Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
2022 | LIU-IDA/LITH-EX-A--22/063--SE

Anomalous Behavior Detection in Aircraft based Automatic Dependent Surveillance–Broadcast (ADS-B) system using Deep Graph Convolution and Generative model (GA-GAN)

- Jayesh Dinesh Kenaudekar

Supervisor : Phd. Student Suleman Khan
Examiner : Prof. Andrei Grutov

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

The Automatic Dependent Surveillance-Broadcast (ADS-B) is a key component of the Next Generation Air Transportation System (Next Gen) that manages the increasingly congested airspace and operation. From Jan 2020, the U.S. Federal Aviation Administration (FAA) mandated the use of (ADS-B) as a key component of Next Gen project. ADS-B provides accurate aircraft localization via satellite navigation and efficient air traffic management, and also improves the safety of thousands of passengers travelling worldwide. While the benefits of ADS-B are well known, the fact that ADS-B is an open protocol introduces various exploitable security vulnerabilities. One practical threat is the ADS-B spoofing attack that targets the ground station, in which the ground-based attacker manipulates the International Civil Aviation Organization (ICAO) address (which is a unique identifier for each aircraft) in the ADS-B forwarded messages to fake the appearance of non-existent aircraft or masquerade as a trusted aircraft. As a result, this type of attack can confuse and misguide the aircraft pilots or the air traffic control personnel and cause dangerous maneuvers.

In this project, we intend to build a robust Intrusion Detection System (IDS) to detect anomalous behavior and classify attacks in an aircraft ADS-B protocol in real time during air-ground communication. The IDS system we propose is a 3 stage deep learning framework built using Spatial Graph Convolution Networks and Deep auto-regressive generative model. In stage 1 we use a Graph convolution network architecture to classify the data as attacked or normal in the entire airspace of an operating aircraft. In stage 2 we analyze the sequences of air-space states to identify anomalies using a generative Wavenet model and simultaneously output feature under attack. Final stage consist of aircraft(ICAO) classification module based on unique RF transmitter signal characteristics of an aircraft. This allows the ground station operator to examine each incoming message based on the Phy-layer features as well as message data field (such as, position, velocity, altitude) and flag suspicious messages. The model is trained in a supervised fashion using federated learning where the data remains private to the data owner, i.e.: aircraft-ground station without data being explicitly sent to the cloud server. The server only receives the learned parameters for inference, thereby training the entire model on the edge, thus preserving data-privacy and potential adversarial attacks. We aim to achieve a high precision real-time IDS system, with very low false alarm rate for real world deployment.

Acknowledgments

I would like to express my sincere gratitude towards my examiner Professor Andrei Gurtov and my supervisor Suleman Khan for giving me the opportunity to work on this thesis topic and providing me with an excellent support and guidance throughout my research degree project. I am extremely grateful and thankful for this opportunity.

Also we would like to thank Shaun Ying and Junzi Sun, for sharing with us the entire dataset on raw IQ samples and decoded ADS-B messages to help compare our research work against. Also, a big thank you to Ola Runeson, and team for his contribution to the openScope simulator project and Joakim Thorn, Alex Wahlgren for their existing thesis work on detecting ADS-B attacks using machine learning. I would also like to mention the open source tools and frameworks that have been of extremely great help in developing and building my degree project, firstly, to the Pytorch community and team at Meta who help maintains the library. Additionally, I'm also grateful to the TensorFlow board and its community members for their excellent documentation, tools and services.

Finally, I would like to acknowledge the entire team at ADIT Department, Linköping University for their knowledge sharing and contribution to the related work.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Over the previous few decades, demand for air travel has gradually increased. According to the Federal Aviation Administration (FAA), by 2033, the number of passengers in commercial aviation will have risen to 1.15 billion [9]. Eurocontrol predicts 1.6 billion air passengers per year in the airspace by the early 2030s [8]. Air cargo, combat planes, and unmanned aerial vehicles are likely to increase air traffic in the coming years. As a consequence, the number of aircraft in the air will continue to grow, making the airspace even more congested. To provide safe navigation and reduce the cost of air traffic control (ATC), the aviation community has shifted from uncooperative and independent air traffic surveillance, such as primary surveillance radar (PSR) and secondary surveillance radar (SSR), to cooperative and dependent air traffic surveillance (CDS), such as ADS-B [22].

The International Civil Aviation Organization (ICAO) and the Federal Aviation Administration (FAA) have approved ADS-B [1] as a modern implementation of SSR that is projected to play a significant role in aviation in the near future [16]. Using the ADS-B technology, aircraft movements may be tracked constantly and accurately in crowded airspace. An aircraft equipped with an ADS-B transponder (transmitter-responder) is capable of obtaining its location from the navigation satellite system and then broadcasts the aircraft's flight number, speed, position, and altitude at an average rate of 4.2 messages per second.

Unlike cost and accuracy, which were important factors in the development of ADS-B, security was thrown to the wayside. As a consequence, a widely used technology with severely weakened security emerged, notably in terms of the protocol mechanism, as shown below:

No Encryption and Authentication: ADS-B Messages are broadcast in plain text, with no authentication code or digital signature, and may therefore be replayed, modified, or falsified.

No aircraft and ground authentication: Authorized aircraft or ATC stations don't have to prove their identity before sending, so there's no way to distinguish between them and unauthorized ones. So, an unauthorized entity can add messages or change the reports of an authorized entity.

It is reasonably easy to attack ADS-B security using off-the-shelf hardware and software shown in the previous research [35][5][7][3]. The possibility of exploiting the ADS-B system puts billions of passengers at risk every year. Thus academics and businesses have worked together to find ways to address the security issue.

In order to ensure secure message broadcast and avoid eavesdropping, previous research proposed the use of encryption [4], aircraft authentication through challenge-response [20], and message authentication [5][10]. Another method was to employ additional sensors or nodes to validate velocity and position data as well as secure broadcast transmission. However, the most of these methods involve changes to the architecture so that key exchange or trust can be established. Since the FAA requires all aircraft in US airspace to use ADS-B by 2020, which is already a requirement for some aircraft in Europe, and because of the strict rules for implementing avionic systems, it is not possible to make changes to the current protocol at this point. The ADS-B protocol was designed and developed in the early 1990s.

In this thesis, we provide an alternative security method for identifying abnormal ADS-B messages. More specifically, our technique is intended to detect spoofed or altered ADS-B messages that have been delivered by an attacker or compromised aircraft. The solution that has been presented does not require any modifications or extra participating nodes and/or sensors, and it enables airplanes to identify abnormalities in congested airspace on their own without any assistance. Our method detects message spoofing by observing a sequence of messages and determining their trustworthiness. Since flights between airports usually take place via similar routes, we use and train an auto-regressive generative model (Wavenet) based on previous (legitimate) flights for a given route. Using such a model, each aircraft can independently evaluate received ADS-B messages and identify deviations from the legitimate flight path.

1.2 Aim

The aim of this thesis is to highlight security issues in ADS-B systems and proposes a deep learning-based intrusion detection system (IDS) for the ADS-B, which can detect attacks on the ground side. Our IDS solution not only detects anomalies in air-ground communication but can also classify real and ghost aircraft. Our proposed IDS is based on different deep learning architectures, i.e., Deep Graph Convolution (DGC), Wavenet, and Generative Adversarial Network (GAN). In DGC, we represent the entire aircraft as nodes in a graph and use a message passing neural network in DGC in order to classify normal or attack messages. Similarly, we use GAN to generate (fake) attack data from real ADS-B messages such that each aircraft with a unique ICAO address has attack data along with its normal message data. Finally, we use Wavenet to predict the behavior of each feature given the history of ADS-B messages. In order to distinguish between real and spoofed aircraft, we use Deep Neural Network (DNN) for aircraft classification.

1.3 Main issues addressed by this thesis are:

1. Exploitation of the vulnerabilities of Automatic Dependent Surveillance Broadcast (ADS-B) system.
2. Real-time time solution for highly accurate attack detection score with very low false alarm rate for air traffic monitoring using ADS-B protocol.
3. Design end-to-end framework for identifying a variety of attacks using multiple machine learner systems stacked together.

1.4 Research Questions

The primary questions that this study sought to answer are:

1. Study various types of attacks that might be possible on ADS-B protocol?
2. Since ADS-B is a transparent protocol with very limited security measures, could an IDS be a reasonable solution?
3. How can we prevent adversarial attacks using our proposed solution ?
4. Is there a measure to evaluate model robustness? And calculate model uncertainty ?

1.5 De-limitations

The dataset used for attack classification has been taken from the ADS-B data repository containing flight extraction of ADS-B decoded messages which was collected and stored in MongoDB database by Junzi Sun and team published in aerospace information systems [37]. Since this only contains real ADS-B message data, we do not have attack data for it to be used for message classification; thus, we need to generate our own attack data from real data by synthesizing fake data.

However, the synthesized data is not guaranteed to cover every aspect in which an intruder might manipulate or attack the ADS-B message data. The model we train upon might also be biased to the dataset we use. Every operating region will have different flight characteristics that need to be taken care of when deploying the model.

1.6 Risk Analysis

Adversarial attacks - *medium*

In order to secure our system further, we need to prevent our model being trained on a data set that has been externally modified and make sure no external entity can have direct access to the underlying data. Since the model is a representation of its data, if the data set is being hacked or modified according to the intended purposes of the intruder, our model as well will learn these irregularities in it. Thus, in-order to prevent this from happening we need to train our model using decentralized federated learning, wherein no user gets direct access to the data and only trained parameters of the model are returned to you.

Not enough data to validate - *medium*

Since the data set collected is highly imbalanced and limited in nature. It is hard to evaluate the model on the provided dataset and gather a conclusive outcome. To tackle this problem, we need to augment our data sets classes with more training examples using techniques such as generative models and over-sampling. We can use GAN or Generative adversarial networks to generate new training examples and over-sample from the underlying data distribution.

Tiny model size at the edge - *high*

Our goal is to package an intrusion detection system that is real time and robust to new uncertainties. To achieve this we need to train and deploy the model at the edge or on an aircraft rather than hosting it onto a server down at the ground station. To realize this, we need smaller models with faster inference time that can detect anomalies in the incoming stream of messages and identify threats if any without significant delays.

Statistical inference and likelihood estimation - *low*

To be more confident of our outputs obtained from the classification model. We could perform bootstrapping, which is sampling with replacement from the dataset and fit the model. This experiment is done repeatedly for n number of times and the results are noted. These results are used to make statistical conclusions such as the confidence score or skill of the model.

2 Background

This section covers the relevant background on the ADS-B message protocol and briefly introduces the intrusion detection system framework for continuous and real-time detection of intrusion attacks in air-traffic communication and thereafter briefly introduces the use of federated learning.

The ADS-B system is a crucial component of the Next Generation Air Transportation System that manages the increasingly congested airspace. ADS-B stands for Automatic Dependent Surveillance[42]. The term automatic since it periodically transmits information with no pilot supervision, and dependent states from which it derives its position and velocity information at each time from GPS (Global Positioning System) satellites. Air-planes broadcast their current status once every minute through the (ADS-B) system. This ADS-B data can be used by other planes in the vicinity as well as by the Air Traffic Controller (ATC) for planning

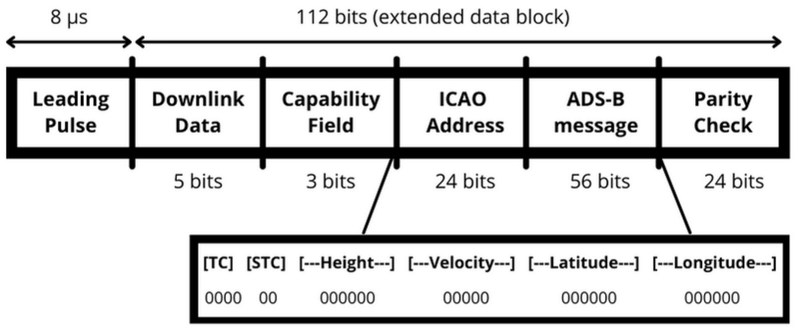


Figure 2.1: ADS-B Message Field

and directing air borne traffic. As of this writing thesis all aircrafts in US are equiped with ADS-B system mandated by FAA Federal Regulations.

ADS-B replaces the radar technology with satellites, bringing in major advantages. It provides accurate aircraft localization and efficient air traffic management, and also improves the safety of billions of current and future travelling passengers. As the position monitoring gets more accurate, the air traffic control system will be able to handle a greater number of air-traffic. The new NextGen ADS-B infrastructure system contains simple UHF radio stations which are considerably cheaper to set up and maintain compared to the old surveillance radar ground stations [16]. It also has the potential benefit of reducing maintenance and operating costs of air traffic control system. While the benefits of ADS-B are well known, but the lack of basic security measures like encryption and authentication introduces various exploitable security vulnerabilities. One practical threat is the ADS-B spoofing attack that targets the ground station, in which the aircraft-based or a ground-based attacker manipulates the ICAO address (a unique identifier for an aircraft) in the ADS-B messages to fake the appearance of non-existent aircraft or masquerade as a trusted aircraft. As a result, this attack can confuse the pilots or the air traffic control personnel and cause life-threatening events.

Due to the very reason and our large scale dependence on ADS-B system as the primary mode of identifying and localizing air-crafts brings the prospects of these signals being spoofed by adversaries with malicious intentions. The events of 9/11 warns us to do the needful to ensure the safety of all passengers travelling on board. These fateful events also establish the fact that malicious entities are always looking for novel ways to target and induce harm to society. Given the ease with which it can be spoofed, we need to ensure that this ever important ADS-B system cannot be compromised by such attacks.

2.1 ADS-B Protocol Stack

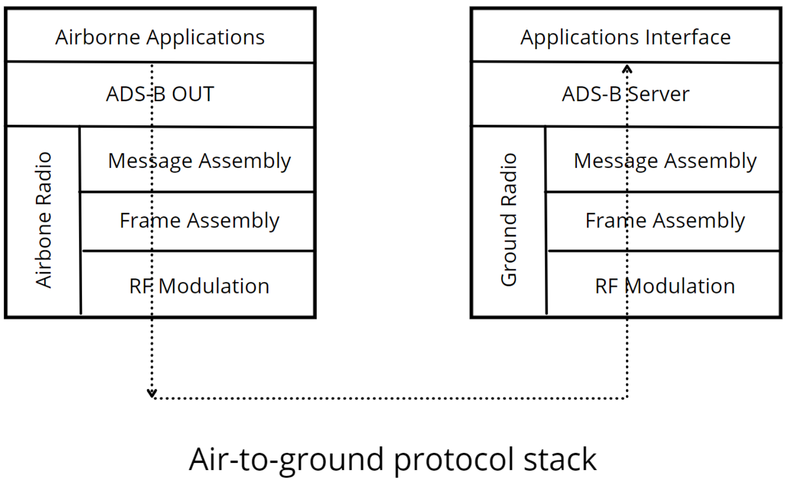


Figure 2.2: Protocol stack

Figure 2.3 shows the air-to-ground protocol stacks of the ADS-B, the airbone segment consists of three layers namely; airborne applications, ADS-B IN/OUT, and airborne radio.

On the other end the ground segment is also composed of three layers: the FAA applications, ADS-B server, and ground radio. Furthermore, the airbone radio has 3 sub-segments i.e. ADS-B message assembly, frame assembly, and RF modulation/demodulation. Here the information is taken from airbone application layer and is prepared for broadcasting, this is done in the airbone radio layers which constructs the ADS-B message in the format shown in fig 2.1, the constructed frames are then modulated and broadcast through the air. Finally, at the receiver end, the data received is demodulated and extracted for ADS-B message information and delivered to the application layer of the receiver.

2.2 ADS-B Vulnerabilities

The vulnerabilities faced by the ADS-B system are essentially derived from the broadcast nature of radio frequency communications and the fact that messages are broadcast as unencrypted plain text[15][7]. This makes the aircraft operating status information prone to new attacks by malicious attackers and easily manipulating the messages. At present, the types of attacks that exist for the ADS-B system are mainly divided into eavesdropping, jamming, message injection, message deletion, and message modification.

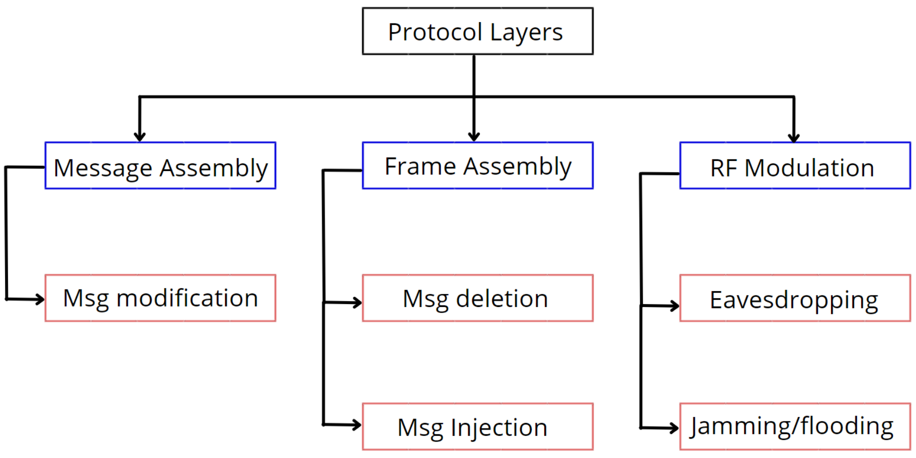


Figure 2.3: Classification of ADS-B attacks with respect to its protocol layers

2.3 Types of ADS-B attacks

Functional operations of ADS-B message protocol can be classified into two categories; ADS-B OUT and ADS-B IN, ADS-B OUT continuously broadcasts speed, altitude, position, identity and rate of climb. Whereas ADS-B IN is an optional service in which an aircraft with appropriate equipment receives the ADS-B messages and displays the information about other operating aircraft's in nearby region.

The former enables the aircraft to receive and display detailed information broadcast by other aircraft operating in the same area. The latter sends the aircraft's position information and other additional information to other aircraft or ground station controllers at a certain

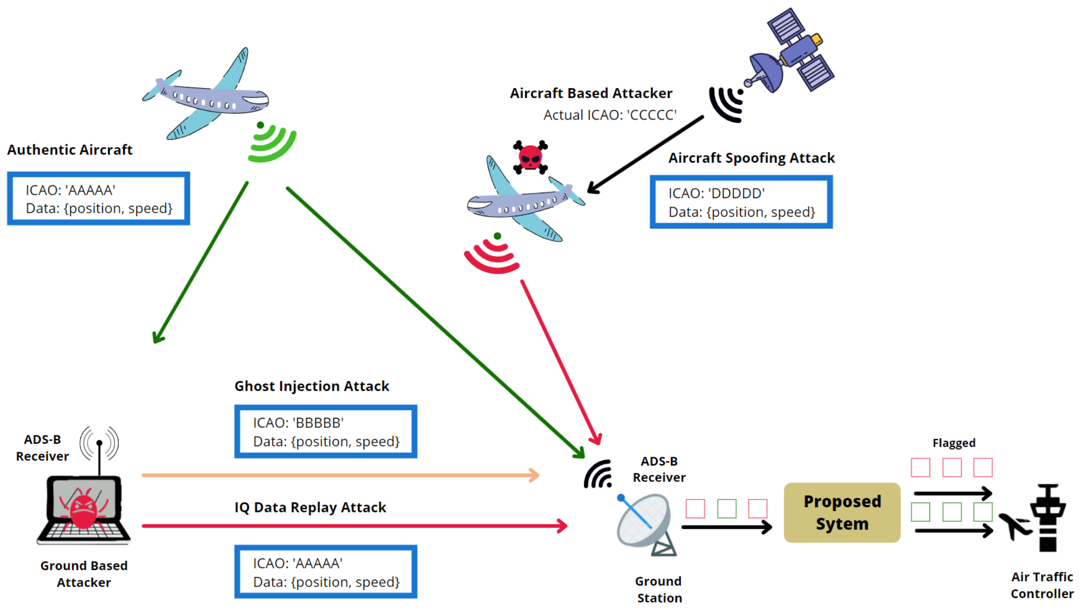


Figure 2.4: Attacks on ADS-B System

period, mainly including aircraft identification information, speed, heading direction, and climb rate etc. The ADS-B protocol format is shown in Fig.2.1.

We can further classify various ADS-B attacks according to the protocol layers they compromise [27]. For instance, the message modification targets the message assembly, whereas message deletion and injection attacks compromise the frame assembly sub-layer and finally since eavesdropping and jamming attacks are usually in the physical layer they compromise the RF modulation layer.

Jamming/Flooding

Making an aircraft disappear for a certain period of time. This scenario represents the unexpected disappearance of an aircraft, which could be caused by a denial-of-service attack (Dos) attack in which an aircraft is prevented from sending or receiving ADS-B messages.

Ghost aircraft (message injection)

In this attack, an evil with bad intentions injects an aircraft advancing on a legitimate route taken from previous flights in the vicinity or nearby areas. Therefore, the values (position, height, speed, and head direction) are correct and progress at a natural rate. The injection is performed starting from a randomly selected point on the route. The appearance of the aircraft in the airspace is sudden..

Aircraft spoofing attack

In this attack, an aircraft-based attacker (malicious aircraft) attempts to masquerade as a known or trusted aircraft by spoofing the ICAO address of the original aircraft and hiding its true identity. Since the aircraft is physically present, the masquerading attack will not be detected even if the secondary radar surveillance system is deployed. These are also called Impersonation attacks.

Message Modification

Modifying the underlying ADS-B messages in the network channel is the most challenging attack an attacker can perform. They need to have access to legitimate network equipment, which is very difficult. One such approach is the Bit flipping which means that the attacker superimposes a forged signal trying to flip the 0′s to 1′s and 1′s to 0′s.

Eavesdropping (Message interception)

Eavesdropping, also called a message interception attack, is the action of listening to broadcast transmissions, which is the most straightforward security vulnerability in ADS-B due to the lack of encryption.

Replay Attacks

In this attack, the ground-based attacker records the message contents or the IQ/message data of the received authentic ADS-B messages using the Software-defined radio device and then re-transmits the same messages later without changing the message contents. Compared to the message replay attack, the IQ data replay attack is made cautiously and surreptitiously, as the recorded IQ data incorporates a lot of information about the Doppler effect, and the transmitter characteristics (such as the carrier frequency offset), and the channel characteristics are difficult to mimic otherwise.

2.4 Proposed Methodology

Recently there have been several studies on detecting and identifying spoofed ADS-B signals [21]. Many of these methods would either require structural changes to the ADS-B protocol (which requires us to modify the underlying protocol) or depend on specific features extracted from the ADS-B data for identification of the spoofed signals [41]. As a result, these methods are of limited use when it comes to detecting a wide range of attacks in ADS-B data and filtering out real from spoofed data. Although there has been recent work on automatic end-to-end transmitter signal recognition for identifying unique transmitter signals from an aircraft, this is particularly achieved using deep learning frameworks to identify and learn features from an ADS-B signal. Radio-Frequency (RF) transmitters have inherent abnormalities that leave their fingerprint on the transmitted signals. We extract and use these unique fingerprints as our basis for unique ADS-B signal classification.

Moreover, today's malicious attackers have got even smarter, with the use of autonomous systems that make use of machine learning algorithms to learn the underlying ADS-B data and make use of generative models to create spoofed data that follows the distribution of real data can be a threat to our ADS-B system which can lead to adversarial attacks.

To this end, we propose a novel deep learning framework to classify attacks and identify an anomaly in ADS-B data to cover a wide range of attacks. The proposed Intrusion Detection is a 3-stage analysis of ADS-B message sequences. The first block performs attack classification using a Graph convolution Network to identify hardware-based attacks (such as replay, rebroadcasting, etc.). Suppose the ADS-B data is normally in the next stage. In that case, we do anomaly detection, which checks for software-based attacks like Jamming, modification, etc. Here, we use a Generative time series model called Wavenet to analyze the history of message sequences to predict and observe anomalous features. In the final block, we have the aircraft classification model to use the raw ADS-B signal to identify unique aircraft in airspace, which helps detect impersonation attacks.

Attack classification introduces a novel deep graph convolutional network, representing the entire airspace as a graph. This is a relatively new approach toward message classification compared to previous work on ADS-B data using deep learning [5]. We use graph

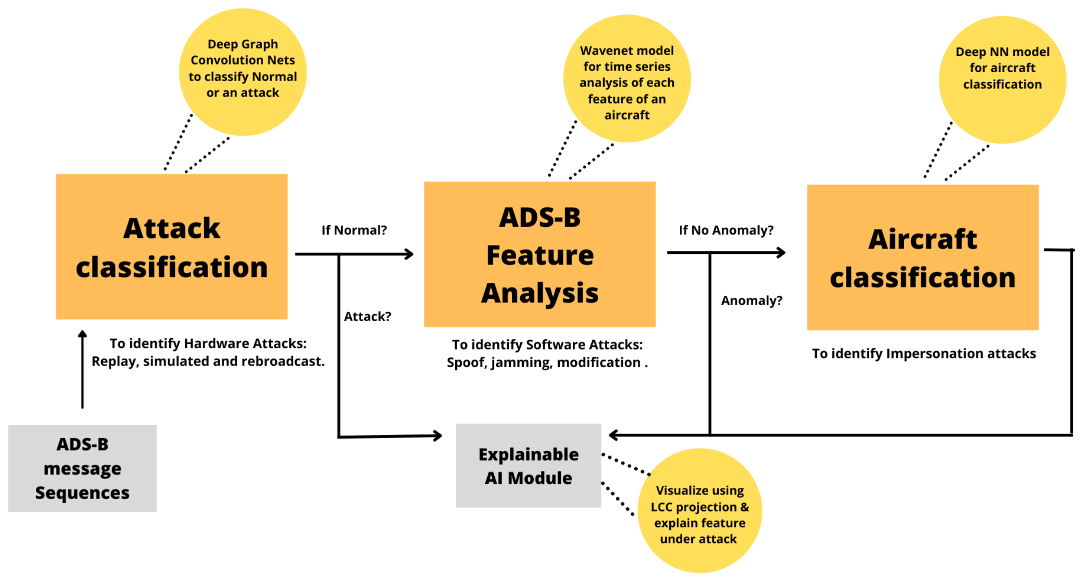

Deep Graph Convolution Nets to classify Normal or an attack
Wavenet model for time series analysis of each feature of an aircraft
Deep NN model for aircraft classification
Attack classification
If Normal?
ADS-B Feature Analysis
If No Anomaly?
Aircraft classification
Attack?
Anomaly?
To identify Hardware Attacks: Replay, simulated and rebroadcast.
To identify Software Attacks: Spoof, jamming, modification .
To identify Impersonation attacks
ADS-B message Sequences
Explainable AI Module
Visualize using LCC projection & explain feature under attack

Figure 2.5: Intrusion Detection System

representation learning and analyze multiple aircraft at each instance in time in a given radar (usually 370 km) rather than looking at only a single ADS-B message from each aircraft. The advantage is that neural graph networks help the model train considerably faster and give us a holistic view of the airspace to identify smaller granularity during attacks that are learned concerning other aircraft operating in the nearby region.

In the following section, we introduce the relevant ideas that are used in this research. More precisely, we introduce the idea of CNN (Convolution Neural Networks), Self Attention mechanism, and GAN for adversarial learning and briefly touch upon federated learning which we will use in our IDS system implementation.

Convolutional Neural Network (CNN)

In our model, 1D CNN[24] plays the role of a neural network that has one or more convolution layers and is mainly used for real-time detection and classification of inconsistent aircraft behavior, which automatically detects the important features in the auto-correlated data field. A convolution is basically a simple application of a filter to an input that results in activation of intrinsic input features. Repeated application of the same filter over an input results in a map of activation called a feature map, which indicates the locations and strength of a detected feature in an input.

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k] \tag{2.1}$$

where x[n] is our n input features, h[n] is the convolution filter, and y[n] is the output. * denotes convolution. Notice that we multiply the terms of x[k] by the terms of a time-shifted h[n] and add them up.

The main advantage of using CNN over its precursors is that it helps us automatically detect important features without any human supervision. Or in other words, it eliminates the need for manual feature engineering. Moreover, CNN, in general, is better than just using a feed-forward network since CNN has an essential parameter sharing property and performs dimensionality reduction. Because of parameter sharing in CNN, the number of parameters is considerably reduced. Thus it is computationally efficient and detects hidden patterns or sequences in intrusion attacks extremely well. The input to the CNN is the ADS-B message field.

Self Attention mechanism

The Attention mechanism in Deep Learning is based on the concept of directing your focus such that the model gives greater attention to certain factors when processing the input data. We use an attention mechanism within the input features of our message sequence, also called the Self-Attention mechanism inspired from the paper on attention is all you need by Vsnani[38]. Equation 2.2 shows a scaled Dot-Product Attention.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

The first step in calculating self-attention is to create three vectors from the output given by our 1D CNN layers. We create a Query, Key, and Value vectors. We represent Query and Key as the output from the first convolution layer and Value from the second convolution layer in the network.

The second step is to calculate a score. The score is calculated by taking the dot product of the Query vector with the Key vector of the respective feature we're scoring. So if we're processing the self-attention for the feature in position 1, the first score would be the dot product of Query 1 and Key 1[]. The second score would be the dot product of Query 1 and Key 2, and so on. The score determines how much focus on placed on other parts of the input sequence as we encode a feature at a certain position. We can use a matrix of alignment scores to show the correlation between source and target features, i.e. (query and key), as shown in Figure 1.1. Once we have the alignment scores, the output of this result passes through a softmax function. Finally, we perform matrix multiplication with the Values, which concludes our self-attention calculation.

Generative Adversarial Network

Generative adversarial networks, or GANs for short, is an approach toward generative modeling using deep learning. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The model is inspired by game theory. The two models are trained together in a zero-sum game, adversarial (generator model), until the discriminator model is fooled into believing that the generated examples are real, meaning that the generator model is generating plausible examples. GANs are an exciting and rapidly evolving area of interest, delivering on the promise of generative models in their ability to generate realistic examples across a wide range of problem domains. But, GANs, in general, are tricky to train since they are notorious for their instability during training, which leads to problems such as mode collapse. Although such problems are mitigated using advanced GAN techniques such as WGAN [2]. The goal is to find the right balance between generator and discriminator during training such that the generator can produce a variety of new data points, whereas the discriminator is successfully able to classify at least more than 50% data correctly.

2.5 Why use Federated learning?

Federated learning is one of the most widely deployed techniques in the context of private deep learning introduced by Brenda McMann et al., and Google is the pioneer in this field

[25]. In simple terms, federated learning is a technique for training ML models on data you do not have direct access to. So, instead of training a model on a server where the data resides in FL, we send the model to the client for training and fetch the trained parameters to the server. The whole idea of federated learning is that you as a client do not require to upload data to the server. Instead, a model is sent to your device locally for training the ML model on your generated data; at the end, the server receives an aggregation of all the learned parameters from one if not thousands of client devices that take part in training a large model.

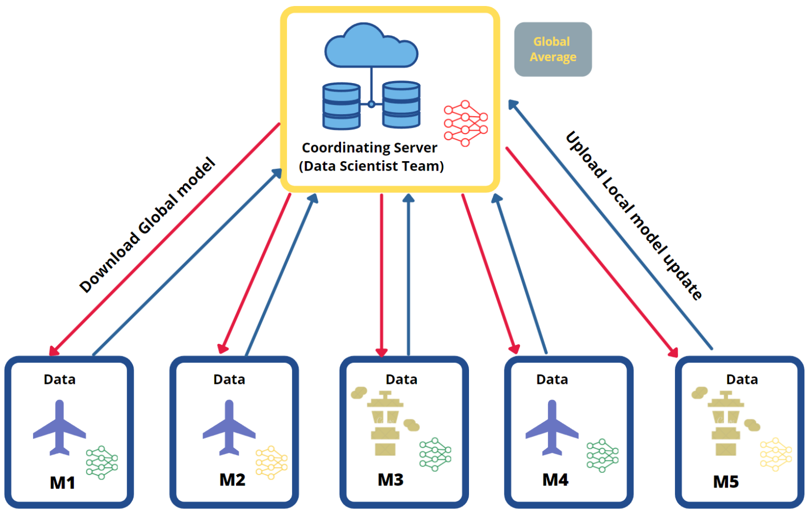


Figure 2.6: An illustration of the federated learning in distributed network of clients

This form of collaborative training, in turn, makes the final model more intelligent and smarter. FL can be thought of as a distributed AI that is widely used today in many applications such as phone text message prediction, browser search, wearable medical devices, predictive maintenance in automobiles, etc.

In the context of air-traffic management, we deploy a federated model to the client, i.e., the air-traffic controller. Thus, the ML model is trained locally on every Air traffic management (ATM) device, and every time it learns from its data, a new, smarter version of the model is updated to the cloud for inference to be used by other parties. Imagine the benefits of this technique. It can enable multiple organizations across borders and countries to collaborate and take part in training the model without compromising data privacy. Everyone could equally benefit from the process. Another major advantage is that FL can help significantly reduce the bandwidth cost of uploading datasets to the cloud. Thus in this project, we emphasize using this technique and training our model in a federated fashion.

3 Related Work and Limitations

In recent years, scientists have carried out related research work on the security issues of the ADS-B system and have proposed several security measures and solutions. It is also important to mention that there has been a significant development, and many possibilities have been explored with this protocol by using cryptography techniques and building machine learner systems as an IDS(Intrusion Detection System) on which we try to build upon using the following research methods as our baseline work.

3.1 Encryption and Authentication

Strohmeier et al. [15] asked the question of whether the current implementation of ADS-B can be encrypted, and Finke et al. [6] introduced several encryption schemes for ADS-B. In comparison, Costin et al. [5] and Feng et al. [8] suggested public key infrastructure solutions based on transmitting signatures and elliptic-curve cryptography. It is important to note that some of these techniques require modifications to the current structure of the protocol by adding a new message type for key publishing. While since the authentication information is included in the Mode S squitters, this framework does not require a resource-demanding public-key cryptography solution. The disadvantages of these methods are reflected in their large overhead size caused by the retroactive key size, thus requiring more resources from the communication channel.

3.2 Challenges of nextgen: ADS-B

In the paper Realities and challenges of next-gen air traffic management, [36] specifically discusses ADS-B as a new technology for air traffic monitoring. It points out how next-gen systems are crucial in successfully handling air-traffic growth and improving the safety of worldwide passengers. It also addresses issues with the current state of ADS-B as it is being rolled out and also important security challenges faced by ADS-B. Overall, this study intends to help identify open research questions, thereby gathering new research interests and developments in this field.

3.3 Works Related ADS-B Using Machine Learning

The paper on Detecting ADS-B Spoofing Attacks using Deep Neural Networks [42] presents a Spoofing Detector for ADS-B protocol, a two-stage Deep Neural Network (DNN)-based spoofing detector for ADS-B that consists of a message classifier and an aircraft classifier (SODA). Experimental results in the paper show that SODA detects ground-based spoofing attacks with a precision of 99.34%, with a false positive rate of 0.43%. SODA outperforms other prominent machine learning techniques, such as XGBoost, Logistic Regression, and SVM. Although SODA is a very promising machine learning system, it fails to address the feasibility of processing U.S. ADS-B messages in near real-time.

3.4 ADS-B Spoof Attack detection based on LSTM

The article proposes an ADS-B spoofing attack detection method based on an LSTM (Long short term memory) network [40]. Experiment results indicate that this method can effectively detect 10 different kinds of simulated manipulated ADS-B messages (such as) without further increasing the complexity of airborne applications. Moreover, they state that it can also identify the specific features under attack. This method does not require any manual feature engineering and/or modification of the existing protocol. It has strong operability in future practical applications in aviation systems.

3.5 Near-Real-Time IDS for the U.S NextGen ADS-B

Deep learning adoption is increasing in the cybersecurity domain and is being leveraged in adoption toward building Intrusion detection systems for open protocols such as ADS-B. However, studies have not focused on feasibility in the context of aviation and the required computing resources, which the paper[30] assesses. In addition to providing a solution for attack mitigation by packaging a machine learner into an intrusion detection system, none of the previous works look at the feasibility of processing ADS-B messages in near realtime, which this study addresses and lays the groundwork for other researchers in the field to work upon.

3.6 Detecting ADS-B Spoofing Attacks using Deep Neural Networks

It performs IQ signal as message classification using DNN to classify normal or anomaly. Further, it uses phase signal as a feature of the DNN model to identify aircraft for spoof attacks. It uses a straightforward approach of message classification into normal or anomaly attacks. Since the ADS-B messages are time distributed and change over time, detecting an attack with just single message input is insufficient to get a clear picture of the attack, in real-world data is dynamic..

3.7 Analyzing Sequences of Airspace States to Detect Anomalous Traffic Conditions

This method uses the LSTM Encoder-Decoder network to reconstruct a fixed size input data consisting of transformed ADS-B messages. The output is measured as reconstruction error using cosine similarity. If the reconstruction error is high, it is identified as an anomaly. The approach to detect anomaly is not a time series classification, but it maps input data to output and builds a model to reconstruct the given input. It is based on the idea that if there is any new data with the attack, the model will fail to reconstruct the input, thus giving a high error. But this work only limits the data it is trained on. If a new input from different regions with normal ADS-B message data is fed, it can still result in a high reconstruction error since it had not been seen before.

4 Graph Theory

4.1 What is a Graph?

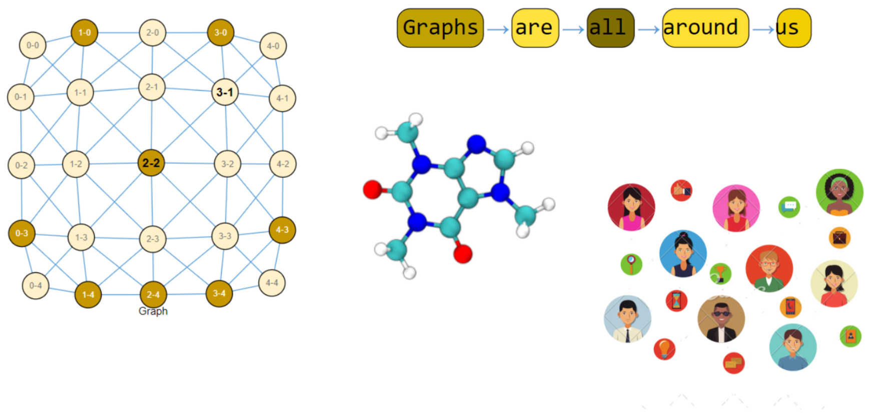


Figure 4.1: Graph networks

Graphs are a well-known data structure and a universal language for describing complex systems. In the most general view, a graph is simply a collection of objects (i.e., nodes) and a set of interactions (i.e., edges) between pairs of these objects. For example, to encode a social network as a graph, we might use nodes to represent individuals and use edges to represent that two individuals are friends. In simple terms, A graph represents the relations (edges) between a collection of entities (nodes).

4.2 Graph representation learning

Graph structured data is all around us, from natural and social sciences, from telecommunication networks to quantum chemistry. Graph representation learning is powerful because it builds a strong relational inductive bias into deep learning architectures, which is crucial if we want systems that can learn, reason, and generalize on such data types. This generally means it imposes constraints on relationships and interactions among entities in a learning process [34]. Recent developments in graph representation learning have increased their capabilities and expressive power, showing practical applications in drug discovery, 3D vision, Traffic forecasting, question-answering, and recommendation systems.

4.3 Types of Graph structure data

Now since we have briefly explored the graph-structured data, let's ask ourselves what task we can perform using this graph data? Using Graph representation, we can generally do 3 types of predictions, i.e., graph level, node level, and edge level. For the graph level, we make predictions using the entire graph space. At the node level, we predict the certain property of each node or a new node. And for edge level, we infer or predict the presence of edges in a graph.

4.4 GNN

GNNs are Neural networks that have been adapted to leverage the structure and properties of graphs. It is an optimizable transformation on all attributes of the graph (nodes, edges, entire graph) that preserves symmetries which means it is permutation invariant. The following sub-sections explore various components needed to build a graph neural network. GNNs follow a graph in - graph out architecture. The model accepts the graph as input, transforms the node, edge, or global-context features into an embedding, and outputs a new graph with new transformed features or learned embedding without changing the graph's connectivity. The simplest GNN is built using the message passing neural network framework proposed by Gilmer et al., which can be approximated as graph convolution and can be seen as an extension of convolution to graph data.

4.5 Graph Convolution

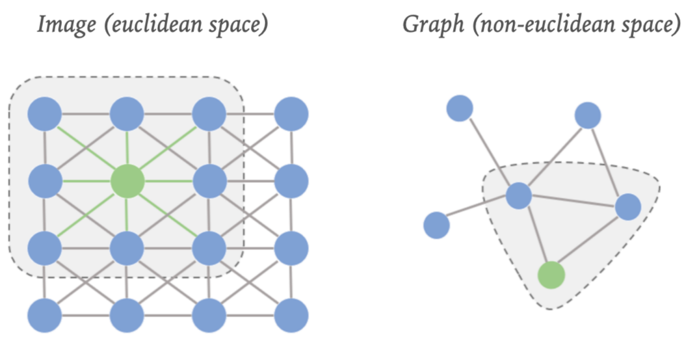


Figure 4.2: Euclidean vs Non-Euclidean

To understand graph convolution, let's look at the difference between convolution operations performed on images v/s graph data. In images, a fixed kernel is applied over a region of the input image to extract the features. Since the data points are euclidean in images, this operation is straightforward. Whereas graph data is non-euclidean, and we just cannot apply a fixed-sized kernel over the graph nodes, we consider each node and aggregate the node features from its neighbor nodes connected to it by an edge. The aggregation is usually a sum or average operation and finally updates this node feature with new compressed information. This step is called a message passing layer, where the information from neighboring nodes is passed into the current node. Thus, all graph nodes are passed through several message-passing layers. These layers help construct node embeddings that contain knowledge about other nodes and edges in a compressed embedding. The size of the embedding is a hyper-parameter to our model. To better understand, let's assume the following input graph at time step t.

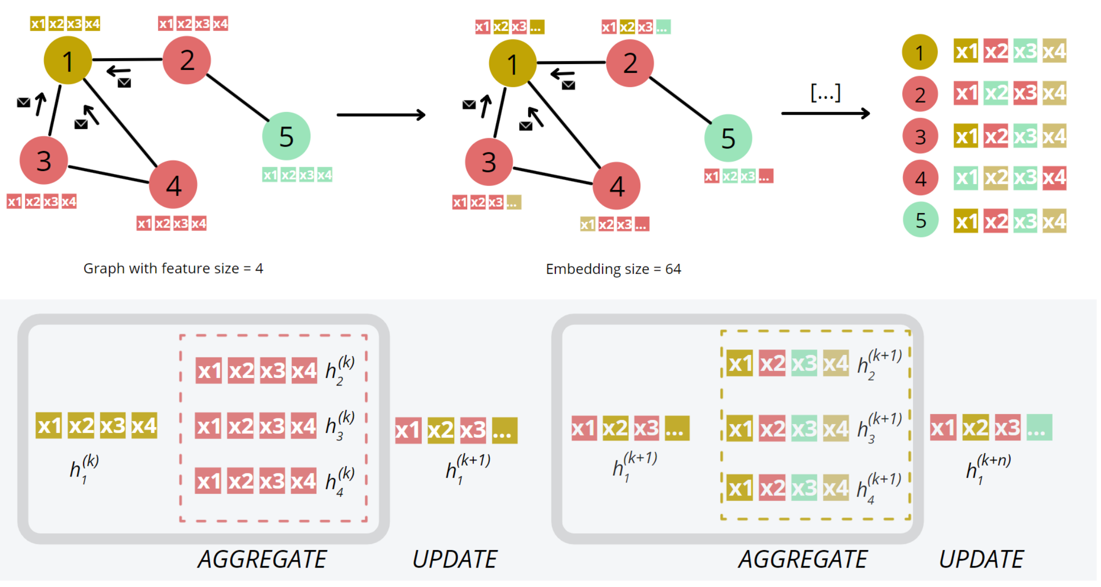


Figure 4.3: Graph Convolution operation

They are iteratively combining node information in a local neighborhood. This local feature aggregation can be compared to learnable CNN kernels. The more message-passing layers we add, the more hops we perform on a graph. The number of layers depends on the task at hand. For example, if it is a small molecule, we can quickly learn about its structure using a few layers. Furthermore, stacking too many messages passing layers in GNN can lead to over-smoothing and require additional techniques to handle these issues. Finally, Each node embedding contains the feature base and structural information of the nodes, which we can eventually use to make predictions. The following equations show message passing in GNN.

$$h_u^{k+1} = UPDATE(h_u^k, AGGREGATE(h_v^k, \forall v \in N(u))) \tag{4.1}$$

To achieve this neural message passing, use a neural network for update and aggregate functions which can be defined as:

GNN variants: GCN by welling 2016, adding a self-loop combines update and aggregate function into one and uses a learnable weight matrix.

$$h_u^{k+1} = \sigma(W_{self}^{(k+1)} h_u^k + \sum_{v \in N(u)} W_{neigh}^{(k+1)} h_v^k) \tag{4.2}$$

The message passing layer can be further simplified to more efficient and generalizable form by inducing sharing parameters, i.e. we collapse W_{self} and W_{neigh} into single W weight matrix by adding self loops to the adjacency matrix A as follows:

$$H^{k+1} = \sigma((A+I)H^k W^{(k+1)}) \tag{4.3}$$

However, in order for our GNN to be seen from a convolutional perspective, the GNN equation is slightly modified such that its adjacency matrix $\hat{A}$ is calculated to normalize the number of nodes in the neighborhood. Here is the final equation we make use of in our implementation originally derived from the paper on GCN by wellings[23].

$$\hat{A} = (D+I)^{(-\frac{1}{2})}(I+A)(D+I)^{(-\frac{1}{2})} \tag{4.4}$$

$$H^{k+1} = \sigma((\hat{A}+I)H^k W^{(k+1)}) \tag{4.5}$$

4.6 Graph Attention Networks (GAT)

As seen previously in GCN, we use the node features (X) along with a learn-able weight matrix (W) and also the adjacency matrix (A) to represent the structure of a graph to obtain new node embedding (H′) as indicated in the equation.

$$h_i' = \sigma(A * \sum_{j \in N(i)} W * h_j) \tag{4.6}$$

Now, we extend this mechanism to leverage self-attention in GNN [39]. The idea behind self-attention in the graph is to learn how important, for example, node[i] features are to node[j]. This is called as attention coefficient, where it weighs the edges of a graph between two nodes.

$$e_{ij} = a(Wh_i, Wh_j) \tag{4.7}$$

To compute, we use node features of i and j. We apply learnable transformation using weight matrix W. This can be seen as h_i' and h_j', which passes through an attention mechanism that returns the attention coefficient for both the nodes as e_{ij}. Thus, we compute the attention score for all the edges in our graph and normalize them using a softmax function.

$$a_{ij} = softmax(e_{ij}) \tag{4.8}$$

Next, let us look at how we compute the attention mechanism (a) used in equation 3.7 above. We use a shared single-layer neural network with an attention weight matrix W_a to calculate. Our inputs to this network are the hidden features of the 2 nodes i.e. h_i' and h_j' concatenated together before passing them to this single-layer neural network. The output of which passes through a LeakyReLU and, finally, the softmax function at the end to normalize the attention scores. In this case, we use Leaky ReLU to compensate for the negative values and focus more on positive values (to emphasize positive relations between nodes) during gradient computation instead of using just simple ReLU for better network learning.

$$a_{ij} = softmax(e_{ij}) \tag{4.9}$$

So in order to incorporate this attention mechanism in GNN, we multiply each of the neighbor nodes with the corresponding attention coefficient for that edge. This can be seen as a linear combination of node feature vectors weighted with the importance of each node. As a result, the important features are amplified, and the less important ones are suppressed.

$$h_i' = \sigma(A * \sum_{j \in N(i)} a_{ij} * W * h_j) \tag{4.10}$$

Fig 4.4 illustrates Step by step block diagram of how node embeddings are calculated with attention weights.

4.7 Transformer Architecture (Graph Transformer Convolution)

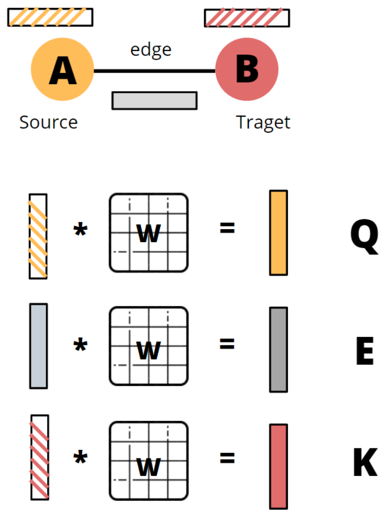


Figure 4.4: Attention weight between two nodes in graph

GNN Transformer layer [6]: To incorporate the transformer layer into GNN, we first start with the standard node and edge features of a graph. The attention weight is calculated using the formula:

$$a^{(l)}_{c,i,j} = \frac{Q^{(l)}_{c,i}, K^{(l)}_{c,j} + E^{(l)}_{c,i,j}}{\sum_{m \in N(i)} Q^{(l)}_{c,i}, K^{(l)}_{c,m} + E^{(l)}_{c,i,m}} \tag{4.11}$$

The attention weight tells us how much attention node i should pay to node j. So here, the Q queries are transformed source node(s) i, K Keys are transformed target nodes(s) j. The edge features are added to the target feature, and a dot product is performed between Q and K vectors and further normalized by the size of attention head d. Finally, to perform the final aggregation, we need to calculate the node embedding, which is done by taking the original node embedding and multiplying it with a learnable weight matrix to obtain a new node embedding at layer (l+1). This form a one transformer layer block in GNN consisting of c multi-head attention and L indicating multi-hop layers in GNN.

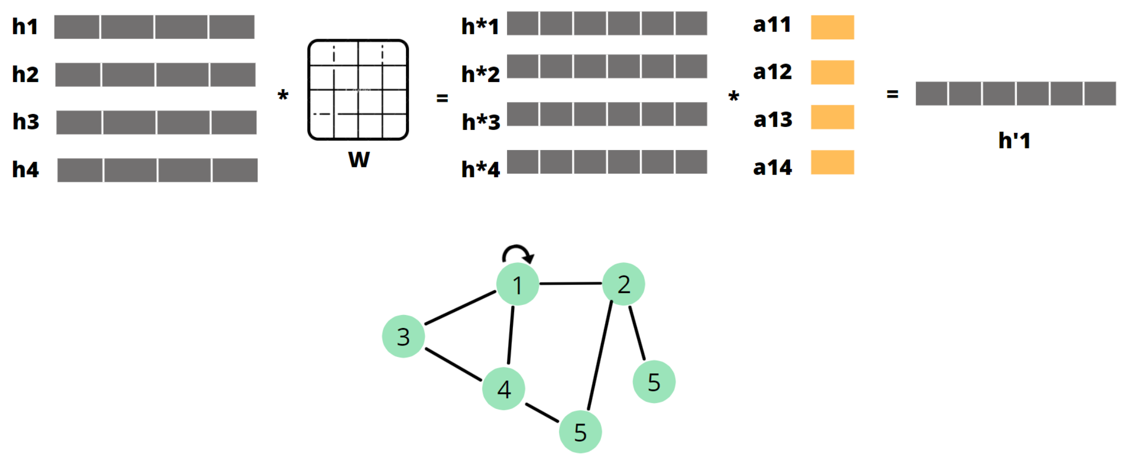


Figure 4.5: Summary of steps in a Transformer Conv block

4.8 Cross Entropy

$$H_p(q) = \sum_x q(x) \log_2 \left(\frac{1}{p(x)} \right) \tag{4.12}$$

$$KL = H_p(q) - H(q) \tag{4.13}$$

The loss function incorporated for ADS-B message classifier training is a classic cross-entropy loss function. Here we compare the two probability distribution of the prediction and ground truth over the vocabulary of words. We simply subtract one from the other. This is done using the Kullback–Leibler (KL) divergence. KL Divergence helps us measure just how much information we lose when choosing an approximation. The equation (3) defines the cross-entropy, which states average length to compute $q(x)$ using distribution of $p(x)$. And equation (4) computes the difference as KL divergence, where $H_p(q)$ is cross-entropy and $H(q)$ is the entropy function.

5 Methods

5.1 Building a Graph Convolutional Network

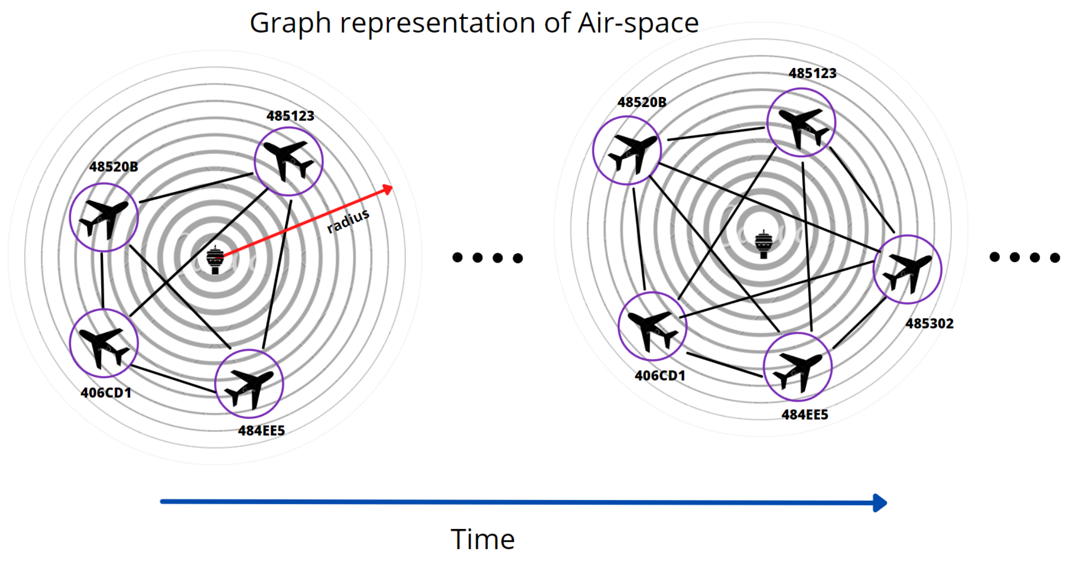


Figure 5.1: Airspace as Graph representation

Transforming ADS-B data into Graph representation

Before transforming data into graphs, first, let's take a look at the challenges of using graphs in machine learning. We know in a typical deep learning model, the inputs are given as a matrix, e.g., NxD (N is the number of examples, D is the dimension or the total number of features in the dataset) or images as a grid of values, and text as the vectorized matrix. But encoding graph data gets a little more complicated, and a graph consists of nodes, edges, global context, and connectivity. Representing the nodes, for example, can be straightforward, i.e., we can create a node feature matrix of N nodes x F features a size of a 2D matrix. But representing the connectivity is more complicated. Perhaps the most obvious choice would be to use an adjacency matrix to encode all the graph connections, but as the graph scales in size, it becomes memory inefficient for large graphs having millions of nodes leads to a sparse adjacency matrix. Another major drawback is that there may be many adjacency matrices for a particular graph encoding the same connectivity. One example of this is shown in figure 5, And thus there is no guarantee that these different matrices would give us the same output in a deep learning model. To counter this, one elegant and memory-space efficient solution is to use an adjacency list that stores a tuple (i,j) such that i, j represent the nodes that are connected using the edge k. Here the number of edges is much lower than entries in an adjacency matrix, thereby avoiding computation and storage over disconnected parts of the graph.

Now, in order to transform the ADS-B message data into a graph representation, we first convert each unique aircraft having an ICAO address into a graph node or vertex figure 5.1. In our implementation, we choose 5 different aircraft as our graph nodes. The number of aircraft(nodes) chosen is based on the range of the radar receiving ADS-B message signals which are typically 370 km.

Part I: Building Wavenet model - ADS-B data forecasting

Wavenet is deep autoregressive, generative model originally designed by Google Deepmind[31] for raw audio generation. The model can be abstracted beyond audio to apply to any time series forecasting problem, providing a nice structure for capturing long-term dependencies without any excessive number of learned weights.

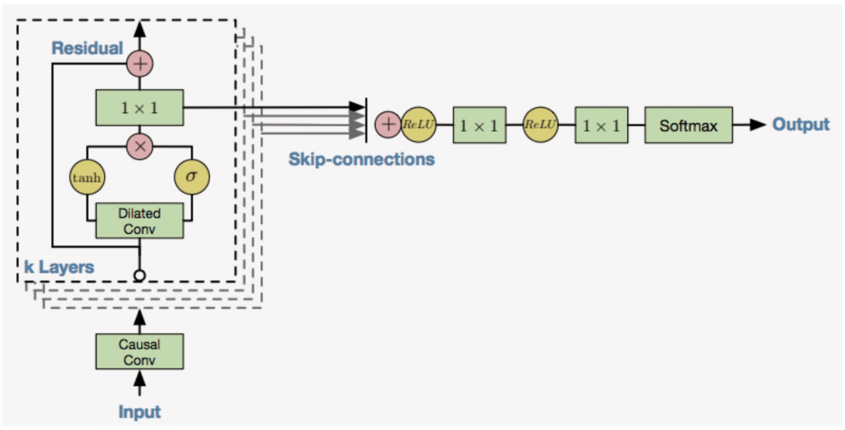


Figure 5.2: Wavenet architecture

WaveNet is basically a fully convolutional neural network. The convolutional layers have various dilation factors that allow their receptive field to grow exponentially with depth and cover thousands of timesteps. The core building block of the wavenet model is the dilated causal convolution layer. The following figure illustrates causal convolution compared to traditional convolution. In a traditional 1-D convolution layer, we slide a filter of weights across an input series, sequentially applying it to (usually overlapping) regions of the series. Consider a filter width of 2 and stride 1 over a sequence of inputs. Since the output clearly depends on the future states, we are using the future states to predict the past. Letting the future of a sequence influence our interpretation of its past makes sense in a context like text classification. We use a known sequence to predict an outcome, but it is not appropriate for time series since we use the previous known states to generate future values.

$$p(x) = \prod_{n=1}^{N} p(x_n | x_1, x_2, ..., x_{N-1}) \tag{5.1}$$

Dilated (Causal) Convolutions

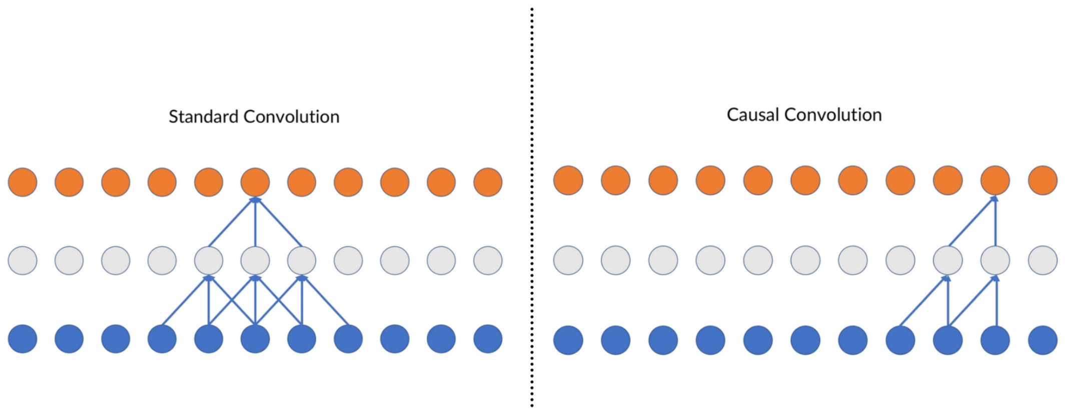


Figure 5.3: Standard v/s Causal

We get the proper tool to handle temporal data with the above causal convolution, but it requires further modifications to handle long-term dependencies. As seen in the above fig for simple causal convolution only 5 recent timestamp influence the output. We require one additional layer for every timestamp. With our time-series data extending over long-range time, using simple causal convolution to learn from entire history would make it way too computationally expensive.

To overcome this, WaveNet uses dilated convolutions, which allow the output receptive field to increase exponentially as a function of the number of convolutional layers. In each dilated convolution layer, filters are applied such that it skips a constant dilation rate in between each of the inputs they process, as indicated in the figure. By increasing the dilation rate multiplicatively at each layer (e.g., 1, 2, 4, 8, ...), we can achieve the exponential relationship between layer count and receptive field size that we desire. In the following diagram, you can see we now require only 4 layers to cover the 16 input sequences.

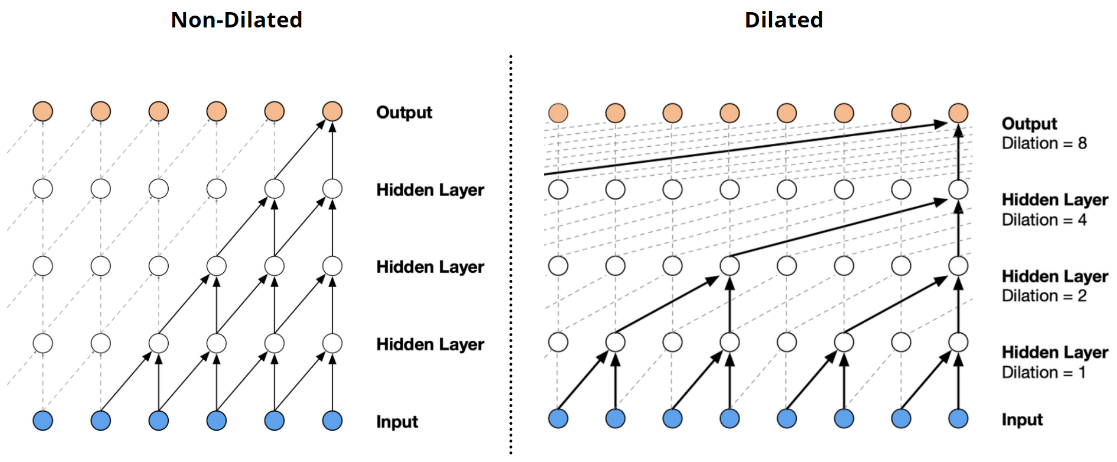


Figure 5.4: Dilated convolution

Use of Gated Activation Units

The purpose of using gated activation units is to model complex operations to induce non-linearity. The gated activation units are represented using the following equation:

$$Z = tanh(W_{f,k} * X).\sigma(W_{g,k} * X) \tag{5.2}$$

Where $*$ is a convolution operator, . is an element-wise multiplication operator, $\sigma(.)$ is the sigmoid activation function, k is the layer index and W_f , and W_g , are weight matrix of filters, and gate respectively.

Residual block and Skip Connections

The use of residual block and skip connections are inspired by pixel CNN architecture for images [32]. The residual block helps train the model faster since training error does not increase as we stack more layers because the residual network enables the model to learn from deeper connections. As seen in the diagram, both residual and parameterized skip connections are used throughout the network to speed up convergence and enable the training of much deeper models.

The only modification we make to the model is avoiding using the softmax function since our output is not a categorical distribution but rather a time series prediction.

Algorithm 1 Wavenet Model

```
1: procedure WAVENET($x, f$)
2:     for dr in dilation-rates do
3:         $y[n] \leftarrow \sum_{k=1}^{N} x[k] * h[n-k]$, apply $1DConv(filters = 16, kernel = 1)$
4:         for k=1 in 1 to 2 do
5:             $X[k] = 1DConv(filters = 32, kernel = 2, dilation = dr) * y$
6:         end for
7:         $Z = tanh(X[1]) * \sigma(X[2])$
8:         $out = 1DConv(16, 1) * Z$
9:     end for
10:    $out = 1DConv(128, 1) * Z$
11:    $out = 1DConv(1, 1) * Z$
12: end procedure
13: return out
```

Part II: Deep Graph Convolution Network architecture - ADS-B attack classification

In order to build an attack classification model on graph data, we exploit the Graph Convolutional Network (GCN) architecture. First, we need to transform the raw ADS-B message data from multiple aircraft in airspace for each instance in time into graph data. To achieve this, we use two sets of data, one containing normal ADS-B messages from various aircraft and the other dataset containing simulated attacks on the normal aircraft data. The simulated attacks are not manually performed. Instead, we use a Generative Adversarial network model to create new fake ADS-B data for use that closely resembles the true data. Using adversarial learning for fake data generation makes the underlying classification model more robust in identifying minor tinkering in training data, thus making it more powerful in identifying unknown attacks.

Once we have normal and attack data ready, we convert the data for all aircraft having ICAO addresses into a graph. The nodes in a graph represent the aircraft, and the edges between them are the distance. A new graph is created every time a new set of ADS-B messages are received from multiple aircraft in the airspace. If all the ADS-B messages in a graph are authentic, it is classified as normal data. Whereas if the graph data contains one or more fake ADS-B messages, it is classified as an attack in the airspace.

Our GCN architecture is a 3 layer deep convolution layer followed by a global mean pooling. The convolution layers perform message passing between the nodes; a 3-layer GCN means every node in the network receives intermixed message information from three deep neighbor nodes connected to the initial node. The hidden layer size chosen is 64 dimensions, and our training data is 2000 samples; 1000 are real authentic messages, and others are fake simulated attack messages.

Algorithm 2 GCN on ADS-B data

```
 Input X ← Decoded ADS-B message/IQ Samples
 Output O ← Normal or Attack

Data Pre-processing, $F : f1, ...f6$ or $F : f1, ...f480$

Sort data based on ICAO and timestamp, $X \leftarrow SORT(X)$

Apply Min-Max Scalar, $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$

Build data-list, $Data(node-features = [a, F], edge-index = [2, 20], label = [1])$

procedure GCN(
    )
    $H^{k+1} = \sigma((A + I)H^k W^{(k+1)})$ , (conv1): GCNConv(6, 128)

    $H^{k+2} = \sigma((A + I)H^{k+1} W^{(k+2)})$ , (conv2): GCNConv(128, 128)

    $final = Softmax(H^{k+2} W^{(k+3)})$, (lin): Linear(in-features=128, out-features=2)

    return $final$
end procedure

model, $M = GCN()$

for epoch in E do
    for each data-item in data-list do
        $pred \leftarrow M(data - item.x, data - item.edge)$ , forward pass
        $loss(pred, data - item.y)$ , calculate cross-entropy
        $M.update(parameters)$ , Update model weights
    end for
end for
return Output classification, O
```

Part III: Simulated Attacks - Data Generation using GAN's

To closely follow the distribution of normal authentic ADS-B messages, we make use of Generative Adversarial network[14] to learn the normal data distribution and generate fake data or attacks. The attack data is generated for every unique ICAO address of the aircraft from the original pool of datasets. GANs consist of two neural networks the generator, which generates fake ADS-B messages from input noise, and the discriminator, which learns to classify if the generated data is fake or real. The entire network is trained in an adversarial fashion; once the generator can fool the discriminator that the data generated is real and not fake, we can then use the trained GAN for new data generation.

There are various types of GAN architectures, in our implementation we make use of the Vanilla GAN model this is the simplest type GAN. Here, the Generator and the Discriminator are simple multi-layer perceptrons. In vanilla GAN, the algorithm is really simple, it tries to optimize the mathematical equation using stochastic gradient descent.

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{5.3}$$

Training of GAN is a 2 step process: Our first step in training the GAN is we train our Discriminator Network as a binary classifier to learn the distribution of existing data points. Basically, we train the discriminator to classify if the input data points are real or spoofed. Once the training step of the discriminator is complete, we switch to Generator Network and generate new data from random noise. The newly generated data is again fed into the discriminator network to classify whether it is real or spoofed. If spoofed, which means the loss is high, the errors are then backpropagated through the network, this time is training the Generator. The two networks are locked in a two-player minimax game defined by the value function V(G, D) in (4), where D(x) is the probability that x comes from the real data rather than the generated data. This completes one training cycle of GAN.

Algorithm 3 Training a Generative Adversarial Network

```
for epoch in (1, 5000) do
    for batch in batched do
        sample batch of m noise $z(1), ..., z(m)$ from noise prior $p_g(z)$

        sample batch of m real examples $x(1), ..., x(m)$ from data dist $p_{data}(x)$

        Update discriminator, $\bigtriangledown_{\theta d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)})] + [\log(1 - D(G(z^{(i)})))]$

    end for
    Update Generator, $\bigtriangledown_{\theta d} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z^{(i)})))]$

end for
return trained GAN network to generate fake ADS-B data
```

5.2 Part IV - Aircraft classification: using DNN

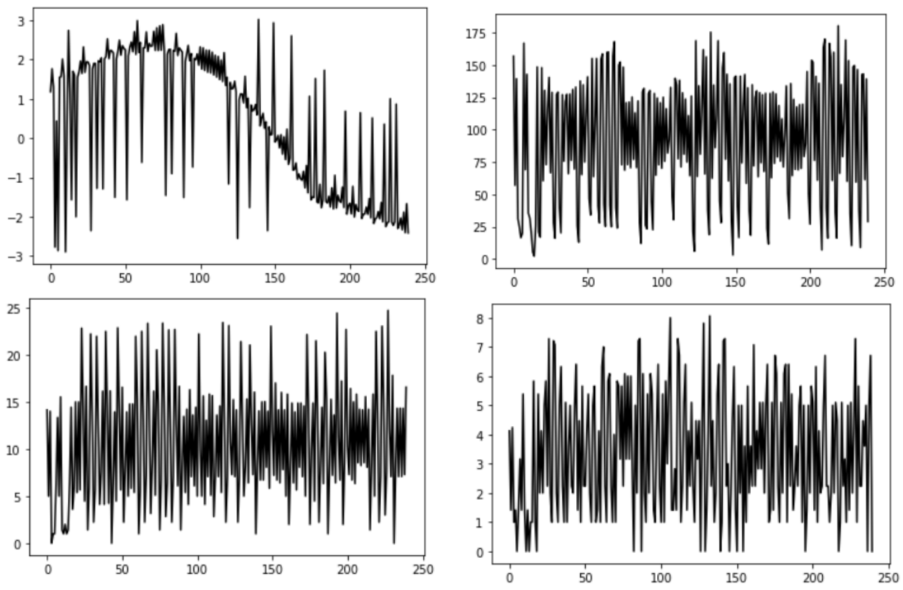


Figure 5.5: Phases of four different aircrafts

Unlike ADS-B message classification, in the aircraft classification module, we cannot make use of raw ADS-B data or IQ samples; instead, we need to rely on phases of the signal that are independent for each of the claimed ICAO addresses of (aircraft) as features to avoid being deceived by impersonation attacks. On the contrary, due to hardware impairments in radio devices, we notice IQ imbalance patterns, thereby providing us a method for identifying airplanes using the onboard ADS-B transmitter's unique signatures. Here, we refer to "IQ imbalance" to denote the imbalance between the in-phase (I) and quadrature (Q) components of the transmitted signal. Now to calculate the phase of k_{th} pair of IQ samples, we use the following formula as described in the paper [41].

$$\phi[k] = tan^{-1}(\frac{x_q[k]}{x_i[k]}) \tag{5.4}$$

From the subject of communication theory, we have learned that phases tend to encode both TX and RX carrier frequency offsets and Doppler shifts. To understand more clearly, consider frequency offset $\triangle f$ and phase offset $\triangle\phi$ in ADS-B pass-band signal in equation 5.5, where phase $\phi(t) = 2\pi\triangle f + \triangle\phi$ As a result of which the phases encode some reach information of the aircraft it is receiving a signal from, which we can use as a feature in our aircraft classification step.

$$x_p(t) \simeq x(t)e^{j2\pi ft} \simeq x(t)e^{j2\pi(f+\triangle f)t\triangle\phi} \tag{5.5}$$

$$and, x(t) = x_i(t) + jx_q(t) \tag{5.6}$$

5.3 Part V - Training model using Federated Learning

Federated learning is a relatively new type of learning that avoids centralized data collection and model training. In a traditional machine learning pipeline, data is collected from different sources (e.g., sensors, mobile, IoT devices) and stored in a central location (i.e., data center or server). Once all data is available at the server, a single machine learning model is trained by using this data. Because the data must be moved from the user devices to a centralized server for building and training the model, this approach is called centralized learning.

On the other hand, federated learning is about training multiple machine learning models on client devices and then combining the results of all trained models into a single model that resides on a server. Thus, a model is trained on devices themselves using ground-truth data, and just the trained model is shared with a server. The user's data is leveraged to build machine/deep learning models while keeping data private.

How does federated training help in securing ADS-B protocol ?

With the use of federated learning, we can enable a smarter and integrated Intrusion detection system, which means it will give us the ability to leverage data from a variety of organizations in a privacy-preserving manner. To understand it in more detail, consider aircraft traveling from one region to another. Now every region has its operating rules, and the ground station listens to the incoming ADS-B messages from every aircraft. We can then train a local model at that ground station and send the trained model to the main server or central region. Since collecting and sending huge amounts of data requires extra bandwidth costs and is also prone to potential adversarial attacks on data. Thus keeping the data private to its operating ground station and sending model parameters is a lot more efficient and feasible solution. At the central region server, the model parameters are aggregated from all its clients (local regions) and combined into one model using, for example, federated averaging, wherein we average over all the client models and later send this updated new model to all the local regions. Thus in the process, every operating region will have an updated global trained model at all times.

Steps for federated learning

Federated learning in theory is fairly simple and can be summarized in the following steps:

Algorithm 4 Federated Averaging[29] on DNN classifier

```
 **X ← ADS-B Data**
 **O ← Normal or Attack**
Data Pre-processing, $F : f1, ...f6$

Label Encode Categorical features, $X[C] \leftarrow L(C)$

Apply Standard Scalar, $X' = \frac{X-\mu}{\sigma}$

Convert train data to PyTorch tensor, X ← Tensor(X')

Split data among workers, A1 ← X[a1],..,A4 ← X[a4]

initialize $w_0$
model, $M = DNN()$

for epoch in E do
    $S_t$ ← (select clients from K = 1,2..)
    for each client k in $S_t$ do
        $k.to(M)$ , send model to client
        $w^k_{t+1} \leftarrow w - \eta * \nabla l(w; b)$ , update client weights
    end for
    Aggregate, $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} * w^k_{t+1}$
end for

$\sum_{k=1}^{K} w^k_{t+1} \leftarrow w_{t+1}$ Update local client with global parameters

Test the model accuracy(%) on each client, **return**
```

1. A generic (shared) model is trained server-side(global region).
2. Several clients(local ground stations) are selected for training on top of the generic model.
3. The selected clients download the model.
4. The generic model is trained on the devices, leveraging the users' private data, based on an optimization algorithm like the stochastic gradient descent.
5. A summary of the changes made to the model (i.e., weights of the trained neural network) is sent to the server.
6. The server aggregates the updates from all devices to improve the shared model. Update aggregation is done using a new algorithm called the federated averaging algorithm.
7. The process of sending the generic model to mobile devices and updating them according to the received summary of updates is repeated.

   The steps have been summarised in the article Introduction to Federated Learning by Ahmed Gad[12].

6 Results

6.1 Data-set Collection

In order to create a robust ADS-B attack detection system, we leverage two forms of ADS-B data 1) which operates on the Quadrature signals, also called I/Q data or IQ samples extracted from the receiver signal of an aircraft used during RF signal modulation. These signals contain intrinsic properties of the transmitters involved in the communication. These unique characteristics help us distinguish the receiver signal of one aircraft from others. The dataset has a collection of real ADS-B messages for a range of aircraft recorded using an ADS-B antenna by Xuhang Ying and the team at the University of Washington [41]. Furthermore, the dataset is augmented with hardware attacks such as rebroadcast, simulated, and relay attacks. The dataset was made available to us on request by the organization.

2) Secondly, we also use the decoded ADS-B message data, to extract the meaningful features from the ADS-B signal, such as latitude, longitude, altitude, speed, heading direction, etc. This message data is used to perform feature analysis using the deep learning method to detect any deviations. This dataset has been open-sourced on Github by Jing Wang, and team [40] and is readily available to use.

Table 6.1: Datasets

IQ Samples	Decoded ADS-B messages
which operates on the Quadrature signals, also called I/Q data extracted from the receiver signal of an aircraft in RF signal modulation	to extract the meaningful features from the ADS-B signal, such as latitude, longitude, altitude, speed, heading direction, etc.
30,000+ consiting of authentic, replay, ghost-aircraft attacks etc.	10,000 real data samples for each aircraft (Total 20 ICAO's)

To build and train our proposed deep learning solution, we use python as the core language to design the system and Google Colaboratory environment, which is a web Interactive Development Environment (IDE) for python. The colab is particularly well suited for building machine learning applications as it comes with pre-installed ML packages and libraries in the cloud that aid in the development process. In our application, we have used deep learning frameworks such as Pytorch[33] and TensorFlow[28] to build the GCN and Wavenet architecture, respectively, along with matrix processing library Numpy[17].

6.2 Hardware Requirements

To build and train our deep learning model we use the Google cloud co-laboratory environment which provides us with Nvidia GPU (Tesla T4) with approximately 15,000 MiB of memory usage for a total 6 hours of daily usage limit, which is sufficient to perform multiple training and testing of our model for each day. We kept the model architecture to be relatively simple and faster to train thus lowering the carbon foot-print of our model usage. For example, using Convolution blocks in our architecture, the training time is significantly reduced over its RNN and attention based counterparts. Since CNN relies on feature extraction mechanism the operation can be easily parallelized.

6.3 Attack Classification using GCN

In this experiment, we evaluate the performance of the GCN(Graph Convolution Network) model and compare it against two GNN variants: GAT(Graph Attention Nets) and Graph Convolution Transformer.

Graph data transformation using decoded real ADS-B messages

We first consider all the normal ADS-B messages of the respective aircrafts having ICAO addresses at the current timestamp and create a data list containing its node and edge features. The node features, in our case, refer to the latitude, longitude, speed, altitude, climb rate, and heading direction. In comparison, the edge feature is the distance between two aircraft. We iteratively add all aircraft node and edge features for every timestamp to the data list (our dataset consists of around 2000 ADS-B message timestamps). Next, we use the simulated attacks generated using the GAN model for each aircraft in our dataset and append them to our previous data list with node and edge features. Finally, we interpolate ADS-B data by inter-mixing authentic ADS-B messages with simulated attack messages and append them to our original data list. 80% of the items in the data list is randomly selected as our training data and 20% is kept for testing.

To initiate model training, we split all our transformed data into 80% and 20% train and test sets. The GCN parameters are summarized in Table II. We consider the following three GCN models:

1. M1: one multi-hop layer with 128 nodes,
2. M2: two multi-hop layers with 64 nodes, and
3. M3: three multi-hop layers with 64 nodes per layer.

Metrics

Since our attack classifier acts as a detector, we adopt the following two metrics:

1. Detection probability (denoted as Pd): the percentage of malicious messages classified as malicious, and

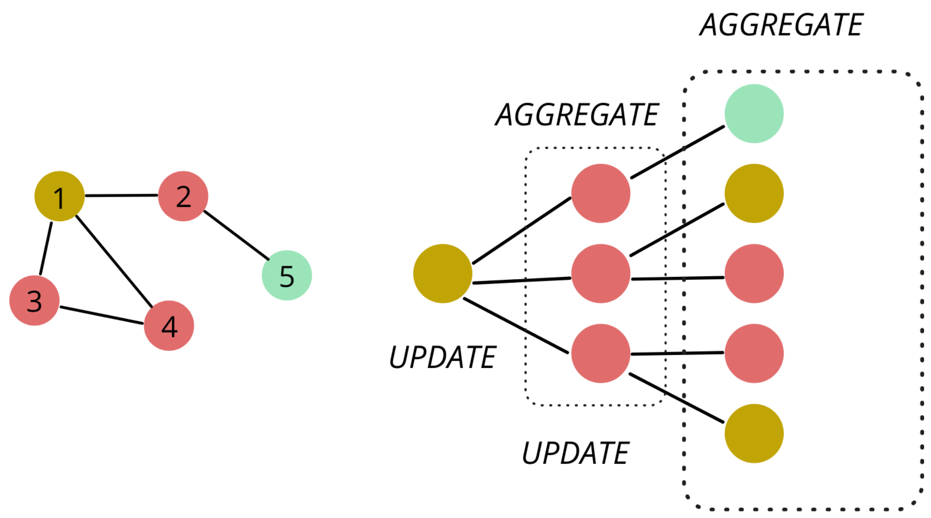


Figure 6.1: 2 Layer Multi-Hop in GNN

2. False alarm probability (denoted as Pfa): the percentage of authentic messages classified as malicious.

Table 6.2: Graph Convolution Network (Multi-hop layers)

	GCN 1-layer	GCN 2-layer	GCN 3-layer
Avg. Loss	0.0881	**0.0262**	0.0467
Accuracy(%)	98.93	**99.25**	97.47

Table 6.3: GCN-2 Confusion Matrix

Pred/Real	**Attack**	**Normal**
Attack	204	0
Normal	3	193

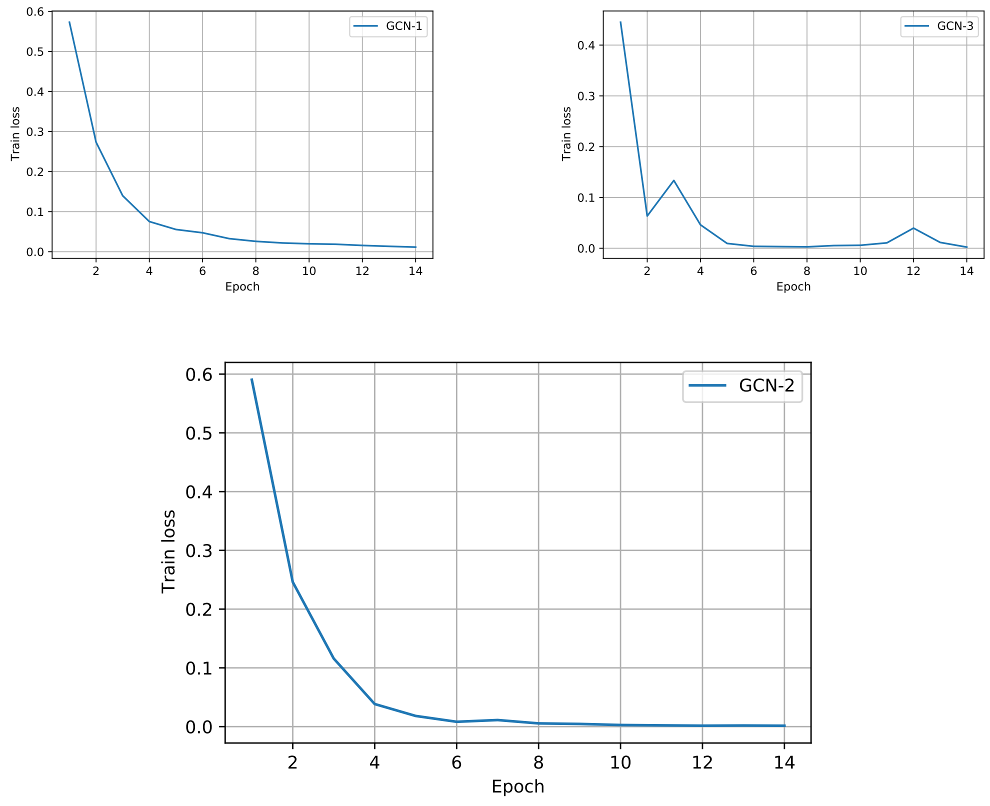


Figure 6.2: GCN 1,2 and 3 Model Train Loss

We first observe that a simple one-layer GCN model (M1) can already achieve excellent performance with an accuracy of 98.93% as compared to other GNN variants. Interestingly, we notice that adding more multi-hop layers does not further improve the accuracy either. The best overall accuracy score of 99.25 % is reported using GCN with 2 multi-hop layers. When we add more multi-hop layers, it leads to message smoothing, which loses information in the process. As indicated in figure 6.1, when we add more multi-hop layers in the GNN message passing process, the information flows from itself to its neighbors and from its neighbors to other neighbors, thus allowing to flow the information outward into the graph network.

Table 6.4: Evaluation on standardised metrics

Class	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Attack	100.0	98.55	99.27
Normal	98.47	100.0	99.23

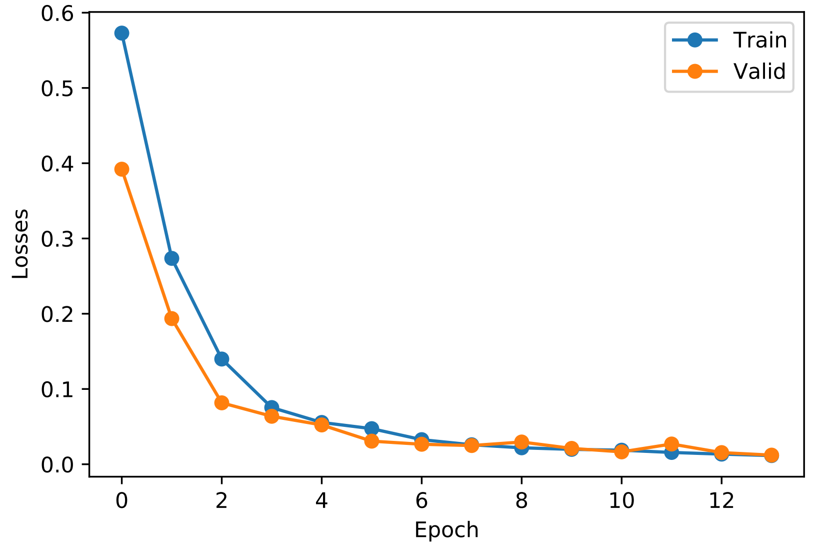


Figure 6.3: GCN-2 Model Loss Plot

As observed GCN-2 produces a smooth training loss and converges to very minimum loss withing first 5 training epochs cycle. Also, the accuracy of the model jumps above 95% within 2 epochs.

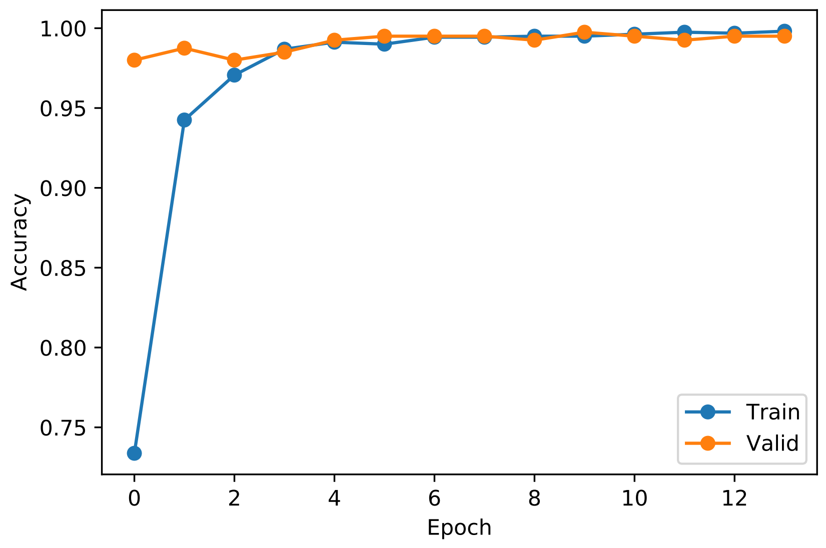


Figure 6.4: GCN-2 Model Accuracy Plot

Comparison with GAT and Graph Transformer Conv models

As we can see, when compared to advanced modified GNN architectures such as the Graph Attention Network (GAT), which exploits the attention mechanism as described in theory section 4.6 on GAT and the Graph Transformer Convolution architecture (GTC), they do not seem to supersede the performance of our classical Graph Convolution network (GCN)

Table 6.5: GNN Variants

GNN Type	Avg Loss	Accuracy (%)
GCN	**0.026**	**99.25**
GAT	0.121	95.67
GTC	0.734	50.58

model. Thus we can conclude that adding more complexity does not help improve the model performance. Instead, a simple GCN with 2 multi-hop layers was shown to perform the best for Attack classification on our decoded ADS-B dataset. It is also important to note here that GCN model inference time was the fastest compared to GAT and GCT models. This is due to added complexity, such as transformer blocks, which increase training and inference time. The training time for each of the GNN variants is shown below. Thus GCN becomes the optimal choice for our real-time attack classification system.

Table 6.6: Training Time of GNN models

GNN Type	**Training Time**
GCN	**3.87**
GAT	11.61
GTC	29.66

Table 6.7: Avg. Testing Time of GNN models

GNN Type	**Inference Time**
GCN	**0.033**
GAT	0.044
GTC	0.123

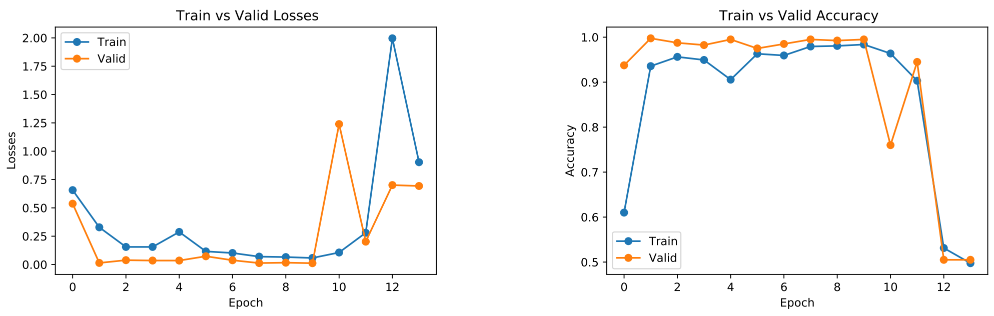


Figure 6.5: GAT Model training and accuracy plot

In the above plot, we observe that GAT training loss fluctuates between epochs before it stabilizes, but when compared to GCN loss, we do not see any smooth transition. For accuracy, our average is not higher than the GCN model either. As can be seen, more epochs do not significantly help in model improvement. Instead, we observe a slight drop in accuracy values.

Table 6.8: GAT Confusion Matrix

Pred/Real	**Attack**	**Normal**
Attack	0	0
Normal	200	200

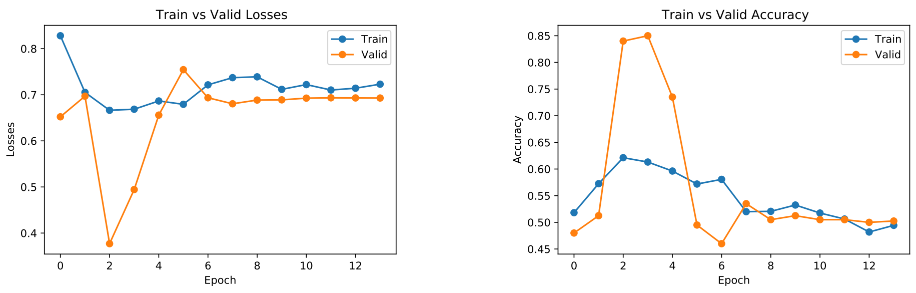


Figure 6.6: GTC Model training and accuracy plot

Table 6.9: GTC Confusion Matrix

Pred/Real	**Attack**	**Normal**
Attack	187	0
Normal	13	200

In the GTC loss graph, the model leads to over-fitting, thus increasing the validation loss and no further improvement in accuracy. The GTC model gives us the least average accuracy among our other two GNN variants.

Bayesian parameter search

We do a Bayesian evaluation of our model to measure the uncertainty of the model in action. There are basically 2 ways you can measure uncertainty 1) uncertainty in data which is also called Aleatoric uncertainty, and 2) uncertainty in the model, known as epistemic uncertainty. In this method, we are going to explore the epistemic uncertainty since it deals with uncertainty in the model, for which we can use techniques such as Bayesian neural networks or Monte-Carlo dropout. There are 2 ways we interpret probability in frequentist vs Bayesianism. In frequentist, we assume data follow a distribution, and we perform point estimates of the model parameters such as maximum likelihood estimation. In Bayesian, we assume both data to follow a distribution and the model. Instead of having a fixed model, the weights are described probabilistic-ally. P(theta) is prior. They believe in the world, which is the prior, and combine this with data. Thus we can use the Bayes rule to calculate maximum A posteriori estimation.

To measure this, we can use Bayesian neural networks or BNN, where the weights for each neuron in the network are sampled from a certain distribution, e.g., a Gaussian distribution. These sampled weights represent an ensemble of networks on which we run our data and calculate the uncertainty at the output. However, it is important to note here that to perform back-propagation in BNN. We require a reparametrization trick since we cannot back-prop through random nodes in the network. The task is to approximate the true posterior distribution. This is done using variational inference. To compare the approximated and true distribution, we use the KL divergence, which helps us measure the dissimilarity between the two. This is then combined with the loss function. Although BNNs, in theory, are powerful, implementing them in practice brings additional computational costs.

Thus in this scenario, we use dropout as a Bayesian approximation inspired by the paper on Representing Model Uncertainty in Deep Learning[13]. The dropout estimation minimized the GCN loss over a uniform distribution of dropout values in the range (0.2, 0.8) for a total max evaluation of 10 steps. After optimizing the training of our GCN model using dropout from a uniform distribution, our model confidently predicts high accuracy for dropout value with **0.33**. Thus we can confidently use the GCN model with the parameters (dropout as 0.33), yielding a high confidence score. The best loss returned reported 0.3081.

GCN on raw IQ samples

When training a GCN model on IQ samples, we report a mean loss of 0.024 and test accuracy score of 99.54% which is comparatively similar to the accuracy of a GCN model trained on decoded ADS-B message data. This shows that our GCN variant can be easily adopted to new incoming data without modifying underlying architecture and compromising on performance. The raw IQ samples consists of 480 features of a quadrature signals with its I and Q components. To accommodate this high dimensional data we use a GCN model with 3 convolution layers and a hidden dimension of 256 units.

The following table 6.10 and 6.11 summarises the confusion matrix of GCN on test data and its precision, recall, and F1 score.

Table 6.10: Confusion Matrix

Real/Pred	**Attack**	**Normal**
Attack	1169	8
Normal	3	1220

Table 6.11: Model Prediction score

Class	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Attack	99.74	99.32	99.53
Normal	99.35	99.75	99.55

The following two tables illustrates the training and inference time of GCN model on IQ sample data. Although the learning rate and epochs were kept the same for GCN on adsb and IQ data, the training time is higher in IQ data due to large number of features of 480 compared to only 6 features in decoded adsb data.

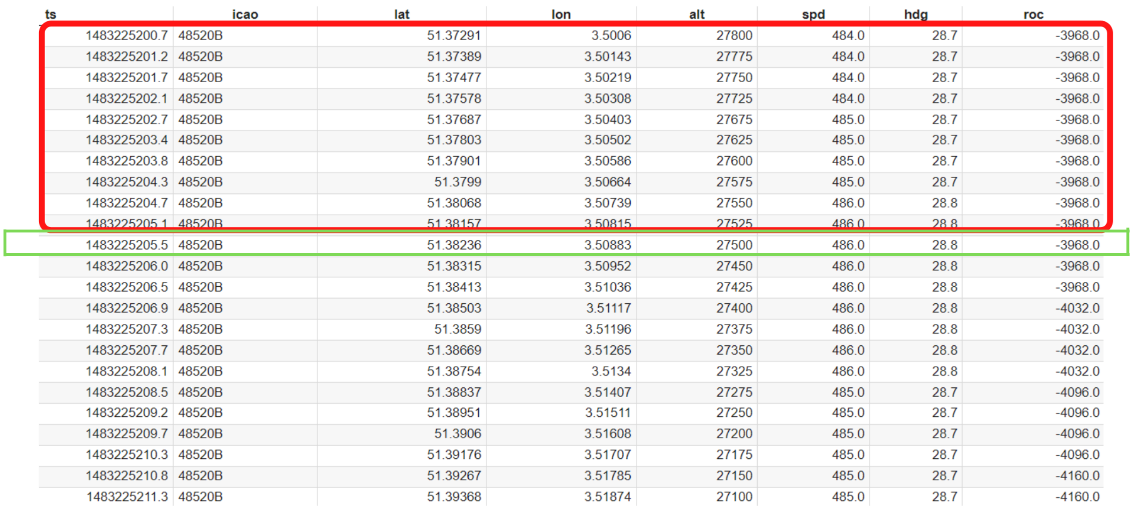
Table 6.12: Training Time of GCN on IQ data

Model	**Training Time**
GCN	86.44

Table 6.13: Avg. Testing Time of GCN on IQ data

Model	**Inference Time**
GCN	0.014

6.4 Anomaly prediction with Wavenet



ts	icao	lat	lon	alt	spd	hdg	roc
1483225200.7	48520B	51.37291	3.5006	27800	484.0	28.7	-3968.0
1483225201.2	48520B	51.37389	3.50143	27775	484.0	28.7	-3968.0
1483225201.7	48520B	51.37477	3.50219	27750	484.0	28.7	-3968.0
1483225202.1	48520B	51.37578	3.50308	27725	484.0	28.7	-3968.0
1483225202.7	48520B	51.37687	3.50403	27675	485.0	28.7	-3968.0
1483225203.4	48520B	51.37803	3.50502	27625	485.0	28.7	-3968.0
1483225203.8	48520B	51.37901	3.50586	27600	485.0	28.7	-3968.0
1483225204.3	48520B	51.3799	3.50664	27575	485.0	28.7	-3968.0
1483225204.7	48520B	51.38068	3.50739	27550	486.0	28.8	-3968.0
1483225205.1	48520B	51.38157	3.50815	27525	486.0	28.8	-3968.0
1483225205.5	48520B	51.38236	3.50883	27500	486.0	28.8	-3968.0
1483225206.0	48520B	51.38315	3.50952	27450	486.0	28.8	-3968.0
1483225206.5	48520B	51.38413	3.51036	27425	486.0	28.8	-3968.0
1483225206.9	48520B	51.38503	3.51117	27400	486.0	28.8	-4032.0
1483225207.3	48520B	51.3859	3.51196	27375	486.0	28.8	-4032.0
1483225207.7	48520B	51.38669	3.51265	27350	486.0	28.8	-4032.0
1483225208.1	48520B	51.38754	3.5134	27325	486.0	28.8	-4032.0
1483225208.5	48520B	51.38837	3.51407	27275	485.0	28.7	-4096.0
1483225209.2	48520B	51.38951	3.51511	27250	485.0	28.7	-4096.0
1483225209.7	48520B	51.3906	3.51608	27200	485.0	28.7	-4096.0
1483225210.3	48520B	51.39176	3.51707	27175	485.0	28.7	-4096.0
1483225210.8	48520B	51.39267	3.51785	27150	485.0	28.7	-4160.0
1483225211.3	48520B	51.39368	3.51874	27100	485.0	28.7	-4160.0

Figure 6.7: Illustration of sliding window

Table 6.14: ADS-B features analysis for attacks

Model	Lat(deg)	Lon(deg)	Alt(m)	Spd(knots)	Hdg(deg)	Roc
LSTM	0.717	0.5529	**917.55**	20.4003	142.30	2734.67
Wavenet	**0.5433**	**0.4485**	5379.98	**2.133**	**13.108**	**51.058**

In order to perform anomaly detection using our Wavenet model, we need to first to transform our ADS-B data into time-series data. The ADS-B messages from each aircraft having a unique ICAO are sorted according to their timestamps. This applies to all the features of the ADS-B message protocol. Later we apply a min-max scalar transformation to the six features in our dataset. We use a sliding window over the input sequences. Here we set the window length to 10 (which means we listen to approximately 5 seconds of ADS-B messages) and predict the next timestamp, i.e., the 11th sequence as indicated in the 6.3. If the predicted values result in a high reconstruction error, then we classify it as an anomaly. This error choice is made using a threshold value which we learn by calculating the average difference between true sequence values and predicted values. To achieve this, we train our Wavenet model on each aircraft time distributed dataset and slide the window over 10 sequences at a time to make the model - learn and predict the next values in the sequence. We train the model over each feature from our aircraft data separately and measure the loss as an absolute mean error. Our Wavenet architecture uses a 1D Convolution with 16 filters as a prepossessing step which transforms our input features sequence into 16 channels followed by two 1D Convolution layers with a dilation factor. This process is repeated over a set of dilation factors. In our case, it is 8. Finally, we add all the results and pass them through the nonlinear relu activation function. The model is trained for a total 15 epochs on each feature. To measure the train and validation loss, we use mean absolute error. Our validation data is 20% of our original training set.

Residual error

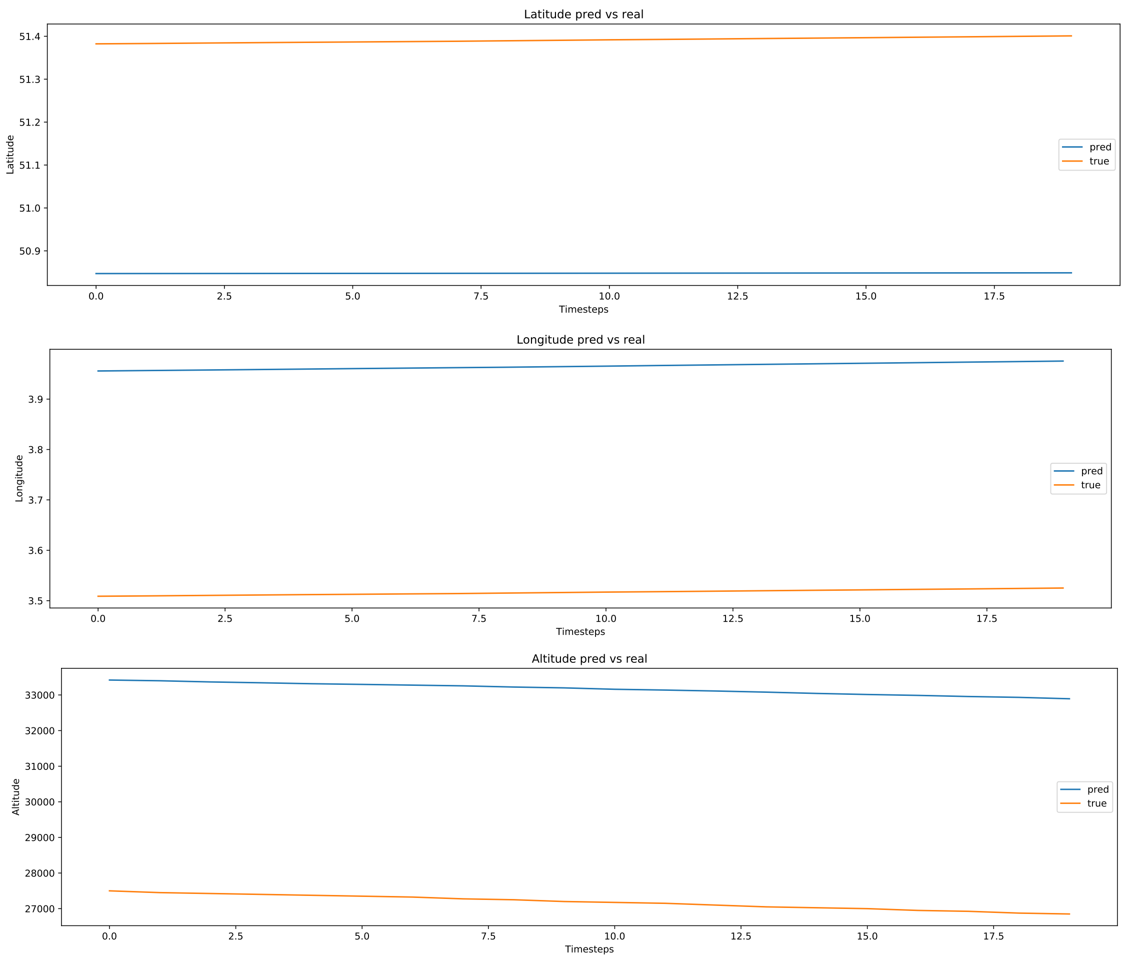


Figure 6.8: Predict next 20 time-steps

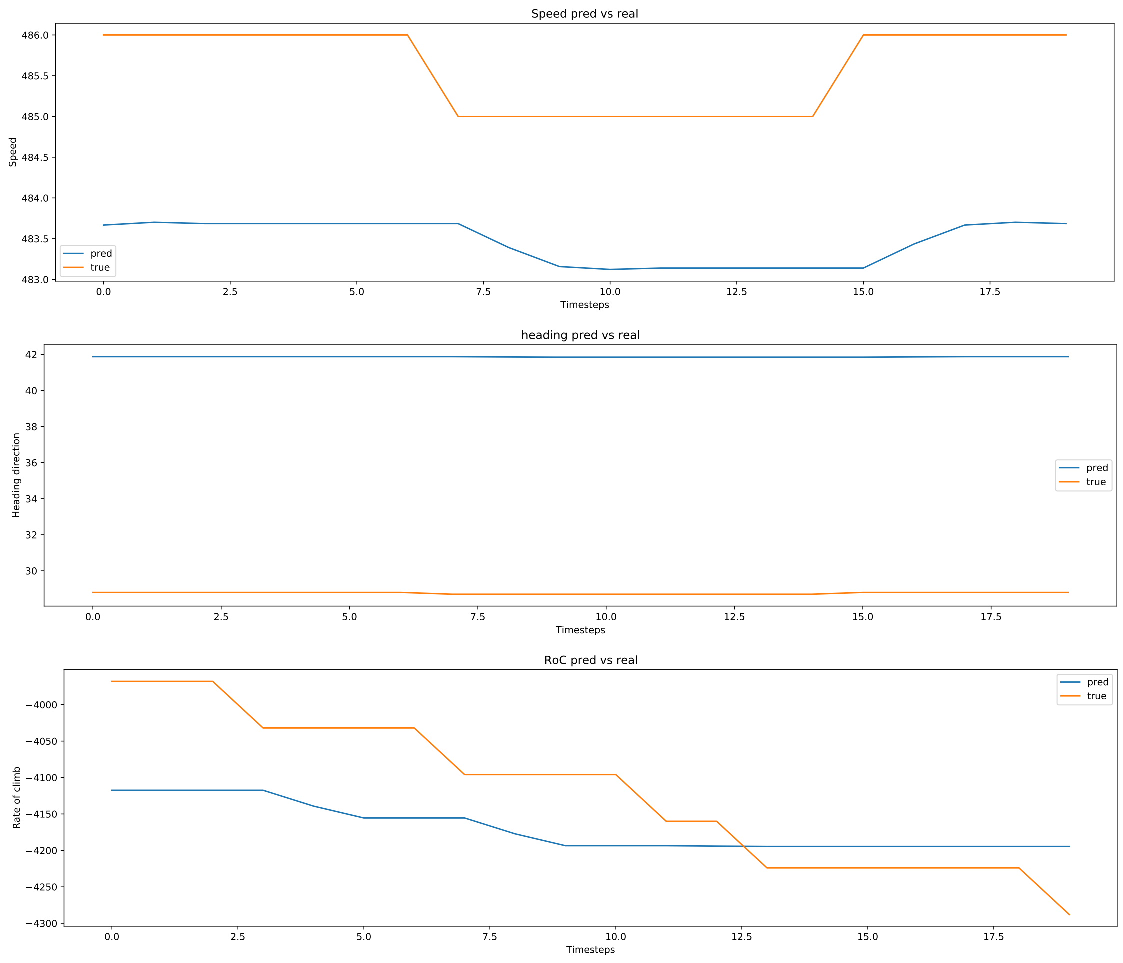


Figure 6.9: Predict next 20 time-steps

In order for us to visualize how the Wavenet model has learned and whether the predictions from the output make sense, we chose a 20-time steps sequence from the ADS-B message data as input and get the output predictions from the model. These predicted outputs are plotted against the actual true values for those 20-time steps and compared. To calculate the average residual error, we take the difference between the predicted and true value for each time step and average it. The goal is to closely follow the trend of true values over period of time. Consider the example of Latitude feature as we see, we get an average residual error of 0.5433 as indicated in Table 6.8. The Fig 6.8 shows us the plot for Latitude, Longitude and altitude values for predicted vs real.

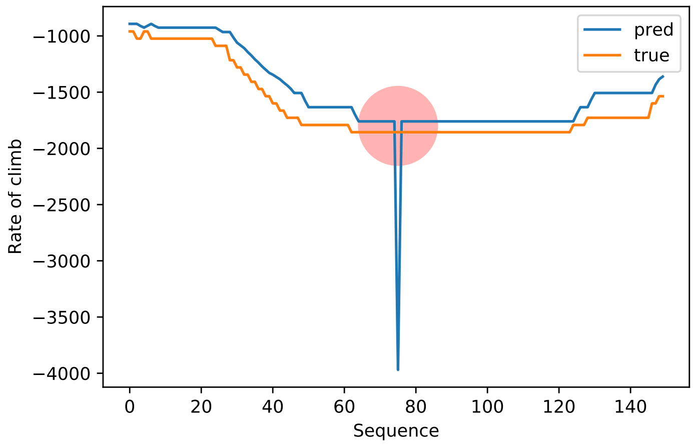
Similarly, we also plot the speed, heading direction, and rate of climb values over 20 time-steps in fig 6.9, respectively. For instance, when we observe the output of speed we notice that the predicted values closely follow the trend of the true speed curve, with an average difference of 2.133. But, we also notice a higher average predicted difference for altitude and rate of climb features. More specifically, the model somehow struggles to capture a close correlation with the altitude feature. This may be because we have a higher range of values in altitude which is measured in feet (ft) compared to other features. This, when scaled down via min-max scalar, in turn, increases the average difference. Although we get a very high average difference for altitude and rate of climb, as highlighted in table 6.8, it can still successfully identify software-based simulated attacks (which we discuss more in the following section) given the set threshold of 0.015 which we calculated above.

Threshold setting:

After the model is fully trained, on our training data, i.e. (M1), we use our test data (M2) as input to the model and obtain a set of predicted values P. The set of true values corresponding to P is V, and the residual set of P and V is defined as D, which is formulated as follows:

$$D = |P - V| \tag{6.1}$$

Once we take the absolute difference among predicted and true values, we average them, which gives us the mean μ, and simultaneously we also calculate the standard deviation σ^2. Using this information, we can set our threshold to $T = 3 * \sigma$. It is worth noting that the anomaly thresholds for different features are different. This way, we can better do a feature by feature analysis rather than relying on one threshold value; nevertheless, we decide to hold on to one threshold value by combining the anomaly threshold values obtained from all features and average them to give users one global threshold value, which in our case was $T = 0.015$



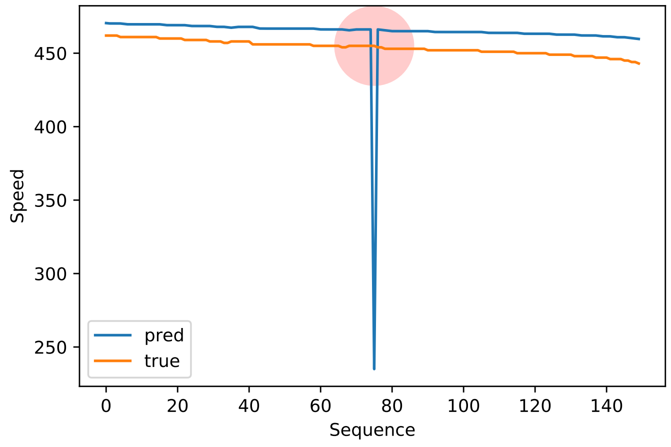


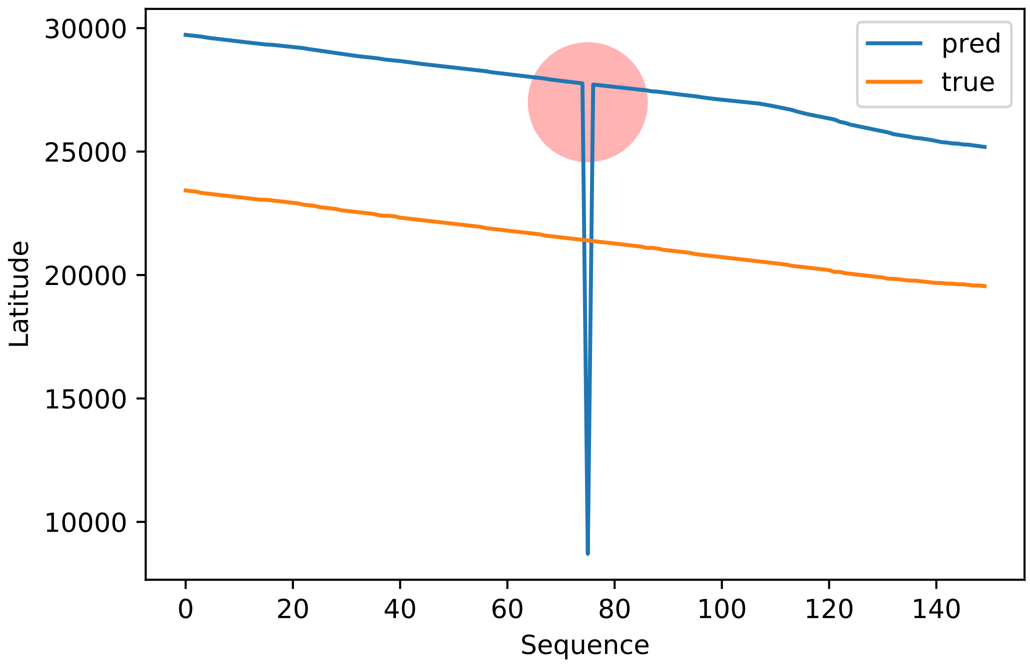
Figure 6.10: Anomaly Detection in RoC and Speed

Test for software simulated attacks:

Table 6.15: ADS-B Data software simulated attacks [40]

Attack	Simulation	Method	Result
Jamming	Random noise	Multiply the flight value obtained in the original ADS-B message by a random value between 0 and 1.	Success
Injection	feature replacement	Given certain feature information, inject different correct feature values to replace the sequence for the selected ADSB sequence segment.	Success
Modification	speed offset(+)	Use 20 knots as multiples to gradually change the speed characteristics of ADS-B messages by increasing speed by 20 knots each time.	Success
	Speed offset(−)	Decrease speed by 20 knots each time.	Success
	altitude offset(+)	Use 400 ft as multiples to gradually change the altitude characteristics of ADS-B messages by increasing altitude by 400 ft s each time.	Success
	Altitude offset(−)	Decrease altitude by 400 ft each time.	Success (0.33 > Th)
	Heading change	Change the value of the heading info contained in the ADS-B message to the opposite of the original value.	Success
	Climb rate change	Change the value of the climb rate contained in the ADS-B message to the opposite of the original value	Success

To comprehensively evaluate our trained Wavenet model, we perform several software simulated attacks such as random noise, jamming, injection, and modification, as shown in the table. The goal here is to successfully identify these attacks. For example, Suppose we modify the speed value in the ADS-B data on its way to the ground station, given the history of ADS-B data during model prediction. In that case, it should be able to detect deviation and anomaly in the predicted value. The resulting difference between predicted and actual should be higher than the threshold.



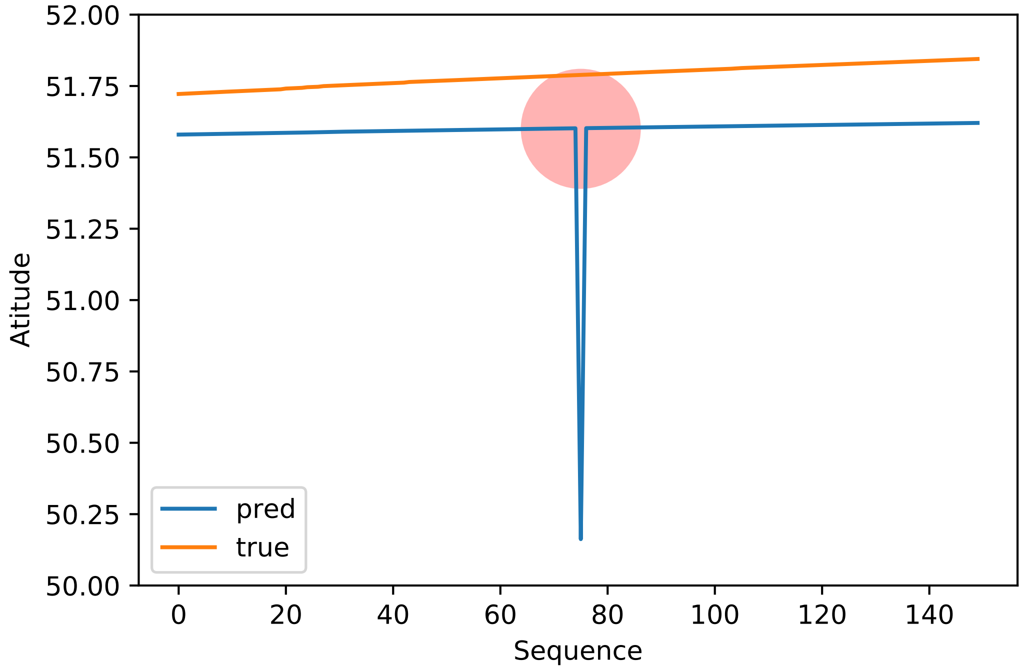


Figure 6.11: Anomaly Detection in Altitude and Latitude

Abnormal score visualization

Figure 6.10 and 6.11 illustrates the abnormal score for jamming attacks on various features such as rate of climb, speed, altitude, and latitude, along with their respective real normal score. The jamming attack was produced by adding random noise between 0 and 1. The abscissa is the data serial number, and the ordinate is the abnormal score. The red circle highlighted for the spikes in the predicted score indicates an anomaly. It means the normal data was attacked at the time when the spike originated. This sudden spike is what we measure as an anomaly, and if it crosses the set threshold of 0.015, we alert the system.

6.5 GAN Model

The GAN model was used to simulate real data as attacks on real ADS-B messages. The GAN training proceeds iteratively, with the generator being initialized with a randomly generated noise vector with 128 dimensions (from a uniform distribution), which is used to generate the first set of "fake" samples. This fake data is used as input to the discriminator. The discriminator also receives the real data as input and, on each sample (real or fake), tries to predict the correct label for the data. The output from the discriminator is used as feedback to the generator, which uses this feedback to improve and update its data generation mechanism. We trained the GAN for 2500 epochs with a batch size of 128. We used the Binary Cross Entropy Loss for our implementation, which was minimized over the given epochs. After tuning, the trained discriminator network was able to identify the fake samples from the generator network with an accuracy of 96.88 %. As a result, the distribution of the generated ADS-B data matches those of the real ADS-B samples very closely.

Figure 6.6 below shows the absolute mean of 6 features from real ADS-B data, and the line is an approximation learned by our GAN model to closely follow the ADS-B message characteristics.

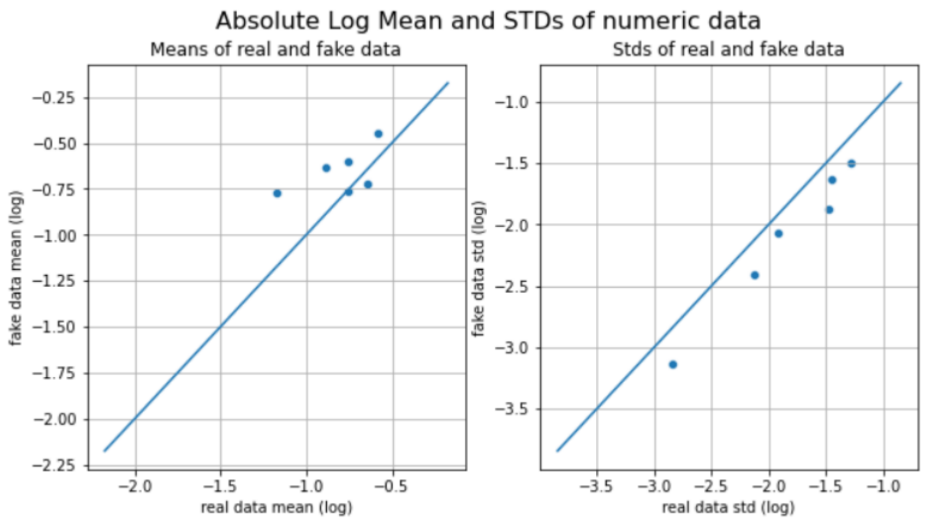


Figure 6.12: Absolute mean and standard deviation of all six features for predicted and real data

To evaluate GAN more robustly we look at cumulative sum plot between the generated and real data for each of the six features as follows:

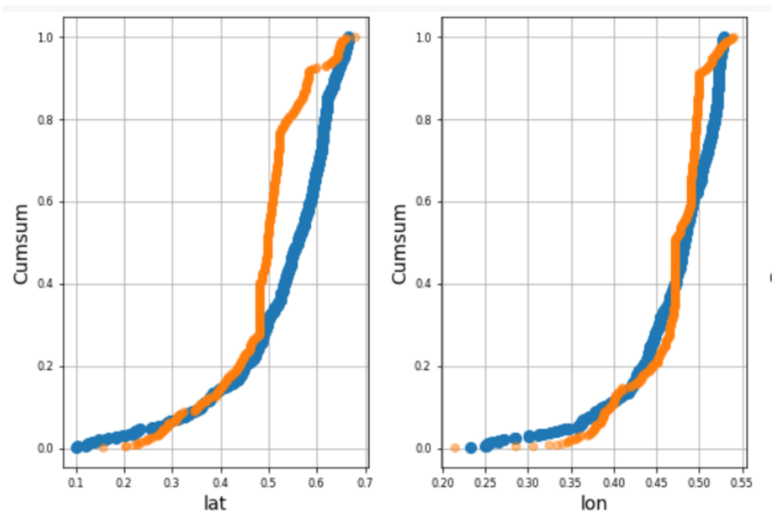


Figure 6.13: Latitude and Longitude generated v/s real

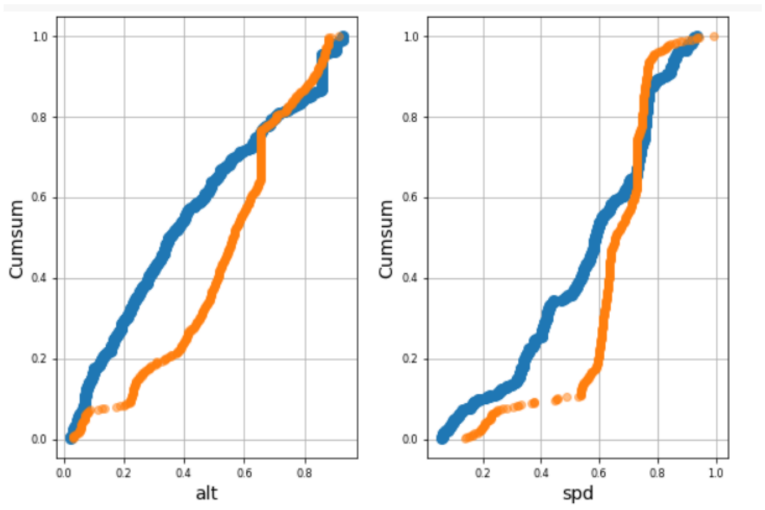


Figure 6.14: Altitude and Speed generated v/s real

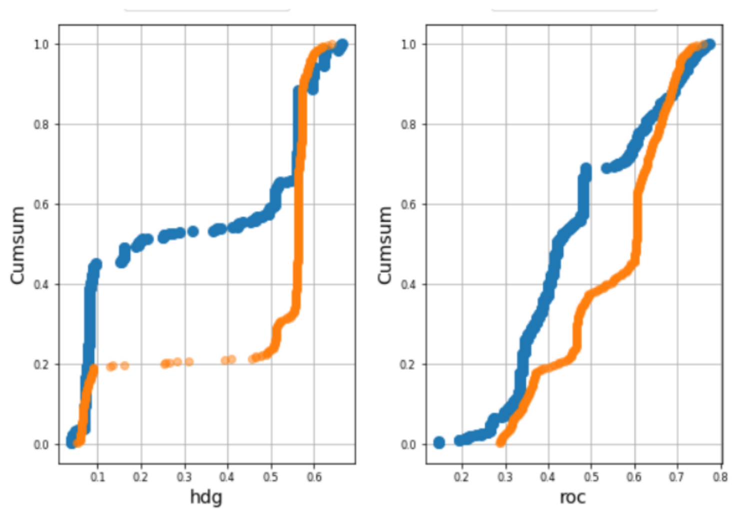
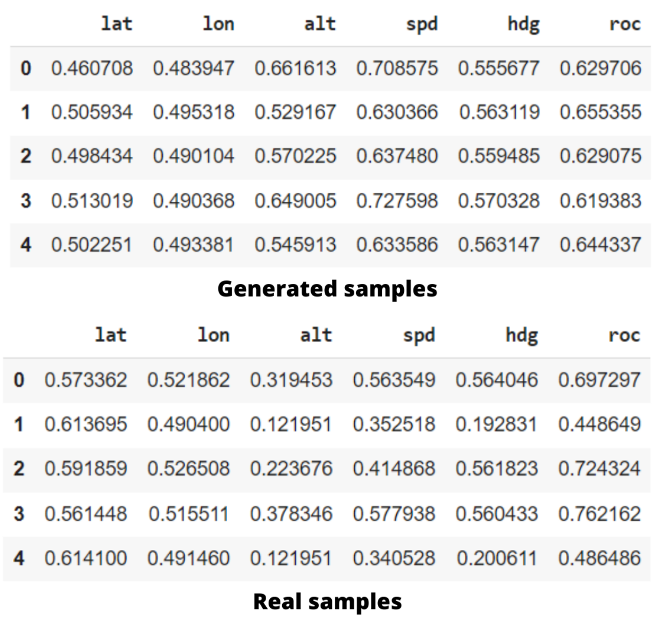


Figure 6.15: Heading direction and rate of climb generated v/s real



	lat	lon	alt	spd	hdg	roc
0	0.460708	0.483947	0.661613	0.708575	0.555677	0.629706
1	0.505934	0.495318	0.529167	0.630366	0.563119	0.655355
2	0.498434	0.490104	0.570225	0.637480	0.559485	0.629075
3	0.513019	0.490368	0.649005	0.727598	0.570328	0.619383
4	0.502251	0.493381	0.545913	0.633586	0.563147	0.644337

Generated samples

	lat	lon	alt	spd	hdg	roc
0	0.573362	0.521862	0.319453	0.563549	0.564046	0.697297
1	0.613695	0.490400	0.121951	0.352518	0.192831	0.448649
2	0.591859	0.526508	0.223676	0.414868	0.561823	0.724324
3	0.561448	0.515511	0.378346	0.577938	0.560433	0.762162
4	0.614100	0.491460	0.121951	0.340528	0.200611	0.486486

Real samples

Figure 6.16: Snapshot of Generated v/s Real samples comparison

We see that for latitude and longitude features, the generated data is closely able to follow the real data point distribution, followed by speed, altitude, and rate of the climb, which to some extend capture the real trend. But for the heading direction, the GAN model somehow struggles to reach the real data distribution. This may be due to the sparsity in the feature values, which may be hard for GAN to learn and generalize. Finally, this generated data, along with real data, is then classified into the attack and normal data and fed to our attack classification model.

6.6 Aircraft classification using DNN

Finally, to counter spoofing attacks in which the ADS-B data will impersonate an existing aircraft with its ICAO address, we need to use transmitter signal characteristics for true aircraft identification, as discussed previously. We cannot use an attack classification or an Anomaly detection framework in this case since the ADS-B message is not altered or harmed, but in fact, the intruder is trying to impersonate and fake its existence as a real aircraft. Thus if we identify an aircraft based on its unique RF signal fingerprint, we can classify which aircraft it is originating from and whether an entity is trying to fake its identity. We use the raw IQ signal data to achieve this and extract two important features, phase, and magnitude, from IQ samples using signal processing techniques. The phase and magnitude act as a unique fingerprint to classify which aircraft it belongs to. Our input to Deep Neural Network (DNN) is a multi-modal input (Phase+Magnitude), and outputs are the class labels for each unique aircraft or ICAO. DNN is 4 layers deep with 500 hidden units. The results after training are shown in the following table.

Table 6.16: Accuracy score for aircraft classification

Model	**Phase**	**Magnitude**	**(Phase+Magnitude)**
Accuracy(%)	69.79	70.3	**70.5**

6.7 Federated Learning

In order to demonstrate federated training in practice, we use the ADS-B data, which consists of Normal messages along with Dos, alterations, and spook attacks. We aim to build a classifier model that is sent to the client-side for training. In this experiment, we choose four different clients (i.e., aircraft) operating in different regions and send our model to be trained on the local data of every aircraft's ADS-B messages. Once training is complete on the client-side, we fetch the trained model parameters on our main server for inference. We use the Pysyft library, which is built on top of Pytorch for secure and private deep learning. It uses federated learning, differential privacy, and encrypted computation. It provides a platform for distributed training of our model using peer-to-peer client-server connections. To begin, we create four virtual workers and hook our tensor to them. We send a dataset of one aircraft to these four workers or clients to one worker. From there on, we build our model architecture on the server-side, and during training, we send our model to each of the workers. For every epoch we send and train the model, we get back the trained parameters, aggregate it using federated averaging and continue training until complete.

The global trained model is averaged over all the training parameters received from its worker clients (aircraft). This model is then sent to each of the aircraft as a new updated model. To estimate the model performance, we evaluate it on test data consisting of attacks and observe its precision and recall score on three different attacks (alteration, Dos, and spoof attacks). The following table shows the results of the global trained model on our test data with an overall test accuracy of **89.0%**.

Table 6.17: Training a classifier with Federated learning

	Precision(%)	**Recall(%)**
Alteration	77.78	80.77
DoS	100.0	78.57
Spoofing	81.48	0.0

6.8 Final Summary: Best performing model

Table 6.18: Summary of all attack classification models

Model (on IQ samples)	**Detection score (%)**	**False alarm rate (%)**
ADS-B (SODA) DNN [41]	99.34	0.43
GAN [19]	98.74	7.1
SVM [30]	80.29	25.67
NN (1 hidden, 10 nodes) [26]	91.4	13.8
Our model (GCN on IQ)	**99.74**	**0.25**
Our model (GCN on adsb)	**99.99**	-

The above table summarizes the detection score and false alarm rate for various models referenced from previous work. Basically, Detection score is: the percentage of malicious messages classified as malicious and False alarm rate is: the percentage of authentic messages classified as malicious. We show that our proposed solution outperforms all other models in terms of high detection score and very low false alarm rate, which means the model on very rare occasions predicts false attacks, this can help us save costly time involved in analysing ADS-B data if there wasn't any attack and lead to less panic situations during air-ground communication.

It is important to note here that adsb data was considerably smaller in size when compared to IQ samples and this experiment shows application of graph representation learning on both data-sets yields a very high attack detection score.

Table 6.19: Comparison of Prediction Indicators of Different Models

Model	**RMSE/Altitude(m)**	**RMSE/Speed (Knots)**	**RMSE/Heading (degrees)**
LSTM [11]	1249.43	11.77	21.92
Bi-LSTM [18]	**810.62**	10.45	19.51
Our proposed model	917.55	**2.133**	**13.108**

We also compare Root mean squared error or RMSE values of different features with previous model implementation such as LSTM and Bi-LSTM and show that our proposed Wavenet model clearly outperforms the previous two implementations except for Altitude feature.

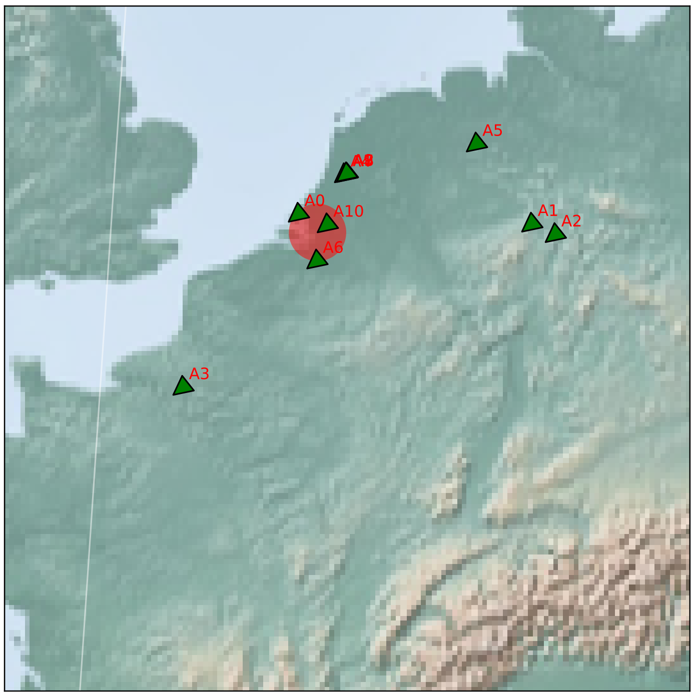


Figure 6.17: Lambert conformal conic projection of airspace

In figure 7.1 we depict Lambert conformal conic projection (LCC) of the entire airspace for different aircraft's position in a region for a fixed time, we can observe in the picture that the area circled in red with aircraft A10 is a fake aircraft and the data for it was generated using GAN model, whereas other aircraft's are legitimate and taken from real ADS-B data.

7 Discussion

7.1 Results

The results for our 3 stage Intrusion detection system is a combination of 3 models, first is the attack classification using Graph Convolutional network to classify ADS-B message data into normal or attack we report a high accuracy of 99.25% for this model. It classifies the entire airspace region consisting of multiple aircraft node and its features into an attack if it finds attack behaviour characteristics in data. Next stage is fed to the Anomaly detection model which observes the ADS-B data over period of time and makes future prediction of values if the resulting values results in abnormal score than we identify it as an anomaly. The model we selected for the time series prediction of ADS-B data is Wavenet, it is successfully able to detect anomaly with an overall threshold of 0.015. Finally the last stage is an aircraft classification module that separates each data signal extracted using the IQ samples of the receiver aircraft into its corresponding ICAO addresses class.

To touch upon how our proposed solution considers all the risk averse factors we discussed in section 1.6, firstly to prevent adversarial attacks we show an example of how federated learning can be used to enforce data privacy and avoid data being externally manipulated by the system. Moreover federated training also allows us to train model locally on each aircraft data and aggregate a global model, thus reducing the bandwidth and server cost. Another risk we discussed was not having enough data to validate, to counter this we use GAN's or Generative Adversarial Network to simulate and generate more data for use to be used for training the model. And finally to measure model uncertainty we used the technique called dropout as Bayesian parameter optimization which helps us evaluate how confident is our model.

7.2 Method

The methods used in the project are motivated from the latest advancement in deep learning applied to graph data and use of convolution based generative models over LSTM and GRU based approach for modelling time series data. The advantage of using graph neural network (GNN) over standard deep neural network (DNN) is that in a typical DNN the attack classification model needs to be trained for each sample of ADS-B data from an aircraft separately which increases the overall training time and may involve risk of over-fitting. Whereas the use of graph based learning helps us analyse the entire airspace in one short, and capture ADS-B of data from multiple air-crafts during attack classification which leads to faster model training and convergence. Moreover it helps us capture higher level of details from air-to-air communication which is message transfer between air-crafts. To illustrate this consider air-craft A sends ADS-B data to ground station as well as to its neighbouring aircraft(s) in this case consider a neighbour aircraft B. Thus if an intruder attacks message signal on its way to ground station, the aircraft B has already received the authentic message data, which can be an important piece of information for attack classification. Thus due to graph enabled learning we can leverage these message passing between air-crafts (nodes) and include it during model training.

The method used for aircraft classification is motivated from paper [19]. Here the data is extracted from IQ signal of each aircraft. The data processed from IQ samples is the phase and magnitude information of the receiver signal. Thus the input to our classification model is a multi-modal which gives additional information about aircraft sender signal characteristics. This is alternative approach from the previous work which only use raw IQ samples as 2D data (I + Q) to train a classifier model.

7.3 The work in a wider context

With the current system in place there is always scope for continuous improvement, one immediate change we can establish in our attack classification model is advancing it from spatial graph classification to Spatio-Temporal graph this helps us model the complete picture of an airspace over a series of time. Thus enabling us model time dependence which is an important aspect for anomaly detection along with its spatial features. An interesting research in this direction has been done using Attention Temporal Graph Convolutional Network for Traffic Forecasting [43]. Furthermore there are still a variety of attacks that needs to be handled and detected autonomously such as Sybil attack (impersonates multiple air-crafts at once) which is outside the scope of this project. This requires additional modification to the architecture and training data, one solution could be to train an AI agent in a simulation environment to detect and identify multiple attacks onto on aircraft, the AI model is trained using reinforcement learning such that AI agent will get rewarded each time it identifies an attack and punished for not. The entire training can be done in simulation thus no need of external hardware interface to test attacks using ADS-B equipment.

Another interesting direction we can explore is using Mel-frequency Spectrogram extracted from RF noise detection signal. Every aircraft transponder equipment sends signal to the ground station with a unique RF prints. This distinct radio noise can be treated as an audio which is transformed into a Spectrogram representation of frequency-time domain by applying Short Time Fourier Transform or STFT, this is done in order for ease of processing of audio signal by the neural network. This 2D spectrogram is then fed into an autoencoder for example which will help use reconstruct the spectrogram input, if it results in high reconstruction loss we could identify it as an anomaly signal.

8 Conclusion

In this thesis, we have demonstrated the use of a novel architecture framework to leverage graph data. We comprehensively discussed how we can transform the ADS-B data into graph representation and how it can be used to learn the airspace feature representation. We have thoroughly touched upon all our research questions from implementation of IDS system to performing various test to evaluate model robustness such as message modification, injection, and replay attacks, moreover we also touched upon how we can handle spoofed attacks using phase/magnitude of IQ signal. To evaluate confidence score we use dropout as Bayesian approximation to find which set of hyper-paremter give us the highest accuracy. We also looked into on how adversarial attacks can be prevented by using GAN's to generate fake ADS-B data that simulate our real ADS-B messages. Though we have established the efficacy of our proposed method with these experiments illustrated in table 6.3 for hardware based attacks using GCN and table 6.8 showcasing simulated software attacks identified by Wavenet, we would like to further exploit the performance of our system with real spoofed ADS-B data. For more comprehensive analysis, we developed sequential time-series model for decoded ADS-B messages that help us do feature by feature analysis in order to identify irregularities or anomaly in values. This helped us identify the software based attacks whereas the former GCN attack classification module was implemented to identify hardware attacks. Above all, having two distinct models for various attack detection scenarios is not suffice as there are more difficult attacks such as spoofed/impersonation and Sybil attack which try to copy one or multiple aircraft behaviours of real aircraft entity, in order to tackle these kind of attacks we built a DNN module at the end which make use of the phase and magnitude features from IQ signal which remain unique to every aircraft based on the fact that every ADS-B emitter signal has unique RF fingerprint which can be leveraged for classification of an aircraft. Although we have discussed and covered multiple attacks and scenarios in during Air-ground communication with ADS-B protocol, the challenge still remains on how we can successfully deploy global model that can identify all kind of attacks in all operating regions with high confidence. This thesis provides methods to solve certain type of attacks using deep learning framework in real time, but requires additional modifications and real world testing to adapt to various cyber-attack and keep pace with advancing technology.

Bibliography

[1] Sefi Akerman, Edan Habler, and Asaf Shabtai. “VizADS-B: Analyzing Sequences of ADS-B Images Using Explainable Convolutional LSTM Encoder-Decoder to Detect Cyber Attacks”. In: June 2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. DOI: 10.48550/ARXIV.1701.07875. URL: https://arxiv.org/abs/1701.07875.

[3] Anton Blåberg, Gustav Lindahl, Andrei Gurtov, and Billy Josefsson. “Simulating ADS-B attacks in air traffic management”. In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE. 2020, pp. 1–10.

[4] An Braeken. “Holistic air protection scheme of ADS-B communication”. In: *IEEE Access* 7 (2019), pp. 65251–65262.

[5] Andrei Costin and Aurélien Francillon. “Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices”. In: *black hat USA* (2012), pp. 1–12.

[6] Vijay Prakash Dwivedi and Xavier Bresson. “A Generalization of Transformer Networks to Graphs”. In: *AAAI Workshop on Deep Learning on Graphs: Methods and Applications* (2021).

[7] Sofie Eskilsson, Hanna Gustafsson, Suleman Khan, and Andrei Gurtov. “Demonstrating ADS-B and CPDLC Attacks with Software-Defined Radio”. In: *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)*. IEEE. 2020, 1B2–1.

[8] “EUROPEAN AVIATION IN 2040”. In: © European Organisation for the Safety of Air Navigation, 2018. URL: https://www.eurocontrol.int/sites/default/files/2019-07/challenges-of-growth-2018-annex1_0.pdf.

[9] “Federal Aviation Administration”. In: United States Department of Transportation, 2022. URL: https://www.faa.gov/data_research/aviation/aerospace_forec.

[10] Ziliang Feng, Weijun Pan, and Yang Wang. “A data authentication solution of ADS-B system based on X. 509 certificate”. In: *27th International Congress of the Aeronautical Sciences, ICAS*. 2010, pp. 1–6.

[11] "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5 (2005). IJCNN 2005, pp. 602–610. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2005.06.042. URL: https://www.sciencedirect.com/science/article/pii/S0893608005001206.

[12] Ahmed Gad. "Introduction to Federated Learning". In: *Medium*. 2020.

[13] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. URL: https://proceedings.mlr.press/v48/gal16.html.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[15] Edan Habler and Asaf Shabtai. "Analyzing Sequences of Airspace States to Detect Anomalous Traffic Conditions". English. In: *IEEE Transactions on Aerospace and Electronic Systems* (Jan. 2021). Publisher Copyright: IEEE. ISSN: 0018-9251. DOI: 10.1109/TAES.2021.3124199.

[16] Edan Habler and Asaf Shabtai. "Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages". In: *Computers & Security* 78 (2018), pp. 155–173.

[17] Charles R. Harris, K. Jarrod Millman, Stéfan J.van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[19] Nikita Susan Joseph, Chaity Banerjee, Eduardo Pasiliao, and Tathagata Mukherjee. "FlightSense: A Spoofer Detection and Aircraft Identification System using Raw ADS-B Data". In: *2020 IEEE International Conference on Big Data (Big Data)*. 2020, pp. 3885–3894. DOI: 10.1109/BigData50022.2020.9377975.

[20] Thabet Kacem, Duminda Wijesekera, Paulo Costa, Jeronymo Carvalho, Márcio Monteiro, and Alexandre Barreto. "Key distribution mechanism in secure ADS-B networks". In: *2015 Integrated Communication, Navigation and Surveillance Conference (ICNS)*. IEEE. 2015, P3–1.

[21] Suleman Khan, Joakim Thorn, Alex Wahlgren, and Andrei Gurtov. "Intrusion Detection in Automatic Dependent Surveillance-Broadcast (ADS-B) with Machine Learning". In: *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. IEEE. 2021, pp. 1–10.

[22] Syed Khandker, Hannu Turtiainen, Andrei Costin, and Timo Hämäläinen. "On the (In) Security of 1090ES and UAT978 Mobile Cockpit Information Systems–An Attacker Perspective on the Availability of ADS-B Safety-and Mission-Critical Systems". In: *IEEE Access* 10 (2022), pp. 37718–37730.

[23] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: `1609.02907`. URL: `http://arxiv.org/abs/1609.02907`.

[24] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. "1D convolutional neural networks and applications: A survey". In: *Mechanical Systems and Signal Processing* 151 (2021), p. 107398. ISSN: 0888-3270. DOI: `https://doi.org/10.1016/j.ymssp.2020.107398`. URL: `https://www.sciencedirect.com/science/article/pii/S0888327020307846`.

[25] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. "Federated Learning: Strategies for Improving Communication Efficiency". In: *NIPS Workshop on Private Multi-Party Machine Learning*. 2016. URL: `https://arxiv.org/abs/1610.05492`.

[26] Mauro Leonardi, Luca Di Gregorio, and Davide Di Fausto. "Air Traffic Security: Aircraft Classification Using ADS-B Message's Phase-Pattern". In: *Aerospace* 4.4 (2017). ISSN: 2226-4310. DOI: `10.3390/aerospace4040051`. URL: `https://www.mdpi.com/2226-4310/4/4/51`.

[27] Mohsen Riahi Manesh and Naima Kaabouch. "Analysis of vulnerabilities, attacks, countermeasures and overall risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) system". In: *International Journal of Critical Infrastructure Protection* 19 (Oct. 2017). DOI: `10.1016/j.ijcip.2017.10.002`.

[28] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[29] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: (2016). DOI: `10.48550/ARXIV.1602.05629`. URL: `https://arxiv.org/abs/1602.05629`.

[30] Dustin M. Mink, Jeffrey McDonald, Sikha Bagui, William B. Glisson, Jordan Shropshire, Ryan Benton, and Samuel Russ. "Near-Real-Time IDS for the U.S. FAA's NextGen ADS-B". In: *Big Data and Cognitive Computing* 5.2 (2021). ISSN: 2504-2289. DOI: `10.3390/bdcc5020027`. URL: `https://www.mdpi.com/2504-2289/5/2/27`.

[31] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: `1609.03499 [cs.SD]`.

[32] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu koray, Oriol Vinyals, and Alex Graves. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper/2016/file/b1301141feffabac455e1f90a7de2054-Paper.pdf`.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[34] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. "A Gentle Introduction to Graph Neural Networks". In: *Distill* (2021). https://distill.pub/2021/gnn-intro. DOI: 10.23915/distill.00033.

[35] Matthias Schäfer, Vincent Lenders, and Ivan Martinovic. "Experimental analysis of attacks on next generation air traffic communication". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2013, pp. 253–271.

[36] Martin Strohmeier, Matthias Schäfer, Vincent Lenders, and Ivan Martinovic. "Realities and challenges of nextgen air traffic management: the case of ADS-B". In: *IEEE Communications Magazine* 52.5 (2014), pp. 111–118. DOI: 10.1109/MCOM.2014.6815901.

[37] Junzi Sun, Joost Ellerbroek, and Jacco Hoekstra. "Flight Extraction and Phase Identification for Large Automatic Dependent Surveillance–Broadcast Datasets". In: *Journal of Aerospace Information Systems* (2017), pp. 1–6.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations* (2018). URL: https://openreview.net/forum?id=rJXMpikCZ.

[40] Jing Wang, Yunkai Zou, and Jianli Ding. *ADS-B spoofing attack detection method based on LSTM*. Mar. 2020. DOI: 10.21203/rs.3.rs-19635/v1.

[41] Xuhang Ying, Joanna Mazer, Giuseppe Bernieri, Mauro Conti, Linda Bushnell, and Radha Poovendran. "Detecting ADS-B Spoofing Attacks Using Deep Neural Networks". In: June 2019, pp. 187–195. DOI: 10.1109/CNS.2019.8802732.

[42] Xuhang Ying, Joanna Mazer, Giuseppe Bernieri, Mauro Conti, Linda Bushnell, and Radha Poovendran. *Detecting ADS-B Spoofing Attacks using Deep Neural Networks*. 2019. arXiv: 1904.09969 [cs.CR].

[43] Jiawei Zhu, Yujiao Song, Ling Zhao, and Haifeng Li. *A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting*. 2020. DOI: 10.48550/ARXIV.2006.11583. URL: https://arxiv.org/abs/2006.11583.