Automatic detection of business data anomalies with deep learning and application to the ADS-B protocol

Ralph Karam

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

RALPH KARAM

Automatic detection of business data anomalies with deep learning and application to the ADS-B protocol

Détection automatique d'anomalies de données métiers avec deep learning et application au protocole ADS-B

Thèse présentée et soutenue à Belfort, le 29 Septembre 2022

Composition du Jury :

CONTASSOT-VIVIER SYLVAIN	PR à l'Université de Lorraine	Président du jury
CHRÉTIEN STÉPHANE	PR à l'Université Lumière Lyon 2	Rapporteur
VERNIER FLAVIEN	MCF HDR - Université Savoie Mont Blanc	Rapporteur
COUTURIER RAPHAËL	PR à l'Université Bourgogne Franche-Comté	Examinateur
SALOMON MICHEL	MCF HDR - Univ. Bourgogne Franche-Comté	Directeur de thèse

N° | X | X | X |

ABSTRACT

Automatic detection of business data anomalies
with deep learning and application to the ADS-B protocol

Ralph Karam
University Bourgogne Franche-Comté, 2022

Supervisor: Michel Salomon

The use of Machine Learning (ML) and Deep Learning (DL) for security anomaly detection, malware analysis and pattern and signature recognition is an extremely active topic both at the research level and in the cybersecurity industry. Extraction of weak signals (rare or deviant element concerning behavioral motives) as well as the demonstration of correlation on the motives of cyberattack are the first sought-after applications of ML and DL techniques in this context. Thus, an FDIA attack on a Smart Grid may involve a fine modification of data from production nodes, while an attack on a defense air control system may involve a falsification of airspace runway data. The ADS-B protocol - Automatic Dependent Surveillance-Broadcast - is an air traffic control data source based on satellite positioning. Each aircraft (and potentially any other flying object, including a drone) periodically sends via messages its position and other information (identification, status of the aircraft) to ground stations and other ADS-B equipped aircraft operating within the reception area. The ADS-B protocol is becoming a mandatory surveillance technology all around the globe, nevertheless this protocol still lacks security measures such as message encryption and authentication mechanisms. One main approach to overcome these shortcomings is using ADS-B anomaly detection based on machine learning and deep learning methods. In the majority of applications, supervised anomaly detection is rarely used due to the lack of labeled anomalous data, despite its superior ability to detect anomalies compared to unsupervised methods. Supervised anomaly detection of ADS-B data was the main focus of this thesis due to its performance advantage. In order to obtain a sufficient amount of labeled anomalies and normal data, a false data generator based on a domain specific language was used. This generator was designed by colleagues at DISC in Besançon who are specialists in the generation of tests. To the best

of our knowledge, this thesis is the only work which used supervised anomaly detection of ADS-B messages using data obtained from a generator of FDIAs. Our approach gave very promising results in detecting various types of ADS-B attacks. The best performance was obtained using the Long Short-Term Memory (LSTM) architecture.

In addition, as a secondary work in the context of this thesis different deep learning and machine learning approaches were studied in order to forecast noise levels and detect punctual noise level anomalies. The data were gathered from an IoT system more specifically a network of smart parkmeters. From the results of our study it was deduced that such techniques, preferably a 1D Convolutional Long Short-Term Memory (CNN-LSTM), can be successfully used in environmental noise monitoring applications.

KEYWORDS: Deep learning, Cybersecurity, ADS-B protocol, Machine learning

RÉSUMÉ

Détection automatique d'anomalies de données métiers
avec deep learning et application au protocole ADS-B

Ralph Karam
Université Bourgogne Franche-Comté, 2022

Directeur de thèse : Michel Salomon

L'utilisation du Machine Learning (ML) et du Deep Learning (DL) pour la détection d'anomalies de sécurité, l'analyse des malwares et la reconnaissance de motifs et de signatures est un sujet extrêmement actif tant au niveau de la recherche que dans l'industrie de la cybersécurité. L'extraction de signaux faibles (élément rare ou déviant concernant les motifs comportementaux) ainsi que la démonstration de corrélation sur les motifs de cyberattaque sont les premières applications recherchées des techniques ML et DL dans ce contexte. Ainsi, une attaque de type injection de fausses données (FDIA - False Data Injection Attack) sur un Smart Grid peut impliquer une modification fine des données des noeuds de production, tandis qu'une attaque sur un système de contrôle aérien de défense peut impliquer une falsification des données relatives aux pistes de l'espace aérien. Le protocole ADS-B - Automatic Dependent Surveillance-Broadcast - est une source de données de contrôle du trafic aérien basée sur le positionnement par satellite. Chaque aéronef (et potentiellement tout autre objet volant, y compris un drone) envoie périodiquement par messages sa position et d'autres informations (identification, statut de l'aéronef) aux stations au sol et aux autres aéronefs équipés de l'ADS-B opérant dans la zone de réception. Le protocole ADS-B est en train de devenir une technologie de surveillance obligatoire dans le monde entier, néanmoins ce protocole manque encore de mesures de sécurité telles que le cryptage des messages et les mécanismes d'authentification. L'une des principales approches pour surmonter ces lacunes consiste à utiliser la détection des anomalies ADS-B basée sur des méthodes d'apprentissage automatique et d'apprentissage profond.

Dans la majorité des applications, la détection supervisée des anomalies est rarement

utilisée en raison du manque de données anormales étiquetées, malgré sa capacité supérieure à détecter les anomalies par rapport aux méthodes non supervisées. La détection supervisée d'anomalies dans les données ADS-B a été le point central de cette thèse en raison de son avantage en termes de performance. Afin d'obtenir une quantité suffisante d'anomalies étiquetées et de données normales, un générateur de fausses données basé sur un langage spécifique au domaine a été utilisé. Ce générateur a été conçu par des collègues de DISC à Besançon qui sont des spécialistes de la génération de tests. A notre connaissance, cette thèse est le seul travail qui a utilisé la détection supervisée d'anomalies de messages ADS-B en utilisant des données obtenues à partir d'un générateur de FDIAs. Notre approche a donné des résultats très prometteurs dans la détection de différents types d'attaques ADS-B. Les meilleures performances ont été obtenues en utilisant l'architecture Long Short-Term Memory (LSTM).

En outre, en tant que travail secondaire dans le cadre de cette thèse, différentes approches d'apprentissage profond et d'apprentissage automatique ont été étudiées afin de prévoir les niveaux de bruit et de détecter les anomalies ponctuelles de niveau de bruit. Les données ont été recueillies à partir d'un système IoT plus précisément un réseau de parcmètres intelligents. Les résultats de notre étude ont permis de déduire que de telles techniques, de préférence un hybride 1D combinant réseaux de neurones convolutionnels et Long Short-Term Memory (CNN-LSTM) peuvent être utilisées avec succès dans les applications de surveillance du bruit environnemental.

Mots clés: Apprentissage profond, Cybersécurité, Protocole ADS-B, Apprentissage automatique

ACKNOWLEDGEMENTS

First of all I would like to thank my PhD advisor, Michel Salomon, for his involved supervision, his precious advice and the confidence he gave me during these three years. I would also like to express gratitude to Raphaël Couturier for his valuable and useful advice he gave me during this journey. I sincerely thank Stéphane Chrétien and Flavien Vernier for accepting the role of rapporteurs for my thesis and the attention given to my work. I also thank Sylvain Contassot-Vivier for having done me the honor of chairing my thesis jury. My thanks also go to my two doctoral colleagues, Aymeric Cretin, and Antoine Chevrot who willingly accepted to collaborate with me in the context of the GeLead project which financed my PhD. My sincere appreciation and thanks to the members and professors of the team AND (Algorithmique Numerique Distribuée) for the positive work environment and for the pleasant moments that we have shared together. I would also like to thank my colleagues and friends: Hassan Noura, Joseph Azar, Anthony Nassar, Bernard Kodjo Agbemadon, and Medane Affo Tchakarom. On a personal level, I want to thank my father Antoine, my mother May, and my brother Roudy for their unconditional love and support, which underlies everything I do.

Contents

List of abbreviations

ACAS . . . . . Airborne Collision Avoidance System

ADS-B . . . . . Automatic Dependent Surveillance-Broadcast

ADS-C . . . . . Automatic Dependent Surveillance-Contract

ARIMA . . . . . Autoregressive Integrated Moving Average

ATC . . . . . Air Traffic Control

ATCS . . . . . Air Traffic Control Specialists

CNN . . . . . Convolutional Neural Network

COTS . . . . . Commercial-Off-The-Shell

DBSCAN . . . . Density-based Spatial Clustering of Applications with Noise

DL . . . . . Deep Learning

DSL . . . . . Domain-specific Language

FDIA . . . . . False Data Injection Attack

FDI-T . . . . . False Data Injection Test

FPR . . . . . False Positive Rate

GB . . . . . Gradient Boosting

GNSS . . . . . Global Navigation Satellites Systems

GPT . . . . . Generative Pre-trained Transformer

GRU . . . . . Gated Recurrent Unit

HDBSCAN . . . Hierarchical DBSCAN

ICA . . . . . Independent Component Analysis

ICAO . . . . . International Civil Aviation Organization

IForest . . . . . Isolation Forest

IoT . . . . . Internet of Things

IP . . . . . Internet Protocol

KNN . . . . . K-Nearest Neighbors

LOF . . . . . Local Outlier Factor

LSTM . . . . . Long Short-Term Memory

ML . . . . . Machine Learning

NM, nmi . . . . Nautical Mile

OPTICS . . . . . Ordering Points To Identify the Clustering Structure

PCA . . . . . Principal Component Analysis

RA . . . . . Resolution Advisories

RNN . . . . . Recurrent Neural Network

SDR . . . . . Software Defined Radio

SR . . . . . Spectral Residual

SSR . . . . . Secondary Surveillance Radar

SVDD . . . . . Support Vector Data Description

TDOA . . . . . Time Difference Of Arrival

TFT . . . . . Temporal Fusion Transformer

VAE . . . . . Variational AutoEncoder

VANet . . . . . Vehicular Ad-hoc Network

WASN . . . . . Wireless Acoustic Sensor Network

1

INTRODUCTION

This thesis focuses on supervised anomaly detection in business data, more specifically anomalous messages of the Automatic Dependent Surveillance-Broadcast (ADS-B) protocol, using deep learning techniques. The ADS-B data used in our study are downloaded from the OpenSky network. To obtain the anomalous data, the downloaded data are introduced into a false data generator called FDI-T - False Data Injection-Testing, which was developed by the VESONTIO team from the Department of Computer Science and Complex Systems (DISC in French) of FEMTO-ST Institute in Besançon in the context of the GeLeaD (Generate Learn and Detect) project. In order to detect anomalies in ADS-B messages, deep learning time series classification approaches were applied. Also, as a secondary contribution, IoT noise forecasting was also performed. The presented research was carried out in the AND team from DISC department, and was supported by the DGA (French defence procurement agency). More precisely it took place and was funded by the GeLeaD project related to the ANR ASTRID research program.

Note that, as part of the GeLeaD project, the false data generator FDI-T was developed in the context of Aymeric Cretin's - a PhD student colleague - PhD thesis. Conversely, in the context of the same project, Antoine Chevrot - another PhD student colleague - was working on unsupervised anomaly detection applied to the ADS-B protocol.

1.1/ IMPORTANCE OF BUSINESS DATA

Due to the widespread digitisation in companies, each of their services generates and uses data on a daily basis. The activity of an entity handling such data is clearly dependent on their quality and trustworthiness. Anomalous data can lead to minor inconveniences in the operation of systems, or even prevent them from functioning completely.

For example, in the aviation sector, aircraft communicate their information with each other and with ground stations using the ADS-B protocol. However, some malicious entities can eavesdrop on ADS-B messages due to their lack of encryption and inject falsified data

for multiple purposes such as false aircraft creation or enemy aircraft identity masking. Such attacks can cause various types of unwanted consequences like denial of service, the confusion of air traffic controllers or even facilitating the enemy aircraft's approach of national territory [66].

Another example where data integrity is crucial to the functioning of the system is the Internet of Things (IoT), a system interconnecting computing devices, digital machines, and objects through a network [115]. Such systems are typically modeled using graphs whose nodes correspond to the connected objects which are analogous to normal computers in the regular internet and the edges are the connections permitting data transfer between the nodes. Thanks to the IoT many innovative solutions have emerged in order to solve problems touching a wide range of domains. One of the most important applications of IoT is smart cities.

A smart city is a city with a technological infrastructure whose purpose is to optimize the operations of the city, manage the resources and provide the well being of its citizens. A smart city relies on three layers of abstraction for its operations. First, a perception layer collects the data using devices connected to the internet of things. Then data are transported in the network layer which comprises of the city telecommunication infrastructure. The third layer, which is the application layer, is responsible for data processing for decision making [57].

Two other notable IoT applications are smart grids and vehicular Ad-Hoc Networks.

A smart grid is an electrical grid made of smart components like smart meters, smart distribution boards and circuit breakers. In such grids, small renewable energy producers supply electrical energy. In addition, individual houses are not just passive consumers, but they can also contribute and supply power to the electrical grid. In smart grids if smart counters are directly controlled by hackers or by man-in-the-middle attacks between the counters and control centers, false data can be injected leading to erroneous estimation of power bills with intent of electrical energy theft. Also, such attacks can be realised with the aim of disrupting the power systems, i.e. leading to a decrease in energy production or even blackouts [93].

The Vehicular Ad-Hoc Network (VANet) is a network connecting vehicles with each other, using wireless technologies in order to have a safe and comfortable traffic. The VANet can also be prone to similar attacks as the two previously mentioned applications. For instance, if vehicles are directly controlled by hackers or through man-in-the-middle attacks, data can be falsified about the state of individual vehicles or their environment. Such attacks may be targeting specific vehicles to hurt particular passengers or they may happen on a larger scale to cause accidents, traffic jams and reduce the performance of the VANet particularly in resource management [79].

1.2/ DATA SECURITY ISSUES

Due to the prevalence and the necessity of data in all types of services, data security issues need to be examined carefully. Data can belong to a defined set of states. It can be created, stored, used, shared, archived and destroyed. When examining the security of data, each one of the previously mentioned states should be taken into consideration individually or in tandem [116]. The data creation stage is mostly susceptible to physical attacks. For example, sensors in sensor networks such as IoT systems are prone to different types of physical alterations like circuitry tampering, sensor reprogramming, and so on [47]. The sensors can also suffer from outages preventing data acquisition and malfunctioning which can threaten the correctness of the generated data [38].

The stored data is prone to many types of attacks like physical data tampering which threaten the integrity of the data regardless if it is encrypted or not. Also encrypted stored data (using full disk encryption schemes) when in use can still be accessed using side channel attacks such as cold booting. To prevent cold booting attacks one can shutdown the computer and wait for the random access memory to lose all its content. Another technique is encrypting the data when the computer is unattended [46].

On the other hand, data sharing is vulnerable to routing attacks like selective forwarding consisting in a node's dismissal of data transmission between nodes in IoT systems or any other type of sensor networks. One way of mitigating such attacks is encrypting the transferred data and applying traffic analysis [67]. Data sharing is also susceptible to spoofing attacks like IP address spoofing. Basically, in pursuance of masquerading as another computer or another connected object in the case of IoT systems, the attacker generates an IP packet and falsify its source IP address [31]. Traditionally, spoofing attacks are avoided using authentication mechanisms [40]. However, these mechanisms might not be feasible due to legacy and practicality reasons, as in the case of the ADS-B protocol in which messages are sent in the clear. The ADS-B protocol, its challenges and ways to address its vulnerabilities, mainly anomaly detection, will be detailed in the remainder of this thesis [58].

Sensor networks in IoT systems are also prone to sinkhole attacks which are a type of routing attacks in which a compromised node typically attracts the traffic of a special subset of a the IoT system by advertising a false optimal shortest path. Such attacks can be easily alleviated when the geographical position of the nodes is well-known [67]. Note that the sinkhole's incoming data is prone to alterations and corruption. Also, there exists many other types of routing attacks like sybil attacks, wormholes attacks and hello flood attacks [30].

As for the disposal of data, they can be recovered in some cases due to data persistence [11]. To prevent such a vulnerability, one can clear data (to prohibit recovery using

software), purge the data (to prohibit recovery using software as well as laboratory procedures) or even destroy the physical medium of storage (to prohibit any kind of recovery) depending on the level of sensitivity of the data [37].

1.3/ DEEP LEARNING FOR DETECTION OF ANOMALIES

Various techniques exist to prevent data integrity compromise like authentication and encryption mechanisms, validation checks, etc. Their feasibility is not always guaranteed or they may not be sufficient for total integrity. Therefore other techniques need to be available to detect potentially flawed data, like anomaly detection techniques.

Machine learning comes as viable approach for anomaly detection. Machine learning is a set of algorithms which learn to map input data to output data. In other words, a machine learning model automatically adapts its internal state based on input data and output data (supervised learning) or even input data solely (unsupervised learning) in order to obtain an appropriate input-output mapping. Machine learning covers many statistical use-cases such as regression, clustering, classification and anomaly detection.

Despite the widespread use of conventional machine learning methods, they usually need to rely on specialised domain knowledge to extract relevant features from the data needed for the duty at hand [85]. To ease the effects of this constraint, deep learning architectures can be used. They are models composed of multiple layers which automatically learn increasingly abstract representations depending on the depth of the architecture hence the name ”deep learning”. Such representations which replace the preprocessing step of feature extraction in traditional machine learning are required for classification, regression, anomaly detection and many more tasks.

Deep learning was the catalyst for many breakthroughs such as protein folding and drug discovery which has the potential to revolutionize the pharmaceutical and health care industry [153]. It has also been used to train intelligent agents for video games to be able to defeat world-class champions, such as Google DeepMind’s AlphaStar [124] for the real-time strategy game StarCraft. The latter application can be thought of as a first step towards general artificial intelligence. Deep learning gave also rise to big autoregressive language models which were trained on a giant corpus of text to be able to generate text in a similar way as humans [135]. In addition, language models were also trained on huge codebases like github to be able to code programs automatically [164]. Even though such automatic coding tools are still in their infancy they give very promising results. Deep learning is also very important for the development of some more specialised tools like face recognition, stock predictions and anomaly detection for medical imagery, etc.

This thesis focuses on machine learning and deep learning techniques for anomaly detection. For several years, anomaly detection was and still is a crucial research subject due to its multidisciplinary scope which touches various fields like cybersecurity, health care and finance. It is defined as the act of identifying data points which are remarkably different from the bulk of data samples [158]. The performance of deep learning for anomaly detection models usually surpasses traditional machine learning based techniques especially for more complex tasks. Deep learning for such use cases learn anomaly scores without the need for manual feature extraction. Some notable deep learning architectures are autoencoders and their variants which will be described in more detail in this thesis while focusing on their application on the ADS-B protocol. Since those anomaly detection techniques will be applied on sequences of ADS-B messages which are usually modeled as time series, architectures like recurrent neural networks especially Long Short-Term Memory networks are usually the basis of time series anomaly detection architectures. Such architectures contain memory mechanisms to capture time dependencies. Unsupervised and semi-supervised techniques like autoencoders cannot detect small attacks like small deviations in the trajectories associated to sequences of ADS-B messages. Therefore, in this thesis we propose the use of supervised deep learning for anomaly detection to try to remedy this problem. As a first contribution, different architectures for supervised anomaly detection were compared. As a second contribution, a strategy for supervised ADS-B anomaly detection was proposed which relies on a false data generator to obtain labeled anomalous data. Labeled anomalies are usually difficult to come by, hence the use of a false data generator which is based on a domain specific language. Finally, as a third but secondary contribution in this thesis, machine learning (gradient boosting methods) and deep learning techniques (recurrent neural networks, convolutional neural networks, transformers, etc) were applied to forecast the noise level gathered using IoT devices more specifically smart parkmeters. Also those forecasts were used to detect injected punctual noise level anomalies.

1.4/ OUTLINE

This thesis is divided into the following chapters:

- Chapter 2 - Case studies. This chapter introduces the Automatic Dependent Surveillance-Broadcast protocol. The benefits and challenges facing the protocol will also be described. In addition, a secondary case study, more specifically environmental noise monitoring using an acoustic sensor network will be presented.
- Chapter 3 - Related works. This chapter introduces anomaly detection techniques. First, for the purpose of anomaly detection in a general sense, machine learning

techniques will be introduced. Then deep learning anomaly detection techniques focused on time series data will be presented. Lastly, anomaly detection applied on the ADS-B protocol will also be examined.

- Chapter 4 - Contribution 1: a comparative study of deep learning architectures for detection of anomalous ADS-B messages. In this chapter, a performance comparison of different machine learning and deep learning architectures for the purpose of supervised ADS-B anomaly detection is carried out. This study focuses on punctual altitude anomalies.
- Chapter 5 - Contribution 2: supervised ADS-B anomaly detection using a false data generator. In this chapter, a new strategy for ADS-B supervised anomaly detection is devised. The detection of different types of attacks is studied: gradual attacks and waypoints attacks.
- Chapter 6 - Contribution 3: deep learning and gradient boosting for urban environmental noise monitoring in smart cities. In this chapter, different machine learning and deep learning architectures are tested for the purpose of noise level forecasting. Also anomaly detection of punctual noise anomalies was carried out.
- Chapter 7 - Conclusion and perspectives.

2

CASE STUDIES

2.1/ AUTOMATIC DEPENDENT SURVEILLANCE-BROADCAST

Airspace surveillance is a critical component of air traffic services. Flight crews as well as air traffic controllers need to have airspace awareness for the safe aircraft maneuvering during the various flight stages while avoiding collisions like taking off, cruising, landing, taxiing and parking the aircraft. Such awareness is obtained using the many existing surveillance technologies such as primary radars and secondary surveillance technologies like Mode-A, Mode-C and Mode-S which will be discussed in this section. The ADS-B protocol, which is part of Mode-S will, be explained more thoroughly, including its mechanism, its benefits, the challenges it faces as well as its vulnerabilities and how they can be exploited.

2.1.1/ PRIMARY RADAR

The idea behind radar (Radio Detection And Ranging) is a radio wave death ray that was proposed by British military officials worried about being outpaced by the Germans during the second world war. The fact that this weapon was not feasible at the time, but that its main approach could be used for aircraft tracking, paved the way for the radar concept [160]. This technology uses a rotative omnidirectional antenna to send radio waves which are reflected by targets such as aircraft and recaptured by the antenna as seen in Figure 2.1. The distance of the target is a function of the duration of the round trip of the signals, whereas the rotation angle of the radar dictates the azimuth angle of the aircraft. To distinguish between moving targets as opposed to static ones, physical phenomena such as the doppler effect can be relied upon. Note that the primary radar used alone is neither capable to give the heights of the targets nor their identity, implying the need for a supplementary technology such as secondary radars.

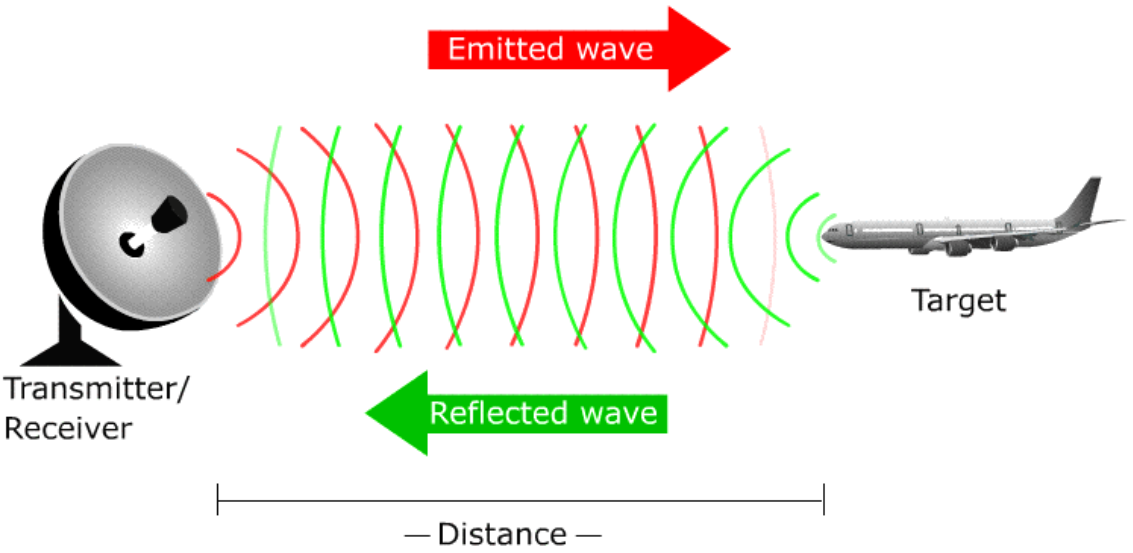


Figure 2.1: The radar mechanism showing emitted and reflected waves whose round trip duration is used for distance estimation (illustration from skyradar.com).

2.1.2/ SECONDARY RADAR

The Secondary Surveillance Radar (SSR) is a cooperative technology based on interrogation that helps ATCS obtain more detailed flight data relative to radar. The SSR which can be added on the primary radar or at another placement sends interrogation pulsations at 1030 MHz to an aircraft and then receives a reply at 1090 MHz as shown in Figure 2.2.

Originally, there were two types of protocols used by SSR: Mode-A which communicates the squawk code (used for aircraft identification) and Mode-C which communicates the altitude. When a SSR interrogates for the squawk/altitude using Mode A/Mode C signals, the reply type is recognised solely on synchronization. This makes the SSR less reliable for an airspace with high aircraft density because all reply signals are at the same frequency of 1090 Hz. In addition, SSR are prone to receiving corrupted replies from multiple airplanes lying in the same direction relative to the SSR due to interference.

To remedy such problems facing Mode A and Mode C, other protocols were developed such as Mode S. In this mode, different aircraft can be chosen for interrogation to obtain different types of information. Uplink signals contain data blocks made of 56 bits in case of short interrogation, compared to 112 bits in case of long interrogation. Equivalently a downlink signal's data block is made of 56 bit (short reply) or 112 bits (long reply). The first five bits of the data block for uplink/downlink signals, called uplink format/ downlink format, are used to indicate the structure of the information present in the data block. A downlink format of 17, 18 or 19 infers the automatic broadcast of messages via extended squitter mainly used for the ADS-B protocol and which will be detailed in the next subsection.

Figure 2.2: Secondary surveillance radar relying on interrogation and replies for communication (illustration from skybrary.aero).

2.1.3/ What is ADS-B?

ADS-B is a communication protocol governing the exchange of flight information such as flight speed and location between airplanes and air traffic control (ATC) centers [166]. An Aircraft obtains its own position and speed via global navigation satellites systems (GNSS) and then broadcasts them along with other information including the emergency status and flight number in the form of ADS-B out signals. These signals, which are Mode S extended squitter (1090 MHz), can be received by aircraft and ATC as ADS-B In signals. The flow of information in the ADS-B protocol is shown in Figure 2.3. ADS-B stands for Automatic Dependent Surveillance-Broadcast where ”Automatic” implies the transmission of aircraft information without any human intervention, ”Dependent” infers that it requires equipment on the sending and receiving end of the ADS-B signals and ”Broadcast” indicates that each aircraft equipped with ADS-B out broadcasts its information (identification, position, and so on).

2.1.4/ Message format

ADS-B frames shown in Figure 2.4 contain the following fields in order: first the downlink format made of 5 bits, then the capability made of 3 bits, the ICAO made of 24 bits, the message extended squitter made of 56 bits, and the parity made of 24 bits. All the fields add up to 112 bits. Every ADS-B frame has a downlink format of 17 or 18. When the downlink format is 18, the emitter of the frame is not a transponder. The capability specifies the Mode S transponder level. A level 1 transponder corresponds to a basic transponder without any datalink capability and hence cannot be used for international travel. Conversely, levels 2, 3 and 4 have increasing datalink capabilities and can be used for international travel [2].

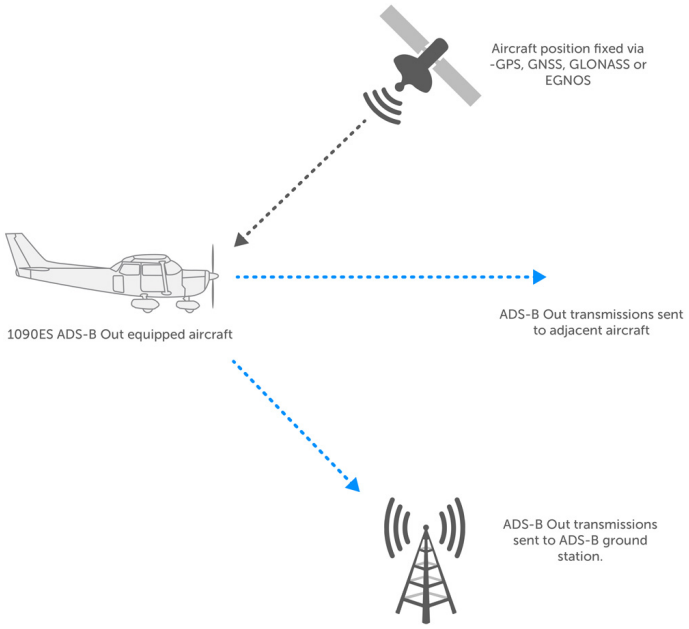


Figure 2.3: The ADS-B protocol communication principle (illustration from trig-avionics.com).

The ICAO is the mode-S transponder code of the emitting aircraft and is designated on the authority of the International Civil Aviation Organization guidelines (ICAO). The ICAO helps identifying an aircraft.

The message extended squitter contains the payload whose first 5 bits called type code is used to specify the content of the ADS-B data frame in the remaining 51 bits. ADS-B messages can take on many types.

First, an identification message can be used to obtain the callsign of an aircraft. A callsign specifies the route of the aircraft. Since different aircraft can take the same route, the callsign does not necessarily identify an aircraft. This type of messages also indicates the wake vortex category of the aircraft. The wake vortex categorization is a way to classify moving objects in the air according to the way they disturb the atmosphere around them. Therefore according to this type of classification an object can be a skydiver, a rotorcraft, a light aircraft, a medium aircraft, a heavy aircraft, etc.

Second, a position message which takes one of two subtypes, surface or airborne, can be used to compute the altitude, latitude and longitude of the aircraft. For the airborne

Downlink Format	Capability	ICAO	Message Extended Squitter	Parity

Figure 2.4: ADS-B frame format.

position case, the latitude and longitude are computed from their encoded values in two successive pair of odd and even messages (this parity is determined by the bit 54). Another method to obtain the actual latitude and longitude is based on a close reference position within a 180 NM range. The encoded latitude and longitude of the message, together with their reference values are used to compute the actual position. In order to obtain the height, only one message is needed. If the type code is between 20 and 22, the height can be directly decoded in meters since it is already computed using GNSS. However, when the type code is between 9 and 18, a barometric height is computed in feet.

For the surface position case, the position is computed in a similar manner as an airborne one using pairs of even and odd messages. However, this technique gives multiple surface positions solutions. So to choose the right solution, a reference position (usually a nearby airport) is required. Note that a surface position can also be computed solely from a reference position and one message. Also from surface position messages, ground speed and ground track can be decoded from a movement field and ground track field respectively.

Third, from a velocity message, vertical rate, ground velocity, ground track, air speed (if ground velocity is not available), and the magnetic heading can be easily computed as well as the difference between GNSS height and barometric height.

Fourth, operation status message is used to give miscellaneous information on an aircraft such as integrity indicators, accuracy and version of the ADS-B messages.

Fifth, aircraft status messages' role is to report resolution advisories (RAs) produced by Airborne Collision Avoidance System (ACAS) [5]. Essentially, ACAS is as system whose purpose is to prevent mid-air collisons. ACAS rely on a secondary radar to interrogate aircraft with mode C or mode S transponders. The replies are used to obtain the relative position of the aircraft and then the flight crew is alerted in the case of a collison threat. That being so, resolution advisories tell the pilot the needed vertical speed to obtain a satisfactory vertical separation [1].

Target state and state information messages give aircraft state and status information such as the next intended altitude and track as well as some other complementary information [5].

2.1.5/ BENEFITS OF ADS-B

The invention of ADS-B brought many improvements in the aircraft surveillance domain. First, the information transmission rate has increased significantly. In fact, radar information is updated every 6 to 12 seconds which is much slower than the minimum trans-

mission rate in ADS-B of 1 message every second. The ADS-B protocol facilitates cooperation by creating a common airspace picture shared between aircraft and ground stations. Aircraft surveillance based on this technology can cover remote areas inaccessible by radar. For example, all of Australia has aircraft surveillance coverage with ADS-B, whereas prior to the introduction of this protocol, much of the Australian outback had no surveillance coverage due to a lack of radars (20 radars across Australia) [165] as seen in Figure 2.5 . ADS-B is also responsible for monitoring airborne and ground traffic. Hence, it helps preventing the ill placement of a person, vehicle and aircraft on runways which can cause serious accidents. The better accuracy of ADS-B data allows for a smaller separation between aircraft, increasing henceforth the airspace capacity and the flexibility to take more efficient routes. Essentially, aircraft can be laterally and longitudinally separated by only a distance of 20 nmi (Nautical mile) and 5 nmi, instead of the respective separation distances of 90 nmi and 80 nmi required before the introduction of ADS-B [78]. This can substantially diminish flight costs, pollution and noise.

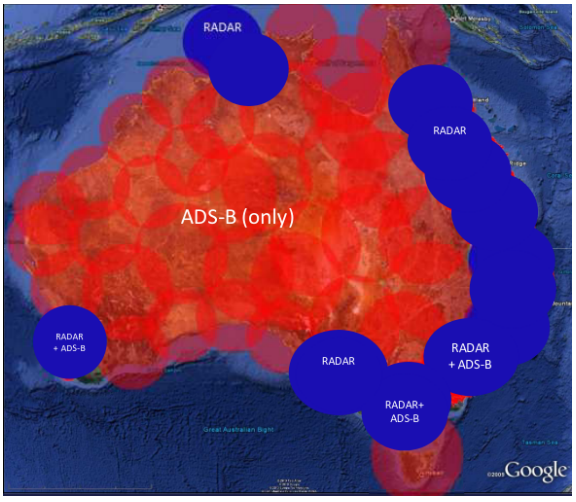


Figure 2.5: The superior surveillance coverage provided by the ADS-B protocol in Australia.

2.1.6/ CHALLENGES FACING ADS-B

2.1.6.1/ FEASIBILITY OF ATTACKS

Even though ADS-B is a practical communication medium it still suffers from many vulnerabilities. Basically, each message is neither encrypted nor has any authentication or challenge-response mechanism and is therefore subject to attacks such as eavesdropping and alteration by unauthorized ADS-B equipment [58].

The authors in [58] tested the feasibility and practicality of such attacks in a controlled setting. They used a software defined radio (SDR), specifically a USRP1, to send ADS-B

messages and relied mainly on a PlaneGadget ADS-B Virtual Radar to receive these messages. In order to create an alteration attack, more specifically impersonation attacks, they used MatLab to encode and modulate ADS-B messages, while the replay attacks were produced using GNURadio's predefined functions. GNU radio is a free and open source application programming interface used for signal processing and other functionalities required for SDRs. Note that the PlaneGadget is a commercial-off-the-shelf (COTS) receiver whose value of money is adequate and can be easily bought by amateurs. The ease in acquiring the material and their deployment showed that in practice, ADS-B attacks can indeed be easily executed.

In order to assess the probability of those attacks, the motivations for such risks need to be known. They can range from financial and recognition incentives to terror, spying from intelligence agencies or even curious individuals [58]. For example, an aviation enthusiast teen was relying on ADS-B data to track celebrities' private jets such as Elon Musk's jet, Bill Gates, Jeff Bezos and others. Then he used bots to publish those aircraft locations in real time on Twitter. Elon Musk later proposed the teen 5000 dollars to take down these bots because he felt that they constitute a security threat to him. The teen refused to do so and demanded more money or an internship but his counter-offer was declined [6]. Hence one can argue that such motives could help in the evaluation of the legitimacy of ADS-B messages. Therefore, using exogenous data expressing attack motives might aid in detecting ADS-B attacks.

2.1.6.2/ VULNERABILITIES AND ATTACKS

In addition to the previously mentioned vulnerabilities, the ADS-B protocol suffers from message loss. Such omissions can arise naturally from message collision, which is mostly present in dense airspace. They can also originate from **Ground Station Flooding** attacks which consist in jamming the 1090 MHz channel causing message loss. Note that the natural loss is high near receivers (doughnut effect). Also, ADS-B messages are usually emitted by two alternating antennas on an aircraft, giving them different transmission properties. Thus additional data processing is often needed, more specifically filtering of ADS-B data. Moreover, messages can be duplicated due to echoes especially in alpine regions as well as due to some faulty transponders. Such transponders can send the same message up to 12 times [77].

Furthermore, many types of attacks other than the ones tested in [58] can be applied. For example, the **Ghost Aircraft Flooding** attack consists in sending a substantial amount of ADS-B messages identifying nonexistent aircraft to confuse ATCS and flight crews as seen in Figure 2.6. Essentially, they will observe a massive amount of fake aircraft identical to real ones on their instruments. Hence, a denial of service is inflicted. It is not

mandatory to flood the 1090 MHz channel with ghost aircraft to confuse ATCS and pilots. In fact, one or few ghost aircraft can be enough to do so. Basically, one can inject an ADS-B message inferring the presence of an aircraft at an undesirable position forcing a specific maneuver from the pilot which can be dangerous. This type of attack is called a **Ghost Aircraft Injection**. One can also delete ADS-B messages to hide a certain aircraft from the receivers. Such an attack is termed **Aircraft Disappearance**. **Virtual Trajectory Modification** is another type of attack in which the position fields in the ADS-B messages are modified leading to a perception of trajectory modification from the receiver. Note that it is also possible to alter the emergency field in the ADS-B messages leading to a **False Alarm Attack**. The ICAO field in ADS-B messages can also be modified for aircraft impersonation also known as **Aircraft Spoofing** [77].

(a) Before the attack (b) After the attack

Figure 2.6: Simulated Ghost Aircraft Flooding attack showing the appearance of multiple ghost aircraft to confuse air traffic controllers and pilots.

2.2/ ENVIRONMENTAL NOISE MONITORING WITH PARKMETERS

In the last decades, cities and metropolises kept attracting an ever increasing population looking for job opportunities and services. This kind of urbanization is not without its pitfalls such as traffic jams, air pollution and environmental noise. Noise pollution is a major nuisance which can be detrimental to people's health. It can cause sleep disruptions, cardiovascular disturbances as well as hearing problems such as hearing loss, hearing distortion, hearing intolerance (Hyperacusis) and transitory or lasting tinnitus. Such hearing impairments can hamper the patients careers as well as socially isolate them and contribute to mental illnesses like depression, anxiety and psychosis [101].

Traditionally, to tackle the problem of environmental noise, experts measure noise levels using sound level meters. Although these measurements are applied at specific sites and duration, they are based on conventional acoustics models and are usually insufficient for the escalating urbanization [123]. Nonetheless, wireless acoustic sensor networks (WASNs) help remedy these shortcomings. Such sensor networks were made possible

by the progress touching the Internet of Things (IoT). This evolution is linked to the decrease in the size and cost of the equipment and its increasing availability. WASNs have many capabilities like integrating noise levels into maps which are delivered to cloud-based central servers. This is done in order to dynamically model the noise level distribution in crowded areas. Note that, typically, such maps were static and obtained every five years from professionals' measurements which is substantially longer than the dynamic update using WASNs (in the order of minutes or even seconds).

2.2.1/ WASNs USING COMMERCIAL SOUND LEVEL METERS

WASNs can rely on commercial acoustic measurement devices like the following projects. In one of the earliest developments in motes, the authors created a sensor unit that optimizes the energy usage so that it can be used for IoT scientific experiments [34]. While in [44], the use of WASNs was shown to be a viable option to monitor urban environmental noise. Also, communication savings were sought by relying on predictions advising the right times to stop communicating data with the server. In [51], the authors implemented WASNs in housing and industrial settings where global synchronisation of a tree like network is obtained [56].

In [68], one main use case of noise data is environmental analysis. In fact the authors measured noise originating from traffic in Xiamen (China) using a WASN. This network comprises of noise devices and works based on the Zigbee and GPRS protocols. Another comparable project was accomplished in Barcelona. Essentially, in efforts to remedy the problem of increasing urban noise, officials decided on devising actionable strategies. However, the enforcement of such plans require the appraisal of their effects. Thus, a noise monitoring network was implemented in the city together with a platform for sensor data handling and interfacing called Sentilo [82]. A somewhat similar initiative was carried out in Paris by the non-profit institution Bruitparif. Basically, it consists in a WASN built with the purpose of gathering noise data for analysis and assessment. Such interpretations are used to execute informed actions to shrink noise levels. Correspondingly, a platform called RUMEUR (Urban Network of Measurement of the Sound Environment of Regional Use) is built to examine the collected data and communicate them with the general public [87].

FI-Sonic is another initiative concerned with noise related applications like updating noise maps, locating and isolating sets of sound sources. These capabilities are made available by the project's development of the required technology for environmental noise collection and manipulation [95]. FI-Sonic is powered by FIWARE (https://www.fiware.org/), a platform of open source solutions, ready to use technologies, support and consultancy services [88].

2.2.2/ WASNs USING AD-HOC HARDWARE

WASNs can also be based on Customized Nodes which are designed exactly for their needed purpose. In [42], as part of the SensorNet proposal, the authors study the feasibility of WASNs and build an ad-hoc noise measuring device to miminize the computation and power consumption of motes. In [60] and [109], the SENSEable initiative in Pisa (Italy) ensured the engagement of cities' inhabitants in real time noise administration using sensors at their residences. Similarly, to allow for noise level comparisons before and after enforcing noise minimizing policies, low cost WASNs with real time tracking capabilities were implemented in Monza (Italy) [99]. In New York as well, a noise sensor network called SONYC helps conceiving plans for urban noise reduction [104].

In order to devise a technique for creating reasonable noise maps in France, a cheap sensors' network was developed in the context of the CENSE project (characterization of urban sound environments) [36]. In Belgium, noise and air quality characterized by harmful gases concentrations as well as other environmental and meteorological metrics are monitored in cities via WSNs as part of the IDEA (Intelligent Distributed Environmental Assessment) initiative [53], [71]. In the same way air, quality and noise levels are also measured using sensor networks in London and CANADA in the context of The MESSAGE [62] and the UrbanSense [96] projects respectively.

Medusa, which is the continuation of RUMEUR, uses special equipment which combines cameras with multiple noise sensors in order to localize the origin of the noise [117]. In another project, special types of settings like freeways (more specifically the National Highway of Burdwan) were the sole targets of noise collection and analysis via WASNs. Bodily ailments (loss of hearing, hypertension, etc) were also diagnosed to study their correlation with the noise levels [55].

In other works, sounds were categorized in relation to sleep quality. This classification was based on perception surveys and acquired noise levels and other complementary sound data from WASNs [70]. Sound events were recognized as well via an identification process and correlated with the perception surveys [111].

Finally, the DYNAMAP project attempted gauging the noise impact of traffic on indivuals health [97], [103]. In the context of this initiative, WASNs were implemented in ROME [90] and MILAN (shown in Figure 2.7) [110] with the intention of having accurate measurements while minimizing expenditures.

2.2.3/ WASN FALSE DATA INJECTIONS

Previously, the security of ADS-B data and ways to compromise the protocol, mainly via false data injection attacks, have been described. In the same manner IoT systems

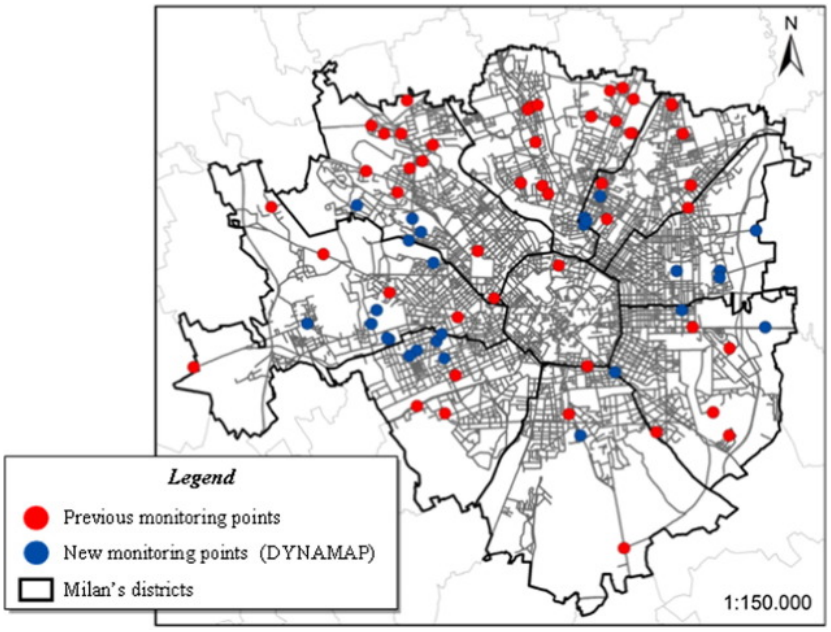


Figure 2.7: Noise monitoring positions in the city of Milan, in the context of the DYNAMAP project which aims at creating dynamic noise maps of urban areas.

are also prone to false data injection attacks. IoT systems are diverse and are rarely standardized; they are usually handled on a case-by-case basis. In other words, each type of IoT system relies on its own data structure. Thus, how to mitigate FDIA attacks depends on the specific IoT system.

Our colleagues, who are part of the GeLeaD project, tested the ability of IoT systems to withstand FDIA attacks, regardless of the type of data they rely on [134, 150]. To do so, they created a FDIA framework and a DSL language to constitute its core. Data is introduced in the framework in the form of CSV or JSON files. In a similar way as FDI-T, which is used in the ADS-B domain, the framework generating FDIA on IoT also relies on attack scenarios. These attack scenarios are creation, alteration, copy and deletion:

- In creation scenarios (using the keyword **Create** in DSL), new data records are created depending on the content of the DSL instruction.
- In alteration scenarios (using the keyword **Alter** in DSL), data records are modified according to the associated DSL instruction.
- In a copy scenario (using the keyword **Copy** in DSL), based on a selection condition data records are chosen to be duplicated and then altered according to an alteration scenario which is part of the copy scenario.
- In deletion scenarios (using the keyword **Delete** in DSL), data from records are conditionally chosen to be deleted.

For the sake of simplicity and in order to show the scope of capabality of the framework, we will discuss a general alteration scenario written in DSL by the authors instead of detailing the formalism of their DSL language. The alteration scenario is as follows:

```
scenario "IncrementationAndAttenuation"
ticker 2
geolocation (47.213865,5.968195)
alter things where location isInsideCircle(47.213865,
    5.968195,500) set particles+=(0.0->99999.0,10.0)
    with attenuation of 10.0 from 0 to 999999999;
```

The scenario is named "IncrementationAndAttenuation" using the scenario keyword. The term ticker expresses the time period between two messages sent by the system under test, whereas geolocation represents the latitude and longitude of the attack. The alteration criteria applied can be briefly described as follows. Particle values should be gradually incremented with a step of 10 between each message. This attack is applied inside a circle whose center has a latitude of 47.21 degrees, a longitude of 5.96 degrees and a radius 500 meters. Then, while moving away, the attack on particles is attenuated by 10 for each meter of distance.

The DSL is not bound to attacks on particles' data, it can also be used on any data field including noise level data which was also available in their acquired sensor data. Those sensors are part of a network of smart parkmeters recording various types of data. Note that in our work in chapter 6 we relied on noise level data from these parkmeters. One can argue that such a FDIA generator could be used to obtain false labeled data to train a false data injection detector similar to our work in detecting ADS-B anomalies in chapter 5. Nevertheless, we did not have access to this framework at the time of our contribution.

2.3/ DISCUSSION

Since the ADS-B protocol is mandatory in the majority of countries, many efforts were made towards ensuring its safety despite its major vulnerabilities such as the lack of authentication and encryption.

For example, the automatic dependent surveillance contract (ADS-C) can be used to ensure the legitimacy of transmitted information between aircraft and ground stations, since it relies on a contract between the communicating parties. This agreement specifies the type of information that should be transmitted to the ground stations, as well as the conditions triggering the communication of those information. However aircraft that support the ADS-C protocol cannot periodically transmit their information, which are rather sent upon request. Such a protocol contradicts the ADS-B paradigm and therefore airspace

systems no longer consider its implementation [76].

Other techniques to verify the correctness of ADS-B messages are secure location verification methods that ensure the validity of ADS-B messages by computing position information by other means. One such technique is multilateration in which the time difference of arrival (TDOA) of messages emanating from the same aircaft and received by four or more ground stations are computed. These TDOAs are used to compute the actual position of the aircraft. The geometrical concept behind multilateration is that only four ground receivers with positions and distances to the aircraft are needed to compute the space-time position of the aircraft in question. Despite the relative success of multilateration, it is costly and logistically difficult to implement due to the need of multiple ground receivers. Also, multilateration can suffer from multi-path propagation effects.

There is also Secure Broadcast Authentication techniques which include fingerprinting methods. One such technique is radiometric fingerprinting which identify the true emitter of a radio signal such as ADS-B messages based on their unique fingerprint. For instance turn-on transients of signals (the transitory portion of a signal right after turning it on) contain features which define a fingerprint. Note that if the exact hardware generating the radio signal is replicated, the consequent fingerprint is cloned as a result, rendering the technique unusable for authentication in such cases [28].

As for cryptography approaches, symmetric encryption is not appropriate for the ADS-B protocol due to the inherent threat of symmetric key leaks, therefore asymmetric cryptography can be more adapted to such a problem [78]. Regardless, cryptography schemes can overwhelm the 1090 MHz frequency channel used for ADS-B which can potentially stifle the proper operation of the protocol. More specifically, exchanging keys in a decentralized system such as the ADS-B infrastructure may require much more additional and bigger messages. In addition, encrypting messages contradict the original philosophy behind ADS-B which is open communication. The previously mentioned limitations, notably key management and the 1090 MHz channel risk of saturation, hindered efforts to implement cryptography to ADS-B.

In the context of IoT systems, smart cities are increasingly integrating WASNs into their infrastructure. However, environmental noise monitoring is still not as reliable, dynamic and prevalent as needed to be. Also, implementing, evaluating and testing WASNs in particular settings is usually an end in itself rather than to monitor and control smart cities. Some WASNs depend on a high number of nodes (for instance 112 nodes in [113]) which implies a wider scope of noise monitoring if the nodes are spaced enough. Other WASNs rely on a few nodes (for instance 4 nodes [120]). Even though the number of nodes influence the perception range of the WASNs, the use of a smaller number of nodes does not eliminate the ability of long-term noise monitoring [120]. Such a capability depends instead on the nodes capacity to function properly and gather clean reliable noise data

without experiencing crashes or power outages. Therefore, special care should be taken while selecting the sensors as well as the auxiliary hardware. In addition, since the data acquisition, preprocessing and treatment needs to be realised in a timely manner, the computational performance of the hardware and algorithms used by the WASNs should also be taken into consideration. Moreover, the choice of topology of WASNs is important to minimize the latency of noise data transmission which also depends on the network technology [64] (4G, 5G, WiFi, ...). Another aspect worth mentioning is the type of noise data acquired by WASNs. It can originate from different sources or be filtered to originate from an individual source such as the DYNAMAP project which restrain gathered noise data to road traffic noise [110].

2.4/ CONCLUSION

In this chapter, the ADS-B protocol was presented including its mode of operation, benefits and challenges, notably its lack of encryption and authentication mechanisms. Such vulnerabilities make the ADS-B protocol prone to many types of attacks which were also described in detail. Many techniques were devised to try to remedy this inherent flaw of ADS-B like multilateration, fingerprinting, and encryption schemes, but they possess some drawbacks preventing their real life implementation. In order to overcome such limitations, many authors proposed anomaly detection techniques [125] based solely on the content of the ADS-B messages. Such methods will be described in detail in the following chapter.

In addition, IoT applications to urban noise monitoring were also presented. They are divided into two categories: sensor networks which rely on commercial sound meters and sensor networks which depend on ad-hoc hardware. A false IoT data generation framework was also examined. It can be used on various types of data including noise data. Note that a similar false data generator based on a domain specific language used to generate ADS-B attack scenarios will also be detailed in the chapter 2. Finally, the factors on which the acoustic sensor networks depend to properly function were also discussed.

3

Related works

3.1/ Introduction

Anomaly detection is the act of identifying unexpected patterns in the data called anomalies or outliers. Anomaly detection is especially vital for decision-making processes that must distinguish between normal and abnormal situations. Therefore, many domains rely on anomaly detection like cybersecurity (intrusion detection), defence (enemy identification), services (fraud detection) and health care (medical diagnosis) [45]. Even though anomaly detection has a widespread range of applications, it still faces many challenges. Typically outlier detection approaches create a boundary for normal data and when a new occurrence lies outside this boundary it is considered as an outlier. However, this boundary is often fuzzy, making its identification difficult. Similarly, anomalies that are the result of malicious attacks are usually readjusted to remain undetected. Finally, it is often very difficult to acquire labeled anomalies due to the low frequency of anomalies relative to normal data and the costly process of labeling the data. In this chapter, general anomaly detection techniques will be discussed followed by anomaly detection approaches used in time series and ADS-B data.

3.2/ Anomaly detection

3.2.1/ Nearest neighbours based methods

Nearest neighbours based methods are a class of distance based anomaly detection methods in which dense neighbourhoods contain normal data whereas anomalous data are distant from their most adjacent neighbours [45]. The distance used for identifying neighbours for continuous features is usually the euclidean distance, but it is not required to be so. As for features used for classification, they usually rely on simple matching coefficients. On the other hand, multivariate data aggregate similarities used for each features

into a global similarity. Nearest neighbours based methods rely either on distance measures of k-Nearest Neighbours (KNN) or density measures like the local outlier factor for anomaly detection. These techniques will be shortly examined in this subsection

3.2.1.1/ ANOMALY DETECTION WITH KNN

The basic k-Nearest neighbours algorithm is a supervised model used for classification or regression. Suppose a point is to be classified, the classes of its k nearest neighbours are counted, the studied point is then classified according to the majority class [9]. Usually, the euclidean distance is the metric used to identify the k-nearest points, while in the case of regression, the output associated with the studied point is the mean of the outputs of the neighbours. The main disadvantage of the KNN algorithm is its high time complexity for high dimensional data.

The KNN approach can also be used in an unsupervised manner for anomaly detection [26]. In this case, if n outliers are searched, then these are the n points that have the greatest nearest neighbour distances.

3.2.1.2/ LOF: IDENTIFYING DENSITY-BASED LOCAL OUTLIERS

The Local Outlier Factor (LOF) is another nearest neighbours method. It is defined as a metric used to characterize the amount at which a data point is considered as an outlier [24], as can be seen in Figure 3.1. This definition contradicts with the binary classification of outliers in previous works and introduces the outlying behavior as a fuzzy property. Intuitively, the outlying behaviour of a point increases when its local density decreases whereas the density of its k nearest neighbours (k is a hyperparameter to be fixed) increases. Thus, the LOF is defined as the mean of the sum of the ratios of the two previous densities. Therefore, given a point P, $LOF(P) > 1$ implies P is an outlier.

3.2.2/ CLUSTERING-BASED ANOMALY DETECTION TECHNIQUES

In addition to nearest neighbours based methods, clustering based anomaly detection techniques are a class of distance based methods in which anomalous data do not reside in any cluster because clusters are solely composed of normal data as seen in Figure 3.2. Some notable algorithms relying on the previously mentioned assumption are DBSCAN, OPTICS and HDBSCAN. Clustering relies on similarity measures whose magnitude increases as distances dwindle. DBSCAN, OPTICS and HDBSCAN will be briefly explained in this subsection.

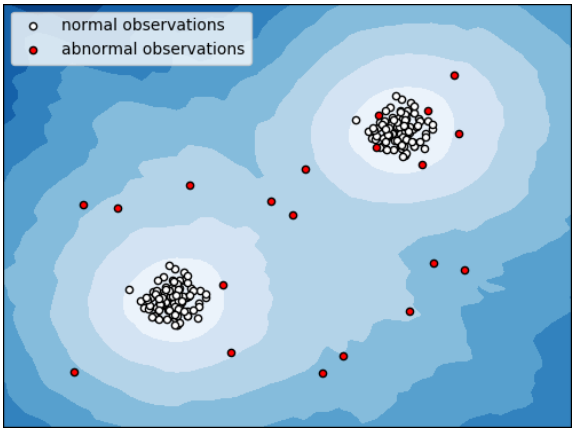


Figure 3.1: Example of a decision boundary plot of the Local Outlier factor used to estimate the outlying behaviour. The deeper the shade of blue, the higher the LOF and consequently the higher the outlying behaviour estimation (plot from scikit-learn.org).

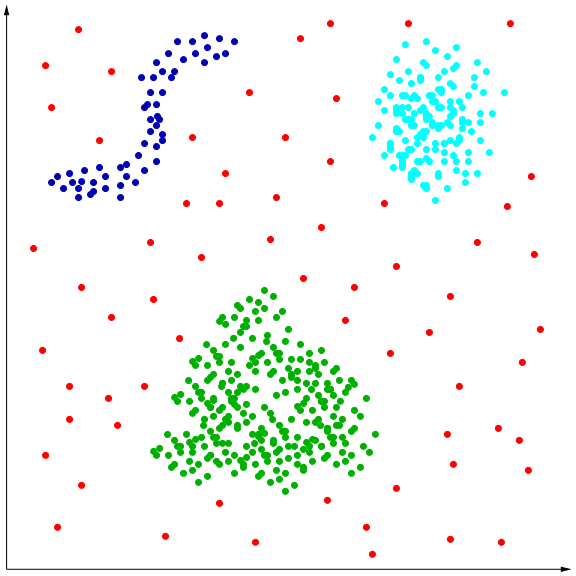


Figure 3.2: Clustering-based anomaly detection example showing normal data located in clusters (green, blue and cyan clusters) and anomalies (red points) not belonging to any cluster.

3.2.2.1/ DBSCAN: A DENSITY-BASED ALGORITHM FOR DISCOVERING CLUSTERS IN LARGE SPATIAL DATABASES WITH NOISE

In [16], while relying on a density measure, data points are partitioned into multiple clusters leaving out noise points or outliers. The density around a point P in this context is conveyed by its number of neighbours which are points located inside a hypersphere centered in P with a predefined radius ϵ. When the density around a point is above a threshold $minPts$, it is considered as a core point. So, for clustering, each connected component made of core points and their neighbours are allocated to a cluster. The remaining non-core points are considered as outliers.

3.2.2.2/ OPTICS: ORDERING POINTS TO IDENTIFY THE CLUSTERING STRUCTURE

In [21], the authors try to remedy the problem of clustering with heterogeneous densities using DBSCAN. The OPTICS algorithm is similar to DBSCAN but it outputs an ordered list of points according to their reachability distance. Given two points A and B, the reachability distance between these two points is defined as the maximum between the $minPts - distance$ and the ordinary distance as long as the size of B's neighborhood is bigger than $minPts$. This type of distance gives a smoothing effect that forces small distances to be bigger than a minimum equal to the $minPts - distance$. A low reachability distance between two points P and Q implies P and Q belong to the same cluster, whereas a high reachability distance indicates outlying behaviour.

3.2.2.3/ HDBSCAN: DENSITY-BASED CLUSTERING USING HIERARCHICAL DENSITY ESTIMATES

In [63], hierachical clustering is combined with density based clustering. Basically, this technique creates a hierachical clustering graph where each level (i.e. ϵ value) corresponds to a DBSCAN clustering solution. Then, the most eminent clusters in the hierarchy are chosen. The remaining points are considered as outliers. The prominence of the extracted clusters is expressed by a variable called stability. This measure represents how long a cluster survives in the hierarchy before being split into two clusters whose individual sizes are greater than a specified minimum cluster size. The advantage of HDBSCAN over DBSCAN is that it focuses on clustering high-density data, which prevents the extraction of small clusters made of outliers.

3.2.3/ ENSEMBLE-BASED MODELS

In addition to clustering techniques, ensemble methods are also used for anomaly detection. Such techniques combine the results (averaging, voting,...) of different anomaly detection models to obtain one global result.

Clustering techniques can be model centered or data centered. Model centered models combine the results of different models which learn on the same part of the data whereas data centered models learn on different parts of the dataset to detect anomalies in different regions of the data [61].

Ensemble techniques can also be categorized as sequential models or independent models. In sequential models such as boosting techniques, the outputs of one model are considered as the inputs of the following model. Usually sequential models are used to refine models results instead of being standalone anomaly detection techniques. Another

approach is combining the results of different independent models in a parallel fashion. One notable independent ensemble technique is the isolation forest and will be briefly discussed afterwards.

ISOLATION FOREST

In [43], an isolation forest is a recursive partitioning algorithm to identify anomalies which can be modeled using binary trees. This model requires a training and testing phase. In the training phase isolation trees are built to create an isolation forest as follows: a feature is chosen randomly from the set of features and its value is selected randomly between its maximum and minimum value in the data to split data points. This action is repeated until a specific limit is reached to obtain an isolation tree as seen in Figure 3.3. In the testing phase, the earlier a point is isolated as a consequence to splitting, the higher its anomaly score. This score is averaged across the forest to decrease the variability in anomaly detection from individual trees.

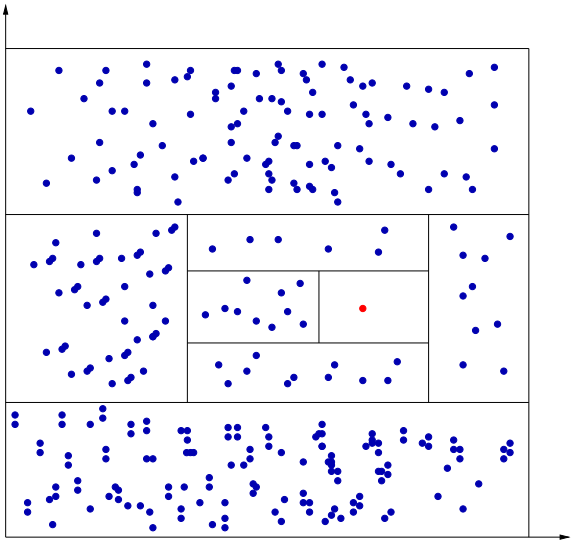
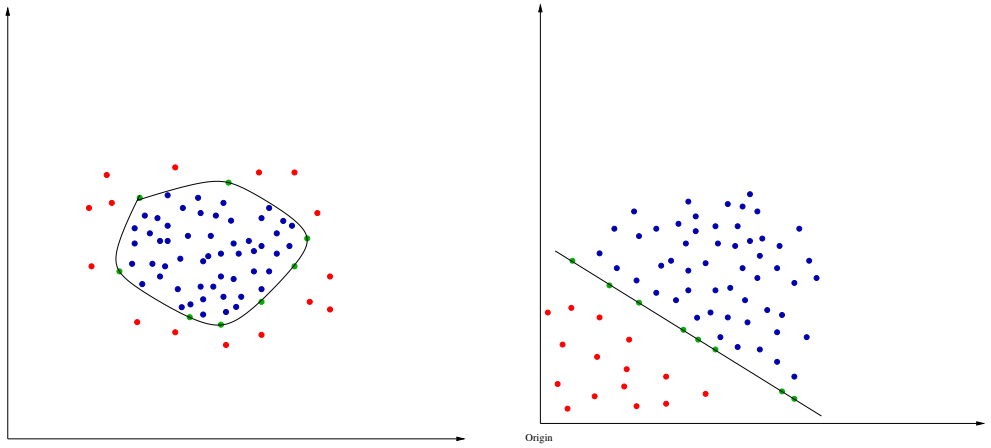


Figure 3.3: Visual example of an isolation forest showing an isolated outlier (red point).

3.2.4/ DOMAIN-BASED ANOMALY DETECTION

Domain-based anomaly detection techniques rely on a boundary to separate normal data from anomalous data [74]. Such techniques disregard data densities and rely solely on a boundary to separate the data. One of the main domain-based anomaly detection technique is the Support Vector Method (SVM) for novelty detection. Using the technique in [23], which is a one-class support vector machine, the data points are projected into a feature space with higher dimensions. They are then separated from the origin using a hyperplane which is henceforth used for outlier identification as seen in Figure 3.4.



(a) Original space (b) Feature space

Figure 3.4: SVM for novelty detection in which the outliers (red points) are separated in the feature space from normal data points (blue points) by a hyperplane passing by support vectors (green points).

3.2.5/ STATISTICAL MODELS

In statistical anomaly detection models, the normal data probability distribution is estimated using training data. When new data points emerge whose probability of showing up is low enough, an anomaly is considered as detected [45]. Statistical models can presume a given known distribution of the data where the parameters are the only unknowns to be estimated and in that case it is called a parametric method. Some notable parametric models to be mentioned are Gaussian mixture models and regression based models like ARIMA, Independent Component Analysis, etc. In other cases, statistical models have no knowledge of the distribution whatsoever and in that case it is termed non-parametric method. Some notable non-parametric statistical methods are histogram based models and kernel function based models. The previously mentioned parametric and non parametric statistical methods will be described in this subsection.

3.2.5.1/ GAUSSIAN MIXTURE MODELS

In gaussian mixture models each cluster is associated with a gaussian distribution. The parameters of these gaussian distributions are learned using expectation maximization [48]. Thus, a random initialization is first applied on the gaussian distributions. Then for each data point, the posterior probability of being associated to a given cluster is computed. The two previous steps are alternatively applied until convergence. Data points can be correctly clustered from now on. If for a given point the likelihood of corresponding to any given cluster is below a specified threshold, then an outlier is identified. A visual example of gaussian mixture models is shown in Figure 3.5.

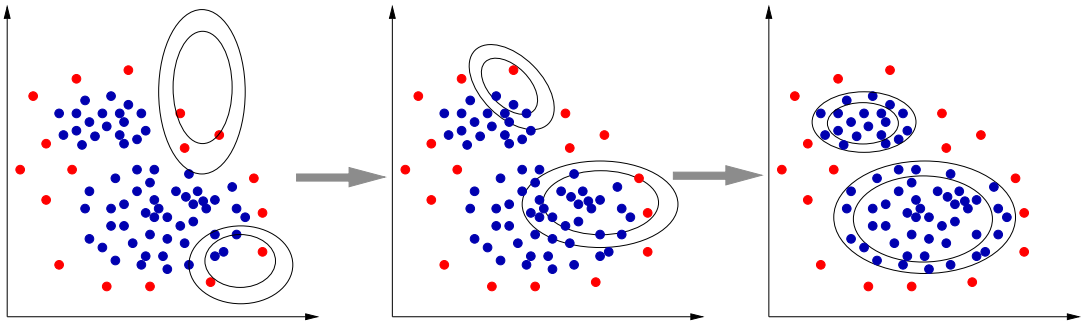


Figure 3.5: Visual example of gaussian mixture models showing the iterative process of estimating the gaussian distributions till convergence. The outliers correspond to the red points and normal points correspond to the blue points.

3.2.5.2/ ARIMA

ARIMA is a statistical method for forecasting in time series. It combines a moving average model, an autoregression model and applies differencing [81]. The moving average and autoregression models try to fit linear models to predict a present variable from past errors and values respectively, while differencing is used to remove trends and seasonality and achieve stationarity of the time series [132]. ARIMA can also be used for anomaly detection.

3.2.5.3/ INDEPENDENT COMPONENT ANALYSIS

Given a signal made of multiple independent signals with non gaussian distributions, ICA is a method of obtaining the source signals from the resulting combined signal [12]. Many approaches can be followed for this purpose such as finding the sources which minimize the mutual information or the sources that minimize the gaussian behaviour. ICA can also be used for anomaly detection in multivariate time series [39].

3.2.5.4/ HISTOGRAM-BASED MODEL

In histogram based models the frequencies for intervals of a univariate training dataset is computed. Then a test sample's anomaly score which depends on frequency values is computed and is used for anomaly detection. Histograms can also be used for multivariate data by computing a histogram for each feature and then computing the associated anomaly scores which are aggregated to obtain a unique anomaly score for all the features. Histogram based models can suffer from an excess of false positives and false negatives due to having narrow and wide intervals respectively [45].

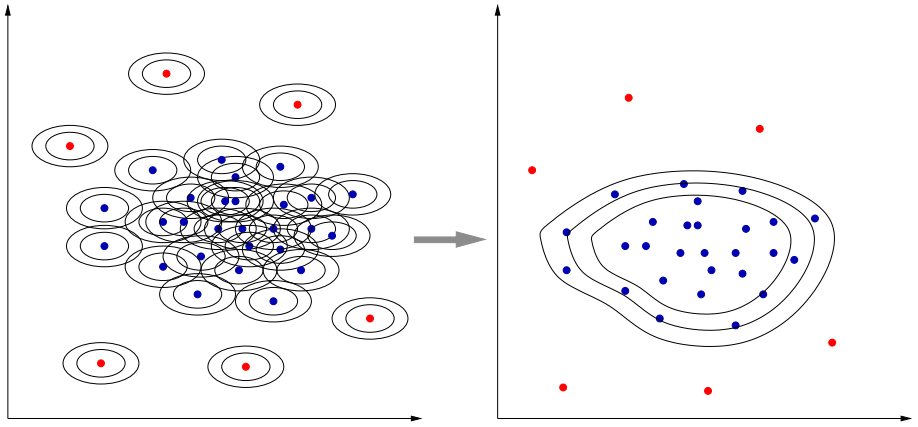


Figure 3.6: Visual example of kernel function-based anomaly detection where the outliers (red points) have a small probability density (below a specified threshold) compared to normal points (blue points).

3.2.5.5/ KERNEL FUNCTION-BASED MODEL

Using this technique, the distribution used for anomaly detection is estimated using a linear combination of continuous distributions around individual data points modeled by a kernel function [7] as seen in Figure 3.6. This gives a smooth approximation of the probability distribution compared to histograms but may suffer from bias around the boundaries of the central distributions.

3.2.6/ DIMENSIONALITY REDUCTION TECHNIQUES

Dimensionality reduction techniques are usually applied to avoid the ”dimensionality curse” [22] such as in the K-nearest neighbors algorithm. Principle component analysis is one such technique that is based on the eigen decomposition of the covariance matrix of the data points [29]. The vector with the highest eigen value is the first principal component which is defined as the direction with the highest variability in data points. As the eigen values are computed in decreasing order of magnitude, their corresponding eigen vectors: the first, second, third, ..., n^{th} principal components are obtained.

The variability change with respect to change in directions can also be used to detect anomalies. More precisely, considering the directions with the lowest variability (last principal components), the higher values for samples are treated as anomalous as seen in Figure 3.7. This PCA-based approach was applied on astronomy data acquired from the Sloan Digital Sky Survey (SDSS) and the Two Micron All Sky Survey (2MASS) [41]. Usually data present non-linearities which cannot be treated with vanilla PCA, but rather using a technique called kernel PCA [19]. The latter technique projects the data into a higher dimensional feature space which transforms the data’s non-linearities into linear behaviours. The main difference in the application of the kernel PCA compared to PCA

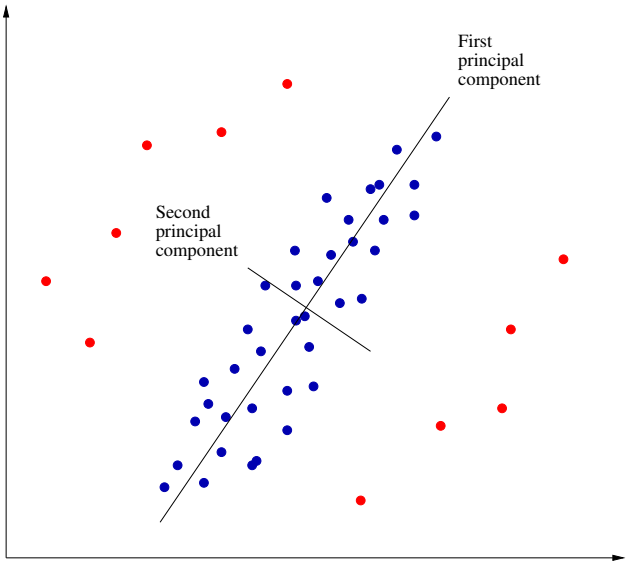


Figure 3.7: Visual example of PCA applied on 2D data, where outliers (red points) are points with high variability relative to the last principal component (the second principal component in this case).

is that a kernel matrix is diagonalized instead of the covariance matrix. Another PCA variant is robust PCA which is tolerant to arbitrarily corrupted observations [54]. Finally, functional PCA is a PCA variant applied to functional data, more precisely square integrable stochastic processes [35].

3.3/ PREDICTION AND DETECTION OF ANOMALIES IN TIME SERIES

The previous section described anomaly detection techniques in a general sense, while this section focuses on techniques for time series anomaly detection. Convolutional neural network (CNN) based methods, recurrent neural network (RNN) based methods, transformer methods, as well as hybrid frameworks and techniques will be discussed in this section.

3.3.1/ CONVOLUTIONAL NEURAL NETWORK-BASED ANOMALY DETECTION

Convolutional neural networks are deep learning architectures that rely on the mathematical operation of convolution to transform tensor data, mainly images, into other tensors for a specific purpose [14]. More concretely, convolutional neural networks use learnable filters to transform the data. The most famous type of CNN is the 2D-CNN which relies on filters that can move in 2 dimensions. This architecture is inspired by the visual cortex in the human brain and is the basis for many types of applications such as image classification, segmentation, pattern recognition, etc.

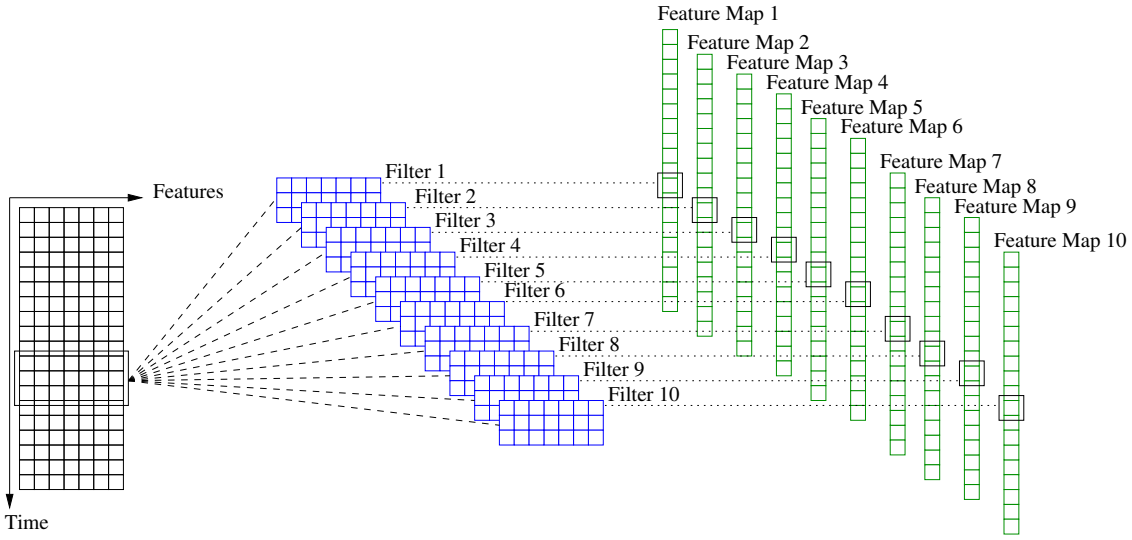


Figure 3.8: Visual example of a 1D-CNN layer. A sliding window convolves with learnable filters to obtain feature maps.

Another type of convolutional neural networks is the 1D-CNN which is mainly used for sequential and time series data, since it is based on filters that can only move in one direction. An example of 1D-CNN layer is shown in Figure 3.8. 1D-CNN can be used for forecasting, classification and other applications like anomaly detection such as DeepANT [118] Microsoft's time series anomaly detection service [118] which will be described afterwards.

3.3.1.1/ DeepAnT: a deep learning approach for unsupervised anomaly detection in time Series

The model presented in [118], called DeepANT, is used for unsupervised anomaly detection in univariate as well as multivariate time series. First, a 1D-CNN architecture (containing 1D-CNN layers as well as 1D Max Pooling layers) is used to extract relevant features and then predict the next sample based on a window of samples. Then, the euclidean distance between the real value and the ground truth is evaluated and is used as an anomaly score. Finally, the anomaly score is compared to a manually defined threshold in order to decide if a given sample is anomalous.

The deepANT has a wider scope of applications relative to LSTMs since it is suitable for both small data and big data. The deepANT architecture can also be used for discord detection which are defined as anomalies in subsequences. When predicting a subsequence, the anomaly scores of each of their individual samples are aggregated into a single one anomaly score used for discord detection based on another predefined threshold.

3.3.1.2/ TIME-SERIES ANOMALY DETECTION SERVICE AT MICROSOFT

The authors of [130] combined spectral residuals (SR) with 1D-CNN to detect anomalies in time series. The SR part of the architecture is based on Fourrier transform and inverse Fourrier transform to obtain a map which highlights the anomalies in the time series data and is called saliency map. However, instead of using a threshold to detect anomalies, normal time series data are transformed using SR into a saliency map. Then synthetic anomalies are introduced in the map. A 1D-CNN followed by a fully connected layer is used to classify the salient map data associated with the time series data as normal or anomalous based on the synthetic anomalies' labels.

3.3.2/ RECURRENT NEURAL NETWORK-BASED ANOMALY DETECTION

Recurrent neural networks (RNN) are a subtype of neural networks that contain loops and hidden states in their structure. They cycle the information from the previous output back to the current input. Such loops can give them memory capabilities and allow the networks to capture sequence dependencies. There exists many variants of RNNs such as the vanilla RNN (shown in Figure 3.9) and gated RNNs like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) which can work on sliding windows with bigger sizes (lookback). RNNs are mostly used for forecasting and classifying sequential data. RNNs can also be used for anomaly detection such as LSTMs [86] and RNNs with skip connections [146] which will be be discussed in what follows.

3.3.2.1/ LONG SHORT-TERM MEMORY NEURAL NETWORKS FOR ANOMALY DETECTION IN TIME SERIES

The LSTM architecture (shown in Figure 3.10) is a special RNN network especially conceived to remedy the problem of vanishing gradients encountered in regular RNN networks [25]. When trying to update the weight of a given connection in a regular RNN the gradient of the prediction error with respect to this weight used for the update can, in some cases, become really small. This can prevent the update and is called the vanishing gradient problem. This is due to the fact that the gradient contains repeated multiplications of $tanh$ derivatives which are positive values smaller than one [167]. A LSTM possess an internal memory modeled by a cell state and a hidden state. While processing sequential data, the LSTM learns to forget irrelevant information from the cell state using a forget gate and to update appropriate information in the cell state using an input gate. The input gate chooses relevant information from a vector of candidate values to update the cell state. Finally, an output gate learns to extract suitable content from the cell state to be output as the new hidden state. Note that the LSTM architecture obligates the gradient

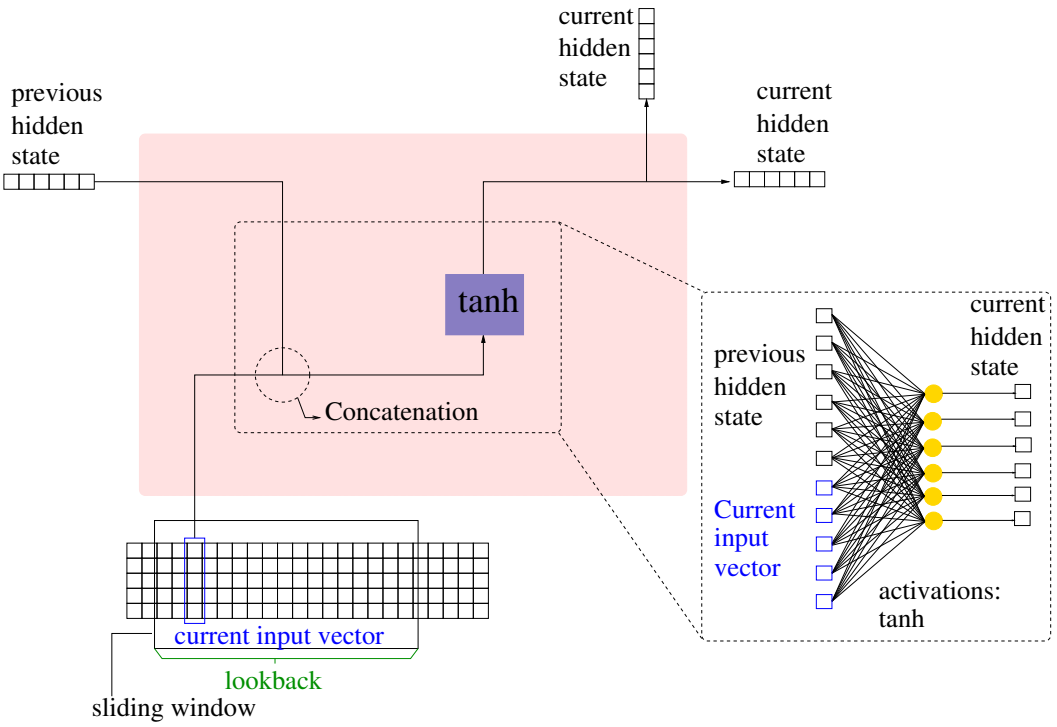


Figure 3.9: Visual example of a regular recurrent neural network layer. A loop connects the current hidden state with the previous hidden state.

of the error to contain 4 added terms (which include the forget gate activation) instead of containing repeated multiplications. The additive structure of the gradient and the presence of the forget gate activation function in this structure help prevent the vanishing of the gradient. More precisely, the forget gate activation can take on values to specifically prevent the gradient form vanishing [167].

The hidden state can be used as an input to another LSTM layer in case of a stacked LSTM, otherwise it gives directly an output without any modification. This mechanism makes the LSTM especially useful for processing sequential data like time series for a variety of application namely forcasting and anomaly detection. For instance, in [86], the authors used a basic LSTM and a multivariate gaussian distribution for time series anomaly detection. In their technique, an LSTM architecture learns to predict a specified number of future samples based on a chosen number of input samples. Then the errors, which are the difference between the predicted samples and their corresponding ground truths, are fit to a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$. When the likelihood of the error extracted from $\mathcal{N}$ is less than a specific threshold an anomaly is detected.

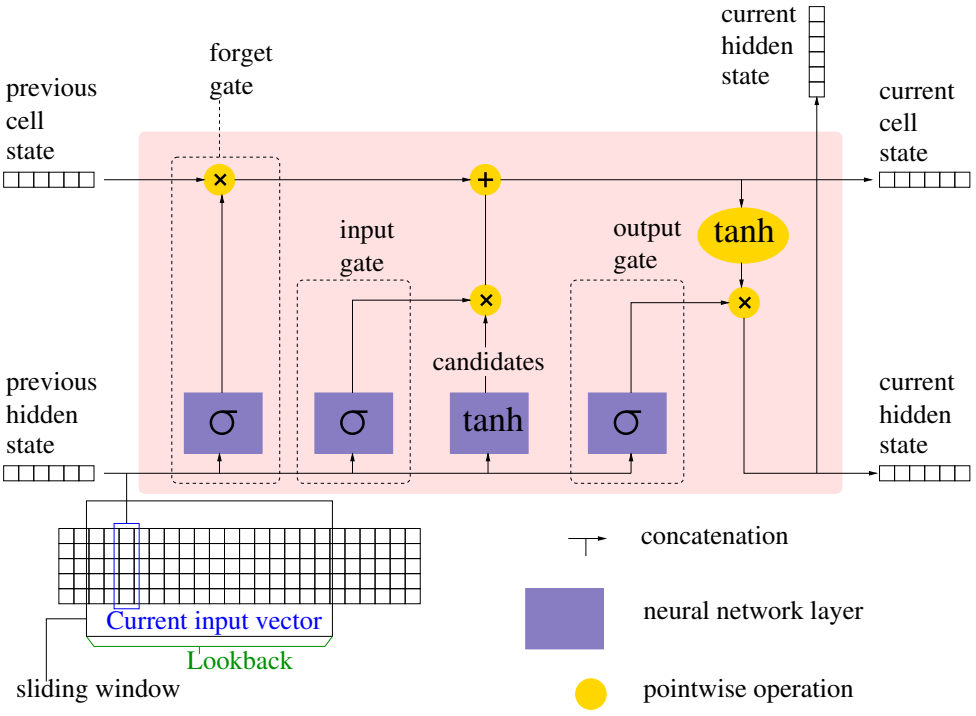


Figure 3.10: Illustration of a LSTM layer.

3.3.2.2/ TIME SERIES ANOMALY DETECTION USING TEMPORAL HIERARCHICAL ONE-CLASS NETWORK

In order to detect anomalies in time series, the authors in [146] created a model which combines deep learning, more specifically recurrent neural networks, with one-class clustering. The architecture is made of two parts. The first part is used to extract temporal features and is made of multiple layers of RNN cells (regular RNN, LSTM or GRU) with skip connections which diminish the effects of vanishing gradients. This ensures the possibility to work with longer time periods. The extracted features are introduced into the second part in order to be fused and hierarchically clustered. The final output is an anomaly score used for anomaly detection. The loss function (metric expressing the amount at which predicted values differ from target values) used to train this model combines three types of loss functions:

- Temporal hierarchical one-class loss: its use is similar to deep SVDD and relies on cosine similarities.
- Self supervised loss: In order to learn additional helpful features to improve the whole process.
- Orthogonal loss: this loss ensures that the clusters are as distinct as possible.

3.3.3/ TRANSFORMER-BASED ANOMALY DETECTION

Classically encoder-decoder architectures encode a sequence of input data into a context vector which is then used to decode a sequence of outputs. The encoders and decoders are often made of RNNs like LSTMs. RNNs usually suffer from vanishing gradients especially vanilla RNNs. In addition to this limitation, all the inputs are summarized sequentially into one context vector hence the accumulation of errors which are translated into the decoded phase. To remedy this problem the attention mechanism was invented.

The attention mechanism is similar to human attention, which selectively focuses on relevant information in order to achieve a specific task. More precisely, the hidden states are multiplied by weights and added together to obtain the context vector. A slightly more advanced attention mechanism is self-attention [108], which is a general case for more conventional attention mechanisms. Unlike traditional attention mechanisms, self-attention requires no RNN to function and its computation can be parallelized.

The transformer architecture whose core is based on the self-attention mechanism was first introduced to be used for Natural Language Processing. It established state of the art scores against many benchmarks mainly language translation benchmarks. The transformer architecture was a monumental paradigm shift in the deep learning domain and it gave rise to big powerful language models like GPT3 [135]. The basic mechanism of self-attention will be explained in the context of natural language processing as follows. Suppose we have a sequence of inputs usually a sentence. When we focus on a target word, other relevant input words need to stay in memory to a certain degree. This memory is essential for the understanding of the sentence and is obtained using self-attention. First embedding vectors are computed from inputs. Relevant information to a specific word, which are distributed across a corpus of text need to be queried, hence the need for a query vector corresponding to the input. Key vectors corresponding to each word in the corpus represent the address needed to access the information associated with those words. Such information is contained in corresponding value vectors. The query, key and value vectors are computed from embeddings by multiplying them by trainable query, key and value weight matrices. Then, the similarities between the query and the keys are computed. This similarity is based on a scaled dot product followed by a softmax operation. Similarities are multiplied by corresponding value vectors to obtain attention values which are finally summed to obtain the output of the self-attention. This is basically what a self-attention layer (shown in Figure 3.11) amount to and it is the most crucial component of the transformer architecture.

The tremendous success of the transformer architecture in natural language processing incentivized its use in other applications. One such application is anomaly detection [151] and will be detailed in what follows.

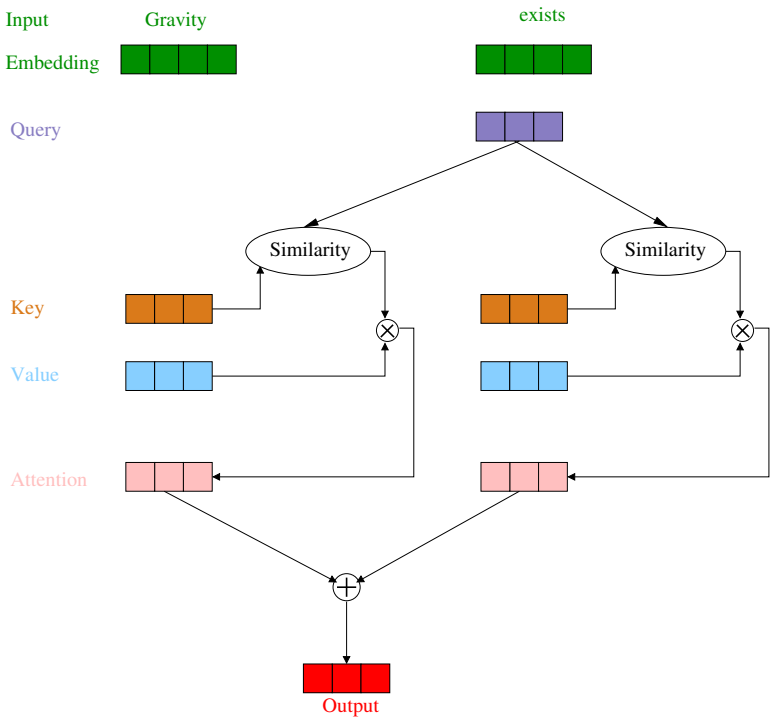


Figure 3.11: Diagram of self-attention showing the process of extracting relevant information to the word “exists” from all the words contained in the sentence “Gravity exists”.

LEARNING GRAPH STRUCTURES WITH TRANSFORMER FOR MULTIVARIATE TIME SERIES ANOMALY DETECTION IN IOT

In [151], the underlying graph structure information of an IoT system is integrated with multivariate time series data for anomaly detection. Before explaining their approach some key concepts need to be explained, namely the Gumbel-max trick, the Gumbel-softmax distribution, graph convolutions and dilated graph convolutions. In some deep learning methods, some samples need to be generated from categorical distributions with class probabilities π. The Gumbel-max trick can be used for this purpose which is expressed as follows: $z = one\_hot(argmax[g_i + log\pi_i])$ where g_i denote independent and identically distributed samples drawn from $Gumbel(0,1)$. $Gumbel(\mu,\beta)$ is the gumbel distribution expressed by $F(x;\mu,\beta) = e^{e^{-(x-\mu)/\beta}}$ where μ is called location and β is called scale. Sampling using the Gumbel-max trick prevent differentiation while training the network. This is due to the non differentiablity of the $argmax$ operation. Since differentiation is mandatory, a special distribution approximating the Gumbel-max trick which allows differentiation is needed. One such technique is called the Gumbel-softmax distribution which approximates the Gumbel-max trick using the softmax operation [92]. As for graph convolutions, these correspond to the counterpart of convolutions in images, but applied to graphs. Given a specific vertex in a graph where each vertex is associated with an embedding vector, a convolution aggregates information from neighbouring nodes and then

updates the embedding of the node in question [159], [128]. Finally, when we talk about a dilated graph convolution, the type of neighborhood used skips every d vertices, where d is called dilation rate. This neighborhood is called dilated neighborhood [128]. Now that the needed keywords are defined, the approach used in [151] for anomaly detection can be explained. First, the graph structure of IoT systems is learned using Gumbel-softmax sampling. Then, graph convolutions are used to propagate the information inside the graph, and dilated convolutions are used to represent long-term time dependencies. Graph convolutions and dilated convolutions are applied alternatively several times. Next, a modified version of a transformer is used for one step prediction. Finally, an anomaly score is computed which is based on the sum of the square error of the prediction for each node of the graph. The anomaly score is compared to a fixed threshold to detect anomalies.

3.3.4/ Generic and scalable framework for automated time-series anomaly detection

In [84], researchers at Yahoo created a framework for time series anomaly detection. It combines many approaches for anomaly detection such as forecasting, seasonality and trends decomposition, residuals distribution estimation and clustering. While analyzing time series, the researchers divided anomalies into three types: outliers, change points, and anomalous time series.

Outliers are data points which are remarkably different from their expected values. Plug-in models (considered as black boxes) are used to predict future samples and absolute or relative errors or even other metrics can be thresholded to detect outliers. Another way of detecting outliers relies on the decomposition of time series into their trends, seasonality and noise in the time or frequency domain. Then the noise is compared to a threshold.

Change points are data points which are markedly different from their context and they differ from outliers in that they last longer in time. In order to detect these anomalies residuals are computed and then their probability density function is obtained using a distribution estimation technique and a statistical distance is also evaluated to express the variation in the distribution indicating a change points anomaly measure.

A time series is anomalous if it is highly distinct in a set of time series. In order to detect these anomalies, samples are clustered according to their statistical features such as seasonality, spectral entropy, trend, etc. Then the distance of each sample from clusters' centroids is computed for anomaly detection.

3.4/ ANOMALY DETECTION IN ADS-B PROTOCOL

3.4.1/ ADS-B SPOOFING ATTACK DETECTION METHOD BASED ON LSTM

In the article [147], the authors relied on forecasting ADS-B vectors based on sliding windows of ADS-B vectors for ADS-B anomaly detection. The forecasted ADS-B vectors are compared with real vectors in order to detect ADS-B anomalies. The ADS-B vectors in this study contain the following features: longitude, latitude, altitude, speed, heading, and climb rate. The authors used an LSTM architecture made of one layer of 14 units which is connected to a fully connected output layer made of 7 units which corresponds to the ADS-B vector to be forecasted.

In order to classify an ADS-B vector as anomalous or normal, a residual between its actual value and its predicted value as well as a specific threshold are computed. When the residual exceeds the threshold, an anomaly is considered as detected. More concretely, the set of ADS-B vectors used in this study is divided into three datasets: a training dataset termed M_1 (containing 80 % of the total set) used to train the forecasting model, a set named M_2 (containing 10 % of the total set) used to compute the anomaly threshold and an anomaly detection test set M_3 (containing 10 % of the total set) which depends on the type of anomaly whose detection is being tested. In the M_2 set, in order to compute the residual of a predicted feature represented by d_i, the following expression is used: $d_i = |p_i - v_i|$, where p_i and v_i denote respectively the predicted feature value and the actual feature value. i represents the index of the ADS-B vector in question. The mean value μ and the standard deviation σ of the d_i values are computed. The threshold t for a specific feature is equal to 3σ. In the test phase i.e. in the M_3 set, each anomaly score a for a given feature in a specific ADS-B vector is computed using the following expression: $a = |di - \mu|$. Nevertheless, usually the totality of a vector needs to be classified as normal or anomalous. Therefore, the overall anomaly score of a vector is just the mean of the anomaly scores for each feature of the vector. Likewise, the overall threshold is also the mean of thresholds associated with each feature of the vector.

Different classes of attacks were considered in the evaluation of the detection method. In the first class of attacks, random noise was inserted in the data, more specifically ADS-B vectors were multiplied by a factor between 0 and 2. In the second class of attacks called injection, there exists two sub-types, namely route replacement and fixed offset. In route replacement, a sequence of ADS-B vectors is replaced by another real sequence of ADS-B vectors from another flight. In fixed offset (+/-), the features of a sequence of ADS-B vectors are increased/decreased by 10 %. The third class of attacks called modification is divided into three sub types: offset, heading change and climb rate change. In an offset attack (+/-) of value x, a specific feature in ADS-B vectors in a fragment of a flight is gradually modified by adding/subtracting x to the first vector, $2x$ to the second, $3x$ to the

third and so on. In the heading change and climb rate change, the heading and climb rate values are modified, respectively, by the replacement of these values by their opposites.

The detection performance of their technique is computed in terms of Recall, Precision and F-score. The Recall also called Sensitivity and True Positive Rate (TPR) represents the fraction of detected attacked messages relative to the total number of attacked messages. The Precision denotes the proportion of detected attacked messages that are correctly identified as real attacks. Finally the F-score is the the harmonic mean of the precision and recall [145].

The detection approach used by the authors gave the following results (averaged over all the mentioned attacks): 93.67 % Precision, 61.59 % Recall and 70,76 % F-score. In order to improve the performance the authors relied on the following technique. They regarded an ADS-B vector as attacked when at least one of the sliding windows containing this vector is associated with an anomaly. As a consequence of this consideration, the previous performance changed to: 93.49 % Precision 98.96 % Recall and 83.46 % F-score. This corresponds to a huge improvement in recall, while leading to a substantial improvement in F-score as well.

3.4.2/ LSTM ENCODER-DECODER FOR DETECTING ANOMALOUS MESSAGES

OVERVIEW

In the ADS-B protocol, the value of each message is influenced by its context, making it harder for predictive models to detect anomalies. For this reason the authors in [114] decided on using an LSTM encoder-decoder to identify anomalous messages without relying on any modification of the ADS-B infrastructure. Anomalies are not detected in individual ADS-B messages, but in windows of messages in which a window is considered malicious if at least one of its messages is anomalous.

Anomalies are detected based on the messages' reconstruction errors in the encoder-decoder architecture shown in Figure 3.12. First, the encoder summarizes a sequence of ADS-B vectors into a latent space where each vector contains relevant features obtained from its associated ADS-B message. Then, a decoder tries to reconstruct the sequence of ADS-B vectors. The anomaly detection model is trained on a normal flight from take off to landing. Since the encoder-decoder is trained on normal messages only, it will struggle to reconstruct malicious ADS-B messages indicating the presence of anomalies. Such an encoder-decoder architecture which relies on reconstruction error for anomaly detection is called autoencoder.

For the purpose of identifying anomalies, the authors used the following features extracted/computed from the ADS-B messages: speed, latitude, longitude, altitude, heading,

four context points, and the distance between the current position and the previous position of the aircraft.

In order to assess the ability of the proposed approach to detect anomalies, ADS-B data were acquired using flightradar24 [4] and were altered using the following attacks:

- Random noise (RND): in this type of attack ADS-B vectors are multiplied by a random decimal number between 0 and 2.
- Different route (Route): in this type of attack a fragment of the route is swapped by another fragment from a different route.
- Gradual drift of 400 feet : in this type of attack the altitude of messages in a fragment of a flight is continuously modified by adding/subtracting 400 feet to the first message, $2 * 400$ feet to the second, $3 * 400$ feet to the third and so on. When addition and subtraction are applied, the attack is called SHIFT Up and SHIFT Down respectively.
- Velocity drift of 5 knot: this attack is analogous to the gradual drift and is applied to the velocity in ADS-B messages.
- Message blocking (BLK): this attack tries to mimic the consequences of a denial-of-service attack. Essentially, the last 4 messages out of every 5 successive messages are deleted.

Since random noise attacks and different routes attacks touch many features, their detection is much faster than drift attacks which only modify one feature each.

The authors were able to successfully detect anomalies of types RND, Route, Gradual drift, Velocity drift and BLK. However they relied on previous records of the same flights for training which might make it easier for the model to detect anomalies. Other techniques like the following subsection do not rely on previous recordings making detection a harder task but potentially more useful especially in detecting anomalies in military flights or small aircraft flights which do not always take the same route.

3.4.3/ ADS-B ANOMALY DATA DETECTION MODEL BASED ON VAE-SVDD

3.4.3.1/ OVERVIEW

In the article [157], the authors used a Variational Autoencoder (VAE) [65] combined with Support Vector Data Description (SVDD) [32] to detect anomalies in sequences of ADS-B messages. First, a VAE (shown in Figure 3.13) is trained to learn the reconstruction of

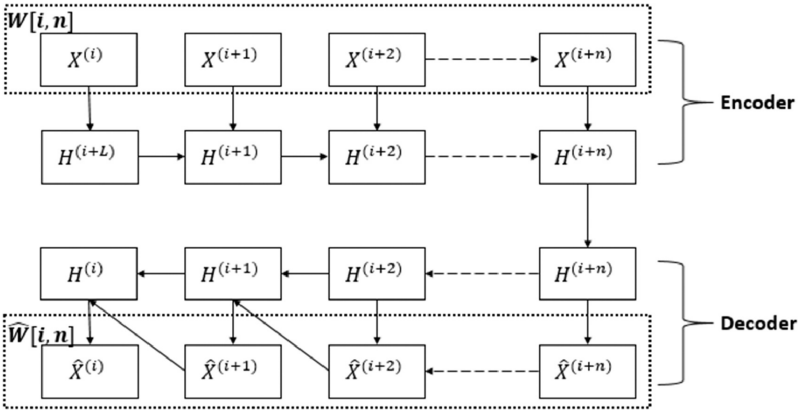


Figure 3.12: LSTM autoencoder for ADS-B anomaly detection where the inputs correspond to vectors containing ADS-B information.

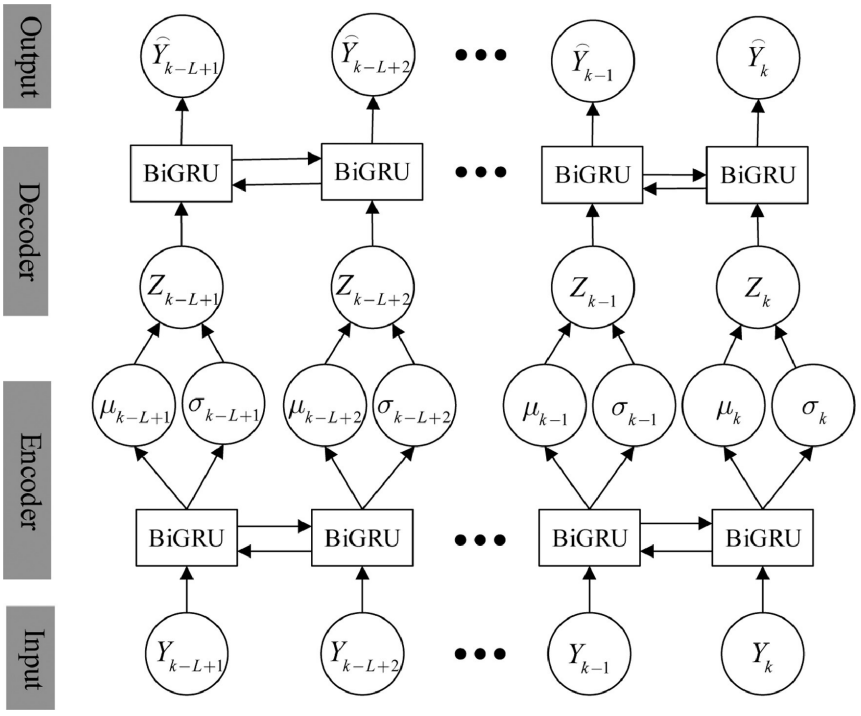


Figure 3.13: LSTM VAE autoencoder for ADS-B anomaly detection where the inputs correspond to vectors containing normalized ADS-B information.

windows of scaled ADS-B messages representing the following features: latitude, longitude, altitude and velocity. The heading feature was not used in their technique. Each message is contained in many windows thus it is reconstructed many times. In order to choose the appropriate reconstructed message, cosine similarities between the original scaled message and the reconstructions is applied. Finally, the differences between the reconstructed messages and their corresponding scaled messages are classified as normal or anomalous using SVDD which is a one class classification technique.

Table 3.1: Average performance of multiple architectures in detecting ADS-B anomalies of the following attacks: Different route (Route), Velocity drift, Constant/Random position deviation.

Evaluation index	IForest	GRU	LSTM	LSTM encoder-decoder	VAE-SVDD
FPR	11.26	6.71	6.92	7.32	5.76
FNR	41.18	29.35	28.24	10.07	7.33
ER	32.59	19.25	18.76	8.85	6.6
F1-score	76.28	82.27	81.81	90.412	92.74

3.4.3.2/ COMPARISON WITH OTHER TECHNIQUES

In order to train and test their models the authors used ADS-B data from 50 and 20 different flights respectively. These datasets are obtained from the OpenSky Network [75], a community driven open database of flight data. Different types of attacks were used to evaluate the detection method:

- Different route (Route): this attack was also used in [114].
- Velocity drift: this attack was also tested in [114], but a multiple of 2 m/s was used instead of 5 knot.
- Constant position deviation attack: the flight is deviated by 1 degree of latitude and longitude.
- Random position deviation attack: noise of mean 0 and standard deviation 0.5 is fed into the latitude and longitude values of the flight.
- DOS attack: the flight vanishes relative to the air traffic surveillance system.

The VAE-SVDD's detection performance was compared with different models: an isolation forest [59], a GRU, an LSTM, and a modified configuration of LSTM encoder-decoder [114] i.e. they used only latitude, longitude, altitude and velocity as features without the heading or any Vincenty distance. In the GRU and LSTM techniques, ADS-B messages are predicted from windows of past messages, then cosine similarities were used to compare these predictions with the actual messages to determine their legitimacy.

It can be noted from Table 3.1 that the VAE-SVDD outperforms the other models in detecting all the previously mentioned attacks. Performance metrics are False Positive Rate (FPR), False Negative Rate (FNR), Error Rate (ER), and F-score. The isolation forest had the worse performance due to the fact that it uses the messages without taking into consideration their existing temporal correlations. The GRU and LSTM did worse than the encoder-decoder architectures because they could not properly detect constant deviation attacks. The reason for this defect is the lack of cumulative change in the flight tracks

and hence the ADS-B messages position values remain within logical bounds. Since the VAE-SVDD takes the statistical distribution of ADS-B messages into consideration, it outperforms the LSTM encoder-decoder. The SVDD's optimized search of anomaly threshold improves its detection ability even further.

Nevertheless, the distributions of the input data need to be expressed accurately using the VAE-SVDD. This architecture is well suited for crude attacks such as velocity drift, but its performance deteriorates on new data giving a high false positive rate because it struggles in the reconstruction of the messages [162].

3.4.4/ CAE: CONTEXTUAL AUTOENCODER FOR MULTIVARIATE TIME-SERIES ANOMALY DETECTION IN AIR TRANSPORTATION

Normal commercial flights are usually divided into three phases: CLIMB, CRUISE and DESCENT. In some cases, when a false data injection attack is applied, it might look like the flight is normal. For example, a gradual descending attack might be applied on the cruising phase. It may seem that a normal descent phase has just begun, when in fact it was a false data injection which was not detected due to the fact that phases were not properly identified beforehand. To remedy this problem, the authors in [162] introduced the contextual autoencoder for multivariate time-series anomaly detection in air transportation.

Their work was based on an autoencoder architecture [148] that sorts out sounds of speakers placed in distinct locations. In order to detect ADS-B anomalies, batches of samples are separated into three mini-batches, one for each phase of flights, using fuzzy logic [107]. The order of the samples in the original batches is saved for a future process. Then each type of mini-batches is encoded in a different latent space, one for each phase. Then mini-batches are reconstructed from the latent spaces and concatenated based on the saved order of samples to reconstruct the original batch. This model is shown in Figure 3.14. Their model is trained with purely normal ADS-B data and when it fails to reconstruct test data an anomaly is detected. The anomaly score used to evaluate the reconstruction is the following:

$$Anomaly(window) = \sum_{i=0}^{n} 1 - (x_i - \hat{x}_i)^2,$$

where x_i is the i^{th} input vector in its associated window of input vectors and $\hat{x}_i$ corresponds to its reconstructed vector, n is the size of the window. A threshold of anomaly score τ is used for each phase, based on the 3σ rule for its ease of use: $\tau = \mu + 3\sigma$ where μ and σ are the mean and standard deviation of the anomaly score of the training data.

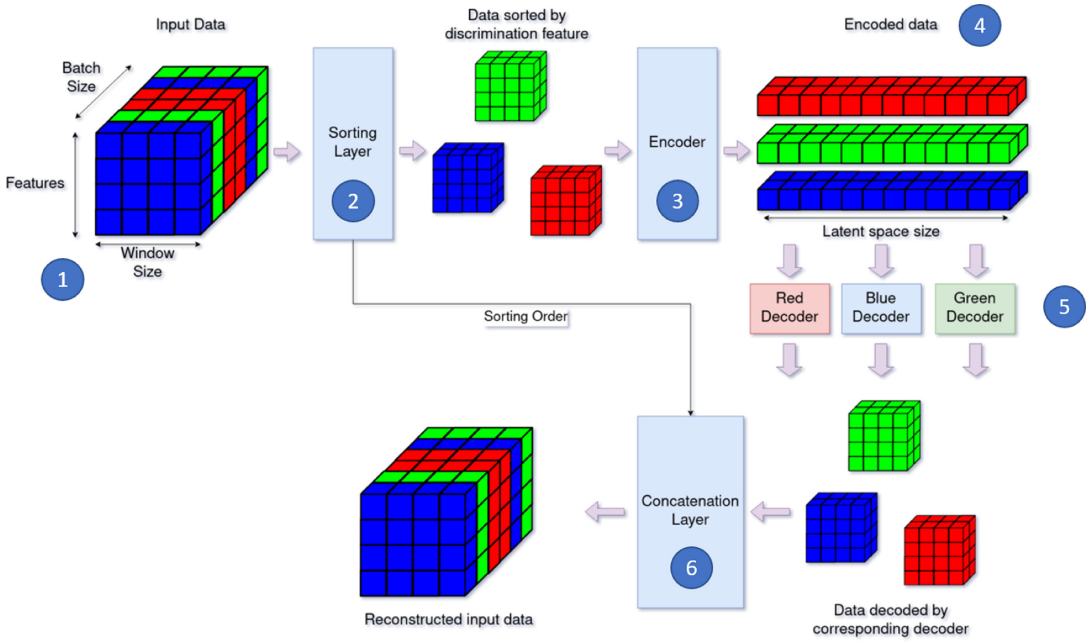


Figure 3.14: Diagram of a contextual autoencoder for ADS-B anomaly detection.

This model was tested against different types of attacks such as velocity drift, made up crashes and constant position offset. Since made up crashes are descents that happen too early, it was especially apt at detecting these attacks due to the fact that flight stages were taken into consideration by the anomaly detection model. This model is also very suitable for speed drift attacks that distort flight stages. However, it may not be appropriate for flights that do not have well-defined clean flight stages, such as military flights.

3.4.5/ VizADS-B: Analyzing Sequences of ADS-B Images Using Explainable Convolutional LSTM Encoder-Decoder to Detect Cyber-Attacks

In the previous techniques anomalies were detected for ADS-B messages emitted by individual aircraft. The difficulty of anomaly detection of ADS-B messages of different origins lies in the fact that the relation between these messages needs to be investigated in addition to the analysis of messages emanating from the same aircraft. To tackle this obstacle, the authors in [122] represented the airspace's ADS-B information using sequences of images and used a convolutional LSTM encoder-decoder network to identify anomalies. The convolutional components of the network catch the spatial dependencies of the data whereas the LSTM components capture the temporal/sequential relationship. A basic convolutional LSTM encoder-decoder network is shown in Figure 3.15.

The ADS-B information is expressed as follows in the images. Each aircraft is represented by an arrow whose size corresponds to the altitude. The direction of the arrow

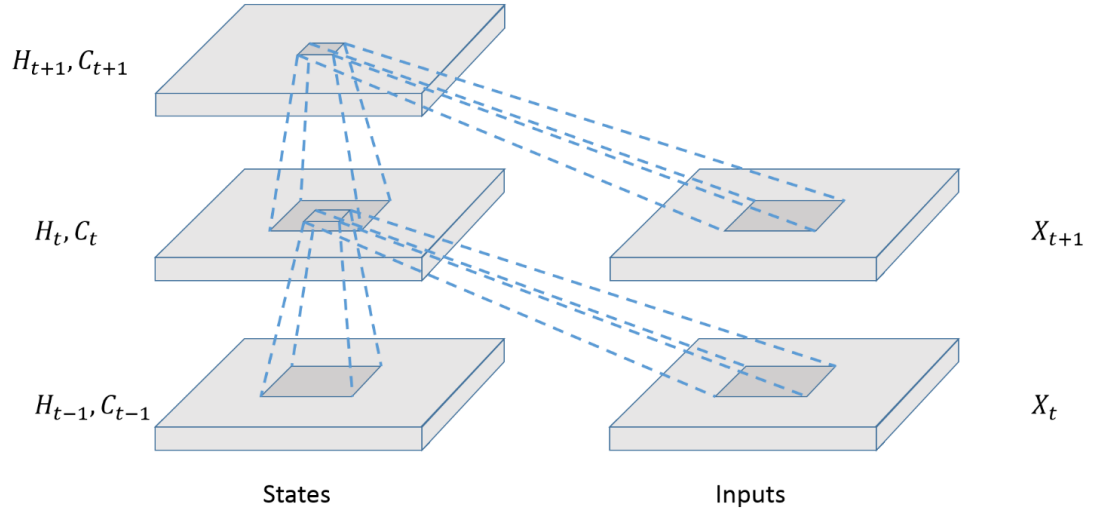


Figure 3.15: Diagram of a Conv-LSTM layer used in the VizADS-B anomaly detection approach. X represent the inputs, whereas H and C represent the hidden and cell states respectively.

is computed from the heading. The arrow's position expresses the longitude and the latitude, while the ground speed is represented by the arrows length. Like the previous models, anomalies are detected using reconstruction errors which are expressed by the color of the arrow. These anomaly scores are not pixel-wise but depend on the structural similarities, hence the use of the SSIM function [72] to compute these errors. In order to evaluate the model, ADS-B data were downloaded from the OpenSky Network [75] and different kinds of anomalies such as flood, ghost and change altitude attacks injected. The model was able to detect such anomalies, but the performance was lower for attacks that modify the messages of real aircraft, since they touch smaller areas in the images.

3.5/ DISCUSSION

Many algorithms can be used for anomaly detection for the ADS-B protocol, however each one of them has its advantages and pitfalls.

Nearest neighbours techniques do not need special assumptions and only rely on a specified number of nearest neighbors. But they suffer from high computational time, especially accentuated by high dimensionality and larger datasets due to the need to compute plenty of distances. They sometimes fail to detect anomalies which have a sufficient amount of close neighbours, therefore it is crucial to properly specify the threshold which is usually done by trial and error.

Clustering models constitute unsupervised anomaly detection techniques that can be quickly tested since the number of clusters is limited and constant. However, anomaly

detection is not the original purpose of clustering algorithms, hence they may not be especially optimal for such goals. Clustering might also slow down anomaly detection due to high time complexities.

Statistical techniques are also a viable option for anomaly detection, which work well when the data distributions are accordingly hypothesised, but may require additional hypothesis testing, which is particularly difficult for high-dimensional data. In addition, the data often do not follow a known distribution, which can lead to poor performance [45].

Domain based anomaly detection are usually suitable for datasets with evident separation boundaries. However such algorithms are not robust against noisy data and are not suitable for training datasets whose number of samples is lower than the number of features. As for dimensionality reduction techniques, they diminish the time and space complexity but the consequent information loss may cause anomaly detection techniques to underperform. Ensemble techniques are robust, in other words they lessen the variability of the predictions. However, they may lack the ease of interpretation. Ensemble models consisting of complex individual models may be time costly since they add their individual computational time. All these techniques are in theory applicable to ADS-B anomaly detection but need manual feature extraction.

Deep learning 1D CNN model rely on simple operations (addition and multiplication) hence they can can be implemented in real-time using cheap hardware. Contrary to ordinary multilayer perceptrons, 1D CNNs can extract features and perform classification using an individual skeleton [155]. RNN networks, especially LSTMs, can capture long-term dependencies but may suffer from vanishing gradients. Also LSTMs or other advanced RNN models have high time complexity compared to convolutional networks because they contain several memory cells [105].

Transformer architectures are not sequential in nature, hence parallel computing can be applied to such models unlike RNNs. Also vision transformers which are transformer architectures especially devised for treating image data outperform convolutional neural networks in terms of accuracy and efficiency [154], [152]. However vision transformers need much more data than regular CNNs, therefore CNNs and vision transformers are suitable for small datasets and big datasets respectively.

Autoencoders architectures try to summarize input information into a latent space with as much information as possible. However the relevance of such information is not necessarily guaranteed. Furthermore, autoencoders may lose some important relevant information if the bottleneck is too narrow. Therefore, special care need to be taken into consideration while choosing the size of the bottleneck layer.

As for variational autoencoders, these are special types of autoencoders that encode normal data into probability distributions [80]. Those distributions are then used for

anomaly detection contrary to regular autoencoders which rely on reconstruction errors for anomaly detection. Therefore, variational autoencoders do not need special thresholds to be set for the purpose of anomaly detection.

3.6/ CONCLUSION

In this chapter, general anomaly detection techniques were described as well as anomaly detection techniques for time series and sequences of ADS-B messages. Such techniques range from traditional machine learning methods to more recent deep learning architectures. Autoencoder architectures have been seen as a popular choice for ADS-B anomaly detection. Nevertheless all the techniques mentioned for ADS-B anomaly detection are unsupervised or semi-supervised and may struggle to detect specially engineered attacks. For this reason supervised anomaly detection was used in our contributions, which will be examined in detail in the following chapters.

4

CONTRIBUTION 1: A COMPARATIVE STUDY OF DEEP LEARNING ARCHITECTURES FOR DETECTION OF ANOMALOUS ADS-B MESSAGES

4.1/ INTRODUCTION

Since the 1920's, air traffic is becoming more prevalent by the year which results in a steady increase of the number of aircraft roaming the airspace. This requires the expansion of the air surveillance systems in order to be able to manage each one of these aircraft. Such an accommodation is planned to be implemented using different technologies and notably the Automatic Dependent Surveillance Broadcast (ADS-B) system. The ADS-B protocol is based on the idea that aircraft as well as air traffic controllers communicate with each other using messages. However, for practicality reasons, those messages are not encrypted thus malicious messages can be injected. Hence, these attacks need to be detected to ensure the safety of the protocol. One approach to detect these attacks is using machine learning and deep learning architectures [89, 85], which have received an increasing interest in the cybersecurity field. In this chapter, we will consider a very basic attack scenario on data vectors containing information from ADS-B messages, namely a rough change of the altitude value in some vectors. Then we will evaluate various deep learning architectures for the purpose of detecting those attacks in a supervised fashion, especially LSTM architectures which appear to be the most promising one. Note that this contribution was presented at the CODIT 2020 conference [140].

The remainder of this chapter is organized as follows:

- First, the studied machine learning and deep learning architectures are briefly presented for the case of alteration scenario targeting the altitude value.

- Second, the experimental work done with the studied architectures is described.
- Finally, we draw some conclusions for the design of a new architecture for FDIAs detection.

4.2/ STUDIED ARCHITECTURES

In the ADS-B protocol, aircraft broadcast messages over time during their operations. Consequently, the data to process to detect a FDIA targeting an aircraft consist in a set of time series obtained after decoding the received ADS-B messages.

The fact that the data have a temporal evolution indicates that machine learning models able to keep information from the past should be the most suited, namely neural networks with a recurrent architecture. Natural candidate deep learning architectures for examination are therefore the LSTM architecture and its variants which suffer less from vanishing gradients compared to regular RNNs. Even if CNNs are mainly used for image processing, they can also be used for other kinds of input data like time series. In our case study, we have also evaluated a CNN, specifically a 1D-CNN, before using a LSTM in order to extract features from input data. Supervised learning was chosen to be applied using these architectures because it is rarely studied in the literature for the case of anomaly detection. This is mainly due to the lack of labeled data.

Apart from neural networks a Gradient Boosting method was investigated, namely XGBoost a technique attracting attention for its prediction speed and accuracy. XGBoost, which stands for eXtreme Gradient Boosting [91], is a recent gradient boosting algorithm designed for speed and performance. Gradient boosting algorithms constitute machine learning techniques suited for solving regression and classification problems. This kind of algorithm produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

4.3/ EXPERIMENTAL WORK

In order to perform a first evaluation of the ability of different deep learning models to detect FDIAs in the ADS-B case study, we have performed some experiments using data in which the altitude values have been roughly altered. Therefore we took vectors which we termed meta-messages containing ADS-B information originating from individual aircrafts and we changed the altitude by adding to the original value a random number equal to $+$ or $-(800 + \text{random}(200))$ feet. Some altitude values are thus increased by adding a value between 800 and 1,000 ft, while others are decreased with a value in the same range.

Overall, for a given aircraft approximately one out of 30 vectors has an altered altitude value. The data used for training and testing respectively consist of meta-messages from 10 and 4 aircrafts ($66,172$ vectors for training and $19,827$ for testing). Since preexisting deep learning architectures are used for anomaly detection in our experiments, and there is no need for low level manipulation of neural networks, the Keras python library is used to build deep learning anomaly detection tools. For more general machine learning techniques such as gradient boosting the Sklearn python library is used.

This type of coarse punctual attack targeting the altitude value was chosen in a first step in order to identify the models most capable of detecting FDIAs. Indeed, if a model is not able to detect a coarse attack, it will not be able to detect finer attacks. Moreover, this allows to restrict the number of models to be studied in the context of other attack scenarios with finer alterations targeting other data.

4.3.1/ Data acquisition

The data used to evaluate the different models are issued from the OpenSky Network [75], a community-based receiver network which continuously collects air traffic surveillance data. In order to obtain the data needed for the anomaly detection, a data pipe based on several python scripts was developed in order to enable the querying of the OpenSky database. More precisely we downloaded raw ADS-B messages and decoded them using the traffic and pyModeS python libraries. Currently, to build our dataset we have used air traffic surveillance data collected on the 13th of January 2019 between 1PM and 2PM GMT near the swiss border.

4.3.2/ Data format

The ADS-B protocol relies mainly on 4 different message types: Aircraft Identification, Airborne Position, Airborne Velocity, and Surface Position. Each type of message contains its own information and is sent by the aircraft transponder in a fixed period of time (5 seconds for Identification messages and 0.5 seconds for the others). In this study, meta-messages were used for anomaly detection. They contain some ADS-B information more specifically the altitude, the ground speed, the track, the latitude and the longitude. Such content can be extracted from Airborne Position and Airborne Velocity messages.

4.3.3/ Confusion matrix

To assess the relevance of the different machine learning models we used confusion matrices to evaluate their accuracy results. A confusion matrix sums up the prediction

results of a classifier. Such a matrix is constructed as follows: each line corresponds to a real class and each column corresponds to a predicted class. For example, in our case where there are two classes (there is an attack / Positive — the altitude value has been altered — or not / Negative) and the matrix looks like:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix},$$

where, in the case of anomalous meta-messages detection,

- TN represents the number of True Negative predictions, in other words the number of correctly detected meta-messages without attack;
- FN, which stands for False Negative, counts the number of altered meta-messages which are classified as not attacked;
- TP means True Positive and thus represents the number of attacked meta-messages which were correctly detected;
- Finally, FP (for False Positive) corresponds to the number of meta-messages without alterations classified as attacked meta-messages.

The objective is then to find a classifier whose predictions produce a diagonal matrix.

In the context of the classification of meta-messages, we define an input sample as a sequence of meta-messages and call its size lookback. Each sample is then used to classify its last meta-message as normal or anomalous as shown in Figure 4.1.

As an illustration of application of the confusion matrix to evaluate the accuracy of a classifier, we present the evaluation of the XGBoost algorithm for anomaly detection. The evaluation of this algorithm is straightforward since XGBoost is directly available in python as a module. Using a lookback value of 50, the obtained confusion matrix is:

$$\begin{bmatrix} 14,400 & 4,557 \\ 17 & 657 \end{bmatrix}.$$

To obtain the number of samples used for testing, the following formula can be used: $S = T - N \times (L - 1)$, where S is the number of test samples effectively used to evaluate a classifier, T is the total number of meta-messages in the testing set, N is the number of aircrafts whose meta-messages are used for testing, and L is the lookback value. Using the previous formula, given the experiment setup ($T = 19,827$, $L = 50$, $N = 4$), the number of samples used to evaluate the XGBoost algorithm is equal to $19,631$. Note that the sum of the elements of the confusion matrix is also equal to the number of samples used for testing i.e. $19,631$.

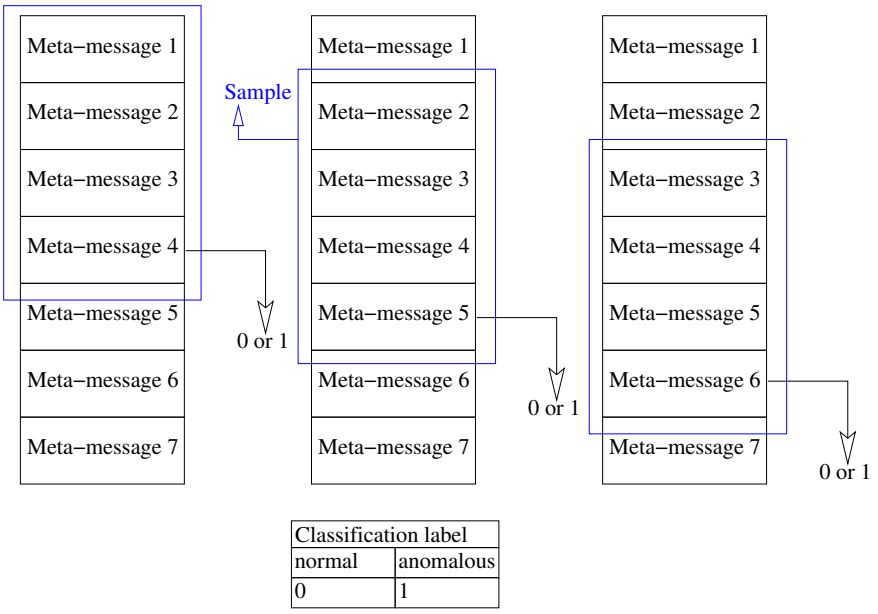


Figure 4.1: Visual example of the classification of the last meta-message of three successive samples, using a lookback (window size) of 4 meta-messages.

According to the confusion matrix, we can observe that the number of FP is very large implying a low precision (12.6 %), which means that XGBoost predictions are quite mitigate. In addition to XGBoost algorithm, we investigated different deep learning architectures.

4.3.4/ LSTM ARCHITECTURE EVALUATION

A stacked LSTM architecture with two layers has been considered. As we need to start with a fixed number of units in each layer, we have chosen 256 units in the first layer and 128 units in the second layer. Later we will see the influence of the LSTM architecture by considering different settings for the sizes of the two layers.

First, different optimizers have been compared: ADAM, ADAMAX, NADAM, RMSPROP, and the classical SGD. Note that an optimizer consists of an algorithm used to minimize the loss function. More precisely, it updates the weights of the network in order to minimize the prediction errors during learning. The parameters controlling the training are set as follows: the maximum number of epochs is set to $3,000$, an early stopping condition on the training loss error is set to $5\text{e}{-4}$, and a lookback value of 15 is chosen. Table 4.1, which presents the obtained respective Precision and Recall, as well as the F-score and the number of epochs needed to converge, shows that NADAM is the optimizer providing the better detection performance. Indeed it has the highest percentages, which means the best detection ability, and it is the fastest optimizer to converge because it requires only 312 epochs. Note that in this case the number of meta-messages (or samples) used

for testing is equal to $19,771$ while it was $19,631$ for XGBoost. This increase in the number of samples is explained by the lower lookback value (15 versus 50).

Table 4.1: Evaluation of different optimizers on a stacked LSTM (layers of 256 and 128 units - lookback value of 15).

Optimizer	%Precision	%Recall	%F-score	Number of epochs for convergence
ADAM	71.6	96.62	82.25	788
RMSPROP	55.1	84.88	66.82	2,223
SGD				did not converge
NADAM	82.6	97.65	89.50	312
ADAMAX	71.58	97.65	82.61	536

Focusing on NADAM, we can observe that the size of the lookback influences greatly the number of epochs to reach the convergence and the quality of the prediction. According to our experiments, as shown in Table 4.2, a lookback value of 35 allows to obtain the best results but it requires a larger number of epochs to converge. The sizes of the different layers appear to have also a great impact on the detection ability of the two-layer LSTM architecture. Table 4.3, which presents the percentages gained for different layer sizes configurations, highlights a best setting of 64 units for the first layer and 32 for the second one.

Table 4.2: Evaluation of different lookback values on a stacked LSTM (layers of 256 and 128 units - NADAM optimizer - *: early stopping loss set to $5e{-3}$).

Lookback	%Precision	%Recall	%F-score	Number of epochs for convergence
5*	91.91	96.63	94.21	685
10	76.48	98.38	86.06	1,635
15	80.05	98.38	88.27	314
20	81.67	97.79	89.01	338
25	76.90	97.05	85.81	598
30	83.83	98.53	90.59	288
35	93.04	98.82	95.84	1,871
40	86.57	98.37	92.09	1,477

Table 4.3: Evaluation of different stacked LSTM architectures (two layers - NADAM optimizer - lookback value of 20).

Layer (units)		%Precision	%Recall	%F-score	Number of epochs
1	2				for convergence
16	8				did not converge
32	16				did not converge
64	32	91.59	99.41	95.34	1,134
128	64	74.36	98.23	84.64	1,339
256	128	81.67	97.79	89.01	20,598

4.3.5/ BIDIRECTIONAL LSTM EVALUATION

Previously we have considered a unidirectional LSTM architecture, but it is possible to use a bidirectional architecture [33]. In that case, the network consists of two different hidden layers: one that processes the input sequence forward, like in the unidirectional architecture, whereas the other one processes it backward. Using a bidirectional LSTM made of two unidirectional LSTM layers (256 units in first layer and 128 units in second one) we obtained the following result with NADAM optimizer and lookback set to 20: 77.32 % Precision, 84.54 % Recall, and 79.94 % F-score after $1,531$ epochs. Clearly this kind of architecture is not interesting.

4.3.6/ CNN ARCHITECTURE

Convolutional neural networks have been used for many years to make very efficient image classifications. In our case study, considering a lookback value of 20 and the temporal data of an aircraft consisting of 5 different multivariate data series (altitude, latitude, etc.), we can see them as a 2D image of size 20×5. In practice, a convolutional layer followed by dense layers can be used. However, such a network converges very slowly and cannot reach the required precision in $3,000$ epochs. Nevertheless, it provides the following result: 84.46 % Precision, 89.69 % Recall, and 87 % F-score, which is not surprising because CNNs are not designed for this kind of classification

4.3.7/ USING CNN AND LSTM SIMULTANEOUSLY

It is possible to make one step of 1D CNN and then to feed its output in a LSTM. This kind of architecture is sometimes used with time series. As a case study, we evaluated an architecture made of 16 convolution kernels of size 3 which are entered into a 3-layer LSTM (256, 128, and 64 units respectively in layer 1, 2, and 3). The results of the evaluation are summarized in Table 4.4. Choosing a first layer of convolutional neurons provides good results.

Table 4.4: Evaluation of a 1D CNN connected to a stacked LSTM for different lookback values (three layers - NADAM optimizer).

Lookback	%Precision	%Recall	%F-score	Number of epochs for convergence
10	77.32	96.62	85.90	565
15	84.38	98.38	90.85	601
20	83.19	97.64	89.84	737

4.4/ CONCLUSION

We have presented a first evaluation of the ability of some machine learning models, and more particularly deep learning ones, to detect a rough FDIA consisting in the alteration of the altitude value. The eXtreme Gradient Boosting algorithm, the well-known LSTM architecture, its bidirectional variant and a combination with a layer of convolutional neurons in order to change the representation of the input data, and finally, a CNN by viewing the input data as an image, were evaluated. The experiments show that, as expected due to the temporal evolution of the data, the LSTM architecture appears to be the most suited for the considered problem.

Different optimizers, lookback values, and settings of the LSTM architecture have been compared. The best detection results have been obtained with a stacked LSTM of two layers composed of 64 units in the first one and 32 in the second, a lookback of 20 time steps, trained with NADAM optimizer. It should be noticed that these results are quite good considering that only 10 aircrafts have been used for the training.

This chapter allowed to draw lessons about the models by considering an attack scenario limited to altitude data. The following chapter goes further by considering various attack scenarios which alter windows of meta-messages instead of individual meta-messages. Such more advanced attacks are created using a false data generator which relies on a domain specific language.

5

CONTRIBUTION 2: SUPERVISED ADS-B ANOMALY DETECTION USING A FALSE DATA GENERATOR

5.1/ INTRODUCTION

In the previous chapter, in order to remedy the problem of lack of authentication and encryption in the ADS-B protocol, different architectures were evaluated to detect ADS-B anomalies in a supervised manner. However, only one type of attacks was examined, namely rough punctual attacks applied on the altitude. In this chapter, a supervised deep learning strategy is designed to detect various types of attacks that modify components of ADS-B messages such as altitude, ground speed, trajectory, latitude and longitude. Therefore a false data generator based on a domain specific language was used to attack ADS-B data and obtain a dataset containing both normal and anomalous data for supervised learning. The detection performance of two types of attacks were evaluated: gradual attacks and waypoints attacks that divert aircraft trajectories to pass through specific waypoints. The experimental results show that the proposed supervised deep learning strategy is able to recall on average 99 % of anomalies in ADS-B messages, mainly property modification attacks. A large part of this contribution was presented at the ICCCR 2022 conference [163].

This chapter is organized as follows:

- First, the detection of false data injection is discussed. Both data generation and detection process are detailed, specifically the false data injection framework used to obtained labeled messages.
- Second, a strategy for the detection of false data injection attacks is proposed.
- Third, the experimental results of the proposed detection strategy are described.

- Fourth, the effect of the number of flights used in training and testing on the detection performance is also examined.
- Fifth, a comparison between supervised and unsupervised anomaly detection is also carried out.
- Conclusion.

5.2/ False data injection framework (FDI-T)

As highlighted previously, anomaly detection in the literature uses only semi-supervised and unsupervised learning. In this chapter we propose a technique to detect anomalies using labeled messages (supervised anomaly detection) to train a deep learning model. These messages are obtained thanks to a software, called FDI-T (False Data Injection Test), designed by colleagues in our laboratory, allowing the generation of false air traffic data [112, 137, 161].

5.2.1/ Overview

This software was created to obtain testing capabilities of ADS-B attacks to ensure the additional security procedures envisioned by ATC authorities namely EURO-CONTROL. More concretely, this software implements an alteration process making it possible to define various scenario attacks according to alteration directives. Figure 5.1 gives an overview of the alteration process showing that it consists of 5 items and its mechanism is described in what follows.

In order to realize attacks on ADS-B data (first item), directives (or instructions) in a domain specific language (DSL) are used to describe the alterations to be performed on the original recorded data (second item). Each instruction is then processed by FDI-T (third item) to call the alteration engine in charge of the specified alteration (fourth item). Finally, the resulting altered data are recorded (fifth item).

A more extensive classification of FDI-T's components is comprised of the following elements: Data acquisition, Scenario Design, Radar sensors network Simulator, Alteration Engine and Execution Engine. The Internet or SSRs more specifically a Mode-S receiver, can be used to collect the required ADS-B recordings via the Data acquisition component. The sequence of ADS-B data in a recording is sorted by reception time. Then Scenarios Designs are created using a domain specific language which specifies the targeted aircraft as well as the time window for the alterations, and the needed triggering criteria. The output of the Scenarios Design component is called an alteration directive. Since each sensor captures ADS-B messages from aircraft located in a particular range, the

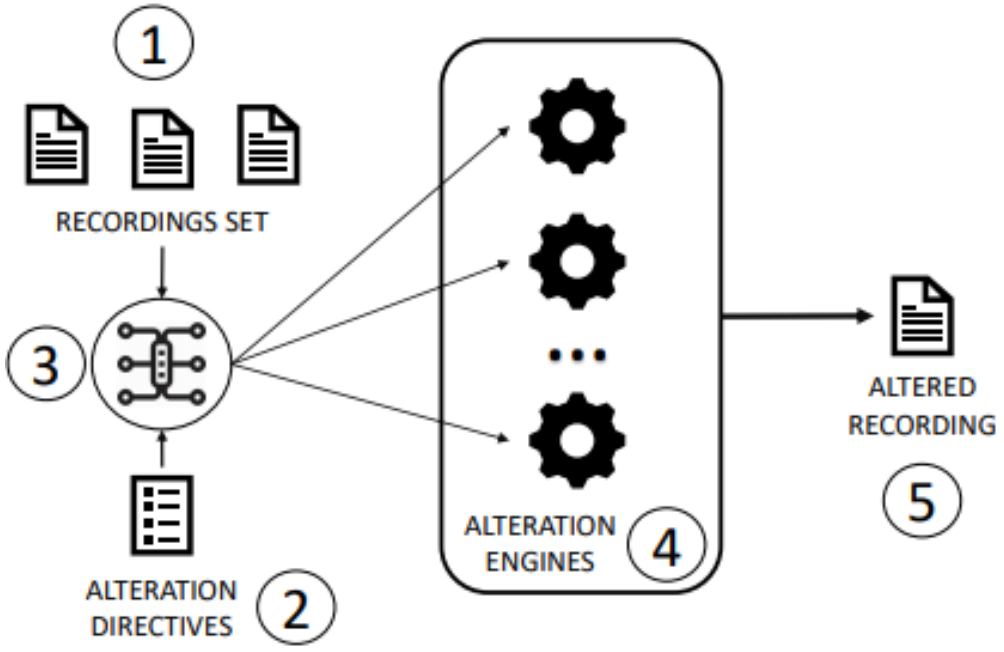


Figure 5.1: Alteration process overview - image drawn from [137].

radar sensors network simulator associates each recording to a sensor. The need for this module originates from the fact that the sensors are usually the components targeted by ADS-B attacks. Then, the Alteration Engine generates the designed attacks on the specified sensors altering subsequently their corresponding recordings. Finally the Execution Engine supplies the ATC with the modified recordings mimicking the mechanism of reception of normal ADS-B messages. The graphical user interface of FDI-T is shown in Figure 5.2.

In order for the authors of [161] to design the DSL language, they relied on ontologies due to their querying and reasoning capabilities as well as their verifiability as opposed

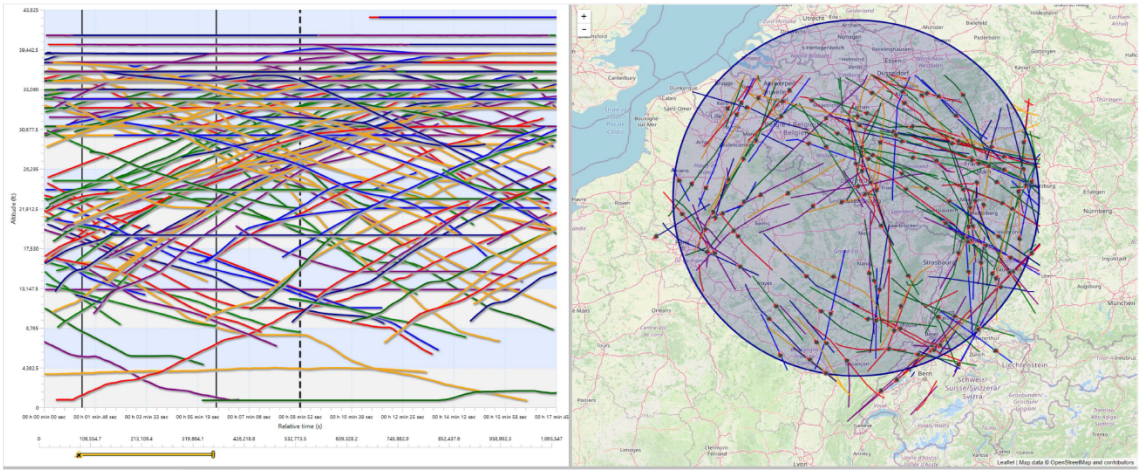


Figure 5.2: Graphical user interface of FDI-T visualizing many aircraft trajectories in 4D.

to informal design. An ontology can be defined informally as a set of notions comprising a set of relationships between them [20]. Note that there are also many rigorous domain analysis techniques which can potentially help in conceiving a DSL language, namely Domain Analysis and Reuse Environment [18], Domain Specific Software Architectures [15], Feature-Oriented Domain Analysis [10], Family-Oriented Abstractions, Specification, and Translation [17]. These techniques are unfortunately time consuming and complex and do not possess every kind of functionality needed for DSL conception [52].

The authors also followed some guidelines in the creation of the DSL which should be able to express all the known attacks. In addition, it needs the ability to implement a filtering mechanism to select the appropriate aircraft whose recordings should be altered. It should also allow the application of attacks based on specific triggers, such as exceeding a certain altitude or speed.

5.2.2/ DSL LANGUAGE

The DSL language which is based on ontologies is divided into multiple grammars: Base expression grammar, Property evaluation grammar, Filter grammar, Trigger grammar, Schema grammar.

The Base expression grammar is used to apply arithmetic expressions (addition , multiplication, division and modulo) on numbers as well as aircraft properties such as altitude. The Property evaluation grammar uses comparison operators (=, ! =, <, >, <=, >=) to evaluate properties conditions e.g.: $LATITUDE>46.22$. The location of aircraft in zones can also be evaluated. For instance, it can be tested if an aircraft is located in a prismatic zone delimited by an upper and lower bound altitude planes of 5600 and 8100 feet respectively and multiple 2D points expressed by pairs of latitudes and longitudes of (43.02°,51.09°), (42.87°,47.26°), (40.66°,53.11°) respectively. The previous test can be expressed using the grammar as follows: outside prism with_vertices (43.02,51.09), (42.87,47.26), (40.66,53.11) and altitude from 5600 to 8100.

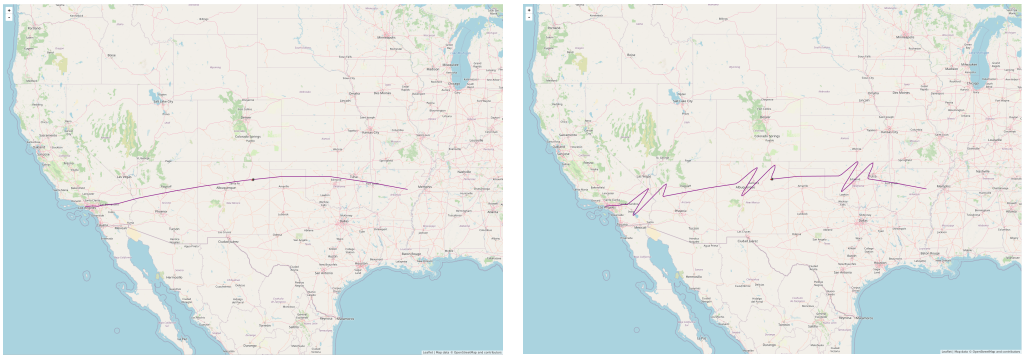
The filter grammar introduces ”eventually” and ”always” operators (respectively denoted by F and G). These operators must be applied to specific conditions in order to select a list of aircraft whose records should be altered. For example, suppose it is required to select aircraft whose altitude is not always higher then 30000 feet and whose altitude and longitude eventually go past 34000 feet and go below 2.3° respectively. Then the previous condition can be expressed as follows: eval not (G($ALTITUDE>30000$)) and F($ALTITUDE>34000$ and $LONGITUDE<2.30$).

The trigger grammar resembles the filter grammar, but it does not allow negation and it returns a schedule specifying when the attack is applied and when it is not. Here are two examples of writing a trigger in DSL: eval when($ALTITUDE>32000$) and eval as_-

soon_as(*LONGITUDE*>2.30). In the first example an alteration is applied only when the targeted aircraft's altitude is above 32000 feet whereas in the second example an attack is applied when the targeted aircraft's longitude is more than 2.3 degrees.

Lastly, using the schema grammar one can alter an existing aircraft recording using the alter keyword such as the DSL example alter all_planes at 0 seconds with_values LATITUDE ++= 0.0005. In this example the messages belonging to the totality of the recordings in a list of recordings are gradually altered. In other words 0.0005 is added to the first message, 2x0.0005 to the second and 3x0.0005 to the third, etc. Such an attack is called **gradual attack**. Ghost aircraft can also be created using the create keyword. One can for example create plane messages of ICAO = "39AC47" from 12 seconds until 251 seconds while having these recordings correspond to a trajectory passing by two specific waypoints. Such an attack can be expressed in DSL as follows:
create plane from 12 seconds until 251 seconds with_values
ICAO = "39AC47" and with_waypoints
[(24.85,53.23) with_altitude 4000 at 12 seconds ,
(24.62,53.94) with_altitude 4250 at 251 seconds]

Note that attacks using waypoints are not bound to attacks which create ghost aircraft. They can also be used to alter the trajectory of existing flights such as the flight shown in Figure 5.3 which connects two US cities. This type of alteration is called **waypoints attack** and it constitutes an alteration in which the trajectory is deviated (using an Akima interpolation [8]) to pass by specific selected points with known latitude, longitude and altitude.



(a) Before the attack (b) After a waypoints attack

Figure 5.3: Screenshots of the ADS-B false data injection software showing an example of a waypoints attack.

The use of a DSL language has many advantages. It mainly facilitates the implementation of test attacks directly by ATCS instead of relying on developers for every type of attack. In fact, the DSL syntax is akin to natural language and relies on ATC jargon making it appropriate for generating FDIAs test scenarios. Also, it helps automating the test attacks which are often repetitive. This saves ATCS and developers precious time and eliminates

any need for a previously required recording pre-analysis (namely identifying the right messages to alter). But more importantly for our case, the significance of this framework lies in its ability to be an extensive source of labeled normal and anomalous data. Such labeled data is necessary to train a supervised ADS-B anomaly detector which represents the contribution of this chapter.

5.3/ DETECTION OF FALSE DATA INJECTION

5.3.1/ GENERATION OF LABELED ATTACKED ADS-B MESSAGES

First, ADS-B messages are downloaded from the OpenSky Network (opensky-network.org) [75]. For the sake of having balanced datasets (50 % attacked messages, 50 % not attacked messages) and using the FDI-T software, half of the recordings are totally attacked and the other half is not touched at all. Note that FDI-T has also been used to acquire ADS-B datasets to test anomaly detection artificial intelligence models in another work done in our laboratory [136, 162]. In our work two types of attacks were considered for anomaly detection: gradual attacks and waypoints attacks.

For the case of gradual attacks detection, attacks are separately inflicted on altitude, ground speed, track, latitude and longitude, i.e.: gradual attack of 75 feet, 1.8 knots, 0.9 degrees, 4.88×10^{-3} degrees, 1.28×10^{-2} degrees for the altitude, ground speed, track, latitude, longitude respectively. These values are computed by finding the mean difference of the features in non attacked messages. 4.88×10^{-3} degrees of latitude correspond to 542.63 meters, while 1.28×10^{-2} degrees of longitude correspond to 906.10 meters which is the distance average over all latitude values.

The following DSL instruction was used for the gradual attack of $\Delta x = 75$ feet targeting the altitude:

alter all_planes at 0 seconds with_values ALTITUDE ++= 75.

The other features were gradually attacked in the same manner. The other type of attack tested for detection is the waypoints attack. The waypoints attack used in our study is presented in detail in Figure 5.4.

5.3.2/ META-MESSAGES GENERATION AND DETECTION

In order to detect attacks, the process described in Figure 5.5 is followed. First ADS-B messages corresponding to 120 randomly chosen flights are downloaded from the Open-Sky Network. These flights are not restricted to a specific area, they are dispersed all over

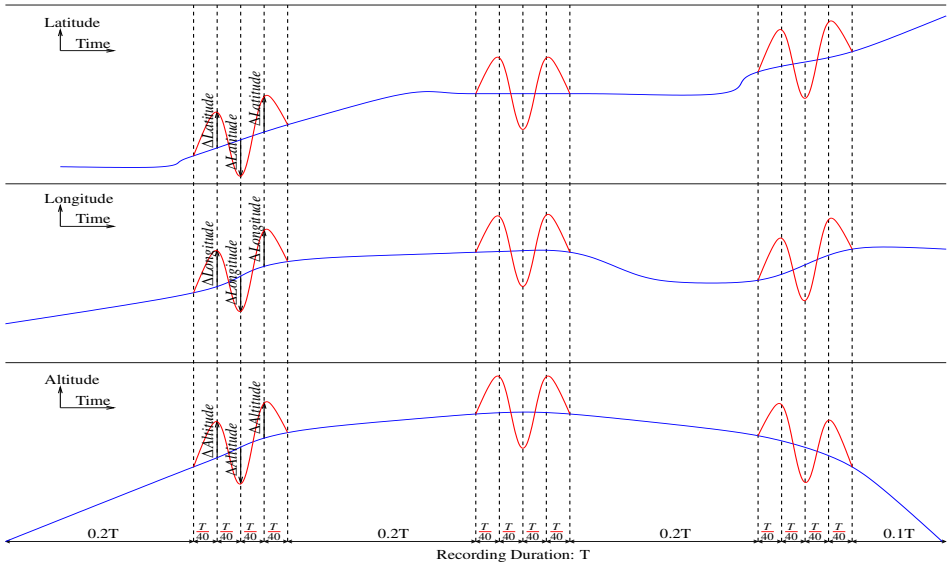


Figure 5.4: Waypoints attack where $\Delta Latitude = 4.88 \times 10^{-3}$ degrees, $\Delta Longitude = 1.28 \times 10^{-2}$ degrees, $\Delta Altitude = 75$ feet.

the globe and contain all of the different phases of flight: take off, climb, cruise, descent and landing. Since the features that will be used for anomaly detection are distributed among multiple categories of messages (Identification, Position and Velocity messages), these messages are combined into meta-messages as can be seen in Figure 5.6.

In other terms, messages are vectors with missing data and the process of combining consecutive different types of messages into meta-messages is just filling missing features from previous messages. Then the difference for all the meta-messages between two consecutive meta-messages is computed as seen in Figure 5.7. This process is applied to all the flight data gathered in the first step. Among the obtained differences of meta-messages from the 120 flights of the dataset, the messages of 100 flights are used to train a model to detect ADS-B anomalies and the remaining 20 flights to test it.

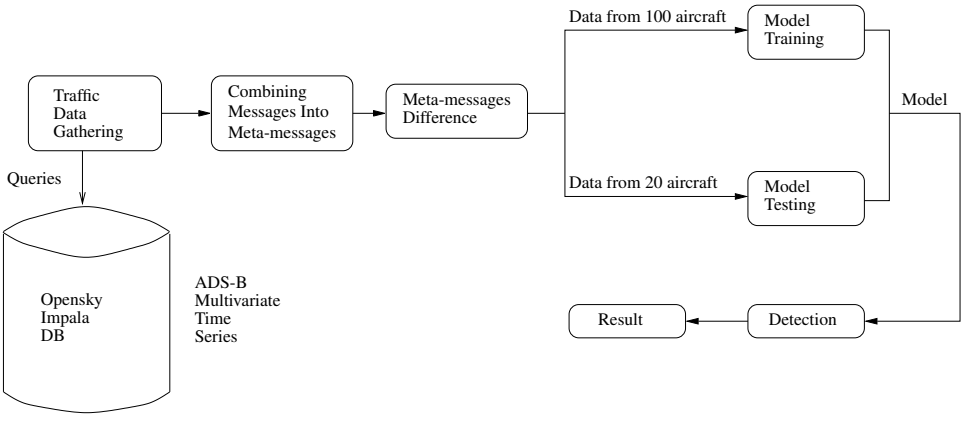


Figure 5.5: View of the whole detection process.

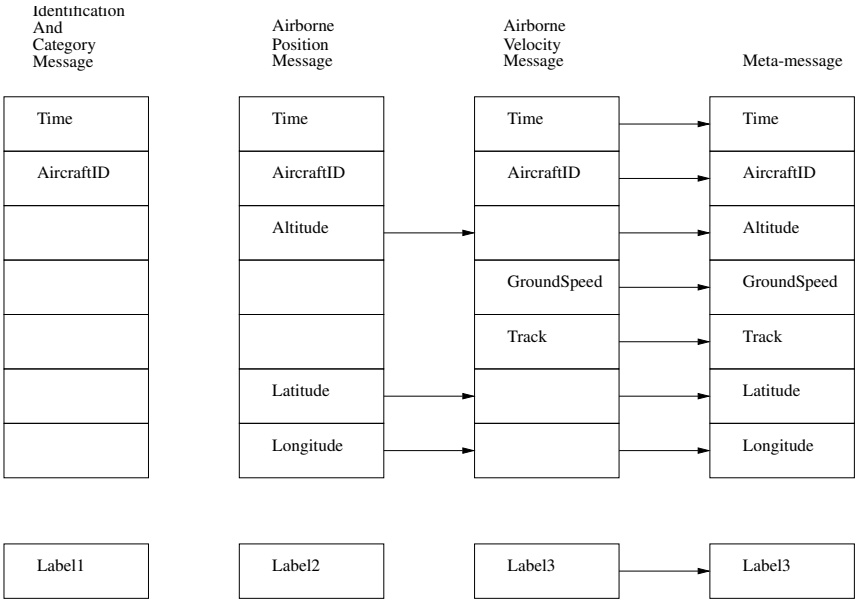


Figure 5.6: Example of combining messages into a meta-message.

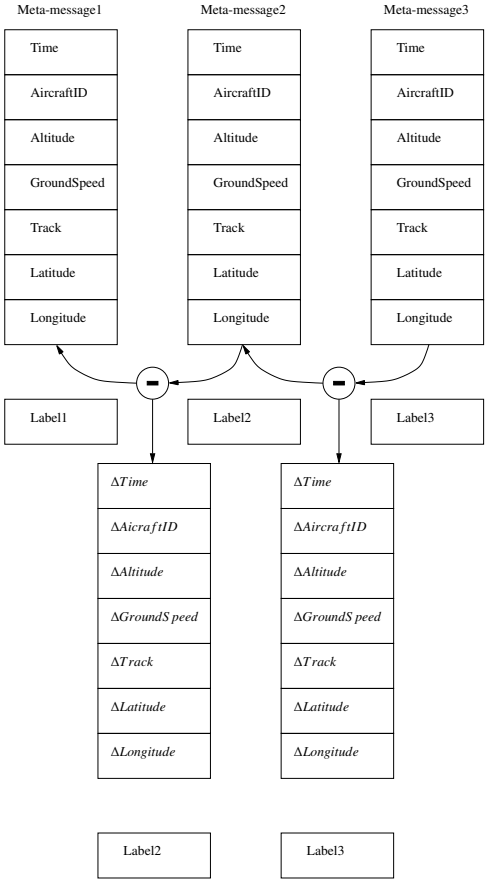


Figure 5.7: Difference of meta-messages.

Table 5.1: Evaluation of the stacked LSTM using different numbers of epochs (layers of 64 and 32 units - lookback value of 10).

Epochs	%Precision	%Recall	%F-score
10	99.89	99.70	99.80
50	99.94	99.65	99.79
100	99.94	99.82	99.88

Finally for the detection of anomalies a LSTM is trained. The input of our LSTM is a look back window of differences of meta-messages, where the number of time steps is defined by the lookback value. Its output is one dense neuron with a linear activation function which captures the state of the window, i.e. either a normal or an attacked window.

The choice of the LSTM architecture used in this study is based on the result of chapter 4 which focused only on the detection of random point changes in altitude. Indeed this work concluded that an LSTM containing intermediate layers of 64 units and 32 units, trained with a NADAM optimizer, gives the best performance for ADS-B anomaly detection compared to other architectures such as the Bidirectional LSTM, 1D Convolutional Neural Network, and so on. Note that in our case, scaling was not applied on the data using our approach.

5.4/ EXPERIMENTAL RESULTS

In order to train and test the LSTM model on GPUs, the supercomputer facilities of the “Mésocentre de Franche-Comté” were used. Our code is based on the Keras Python library. First the architecture is trained for 10 epochs, 50 epochs and 100 epochs in order to test the effect of the number of epochs on the detection performance as seen in Table 5.1. The attack used for this assessment is a gradual attack of the longitude equal to 1.28×10^{-2} degrees.

Since the performance did not increase considerably, 10 epochs can be enough to have a good detection performance for the following tests while having the shorter training time. Indeed the training time is proportional to the number of epochs and an epoch takes 14 seconds to be completed (for example it takes 2.33 minutes for 10 epochs). The difference between training times is not negligible especially due to the following need to also train models for the gradual attacks targeting other features. Lookback values of 5, 10 and 20 were tested in the detection of average gradual attacks. The results are summarized in Table 5.2. It can be noticed that when the lookback value is increased, most of the time the F-score also increases. However this gain in F-score is minor. For this reason a lookback value of 10 was used for the following tests balancing in this manner the training time and the detection performance. To compare the gradual attack detection

Table 5.2: Evaluation of the stacked LSTM using different lookbacks for the detection of gradual attacks.

Feature attacked	Lookback	%Precision	%Recall	%F-score
altitude	5	99.35	99.43	99.39
	10	99.63	99.76	99.69
	20	99.93	99.69	99.81
ground speed	5	98.86	99.48	99.17
	10	99.23	99.60	99.42
	20	99.29	99.68	99.49
track	5	96.91	99.15	98.02
	10	97.46	99.45	98.44
	20	97.52	99.50	98.49
latitude	5	99.90	98.18	99.03
	10	99.67	98.91	99.29
	20	99.72	98.97	99.34
longitude	5	99.83	99.01	99.41
	10	99.89	99.70	99.80
	20	99.60	98.95	99.27

Table 5.3: Detection performance compared to Habler & Shabtai (Moscow Dataset) for 400 feet gradual altitude attacks.

	%TPR	%FPR
Proposal	99.97	2.95×10^{-2}
E. Habler and A. Shabtai	59.04	3.87

with other previous works [114], the altitude was attacked gradually by 400 feet. Our detection performance in terms of True Positive Rate (TPR) and False Positive Rate (FPR) compared to the best performance obtained in [114] using one detection window with a lookback equal to 15 is summarized in Table 5.3. FPR, also called Fallout, represents the proportion of detected attacked messages that are in reality not attacked [145]. It is clear that using our technique, a considerable detection performance improvement is obtained (40.93 increase in %TPR and 3.84 decrease in %FPR).

Now a meta-model is tested. It uses the different models trained separately and detects the presence of an attack if at least one of the previously mentioned features (altitude, ground speed, etc.) is attacked. The performance of this meta-model is summarized in Table 5.4. The F-score obtained is 96.09 % at worse with a Precision of 93.15 % which is still good but a little worse than individual models as seen in the previous tables. The reason for the decrease in F-score is the slight loss of Precision which is caused by the False Positives of the different models adding up. In order to remedy the loss of Precision, we then considered an attack detected if a certain percentage of windows (and above) from a sequence of windows is attacked: the obtained performance is summarized in Table 5.5. Note that it was considered that a given percentage of attacked windows

Table 5.4: Evaluation of a meta-model for the detection of gradual attacks.

Feature attacked	%Precision	%Recall	%F-score
altitude	93.58	99.80	96.59
ground speed	96.50	99.70	98.07
track	96.49	99.53	97.99
latitude	93.15	99.22	96.09
longitude	93.18	99.77	96.36
all features	96.97	100	98.46

Table 5.5: Evaluation of a meta-model for the detection of gradual attacks using sequences of 100 windows, where an attack is detected if at least 95 of them are attacked.

Feature attacked	%Precision	%Recall	%F-score
altitude	99.49	98.85	99.17
ground speed	99.73	98.57	99.15
track	99.72	97.14	98.41
latitude	99.44	95.02	97.18
longitude	99.46	98.73	99.09
all features	99.76	100.0	99.88

(95 % and above) from a sequence of windows need to be attacked instead of the whole sequence to prevent eventual False Negatives from deteriorating the Recall. Using this technique, the Precision becomes at worse 99.44 % instead of 93.15 % and the F-score 97.18 % instead of 96.09 %. Although a better Precision was obtained, there was a small decrease in Recall which highlights a slight trade-off between Recall and Precision.

For the waypoints attack as depicted in Figure 5.4, using the same technique, a good detection performance was also reached: Precision=98.77 %, Recall=97.39 %, F-score=98.08 %. An example of a waypoints attack showing the real scale of the attack is shown in the Figure 5.8. Note that, all the results presented so far were obtained using one training set and one testing set for each result. This way of obtaining the performance is not always enough since deep learning models can give unstable results. For this reason a stratified 6-fold cross-validation was applied whose results are shown in Table 5.6. K-fold cross-validation is a method used to evaluate a model's ability to generalize on new data in a stable fashion. It is applied by dividing a dataset in k groups called folds. Each combination of k-1 folds and a remaining fold are used to respectively train and test the model. A cross-validation method is called stratified if each fold contains a balanced amount of classes and in our case a balanced amount of altered and altered data. This table shows the mean and standard deviation of the Precision, Recall and F-score for gradual and waypoints attacks. The mean F-score is in the order of 99 % and its worst standard deviation is 0.4420 hence our technique is stable enough to detect the previously mentioned anomalies.

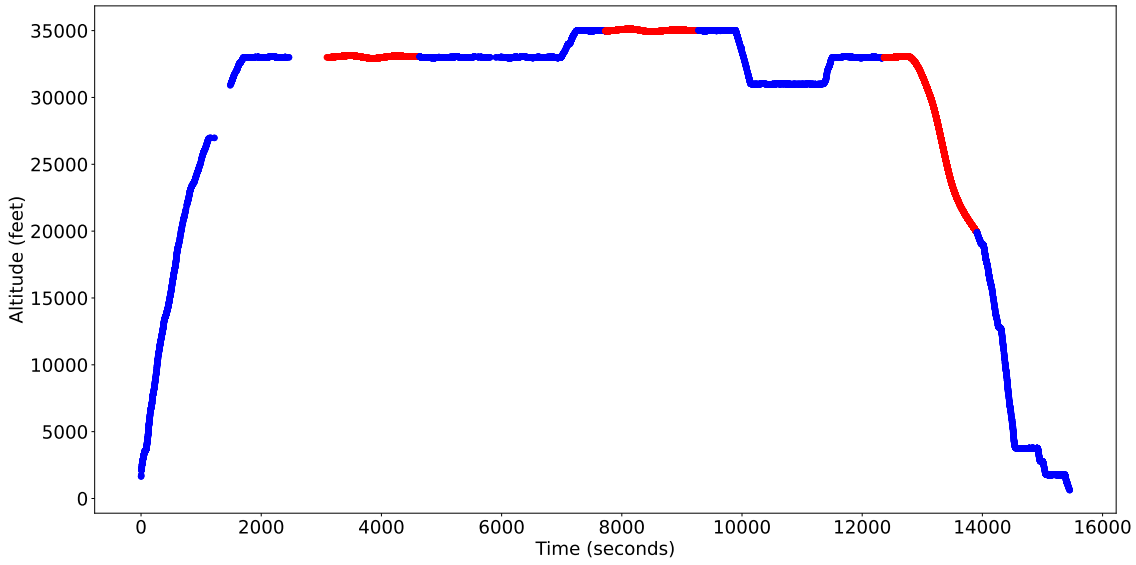


Figure 5.8: A plot showing the actual scale of a waypoints attack (in red) where $\Delta altitude = 75$ feet.

Table 5.6: Evaluation of the stacked LSTM using a stratified 6-fold cross-validation.

Feature attacked	%Precision	%Recall	%F-score
altitude	99.92 ± 0.0130	99.83 ± 0.0294	99.88 ± 0.0114
ground speed	99.77 ± 0.0301	99.74 ± 0.0178	99.76 ± 0.0120
track	99.45 ± 0.0941	99.41 ± 0.1680	99.43 ± 0.0648
latitude	99.41 ± 0.3280	99.26 ± 0.5940	99.34 ± 0.4420
longitude	99.89 ± 0.0482	99.66 ± 0.1320	99.78 ± 0.0728
waypoints - Figure 5.4	98.42 ± 2.23	99.08 ± 0.154	98.73 ± 1.08

5.5/ IMPACT OF THE NUMBER OF FLIGHTS ON THE DETECTION PERFORMANCE

In order to test the impact of the number of flights on the detection performance, two training, validation and testing configurations were used. The first configuration contains 120 flights in total, divided into 90 flights for training, 10 for validation and 20 for testing. The second configuration contains 1000 flights in total, divided into 600 flights for training, 100 for validation and 300 for testing.

Half of the flights of the training and testing sets are not touched and the other half is attacked using a general alteration. A general alteration can be described as follows: the half of messages that are attacked is divided into seven equal subsets where each subset is altered by a different type of attack (gradual altitude attack, gradual ground speed attack, gradual track attack, gradual latitude attack, gradual longitude attack and waypoints attack). The detection performance for the two configurations is summarized in Table 5.7. It can be observed that the detection performance as described by the F-score does not

Table 5.7: Impact of the number of flights used in training and testing on the detection performance. The attack used is a general alteration.
-*: the number of flights in the training set
-**: the number of flights in the validation set
-***: the number of flights in the testing set

Number of flights	Epochs	%Accuracy	%Precision	%Recall	%F-score
90*/10**/20***	10	86.11	88.11	91.71	89.88
	20	87.33	88.98	92.55	90.73
	100	85.39	89.33	89.66	89.50
600/100/300	10	90.53	89.54	94.84	92.11
	20	93.63	90.43	97.46	93.81
	100	94.85	93.79	97.78	95.74

vary that much for the configuration containing less flights. However, as the number of flights increases, the effect of increasing epochs is more apparent. More concretely, the F-score increases from 92.11 % for 10 epochs to 95.74 % for 100 epochs. Therefore, increasing the amount of data used to train supervised ADS-B anomaly detection has the potential to improve the detection performance.

5.6/ COMPARISON BETWEEN SUPERVISED AND UNSUPERVISED ANOMALY DETECTION

5.6.1/ DETECTION ANALYSIS

In order to show the benefits of using supervised learning over semi-supervised learning, a comparison of multiple detection approaches was carried out. For the semi-supervised approaches, an LSTM autoencoder architecture was tested as well as an LSTM forecasting approach. In the LSTM forecasting approach, a window of meta-messages is used to forecast the following meta-message. Then, the residual between the prediction and the real meta-message is used to classify the meta-message as normal or anomalous. The forecasting approach is based on the work in [147]. The LSTM used for this approach contains intermediate layers of 64 units and 32 units. Also the training dataset was reused to compute the threshold of anomaly detection as opposed to [147] whose authors used a separate dataset to determine the threshold. The autoencoder architecture used the same approach as [114]. Nevertheless the autoencoder is made of a stacked LSTM with an intermediate layer of 128 units (from the encoder side), then a bottleneck layer of 64 units followed finally by a layer of 128 units (from the decoder side). In the supervised approach, the transformer architecture used a minimal configuration that relies on one attention head and is made of one layer. Indeed, changing its configuration did not change its performance (or only slightly). The supervised LSTM approach contains intermediate

layers of 64 units and 32 units. The equivalent of 90 and 10 aircrafts was used for training and validation, respectively, whereas 20 aircrafts were used for testing. Tables 5.8 and 5.10 show that regardless if the difference on successive meta-messages is applied or not, the supervised approaches give the best performance overall. If the difference on successive meta-messages is applied, the LSTM beats the transformer by 0.13 % in F-score, otherwise the transformer beats the LSTM by 1.03 % in F-score. Nevertheless, the highest F-scores are obtained by applying the difference on successive meta-messages and using supervised approaches. The LSTM takes the lead in that case. Note that applying difference deteriorates the semi-supervised approaches' results totally as shown in Table 5.10. Also, scaling was required and used for unsupervised approaches as opposed to supervised approaches which did not use scaling.

5.6.2/ Alarm evaluation

Using a sequence of windows of meta-messages for detecting attacks can improve the precision as previously shown in Table 5.5. However, in practice, it might be necessary to launch an alarm after detecting a sufficient number of anomalous meta-messages to prove that these detections are not accidental. The use of an alarm based on fixed size meta-messages sequence akin to Table 5.5 might hamper the flexibility of the alarm mechanism. Therefore, in order to diminish the amount of false positives, we chose to launch an alarm for a given flight if the number of detected anomalies exceeds a specified threshold, in our case 100 meta-messages. This alarm mechanism is shown in Figure 5.9 and its performance is shown in Tables 5.9 and 5.11.

The LSTM supervised approach is the most viable option since all of the attacked test flights for most types of attacks (altitude, latitude, longitude, waypoints) were correctly classified as attacked. In order to correctly classify the remaining flights the alarm threshold can be increased to 400, 700 for ground speed, track respectively. In the case of a threshold equal to 700 the lookback value is changed from 10 to 100 and the precision is thus 100 % but the recall is 90 %. Nevertheless, this performance is identical to a theoretical perfect alarm which relies on a flawless hypothetical anomaly detector. This theoretical anomaly detector never fails to properly identify the anomalies in sequences of meta-messages.

5.7/ Conclusion

A supervised deep learning strategy to detect false data injection attacks aiming at altering properties of messages in the ADS-B protocol has been presented. This strategy was evaluated for the identification of gradual attacks and waypoints attacks created us-

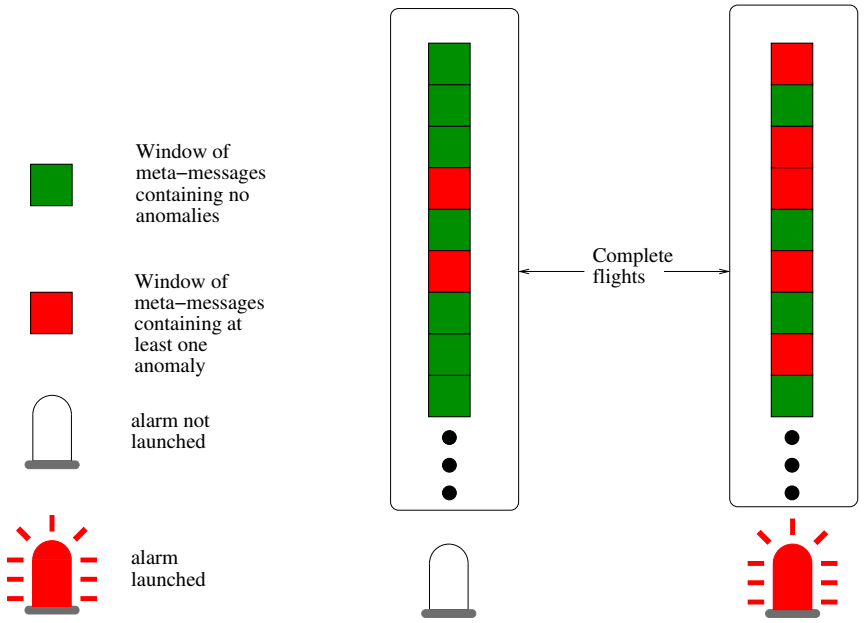


Figure 5.9: Illustrated example of the alarm mechanism. In this example the chosen threshold is 4, therefore since in the flight on the left only 2 anomalous windows are detected till present (number of detected anomalous windows<threshold), the alarm is not launched. The flight on the right contains at least 5 anomalous windows (number of detected anomalous windows>threshold), thus the alarm is launched.

ing a false data generator. The effects of the number of training epochs and the lookback value on the detection performance were tested. It was found that only a minor improvement of the detection performance results from increasing the number of training epochs and the lookback value. We were able to detect gradual attacks of the altitude, ground speed, track, latitude and longitude using individual models trained on each individual attack, as well as using one meta-model made of these individual models. In addition to that the strategy was successful in detecting waypoints attacks. Finally the proposed supervised strategy was compared to other semi-supervised techniques showing a better performance in the case of the supervised approach. An alarm mechanism was also proposed to prevent accidental anomalous flight detections.

Table 5.8: Evaluation of detection for different types of attacks without applying difference on successive meta messages.

Attack	Model Metrics	Autoencoder	LSTM Forecasting	LSTM	Transformer
Altitude	Accuracy	85.00	93.00	98.00	99.00
	Recall	75.70	88.62	97.03	97.71
	Precision	99.90	99.84	99.98	100.00
	F-score	86.13	93.89	98.49	98.84
Ground speed	Accuracy	91.00	97.00	99.00	99.00
	Recall	88.37	96.59	99.29	99.64
	Precision	99.95	99.92	99.15	99.07
	F-score	93.81	98.23	99.22	99.36
Latitude	Accuracy	39.00	52.00	61.00	63.00
	Recall	0.05	22.06	100.00	91.68
	Precision	38.75	99.30	61.11	63.60
	F-score	0.11	36.09	75.86	75.10
Longitude	Accuracy	39.00	41.00	61.00	86.00
	Recall	0.08	3.31	100.00	99.96
	Precision	47.87	95.53	61.11	81.22
	F-score	0.15	6.39	75.86	89.62
Track	Accuracy	87.00	96.00	92.00	98.00
	Recall	82.66	95.29	89.19	98.76
	Precision	99.95	99.92	100.00	99.00
	F-score	90.49	97.55	94.29	98.88
Waypoints	Accuracy	63.00	63.00	72.00	65.00
	Recall	0.34	1.27	83.38	61.71
	Precision	64.16	54.28	58.16	52.33
	F-score	0.68	2.48	68.52	56.63
Total	Accuracy	67.33	73.66	80.50	85.00
	Recall	41.20	51.19	94.81	91.57
	Precision	75.09	91.46	79.91	82.53
	F-score	45.22	55.77	85.37	86.40

Table 5.9: Evaluation of alarm for different types of attacks without applying difference on successive meta messages in the anomaly detection step (threshold for 100 cumulative detections).

Attack	Model Metrics	Autoencoder	LSTM Forecasting	LSTM	Transformer
Altitude	Accuracy	90.00	95.00	100.00	95.00
	Recall	80.00	90.00	100.00	90.00
	Precision	100.00	100.00	100.00	100.00
	F-score	88.89	94.74	100.00	94.74
Ground speed	Accuracy	95.00	95.00	85.00	85.00
	Recall	90.00	90.00	100.00	100.00
	Precision	100.00	100.00	76.92	76.92
	F-score	94.74	94.74	86.96	86.96
Latitude	Accuracy	50.00	60.00	50.00	50.00
	Recall	0.00	20.00	100.00	100.00
	Precision	NaN	100.00	50.00	50.00
	F-score	NaN	33.33	66.67	66.67
Longitude	Accuracy	50.00	55.00	50.00	60.00
	Recall	0.00	10.00	100.00	100.00
	Precision	NaN	100.00	50.00	55.56
	F-score	NaN	18.18	66.67	71.43
Track	Accuracy	90.00	95.00	85.00	85.00
	Recall	80.00	90.00	70.00	100.00
	Precision	100.00	100.00	100.00	76.92
	F-score	88.89	94.74	82.35	86.96
Waypoints	Accuracy	60.00	65.00	65.00	55.00
	Recall	11.11	22.22	66.67	88.89
	Precision	100.00	100.00	60.00	50.00
	F-score	20.00	36.36	63.16	64.00
Total	Accuracy	72.50	77.50	72.50	71.67
	Recall	43.52	53.70	89.44	96.48
	Precision	100.00	100.00	72.82	68.23
	F-score	73.13	62.01	77.64	78.46

Table 5.10: Evaluation of detection for different types of attacks while applying difference on successive meta messages.

Attack	Model Metrics	Autoencoder	LSTM Forecasting	LSTM	Transformer
Altitude	Accuracy	37.0	38.00	100.00	99.00
	Recall	0.0	0.35	99.70	99.41
	Precision	NaN	79.56	99.71	99.54
	F-score	NaN	0.69	99.71	99.48
Ground speed	Accuracy	24.0	24.00	99.00	100.00
	Recall	0.0	0.21	99.61	99.76
	Precision	NaN	81.76	99.41	99.66
	F-score	NaN	0.42	99.51	99.71
Latitude	Accuracy	39.0	39.00	99.00	99.00
	Recall	0.0	0.35	98.91	98.97
	Precision	NaN	78.63	99.91	99.94
	F-score	NaN	0.70	99.41	99.45
Longitude	Accuracy	39.0	39.00	100.00	99.00
	Recall	0.0	0.35	99.76	99.08
	Precision	NaN	78.71	99.91	99.82
	F-score	NaN	0.70	99.84	99.45
Track	Accuracy	24.0	24.00	98.00	97.00
	Recall	0.0	0.21	99.21	99.44
	Precision	NaN	81.33	97.59	96.63
	F-score	NaN	0.41	98.39	98.01
Waypoints	Accuracy	63.0	63.00	98.00	98.00
	Recall	0.0	0.48	95.18	95.62
	Precision	NaN	47.03	98.91	98.43
	F-score	NaN	0.96	97.01	97.01
Total	Accuracy	37.67	37.83	99.00	98.67
	Recall	0.00	0.32	98.73	98.71
	Precision	NaN	74.50	99.24	99.00
	F-score	NaN	0.65	98.98	98.85

Table 5.11: Evaluation of alarm for different types of attacks while applying difference on successive meta messages in the anomaly detection step (threshold for 100 cumulative detections).

Attack	Model Metrics	Autoencoder	LSTM Forecasting	LSTM	Transformer
Altitude	Accuracy	50.0	55.00	100.00	95.00
	Recall	0.0	10.00	100.00	100.00
	Precision	NaN	100.00	100.00	90.91
	F-score	NaN	18.18	100.00	95.24
Ground speed	Accuracy	50.0	55.00	90.00	95.00
	Recall	0.0	10.00	100.00	100.00
	Precision	NaN	100.00	83.33	90.91
	F-score	NaN	18.18	90.91	95.24
Latitude	Accuracy	50.0	55.00	100.00	100.00
	Recall	0.0	10.00	100.00	100.00
	Precision	NaN	100.00	100.00	100.00
	F-score	NaN	18.18	100.00	100.00
Longitude	Accuracy	50.0	55.00	100.00	100.00
	Recall	0.0	10.00	100.00	100.00
	Precision	NaN	100.00	100.00	100.00
	F-score	NaN	18.18	100.00	100.00
Track	Accuracy	50.0	55.00	65.00	60.00
	Recall	0.0	10.00	100.00	100.00
	Precision	NaN	100.00	58.82	55.56
	F-score	NaN	18.18	74.07	71.43
Waypoints	Accuracy	55.0	65.00	100.00	100.00
	Recall	0.0	22.22	100.00	100.00
	Precision	NaN	100.00	100.00	100.00
	F-score	NaN	36.36	100.00	100.00
Total	Accuracy	50.83	56.67	92.50	91.67
	Recall	0.00	12.04	100.00	100.00
	Precision	NaN	100.00	90.36	89.56
	F-score	NaN	21.21	94.16	93.65

6

CONTRIBUTION 3: DEEP LEARNING AND GRADIENT BOOSTING FOR URBAN ENVIRONMENTAL NOISE MONITORING IN SMART CITIES

6.1/ INTRODUCTION

Every day the innovative IoT technology is expanding further and further in our environment, with applications deployed in various contexts including cities. Communities can indeed address problems linked to urbanization thanks to this technology through the Smart City concept and thus support a sustainable development of their cities. Artificial intelligence and namely its machine learning branch is expected to reinforce this trend by making smart cities even smarter. However, smart cities can only be successful if they can be trusted and, with this in mind, machine learning can potentially be an efficient tool to mitigate cyberattacks. In order to remedy such problems, more specifically in the urban noise monitoring context, the ability of Gradient Boosting and Deep Learning to make long-term predictions of noise level is examined. This study is based on noise data collected in the suburb of an English city. The noise predictions were then used to evaluate the possibility of detecting anomalous data resulting from malicious injections. The obtained results show the relevance of a deep learning architecture. This contribution has been submitted to the international journal Expert Systems with Applications (ESWA) and is currently under review.

This chapter is organized as follows:

- IoT and smart cities. Some IoT applications in smart cites and the role of machine learning related to those applications are discussed.

- Noise prediction approaches. Some related works concerning noise prediction using deep learning are described.
- Materials and methods. The materials and methods used are presented.
- Results. The noise level forecasting performance of different machine learning and deep learning architectures is evaluated. The use of CNN-LSTM architectures for punctual noise level anomaly detection is investigated as well.
- Conclusion.

6.2/ IoT AND SMART CITIES

Based on the advent of the Internet of Things (IoT), a system interconnecting computing devices, digital machines, and objects through a network, many innovative solutions have emerged in order to solve problems touching a wide range of domains. Indeed, the deployment of such IoT devices can make our environment smarter. The IoT can, for example, help us in our daily life at home by monitoring the lighting, media or security systems and so on. It can also provide better transportation conditions or improve manufacturing processes.

Among the possible applications of the IoT, the concept of Smart City addresses urban problems in order to encourage the rise of more citizen-friendly cities [121]. Today, more than half of the world's population lives in cities and according to a United Nations forecast this number will continue to increase greatly in the future, reaching an expected proportion of 68 % by 2050 [131]. Obviously, as the resident population of a city increases, many aspects of the city are impacted, namely the energy and water distribution, the transportation system, environment, etc. Therefore municipal governments need a framework to help them face these urbanization challenges, make the best decisions for a sustainable development of the city and to improve the quality of life. Fortunately, the combination of IoT technology and artificial intelligence allows to develop a fully functional Smart City providing such a framework. On the one hand the IoT infrastructure collects data from connected objects and machines, while, on the other hand, artificial intelligence extracts information from a large database.

Many cities throughout the world have already successfully deployed the IoT technology, giving an open access to data to promote the innovative development of solutions by public or private parties. For example, regarding urban mobility challenges, a congestion management system called Midtown in Motion [69] was deployed in New York City, while real-time traffic and transportation data are free of access in several cities like Amsterdam or Copenhagen [141]. Artificial intelligence and more particularly the machine

learning technology can convert these data into insights and is thus a key to expand capabilities of smart cities [119, 139]. Typically, machine learning allows to make useful predictions when making decisions or detecting problems. As an illustration, considering another urban challenge, air and noise pollution control, long-term predictions can provide city administrators with the necessary information to choose the best transportation investments to possibly tackle these problems in the future, while real-time short-term predictions could enable an immediate action on the motor vehicle traffic congestion. This work focuses on noise pollution, which according to European Environment Agency is a growing problem [138], a problem whose impact on people's health is not fully measured right now.

6.3/ NOISE PREDICTION APPROACHES

In the literature, many articles broach the subject of noise prediction using deep learning.

In [149] a noise monitoring system was set up to obtain noise data (via a noise sensor) in specific urban locations. This system is powered by a solar panel and uses a Zigbee communication module which transfers the data to be stored to the network coordinator. Basic Long-Short Term Models (LSTM) were also used to forecast noise levels in the monitored locations. In [142], an LSTM was used to forecast the sound pressure and loudness levels of periods of 1 min, 5 min, 15 min, 30 min and 60 min. The data used for training and testing were issued from sensors located in an open office room where the majority of the noise is produced by human activities and speech.

In [13, 49, 50, 73] vanilla feedforward Artificial Neural Networks (ANN) were used for noise prediction. In [13], the architecture used is made of a learning vector quantization network to discard wrong input data and a feedforward network to predict the noise levels. In [49] the number of features used as inputs is 25 (traffic flow type, average speed of vehicles, etc.), while 5 features (traffic flow, percentage of heavy vehicles, etc.) are used in [50]. In [73] an ANN is used to predict the noise level caused by circulating vehicles. This network has 5 inputs: number of light motor vehicles, number of medium trucks, average speed, etc. Its prediction performance was compared to classical techniques and it was deduced that the ANN has the upper hand. In addition to neural networks some techniques use mathematical models for noise prediction [27].

Ensemble techniques were also tested for noise prediction. In [143], the authors used two ensemble approaches (linear and nonlinear) to predict noise level by combining the result of an Adaptive Neuro Fuzzy Inference System (ANFIS), a Feed Forward Neural Network (FFNN), a support vector regression and a multilinear regression. In the linear ensemble approach the simple and weighted averages were tested. In the nonlinear

approach, a FFNN and an ANFIS were trained to average the single models outputs. It was deduced that the nonlinear ANFIS ensemble gave the best prediction performance. 7 features were used as inputs for the prediction such as the number of cars, number of van/pickups, average speed, etc. In [144] an Emotional Artificial Neural Network (EANN) is used to predict noise. An EANN is an augmented neural network containing emotional units that can secrete hormones able to modify some neuron properties such as the weights and activation function. Two sets of inputs were tested to predict the noise level:

1. Volume of cars, volume of medium vehicles, volume of heavy vehicles, average traffic speed (km/h);
2. Traffic volume, percentage of heavy vehicles, average traffic speed (km/h).

Both sets of inputs improved the performance of noise prediction relative to empirical techniques and conventional multilinear regression models with the second set of inputs taking the lead.

In [127], the authors proposed an ensemble deep learning approach to detect sound anomalies for the purpose of machinery and factory monitoring. First, blind dereverberation as well as sound extraction are applied to extract background noise that can deteriorate the anomaly detection procedure. Then, autoencoders are used to reconstruct the noise for anomaly detection. In [83] dangerous road events such as skidding or crashing are identified by detecting anomalous sounds. Relevant features are extracted first and then a bag of words approach is applied to spot the events. In [106], using noise sensors near roads, the authors detect anomalous sounds unrelated to road traffic by using binary classification. They tested multiple models such as discriminant analysis, Gaussian mixture models, support vector machines and also k-nearest neighbors. Subsequently a Gaussian mixture model was selected for detection since it offers a good trade-off between detection, F-score and time complexity. In [100] Convolutional Recurrent Neural Networks (CRNN) were used to classify polyphonic sounds. This technique was tested on the following benchmark datasets corresponding to every day sound events: TUT-SED Synthetic 2016, TUT-SED 2009, TUT-SED 2016 CHiME-Home. The CRNN showed an improvement in the classification performance on the previously mentioned datasets compared to CNN, RNN and other techniques.

6.4/ MATERIALS AND METHODS

6.4.1/ URBAN ENVIRONMENTAL NOISE DATA

To obtain noise pollution data, it is necessary to have access to an acoustic sensor network that consists of nodes capturing sound in near real-time. Such nodes can take measurements over time and preproccess them, in order to send sound pressure levels to a gateway that will transmit these data to a storage platform. Many working smart cities have already deployed sensor networks allowing to collect and publish data through a cloud platform. For example, the Smart Santander facility provides access to data captured by noise sensors [94] thanks to the open source platform called FIWARE [126].

This work benefits from a collaboration with the Flowbird company[1], world leader in parking and transport ticketing solutions. The data used in this work was issued from Flowbird's Park&Breathe solution. This latter allows to create a network of noise and pollution sensors by using existing Strada parking terminals on which multi-sensor kits are integrated. Figure 6.1(a) displays a terminal with an integrated Park&Breathe kit on its top. Let us notice that this solution is very interesting as it operates on existing urban furniture and can take advantage of the fine-meshed network built by parking terminals in many cities. However, not all terminals are usually upgraded with a sensing kit, it depends on the coverage of the area of interest to be monitored.

[1] Website: flowbird.group

(a) Park&Breathe kit (b) Measurement points along Stratford Road

Figure 6.1: Multi-sensor kit integration on a terminal (a) and (b) positions of the three upgraded terminals (red marks) as well as the manual count point (blue mark).

6.4.1.1/ MONITORED AREA

Practically, the noise data are issued from parking terminals deployed in the city of Solihull, in the county of West Midlands in western-central England, and more precisely in one of its suburb called Shirley. Many terminals cover this area among which three of them are fitted with the Park&Breathe solution. Figure 6.1(b) displays the locations of these terminals, where each of them is spotted by a red mark and a code. As can be seen on the map they monitor a part of the Stratford Road, with two of them which are located opposite one another.

An idea of the traffic flow that can be observed on the road in question is given by the data corresponding to the manual count point 46367 (flagged by the blue mark on the map) managed by the English Department of Transport. According to the most recent available data, an average daily flow of about 31,500 motor vehicles uses this road. Moreover, as this road is also numbered A34, it means that it is a trunk road of the English network. The monitored area is thus a good representative of urban areas where citizens are exposed to large noise nuisances.

6.4.1.2/ DATA ACQUISITION AND FORMAT

Noise data are available in the form of a file per terminal. A file in which a line represents a histogram composed of 77 sound levels as abscissa, while the number of times a particular sound level occurs during a time period is plotted as ordinate. For this work, noise levels of a time frame ranging from May 2019 up to end January 2020 were used, where each line in a data file of a terminal corresponds to sound levels collected during 15 minutes.

However, these data were not used directly. In fact, they were processed in order to rather work with average values over periods of 1 hour. Therefore, the initial values of the three files were replaced by average values following two methods. The first method consists in replacing a set of 77 sound level values obtained at a time step by a single one computed as follows:

$$L(t) = \frac{\sum_{l=0}^{76} (N_l(t) \times (l + 35))}{\sum_{l=0}^{76} N_l(t)}$$

where $L(t)$ is the average value and $N_l(t)$ the number of times sound level l occurred during time step t. Notice that one needs to add 35 to l to obtain the true measure in decibels. This is due to the fact that l is not a sound level but rather a sound level index such that its starting value ($l = 0$) corresponds to a sound level of 35 dB. The second method keeps the histogram, but computes for each sound level its average value over a period of 1 hour. Since both methods consider a same period duration, they result in a same data set size of $19,443$ samples ($6,570$ samples for each of the terminals 500 and

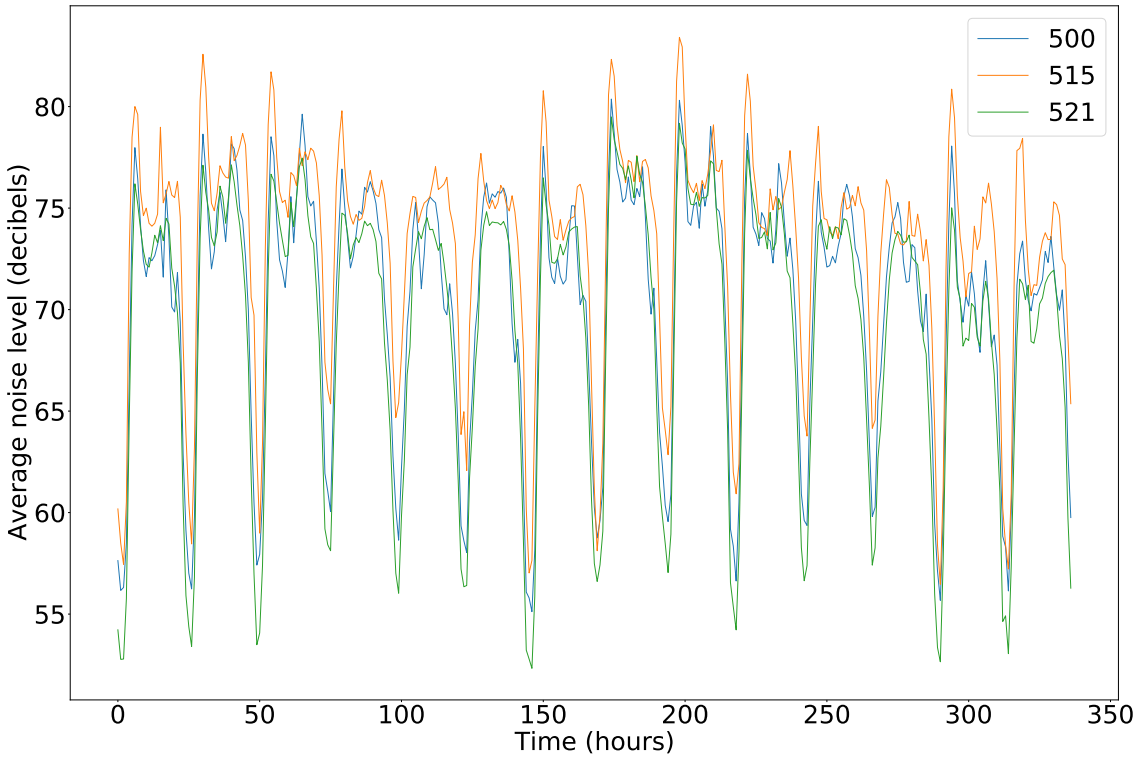


Figure 6.2: Average sound levels (dB) observed between May 1st 2019 and mid May - the x-axis shows the number of elapsed hours since May 1st 00:00:00.

515, 6,303 for terminal 521). Figure 6.2 shows the curves obtained with the first method for a time range of two weeks going from May 1st 2019 until mid May. The three curves present the same evolution, with extreme values reflecting the proximity of the terminals to the road and the distance between them. Thus, the blue and green curves are the most similar as they belong to the two terminals almost facing each other. In addition, the orange one has higher noise levels because the corresponding terminal is closer to the road. One can see a periodic daily evolution where the lowest average noise level is captured early in the morning. The days of the weekend and particularly the Sundays are easily recognizable on the graph (May 5th going from 96 hours to 120 hours and May 12th going from 264 hours to 288 hours), as well as the first Monday in May corresponding to the Early May Bank Holiday (May 6th going from 120 hours to 144 hours).

In the following, only data issued from the first method are used. This was done for several reasons. First, using the second method would have involved the prediction of several output values and thus probably would have given a more complex model. Second, among the 77 sound level values, some are more prevalent than others, so preprocessing should be investigated to limit their number and only keep the most significant ones. Third, regardless of the method used, both provide a dataset that has the same number of samples, but since with the second method the predicted output would have given finer information about the noise, more data certainly would have been needed.

6.4.2/ STUDIED MACHINE LEARNING MODELS

When dealing with time series data, the most obvious deep learning architecture to investigate is the Long Short-Term Memory (LSTM). Indeed, for most deep learning practitioners, sequence modeling usually means recurrent networks. An alternative to a pure recurrent neural network is to combine aspects of recurrent and convolutional architectures, using convolutions to change the representation of the original data. Finally, more recent deep learning models using attention, namely the Transformer and the Temporal Fusion Transformer, will also be studied. Deep learning is very powerful, but gradient tree boosting is also able to model relationships in the data and thus seems worthy of investigation. In this work, an evaluation of a representative of each of these machine learning models is conducted. But first, the mechanisms of Temporal Fusion Transformer and Gradient Boosting (LightGBM) will be briefly discussed since the mechanisms of LSTMs and convolutional architectures and their combination as well as the Transfomer architecture were examined previously.

6.4.2.1/ TEMPORAL FUSION TRANSFORMER

The temporal fusion transformer (TFT) is a deep learning model that integrates self-attention with multi-horizon forecasting [156]. The TFT can adapt to the complexity of the predicted time series using gating mechanisms. Short-term dependencies are captured using a sequence to sequence layer while long term ones are identified by masked interpretable multi-head attention. At each time step, pertinent input variables are chosen using variable selection networks. For further control of predictions, static metadata are incorporated into the forecasting system via static covariate encoders. In addition to static features and observed inputs from the past, known future inputs are also fed into the network. These future inputs are not predicted, they are known in advance and are only used to help predicting other features. This network also provides the ability to predict a range of probable values for each prediction horizon through the use of quantile forecasts.

6.4.2.2/ GRADIENT BOOSTING (LIGHTGBM)

LightGBM is a popular gradient boosting framework created by Microsoft. It is a tree based learning algorithm designed to have a faster training speed, a lower memory usage and a better accuracy [102]. It supports execution with GPU and parallelization, has a function for testing and keeps the best set of parameters. Traditional gradient boosting algorithms work by first fitting a weak learner to pairs of sampled inputs and outputs. Then, repeated estimation of the remaining errors via weak learners is realized, typically

using tree based learners [133]. Compared to other boosting algorithms, LightGBM differs in the approach used for tree growth that is leaf-wise (best-first) and not level-wise (depth-first).

6.5/ RESULTS

The deep learning architectures, except the TFT, have been implemented and evaluated with the easy-to-use Keras Python library. This library is user-friendly and allows to perform the computations with different backends such as TensorFlow. It is particularly suited for the implementation of preexisting deep learning architectures, when there is no need for low level operations design. In the case of gradient boosting, many models are already available in the Python Scikit-learn package, namely the LightGBM model. Finally the temporal fusion transformer was implemented using the Pytorch-forecasting library. This high level API of deep learning architectures is used to forecast time series for real world and research scenarios.

6.5.1/ SETUP OF THE DEEP LEARNING ARCHITECTURES

- The Stacked-LSTM architecture that was chosen is composed of a first layer of 150 LSTM units, followed by a second layer of 100 units and a final dense layer reduced to a single neuron. Both LSTM layers use a ReLU activation function and the dense layer uses a linear activation function.

- The CNN-LSTM is made up of two parts. The first one is a fully convolutional network that consists of three layers of 16 neurons where each neuron computes one-dimensional convolutions. The convolution kernels have a size of 3 and are initialized using the Glorot uniform initializer available in Keras. The shape of the data in the input of the convolutional part is a vector whose size corresponds to the lookback value explained thereafter. The input and output data have the same shape. The second part is a LSTM similar to the Stacked-LSTM.

- The transformer used is made up of a transformer encoder containing several transformer blocks which are then connected to multiple dense layers with ReLU activation and dropout. The output of the model is a single neuron with linear activation. Each transformer block contains a layer normalization followed by a multi-head attention and then by a dropout and a feedforward part. The feedforward part starts with layer normalization followed by two one dimensional convolutional networks separated by a dropout.

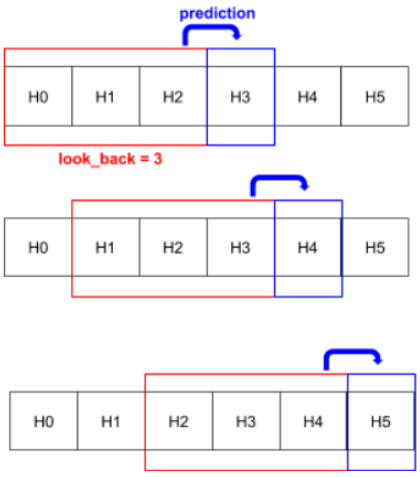


Figure 6.3: Lookback principle for time series forecasting.

- The temporal fusion transformer architecture used is exactly the same as the one mentioned in the original paper by [156].

A key parameter when working with time series is the lookback value. It corresponds to the amount of previous data that will be provided to the neural network so that it can predict the next value. For example, as can be seen in Figure 6.3, with a lookback of 3 the network will have access to data $H0$, $H1$ and $H2$ to predict $H3$. Both architectures with LSTM layers are first trained with Nadam optimizer, the version of Adam that uses the Nesterov momentum instead of a regular momentum, while the Transformer is trained with Adam.

6.5.2/ Average noise level prediction with LSTM architectures and LightGBM

For these experiments, only the data samples from terminal 515 were used at first, using 90 % of them for training and the remaining 10 % for testing. In practice, it means that $5,912$ samples were used for training and 657 samples for testing. The prediction results obtained with the two LSTM deep learning architectures (the Stacked one and the CNN-LSTM), considering different combinations of maximum training epochs and lookback values, are shown in Table 6.1. To assess the quality of the predictions, the Root Mean Square Error (RMSE) which is a standard way to measure the error of a model in its prediction ability, was chosen. As can be seen, for the combinations with a low number of training epochs, the best prediction performance is always provided by the Stacked-LSTM, while the CNN-LSTM's performance is in the same order of magnitude or only slightly worse for a same lookback value. However, the gap between both architectures reduces as the number of epochs and the lookback value increase and finally it is the

CNN-LSTM that takes the lead for the configuration with 100 epochs and a lookback value equal to 36 (the lines in italic in the table). The overall best prediction performance is obtained with the Stacked-LSTM for the configuration with 100 epochs and a lookback of 24 (the bold line in the table). It can also be noticed that the CNN-LSTM appears to be more sensitive to the lookback value than the Stacked-LSTM. A possible explanation of these observations is that the CNN-LSTM is deeper than the Stacked-LSTM, thus it needs a larger number of epochs and enough input data to take advantage of the CNN part (to be trained to extract relevant features).

Table 6.1: Evaluation of the two LSTM-based deep learning architectures for different values of training epochs and lookback.

Deep architecture	Epochs	Lookback	RMSE on train	RMSE on test
Stacked-LSTM	25	12	1.35	1.29
	50	3	1.43	1.48
	50	12	1.22	1.34
	50	24	1.19	1.17
	100	**24**	**0.96**	**1.01**
	100	*36*	*0.99*	*1.05*
CNN-LSTM	25	12	1.37	1.48
	50	3	4.28	4.32
	50	12	1.27	1.52
	50	24	1.09	1.16
	100	24	1.03	1.17
	100	*36*	*0.98*	*1.05*

Table 6.2: Evaluation of LightGBM for different configurations of tree growth and learning rates.

LightGBM				
Num_leaves	Num_trees	Learning rate	RMSE on train	RMSE on test
11	500	0.05	0.78	1.00
11	**1000**	**0.04**	**0.71**	**0.99**

For the LightGBM gradient boosting method, the parameters controlling the learning process are different. Num_leaves defines the maximum number of leaves in one tree, while the number of trees is set with the parameter denoted Num_trees. The learning rate is the step size at each iteration to reach the minimum of the loss function. Table 6.2 presents the prediction errors for two configurations of tree growth and learning rate. When comparing them to the errors shown in Table 6.1, LightGBM seems to be a very good competitor for the deep learning architectures, giving lower training and testing errors, even if in this latter case the improvement is very small. However, LightGBM exhibits large differences between the training and testing errors that might be a first evidence of overfitting to which leaf-wise growth is known to be more sensitive than its level-wise counterpart. In the following we first try to optimize the two deep learning architectures and to investigate

Table 6.3: Evaluation of deep learning architectures trained with different optimizers.

Deep architecture	Optimizer	Epochs to converge	RMSE on train	RMSE on test
Stacked-LSTM	Nadam	134	1.05	1.04
	Adam	95	1.09	1.06
	RMSprop	259	0.96	0.97
CNN-LSTM	Nadam	223	0.94	0.94
	Adam	187	0.99	0.99
	RMSprop	**265**	**0.92**	**0.95**

different forecasting case studies.

6.5.3/ Optimization of the Stacked-LSTM and CNN-LSTM

To increase the accuracy of the predictions given by the deep networks with LSTM layers, the data used for training have been shuffled (activation of the shuffle option in time series generator provided by Keras). This improves a neural network ability to generalize and prevent overfitting. Moreover, different optimizers have been evaluated with an early stopping when the validation loss stops decreasing. It was then possible to conduct an analysis of the convergence for each optimizer. As highlighted by Table 6.3, from the RMSE values' point of view, ADAM is slightly outperformed by NADAM, which is itself overrun by RMSPROP, which is therefore the optimizer providing the lowest loss values. Besides, the optimizer RMSPROP exhibits the same behavior (almost the same number of epochs to reach convergence) for both deep learning architectures, while ADAM and NADAM begin to overfit much earlier in the case of Stacked-LSTM.

6.5.4/ 6 Day Forecasts

Up to now the emphasis was put on forecasting the average noise level one hour ahead (H+1), but this is not enough. Therefore, the two machine learning models that have achieved the lowest errors (CNN-LSTM and LightGBM) were used to make predictions for the following 6 days (H+144). One can see on Figure 6.4 that both models learned the underlying pattern of the average noise level evolution, with a small preference for the deep neural network as its predictions (the green curve) seem to be closer to the real measurements (the blue curve which stops when the x-axis value is equal to 350 hours). Therefore the CNN-LSTM was selected to assess its ability to detect anomalies.

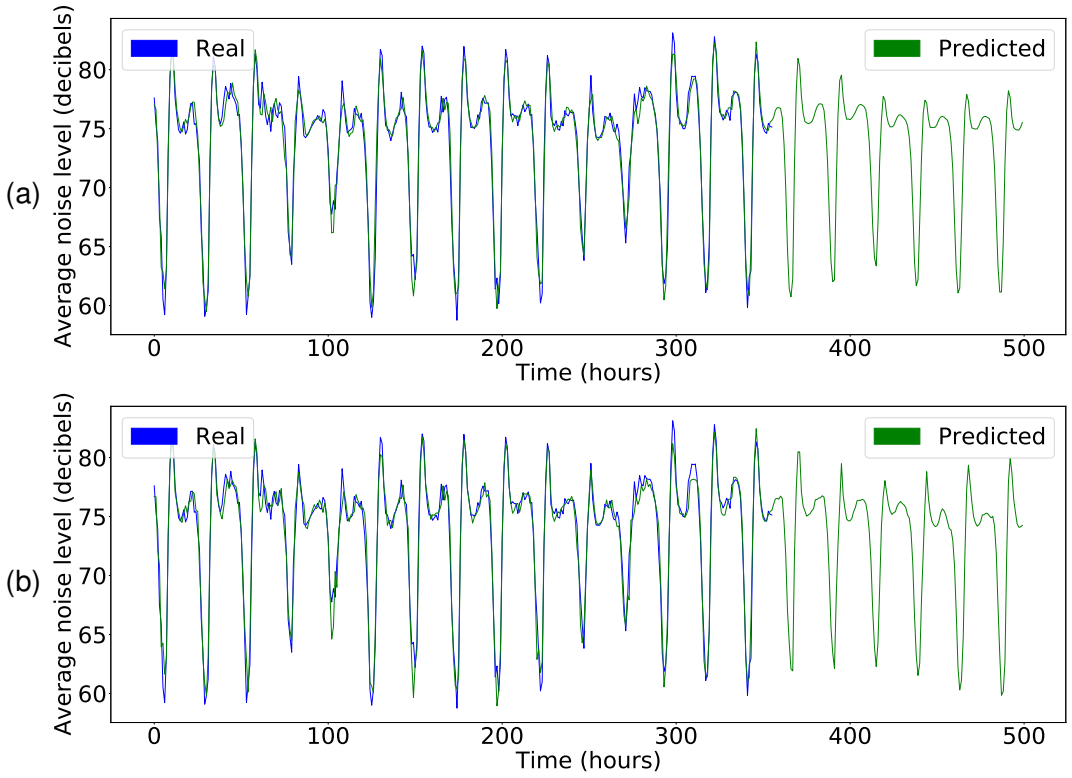


Figure 6.4: 6 days forecasting with CNN-LSTM (a) and LightGBM (b).

6.5.5/ TRAINING ON ONE TERMINAL AND TESTING FORECASTS ON OTHERS

Since the CNN-LSTM performs better than LightGBM, it is also tested for its ability to do forecasts on sound levels at different locations (training data from terminal 500 and testing with data from terminals 515 and 521). The training was realised on the CNN-LSTM which performed better than Stacked-LSTM and LightGBM. 70 % of the data from terminal 500 were used for training, 15 % for validation and 15 % for testing. The hyperparameters used are the following: RMSprop optimizer, lookback of 36 and 84 epochs (till validation loss stops decreasing). The performance of the CNN-LSTM on the terminals 500, 515 and 521 is summarized on the line denoted *Training data 500* in Table 6.4. The second line shows the performance when the training is conducted on terminal 515 (till validation loss stops decreasing, i.e. 150 epochs) and tested on terminals 500 and 521, while the third line correspond to last case where the training is completed on terminal 521. When the model is trained on terminal 500 or 515 (first and second line), the RMSE is the closest between these 2 terminals because they are the closest to the main road compared to the terminal 521 so they have relatively close sound levels. But when the model is trained on terminal 521, the RMSE is the closest between terminals 500 and 521 since they are the closest to each other in distance compared to the terminal 515.

Table 6.4: Evaluation of CNN-LSTM on testing sets of the terminals 515, 500 and 521 (training on data from terminal in italic).

CNN-LSTM			
Training data 500 RMSE	*Terminal 500* 1.32	Terminal 515 1.86	Terminal 521 2.24
Training data 515 RMSE	Terminal 500 2.18	*Terminal 515* 1.03	Terminal 521 3.01
Training data 521 RMSE	Terminal 500 1.91	Terminal 515 2.52	*Terminal 521* 1.34

6.5.6/ Training and testing forecasts on different terminals simultaneously

A CNN-LSTM was also trained and tested on terminals 500, 515 and 521 simultaneously (multivariate time series) for the purpose of sound level forecasting. The CNN-LSTM has the same structure as the one used for one terminal forecasting. However the output contains 3 dense neurons one for each terminal. Using the same hyperparameters as before, and after training for 51 epochs (till validation loss stops decreasing), the forecasting RMSE is 1.5 on the test set. In Table 6.4, the average of all RMSE values is 1.93 which is greater than 1.5. Hence training simultaneously on different terminals using a model with same intermediate layers gives more accurate forecasts for multiple terminals. However, if forecasting on only one terminal is needed the model with one dense output neuron and the same intermediate layers is more accurate.

6.5.7/ Transformer and Temporal fusion transformer

Finally, the prediction ability of the two deep learning architectures with attention has been evaluated. First, a transformer was trained and tested on terminal 515 to predict future sound level for different values of training epochs and lookbacks. Default hyperparameters were chosen i.e. head size: 256, number of heads: 4, feed forward dimension in the transformer block: 4, number of transformer blocks: 4, MLP units: 128, MLP dropout: 0.4, dropout: 0.25. The results are shown in Table 6.5. After 500 epochs the lookback of 72 gave the lowest RMSE value (the bold line).

A temporal fusion transformer (TFT) was also trained and tested on terminal 515 to predict future sound level for different values of training epochs and lookbacks. The results are shown in Table 6.6 for the default choice of hyperparameters of the TFT i.e. learning rate: 0.03, hidden size: 32, attention head size: 1, dropout: 0.1, hidden continuous size: 16. After 100 epochs the lookback of 36 gave the lowest RMSE (the bold line).

It can be noticed that the TFT needs much less epochs (100 epochs) to reach maximum performance compared to the transformer architecture (500 epochs). The reason for this

Table 6.5: Evaluation of the transformer for different values of training epochs and lookback.

Epochs	Lookback	RMSE on train	RMSE on test
25.0	12.0	2.00	2.01
50.0	3.0	2.02	1.93
50.0	12.0	1.72	1.73
50.0	24.0	1.77	1.88
100.0	24.0	1.40	1.42
100.0	36.0	1.33	1.41
150.0	36.0	1.25	1.27
200.0	36.0	1.19	1.20
200.0	48.0	1.14	1.15
250.0	48.0	1.15	1.20
300.0	48.0	1.12	1.16
350.0	48.0	1.13	1.14
350.0	60.0	1.08	1.13
400.0	60.0	1.07	1.13
450.0	60.0	1.05	1.12
450.0	72.0	1.05	1.12
500.0	**72.0**	**1.04**	**1.11**
700.0	100.0	0.95	1.12

Table 6.6: Evaluation of the temporal fusion transformer for different values of training epochs and lookback.

Epochs	Lookback	RMSE on train	RMSE on test
25.0	12.0	1.26	1.30
50.0	3.0	1.64	1.51
50.0	12.0	1.24	1.30
50.0	24.0	1.19	1.23
100.0	24.0	1.17	1.23
100.0	**36.0**	**1.06**	**1.11**
150.0	36.0	1.07	1.18
200.0	36.0	1.07	1.18
200.0	48.0	1.12	1.17

discrepancy lies in the fact that the TFT is a much more complex model than the transformer architecture. This higher complexity is also translated into a longer duration to train an epoch which is 17 seconds for the TFT compared to 5 seconds for the transformer.

Since the time series data is not complex and large enough for the use of large deep models such as transformers and TFT, the best RMSE obtained from these two architectures is still slightly worse than other simpler models such as LSTM and LightGBM.

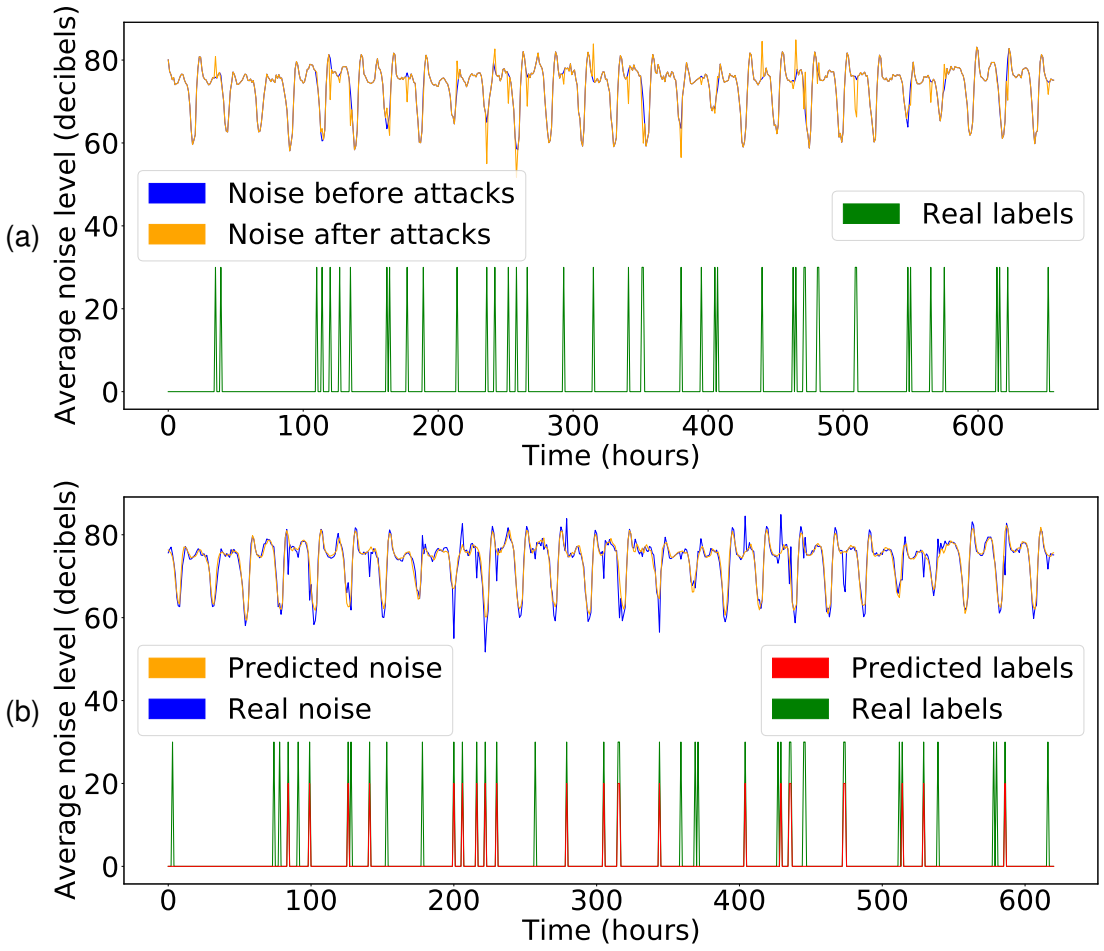


Figure 6.5: Dataset with attacks (a) and detection of anomalies on average noise curve (b). In the lower part of both graphs, the green peaks indicate false data injections, while the red peaks highlight those that were detected. Note that the green and red curves are not sound levels.

6.5.8/ Detection of anomalous sound data

Since the CNN-LSTM architecture can successfully forecast sound levels, it is worthwhile to assess more of its abilities such as detecting potential anomalies in the data collected by the sensors. If a sample is far from the corresponding forecast, this sample is considered as an anomaly which might result from a malicious FDIA attack. The first step is the creation of a dataset with a few anomalies. In order to do that, the input data were randomly disturbed by adding / subtracting values to some of the original average noise values. More precisely, values between 1 and 10 dB sampled from a uniform distribution were added / subtracted to 5% randomly chosen instances of the original values.

As explained above, since the CNN-LSTM neural network makes satisfactory predictions, a big difference between real measured data and the forecast implies the presence of an anomaly. In Figure 6.5(a), the blue curve corresponds to the original data, the or-

ange curve represents the data obtained after the false data injections. In Figure 6.5(a) and Figure 6.5(b) the green curves show where data have been attacked (peaks denote anomalous data otherwise the data is normal). In Figure 6.5(b), the blue curve corresponds to the real data which underwent false data injections (it is equivalent to the orange curve in Figure 6.5(a)), the orange curve represents the prediction of the deep neural network, and the red curve shows where an anomaly can be detected using the CNN-LSTM's predictions (peaks denote detected anomalous data otherwise the data is considered as normal). In fact, a false data injection attack is detected as soon as the difference between a real measured data and its counterpart predicted by the neural network is larger than a given threshold. According to [3], a threshold of 5 dB of noise change is clearly noticeable by the human ear. Therefore a threshold of 5 dB was chosen for anomaly detection. All the injected anomalies which are higher than 5 dB were detected. This shows that the detection model is able to identify all clearly noticeable noise changes. This is considered enough since unnoticeable changes are typically not disruptive. Note that the detection threshold can be lowered to 4 dB, since the highest prediction error of the neural network is 4 dB. However, lowering the detection threshold furthermore results in some false positive cases. To detect smaller noise level attacks, the prediction model must be improved.

6.6/ CONCLUSION

The ability of deep learning architectures and gradient boosting to predict the real average urban noise level has been investigated in the context of smart cities. The sensing platform used to collect the data has been presented, in particular the monitored area and the way the data were obtained and formatted. The experimental results show that relevant short-term predictions can be provided with a CNN-LSTM and LightGBM, as well as 6 day forecasts capturing the daily pattern exhibited in the data. Thanks to these results the possibility of detecting anomalous data was also evaluated with the CNN-LSTM. It has been shown that an increase in sound intensity of 5 dB greater can be detected without any error. Being able to identify such problems is of great interest to make smart cities more robust against possible false data injection attacks.

7

CONCLUSION AND PERSPECTIVES

7.1/ CONCLUSION

In this thesis, deep learning anomaly detection strategies have been proposed to detect false data in business data, focusing first on the ADS-B protocol. Such false data may be injected due to the lack of encryption and authentication of ADS-B messages. The main approach used for anomaly detection in the context of this thesis is supervised learning. Another secondary work focuses on the IoT domain, specifically on predicting noise collected from a network of smart parkmeters.

The two case studies, the ADS-B protocol and environmental noise monitoring with sensor networks, were presented in the second chapter. We first showed the inherent vulnerabilities of the ADS-B protocol, mainly its lack of authentication and encryption which makes it vulnerable to many types of fake data injection attacks. Then, we presented the types of acoustic sensor networks and discussed the requirements for their proper operation. In addition, an IoT false data generator was also described.

The third chapter presented various related works regarding anomaly detection. First traditional anomaly detection techniques are presented followed by machine learning and deep learning anomaly detection approaches used in time series. Then anomaly detection techniques applied on ADS-B data are also described. These methods show the prevalence of semi-supervised approaches for ADS-B anomaly detection, specifically autoencoder architectures. To the best of our knowledge, supervised approaches for ADS-B anomaly detection are absent from the literature due to the lack of attacked data.

Afterwards three contributions on anomaly detection in business data have been presented. Two in the context of the ADS-B air traffic surveillance protocol and the third in the context of an acoustic sensor network.

A comparative study of deep learning architectures for detection of anomalous ADS-B messages In the first contribution, a comparative study on anomaly detection of altered ADS-B data was realised. Only one type of alterations was investigated, mainly rough altitude alterations. Many models were compared such as XGBoost, 1D-CNN, LSTM, 1D-CNN connected to LSTM and bidirectional LSTM. XGBoost gave a very low precision. LSTM was compared against different types of optimizers, lookback values and structure. Bidirectional LSTM and CNN gave worse results than individual LSTM, but when CNN is combined with LSTM the performance increases. Nevertheless, the best architecture obtained for anomaly detection is an LSTM of two layers composed of 64 units in the first one and 32 in the second, a lookback of 20 time steps, trained with NADAM optimizer.

Supervised ADS-B anomaly detection using a false data generator In the second contribution, a strategy for supervised ADS-B anomaly detection was devised. It relies on a false data generator (FDI-T) to obtain labeled normal and anomalous data used to train the model. This generator is based on a DSL language which gives the needed flexibility and expressiveness to generate a multitude of distinct ADS-B alteration scenarios. Different models were trained to detect different types of attacks individually: gradual attacks (altitude attack, ground speed attack, track attack, latitude attack and longitude attack) and waypoints attacks (an attack which deviates a flight to pass by specific points). The results of testing those models showed that the increase of the lookback and the number of epochs enhances the detection performance but only slightly. These results also showed that the previously mentioned attacks can be detected in a supervised fashion using our strategy. Semi-supervised approaches were also compared to the supervised approach showing the performance advantage of the supervised approach. In addition, a meta model made of individual models was also created and it was successful in detecting different types of gradual attacks simultaneously. An alarm mechanism was also proposed and its performance was evaluated as well.

Deep learning and gradient boosting for urban environmental noise monitoring In the third contribution, different machine learning and deep learning models were compared in the context of forecasting urban noise levels namely LightGBM, LSTM, CNN-LSTM, Transformer and Temporal Fusion Transformer. The noise data used for training those models were gathered from a smart parkmeters' network. The components of this network were described as well as the data acquisition process. It was deduced that the CNN-LSTM was the most suitable for forecasting the noise levels and was also usable to detect rough punctual noise alterations (5 dB alteration) based on the forecasting task. Such capabilities can be used to ensure the security and safety of smart city systems.

7.2/ PERSPECTIVES

Few shots learning for ADS-B anomaly detection There are many perspectives to consider in the aviation domain which requires ever increasing safety measures. The key focus of this thesis was to propose a supervised deep learning approach for ADS-B anomaly detection. ADS-B data are readily available from the OpenSky network and anomalous labeled data can be obtained using FDI-T. Nevertheless downloading and attacking entire flights as well as training deep learning models is time consuming especially if our goal is to train models on substantial number of aircraft. One way to tackle such a problem is to use few shots learning. This technique relies on only a few samples belonging to different classes which comprise what we call a support set. The test dataset is called query dataset. In order to apply the few shots learning technique, a similarity function is trained to identify if a pair of samples (support sample, query sample) belong to the same class. In the case of supervised ADS-B anomaly detection the only classes present in the support and query sets are normal data and anomalies, and the samples are sequences of ADS-B data. A traditional deep learning architecture used in few shots learning is the siamese network which is made of a separate pair of neural networks. Those two networks have the same output which is usually used to obtain a similarity measure between the inputs.

Ghost aircraft injections and aircraft spoofing anomaly detection Other types of attacks could also be taken into consideration in anomaly detection using ADS-B data such as Ghost Aircraft injections and Aircraft Spoofing. Ghost Aircraft injection detection and aircraft spoofing detection might rely on past flights taking the same route in the case of commercial flights. For other types of flights, normal flight patterns need to be learned. However, for the detection to be reliable, much more data is most probably needed since the attackers might want to create ghost flights imitating normal flights.

Mixed airspace anomaly detection In our contributions, ADS-B anomaly detection was applied by considering the message flow of a single aircraft. Each aircraft is treated independently of the others. Therefore a future work worth considering would be to combine the message flow of several aircraft in a given zone and then to detect the abnormal behavior of an aircraft compared to the others. To be able to do such a task, relations between different aircraft should be examined. These relations could depend on many metrics such as the distance between aircraft, their ground speed, etc. One way to model such relations is the use of graphs. In this case, graph convolutional networks could be used to classify in a supervised fashion ADS-B data as normal or anomalous. Such networks were shown to be successful in processing graph data like social networks data.

Semi-supervised approaches for ADS-B anomaly detection could also be investigated such as graph convolutional autoencoders akin to the works in [129].

Generative Adversarial Networks to generate mixed airspace attacks In [122], the authors proposed the use of two dimensional CNN-LSTM autoencoders to detect anomalies in a flow of ADS-B messages originating from different aircraft. In their approach, the flow of messages is modeled using a sequence of images which are then fed into the autoencoder for semi-supervised anomaly detection. Each image expresses the state of multiple aircraft in a visually explainable fashion by using shapes such as arrows. Therefore, another future work worth examining is using 3D convolutional Generative Adversarial Networks (GAN) [98] to generate false sequences of mixed airspace images. Three-dimensional convolutions are used to process sequences of images where two dimensions and one dimension correspond to the images' and time's/sequence's dimensions respectively. Nevertheless, in order to ensure the meaningfulness of the generated images (for instance the necessity that an aircraft is modeled by an arrow), specific conditions need to be enforced into the structure of the network.

PUBLICATIONS

CONFERENCE PAPERS

- Ralph Karam, Michel Salomon, and Raphael Couturier. *"A Comparative Study of Deep Learning Architectures for Detection of Anomalous ADS-B messages"*. In **2020 7th International Conference on Control, Decision and Information Technologies (CoDIT), 2020, pp. 241-246, doi: 10.1109/CoDIT49905.2020.9263880**.
- Ralph Karam, Michel Salomon, and Raphael Couturier. *"Supervised ADS-B Anomaly Detection Using a False Data Generator"*. In **2022 2nd International Conference on Computer, Control and Robotics (ICCCR), 2022, pp. 218-223, doi: 10.1109/ICCCR54399.2022.979014**.

SUBMITTED PAPERS

- Jeremy Renaud, Ralph Karam, Michel Salomon, and Raphael Couturier. *"Deep Learning and Gradient Boosting for Urban Environmental Noise Monitoring in Smart Cities"*. In International Journal **Expert Systems with Applications**. Submission under revision.

BIBLIOGRAPHY

[1] **Airborne collision avoidance system (ACAS)**. https://skybrary.aero/articles/airborne-collision-avoidance-system-acas.

[2] **Concept of operations Mode S in Europe**. https://www.eurocontrol.int/sites/default/files/2019-04/surveillance-mode-s-concept-of-operations-19961128.pdf.

[3] **Human perception of sound**. https://modularwalls.com.au/wp-content/uploads/Human-perception-of-sound_April2016-WEB.pdf.

[4] **Live flight tracker - real-time flight tracker map**.

[5] **Technical provisions for Mode S services and extended squitter**. http://www.aviationchief.com/uploads/9/2/0/9/92098238/icao_doc_9871_-_technical_provisions_for_mode_s_-_advanced_edition_1.pdf.

[6] **Teen monitoring Elon Musk's jet 'tracking Gates, Bezos and Drake too'**. https://www.theguardian.com/technology/2022/feb/02/teen-tracking-elon-musk-jet-bill-gates-jeff-bezos-drake-jack-sweeney-tesla-flight-tracker-bot.

[7] PARZEN, E. **On estimation of a probability density function and mode**. *The annals of mathematical statistics 33*, 3 (1962), 1065–1076.

[8] AKIMA, H. **A new method of interpolation and smooth curve fitting based on local procedures**. *Journal of the ACM (JACM) 17*, 4 (1970), 589–602.

[9] FIX, E., AND HODGES, J. L. **Discriminatory analysis. nonparametric discrimination: Consistency properties**. *International Statistical Review/Revue Internationale de Statistique 57*, 3 (1989), 238–247.

[10] KANG, K. C., COHEN, S. G., HESS, J. A., NOVAK, W. E., AND PETERSON, A. S. **Feature-oriented domain analysis (FODA) feasibility study**. Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.

[11] GALLAGHER, P. R. **A guide to understanding data remanence in automated information systems**, 1991.

[12] COMON, P. **Independent component analysis, a new concept?** *Signal processing 36*, 3 (1994), 287–314.

[13] CAMMARATA, G., CAVALIERI, S., AND FICHERA, A. **A neural network architecture for noise prediction**. *Neural Networks 8*, 6 (1995), 963–973.

[14] LECUN, Y., BENGIO, Y., AND OTHERS. **Convolutional networks for images, speech, and time series**. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995.

[15] TAYLOR, R. N., TRACZ, W., AND COGLIANESE, L. **Software development using domain-specific software architectures: CDRI A011—a curriculum module in the SEI style**. *ACM SIGSOFT Software Engineering Notes 20*, 5 (1995), 27–38.

[16] ESTER, M., KRIEGEL, H.-P., SANDER, J., XU, X., AND OTHERS. **A density-based algorithm for discovering clusters in large spatial databases with noise.** In *kdd* (1996), vol. 96, pp. 226–231.

[17] COPLIEN, J., HOFFMAN, D., AND WEISS, D. **Commonality and variability in software engineering**. *IEEE software 15*, 6 (1998), 37–45.

[18] FRAKES, W., DIAZ, R., FOX, C., AND OTHERS. **DARE: Domain analysis and reuse environment**. *Annals of software engineering 5*, 1 (1998), 125–141.

[19] SCHÖLKOPF, B., SMOLA, A., AND MÜLLER, K.-R. **Nonlinear component analysis as a kernel eigenvalue problem**. *Neural computation 10*, 5 (1998), 1299–1319.

[20] STUDER, R., BENJAMINS, V. R., AND FENSEL, D. **Knowledge engineering: principles and methods**. *Data & knowledge engineering 25*, 1-2 (1998), 161–197.

[21] ANKERST, M., BREUNIG, M. M., KRIEGEL, H.-P., AND SANDER, J. **Optics: Ordering points to identify the clustering structure**. *ACM Sigmod record 28*, 2 (1999), 49–60.

[22] BEYER, K., GOLDSTEIN, J., RAMAKRISHNAN, R., AND SHAFT, U. **When is "nearest neighbor" meaningful?** In *In Int. Conf. on Database Theory* (1999), pp. 217–235.

[23] SCHÖLKOPF, B., WILLIAMSON, R. C., SMOLA, A., SHAWE-TAYLOR, J., AND PLATT, J. **Support vector method for novelty detection**. *Advances in neural information processing systems 12* (1999).

[24] BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., AND SANDER, J. **LOF: identifying density-based local outliers**. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (2000), pp. 93–104.

[25] GERS, F. A., SCHMIDHUBER, J., AND CUMMINS, F. **Learning to forget: Continual prediction with LSTM**. *Neural computation 12*, 10 (2000), 2451–2471.

[26] RAMASWAMY, S., RASTOGI, R., AND SHIM, K. **Efficient algorithms for mining outliers from large data sets**. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (2000), pp. 427–438.

[27] STEELE, C. **A critical review of some traffic noise prediction models**. *Applied acoustics 62*, 3 (2001), 271–287.

[28] HALL, J., BARBEAU, M., KRANAKIS, E., AND OTHERS. **Detection of transient in radio frequency fingerprinting using signal phase**. *Wireless and Optical Communications* (2003), 13–18.

[29] JOLLIFFE, I. T. **Principal component analysis**. *Technometrics 45*, 3 (2003), 276.

[30] KARLOF, C., AND WAGNER, D. **Secure routing in wireless sensor networks: Attacks and countermeasures**. *Ad hoc networks 1*, 2-3 (2003), 293–315.

[31] TANASE, M. **IP spoofing: an introduction**. *Security Focus 11* (2003), 1674–1680.

[32] TAX, D. M., AND DUIN, R. P. **Support vector data description**. *Machine learning 54*, 1 (2004), 45–66.

[33] GRAVES, A., AND SCHMIDHUBER, J. **Framewise phoneme classification with bidirectional LSTM and other neural network architectures**. *Neural networks 18*, 5-6 (2005), 602–610.

[34] POLASTRE, J., SZEWCZYK, R., AND CULLER, D. **Telos: Enabling ultra-low power wireless research**. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.* (2005), IEEE, pp. 364–369.

[35] RAMSAY, J. O., AND SILVERMAN, B. W. **Fitting differential equations to functional data: Principal differential analysis**. Springer, 2005.

[36] KANG, J. **Urban sound environment**. CRC Press, 2006.

[37] KISSEL, R., SCHOLL, M. A., SKOLOCHENKO, S., AND LI, X. **SP 800-88 Rev. 1. guidelines for media sanitization**, 2006.

[38] PATHAN, A.-S. K., LEE, H.-W., AND HONG, C. S. **Security in wireless sensor networks: issues and challenges**. In *2006 8th International Conference Advanced Communication Technology* (2006), vol. 2, IEEE, pp. 6–pp.

[39] BARAGONA, R., AND BATTAGLIA, F. **Outliers detection in multivariate time series by independent component analysis**. *Neural computation 19*, 7 (2007), 1962–1984.

[40] CHEN, Y., TRAPPE, W., AND MARTIN, R. P. **Detecting and localizing wireless spoofing attacks**. In *2007 4th Annual IEEE Communications Society Conference on sensor, mesh and ad hoc communications and networks* (2007), IEEE, pp. 193–202.

[41] DUTTA, H., GIANNELLA, C., BORNE, K., AND KARGUPTA, H. **Distributed top-k outlier detection from astronomy catalogs using the demac system**. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (2007), SIAM, pp. 473–478.

[42] FILIPPONI, L., SANTINI, S., AND VITALETTI, A. **Data collection in wireless sensor networks for noise pollution monitoring**. In *International Conference on Distributed Computing in Sensor Systems* (2008), Springer, pp. 492–497.

[43] LIU, F. T., TING, K. M., AND ZHOU, Z.-H. **Isolation forest**. In *2008 eighth ieee international conference on data mining* (2008), IEEE, pp. 413–422.

[44] SANTINI, S., OSTERMAIER, B., AND VITALETTI, A. **First experiences using wireless sensor networks for noise pollution monitoring**. In *Proceedings of the workshop on Real-world wireless sensor networks* (2008), pp. 61–65.

[45] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. **Anomaly detection: A survey**. *ACM computing surveys (CSUR) 41*, 3 (2009), 1–58.

[46] HALDERMAN, J. A., SCHOEN, S. D., HENINGER, N., CLARKSON, W., PAUL, W., CALANDRINO, J. A., FELDMAN, A. J., APPELBAUM, J., AND FELTEN, E. W. **Lest we remember: cold-boot attacks on encryption keys**. *Communications of the ACM 52*, 5 (2009), 91–98.

[47] PADMAVATHI, D. G., SHANMUGAPRIYA, M., AND OTHERS. **A survey of attacks, security mechanisms and challenges in wireless sensor networks**. *arXiv preprint arXiv:0909.0576* (2009).

[48] REYNOLDS, D. A. **Gaussian mixture models.** *Encyclopedia of biometrics 741*, 659-663 (2009).

[49] GENARO, N., TORIJA, A., RAMOS-RIDAO, A., REQUENA, I., RUIZ, D. P., AND ZAMORANO, M. **A neural network based model for urban noise prediction**. *The journal of the Acoustical Society of America 128*, 4 (2010), 1738–1746.

[50] GIVARGIS, S., AND KARIMI, H. **A basic neural traffic noise prediction model for Tehran's roads**. *Journal of Environmental Management 91*, 12 (2010), 2529–2534.

[51] HAKALA, I., KIVELÄ, I., IHALAINEN, J., LUOMALA, J., AND GAO, C. **Design of low-cost noise measurement sensor network: Sensor function design**. In

2010 First International Conference on Sensor Device Technologies and Applications (2010), IEEE, pp. 172–179.

[52] LISBOA, L. B., GARCIA, V. C., LUCRÉDIO, D., DE ALMEIDA, E. S., DE LEMOS MEIRA, S. R., AND DE MATTOS FORTES, R. P. **A systematic review of domain analysis tools**. *Information and Software Technology 52*, 1 (2010), 1–13.

[53] BOTTELDOOREN, D., DE COENSEL, B., OLDONI, D., VAN RENTERGHEM, T., AND DAUWE, S. **Sound monitoring networks new style**. In *Acoustics 2011: Breaking New Ground: Annual Conference of the Australian Acoustical Society* (2011), Australian Acoustical Society, pp. 1–5.

[54] CANDÈS, E. J., LI, X., MA, Y., AND WRIGHT, J. **Robust principal component analysis?** *Journal of the ACM (JACM) 58*, 3 (2011), 1–37.

[55] GUPTA, S., AND GHATAK, C. **Environmental noise assessment and its effect on human health in an urban area**. *International Journal of Environmental Sciences 1*, 7 (2011), 1954–1964.

[56] KIVELÄ, I., GAO, C., LUOMALA, J., AND HAKALA, I. **Design of noise measurement sensor network: Networking and communication part**. In *Proceedings of the 5th International Conference on Sensor Technologies and Applications, Côte d'Azur, France* (2011), pp. 21–27.

[57] SU, K., LI, J., AND FU, H. **Smart city and the applications**. In *2011 international conference on electronics, communications and control (ICECC)* (2011), IEEE, pp. 1028–1031.

[58] COSTIN, A., AND FRANCILLON, A. **Ghost in the air (traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices**. *black hat USA* (2012), 1–12.

[59] LIU, F. T., TING, K. M., AND ZHOU, Z.-H. **Isolation-based anomaly detection**. *ACM Transactions on Knowledge Discovery from Data (TKDD) 6*, 1 (2012), 1–39.

[60] NENCINI, L., DE ROSA, P., ASCARI, E., VINCI, B., AND ALEXEEVA, N. **SENSEable Pisa: A wireless sensor network for real-time noise mapping**. *Proceedings of the EURONOISE, Prague, Czech Republic* (2012), 10–13.

[61] AGGARWAL, C. C. **Outlier ensembles: position paper**. *ACM SIGKDD Explorations Newsletter 14*, 2 (2013), 49–58.

[62] BELL, M. C., AND GALATIOTO, F. **Novel wireless pervasive sensor network to improve the understanding of noise in street canyons**. *Applied Acoustics 74*, 1 (2013), 169–180.

[63] CAMPELLO, R. J., MOULAVI, D., AND SANDER, J. **Density-based clustering based on hierarchical density estimates**. In *Pacific-Asia conference on knowledge discovery and data mining* (2013), Springer, pp. 160–172.

[64] GRIGORIK, I. **High Performance Browser Networking: What every web developer should know about networking and web performance**. " O'Reilly Media, Inc.", 2013.

[65] KINGMA, D. P., AND WELLING, M. **Auto-encoding variational bayes**. *arXiv preprint arXiv:1312.6114* (2013).

[66] TESO, H. **Aircraft hacking: Practical aero series**. In *4th Hack in the Box Security Conference in Europe* (2013).

[67] WALLGREN, L., RAZA, S., AND VOIGT, T. **Routing attacks and countermeasures in the RPL-based internet of things**. *International Journal of Distributed Sensor Networks 9*, 8 (2013), 794326.

[68] WANG, C., CHEN, G., DONG, R., AND WANG, H. **Traffic noise monitoring and simulation research in Xiamen city based on the environmental internet of things**. *International Journal of Sustainable Development & World Ecology 20*, 3 (2013), 248–253.

[69] XIN, W., CHANG, J., MUTHUSWAMY, S., AND TALAS, M. **"Midtown in Motion": A new active traffic management methodology and its implementation in New York City**. In *Transportation Research Board 92nd Annual Meeting* (2013).

[70] DE COENSEL, B., AND BOTTELDOOREN, D. **Smart sound monitoring for sound event detection and characterization**. In *43rd International Congress on Noise Control Engineering (Inter-Noise 2014)* (2014).

[71] DOMINGUEZ, F., DAUWE, S., CUONG, N. T., CARIOLARO, D., TOUHAFI, A., DHOEDT, B., BOTTELDOOREN, D., AND STEENHAUT, K. **Towards an environmental measurement cloud: Delivering pollution awareness to the public**. *International Journal of Distributed Sensor Networks 10*, 3 (2014), 541360.

[72] KINGMA, D. P., AND BA, J. **Adam: A method for stochastic optimization**. *arXiv preprint arXiv:1412.6980* (2014).

[73] NEDIC, V., DESPOTOVIC, D., CVETANOVIC, S., DESPOTOVIC, M., AND BABIC, S. **Comparison of classical statistical methods and artificial neural network in traffic noise prediction**. *Environmental Impact Assessment Review 49* (2014), 24–30.

[74] PIMENTEL, M. A., CLIFTON, D. A., CLIFTON, L., AND TARASSENKO, L. **A review of novelty detection**. *Signal processing 99* (2014), 215–249.

[75] SCHÄFER, M., STROHMEIER, M., LENDERS, V., MARTINOVIC, I., AND WILHELM, M. **Bringing up OpenSky: A large-scale ADS-B sensor network for research**. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks* (2014), IEEE, pp. 83–94.

[76] STROHMEIER, M., LENDERS, V., AND MARTINOVIC, I. **On the security of the automatic dependent surveillance-broadcast protocol**. *IEEE Communications Surveys & Tutorials 17*, 2 (2014), 1066–1087.

[77] STROHMEIER, M., SCHÄFER, M., LENDERS, V., AND MARTINOVIC, I. **Realities and challenges of nextgen air traffic management: the case of ADS-B**. *IEEE Communications Magazine 52*, 5 (2014), 111–118.

[78] WESSON, K. D., HUMPHREYS, T. E., AND EVANS, B. L. **Can cryptography secure next generation air traffic surveillance?** *IEEE Security and Privacy Magazine* (2014).

[79] AMOOZADEH, M., RAGHURAMU, A., CHUAH, C.-N., GHOSAL, D., ZHANG, H. M., ROWE, J., AND LEVITT, K. **Security vulnerabilities of connected vehicle streams and their impact on cooperative driving**. *IEEE Communications Magazine 53*, 6 (2015), 126–132.

[80] AN, J., AND CHO, S. **Variational autoencoder based anomaly detection using reconstruction probability**. *Special Lecture on IE 2*, 1 (2015), 1–18.

[81] BOX, G. E., JENKINS, G. M., REINSEL, G. C., AND LJUNG, G. M. **Time series analysis: forecasting and control**. John Wiley & Sons, 2015.

[82] FARRÉS, J. C. **Barcelona noise monitoring network**. In *Proceedings of the Euronoise* (2015), pp. 218–220.

[83] FOGGIA, P., PETKOV, N., SAGGESE, A., STRISCIUGLIO, N., AND VENTO, M. **Audio surveillance of roads: A system for detecting anomalous sounds**. *IEEE transactions on intelligent transportation systems 17*, 1 (2015), 279–288.

[84] LAPTEV, N., AMIZADEH, S., AND FLINT, I. **Generic and scalable framework for automated time-series anomaly detection**. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (2015), pp. 1939–1947.

[85] LECUN, Y., BENGIO, Y., AND HINTON, G. **Deep learning**. *nature 521*, 7553 (2015), 436–444.

[86] MALHOTRA, P., VIG, L., SHROFF, G., AGARWAL, P., AND OTHERS. **Long short term memory networks for anomaly detection in time series**. In *Proceedings* (2015), vol. 89, pp. 89–94.

[87] MIETLICKI, F., MIETLICKI, C., AND SINEAU, M. **An innovative approach for long term environmental noise measurement: Rumeur network in the paris region**. In *Proceedings of the EuroNoise* (2015).

[88] PAULO, J., FAZENDA, P., OLIVEIRA, T., CARVALHO, C., AND FELIX, M. **Framework to monitor sound events in the city supported by the FIWARE platform**. In *Proceedings of the 46o Congreso Español de Acústica, Valencia, Spain* (2015), pp. 21–23.

[89] SCHMIDHUBER, J. **Deep learning**. *Scholarpedia 10*, 11 (2015), 32832.

[90] BELLUCCI, P., AND CRUCIANI, F. R. **Implementing the dynamap system in the suburban area of rome**. In *Inter-Noise and Noise-Con Congress and Conference Proceedings* (2016), vol. 253, Institute of Noise Control Engineering, pp. 5518–5529.

[91] CHEN, T., AND GUESTRIN, C. **Xgboost: A scalable tree boosting system**. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), pp. 785–794.

[92] JANG, E., GU, S., AND POOLE, B. **Categorical reparameterization with gumbel-softmax**. *arXiv preprint arXiv:1611.01144* (2016).

[93] LIANG, G., ZHAO, J., LUO, F., WELLER, S. R., AND DONG, Z. Y. **A review of false data injection attacks against modern power systems**. *IEEE Transactions on Smart Grid 8*, 4 (2016), 1630–1638.

[94] NAVARRO, J., TOMASGABARRON, J., AND ESCOLANO, J. **On the application of big data techniques to noise monitoring of smart cities**. In *Euroregio 2016 Oporto* (2016), Portuguese Acoustical Society (SPA) and Spanish Acoustics Society (SEA).

[95] PAULO, J., FAZENDA, P., OLIVEIRA, T., AND CASALEIRO, J. **Continuos sound analysis in urban environments supported by FIWARE platform**. *Proceedings of the EuroRegio2016/TecniAcústica 16* (2016), 1–10.

[96] RAINHAM, D. **A wireless sensor network for urban environmental health monitoring: Urbansense**. In *IOP Conference Series: Earth and Environmental Science* (2016), vol. 34, IOP Publishing, p. 012028.

[97] SEVILLANO, X., SOCORÓ, J. C., ALÍAS, F., BELLUCCI, P., PERUZZI, L., RADAELLI, S., COPPI, P., NENCINI, L., CERNIGLIA, A., BISCEGLIE, A., AND OTHERS. **Dynamap–development of low cost sensors networks for real time noise mapping**. *Noise mapping 3*, 1 (2016).

[98] VONDRICK, C., PIRSIAVASH, H., AND TORRALBA, A. **Generating videos with scene dynamics**. *Advances in neural information processing systems 29* (2016).

[99] BARTALUCCI, C., BORCHI, F., CARFAGNI, M., FURFERI, R., GOVERNI, L., SILVAGGIO, R., CURCURUTO, S., AND NENCINI, L. **Design of a prototype of a smart noise monitoring system**. In *Proceedings of the 24th International Congress on Sound and Vibration (ICSV24), London, UK* (2017), pp. 23–27.

[100] CAKIR, E., PARASCANDOLO, G., HEITTOLA, T., HUTTUNEN, H., AND VIRTANEN, T. **Convolutional recurrent neural networks for polyphonic sound event detection**. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 25*, 6 (2017), 1291–1303.

[101] JARIWALA, H. J., SYED, H. S., PANDYA, M. J., AND GAJERA, Y. M. **Noise pollution & human health: a review**. *Noise and Air Pollutions: Challenges and Opportunities, Ahmedabad: LD College of Eng* (2017).

[102] KE, G., MENG, Q., FINLEY, T., WANG, T., CHEN, W., MA, W., YE, Q., AND LIU, T.-Y. **Lightgbm: A highly efficient gradient boosting decision tree**. In *Advances in neural information processing systems* (2017), pp. 3146–3154.

[103] MUELLER, N., ROJAS-RUEDA, D., BASAGAÑA, X., CIRACH, M., COLE-HUNTER, T., DADVAND, P., DONAIRE-GONZALEZ, D., FORASTER, M., GASCON, M., MARTINEZ, D., AND OTHERS. **Health impacts related to urban and transport planning: A burden of disease assessment**. *Environment international 107* (2017), 243–257.

[104] MYDLARZ, C., SALAMON, J., AND BELLO, J. P. **The implementation of low-cost urban acoustic monitoring devices**. *Applied Acoustics 117* (2017), 207–218.

[105] SALEHINEJAD, H., SANKAR, S., BARFETT, J., COLAK, E., AND VALAEE, S. **Recent advances in recurrent neural networks**. *arXiv preprint arXiv:1801.01078* (2017).

[106] SOCORÓ, J. C., ALÍAS, F., AND ALSINA-PAGÈS, R. M. **An anomalous noise events detector for dynamic road traffic noise mapping in real-life urban and suburban environments**. *Sensors 17*, 10 (2017), 2323.

[107] SUN, J., ELLERBROEK, J., AND HOEKSTRA, J. **Flight extraction and phase identification for large automatic dependent surveillance–broadcast datasets**. *Journal of Aerospace Information Systems 14*, 10 (2017), 566–572.

[108] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. **Attention is all you need**. *Advances in neural information processing systems 30* (2017).

[109] VINCI, B., TONACCI, A., CAUDAI, C., DE ROSA, P., NENCINI, L., AND PRATALI, L. **The senseable pisa project: Citizen-participation in monitoring acoustic climate of mediterranean city centers**. *CLEAN–Soil, Air, Water 45*, 7 (2017), 1600137.

[110] ZAMBON, G., BENOCCI, R., BISCEGLIE, A., ROMAN, H. E., AND BELLUCCI, P. **The life dynamap project: Towards a procedure for dynamic noise mapping in urban areas**. *Applied Acoustics 124* (2017), 52–60.

[111] BROWN, A., AND DE COENSEL, B. **A study of the performance of a generalized exceedance algorithm for detecting noise events caused by road traffic**. *Applied Acoustics 138* (2018), 101–114.

[112] CRETIN, A., LEGEARD, B., PEUREUX, F., AND VERNOTTE, A. **Increasing the resilience of atc systems against false data injection attacks using DSL-based testing**. In *International Conference on Research in Air Transportation* (2018).

[113] FARRÉS, J. C., AND NOVAS, J. C. **Issues and challenges to improve the barcelona noise monitoring network**. In *Proceedings of the 11th European Congress and Exposition on Noise Control Engineering* (2018), pp. 27–31.

[114] HABLER, E., AND SHABTAI, A. **Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages**. *Computers & Security 78* (2018), 155–173.

[115] KHAN, M. A., AND SALAH, K. **IoT security: Review, blockchain solutions, and open challenges**. *Future generation computer systems 82* (2018), 395–411.

[116] KUMAR, P. R., RAJ, P. H., AND JELCIANA, P. **Exploring data security issues and solutions in cloud computing**. *Procedia Computer Science 125* (2018), 691–697.

[117] MIETLICKI, C., AND MIETLICKI, F. **Medusa: A new approach for noise management and control in urban environment**. In *Proceedings of the EuroNoise* (2018), pp. 727–730.

[118] MUNIR, M., SIDDIQUI, S. A., DENGEL, A., AND AHMED, S. **Deepant: A deep learning approach for unsupervised anomaly detection in time series**. *Ieee Access 7* (2018), 1991–2005.

[119] NAVARATHNA, P. J., AND MALAGI, V. P. **Artificial intelligence in smart city analysis**. In *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)* (2018), pp. 44–47.

[120] PECKENS, C., PORTER, C., AND RINK, T. **Wireless sensor networks for long-term monitoring of urban noise**. *Sensors 18*, 9 (2018), 3161.

[121] SILVA, B. N., KHAN, M., AND HAN, K. **Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities**. *Sustainable Cities and Society 38* (2018), 697 – 713.

[122] AKERMAN, S., HABLER, E., AND SHABTAI, A. **VizADS-B: Analyzing sequences of ADS-B images using explainable convolutional LSTM encoder-decoder to detect cyber attacks**. *arXiv preprint arXiv:1906.07921* (2019).

[123] ALÍAS, F., AND ALSINA-PAGÈS, R. M. **Review of wireless acoustic sensor networks for environmental noise monitoring in smart cities**. *Journal of sensors 2019* (2019).

[124] ARULKUMARAN, K., CULLY, A., AND TOGELIUS, J. **Alphastar: An evolutionary computation perspective**. In *Proceedings of the genetic and evolutionary computation conference companion* (2019), pp. 314–315.

[125] BASORA, L., OLIVE, X., AND DUBOT, T. **Recent advances in anomaly detection methods applied to aviation**. *Aerospace 6*, 11 (2019), 117.

[126] CIRILLO, F., SOLMAZ, G., BERZ, E. L., BAUER, M., CHENG, B., AND KOVACS, E. **A standard-based open source IoT platform: FIWARE**. *IEEE Internet of Things Magazine 2*, 3 (2019), 12–18.

[127] KAWAGUCHI, Y., TANABE, R., ENDO, T., ICHIGE, K., AND HAMADA, K. **Anomaly detection based on an ensemble of dereverberation and anomalous sound extraction**. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 865–869.

[128] LI, G., MULLER, M., THABET, A., AND GHANEM, B. **Deepgcns: Can gcns go as deep as cnns?** In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 9267–9276.

[129] PARK, J., LEE, M., CHANG, H. J., LEE, K., AND CHOI, J. Y. **Symmetric graph convolutional autoencoder for unsupervised graph representation learning**. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 6519–6528.

[130] REN, H., XU, B., WANG, Y., YI, C., HUANG, C., KOU, X., XING, T., YANG, M., TONG, J., AND ZHANG, Q. **Time-series anomaly detection service at Microsoft**. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (2019), pp. 3009–3017.

[131] UNITED NATIONS DEPARTMENT OF ECONOMIC AND SOCIAL AFFAIRS. **World Urbanization Prospects: The 2018 Revision**. United Nations, 2019.

[132] WANG, S., LI, C., AND LIM, A. **Why are the ARIMA and SARIMA not sufficient**. *arXiv preprint arXiv:1904.07632* (2019).

[133] ZHANG, Z., ZHAO, Y., CANES, A., STEINBERG, D., LYASHEVSKA, O., AND WRITTEN ON BEHALF OF AME BIG-DATA CLINICAL TRIAL COLLABORATIVE GROUP. **Predictive analytics with gradient boosting in clinical medicine**. *Annals of translational medicine 7*, 7 (April 2019), 152.

[134] BRILAND, M., AND BOUQUET, F. **An approach for testing false data injection attack on data dependent industrial devices**. *Journal of Universal Computer Science 27*, 7 (2020), 774–792.

[135] BROWN, T., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J. D., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AND OTHERS. **Language models are few-shot learners**. *Advances in neural information processing systems 33* (2020), 1877–1901.

[136] CHEVROT, A., VERNOTTE, A., BERNABE, P., CRETIN, A., PEUREUX, F., AND LEGEARD, B. **Improved testing of AI-based anomaly detection systems using synthetic surveillance data**. In *Multidisciplinary Digital Publishing Institute Proceedings* (2020), vol. 59, p. 9.

[137] CRETIN, A., VERNOTTE, A., CHEVROT, A., PEUREUX, F., AND LEGEARD, B. **Test data generation for false data injection attack testing in air traffic surveillance**. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (2020), IEEE, pp. 143–152.

[138] EUROPEAN ENVIRONMENT AGENCY. **Environmental noise in Europe**. Publications Office of the European Union,, 2020.

[139] GAUTAM, K., PURI, V., TROMP, J. G., NGUYEN, N. G., AND VAN LE, C. **Internet of things (IoT) and deep neural network-based intelligent and conceptual model for smart city**. In *Frontiers in Intelligent Computing: Theory and Applications* (Singapore, 2020), S. C. Satapathy, V. Bhateja, B. L. Nguyen, N. G. Nguyen, and D.-N. Le, Eds., Springer Singapore, pp. 287–300.

[140] KARAM, R., SALOMON, M., AND COUTURIER, R. **A comparative study of deep learning architectures for detection of anomalous ADS-B messages**. In *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)* (2020), vol. 1, IEEE, pp. 241–246.

[141] KOSOWATZ, J. **10 Smart Cities**. *Mechanical Engineering 142*, 02 (02 2020), 32–37.

[142] NAVARRO, J. M., MARTÍNEZ-ESPAÑA, R., BUENO-CRESPO, A., MARTÍNEZ, R., AND CECILIA, J. M. **Sound levels forecasting in an acoustic sensor network using a deep neural network**. *Sensors 20*, 3 (2020), 903.

[143] NOURANI, V., GÖKÇEKUŞ, H., AND UMAR, I. K. **Artificial intelligence based ensemble model for prediction of vehicular traffic noise**. *Environmental research 180* (2020), 108852.

[144] NOURANI, V., GÖKÇEKUŞ, H., UMAR, I. K., AND NAJAFI, H. **An emotional artificial neural network for prediction of vehicular traffic noise**. *Science of the Total Environment 707* (2020), 136134.

[145] POWERS, D. M. **Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation**. *arXiv preprint arXiv:2010.16061* (2020).

[146] SHEN, L., LI, Z., AND KWOK, J. **Timeseries anomaly detection using temporal hierarchical one-class network**. *Advances in Neural Information Processing Systems 33* (2020), 13016–13026.

[147] WANG, J., ZOU, Y., AND DING, J. **ADS-B spoofing attack detection method based on LSTM**. *EURASIP Journal on Wireless Communications and Networking 2020*, 1 (2020), 1–12.

[148] YOOK, D., LEEM, S.-G., LEE, K., AND YOO, I.-C. **Many-to-many voice conversion using cycle-consistent variational autoencoder with multiple decoders**. In *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop* (2020), pp. 215–221.

[149] ZHANG, X., ZHAO, M., AND DONG, R. **Time-series prediction of environmental noise for urban IoT based on long short-term memory recurrent neural network**. *Applied Sciences 10*, 3 (2020), 1144.

[150] BRILAND, M., AND BOUQUET, F. **A language for modelling false data injection attacks in internet of things**. In *2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)* (2021), IEEE, pp. 1–8.

[151] CHEN, Z., CHEN, D., ZHANG, X., YUAN, Z., AND CHENG, X. **Learning graph structures with transformer for multivariate time series anomaly detection in IoT**. *IEEE Internet of Things Journal* (2021).

[152] D'ASCOLI, S., TOUVRON, H., LEAVITT, M. L., MORCOS, A. S., BIROLI, G., AND SAGUN, L. **ConViT: Improving vision transformers with soft convolutional**

inductive biases. In *International Conference on Machine Learning* (2021), PMLR, pp. 2286–2296.

[153] JUMPER, J., EVANS, R., PRITZEL, A., GREEN, T., FIGURNOV, M., RONNEBERGER, O., TUNYASUVUNAKOOL, K., BATES, R., ŽÍDEK, A., POTAPENKO, A., AND OTHERS. **Highly accurate protein structure prediction with AlphaFold**. *Nature 596*, 7873 (2021), 583–589.

[154] KHAN, S., NASEER, M., HAYAT, M., ZAMIR, S. W., KHAN, F. S., AND SHAH, M. **Transformers in vision: A survey**. *ACM Computing Surveys (CSUR)* (2021).

[155] KIRANYAZ, S., AVCI, O., ABDELJABER, O., INCE, T., GABBOUJ, M., AND INMAN, D. J. **1D convolutional neural networks and applications: A survey**. *Mechanical systems and signal processing 151* (2021), 107398.

[156] LIM, B., ARIK, S. Ö., LOEFF, N., AND PFISTER, T. **Temporal fusion transformers for interpretable multi-horizon time series forecasting**. *International Journal of Forecasting* (2021).

[157] LUO, P., WANG, B., LI, T., AND TIAN, J. **ADS-B anomaly data detection model based on VAE-SVDD**. *Computers & Security 104* (2021), 102213.

[158] PANG, G., SHEN, C., CAO, L., AND HENGEL, A. V. D. **Deep learning for anomaly detection: A review**. *ACM Computing Surveys (CSUR) 54*, 2 (2021), 1–38.

[159] SANCHEZ-LENGELING, B., REIF, E., PEARCE, A., AND WILTSCHKO, A. B. **A gentle introduction to graph neural networks**. *Distill 6*, 9 (2021), e33.

[160] SUN, J. **The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals**, 2 ed. TU Delft OPEN Publishing, 2021.

[161] VERNOTTE, A., CRETIN, A., LEGEARD, B., AND PEUREUX, F. **A domain-specific language to design false data injection tests for air traffic control systems**. *International Journal on Software Tools for Technology Transfer* (2021), 1–32.

[162] CHEVROT, A., VERNOTTE, A., AND LEGEARD, B. **CAE: Contextual auto-encoder for multivariate time-series anomaly detection in air transportation**. *Computers & Security* (2022), 102652.

[163] KARAM, R., SALOMON, M., AND COUTURIER, R. **Supervised ADS-B anomaly detection using a false data generator**. In *2022 2nd International Conference on Computer, Control and Robotics (ICCCR)* (2022), pp. 218–223.

[164] LI, Y., CHOI, D., CHUNG, J., KUSHMAN, N., SCHRITTWIESER, J., LEBLOND, R., ECCLES, T., KEELING, J., GIMENO, F., LAGO, A. D., AND OTHERS. **Competition-level code generation with alphacode**. *arXiv preprint arXiv:2203.07814* (2022).

[165] DUNSTONE, G. **ADS-B technology the experience in australia**. https://www.icao.int/SAM/Documents/2017-ADSB/10%20Australia.pdf.

[166] RICHARDS, W. R., O'BRIEN, K., AND MILLER, D. C. **Aero - new air traffic surveillance technology**. https://www.boeing.com/commercial/aeromagazine/articles/qtr_02_10/2/.

[167] WEBER, N. **Why lstms stop your gradients from vanishing: A view from the backwards pass (weberna's blog)**. https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html.

List of Figures

List of Tables

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

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

Title: Automatic detection of business data anomalies with deep learning and application to the ADS-B protocol

Keywords: Deep learning, Cybersecurity, ADS-B protocol, Machine learning

Abstract:

The use of Machine Learning (ML) and Deep Learning (DL) for security anomaly detection is an extremely active topic. Anomaly detection touches many domains, namely air traffic control, IoT, etc. One main air traffic control technology is the ADS-B protocol (Automatic Dependent Surveillance-Broadcast). It constitutes an air traffic control data source based on satellite positioning. Each aircraft periodically sends via ADS-B messages its information to ground stations and other aircraft. ADS-B is becoming globally mandatory but it still lacks security measures like encryption and authentication. One way to tackle this problem is ML and DL based ADS-B anomaly detection. Supervised ADS-B anomaly detection was the main focus of this thesis due to its performance advantage compared to unsupervised methods. However, it suffers from the lack of labeled data. In order to obtain enough labeled anomalies and normal data, a false data generator was used. To the best of our knowledge, this thesis is the only work which used supervised ADS-B anomaly detection. Our approach gave very promising results in detecting various types of attacks. The best performance was obtained using the Long Short-Term Memory (LSTM) model. In addition, as a secondary contribution in this thesis, different DL and ML approaches were studied in order to forecast noise levels and detect punctual noise level anomalies. The data were gathered from an IoT system more specifically a network of smart parkmeters. The results of our study inferred that such methods, preferably a 1D Convolutional Long Short-Term Memory (CNN-LSTM), can be successfully used in environmental noise monitoring applications.

Titre : Détection automatique d'anomalies de données métiers avec deep learning et application au protocole ADS-B

Mots-clés : Apprentissage en profondeur, Cybersécurité, Protocole ADS-B, Apprentissage automatique

Résumé :

L'utilisation du Machine Learning (ML) et du Deep Learning (DL) pour la détection des anomalies de sécurité est un sujet extrêmement actif. La détection d'anomalies s'applique à de nombreux domaines, à savoir le contrôle du trafic aérien, l'IoT, etc. L'une des principales technologies de contrôle du trafic aérien est le protocole ADS-B (Automatic Dependent Surveillance-Broadcast). Il constitue une source de données de contrôle du trafic aérien basée sur le positionnement par satellite. Chaque avion envoie périodiquement via des messages ADS-B ses informations aux stations au sol et aux autres avions. L'ADS-B devient obligatoire dans le monde entier, mais il n'intègre pas des mesures de sécurité comme le cryptage et l'authentification, ce qui le rend vulnérable à l'injection de fausses données. Une façon de s'attaquer à ce problème est la détection d'anomalies ADS-B basée sur des approaches de ML et DL. La détection supervisée des anomalies ADS-B a été le point central de cette thèse en raison de son avantage en termes de performance par rapport aux méthodes non supervisées. Cependant, une approche supervisée suppose d'avoir des données labellisées. Afin d'obtenir suffisamment d'anomalies labellisées et de données normales, un générateur de fausses données a été utilisé. A notre connaissance, cette thèse est le seul travail qui a utilisé la détection supervisée d'anomalies ADS-B. Notre approche a donné des résultats très prometteurs dans la détection de différents types d'attaques. Les meilleures performances ont été obtenues en utilisant un modèle Long Short-Term Memory (LSTM). Comme contribution secondaire, différentes approches de ML et DL ont été étudiées afin de prévoir les niveaux de bruit et de détecter des anomalies ponctuelles de niveau de bruit. Les données sont issues d'un système IoT et plus précisément d'un réseau de parcmètres. Les résultats obtenus ont permis de déduire que de telles méthodes, de préférence un hybride 1D combinant réseau de neurones convolutionnels et Long Short-Term Memory (CNN-LSTM), peuvent être utilisées avec succès dans les applications de surveillance du bruit environnemental.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

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