. This is still a work in progress yet to be published and this section does not present final results.

First, the data acquisition and formating are presented with an emphasis on the features extracted from the AIS data. Then, the remainder of the section presents the results obtained so far including the effect of the affinity score, the choice of status groups as well as the accuracies obtained by the different trained decoders.

8.3.1/ DATA ACQUISITION AND FORMATING

For the experiments, the satellite-based AIS dataset used is provided by the Norwegian Coastal Administration and collected by Statsat and consists of 4 050 019 441 AIS messages from all over the world, which corresponds to the messages

collected during the year 2020. From this dataset, messages from fishing boats are extracted using their ID resulting in a dataset of over 35,000 windows of 50 AIS messages.

For the first early experiments, 5 different types of fishing boats were used (namely drifting longlines, set longlines, squid jigger, trawlers, tuna purse seines), each coupled to the two main statuses of these boats (Engaged in Fishing, Under-way using engine) resulting in 10 different contexts. Table 8.1 shows the different contexts created with the number of windows they each have.

Context no	Boat Type	NS	samples	Group no
0	drifting longlines	Under way using engine	42663	1
1	drifting longlines	Engaged in Fishing	42663	1
2	set longlines	Under way using engine	35023	1
3	set longlines	Engaged in Fishing	33802	2
4	squid jigger	Under way using engine	34698	2
5	squid jigger	Engaged in Fishing	35000	2
6	trawlers	Under way using engine	29139	1
7	trawlers	Engaged in Fishing	30948	1
8	tuna purse seines	Under way using engine	34435	1
9	tuna purse seines	Engaged in Fishing	38216	1

Table 8.1: The different contexts were created using the chosen types and NS. It also shows the groups created by the affinity calculation.

From each message of the AIS dataset, relevant features were selected, namely position (latitude, longitude), timestamp (t), and speed (s). Also, we enrich the features with:

- Δt: the time difference compared to the preceding message from the same ship
- ΔDV : the difference in meters on the latitude with the preceding message from the same ship
- ΔDH : the difference in meters on the longitude, DP the distance to the port

We have chosen to split the relative distance with the previous message between ΔDH and ΔDV to keep the direction of movement.

Each window is linked to the context label that is used, like the flight phase for ADS-B, to separate the dataset into separate sets for each of the decoders to be trained on.

These data are then separated into a context-labeled training dataset as well as an evaluation dataset. The vessels contained in the evaluation dataset are not present in the training dataset. Indeed, we made sure through their identifier that they were not used in both. Once the model is trained with the training dataset, each decoder reconstructs the whole evaluation dataset to compare the reconstruction scores obtained in the different contexts. These scores can give first insights into the efficiency of the CAE model to detect abnormal activities in a given context.

8.3.2/ PRELIMINARY RESULTS OF THE CAE

We trained the CAE architecture presented in Chapter 7 with some different hyperparameters: latent space of 75, batch size of 256, a window size of 50.

Using this CAE architecture, with 10 decoders, we obtain $10 * 10$ losses distributions, one for each context reconstructed by each decoder. In Figure 8.1, the different loss distributions for each context obtained on the decoder specialised in context 3 are shown. The average loss of every context seems to be overall similar, meaning that this decoder in particular seems to properly reconstruct most of the data, regardless of their context. Results on other decoders show similar behaviours.

These results can be explained by three different phenomena:

1. First, the different decoders might not have managed to properly specialise themselves in a specific context. This could be caused by a training being too short or the decoders' architecture being too small.

2. The uniformity of the data. If regardless of contexts, vessels follow the same straight trajectory shape in most of their cruises, then these trajectories can

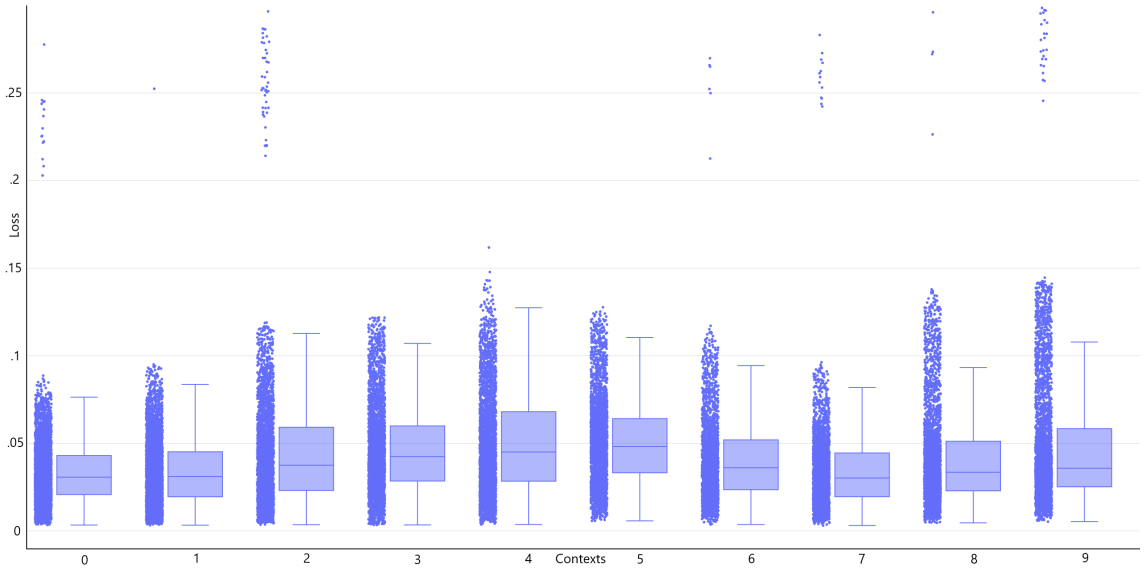


Figure 8.1: Losses of the decoder specialized in context 3 for each of the evaluation subsets.

be easily reconstructed by any of the decoders.

3. The chosen latent space is too wide. If the encoder properly manages to encapsulate all the data contained in its input in a latent space that is big enough, then the different decoders, similarly to point 2, regardless of their context, will be able to reconstruct most of the data. Only the more complex trajectories would be troublesome and yield higher losses.

For the first point, when analysed, the losses of each decoder, on their context data, have a limited number of data yielding high losses when compared to other contexts. This shows a specialisation acquired during training. This point is further discussed with the G-CAE.

For the third point, reducing the latent space until the reconstruction loss gets higher is a simple way to limit this phenomenon. Lastly, the second point could be a result of the data being actually all very similar. If the data are uniform across the different contexts, then it can be hard to prevent a small loss on most of the data using any of the decoders. This means that, while these contexts have a meaning for domain experts, they are not relevant in regards to the data analysis. In addition, this means that different decoders are trained on very similar data and

this introduces redundancy in the CAE architecture.

To avoid that, we can limit the number of decoders and specialise them in multiple contexts. The choice of the concatenated contexts would then be done through the use of the affinity score presented earlier in this chapter to transform this classical CAE into a G-CAE.

8.3.2.1/ AFFINITY SCORE AND GROUPING

After 3 epochs representing over 3000 steps of Algorithm 3, the calculation of the different affinities led to an inter-gain affinity between each context. Using a spectral clustering technique (Von Luxburg, 2007) on these affinities, the different similarity matrices created on a different number of groups showed that having 2 groups of contexts led to the best results. We used this technique because it creates clusters using the eigenvectors of a matrix usually deriving from a set of pairwise similarities between the points to be clustered, which is exactly what the inter-gain affinity matrix represents.

These two groups of context are: contexts 3,4 and 5 in one group and the rest in the other. The group numbers are shown in Table 8.1 in the column *Group no*.

8.3.2.2/ RESULTS WITH G-CAE

Using the groups of contexts defined by the spectral clustering on the affinity score, the G-CAE model is trained with only 2 decoders. The first effect of this reduction is a CAE architecture having less trainable variables, reducing its training time as well as its size. To be more precise, the trainable parameters went from 1,930,000 parameters to 450,000.

As shown in Figure 8.2, the losses of the epochs on both training and validation data are presented. While there is a slight discrepancy between the training losses, the validation losses stayed very similar with and without the grouping, meaning the G-CAE did not lose reconstruction capabilities compared to the CAE.

Once trained, the losses of the two decoders are presented in the boxplots in Figure 8.3 and Figure 8.4.

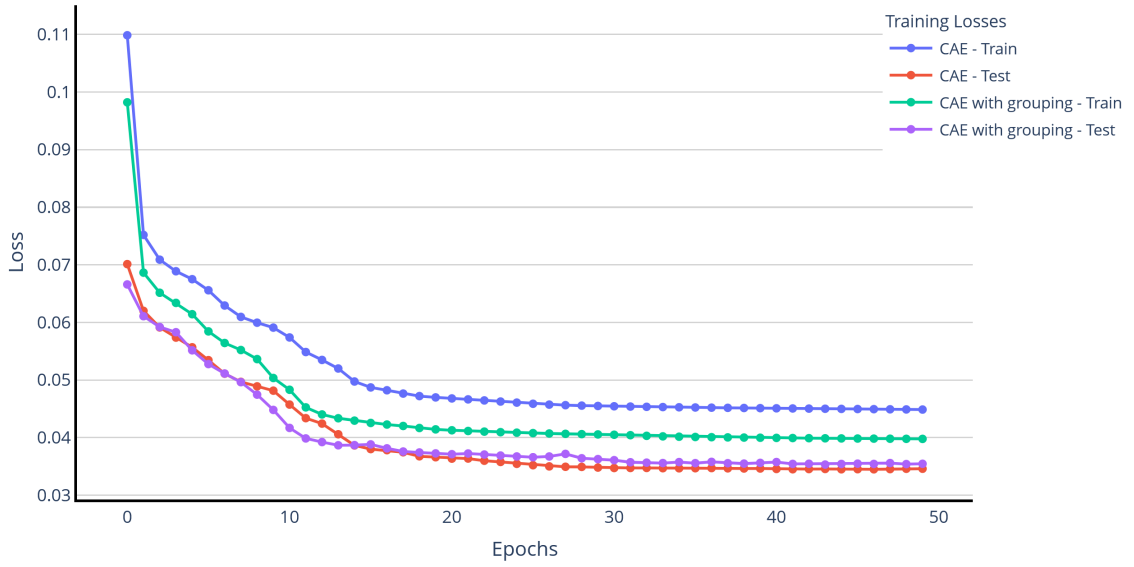


Figure 8.2: Training losses of the CAE with and without context grouping.

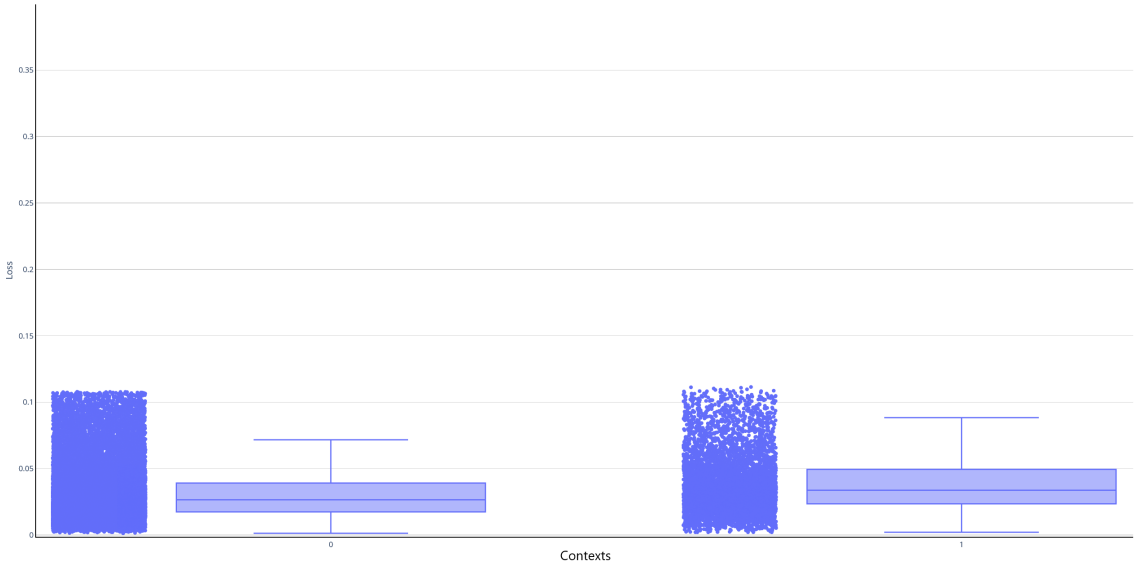


Figure 8.3: Boxplots of the evaluation losses of the decoder trained on group 1.

These boxplots confirm the trend that was noticed on the CAE architecture without the grouping. On data group 1, there seem to be data points that are well reconstructed by the first decoder, but not by the second decoder. Such a difference between the two decoders can be explained by their training on different

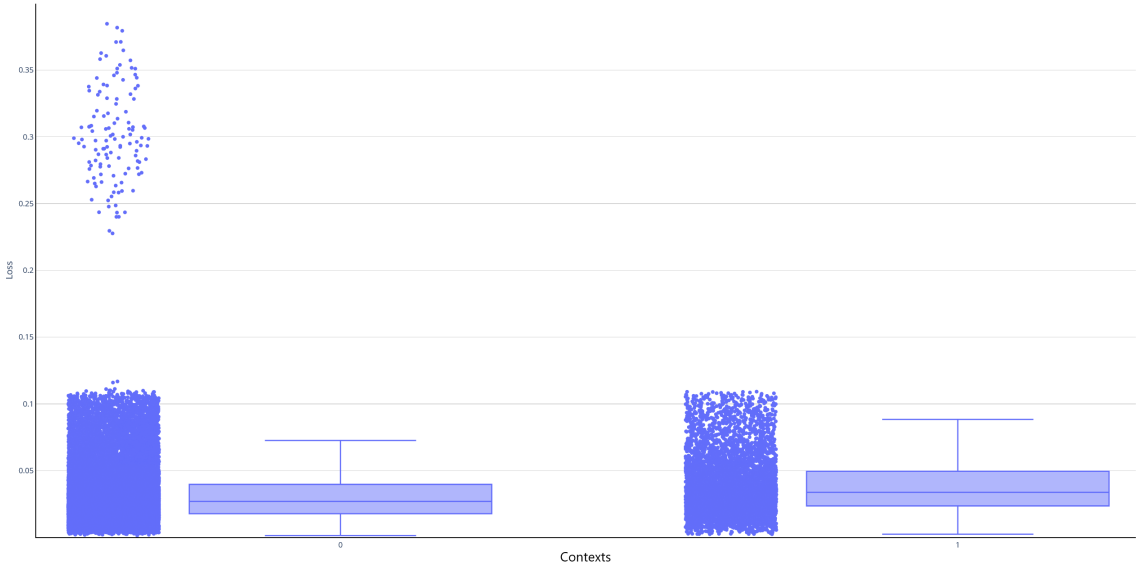


Figure 8.4: Boxplots of the evaluation losses of the decoder trained on group 2.

data, effectively specializing them. This, for at least one of the two groups of context, resulted in the detection of context-dependent data and with that, enabling the possibility of detecting contextual anomalies in AIS using the CAE.

8.3.3/ Discussion

The goal of these experiments in AIS data is to answer the last research question **RQ3: Is the CAE a field-specific approach or can it be extended to other transport domains?**.

Using the CAE on AIS data has shown that, regardless of the context, most fishing vessel trajectories can be reconstructed by any decoders trained on a subset of the training data. This is due to the fact that most of the said trajectories are straight lines that could not be associated with any specific context.

However, these first experiments also showed that some outliers were detected by some decoders, but not by others. This phenomenon was accentuated by the G-CAE results. This is an encouraging observation for anomaly detection as this would mean that some trajectories can be very specific to a given context, and could go unnoticed by a generic auto-encoder but not by the CAE.

Regardless of the potential efficiency of the CAE approach, detecting data specific to a context proves that the architecture can be used in AIS data, answering **RQ3**.

8.4/ SUMMARY

This chapter presented the first experiments on the AIS data using both the CAE model and its updated version, the G-CAE, with its affinity score calculation to reduce the number of decoders. This effectively limited the size of the model without having any noticeable influence on the accuracy of the CAE model.

Using Pierre Bernabé's own AIS data architecture, we successfully showed that it was possible to acquire context-awareness on vessel data despite most of the data being very similar. This means that the model could effectively be used to detect vessels claiming a false NS.

Future experimentation to the presented results will propose a cross-evaluation dataset with mislabeled trajectories to evaluate the anomaly detection capabilities of the approach.

This ends the third part of this thesis about the experiments done using the CAE model. The next and last chapter concludes this dissertation and presents perspectives and future work.

9

Conclusion and Future Work

This last chapter draws up a review of the thesis work that has been carried out over the last three years. First, we provide a concluding word on the research objective and its different research questions proposed in Chapter 1. Then, we propose perspectives and future work aiming at completing and improving our approach.

Conclusion of this Dissertation

The ADS-B protocol is currently being deployed world-wide in an effort to improve flight management. ADS-B requires participating aircraft to broadcast their information periodically in an encoded message. This technology embodies the shift from independent and non-cooperative surveillance technologies, historically used for aircraft surveillance, to dependent and cooperative technologies. In this new context, ground stations need aircraft to cooperate as they depend on aircraft's GNSS capabilities to determine their position.

Since 2020, ADS-B has been compulsory in most air-spaces but the protocol itself stayed sensibly the same as it was imagined twenty years ago when cybersecurity was not of the highest priority. As a result, anyone with the proper equipment can receive and create messages freely. This liberty in both emission and reception makes ADS-B vulnerable to spoofing, and more precisely to attacks called FDIA whose purpose is to create fake surveillance messages conscientiously re-

specting the protocol in order to dupe the air traffic controllers to believe in untrue situations.

Although ADS-B is not the only protocol used for flight tracking – e.g., radar technologies – it is, as of today, a central brick in the means of surveillance used by public air transportation. In this context, there has been a growing interest in conducting research on anomaly detection systems that address these new threats (Strohmeier et al., 2015b). Among the different existing solutions, some are based on Machine Learning (ML) anomaly detection models. These models already find applications in many different domains like power systems (Wang et al., 2018) or sensor networks (Malhotra et al., 2016) and have become quite popular in recent years. One downside of these models is their need for consequent data availability to achieve meaningful results. It is indeed critical for ML researchers to have access to reliable and genuine data sources to train their models. Thankfully, for ADS-B data, the OpenSky Network (Schäfer et al., 2014) is one of the best ADS-B source in terms of accessibility and data history in air transportation. Thanks to its historical database, any researcher can easily obtain surveillance data from almost anywhere on the globe.

From this environment, the question of securing ADS-B using ML led to the establishment of the examination of the state of the art in three different domains: False Data Injection Attacks (FDIA), the different existing cybersecurity technologies for ADS-B and the ML models for detecting anomalies in time-series. The analysis of these domains led to a better definition of the different baselines this thesis was built upon as well as different shortcomings that needed to be addressed:

- While many different models exist in the literature to detect anomalies in ADS-B data, none of them properly tackles the question of FDIAs and to what extent ML models could be used to detect such attacks. Most of the studies create their own handmade attacks, often disregarding any realism, thus not really giving any closure on the efficiency of ML models to detect FDIAs.

- The recent solutions to detect anomalies in multivariate time-series do not seem to properly acknowledge context from which the data are issued. In applications like the ADS-B protocol, it is primordial to add contextual awareness to a model to properly differentiate normal situations from anomalous ones.

From these different findings stems the main research objective of this thesis that is to **investigate the benefits of adding contextual awareness to unsupervised ML models to better detect FDIAs in air traffic surveillance data.**

This objective led to three research questions we identified as follows:

- **RQ1: To what extent unsupervised ML algorithms can detect False Data Injection Attacks in ADS-B data?**
- **RQ2: Is using the context of a data helpful in detecting anomalies?**
- **RQ3: Is the CAE a field-specific approach or can it be extended to other transport domains?**

From these observations, we oriented our work towards the different contributions of this thesis: the CAE model, the data pre-processing architecture for ADS-B data and the experiments in both ADS-B and AIS protocols.

The CAE Model

The basis of the CAE model itself uses the architecture of a classic AE model (Liou et al., 2014) made of an encoder and a decoder. The main difference with other AE is its unbalanced numbers of encoder and decoder depending on a context feature. The number of classes a context feature can take directly impacts the number of decoder the CAE will have.

The creation of the CAE model aimed at answering **RQ2**. The hunch behind that idea was that to detect FDIAs in ADS-B time-series, the context in which they

are issued is primordial. From the related work, while neural network models like AE are well suited to detect FDIAs, they are usually trained with as many training examples as possible, disregarding the context of the input data. With the CAE however, while ADS-B data is still always encoded by the same encoder, the different decoders manage to specialise in one specific context. Chapter 5 extensively presents the CAE architecture and how it is used along with a threshold selection method to detect anomalies in multivariate time-series.

The CAE model by itself does not quite answer **RQ2** though. There is a need to evaluate it and compare it against other ML models used to detect FDIAs and see if context helps to the detection of FDIAs, which is done in Chapter 7.

The ADS-B Data Architecture

For these experiments, we needed a stable and reliable source of ADS-B. Thanks to already existing tools to both aggregate and pre-process ADS-B data like the Opensky-Network (Schäfer et al., 2014) or the traffic library (Olive, 2019), creating an end-to-end pipeline from raw data to a training and evaluation dataset for ML model has never been as possible as it is today.

To complete said pipeline presented in Chapter 6, we added different missing parts:

- Different format conversions enabling the use of raw data.
- A method based on the calculation of Vincenty distances between related ADS-B messages enabling the detection and the filtering of outlier messages.
- The improved phase identification using the fuzzy logic introduced by Sun et al. (2017) and taking into account the distance from the departure and arrival airport to improve its accuracy in case of an FDIA attack.

The ADS-B Experiments

With the CAE and an operational data pipeline to evaluate ML models, experiments on ADS-B data could be realised in order to answer **RQ1** and **RQ2**.

RQ2 was directly answered by the creation and evaluation of the CAE model. The different experiments made during this thesis to compare this model to other baselines used to detect anomalies in ADS-B messages showed an improvement at detecting more realistic abnormal situations while still detecting coarse FDIAs. It also showed good reactivity on real-life anomalies compared to other AE architectures.

As to the **RQ1**, answering it is not straightforward.

First, ML model **can** be used to detect FDIAs using solely ADS-B data. They have shown good accuracies on most coarse anomalies and the addition of the context awareness improved further these results.

On the other hand, for more complex anomalies, even though adding context awareness to a model helped detect more subtle anomalies than with other models, there is a limit from which these attacks go undetected. Future consideration in this domain could be to study the potency of an FDIA going undetected by ML models. Indeed, if an undetected FDIA does not disrupt the traffic, is it then worth detecting?

The AIS Experiments

Finally, and as a mean to broaden horizons of the CAE model, first experiments on the AIS protocol were presented in Chapter 8. The AIS protocol is the VTS equivalent of ADS-B for ATC and, as such, has very similar problems related to security.

This opportunity allowed us to have a ground for experiments aiming at answering our **RQ3**. These experiments, done with Ph.D. candidate Pierre Bernabé, showed encouraging results on the use of AIS data to detect anomalous behaviour from

fishing boats. This is still a work in progress and the results need to be refined and compared to other approaches to give a definitive answer to the use of the CAE model in VTS domain.

Future Work

From the different presented contributions, several extensions and improvements can be identified for future work. The CAE approach itself on anomaly detection tasks, while it has already shown good results on ADS-B data, needs to be confronted with other datasets from domains outside transport data. To that end, many time-series datasets exist like the SWaT (Secure Water Treatment) dataset (Goh et al., 2016) or the SMD (Server Machine Dataset)[1] to further experiment on the CAE and show its genericity.

These extra experiments would be a good opportunity to test on aspects not tackled by experiments presented in this thesis. These aspects can be divided into 3 different orientations: new sources of data, CAE model improvements and extended experiments.

New Types of Data

The frame of the thesis was limited to the use of ADS-B. One of the conclusions of the different experiments presented in this dissertation is that, with ADS-B alone, it can be tedious to detect all FDIAs and to properly discriminate them from real-life anomalies. Including other data sources to the training data could help to add additional context to models. Some of these sources include:

- Weather data. Course inconsistencies and changes of altitude in ADS-B are often due to meteorological events. Including them in the input data could

[1] https://github.com/NetManAIOps/OmniAnomaly

give additional insights to the model not to treat some events as anomalous but rather due to weather emergencies.

- Other mode S data. For instance, in his thesis, Sun (2019) showed that it was possible to decode COMM-B messages and compare the data found there to the ones found in ADS-B. In addition, these data combined together could enable the calculation of the wind spind by comparing the true airspeed to the ground speed of a flight.
- Adding data from the close surrounding flights. This could add another layer of context to the data of a given flight by showing the trends of the trajectories of other airplanes in its vicinity.

Developing the CAE Model

The CAE model developed in this dissertation brings forth other implementation ideas to improve its own effectiveness and the effectiveness of other models, such as:

- Multi-thresholding to try and separate the gravity and nature of anomalies. Indeed, the different experiments showed disparities of anomaly score depending on the anomaly detected. Using different thresholds would enable the model to take into account these disparities. This could lead to a different level of alarms and possibly allow, to some extent, a distinction between real-life anomalies and FDIAs.
- Expending the contextual awareness to other types of models. The approach of dividing the contexts could be implemented into different other architectures such as VAE or transformers. This architecture could then be able to eventually take advantage of context aweraness to improve their accuracy.
- Different decoders depending on the complexity of the data. As shown in the experiments, the complexity of the data between the different contexts

can vary quite sensibly leading to disparities in loss between the decoders. One way to palliate this would be to use bigger decoders for contexts with more complex data. While this approach could be hard to implement on applications with many different contexts, this could help have better results on a smaller number of contexts.

Extended Experiments

Extended experiments could bring more insights on different aspects of the models like:

- A deeper evaluation of the impact of the hyperparameters. For instance, parameters like the size of the latent space or the size of the different encoder/decoders have shown to have quite an extensive effect on the reconstruction and the overall performances of the CAE model. Developing a more extensive analysis on the impact of the different choices made to train the model could help improve its performance.
- Complexity analysis for the affinity calculation. As shown in Chapter 8, the pre-training affinity calculation can help reduce the number of decoders the CAE model gets by merging similar contexts together. While experiments are still underway for publication, an analysis of the complexity of this approach, based on the number of epochs necessary to calculate this affinity score, could help analyse the time gains compared to a standalone CAE model.

Final Words

Through a large amount of open ADS-B data available for researchers, the study of air traffic anomalies using Machine Learning has never been as easy as it is now. The different chapters of this thesis presented a new take on analysing

ADS-B data to raise potential alarms in the sky. The end-goal of the work presented would be the integration of the trained model into a security chain aiming at helping controllers in their everyday work.

Concerning the CAE architecture, the experiments made during this thesis showed that its contextual awareness allowed it to better detect anomalies compared to other ML models. These results will hopefully lead to more comparable studies using context when dealing with multivariate time-series.

BIBLIOGRAPHY

F. A. Administration. Automatic dependent surveillance—broadcast (ads–b) out performance requirements to support air traffic control (atc) service. Final Rule, Final Rule, CFR Part 91, Federal Register 75 (103), 2010.

S. Akerman, E. Habler, and A. Shabtai. Vizads-b: Analyzing sequences of ads-b images using explainable convolutional lstm encoder-decoder to detect cyber attacks, 2019.

A. Ameli, A. Hooshyar, E. F. El-Saadany, and A. M. Youssef. Attack detection and identification for automatic generation control systems. IEEE Transactions on Power Systems, 33(5):4760–4774, 2018. doi: 10.1109/TPWRS.2018.2810161.

J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga. USAD: Unsupervised Anomaly Detection on Multivariate Time Series, page 3395–3404. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450379984. URL https://doi.org/10.1145/3394486.3403392.

J. Baek, E. Hableel, Y.-J. Byon, D. S. Wong, K. Jang, and H. Yeo. How to protect ads-b: Confidentiality framework and efficient realization based on staged identity-based encryption. IEEE Transactions on Intelligent Transportation Systems, 18(3):690–700, 2017. doi: 10.1109/TITS.2016.2586301.

M. Balduzzi, A. Pasta, and K. Wilhoit. A security evaluation of ais automated identification system. In Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC '14, page 436–445, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450330053. doi: 10.1145/2664243.2664257. URL https://doi.org/10.1145/2664243.2664257.

J. G. A. Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Computers and

Electronics in Agriculture, 153:46–53, 2018. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2018.08.013. URL https://www.sciencedirect.com/science/article/pii/S0168169918304617.

L. Basora, X. Olive, and T. Dubot. Recent advances in anomaly detection methods applied to aviation. Aerospace, 6(11), 2019. ISSN 2226-4310. doi: 10.3390/aerospace6110117. URL https://www.mdpi.com/2226-4310/6/11/117.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11, page 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc. ISBN 9781618395993.

A. Bianco, M. Garcia Ben, E. J. Martinez, and V. Yohai. Outlier detection in regression models with arima errors using robust estimates. Journal of Forecasting, 20, 12 2001. doi: 10.1002/for.768.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pages 93–104, 2000.

R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey, 2019.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. ACM Comput. Surv., 41(3), July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL https://doi.org/10.1145/1541880.1541882.

A. Chevrot, A. Vernotte, P. Bernabe, A. Cretin, F. Peureux, and B. Legeard. Improved testing of ai-based anomaly detection systems using synthetic surveillance data. Proceedings, 59(1), 2020. ISSN 2504-3900. doi: 10.3390/proceedings2020059009. URL https://www.mdpi.com/2504-3900/59/1/9.

A. Chevrot, A. Vernotte, and B. Legeard. Cae : Contextual auto-encoder for multivariate time-series anomaly detection in air transportation. Computers

& Security, page 102652, 2022. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2022.102652. URL https://www.sciencedirect.com/science/article/pii/S0167404822000517.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

E. Cook. Ads-b, friend or foe: Ads-b message authentication for nextgen aircraft. In 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, pages 1256–1261, Aug 2015. doi: 10.1109/HPCC-CSS-ICESS.2015.201.

A. Cretin. Conception et génération de tests par altération de données pour les systèmes de contrôle et de surveillance des transports : application aux domaines aérien et maritime. Theses, Université Bourgogne Franche-Comté, Mar. 2021. URL https://tel.archives-ouvertes.fr/tel-03324424.

A. Cretin, B. Legeard, F. Peureux, and A. Vernotte. Increasing the resilience of ATC systems against false data injection attacks using DSL-based testing. In ICRAT'18, Doctoral Symposium, pages 1–4, Barcelona, Spain, June 2018.

G. Dan and H. Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, pages 214–219. IEEE, 2010.

Z. Ding and M. Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proceedings Volumes, 46(20):12–17, 2013. ISSN 1474-6670. doi: https://doi.org/10.3182/20130902-3-CN-3020.00044. URL https://www.sciencedirect.com/science/article/pii/S1474667016314999. 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013.

H. Dutta, C. Giannella, K. Borne, and H. Kargupta. Distributed top-k outlier detection from astronomy catalogs using the demac system. 04 2007. doi: 10.1137/1.9781611972771.47.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, page 226–231. AAAI Press, 1996.

T. Fawcett and F. J. Provost. Combining data mining and machine learning for effective user profiling. In KDD, 1996.

C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn. Efficiently identifying task groupings for multi-task learning. Advances in Neural Information Processing Systems, 34, 2021.

C. Finke, J. Butts, and R. F. Mills. Ads-b encryption: confidentiality in the friendly skies. In CSIIRW '13, 2013.

P. H. Franses. Seasonality, non-stationarity and the forecasting of monthly time series. International Journal of forecasting, 7(2):199–208, 1991.

A. Fried and M. Last. Facing airborne attacks on ads-b data with autoencoders. Computers & Security, page 102405, 2021. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2021.102405. URL https://www.sciencedirect.com/science/article/pii/S0167404821002297.

N. Fumo and M. Rafe Biswas. Regression analysis for prediction of residential energy consumption. Renewable and Sustainable Energy Reviews, 47:332–343, 2015. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2015.03.035. URL https://www.sciencedirect.com/science/article/pii/S1364032115001884.

S. Fute, W. Buhong, Y. Fuhu, and L. Tengyao. Multidevice false data injection attack models of ADS-b multilateration systems. Security and Communication Networks, 2019:1–11, mar 2019. doi: 10.1155/2019/8936784.

N. Ghose and L. Lazos. Verifying ads-b navigation information through doppler shift measurements. 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), pages 4A2–1–4A2–11, 2015.

J. A. Goguen. L. a. zadeh. fuzzy sets. information and control, vol. 8 (1965), pp. 338–353. - l. a. zadeh. similarity relations and fuzzy orderings. information sciences, vol. 3 (1971), pp. 177–200. Journal of Symbolic Logic, 38(4):656–657, 1973. doi: 10.2307/2272014.

J. Goh, S. Adepu, K. Junejo, and A. Mathur. A dataset to support research in the design of secure water treatment systems. 10 2016.

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT press, 2016.

A. Goudosis and S. Katsikas. Towards a secure automatic identification system (ais). Journal of Marine Science and Technology, 24, 05 2018. doi: 10.1007/s00773-018-0561-3.

E. Habler and A. Shabtai. Using lstm encoder-decoder algorithm for detecting anomalous ADS-B messages. Computers & Security, 78:155–173, 2018.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

D. Iverson, R. Martin, M. Schwabacher, L. Spirkovska, W. Taylor, R. Mackey, and J. Castle. General purpose data-driven system monitoring for space operations. Journal of Aerospace Computing, Information, and Communication, 9, 10 2012. doi: 10.2514/1.54964.

V. M. Janakiraman and D. Nielsen. Anomaly detection in aviation data using extreme learning machines. In 2016 International Joint Conference on Neural Networks (IJCNN), pages 1993–2000, 2016. doi: 10.1109/IJCNN.2016.7727444.

G. Jarry, D. Delahaye, F. Nicol, and E. Feron. Aircraft atypical approach detection using functional principal component analysis. Journal of Air Transport Management, 84:101787, 2020. ISSN 0969-6997. doi: https://doi.org/10.1016/

j.jairtraman.2020.101787. URL https://www.sciencedirect.com/science/article/pii/S0969699719303266.

A. Kundu, A. Sahu, E. Serpedin, and K. Davis. A3d: Attention-based auto-encoder anomaly detector for false data injection attacks. Electric Power Systems Research, 189:106795, 2020. ISSN 0378-7796. doi: https://doi.org/10.1016/j.epsr.2020.106795. URL https://www.sciencedirect.com/science/article/pii/S0378779620305988.

S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In The 2010 International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2010. doi: 10.1109/IJCNN.2010.5596468.

M. Leonardi. Ads-b anomalies and intrusions detection by sensor clocks tracking. IEEE Transactions on Aerospace and Electronic Systems, 55(5):2370–2381, 2019. doi: 10.1109/TAES.2018.2886616.

J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in nlp. arXiv preprint arXiv:1506.01066, 2015.

T. Li, B. Wang, F. Shang, J. Tian, and K. Cao. Ads-b data attack detection based on generative adversarial networks. In J. Vaidya, X. Zhang, and J. Li, editors, Cyberspace Safety and Security, pages 323–336, Cham, 2019. Springer International Publishing. ISBN 978-3-030-37337-5.

T. Li, B. Wang, F. Shang, J. Tian, and K. Cao. Dynamic temporal ADS-B data attack detection based on shdp-hmm. Comput. Secur., 93:101789, 2020. doi: 10.1016/j.cose.2020.101789.

C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou. Autoencoder for words. Neurocomputing, 139:84–96, 2014. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2013.09.055. URL https://www.sciencedirect.com/science/article/pii/S0925231214003658.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In 2008 eighth ieee international conference on data mining, pages 413–422. IEEE, 2008.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data, 6(1), Mar. 2012. ISSN 1556-4681. doi: 10.1145/2133360.2133363. URL https://doi.org/10.1145/2133360.2133363.

Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. ACM Transactions on Information and System Security (TISSEC), 14(1):13, 2011.

X. Lu, Y. Tsao, S. Matsuda, and C. Hori. Speech enhancement based on deep denoising autoencoder. In Interspeech, volume 2013, pages 436–440, 2013.

P. Luo, B. Wang, T. Li, and J. Tian. Ads-b anomaly data detection model based on vae-svdd. Computers & Security, 104:102213, 2021. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2021.102213. URL https://www.sciencedirect.com/science/article/pii/S0167404821000377.

J. Ma and S. Perkins. Time-series novelty detection using one-class support vector machines. In Proceedings of the International Joint Conference on Neural Networks, 2003., volume 3, pages 1741–1745 vol.3, 2003. doi: 10.1109/IJCNN.2003.1223670.

M. Ma. Resilience against false data injection attack in wireless sensor networks. In Handbook of Research on Wireless Security, pages 628–635. IGI Global, 2008.

P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148, 2016.

J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In T. Honkela, W. Duch, M. Girolami, and S. Kaski, editors, Artificial Neural Networks and Machine Learning – ICANN 2011, pages 52–59, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21735-7.

F. Mazzarella, M. Vespe, D. Damalas, and G. Osio. Discovering vessel activities at sea using ais data: Mapping of fishing footprints. In 17th International Conference on Information Fusion (FUSION), pages 1–7, 2014.

D. McCallie, J. Butts, and R. Mills. Security analysis of the ads-b implementation in the next generation air transportation system. International Journal of Critical Infrastructure Protection, 4(2):78–87, 2011.

G. Miebs, M. Mochol-Grzelak, A. Karaszewski, and R. A. Bachorz. Efficient strategies of static features incorporation into the recurrent neural network. Neural Processing Letters, 51(3):2301–2316, Jun 2020. ISSN 1573-773X. doi: 10.1007/s11063-020-10195-x. URL https://doi.org/10.1007/s11063-020-10195-x.

M. Monteiro, A. Barreto, R. Division, T. Kacem, J. Carvalho, D. Wijesekera, and P. Costa. Detecting malicious ads-b broadcasts using wide area multilateration. In 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), pages 4A3–1–4A3–12, Sep. 2015. doi: 10.1109/DASC.2015.7311413.

A. Ng et al. Sparse autoencoder. CS294A Lecture notes, 72(2011):1–19, 2011.

M. S. Nolan. Fundamentals of air traffic control. Cengage learning, 2011.

X. Olive. traffic, a toolbox for processing and analysing air traffic data. Journal of Open Source Software, 4:1518, 2019. ISSN 2475-9066. doi: 10.21105/joss.01518.

X. Olive and L. Basora. Identifying Anomalies in past en-route Trajectories with Clustering and Anomaly Detection Methods. In ATM Seminar 2019, VIENNE, Austria, June 2019. URL https://hal.archives-ouvertes.fr/hal-02345597.

X. Olive, J. Grignard, T. Dubot, and J. Saint-Lot. Detecting Controllers' Actions in Past Mode S Data by Autoencoder-Based Anomaly Detection. In SESAR Innovation Days 2018, SALZBURG, Austria, Dec. 2018. URL https://hal.archives-ouvertes.fr/hal-02338690.

S. Ozdemir. Energy-efficient false data detection in wireless sensor networks. In Wireless Sensing and Processing, volume 6248, page 62480E. International Society for Optics and Photonics, 2006.

D. Park, Y. Hoshi, and C. C. Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. IEEE Robotics and Automation Letters, 3(3):1544–1551, July 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2801475.

M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. Signal Processing, 99:215–249, 2014. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2013.12.026. URL https://www.sciencedirect.com/science/article/pii/S016516841300515X.

D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061, 2020.

M. Riveiro, G. Pallotta, and M. Vespe. Maritime anomaly detection: A review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8:e1266, 05 2018. doi: 10.1002/widm.1266.

S. Samonas and D. Coss. The cia strikes back: Redefining confidentiality, integrity and availability in security. Journal of Information System Security, 10(3), 2014.

M. Schäfer, V. Lenders, and I. Martinovic. Experimental analysis of attacks on next generation air traffic communication. In International Conference on Applied Cryptography and Network Security, pages 253–271. Springer, 2013.

M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm. Bringing up opensky: A large-scale ads-b sensor network for research. In Proceedings of the 13th international symposium on Information processing in sensor networks, pages 83–94. IEEE Press, 2014.

M. Schäfer, P. Leu, V. Lenders, and J. Schmitt. Secure motion verification using the doppler effect. In Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, pages 135–145, 2016.

L. Sergey, O. Hubbard, Z. Ding, H. Ghadaki, J. Wang, and T. Ponsford. Advanced mitigating techniques to remove the effects of wind turbines and wind farms on primary surveillance radars. In 2008 IEEE Radar Conference, pages 1–6. IEEE, 2008.

A. N. Skauen. Ship tracking results from state-of-the-art space-based ais receiver systems for maritime surveillance. CEAS Space Journal, 11(3):301–316, 2019. ISSN 1868-2510. doi: 10.1007/s12567-019-00245-z. URL https://doi.org/10.1007/s12567-019-00245-z.

M. I. Skolnik. Radar handbook, 3rd edition. 2008.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

M. Strohmeier, V. Lenders, and I. Martinovic. Intrusion detection for airborne communication using phy-layer information. In DIMVA, 2015a.

M. Strohmeier, V. Lenders, and I. Martinovic. On the security of the automatic dependent surveillance-broadcast protocol. 17:1066–1087, 2015b. ISSN 2373-745X. doi: 10.1109/COMST.2014.2365951.

M. Strohmeier, M. Schäfer, R. Pinheiro, V. Lenders, and I. Martinovic. On perception and reality in wireless air traffic communications security. IEEE Transactions on Intelligent Transportation Systems, 18(6):1338–1357, 2017. doi: 10.1109/TITS.2016.2612584.

Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, page 2828–2837, New York, NY,

USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330672. URL https://doi.org/10.1145/3292500.3330672.

J. Sun. Open Aircraft Performance Modeling Based on an Analysis of Aircraft Surveillance Data. PhD thesis, Delft University of Technology, 2019.

J. Sun. The 1090 megahertz riddle: A guide to decoding mode s and ads-b signals. TU Delft OPEN Publishing, 2021.

J. Sun, J. Ellerbroek, and J. Hoekstra. Flight extraction and phase identification for large automatic dependent surveillance–broadcast datasets. Journal of Aerospace Information Systems, pages 1–6, 2017.

R. Trim. Mode s: an introduction and overview (secondary surveillance radar). Electronics & Communication Engineering Journal, 2(2):53–59, 1990.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

T. Vincenty. Geodetic inverse solution between antipodal points, Aug. 1975. URL https://doi.org/10.5281/zenodo.32999. Scanned by Charles Karney from the copy in R. H. Rapp's library at Ohio State University. The report is a work of the U.S. Government and so is in the public domain.

U. Von Luxburg. A tutorial on spectral clustering. Statistics and computing, 17(4): 395–416, 2007.

J. Wang, D. Shi, Y. Li, J. Chen, H. Ding, and X. Duan. Distributed framework for detecting pmu data manipulation attacks with deep autoencoders. 10:4401–4410, 2018. ISSN 1949-3061. doi: 10.1109/TSG.2018.2859339.

D. N. Wiley, M. Thompson, R. M. Pace, and J. Levenson. Modeling speed restrictions to mitigate lethal collisions between ships and whales in the stellwagen bank national marine sanctuary, usa. Biological Conservation, 144(9):2377–2381, 2011. ISSN 0006-3207. doi: https://doi.org/10.1016/j.biocon.2011.05.007. URL https://www.sciencedirect.com/science/article/pii/S0006320711001947.

L. Xie, Y. Mo, and B. Sinopoli. False data injection attacks in electricity markets. In Smart Grid Communications (SmartGridComm), First International Conference on, pages 226–231. IEEE, 2010.

K. Yang, M. Bi, Y. Liu, and Y. Zhang. Lstm-based deep learning model for civil aircraft position and attitude prediction approach. In 2019 Chinese Control Conference (CCC), pages 8689–8694, July 2019. doi: 10.23919/ChiCC.2019.8865874.

X. Ying, J. Mazer, G. Bernieri, M. Conti, L. Bushnell, and R. Poovendran. Detecting ads-b spoofing attacks using deep neural networks. 2019.

D. Yook, S.-G. Leem, K. Lee, and I.-C. Yoo. Many-to-many voice conversion using cycle-consistent variational autoencoder with multiple decoders. In Proc. Odyssey 2020 The Speaker and Language Recognition Workshop, pages 215–221, 2020.

J. J. Q. Yu, Y. Hou, and V. O. K. Li. Online false data injection attack detection with wavelet transform and deep neural networks. IEEE Transactions on Industrial Informatics, 14(7):3271–3280, 2018. doi: 10.1109/TII.2018.2825243.

R. Zhang, G. Liu, J. Liu, and J. P. Nees. Analysis of message attacks in aviation datalink communication. IEEE Access, 2017.

D. Zhao, J. Sun, and G. Gui. En-route multilateration system based on ads-b and tdoa/aoa for flight surveillance systems. In 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), pages 1–6, 2020. doi: 10.1109/VTC2020-Spring48590.2020.9129436.

J. Zhao, G. Zhang, M. La Scala, Z. Y. Dong, C. Chen, and J. Wang. Short-term state forecasting-aided method for detection of smart grid general false data injection attacks. IEEE Transactions on Smart Grid, 8(4):1580–1590, 2017. doi: 10.1109/TSG.2015.2492827.

T. Zhou and K. Chakrabarty. Authentication of sensor network flooding based on neighborhood cooperation. In IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006., volume 2, pages 665–670. IEEE, 2006.

S. Zhu, S. Setia, S. Jajodia, and P. Ning. Interleaved hop-by-hop authentication against false data injection attacks in sensor networks. ACM Transactions on Sensor Networks (TOSN), 3(3):14–es, 2007.

List of Figures

List of Tables

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — https://github.com/gallandarakhneorg/tex-templates
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

SPIM
école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Detection of Contextual Anomalies in Air Traffic Data Using Neural Network Models

Keywords: Anomaly detection, Machine Learning, Time-series, ADS-B

Abstract:

To face the ever-growing number of aircraft flying in the world airspace, the Air Traffic Control (ATC) needs to adapt and propose new technologies that are both cheaper to produce and more accurate, sometimes at the expense of the cybersecurity of its systems. The Automatic Dependent Surveillance-Broadcast (ADS-B) protocol is one of the latest compulsory advances in air surveillance and perfectly depicts this tendency: cheaper to maintain and to implement thanks to inexpensive transponders installed in each aircraft broadcasting their GPS information but way more vulnerable than older technologies. This is due to the change of paradigm that the ADS-B embodies. Older technologies like primary and secondary radars were watching the sky using powerful antennas tracking flights in a given area while with ADS-B, radars do not scan the air but rather directly receive information sent by each aircraft. For this reason, and because of the absence of encryption and authentication in the protocol, it is particularly vulnerable to False Data Injection Attacks (FDIA). FDIAs are messages either created, modified, or deleted by malicious individuals with the intent to disrupt traffic management. To limit these threats, different solutions were proposed using both the physical layer (analysis of the signals' strength, multilateration ...) and the logical layer (data fusion, group verification) including Machine Learning models. The main incentives for the latter are the recent data sources and tools available to obtain flight tracking records. This allowed the researchers to create datasets and develop Machine Learning models capable of detecting anomalies in En-Route trajectories. In this context, we propose a novel multivariate anomaly detection model called Contextual Auto-Encoder (CAE). It uses the baseline of a regular LSTM-based auto-encoder but with several decoders, each getting data of a specific flight phase (e.g., climbing, cruising, or descending) during its training. To illustrate the CAE's efficiency, an evaluation dataset was created using real-life anomalies as well as realistically crafted trajectory modifications, with which the CAE, as well as three anomaly detection models from the literature, were evaluated. To complete this work, and to show the genericity of our approach, experiments on the maritime domain are presented at the end of this thesis. This choice was motivated by the Automatic Identification System or AIS being a similar protocol to the ADS-B with similar problematics but for vessels. These new experiments led to an extension of the original model using an affinity score to merge contexts together.

Titre : Détection d'anomalies contextuelles dans les données de trafic aérien à l'aide de réseaux de neurones

Mots-clés : Détection d'anomalies, Machine Learning, séries temporelles, ADS-B

Résumé :

Pour faire face au nombre sans cesse croissant d'avions volant dans l'espace aérien mondial, le contrôle du trafic aérien (ATC pour Air Traffic Control) doit s'adapter et proposer de nouvelles technologies à la fois moins coûteuses à produire et plus précises, parfois au détriment de la cybersécurité de ses systèmes. Le protocole ADS-B (Automatic Dependent Surveillance-Broadcast) est l'une des dernières avancées obligatoires en matière de surveillance aérienne et illustre parfaitement cette tendance : moins cher à entretenir et à mettre en œuvre grâce à des transpondeurs peu coûteux installés dans chaque avion et diffusant leurs informations GPS, mais bien plus vulnérable que les technologies plus anciennes. Ceci est dû au changement de paradigme que l'ADS-B incarne. Les anciennes technologies, comme les radars primaires et secondaires, surveillaient le ciel à l'aide de puissantes antennes pour suivre les vols dans une zone donnée, alors qu'avec l'ADS-B, les radars ne scrutent pas l'air mais reçoivent directement les informations envoyées par chaque avion. Pour cette raison, et du fait de l'absence de cryptage et d'authentification dans le protocole, celui-ci est particulièrement vulnérable aux attaques par injection de fausses données (FDIA pour False Data Injection Attack). Les FDIA sont des messages créés, modifiés ou supprimés par des personnes malveillantes dans le but de perturber la gestion du trafic. Pour limiter ces menaces, différentes solutions ont été proposées en utilisant à la fois la couche physique (analyse de la force des signaux, multilatération...) et la couche logique (fusion de données, vérification de groupe), y compris des modèles d'apprentissage automatique. Les principales motivations de ces derniers sont les sources de données récentes et les outils disponibles pour obtenir des enregistrements de suivi de vol. Ceci a permis aux chercheurs de créer des jeux de données et de développer des modèles de Machine Learning capables de détecter des anomalies dans les trajectoires En-Route. Dans ce contexte, nous proposons un nouveau modèle multivarié de détection d'anomalies appelé Contextual Auto-Encoder (CAE). Il utilise la base d'un auto-encodeur LSTM ordinaire mais avec plusieurs décodeurs, chacun recevant des données d'une phase de vol spécifique (par exemple, montée, croisière ou descente) pendant son apprentissage. Pour illustrer l'efficacité du CAE, un ensemble de données d'évaluation a été créé en utilisant des anomalies réelles ainsi que des modifications de trajectoire réalisées de manière réaliste, avec lesquelles le CAE, ainsi que trois modèles de détection d'anomalies de la littérature, ont été évalués. Pour compléter ce travail, et pour montrer la généricité de notre approche, des expériences sur le domaine maritime sont présentées à la fin de cette thèse. Ce choix a été motivé par le fait que le système d'identification automatique ou AIS est un protocole similaire à l'ADS-B avec des problématiques similaires mais pour des navires. Ces nouvelles expériences ont conduit à une extension du modèle original en utilisant un score d'affinité pour fusionner les contextes entre eux.

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France