CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORT SCIENCES

DEPARTMENT OF AIR TRANSPORT

Bc. Max Andreas Minev

Motion Dynamics Based Aircraft Trajectory Classification Using Neural Networks

Master´s Thesis

2022

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Transportation Sciences
Dean's office
Konviktská 20, 110 00 Prague 1, Czech Republic

K621 ..... Department of Air Transport

BACHELOR'S THESIS ASSIGNMENT

(PROJECT, WORK OF ART)

Student's name and surname (including degrees):

Bc. Max Andreas Minev

Study programme (field/specialization) of the student:

master's degree – PL – Air Traffic Control and Management

Theme title (in Czech): **Klasifikace trajektorií letadel dle dynamiky pohybu s využitím neuronových sítí**

Theme title (in English): **Motion Dynamics Based Aircraft Trajectory Classification Using Neural Networks**

Guidelines for elaboration

During the elaboration of the master's thesis follow the outline below:

- The goal of the thesis is to propose and develop a neural network capable of aircraft track classification based on the aircraft flight dynamics and evaluate its performance using available real-life ADS-B data.
- Research of the related field.
- Proposal of the method of track segmentation using real-life ADS-B data for the training of the neural network.
- Proposal of a neural network capable of track classification based on the flight dynamic.
- Training the neural network on real-life ADS-B data.
- Evaluation of the neural network classification performance and discussion of the added benefits to the field of aircraft tracking.

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Graphical work range: according to the instructions of thesis supervisor

Accompanying report length: minimum of 55 text pages (including figures, graphs and sheets which are part of the main text)

Bibliography:

BROOKNER, Eli. Tracking and Kalman filtering made easy. New York: Wiley, 1998.

Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2 ed. O'Reilly, 2019

Ismail Fawaz, et al. Deep learning for time series

Master's thesis supervisor: **Ing. Petr Lukeš**

Date of master's thesis assignment: **July 16, 2021**
(date of the first assignment of this work, that has be minimum of 10 months before the deadline of the theses submission based on the standard duration of the study)

Date of master's thesis submission: **November 30, 2022**
a) date of first anticipated submission of the thesis based on the standard study duration and the recommended study time schedule
b) in case of postponing the submission of the thesis, next submission date results from the recommended time schedule

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní L.S. -3-

doc. Ing. Jakub Kraus, Ph.D.
head of the Department of Air Transport

prof. Ing. Ondřej Přibyl, Ph.D.
dean of the faculty

I confirm assumption of master's thesis assignment.

Bc. Max Andreas Minev
Student's name and signature

Prague ..... May 17, 2022

Acknowledgement

Firstly, I would like to thank my supervisor Ing. Petr Lukeš for his helpful and insightful guidance when conducting the research and writing this Master´s thesis. Ing. Jakub Mejznar also deserves a mention thanks to his insightful assistance with the SW development for this thesis. I would also like to thank Ing. Andrea Mineva for her assistance with the graphics that are present in this thesis. My thanks also belong to my family, close friends and colleagues who supported me in my studies throughout the years.

Affidavit

I declare that I have prepared my thesis independently and that all used information sources I have provided in accordance with the Methodological Guideline on the observance of ethical principles in the preparation of university theses.

I have no valid reason against the use of this school work within the meaning of § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

In Prague on 30.11. 2022 Signature: _____

Bc. Max A. Minev

List of Abbreviations

ANN	-	Artificial Neural Network
SML	-	Supervised machine Learning
DAG	-	Directed Acyclic Graph
SVM	-	Support Vector Machine
UML	-	Unsupervised Machine Learning
DBSCAN	-	Density-Based Spatial Clustering of Applications with Noise
ReLU	-	Rectified Linear unit
ATC	-	Air traffic Control
RADAR	-	Radio Detection And Ranging
IFF	-	Identification Firend or Foe
EW	-	Early Warning
CF	-	Carrier Frequency
USNRL	-	United States Naval Research Center
PSR	-	Primary Surveillance Radar
SSR	-	Secondary Surveillance Radar
SIF	-	Selective Identification Feature
UAV	-	Unmanned Aerial Vehicle
FRUIT	-	False Reply Un-synchronized in Time
UF	-	Uplink Format
DF	-	Downlink Format
BPSK	-	Binary Phase Shift Keying
IC	-	Interrogation Code
ACAS	-	Aircraft Collision Avoidance System
ADS-B	-	Automatic Dependent Surveillance Broadcast
VTS	-	Vehicle Tracking System
ES	-	Extended Squitter
MLAT	-	Multilateration
TOA	-	Time Of Arrival
TDOA	-	Time Difference Of Arrival
PDOP	-	Positional Dilution Of Precision
MLP	-	Multi-Layer Perceptron
MSE	-	Mean Square Error
CNN	-	Convolutional Neural Network
ROC	-	Receiver Operating Characteristic Curve

Abstract

Machine learning is considered to be the direction of current and future ATM systems development. Providing new approaches to challenges presented by the ever-increasing demands on safety and efficiency, it represents a powerful tool for solving problems across the ATM industry. The core of any ATM system are aircraft tracking algorithms that receive large amounts of surveillance data and apply complex computations in order to provide precise tracking data. Track filtering poses a large challenge when the computational power needed is considered. With the goal of optimizing the computations required for effective tracking, an Artificial Neural Network tasked with track classification based on the tracked aircraft dynamics is presented. The purpose of the ANN is to classify tracks in correlation with the behaviour of the tracked aircraft, in order to allow for dynamic filtering algorithm designation. The ANN is trained and tested using ADS-B data and evaluated by defined performance indicators.

Keywords: Machine Learning, ANN, MLP, CNN, Classification

Supervisor: Ing. Petr Lukeš

Abstrakt

Strojové učení je považováno za směr dnešního a budoucího vývoje ATM systémů. Strojové učení představuje nové přístupy k řešení problémů spojených s narůstajícími nároky na bezpečnost a efektivitu, díky čemuž se jedná o kompetentní nástroj s možnou implementací napříč oblastí ATM. Jádrem každého ATM systémů jsou algoritmy určené k sledování polohy letadel. Tyto algoritmy přijímají velké množství dat ze sledovacích systémů, a aplikují na ně komplexní výpočty s cílem efektivního a přesného sledování. Filtrace sledování v čase klade velké požadavky na výpočetní výkon. S cílem umožnit optimalizaci potřebného výpočtu pro filtrování sledování, je navržená neuronová síť, která má za úkol klasifikaci sledovaných letadel na základě dynamiky jejich pohybu. Účelem neuronové sítě je klasifikovat sledovaná letadla v korelaci jejich dynamikou letu, což umožní implementaci systémů s dynamickým přiřazováním různých filtrů v závislosti na klasifikaci. Navržená neuronová síť je trénována a testována pomocí ADS-B dat a je dále hodnocena pomocí definovaných ukazatelů výkonnosti.

Klíčová slova: Strojové učení, ANN, MLP, CNN, Klasifikace

Vedoucí práce: Ing. Petr Lukeš

Contents

1 INTRODUCTION

Aircraft tracking systems employ track filtering algorithms to improve the accuracy and integrity of the surveillance systems. These filtering algorithms require significant computational power, which only increases with the number of tracked aircraft in the surveyed area. Even with setbacks in recent years, which had negative effects on the Air transport industry, the trend of air traffic volume keeps rising. This poses a challenge for the ATM systems not only in the need for accurate and reliable ATM systems but also increases demands on the processing capacity of such systems. Track filtering is dependent on the dynamic model of the system (aircraft). There are different track filtering methods, and each is suitable for a different dynamic model.

Machine learning is the current trend of research and development in almost any technical field. There are numerous benefits of implementing machine learning algorithms into ATM systems at any level. This research explores the implementation of Artificial Neural Networks into ATM systems. The aim of this research is to propose an Artificial Neural Network tasked with track classification with the correlation to the tracked aircraft flight dynamics so that different filter algorithms can be dynamically assigned to such classified tracks. This is expected to optimize the computational power needed since simpler, less demanding track filtering algorithms can be used for tracks with predictable flight dynamics.

ADS-B data is used for Artificial Neural Network training and testing. A track classification ANN algorithm is presented with the goal of separating different tracks based on the tracked aircraft flight dynamics.

This thesis makes an effort in describing the environment for which it is valuable. The theory behind Machine learning is provided, as well as different air traffic surveillance systems that can be considered as a data source for training, or as an input for the already trained Artificial Neural Network. A variety of track filtering algorithms are closely examined providing a deeper understanding of the research's goal.

The methodology of ADS-B data pre-processing and classification is described, as well as the development of the ANN. Furthermore, the trained Artificial Network is evaluated using defined performance indicators, and the results are further discussed.

2 MACHINE LEARNING

Machine Learning algorithms fall under the Artificial Intelligence *"umbrella"* of computer science. The term Machine Learning was first used in 1959 by Mr Arthur Samuel, a graduated electrical engineer from the College of Emporia in Kansas. At the time, Arthur worked at IBM and later was behind one of the first working self-learning computer programs.

In the 1960s, a machine by the name of "Cybertron" was developed by Raytheon to analyze sonar signals, electrocardiograms and speech. This machine was trained by a human instructor to recognize patterns in collected data sets.

Today, Machine Learning has several tasks, ranging from data sets classification based on certain models that have already been developed, future outcomes prediction (regression) based on such models, data clustering based on the similarity of data in a dataset, reduction of data set dimensionality to help with "big data" processing, and many more. Machine Learning is one of (if not the) fastest-growing subsets of computer science and is used to support many applications that we use every day from translation software, and personalized advertising to purchase recommendations. Throughout the years of research and development, there are two main approaches to Machine Learning. *Supervised* and *Unsupervised*. Both are closely examined in the following chapters. Besides those two main approaches, there are several other such as semi-supervised learning or reinforcement learning. [1]

2.1 SUPERVISED MACHINE LEARNING

Supervised Machine Learning (SML) utilizes labelled datasets to train algorithms in such a way, that the output from them is classified data, or accurately predicted outcomes. Labelled data is fed through the algorithm and its influence on the outcome is controlled by a variable called weight. This weight and several additional variables, (like bias for example), can be adjusted until the returned data fits the desired output. There are two types of problems that can be tackled by SML. Classification problems, where input data is classified into determined categories. An example of such Classification could be separating spam from the inbox of an email client. Regression is used to help understand the relationship between independent and dependent variables. Such regression algorithms are very capable of predicting values given different data points. An example of a regression application can be a prediction of sales for a business. SML can be conducted using several algorithms, some of which are explained in the following chapter. [2]

2.1.1 Linear Classifiers

Linear Classifiers group items with similar *features* into distinctive categories. Features are some qualitative metrics that can be used to represent the data (for example age, height, etc..) This grouping is achieved by a decision based on the linear combination of said features. Linear Classifiers are practically applied to tasks such as document classification, email spam separation, or any other application where there is a large number of variables (features) supplied to the model. Linear Classifiers have an advantage when compared to other techniques thanks to their speed of classification when properly set up. That makes them suitable for scenarios where there

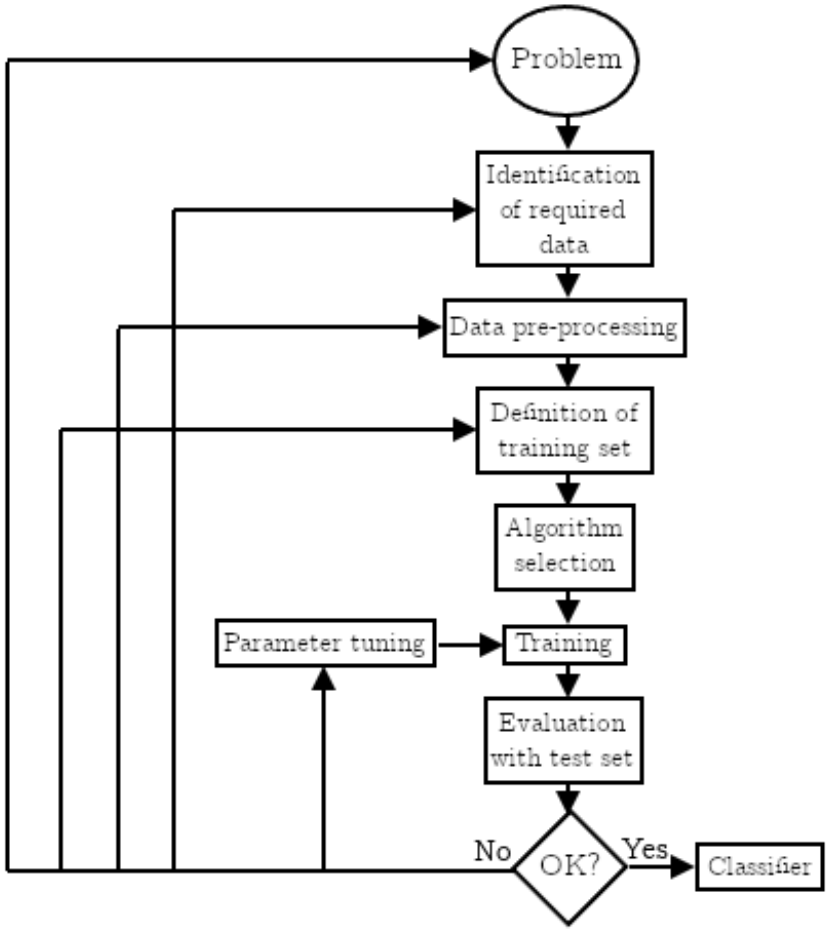

Figure 1: A flow chart displaying the process of Supervised Learning algorithms. Crucial part is the feed-back loop used to adjust the processing of the algorithm. Flowchart created by the author based on [1].

are strict time requirements. Linear Classifiers fall under the *Discriminative* classifiers umbrella. Discriminative classifiers are different from *Generative* classifiers in their underlying principle. Discriminative simply divides the data space so that each data group is occupying one part of the split data space. Generative classifiers define boundaries around each class in the data space. One example to represent the application of linear classifiers is classifying two groups based on their colour. For datasets where the colour greatly affects their distribution within the space, meaning that the position of the data is dependent on the colour of the data point, and different colours map data points far from each other, linear regression works quite well if applied correctly, as can be seen in the first example in figure 2. Even with a good dataset, poorly chosen linear regression does not perform well when classifying data. This can be seen in the second example. 3. For data sets where the colour does not affect the distribution of data points n the data space, linear regression will not perform as well as other classification algorithms. [3] [4].

2.1.2 Logistic Regression

Logistic Regression is used to predict the probability of an output (label). There are three types of Logistic Regression. *Binomial* Logistic Regression considers only two possible types of output

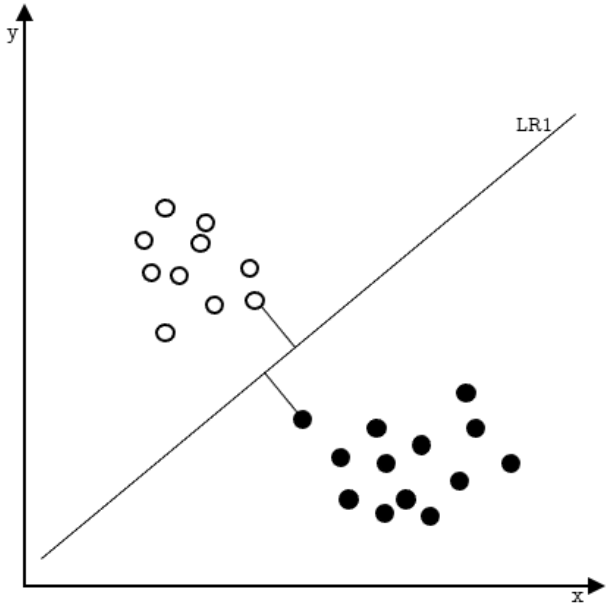

Figure 2: Example of well executed linear classifier splitting two groups effectively. Image created by the author based on [5].

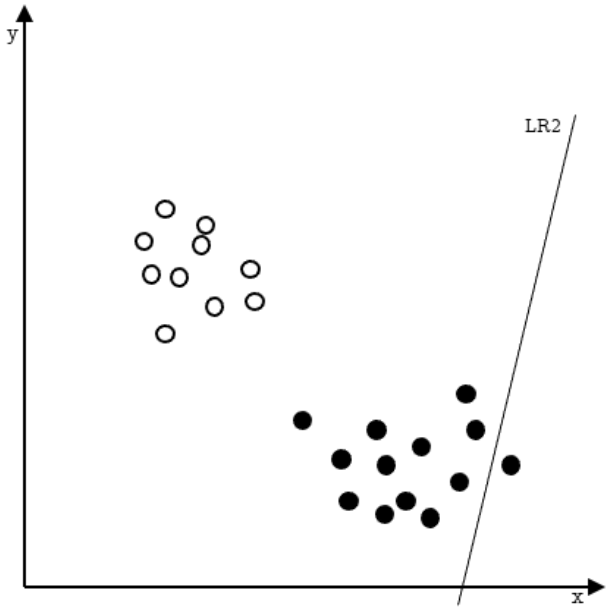

Figure 3: Example of poorly executed linear classifier not separating the two groups accurately. Image created by the author based on [5].

(yes/no, 0,1). *Multinomial* Logistic Regression considers more than 2 possible unordered types (fork, knife, spoon). *Ordinal* Logistic Regression allows for more than two ordered types (hot, warm, cold). Logistic Regression is considered a Linear Classification model, making its naming quite misleading. Logistic Regression states the boundary between two possible groups and outputs the probability of classified data being a part of a group, based on the distance of the group from the established boundary. Some use cases utilizing the Logistic Regression can be predicting the possibility of illness occurring within a population, or analyzing the possibility of fraud when an individual files an insurance claim. Like Linear Classifiers, Logistic Regression is

one of the applications of *Discriminative* classifiers. [6] [7].

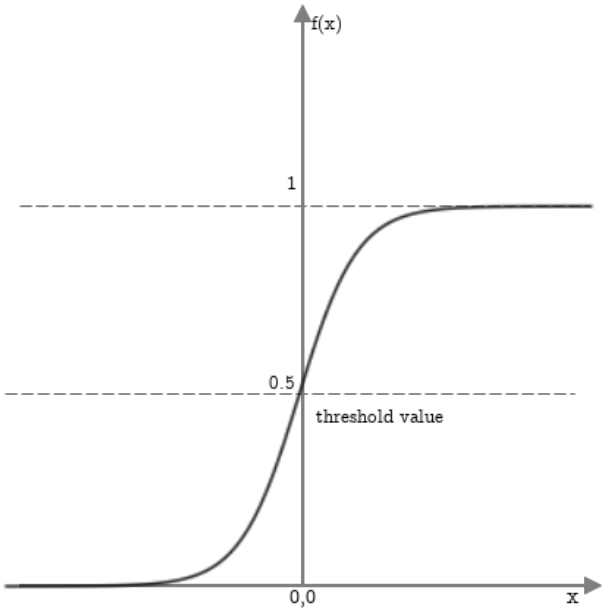

Figure 4: Logistic Regression application example. Binary classification of data with a threshold value. Closer the data is to the threshold value, the lower is the certainty in the output state. Image created by the author based on [8].

2.1.3 Bayesian Networks

Bayesian Networks are used as a probabilistic graphical model. Such model represents a set of variables and their dependencies using a directed acyclic graph (DAG). Bayesian Networks are suitable for predicting the likelihood of a possible cause being the contributing factor to a specific event that has occurred. A use case for a Bayesian Network can be the representation of a probabilistic relationship between the symptoms, and the disease they could be a contributing factor. Let's imagine a scenario where a person wants to start the engine of a car, but the engine won´t start. Observing this malfunction, we can construct a simplified acyclic graph of the causes and the observed evidence. In this example there is no edge between the causes, however, it is not impossible to have two causes that share an edge. Once the causes are given a probability of occurrence, it is possible to predict which cause is responsible for the observed outcome. [9].

2.1.4 Naive Bayesian Networks

Naive Bayesian Networks are nothing more than simplified regular Bayesian Networks. Such networks are composed of directed acylic (not forming circles) graphs (DAG) with just one "parent", and several "children". What makes such networks naive is the assumption of independence amongst the "children". Naive Bayesian Networks are considered less accurate on their own when compared to other Machine Learning algorithms. Some applications showed however that when combined with kernel density estimation, for some cases they provide high accuracy levels. The upside of the network's simplicity is its scalability. There are again several types of Naive Bayesian Networks. Multinomial Naive Bayesian Networks, where there are several possible output classes. An example of such classification can be if a specific car belongs to a car class (sports car, SUV, saloon, hatchback etc..). Bernoulli Naive Bayesian Networks are similar to

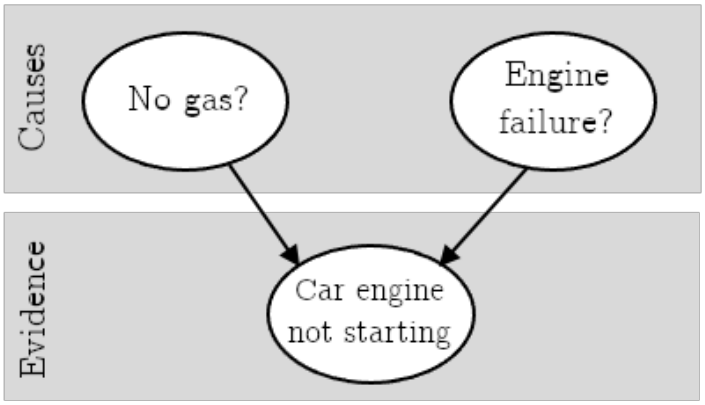

Figure 5: Bayesian Network example of a event, in this case not starting the engine when turning the ignition key in a vehicle, and two, non-correlated possible causes. Image created by the author based on [9].

Multinomial, however, the variables used to predict the output are Boolean, meaning only yes or no values. For example, if a specific word can be found in a song. Gaussian Naive Bayesian Networks are used when the variables used to predict the output are not discrete, and it is assumed that they are forming and Gaussian distribution. [10] [11]

2.1.5 Support Vector Machines

SVMs are said to be one of the most robust classification and prediction methods. SVMs have to be provided with a set of training examples already labelled as being a part of a specific category. Then, the algorithm constructs a model that can assign new examples to given categories for which it was trained on. That makes it a non-probabilistic binary classifier, however, there are some ways to apply SVMs in probabilistic classification tasks as well. SVM maps the training dataset as points in space, with the goal of making the gaps between the categories as large as possible. This is where the name comes from since the data on the "boundary layer" are called *support vectors*. With new data, the algorithm again maps the corresponding point into already created space and classifies them as belonging to one of the classes previously established with the training dataset. SVMs are not only capable of linear classification, but when utilizing the "kernel trick" of avoiding the explicit mapping that is needed to get linear algorithms to learn non-linear functions, they can be used for non-linear classification as well. Support Vector Machines allow for a wide variety of use cases such as image classification, writing recognition or synthetic-aperture radar data grouping (stitching). [12] [13]

2.1.6 Decision Trees

Decision Trees can be used as a classification algorithm assuming the outcomes are only discrete values. For such Decision Trees, each node represents a feature to be classified, and each branch represents a value that the node can assume. Decision Trees allow the combination of numerical values with binary values. Decision Trees also propose an efficient way to optimize the algorithm. The so-called *"pruning"* technique uses a validation set to remove nodes and assign the most common instance instead. For purposes where the outcomes are not strictly discrete, but can assume value described as any real number, Regression Trees are used instead. Decision Trees

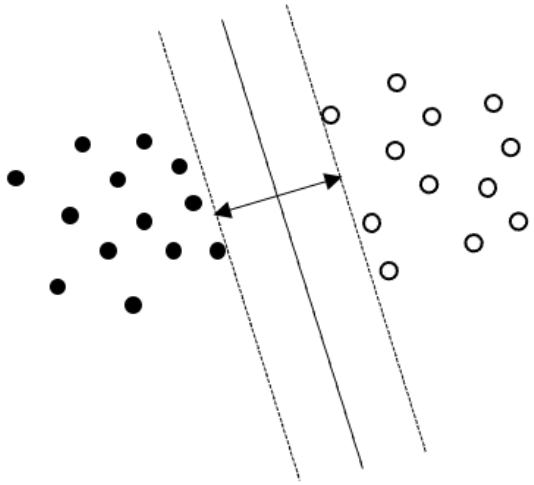

Figure 6: Example of a SVM with large margins in data classification thanks to utilizing support vectors. Image created by the author based on [14]

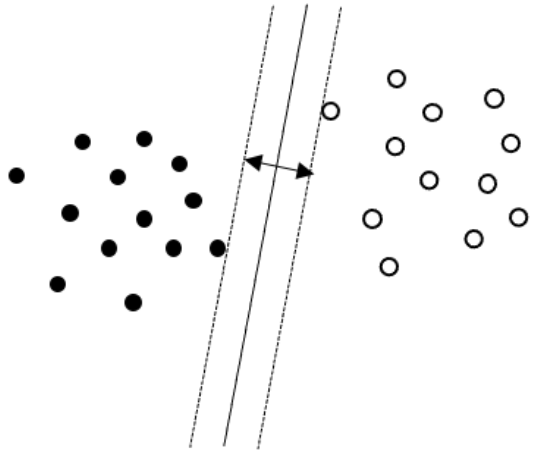

Figure 7: Example of a classification not utilizing support vectors, having significantly smaller margins.Image created by the author based on [14]

are easy to design, read and interpret. However for real word applications, there is one downside preventing it from being the go-to algorithm, and that is its inaccuracy. To combat that, Random Forests are examined below. [15] [16]

2.1.7 Random Forrest

Building on the knowledge of Decision Trees, understanding Random Forests is quite straightforward. Random Forests combine the simplicity of Decision Trees, and flexibility to combat inaccuracy. The basic principle of Random Forest lies in employing several Decision Trees

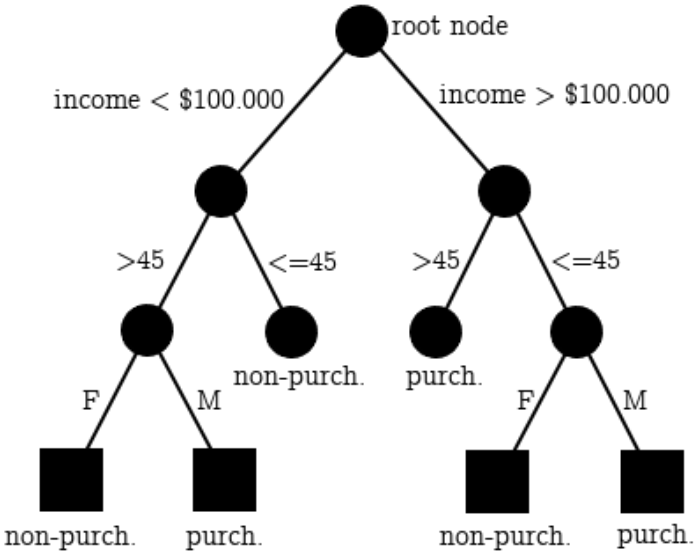

Figure 8: Simple Decision Tree algorithm example, in which it is trying to predict the possibility of a expensive vehicle sale to a specified group of people. Image created by the author based on [17].

provided with the same data set and then comparing the result with each other. The class that is most represented as the output of single Decision Trees becomes the one, that is outputted as the decided class by the Random Forest algorithm. In the case of Regression Trees, the average of the output is selected as the decided output of the whole algorithm. One important aspect of Random Forests performing accurately is the Decision Trees not being correlated with each other. One method of de-correlating Decision Trees is called *"bootstrapping"*. Bootstrapping is the procedure of assigning a different subset of the data to each Decision Tree. This ensures that each Decision Tree will perform differently since it is working with different input data. Another way to ensure the de-correlation of Decision Trees is called "Feature Randomness". This method assigns different sets of features to different Decision Trees. [18] [19]

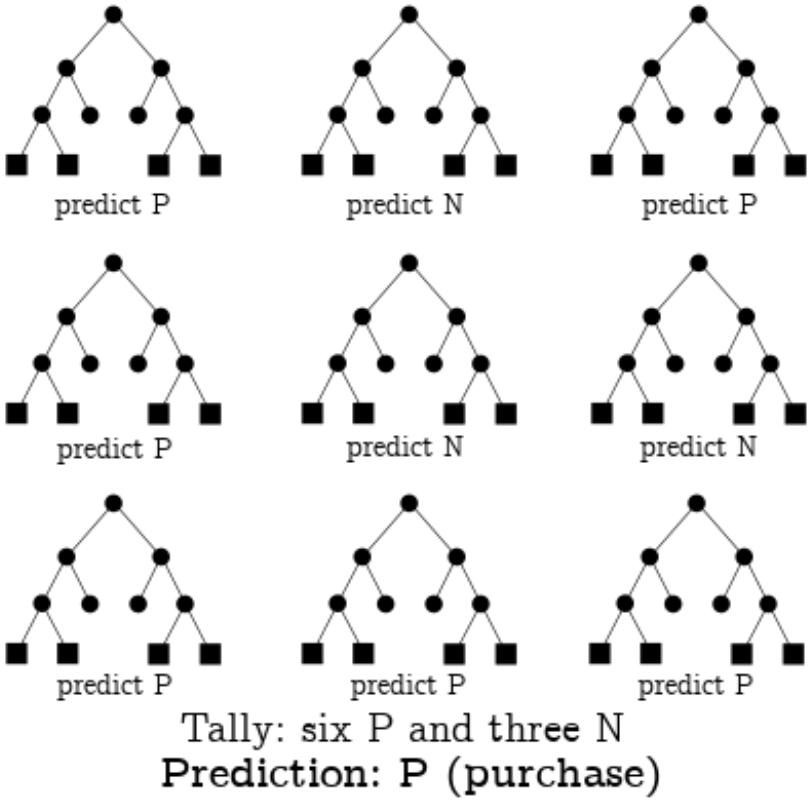

Figure 9: Simple example of a Random Forest algorithm. The prediction that is most represented in the entire forest is regarded as the output from the Random Forest. Image created by the author based on [18].

2.2 UN-SUPERVISED MACHINE LEARNING

Unsupervised Machine Learning (UML) differentiates itself from SML by its ability to process unlabeled data. Such algorithms are described as being able to detect patterns in data sets, that are otherwise hidden from the user's point of view. There are many problems that can be solved using UML algorithms, some of which are discussed in this thesis. Clustering means grouping data that are not labelled into groups by their similarity. Clustering is strong with tasks such as image compression, or market segmentation for optimized marketing. For scenarios where it is needed to find the relationships between a data set and variables in that data set, association algorithms are employed. This method of UML can be used to create recommendation engines for potential buyers based on their purchase history. Dimensionality reduction is a technique that is widely used in data pre-processing scenarios. Such UML techniques reduce the number of inputs while preserving the quality of the whole data set. [20] [21]

2.2.1 K-Means Clustering

K-Means Clustering can be also referred to as a *Vector Quantization*. Vector Quantization is a technique originally used in signal processing to allow the modelling of the probability density function of prototype vectors. This technique served for data compression with the goal of data transfer optimization. It divides a large set of points represented as vectors into groups based on their proximity to each other and ensures that each group has approximately the same number of points. A *centroid* point then represents each group. A data space divided by such an algorithm forms several so-called *Voronoi* cells. A Voronoi cell is created by a partition of a plane divided by the proximity of the points in that space to each other. A Voronoi cell contains all points, that are closest to the seed of the specific cell, than to any other seed of the Voronoi cell in that shared space. Knowing this, the K-Mean method can be used to group points from a data set into a "K" number of clusters. The data clustered is unlabeled, making the K-Means Clustering a UML algorithm. There are several steps needed to apply K-Means Clustering to data. The first step is to select the *K-value*. As already stated, the K-value represents the number of clusters to be created. The next step is initializing a centroid - selecting a random data point as the centroid (seed). The third step is assigning each data point to its closest centroid, which will create the clusters. The fourth step is calculating the variance, and placing a new centroid in each cluster. Once the new centroid is established, the third step repeats and some data points will be reassigned to their new cluster based on the distance from their new closest centroid. This process is finished once none of the data points changes its centroid after the variance correction. K-Means Clustering is one of the most widespread uses of a UML algorithm.

Let's imagine a scenario, where a real estate agency just introduced two housing developments to the market. One is a luxury development in the centre of the city, and the second is an affordable housing complex on the outskirts. The agency knows that in order to optimize the advertisement cost, it needs to advertise each project to a different potential buyer class. the model they develop clusters targets the advertisement audience into three groups (3 clusters). Those who are unlikely to buy into any of those developments, those, who are more likely to purchase the more affordable apartments, and those who are after the luxury apartments. Two key pieces of information to cluster buyers are their current living location - if the person already

lives in the centre, he is more likely to buy the more expensive apartment, and second is their income - the more they earn, the more likely they will be to spend on the luxury apartments. Like this, the agency can divide the potential buyers into three clusters, and each advertises a different project in order to make the marketing as efficient as possible. [22] [23]

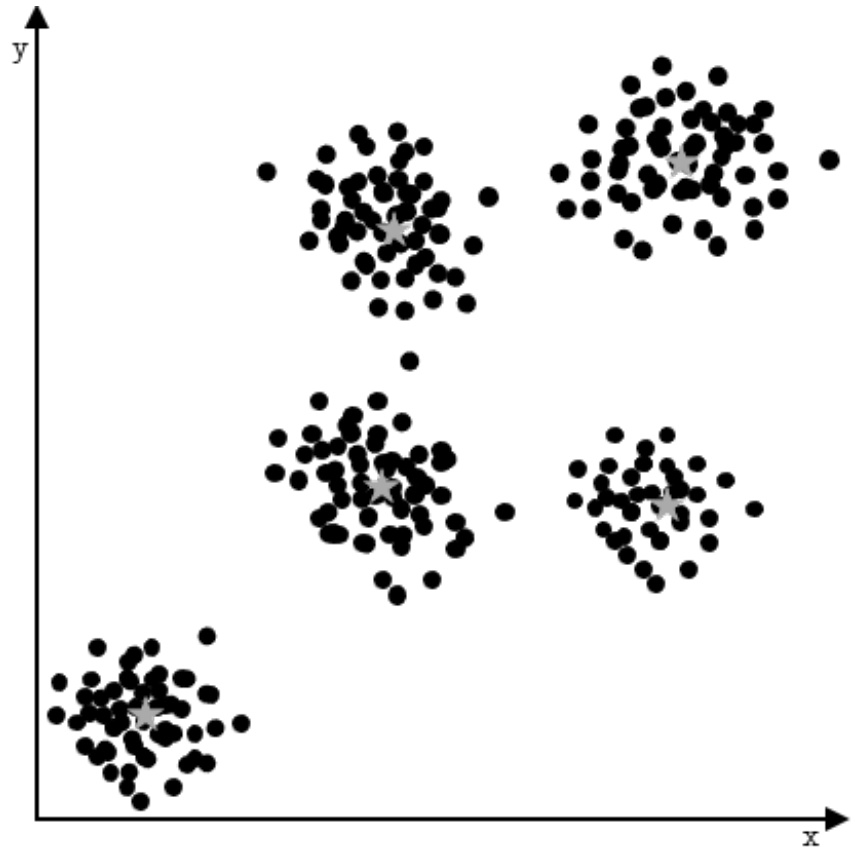

Figure 10: Example showing the several clusters of data with each assigned to its *centroid* point. Image created by the author based on [24].

2.2.2 Hierarchical Clustering

This method builds a hierarchy of clusters. There are two approaches to building said hierarchy. The agglomerative "bottom-up" approach is where the observation starts in the singular clusters, and when moving up the hierarchy, corresponding clusters merge together. At the start, each data point is considered its own cluster, and with each iteration, similar clusters are merged, until a one or predetermined number of clusters are established. The steps of Agglomerative Hierarchical Clustering are computing the proximity matrix of all data points, then establishing each data point as its own cluster, merging the selected number of closest clusters (at this point still data points), updating the proximity matrix, and again merge the select number of closest cluster. Repeat until the desired number of clusters is formed.

The divisive "top-down" method starts in the "top cluster and then splits as the observer moves down the hierarchy of clusters. This method is basically the Agglomerative method but turned the other way around. Instead of starting at individual data points, all the data points are considered to be a part of a single cluster, and then the ones that are least similar to the rest (outliers) are removed from the cluster.[**"Cite –atlolla˙understanding˙2020″**]

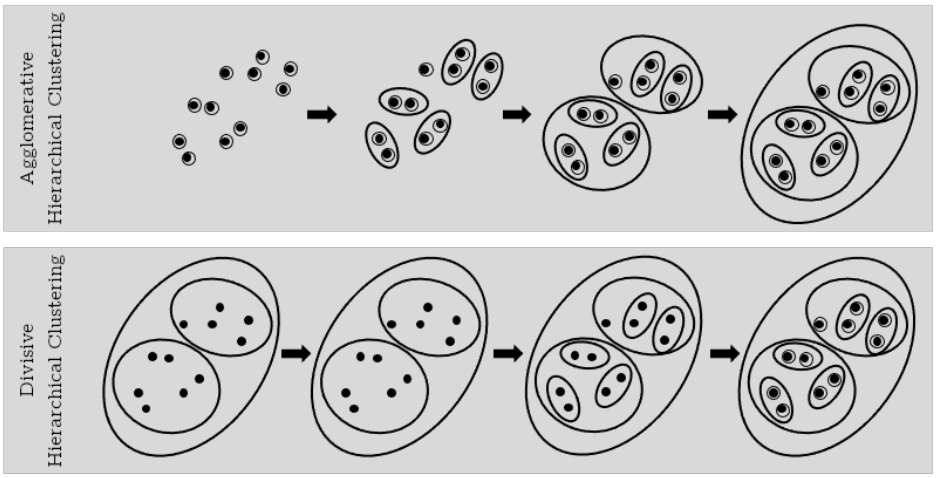

Figure 11: Example showing the difference between Agglomerative hierarchical Clustering, and Divisive hierarchical Clustering. Image created by the author based on [25].

Now the question of how the algorithm compares clusters among each other to determine which ones are the most similar, and which they should group together, or split away. There are several ways into deciding just that. Single-linkage algorithm ("MIN" method) is defined as the minimum similarity between two points, each belonging to a separate cluster, which is equal to the similarity of involved clusters. In other words, the clusters are defined as similar to each other based on the similarity of the closest points in each cluster. This approach works well as long as there is minimal to no noise in the data points.[26]

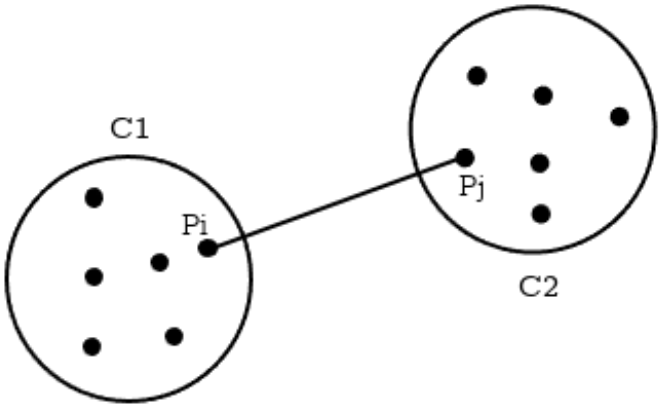

Figure 12: Example of Hierarchical Clustering employing the "MIN" method. Clusters are considered similar based on the distance of the closest data points from each cluster. Image created by the author based on [26].

For such cases, it is advisable to utilize the so-called Complete-linkage algorithm ("MAX" method). This algorithm defines similar clusters based on the maximum similarity between two points from each cluster. To simplify again, two clusters are considered similar, based on two points (each from a different cluster) that are furthest apart. Even though this approach solves the problem with noisy data points, it has issues when the size of clusters is not approximately the same.

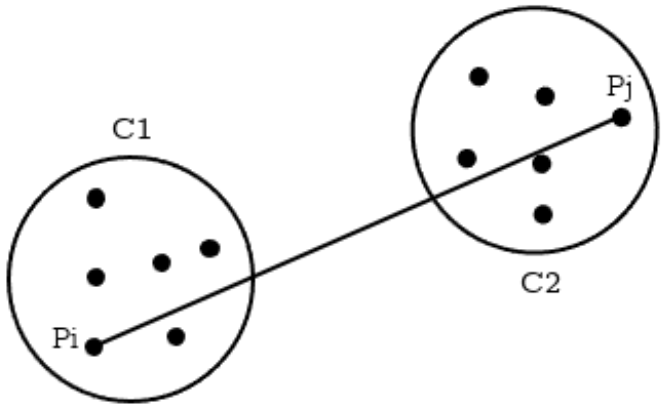

Figure 13: Example of Hierarchical Clustering employing the "MAX" method. Clusters are considered similar based on the distance of the furthest data points from each cluster. Image created by the author based on [26].

Group averaging is another approach that instead of comparing two single points from each cluster, averages all the similarities between points from each cluster. This approach again does well with data sets where there is noise present in the data space, however, it tends to be biased towards global clusters. Also, the amount of computation needed for large data sets makes it less feasible in real-life applications. [26]

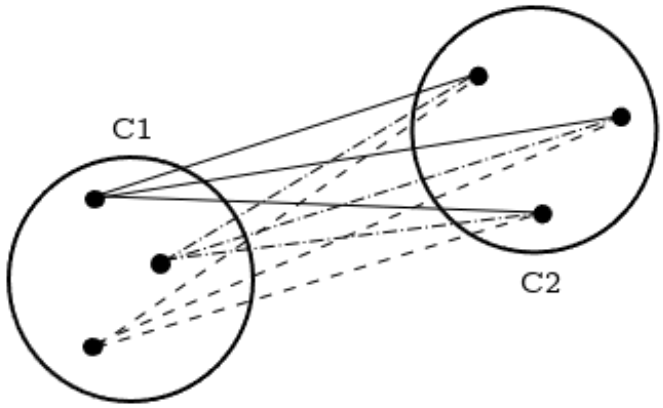

Figure 14: Example of Hierarchical Clustering employing the Group averaging method. Clusters are considered as similar based on the average distance between points from each cluster.Image created by the author based on [26].

The methods explained above are the most common ones, and the results that are returned by them meet the requirements set for the algorithm, however, there are several other methods that can be used in special cases such as calculating and comparing the distances between centroids of different clusters. Lastly, Ward´s method is very similar to the group averaging method, and it carries the same advantages and disadvantages, the difference is that instead of comparing

averages, Warg´s method calculates the sum of the squared distances between points from each cluster. [26]

Overall, hierarchical Clustering has an advantage over other clustering methods in that it not only returns clusters themselves but also the hierarchy in which they were constructed, meaning it provides more information about the relationship between the clusters. On the other hand, when processing large data sets it tends to be quite slow when compared to other clustering methods, due to the additional steps needed to merge/divide clusters. Hierarchical Clustering does not perform the best when working with data sets with a significant amount of noise, or with data sets containing outliers. [27] [28] [26]

2.2.3 Mean-Shift Clustering

Mean-Shift is an algorithm used for UML clustering, although it does have other applications as well. Today, it is widely used for data analysis, such as image or voice segmentation. Mean-Shift is an iterative algorithm, where with each iteration each data point is "shifted" to the "mean" of the region. In the end, the final destination of those data points is the cluster they belong to.

The steps of the mean shift algorithm are as follows. First, it is important to set what is the data point's local range of influence i.e. the *bandwidth*. In other words how far away within the whole data space does a selected point calculate the local mean? Once the range is set, each point calculates the mean of the local area that is in the range (that means that for each point, a different mean will be calculated). In the next step, the point moves from its original position to the position of the calculated mean for the given local area. This two-step process iterates until the point does not move to a new position, meaning it is finally located at the mean of the cluster it belongs to, i.e. the centroid.

The algorithm as presented here would need to conduct an unreasonable amount of calculation if processing the introduced steps for each point in the data set. To streamline the process, there is a technique applied that allows for ignoring calculations of means of the bandwidth with fewer data points when two bandwidths overlap. Since the bandwidth is the only parameter used for this method, it is crucial to set it appropriately to the data set examined. Too little bandwidth could result in false clusters when working with noisy data, and bandwidth too large could result in the grouping of two separate clusters. [29] [30]

2.2.4 Density-Based Clustering

A cluster, under the Density-Based Clustering methodology, is defined as an area of higher density of data points, when compared to the rest of the data set. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) introduced in 1996 by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu, is the most popular implementation of the Density-Based Clustering method. It uses a specific number to distinguish clusters from outliers i.e. noise. Even though it is the fastest Density-Based Clustering method, it is not always the best option for some particular cases. Since the distance used for the separation is fixed, it could potentially not be sufficient if used for data sets where all the clusters are similarly dense.

Two parameters are needed for the DBSCAN to create clusters, and decide if and to which a

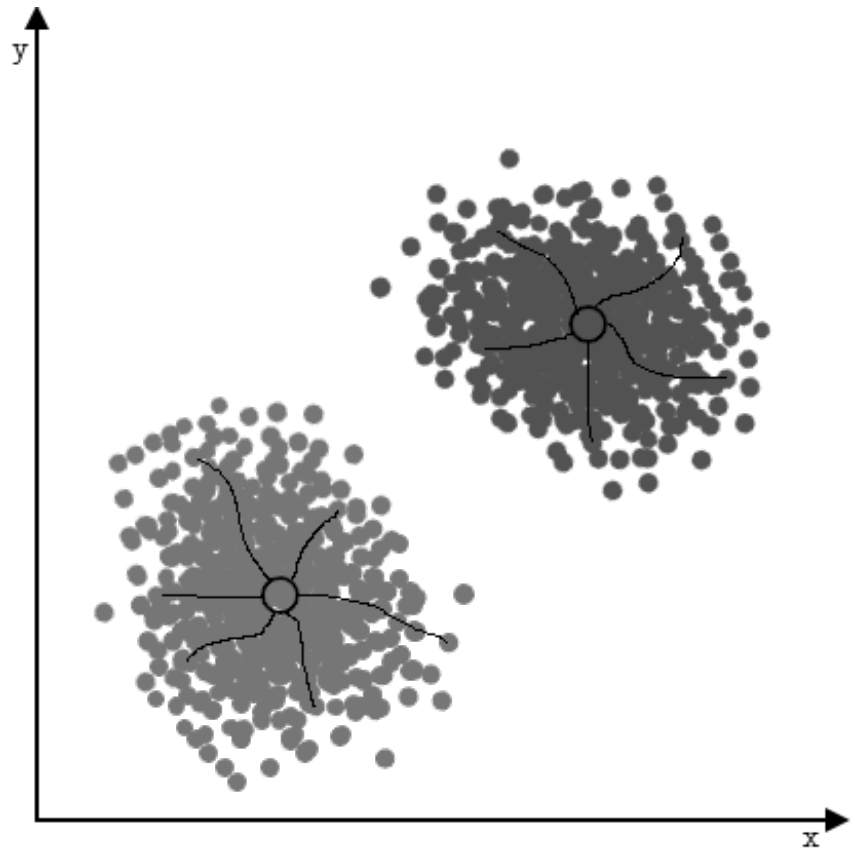

Figure 15: Graphical example of the Mean-Shift clustering algorithm. Each points "shifts" to the local "centroid" of the cluster. Image created by the author based on [30].

data point belongs. ϵ defines the radius of the neighbourhood, and the **m** defines the minimum number of data points in the neighbourhood. Data points that are not part of the density-defined cluster are considered outliers or noise. Points, that are at the centre of a cluster that passes both conditions of the minimum number of points, within the defined radius defining the neighbourhood are considered core points. Data points that are a part of a cluster that passes both conditions, however, the cluster they are at the centre of do not are considered border points.

Point is considered as "density-reachable" by another point in the data space, if it is possible to connect the two points using only core points, that are each in the neighbourhood of the previous one, with the possible exception of the destination i.e. reached point. Also, a point is considered "directly density-reachable", by a point, if it is located inside the epsilon-defined neighbourhood of the core point.

The first step of DBSCAN is selecting a random point from the data set that has not been yet assigned to any cluster or marked as an outlier. By applying the two conditions of the ϵ and the minimum number of points in the neighbourhood, the algorithm decides if it is a core point. If it is not, the point is marked as an outlier (this does not yet mean that it will end up as an outlier at the end.) The next step is to add all the points that are directly density-reachable by the core point to its cluster. Next, "jump" to all the points that are "density-reachable", and assign them to the forming cluster. If the point that is at this moment reached has been previously marked

as an outlier, it is re-marked as a border point. Now, a cluster is formed, and this process is repeated until all the points are assigned to a cluster, or considered as an outlier.

There are other Density-Based Clustering algorithms, such as the Self-Adjusting HDBSCAN that utilize several distances to cover such data sets, where the clusters have different densities. HDBSCAN requires the least amount of user input since it is the most data-driven algorithm. Multi-Scale OPTICS creates a so-called "reachability plot", that is used to separate clusters with different densities from outliers. [31] [32] [33]

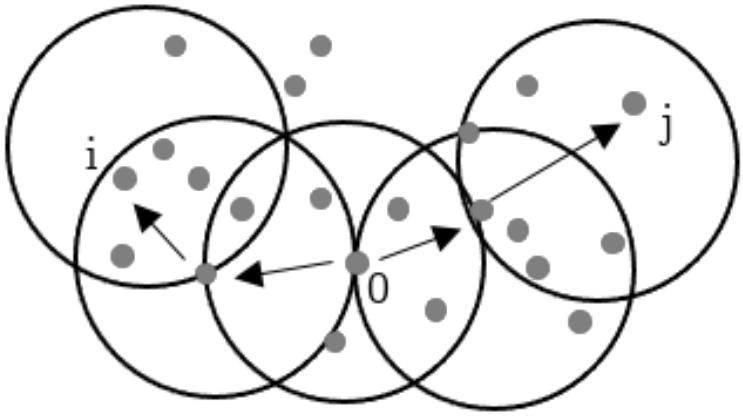

Figure 16: Example of points that are "density reachable" by a DBSCAN clustering algorithm. Image created by author based on [32].

2.3 ARTIFICIAL NEURAL NETWORKS

Neural Networks, also referred to as Artificial Neural Networks (ANNs), are one of, if not the most widely recognized and examined executions of Machine Learning principles. The origins could be traced back to the mid-1940s, when researchers like Warren McCulloch, Walter Pitts, and later D.O.Hebb laid the groundwork for what we today call Artificial Neural Networks. Hebbian Learning is the mechanism described by the hypothesis based on the neural plasticity created by the already introduced D.O.Hebb. Later, in the mid-1950s, Wesley A. Clark was the first to use a "computer" to simulate such a Hebbian network. Frank Rosenblat, an American psychologist, invented in 1958 the "Perceptron", which earned him the widely accepted status as the father of Deep Learning. Perceptron is considered the first ANN. All other work in this field has been possible thanks to his breakthrough. Since ANN is the method utilized in this thesis, it will be examined closer, than the algorithms described in the previous chapters. [34]

2.3.1 Mechanisms of Artificial Neural Networks

Three layers are needed to form an ANN. The input layer is responsible for the information input from the outside world to the following Hidden layer. Input layer neurons are considered passive, since they don´t receive inputs from other neurons, but the data is provided by the user. The output layer is the last layer of the ANN and is responsible for producing outputs generated by the Hidden layers of the ANN. Output can be either one decided value (or based on how the neural network is used class etc.), or a set of values each presented with how confident the ANN is in each one of them. The hidden layer is topologically located between the Input and output layers and is in fact comprised of several layers of neurons.

Each layer is connected via channels (tensors), and each channel is assigned with a *weight*. The weight of a channel represents how much the value of the specific neuron connected by the channel influence the following neuron. Simply put a neuron that is connected via a channel with a low weight will not affect the following neuron as much, as a neuron that is connected via a channel with a higher weight.

Another parameter influencing the output of the ANN is called *Bias*. Bias is used to better fit the activation function of the Neural network to the data set that is used to train the ANN. Both weight and Bias are "trainable" values, meaning at the beginning of the training cycle, both are set at random, and are adjusted by each iteration of the training cycle so that the output the ANN returns is as close as possible to the desired one. The state of a neuron in the Hidden layer is either active, or inactive, decided by the activation function that is inputted the weighted sums of previous inputs, or it can be a discrete value. [34] [35] [36]

2.3.2 Activation Function

Activation Functions are used to decide if a neuron is to be activated or not, and if so then what should the output from said neuron be? There are several Activation Functions that are suitable and widely used for that purpose. Activation Functions can be categorized as Linear, or Non-Linear Activation Functions, Some of the functions that were encountered most frequently during the research for this thesis are examined in the following section. [38]

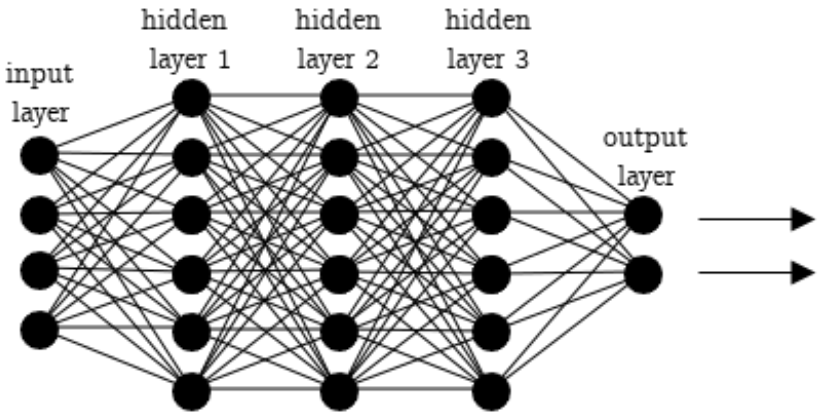

Figure 17: Topology of a general Artificial Neural Network showing Input layer, several Hidden layers and the Output layer. Image created by the author based on [36].

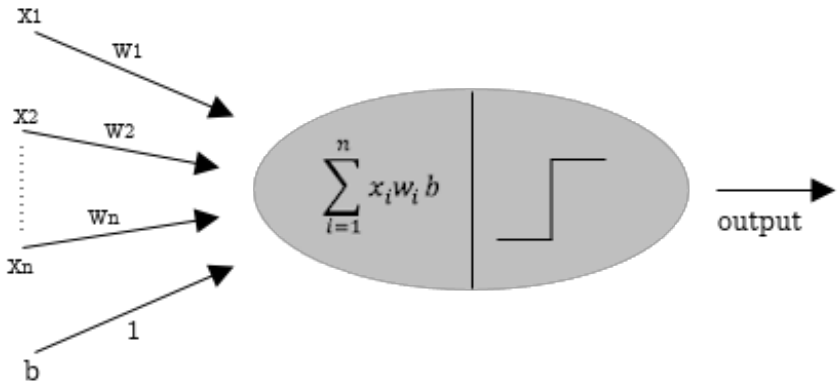

Figure 18: A disection of a ANN Neuron showing the channels with assigned weights, the sum of the weights and bias, and the activation function that sum is applied to. Image created by the author based on [37].

Step Activation Function

The step Activation Function simply decides if the value of the weighted sum is greater than a set threshold. If the value clears the threshold then the neuron is "activated". This binary state possibility however limits the use of the simple Step Activation Function, since it does not allow for multiple output classes. Another issue arises when considering back-propagation during the training of the ANN. That is because the back-propagation uses derivatives of Activation Functions, that in the case of a step function is equal to 0. The equations for the step activation function are shown in 1 and 2. [39]

$$f(x) = 1, \text{ if } x >= 0 \tag{1}$$

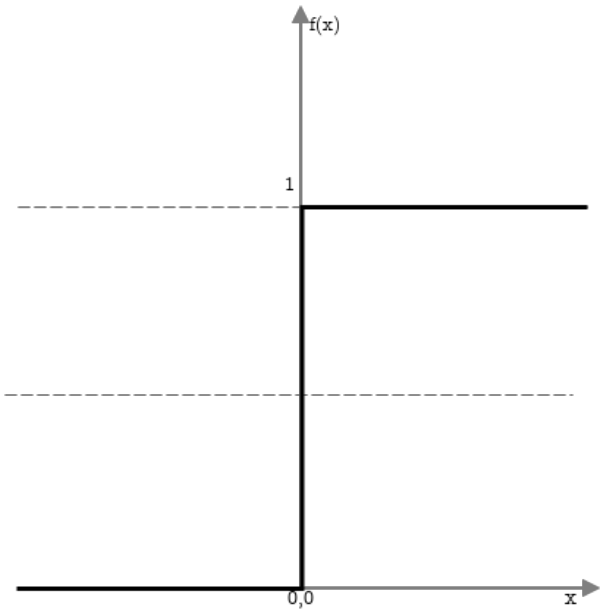

Figure 19: A Step Activation function with a binary output. Image created by the author based on [39].

$$f(x) = 0, \text{ if } x <= 0 \tag{2}$$

Where:
x is the input

Linear Activation Function

Linear Activation Function would allow for a range of possible output values, however, if there were only Linear Activation Functions used throughout the ANN, it would only be possible to use linear data sets as input. Another drawback is that in back-propagation, it remains constant. That means it does not change while learning. Possibly the worst characteristic of the Linear Activation Function is that the last layer of the neural network will always be a function of the first layer, no matter how deep (how many layers) the ANN is comprised of. It is a good practice to use Linear Activation Function in the output layer of such ANNs, that deal with regression problems. The equation for the linear activation function is shown in 3. [40]

$$f(x) = x \tag{3}$$

Where:
x is the input

Sigmoid Activation Function

The sigmoid Activation Function suits the needs of an output of predicted probability since it exists in its entirety between the values of 0 and 1. Sigmoid is non-linear, differentiable, and

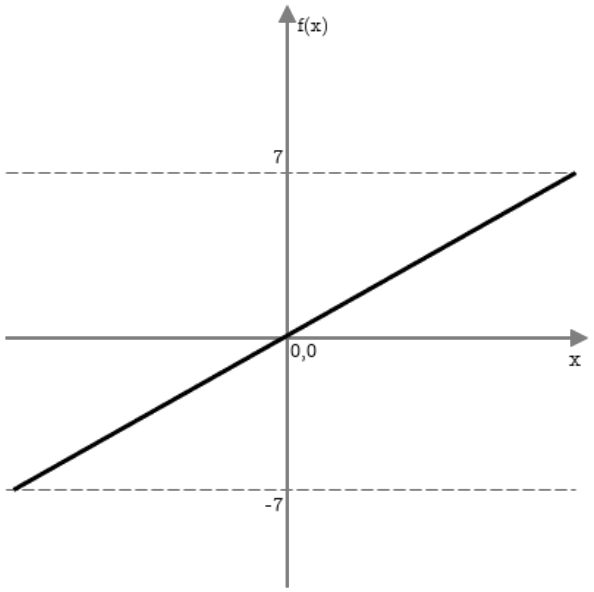

Figure 20: Linear Activation function with direct mapping of output. Image created by the author based on [41].

for multi-class classification is generalized into a "softmax function". This means, especially the non-linear characteristic, that the Sigmoid Activation Function is used for data that is not linearly separable. For a function to be used as an Activation Function, it has to be monotonically increasing, which Sigmoid Activation Function indeed is. The equation for the Sigmoid activation function is shown in 4. [42] [43]

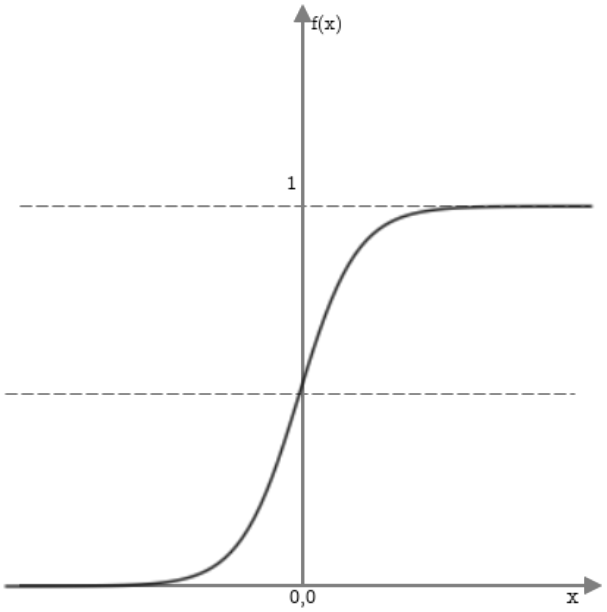

Figure 21: A Sigmoid Activation Function that compresses the data space into values between 0 and 1. Image created by the author based on [39].

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \tag{4}$$

Where:
x is the input

Hyperbolic Tangent Activation Function

Hyperbolic Tangent Activation Function is similar to the Sigmoid Activation Function in its shape, and in some of its properties such as it is also differentiable. The difference is that its values range from -1 to 1. That means that inputs that are negative are mapped as strongly negative. Hyperbolic Tangent Activation Function is mostly used for classification between two groups. The equation for the hyperbolic tangent activation function is shown in 5. [44] [41]

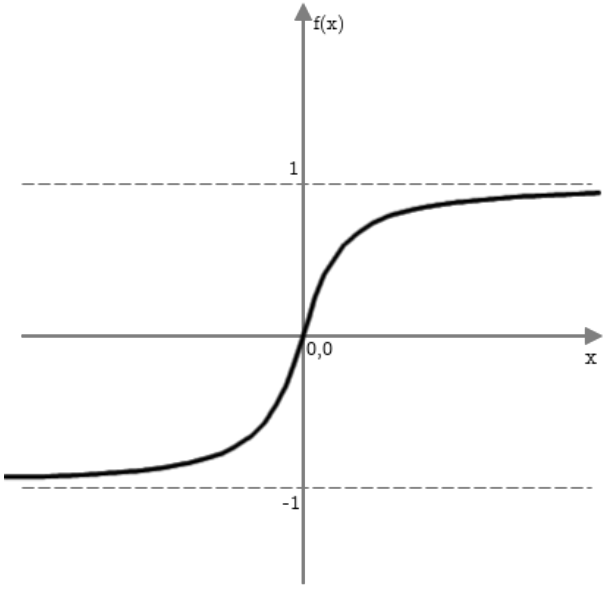

Figure 22: A Tangent Activation function allowing mapping data with negative values, as well as positive values. Image created by the author based on [41].

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{5}$$

Where:
x is the input

ReLU Activation Function

ReLU (Rectified Linear Unit) is the most common Activation Function used today. ReLU Activation Function is rectified at the bottom, meaning the function is equal to 0 when the input value is equal to or less than 0. With other values, it returns the value of the input directly. Its implementation is regarded as one of the easiest, and it does not suffer from the same issues that some of the previous Activation Functions do. Vanishing Gradient is one of the main issues that other Activation Functions suffer from, but ReLU fares well with it. ANNs that employ the ReLU activation function are sometimes labelled as "Rectified Networks". The fact that all negative values are returned as 0 could be potentially a problem for such data sets, where

the inputs do reach negative values. For such cases, the so-called "leaky" ReLU was defined to tackle them. The *leak* refers to the part of the function in the negative values, where it does not automatically plot them as 0. This allows the ReLU Activation Function to cover a large array of cases where the negative values have to be accounted for since its range is from - inf to +inf. The equation for the ReLU activation function is shown in 7. The equation for the Leaky ReLU is shown in 8. [45] [46]

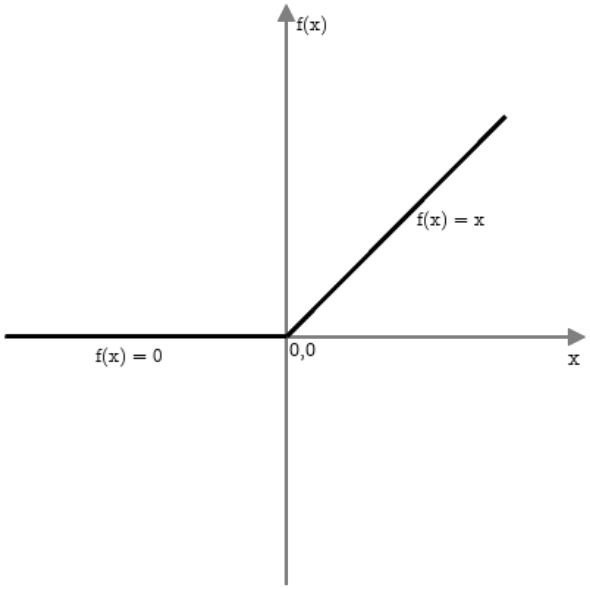

Figure 23: A ReLU Activation function that is rectified from the bottom. This is not ideal fort input data that assumes negative values. Image created by the author based on [39].

$$f(x) = max(0, x) \tag{6}$$

Where:
x is the input

$$f(x) = ax, x < 0 \tag{7}$$

$$f(x) = max(a * x, x) \tag{8}$$

Where:
x is the input
a is the "leaky" constant

There are a number of activation functions that can be used in some specific cases of particular data sets and needs of the user, however, they are not examined in this thesis since they are not related to the topic discussed.

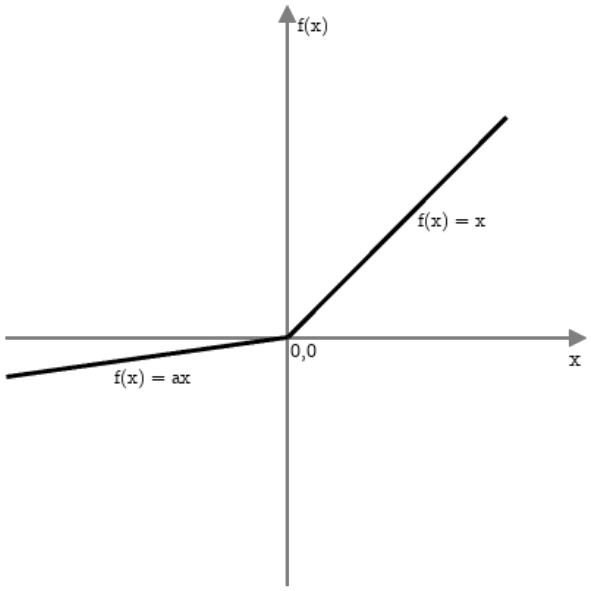

Figure 24: *Leaky* ReLU solves the shortcoming of ReLU by allowing datasest with negative inputs to be processed. Image created by the author based on [39].

2.3.3 Training the Neural Network

ANN training is an iterative process, in which with each iteration, the behaviour of the network in regard to processing input data changes. At the beginning of the training, the weights are initialized randomly. Since the weight affects the output of the ANN quite significantly. The output of such ANN with random weights will in fact be just as random. By adjusting the weights of the channels in the ANN, the training slowly with each iteration enables the network to achieve such outputs, that are increasingly closer to the desired outputs. The goal of the training is to minimize the "loss function". [47]

Loss Function

A Loss function is one of possible representations of the neural network's ability to process a specific task. There are several functions that can be used as a ANN loss function, some of which are Mean Square Error, Mean Absolute Error, Quantile Loss etc. The goal of training a Neural Network is to minimise the loss function. There are several algorithms used to do just that. Gradient Descent (or Stochaistic Gradient Descent) is the most commonly used algorithm for the Loss Function minimisation and is examined in the following section. [48] [49]

Gradient descent

A gradient can be interpreted as the slope of a surface. Descent means to move downwards. So put together, Gradient Descent means moving down a slope unlit reaching the lowest point on said surface. Algorithms based on Gradient Descent are iterative. The initialize at a random point of a function, and with each iteration travels down the function until it reaches the point, that is located at the bottom of the function. The math behind the algorithm will not be shown in this thesis, however, it can be mentioned that steps that are further away are large, and get smaller the closer it gets to the desired lowest point of the loss function. The step (distance) is

regarded as the "learning rate". As an example of the Gradient-based Optimization algorithms, a Stochastic Gradient Descent is examined closer. [50] [51]

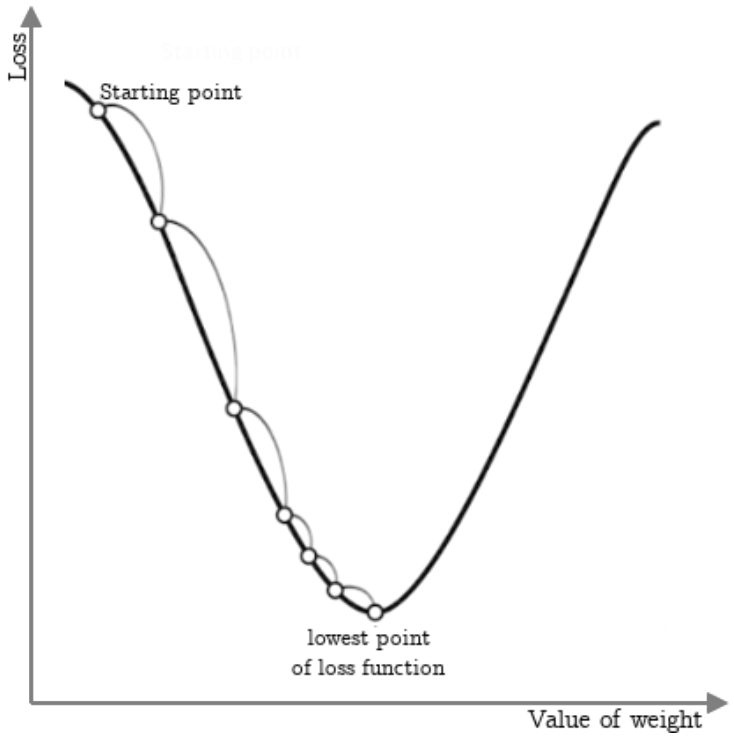

Figure 25: Example of a gradient descent, where the calculation travels from a starting point to the lowest point on the Loss Function. Image created by the author based on [50].

Stochastic Gradient descent

Gradient Descent algorithms inherently carry a drawback of a massive amount of computation needed in order for the algorithm to work given that for each feature of each data point it has to compute the derivative of the function. Stochastic Gradient Descent greatly reduces the amount of computation needed without degrading the performance of the algorithm. Instead of computing the derivative for every data point, it can randomly pick one data point from the data set for the calculation in each iteration. Alternatively, it can select a small sample of all the data points, and perform the calculation for the selected few. This is called "mini-batching". This combines the computational optimization of not selecting all the data points, with the accuracy and performance of considering all the data points. [52] [53]

Stochastic Gradient Descent is the most widely used Loss Function optimization algorithm, however, there are some cases that are problematic. Such a case can be a Non-convex Loss Function. In such a scenario, Gradient Descent is might not be able to locate the global minimum since there can be several local minima. [47]

Backpropagation

Before the examination of Back propagation, let´s introduce something called Forward Propagation. Forward Propagation is a term used for the "forward" flow of data in the ANN from the Input layer, through the Hidden layer up to the Output layer using the channels connecting the neurons. As already stated in the section describing Artificial Neural Networks as a whole, the

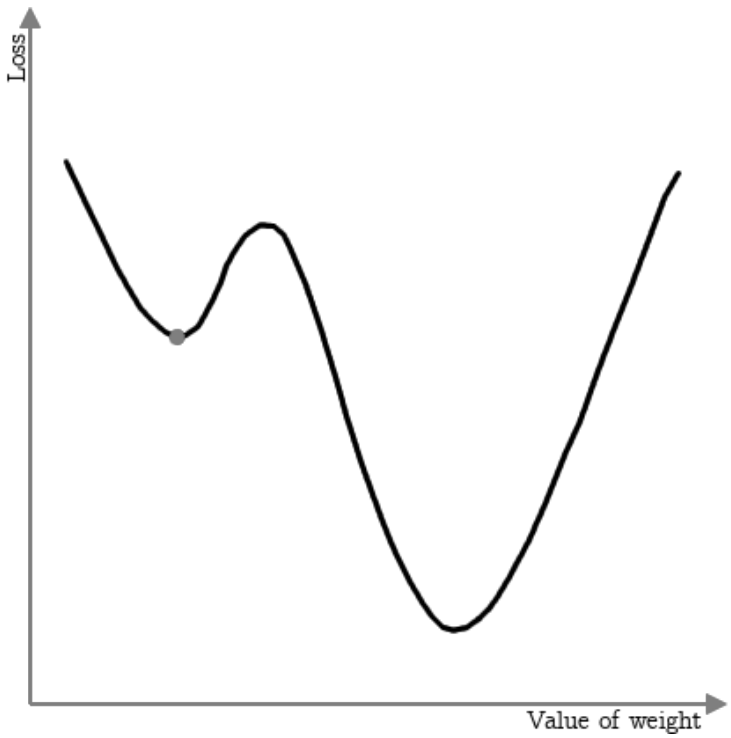

Figure 26: A non-convex Loss Function leading the gradient descent calculation to finding just the local lowest point, but failing to find the global lowest point of the Loss Function. Image created by the author based on [50].

activation of a neuron can be described as the weighted sum of all the previous activation in the preceding layer, plus a given bias that is then "inserted" into a chosen Activation Function. Knowing this, there are three variables that can be adjusted in order to change the activation of a neuron. The Bias, the Weight and the selection of an Activation Function.

For large ANNs, the Loss Function can be very complicated, meaning that the calculation of the gradient of such a function is very complex. Backpropagation is an algorithm that is tasked with just that. Simply put, Backpropagation optimizes the paths of the ANN with the goal to strengthen the paths that end up in the desired output (increases the weight of the channels and adjusts the Bias in that path) and weakens the paths that would otherwise lead to undesirable output (lowers the weights of the channels and adjusts the Bias in that path). [54] [55]

Vanishing Gradient

In the segment of this thesis where the different Activation Functions are introduced and explained, the term "Vanishing Gradient" is mentioned. As already obvious from the definitions presented above, the Activation Functions "map" the outputs into a space defined by them, which means in most cases it largely decreases the scale. This means that a large change in input will be returned as a small change in the output. The way how the functions are placed into the space also influences the derivative of said function. For example, a derivative of a Sigmoid Activation Function is relatively large when deriving around 0, but the derivative 9 gets closer to zero when the function gets larger or smaller.

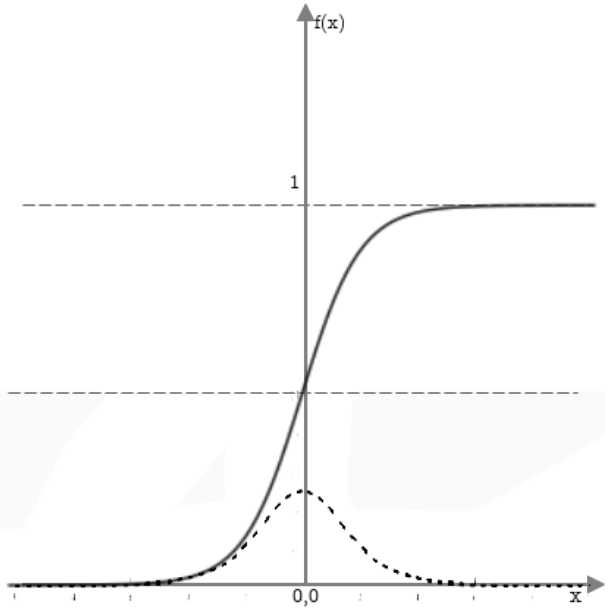

Figure 27: A derivation of a Sigmoid Activation Function showing the reduction of data space causing a significant Vanishing Gradient. Image created by the author based on [56].

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \tag{9}$$

Where:
x is the input

This does not cause a problem in ANNs that do not employ a large number of layers, however, with increasing layer counts, an effect called Vanishing Gradient starts to appear. Vanishing Gradient is simply put the decrease of a gradient in ANNs layers when the Backpropagation calculates the derivatives throughout the Artificial Neural Network. By the time the Backpropagation arrives at the Input layer, the gradient gets very small. This in practice means that the weights and Biases of layers in the beginning layers of the network will not be updated efficiently by the Backpropagation. The simplest way of solving the issue of Vanishing Gradient lies in the selection of an Activation Function that has such properties allowing for derivatives that don´t end up diminishing the values of the data space. [57] [58] [59]

2.4 MACHINE LEARNING CONCLUSION

In this section of the thesis, the topic of Artificial Neural Networks has been introduced and somewhat closely examined. The differences between Supervised and Unsupervised learning techniques have been explained, and for each, a selected number of specific learning techniques were presented.

The topic of ANNs has been dissected in further detail since its principles are chosen as the underlying learning technique for the practical part of this thesis. The mechanics behind ANN decision-making are somewhat of an advanced topic, but this thesis makes an effort in explaining

it in simple, understandable terms. Training of the ANN is a crucial part of the practical part of this thesis, and as such, it has been given substantial attention as well.

3 AIR TRAFFIC SURVEILLANCE

Air traffic surveillance is a key component for Air traffic management. Providing the ATC continuously with accurate positional and other data about the traffic is necessary for safe and efficient air coordination efforts. Even with some of the setbacks in recent years that negatively affected the growth of air traffic numbers, the amount of aircraft using the airspace is increasing when considering the long-term trends. Key systems and principles are important to understand in order to design methods and applications that can be used to improve the aircraft tracking capability of such surveillance systems. In the following section, such systems are presented, and the principle of operation is explained.[60][61]

3.1 EARLY DAYS OF RADIO WAVES

The principle today used by Primary Surveillance Radars (PSR) dates back to the early theoretical work of a Scottish physicist James Clerk Maxwell who developed the general equation describing electromagnetic fields in his work *A Dynamical Theory of the Electromagnetic Field* in 1865. In the 1880´s German physicist Heinrich Hertz build on the work done by Maxwell and defined a principle of reflecting radio waves off objects. Hertz proved his theory in 1888 when he did just that utilizing electromagnetic waves at a frequency of 455 MHz. This proof of Maxwell's theory laid the groundwork for all the uses of electromagnetic transmission employed today in our wireless technology.[62][63]

Gugliemo Marconi was an Italian electrical engineer credited as the inventor of the Radio thanks to his experiments in 1899. This experiment was the outcome of his interest and research of a *"Wireless Telegraph"*. In his experiments, Marconi observed that the electromagnetic waves did reflect from the objects back to the transmitter. In 1916 he followed his experiment when using short radio waves, crucial for the development of the RADAR. [63][64]

Christian Hülsmeyer, a German physicist and entrepreneur, was granted a patent for his *Telemobilscope* in 1904. this device was closest to a RADAR of its day, however, it could not determine the range of the object. This device used electromagnetic waves to intercept distant objects at sea (i.e. ships). Hülsmayer did demonstrate the system's capabilities to the German Navy forces, however, at the time it did not spark further interest due to the socio-economical environment in Germany (and Europe) at that time. Hülsmayer also developed a technique which estimated the range of the object having the Telebobilscope deployed on a mast and measured the angle at which the returned signal was strongest. Then, with triangulation calculation, the estimate of the distance to the object was returned.[64][65]

3.1.1 Early Military Developments

As with any major technological development in human history, military efforts were the main driving forces behind the research and development of Radio detection and ranging technology. With the rising tension throughout Europe in the 1930s, several countries began heavily investing in the research and development of a system using electromagnetic waves to intercept potential airborne threats. Some of the leading regions of the most significant efforts were the USA, Great Britain, France, the Netherlands, the (at the time relatively freshly formed) Soviet Union and

Japan.

One of the most significant and earliest functioning RADAR systems were developed and deployed in Great Britain in 1938. The RADAR system called Chain Home is regarded as one of the key contributors to the successful defence against German Air-raids, even with the British Royal Airforce being outnumbered by the German Luftwaffe. Chain Home remained in operation until the end of the Second World War. The Chain Home radar can be seen in figure 28.[64][63]

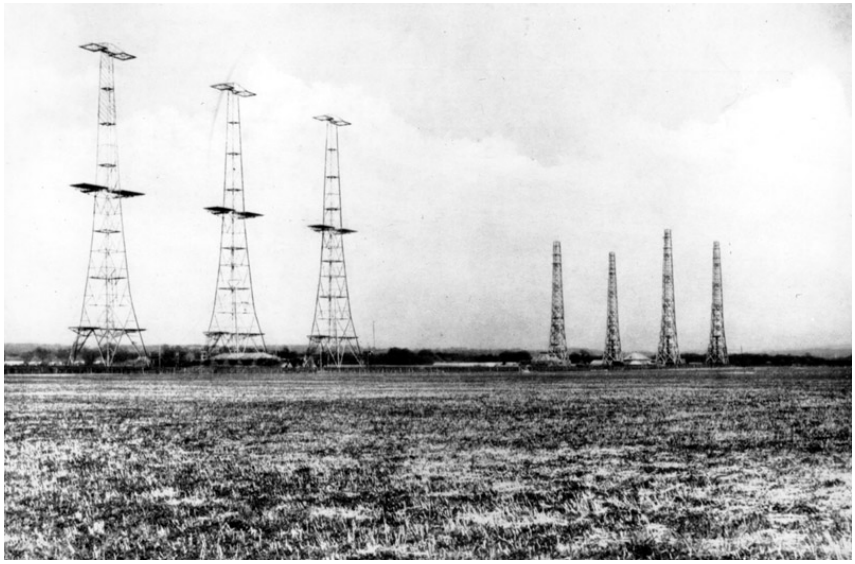

Figure 28: Early Warning Radar system Chain Home with the CF of 30 MHz deployed on the British east coast as part of the defence against German Luftwaffe Air-raids at the initial stages of WW2.[66]

3.1.2 Identification Friend or Foe

With a working Radar System with the capability to intercept incoming Aircraft, a question of determining the Aircraft´s designation arose. Having the knowledge of the presence of an Aircraft in the airspace is a great advantage, however, to determine if it is a friendly Aircraft coming home from a mission, or an enemy Aircraft on its way to bomb targets in the homeland is perhaps just as important in order to effectively manage the friendly fleet of defence fighters. Even though the principle of IFF (Identification Friend-or-Foe) was examined even before the construction of the Chain Home EW (Early-Warning) RADAR system, the issue quickly sprung into the researcher's focus after the incident in the Battle of Barking Creek in 1939 where a miss-identified friendly aircraft ended up shot down as a result of friendly-fire by a scrambled squadron of friendly Hawker Hurricanes.

The first concepts of such systems were based on a passive "reflector" antenna mounted on the Aircraft. This antenna was designed in such a way that it would provide the best possible gain for the Carrier Frequency of the ground RADAR. This design would result in a much stronger reflection from a friendly aircraft equipped with this type of antenna.[63][67]

IFF Mark I, II, III, IV, V

IFF Mark I is the first identification RADAR system that required a transponder to be installed

on the aircraft. This transponder helped to distinguish friendly aircraft with a *Regenerative* Receiver. This receiver would be tuned to the same frequency, as the CF (Carrier Frequency) of the RADAR, and the signal irradiating the aircraft would be amplified and fed back to a transmitter, sending it back to the ground radar. This would display a longer "blip" on the operator's radar display. At that time, there was no standardized frequency to be used. That meant that Mark I transponders were not a viable option in the long term.

In 1940, the *IFF Mark II* was introduced. Transponders capable of Mark II were equipped with several tuning boards and were able to change the operating frequency with regard to the corresponding ground RADAR. Also, the problem of overpowering the ground RADAR by the amplified reflection was solved by implementing an automatic gain control. With the accelerating development of RADARs, Mark II soon was unable to keep up.

Freddie Williams, an English engineer, proposed a single standardized frequency for all IFF already in 1940, however at the time, there was simply no need for such separation. Thanks to the successful development of the Magnetron, the allied forces started with the development of *IFF Mark III*. Transponders capable of operation under the Mark III standard were designed so that they would generate *replies* to *interrogations* sent from surveillance ground RADARs on a specific set of CFs. This was the first time that an early form of communication was enabled between the RADAR and the aircraft (aircraft could for example send a "mayday" reply informing the ground RADAR operators about the situation in the air). One of the biggest drawbacks of the IFF systems described was that a response from an aircraft could potentially solicit a reply from another aircraft in its vicinity, or in the beam of the onboard IFF antenna.

Even before WW2, researchers at the USNRL (United States Naval Research Center) were developing an IFF system of their own. The key feature that differentiated this IFF from the ones developed in Europe was that it used a set frequency for the interrogation, and a second set frequency designated for the reply. This would solve the issue of creating "fake" replies from other aircraft by the reply from the aircraft that is being interrogated. This system would eventually be known as the *IFF Mark IV*, however, the United States chose to use the already deployed solutions from the British. *IFF Mark V* would solve the problem of the Mark IV (sharing a similar frequency of 600MHz with a German Artillery RADAR) by moving the frequencies used up to around 1GHz. By the time IFF Mark V testing would have finished the Second World War has ended. That meant the further research pivoted to the research of the *IFF* Mark X.[68]

3.2 PRIMARY SURVEIILANCE RADAR

Primary Surveillance Radar (PSR) is the oldest and one of the most widely used Air Traffic Control (ATC) equipment used today. The history of early development and research was already presented in the previous chapters, however, its further development continues to this day, and it will likely remain a key system in the future as well.

3.2.1 PSR - Principle of operation

To understand the principle of how PSRs work, first it is needed to have an understanding of the components and their purpose of a PSR. To generate an electro-magnetical signal, a RADAR needs to be equipped with a *Transmitter*. A transmitter is mostly an electronic circuit that generates an electrical signal, that is fed to the antenna. The antenna then ensures the conversion from an electrical signal into an electromagnetic signal. The design of the antenna is crucial for the type of propagation of the electromagnetic signal. A parabolic antenna is the most widely used design for PSR RADARS. Since a PSR needs to be able not only to transmit a signal but also to receive it, it has to be equipped with a *receiver*. Receivers are again mostly electronic devices that can receive the electrical signal that has been received by the antenna it is connected to. Since a Radar is using the same antenna to transmit and receive electromagnetic signals it needs to be able to switch the signal path between the transmitter and receiver depending on which is used at the moment. *Duplexer* is responsible for just that.

The principle of PSR aircraft location lies in the transmission of an electromagnetic pulse (series of pulses) in a specific direction and then receiving the pulses from that direction that were reflected by the aircraft. The measurements that are made by the PSR are the distance of the aircraft from the position of the PSR, and the azimuth from the PSR. The way the PSR calculates the distance is fairly simple and shown in the figure (XY). The azimuth is determined by the current azimuth in which the antenna is oriented. This process is entirely passive, and it does not require any active system onboard the aircraft (*transponder*. The Range of the PSR is given by the Pulse Repetition Frequency (frequency of pulse transmission, not the Carrier frequency of the signal itself), the accuracy in distance measurement is given by the time measurement capability of the PSR, and the accuracy in azimuth is given by the beamwidth of the antenna.

There are several drawbacks to using PSRs. Due to the design of the antenna, there is a *cone of silence* directly above the PSRs antenna. Aircraft in this space will likely not be detected. Another drawback is having a high power consumption since the transmitter has to generate a pulse strong enough so that it can be detected after travelling towards an aircraft and reflected back by the aircraft. Such transmitters generate pulses in the range between 10 and 30kW. Acquiring identification is another issue the PSR system cant solve as well as determining the height of the aircraft due to the characteristic of the antenna. The issue of determining height also affects the accuracy of the RADAR in distance, as shown in figure (XY). The last two issues were the main reasons for the development of the next type of RADAR, Secondary Surveillance RADAR (SSR).[69]

Figure 29: Primary Surveillance RADAR by the Czech manufacturer ELDIS.[70]

3.3 SECONDARY SURVEIILANCE

The history of IFF development and use was already discussed in the chapter **Early Military Developments**. The IFF MARK V forms the basis for the principle of operation of the SSR in that it used two separate CFs for interrogation, and for the reception of replies. After the Second World War, an *IFF Mark X* (which originally started as an experimental research project) became the standard.[71]

3.3.1 SSR Principle of operation

The SSR works on an interrogation/reply principle. The signal sent from the SSR carries a specific "inquiry", that is received and decoded by the transponder mounted on the aircraft. Based on that interrogation from the SSR, the transponder generates a *reply* and encodes it into a signal that is then transmitted from the aircraft. This transmission is received and decoded by the SSR to retrieve the information encoded into the reply. This capability of encoding aircraft-specific information into the reply is called a Selective Identification Feature (SIF). The azimuth is determined the same way as with the PSR, however, for the calculation of the distance, the PSR has to adjust for the *processing* time of the transponder (de-coding the interrogation and encoding the reply). The adjusted formula for the distance calculation is shown in (XY).

The capability to transmit a signal that "carries" some information hugely improves the drawbacks of PSR. With an SSR, it is possible to identify an aircraft, and also retrieve its height besides other information depending on the SSR *mode* that is used. Another improvement is the reduced power consumption. Since the signal transmitted by the SSR has to only travel the distance to the aircraft, it can be significantly less powerful. This is also the reason why the antenna of an SSR can be by a planar phased array instead of the parabolic antenna of a PSR. Some of the drawbacks however remain, such as the *cone of silence*, and a large rotating antenna.

The principle of operation of an SSR allows for a number of possible information to be transmitted from the aircraft to the SSR enhancing the surveillance capabilities of the system. There are several modes, each defining what the provided information from the aircraft to the SSR will be, that are closely examined in the following text. Throughout the described Modes, there is a certain overlap between civilian and military-used modes, however, this thesis will be focused on the civilian-used modes.[71]

3.3.2 Mode A/C

With IFF Mark X, the civilian used SSR Mode A (alfa) and SSR Mode C were defined. SSR Mode A allowed for the 4-digit octal identification number of the aircraft. This code would be set on the transponder by the pilot. In order to interrogate the aircraft about its 4-digit identification code, the aircraft sends a series of pulses, with a time delay between the first and third of 8 μs. The function of the second pulse is described in a later chapter. Once the signal is received by the transponder, it generates a reply with the 4-digit octal identification set by the pilot into the reply and transmits it. There are 4096 possible combinations with some reserved. The structure of the interrogation and reply is shown in the pictures (XY) and (XY). In order for the SSR to obtain the altitude of the aircraft, it has to interrogate the aircraft by the SSR

Mode C. Mode C interrogation is similar to the interrogation in Mode A, with the difference in the time delay between the first and third pulse being 21μs instead of 8μs. The reply is again coded as 4 digits octal reply. The altitude encoded into the Mode C reply is retrieved from the aircraft's onboard Altimeter. Due to the limitations of the possible combinations of Mode C, the altitude is encoded in 100-ft increments.[72]

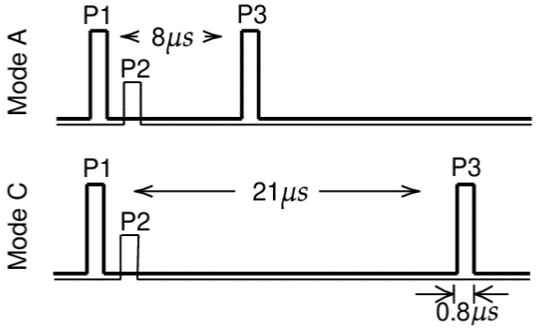

Figure 30: Graphic explaining the pulse modulation of a Mode A/C interrogation. The P2 pulse serves for the side lobe supression.[73]

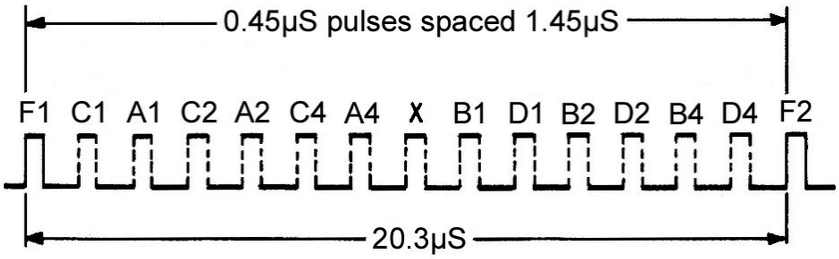

Figure 31: Graphic explaining the bits set up of the pulsed-modulated reply from the transponder ind MODE A/C. Bit groups ABCD can each facilitate values between 0 and 7. X bit was once used for identification of UAV traffic, some systems use it to identify garbled replies.[74]

Now let´s explain the P2 pulse in the interrogation. Besides the main lobe of the radar that directs the signal in the desired direction, there are "undesirable" side and back lobes. This can pose an issue since an aircraft that is located in the side lobe at the time of interrogation, could generate a reply. For that reason, there is a second omnidirectional antenna co-located with the SSR. This antenna transmits a pulse timed with the P1 and P3. The strength of this P2 pulse is set up so that it overpowers the potential side-lobe reception of P1 and P3 pulses by aircraft that are not in the main beam of the SSR. The principle is shown in the figure 32.[75]

There are some unwanted "features" of an SSR besides the side lobe interrogation. One of them is the *False Reply Un-synchronized in Time* (FRUIT). This phenomenon happens when a single aircraft is interrogated by two SSRs at around the same time and it causes "false" replies to the interrogation by either of the radars, making the target appear further or closer to the radar than it actually is. Another issue comes in the form of *garbling*. Garbling occurs when two signals "arrive" at the receiver close to each other. This can cause a change in the message transmitted since the pulse-modulated signal can be compromised but still decoded as a valid message.[76]

With the increasing amount of air traffic at the time, Mode A/C quickly became insufficient.

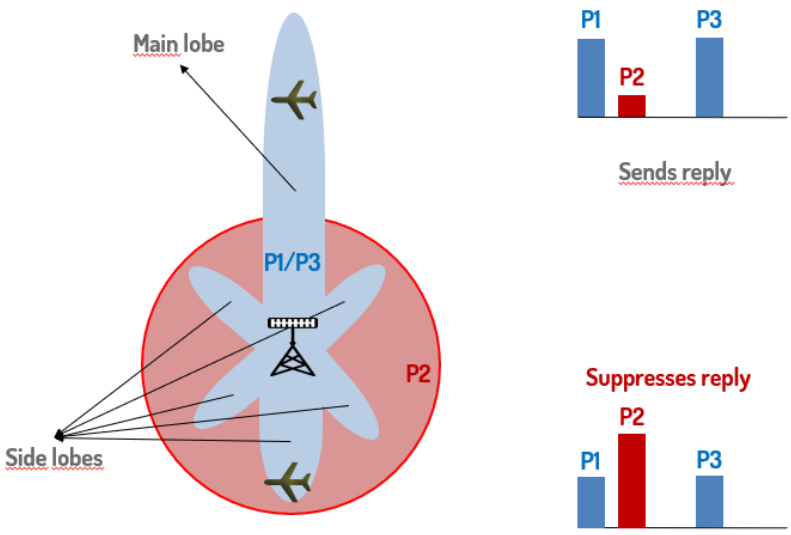

Figure 32: Graphics explaining the Side Lobe Suppression technique. Aircraft interrogated by the main lobe of the SSR will reply to the interrogation. Aircraft interrogated by the side lobe will not reply to the interrogation due to the P2 pulse overpowering the P1 and P3 pulses. Image created by the author.

That is one of the reasons that a system provides a data transfer capability to transmit a larger amount of data between the SSR and the aircraft's transponder. Since Mode A was set by the pilot on the transponder based on the request by the ATC controller, in high-capacity airspace it could trigger a situation where two aircraft would bare the same Mode A identification, The introduced *SSR Mode S* solved some of the drawbacks of previous mode issues by implementing *phase modulated* signal transmission.[76]

3.3.3 MODE S

The implementation of phase modulation increases the transmitting bandwidth significantly. Mode S capable transponders can encode a 24-bit address, giving almost 16 777 214 possible unique address combinations. Every transponder capable of Mode S is assigned his unique 24-bit address. With the increased capability of Mode S, it is possible to interrogate and send replies in several possible formats. Different *Uplink Formats* (UF) are standardized with each inquiring for different information from the aircraft. *Downlink Formats* (DF) are the standards for the replies generated by the Mode S interrogation.[76]

Mode S interrogation starts with a preamble consisting of a P1 and P2 pulse spaced by 2μs. After the preamble, the 56 (or 112) but data block is transmitted with the encoded interrogation. This data block is modulated using Binary phase shift keying (BPSK) and encodes the inquiry of the interrogation. The UF of the interrogation determines if the data block is 56 or 112 bits long. The relevant UF formats and their content is listed in Table 1.

For seamless integration between the Mode S and Mode A/C capable traffic, the method of All-Call/Roll-Call is implemented. In the All-Call period of interrogation, the surveillance of Mode A/C equipped aircraft takes place, as well as the acquisition of 24-bit aircraft addresses of Aircraft in the surveyed area (and of course their position). The Roll-Call is the selective

surveillance feature of Mode S is applied since the 24-bit aircraft addresses of the aircraft in the area of interest were acquired in the All-Call period.

Each SSR includes its unique Interrogation Code (IC) in its interrogations. This IC is also included in the reply transmitted by the interrogated aircraft after being acquired by the All-Call interrogation by the SSR. The SSR will instruct the aircraft to a *lockout* in the following Roll-Call, prohibiting the aircraft from responding to an All-Call interrogation with the SSR´s IC in the following 18 seconds.[77][78]

UF0 is a short air-to-air surveillance request utilized by the Aircraft Collision Avoidance System (ACAS). UF4 is the Mode S short surveillance request for the Mode S 24-bit identity, GND Flag and barometric altitude (MODE C). UF5 is similar to UF4, however, it requests the 4-digit identification (MODE A) instead of the barometric altitude. UF11 serves as the Mode S All-Call interrogation. UF16 is designated for the long air-to-air surveillance request used by the ACAS. UF20 and 21 are requested for Comm-B with altitude and identity respectively. Comm-D employs the UF24.

Uplink Formats Table		
UF number	bits	Type
UF0	56	Short air-air surveillance (ACAS)
UF4	56	Surveillance, altitude request
UF5	56	Surveillance, identity request
UF11	56	Mode S All-Call
UF16	112	Long air-air surveillance (ACAS)
UF20	112	Comm-B, altitude request
UF21	112	Comm-B, identity request
UF24	112	Comm-D (ELM)

Table 1: Table of used Uplink Formats, their length in bits and type of use.

Mode S encounters the same problem as Mode A/C in terms of unwanted side-lobe interrogation. For Mode S interrogation, the P2 pulse is used as a part of the preamble, so it does not serve the same function as in the Mode A/C interrogation. A control pulse P5 is transmitted omnidirectionally so that it overpowers the synchronization phase reversal that is placed 1.25μs into the transmitted data block if the aircraft is interrogated by an SSR side lobe.[79]

Based on the UF format of the interrogation, the transponder encodes a message with the corresponding attributes of a given DF. The reply comprises several pieces of information. Every Mode S DF includes the 24-bit address of the aircraft, but the rest of the possible data is DF-specific. The bit lengths and contents are listed in Table 2.

The DF selected for the reply depends on the UF used in the interrogation by the Mode S SSR. Two DFs that should however be mentioned are the DF18 and DF18. Those are used by the Automatic Dependent Surveillance Broadcast (ADS-B). The difference is that the DF18 is transmitted by a non-transponder equipped *beacon*, usually a Vehicle Tracking System (VTS).

Mode S provides a mechanism for the detection of garbled messages. This is ensured by the *Parity* bits that are included at the end of each message. Parity is a calculated value, that is

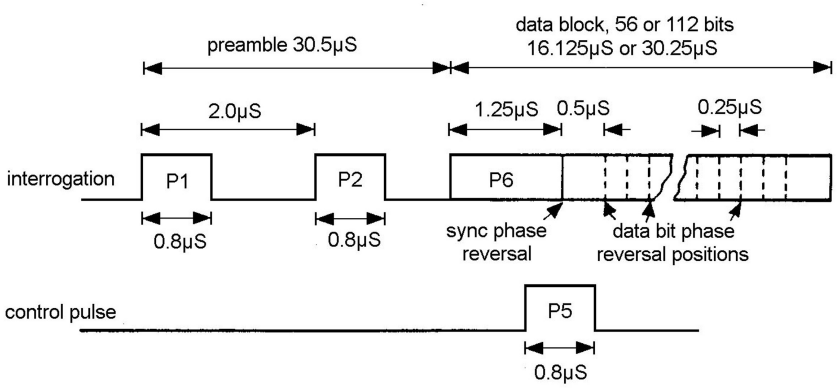

Figure 33: The interrogation in Mode S. The pulses P1 and P2 serve as the preamble. The data block with the BPSK encoded transmission is either 56, or 112 bits long depending on the UF used. The control pulse P5 ensures the side lobe suppression by overpowering the synchronization phase reversal in the data block.[79]

Downlink Formats Table		
DF number	bits	Type
DF0	56	Short air-air surveillance (ACAS)
DF4	56	Surveillance, altitude reply
DF5	56	Surveillance, identity reply
DF11	56	Mode S All-Call reply
DF16	112	Long air-air surveillance (ACAS)
DF17	112	Extended squitter
DF18	112	non-transponder Extended squitter
DF20	112	Comm-B, altitude reply
DF21	112	Comm-A, identity reply
DF24	112	Comm-D (ELM)

Table 2: Table of used Downlink Formats, their length in bits and type of use. [73]

DF 04	FS:3	DR:5	UM:6	AC:13	**AP:24**	
DF 05	FS:3	DR:5	UM:6	ID:13	**AP:24**	
DF 11	CA:3	AA:24			**AP:24**	
DF 17	CA:3	AA:24			ME:56	**AP:24**
DF 18	CA:3	AA:24			ME:56	**AP:24**
DF 20	FS:3	DR:5	UM:6	AC:13	MB:56	**AP:24**
DF 21	FS:3	DR:5	UM:6	ID:13	MB56	**AP:24**

Figure 34: DF00, DF05 DF11, DF17, DF18, DF20, and DF21 contents 3.[73]

specific for each message, based on its contents. At the reception, the receiver applies the same calculation to the received message and compares it to the Parity data block received at the end of the transmission. If the calculated values are the same as the received Parity message, then the receiver can be sure that the message that was transmitted was also received with the same content, and was not garbled with another message.[73].

To determine which information should be encoded into the MB data field, the 8-bit Comm-B

Data Fileds and their content		
Field	bits	content
DF	5	Used Downlink Format for the reply
FS	3	Current flight status
DR	5	Downlink Request
UM	6	Transponder Utility Mode
ID	13	Mode A identity code
AC	13	Altitude (25feet increments)
AP	24	Address Parity
MB	56	Message Comm-B

Table 3: Table of DF Data fields and their content.[73]

Data Selector (BDS) is used. BDS also defines the data to be encoded into ADS-B messages, even though they are considered Mode S Extended Squitter (ES) messages, and not Comm-B messages. Without the knowledge of the specific BDS register used for the message, it is not possible to decode the MB field. The Mode S DF and its relationship with different BDS registers with their contents are shown in the figure 35.[73]

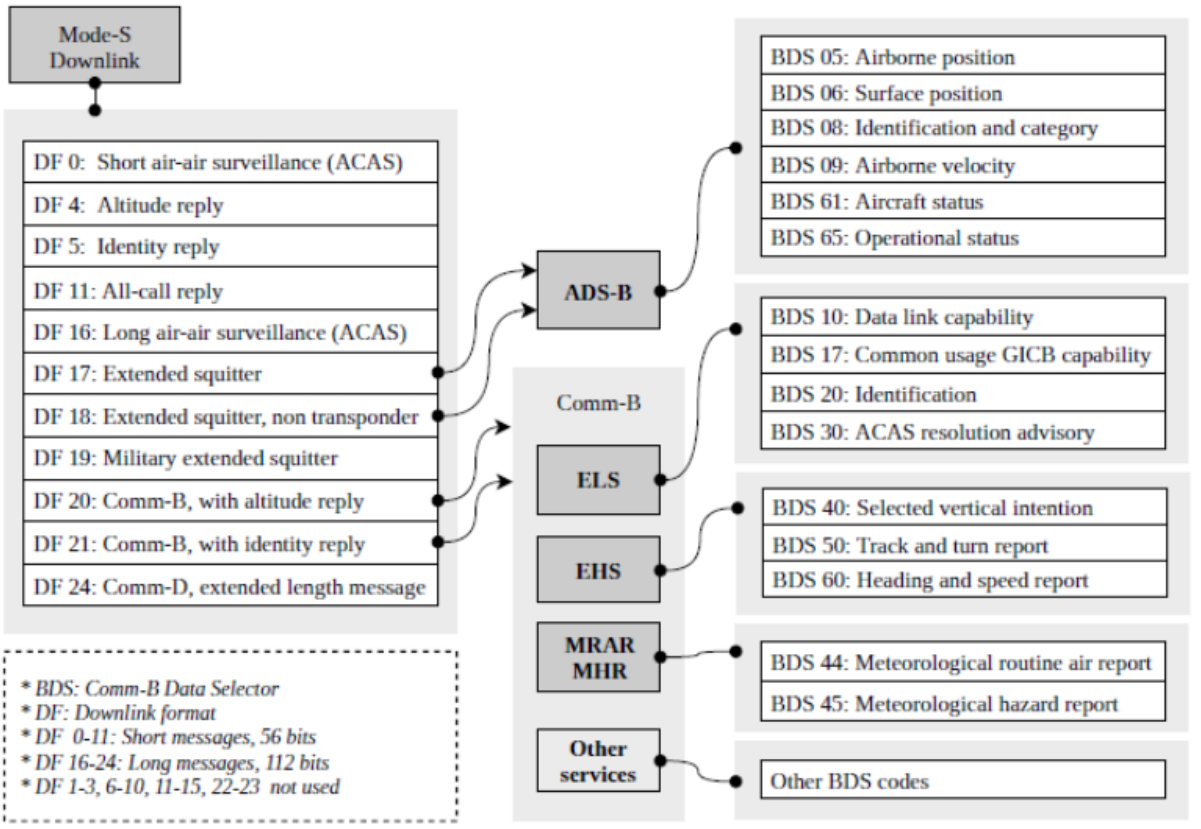

Figure 35: This figure show the BDSs assigned to their respective DFs. [73]

In Figure 35, the ELS, EHS and MRAR/MHS BDS groups are depicted as a part of the Comm-B Mode S messages. ELS stands for elementary Surveillance, and it contains, BDS registers 10, 17, 20 and 30. The EHS stands for Enhanced Surveillance, and it contains the BDs registers 40,50 and 60. Meteorological Routine Air Report (MRAR) consist of BDS registers 44 and 45. The explanation of each BDS register is not explained since it is beyond the intended scope of this

thesis.[80]

3.4 ADS-B

ADS-B is a cooperative dependent surveillance system. The position determined on board by a GNSS receiver together with other data such as velocity and ID is periodically broadcasted utilizing the Mode S Extended Squitter (ES). Since it is the positioning data source used for the practical part of this thesis, it will be given a closer look.[80]

3.4.1 ADS-B Versions

There have been 4 different versions of ADS-B since its introduction three. The reason behind the updates is the need for extending the pool of data that can be transmitted using ADS-B.[73]

The RTCA (Radio Technical Commission for Aeronautics) DO-260 defines the **ADS-B version 0** (V0). The only way to indicate the accuracy or integrity of the (horizontal) position data of the ADS-B system was the Navigation Uncertainty for Position (NUCp). The Navigational Uncertainty Category for Velocity Rate (NUCr) defined the accuracy and integrity of the velocity data. The NUCp is directly mapped by the Message Type Code (TC). The influence of the TC over the NUCp is shown in table 4.[80]

Relationship between the TC and NUCp in ADS-B v0 with accuracy values.				
Surface Position				
TC	NUCp	HPL	Horizontal Error	Vertical Error
0	0			
5	9	$\leq 7.5m$	$\leq 3m$	-
6	8	$\leq 25m$	$\leq 10m$	-
7	7	$\leq 185m$	$\leq 93m$	-
8	6	$\geq 185m$	$\geq 93m$	-
8	6	$\geq 185m$	$\geq 93m$	-
Airborne position with Barometric Altitude				
9	9	$\leq 7.5m$	$\leq 3m$	-
10	8	$\leq 25m$	$\leq 10m$	-
11	7	$\leq 185m$	$\leq 93m$	-
12	6	$\leq 370m$	$\leq 185m$	-
13	5	$\leq 926m$	$\leq 463m$	-
14	4	$\leq 1852m$	$\leq 926m$	-
15	3	$\leq 3704m$	$\leq 1852m$	-
16	2	$\leq 18520m$	$\leq 9260m$	-
17	1	$\leq 37040m$	$\leq 18520m$	-
18	0	$\geq 37040m$	$\geq 18520m$	-
Airborne position with GNSS Altitude				
20	9	$\leq 7.5m$	$\leq 3m$	$\leq 4m$
21	8	$\leq 25m$	$\leq 10m$	$\leq 15m$
22	0	$\geq 25m$	$\geq 10m$	$\geq 15m$

Table 4: Table of NUCp to TC mapping.[73]

The following **ADS-B version 1** (V1) is defined by the RTCA DO-260A. This version allowed for separate reports of Integrity and Accuracy by replacing the NUCp with the Navigation Accuracy Category for Position (NACp), Navigation Integrity Category (NIC) and Surveillance Integrity Level (SIL). The NUCr is replaced by the Navigation Accuracy Category for Velocity

(NACv). The NIC introduces more levels than the previous NUCp. Since there are more levels defined, it is no longer possible to map the just based on the TC. NIC requires an additional bit in order to distinguish between NIC codes within one TC. This bit is referred to as NICs (NIC supplement bit) and is located in the 44th position in the ME data field. The values of NIC and their dependence on NICs and TC are shown in the table 5.[73]

The surveillance Integrity Level (SIL) indicates the certainty of the measurement. in other words, what is the probability of the measurement being outside a defined acceptable range? The SIL parameters are depicted in the table 6.[73]

The Navigation Accuracy Category of Position (NACp), and is considered a "supportive" indicator of the NIC. NACp replaces the NUCp from ADS-B V1. NACp is designated to bits 45-48 in the ME message field. NACp allows for the determination of the 95% horizontal and vertical accuracy boundaries. The values for NACp are are the Horizontal Figure of Merit (HFOM), and the Vertical Figure of Merit (VFOM). The values for each NACp are depicted in table 7.[73]

Similar to NACp, the NACv also refers to the Navigation Accuracy category, however now it is reserved for the velocity. It replaces the NUCr from ADS-B V1. The bits 11-13 in the ME fields are taken by the NACv. NACv is used to define 95% of the errors in horizontal and vertical velocities. The values of HFOM and VFOM for each NACv category are depicted in table 8.[80]

ADS-B V1 NIC values, and the relationship with TC and NICs				
TC	NIC	NICs	RC	VPL
0	-	-	-	-
5	11	0	$\leq 7.5m$	-
5	11	0	$\leq 7.5m$	-
6	10	0	$\leq 25m$	-
7	9	1	$\leq 75m$	-
7	8	0	$\leq 185m$	-
8	0	0	$\geq 185m$	-
9	11	0	$\leq 7.5m$	$\leq 11m$
10	10	0	$\leq 25m$	$\leq 37.5m$
11	9	1	$\leq 75m$	$\leq 112m$
11	8	0	$\leq 185m$	-
12	7	0	$\leq 370m$	-
13	6	0	$\leq 926m$	-
13	-	1	$\leq 1111m$	-
14	5	0	$\leq 1852m$	-
15	4	0	$\leq 3704m$	-
16	3	1	$\leq 7408m$	-
16	2	0	$\leq 14800m$	-
17	1	0	$\leq 37040m$	-
18	0	0	$\geq 37040m$	-
20	11	0	$\leq 7.5m$	$\leq 11m$
21	10	0	$\leq 25m$	$\leq 37.5m$
22	0	0	$\geq 25m$	$\geq 112m$

Table 5: Table of ADS-B V1 NIC values determined by the TC and NICs.[73]

(The 1090MHz Riddle) (P01-FAA-AgendaItem3.pdf)

ADS-B V1 SIL values and their corresponding parameters P-RC and P-VPL		
SIL	P-RC	P-VPL
0	-	-
1	$\leq 1 \times 10^{-3}$	$\leq 1 \times 10^{-3}$
2	$\leq 1 \times 10^{-5}$	$\leq 1 \times 10^{-5}$
3	$\leq 1 \times 10^{-7}$	$\leq 2 \times 10^{-7}$

Table 6: Table of ADS-B V1 SIL values and the corresponding parameters of P-RC and P-VPL.[80]

ADS-B V1 NACp values and the co responding parameters of HFOM and VFOM.		
NACp	HFOM	VFOM
11	$\leq 3m$	$\leq 4m$
10	$\leq 10m$	$\leq 15m$
9	$\leq 30m$	$\leq 45m$
8	$\leq 93m$	-
7	$\leq 185m$	-
6	$\leq 556m$	-
5	$\leq 926m$	-
4	$\leq 1852m$	-
3	$\leq 3704m$	-
2	$\leq 7408m$	-
1	$\leq 18520m$	-
0	$\geq 18520m$	-

Table 7: Table of ADS-B V1 NACp values, and the corresponding values of HFOM and VFOM.[80]

ADS-B V1 NACv values and their corresponding parameters HFOMr and VFOMr		
NACv	HFOMr	VFOMr
0	-	-
1	≤ 10 ms^{-1}	≤ 15.2 ms^{-1}
2	≤ 3 ms^{-1}	≤ 4.5 ms^{-1}
3	≤ 1 ms^{-1}	≤ 1.5 ms^{-1}
3	≤ 0.3 ms^{-1}	≤ 0.46 ms^{-1}

Table 8: Table of ADS-B V1 NACv values and the corresponding parameters of HFOMr and VFOMr.[80]

With the **ADS-B version 2** (V2), defined by the RTCA DO-260B, there are some additions to the NIC, and some adjustments in the definitions of SIL parameters when compared to the ADS-B V1. There are two additional supplement bits that are added to the NIC specification. NICa corresponds to the NICs from the ADS-B V1. NICb is located in the 8th bit of the ME message field. NICc is located in the 20th bit of the ME message field. These additional bits widen the available data space in the TC. The updated table 9 for ADS-B V2 NIC values and parameters is depicted below. ADS-B V2 also introduces a supplementary bit for SIL. The SILs can be found in the 55th bit of the ME message field. The bit defines the time period used for the sampling of the probability. If SLSs bit is equal to 0, then the probability is sampled per hour. If the SLSs it is equal to 1, then the probability is sampled per each sample. The ADS-B V2 does not present any changes to the NACp or NACv.[80]

RTCA DO260C is currently the newest update document for the ADS-B.

ADS-B V NIC values, and the relationship with TC and NICa, NICb and NICc					
TC	NIC	NICa	NICb	NICc	Rc
5	11	0	-	0	$\leq 7.5m$
6	10	0	-	0	$\leq 25m$
7	9	1	-	0	$\leq 75m$
7	8	0	-	0	$\leq 185m$
8	7	1	-	1	$\leq 370m$
8	6	1	-	0	$\leq 556m$
8	-	0	-	1	$\leq 1111m$
8	0	0	-	0	$\geq 1111m$
9	11	0	0	-	$\leq 7.5m$
10	10	0	0	-	$\leq 25m$
11	9	1	1	-	$\leq 75m$
11	8	0	0	-	$\leq 185m$
12	7	0	0	-	$\leq 370m$
13	6	0	1	-	$\leq 556m$
13	-	0	0	-	$\leq 926m$
13	-	1	1	-	$\leq 1111m$
14	5	0	0	-	$\leq 1852m$
15	4	0	0	-	$\leq 3704m$
16	3	1	1	-	$\leq 7408m$
16	2	0	0	-	$\leq 14800m$
17	1	0	0	-	$\leq 37040m$
18	0	0	0	-	$\geq 37040m$
20	11	-	-	-	$\leq 7.5m$
21	10	-	-	-	$\leq 25m$
22	0	-	-	-	$\geq 25m$

Table 9: Table of ADS-B V2 NIC values determined by the TC and NICa, NICb and NICc. [80]

3.4.2 Structure and Contents of an ADS-B Messages

The message structure is 112 bits long- First 5 bits contain the Downlink format used. The next 3 bits contain the transponder *Capability* (CA). there are 7 possible capability values. CA0 informs about **Level 1 Transponder**, CA1, CA1 and CA3 three are reserved and currently not in use. CA4 informs about a **Level 2 Transponder** that is also capable of setting the capability to CA7 if on-ground. CA5 informs about **Level 2 Transponder** capable of setting the CA7 if airborne. CA6 informs about a **Level 2 Transponder** that is capable of setting CA7 either on-ground or airborne. CA7 signifies either that the Downlink request is valued at 0, or that the Flight Status is 2,3,4 or five on-ground or airborne. The following 24 bits carry the *ICAO* Aircraft address. The type of the ADS-B message is encoded to the following 5-bits long *Type Code* (TC). The TCs used are Aircraft Identification (TC1-4), Surface Position (TC5-8), Airborne Position with Barometric Altitude (TC9-18), Airborne velocity (TC19), Airborne Position with GNSS Height (TC20-22), Aircraft Status (TC28), Target State and Status Information (TC29), and Aircraft Operational Status (TC31). The TC is included in the 56-bit Message (ME), which contains data specified by the TC. The last 24 bits are used for the encoding of Parity data (PI). An example of an ADS-B message structure is shown in figure 36.[80]

The contents of the ME message filed are determined by the TC. Now let's have a closer look at

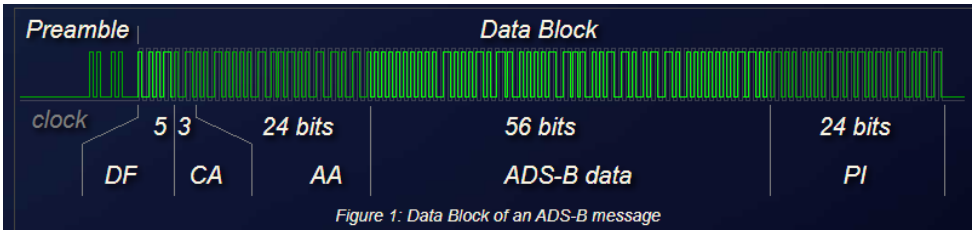

Figure 36: The structure of an ADS-B message containing the Preamble,5-bit Downlik Format (DF), 3-bit Transponder Capability (CA), ICAO 24-bit adress (AA), the 56-bit Message (ME/ADS-B), and the 24-bit Parity Check (PI).[76]

the specific messages that can be encoded into the ME.[80]

Aircraft Identification and Category

For a TC 1-4, the ME contains the *Aircraft's Identification*, or in other words it call-sign, and the *Aircraft´s vortex category*. The structure of the ME field is depicted in figure 37. The 56 bits are divided into 10 parts. The first, part is designated for the TC (5 bits), the second part for the Aircraft Vortex Category (CA, 3 bits), and the last 8 parts are used to encode the Aircraft's Call-sign (6 bits per part). To encode the Call-sign, the ASCII encoding format is used. The Aircraft vortex categories determined by the TC and CA are listed in table 10. The ADS-B-defined vortex categories are different from the ICAO-defined vortex categories, however, they do correlate. ICAO WTC L (Lights) corresponds with the TC4, CA1, WTC M (Medium) corresponds with the TC4 CA2/CA3, and the WTC H (Heavy) corresponds with the TC4, CA5.[80]

| TC,5 | CA,3 | C1,6 | C2,6 | C3,6 | C4,6 | C5,6 | C6,6 | C7,6 | C8,6 |

Figure 37: The structure of the ME field with the Aircraft Category and Identification. [73]

Aircraft Airborne Position

For TC9-18 the ME field contains the Aircraft´s Airborne position and the *Barometric* Altitude. For TC20-22, the ME field contains the Aircraft´s Airborne position and the *GNSS* height. The structure of the Airborne position ME field is depicted in figure *fig: Airborne.*[80]

| TC, 5 | SS, 2 | SAF, 1 | ALT, 12 | T, 1 | F, 1 | LAT-CPR, 17 | LON-CPR, 17 |

Figure 38: The structure of the ME field with the Airborne position encoded.TC-Type Code, SS - Surveillance Status, SAF - Single Antenna Flag, ALT - Altitude, T - Time, F - CPR Format, LAT-CPR - Encoded Latitude as per the CPR, LON-CPR - Encoded Longitude as per the CPR. [73]

The CPR Format is worth explaining. CPR stands for *Compact Position Reporting*, and it allows for position encoding using fewer bits than if the system were to encode the traditional Longitude Latitude values into the transmitted message. The CPR utilizes encoding two types of positioning data into separate odd and even messages.[80]

Aircraft Vortex Category CA designation with the relationship to the TC used.			
Vortex Category	CA	TC	Defining parameters
Reserved	not-specified	T1	-
No info	0	not-specified	-
Surface Emergency Vehicle	1	2	-
Surface Service Vehicle	3	2	-
Ground obstruction	4-7	2	-
Glider	1	3	-
Aircraft Lighter than Air	2	3	-
Skydiver	3	3	-
Ultralight, Paraglide,Hang-glider	4	3	-
Reserved	5	3	-
UAV	6	3	-
Space/Transatmospheric Vehicle	7	3	-
Light Aircraft	1	4	$\leq 7000kg$
Medium 1 Aircraft	2	4	7000 - 34000 kg
Medium 2 Aircraft	3	4	34000 - 136000 kg
High Vortex Aircraft	4	4	-
Heavy Aircraft	5	4	$\geq 136000kg$
High Performance Aircraft	6	4	$\geq 5gacc, \geq 400ktv$
Rotorcraft	7	4	-

Table 10: Table of Aircraft Vortex Categories and their corresponding TC an CA. [80]

In order to calculate the Aircraft position for the first time (without the knowledge of the previous position of the aircraft), the system has to approach the localization using the *Globally unambiguous position decoding*. This localization technique requires the receiver to include both messages (odd and even) in the calculation of the position. Once the position of the aircraft is acquired for the first time, the receiver can include only one message since the *Global* ambiguity is solved by the previous localization. This localization is called *Locally Unambiguous Position Decoding*. In order for the CPR to work, there are CPR zones defined for the entire Globe.[80]

The CPRs capability of shrinking the number of bits needed for the position transmission is the reason for its employment in the encoding, however, the drawback is that for a global position decoding there are two messages needed. ADS-B reception systems can implement mechanisms to check if the decoded calculation makes sense. One of these checks can be referencing the receiver position, and determining if the decoded position of the aircraft is in range of the ADS-B transmission.[80]

Aircraft Surface Position

For TC5-8, the aircraft Surface position is encoded in the ME field. This is implemented for such cases when the aircraft is on the ground. Since there is no need for altitude information, the bits that would be used fur just that are now encoded with the velocity and track angle information.

The structure of the message is depicted in figure 39. The Movement Field encodes the ground speed of the aircraft. The speed is not encoded linearly for the whole range of speeds in order to have better precision at lower speeds, and a reasonable amount of data at high speeds. Possible entries are depicted in the table 11. The status for Ground Track in the S field tells if the given ground track should be considered Valid (S1), or invalid (S0). The position of the aircraft is encoded and for that matter decoded the same way as in the3.4.2.[80]

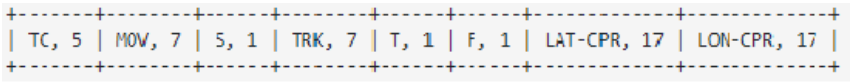

Figure 39: The structure of the ME field with the Surface position encoded.TC - Type Code, MOV - Movement, S - Status for ground track, TRK, Ground Track, T - Time, F - CPR Format, LAT-CPR - Encoded Latitude as per the CPR, LON-CPR - Encoded Longitude as per the CPR.[80]

Aircraft Surface position: Movement field values.		
Encoded Ground Speed	Range	Discrete
0	Not available	-
1	"Stopped", v $\leq 0.125kt$	-
2-8	$0.125 \geq v \leq 1kt$	0.125 kt
9-12	$1 \geq v \leq 2kt$	0.25 kt
13-38	$2 \geq v \leq 15kt$	0.5 kt
39-93	$15 \geq v \leq 70kt$	1 kt
94-108	$70 \geq v \leq 100kt$	2 kt
109-123	$100 \geq v \leq 175kt$	5 kt
124	v $\geq 175kt$	-
125-127	Reserved	-

Table 11: Table of Aircraft Ground Movement speeds and their encoding. [80]

Aircraft Airborne Velocity

For a TC19, the ME field contains the Airborne velocities of the transmitting aircraft. The decoding of TC19 ME field is quite challenging due to its large number of sub-fields. The sub-fields and their content is depicted in figure 40. The fields that are not Subtype specific are *IC* (1 bit) giving the Intent change flag, *NUCr/NUCv* (3 bits) giving the Navigation Uncertainty Category for velocity, *VrSrc* (1bit) giving the Source of the vertical rate (VrSr1=Baro, VrSrc0=GNSS), *Svr* (1 bit) giving the Sign for Vertical Rate (Svr1=Down, Svr0=Up), *VR* (9 bits) giving the Vertical Rate, *SDif* (1 bit) giving the Sign for GNSS/Baro altitude difference (SDif1=GNSS above Baro, SDif0=Baro above GNSS) and *dAlt* (7 bits) giving the GNSS/Baro difference.[80]

The subtype-specific field content depends on the subtype selected. Subtype 1 and 2 is used when the data on the Aircraft's ground speed is available. There are four main components, Dew, Vew, Dns and Vns for the subtype 1 and 2 fields. The *Dew* field (1 bit) marks the Direction of the East/West velocity component (Dew1=East to West, Dew0=West to East). The *Vew* field marks the East/West velocity components themselves (10 bits). The *Dns* field (1 bit) is the equivalent of Dew field, but for the North/South velocity component. The *Vns* (10 bits) is the equivalent of Vew, but for the North/South velocity component.[80]

Subtypes 3 and 4 are designated for situations, where the Aircraft´s ground speed is not available. his can be caused by several reasons, one of which can be the unavailability of any GNSS signal. In such cases, The Subtype specific fields are *SH* field (1 bit), giving the status of the magnetic heading information (SH1=available, SH0=unavailable), the *HDG* field (10bits) contains the Magnetic heading data. The *T* field gives the Airspeed Type (T1=Indicated Air Speed (IAS), T0=True Air Speed (TAS)), and *AS* contains the Air Speed.[80]

Figure 40: The structure of the ME field with the Airborne Velocity encoded. TC - Type Code, ST - Subtype, IC - intent change flag, IFR - IFR capability flag, NUCr/NUCv - Navigation uncertainty category for velocity (ADS-B V0/1,2), STSf - Subtype Specific Field, VrSrc - Vertical Rate Source, Svr - Sign for Vertical Rate, VR - Vertical Rate, SDif - Sign for GNSS/Baro Altitude Difference, dAlt - GNSS/Baro Altitude difference. [73]

Aircraft Operation Status

With the TC31, the ME field can contain the Aircraft Operation Status. This Operation status contains information regarding the surveyed aircraft. The Aircraft operation Status message differs between the ADS-B V0/1/2. For ADS-B V0, the message structure is depicted in figure 41. Even though the TC31 is defined for the ADS-B V0, version 0 transponders do not transmit any operation status messages.[80]

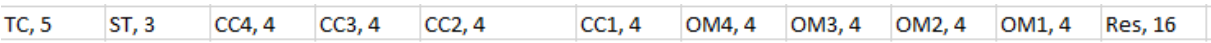

Figure 41: The structure of the ME filed for TC31 for ADS-B V0. TC - Type Code, ST - Subtype code, CC4 - Enroute Operational Capabilities, CC3 - Terminal Area Operational Capabilities, CC2 - Approach/Landing Operational Capabilities, CC1 - Surface Operational Capabilities, OM4 - Enroute Operational Status, OM3 - Terminal Area Operational Status, OM2 - Approach/Landing Operational Status, OM1 - Surface Operation Status. [80]

For ADS-B V1, the message structure is depicted in the figure42. The operation status is implemented in the broadcast of the V1 transponders. The ADS-B V2 Aircraft Operational Message is very similar to the ADS-B V1, updated with some additions to the message. THE *GVA* (2 bits message field is added providing the Geometric Vertical Accuracy if ST0, and being reserved if ST1.[73]

Figure 42: The structure of the ME filed for TC31 for ADS-B V1. TC - Type Code, ST - Subtype code, CC - Capacity Class, OM, Operational Mode, Ver - ADS-B Version, NICs - NIC supplement bit, NACp - Navigational Accuracy Category - position, BAQ - (if ST0) - Barometric Altitude Quality (if ST1) - Reserved, SIL - (if ST0) - Barometric Altitude Integrity (if ST1) - Track Angle od Heading, HRD - Horizontal Reference Direction.[80]

3.5 MULTILATERATION

Multilateration (MLAT) could be considered an alternative form for obtaining surveillance of Air traffic, however, it is currently one of the fastest-growing numbers of deployments and most saw after by Air Traffic services providers. Unsurprisingly, multilateration began as a research and application for military efforts but slowly transitioned into civilian applications.[81]

3.5.1 Multilateration Principle of operation

Multilateration is based on the *Time Difference of Arrival* (TDOA) method. The principle is based on the *multistatic* reception of a signal transmitted by an aircraft. With the information about the *Time of Arrival* (TOA) of the signal at several receiving stations deployed in the area, the system calculates the differences of that reception between the stations and determines the position of the aircraft mathematically. In order to understand the TDOA method, let's revise the definition of a hyperbola, since the TDOA method is also referred to as the *Hyperbolic* localization method.[81]

Hyperbola can be defined as a curve constructed by a set of points with a constant difference in distances between each point and two foci. Figure 43 depicts the construction of a hyperbola. With this knowledge, if the two foci are considered the two receiving stations, the calculation of the TDOA will provide the *hyperboloid* (3D localization) on which the aircraft that transmitted the signal is located at. Now if there are 4 receiving stations, it is possible to calculate 3 TDOAs meaning it is possible to construct 3 hyperboloids. These three hyperboloids will intersect (in most cases) at two points in space, one of which can be easily discarded since it is ussaly claculated to be under the earth surface. The second calculated position is considered to be the position of the surveyed aircraft. Figure 44 makes an effort to simply explain the localization principle of multilateration.[81][82]

The basic MLAT system is passive, meaning it does not require any transmission to be sent by the system. It utilizes the existing SSR replies (ADS-B) transmissions in order to localize the aircraft. However, usually, the MLAT surveillance systems are deployed with active SSR interrogators. This provides the system not only with some sort of control over the generated replies, but such an active system is capable of so-called *Multi-Ranging*. What is Multi-Ranging and the reason for its implementation is examined in the following chapter. The advantage of MLAT systems is the sufficient use of already existing messages transmitted by the Aircraft. Only information that the MLAT system need is the TOA of the same signal at each receiving station. [81][82]

3.5.2 Multilateration Accuracy

There are three main components that negatively affect the accuracy of MLAT systems. σt σt is the standard deviation of the TOA measurement at the receiver. Some of the effects that influence the TOA measurements are the shape of the pulse (a pulse with a sharp leading edge will provide a better TOA measurement), the rounding of the TOA measurement, and the time measurement discrete of the receiver. $\sigma \mathbf{c}$ is the deviation of the travel speed of the signal (even though the signal travels at a constant speed of light, due to atmospheric influence over the

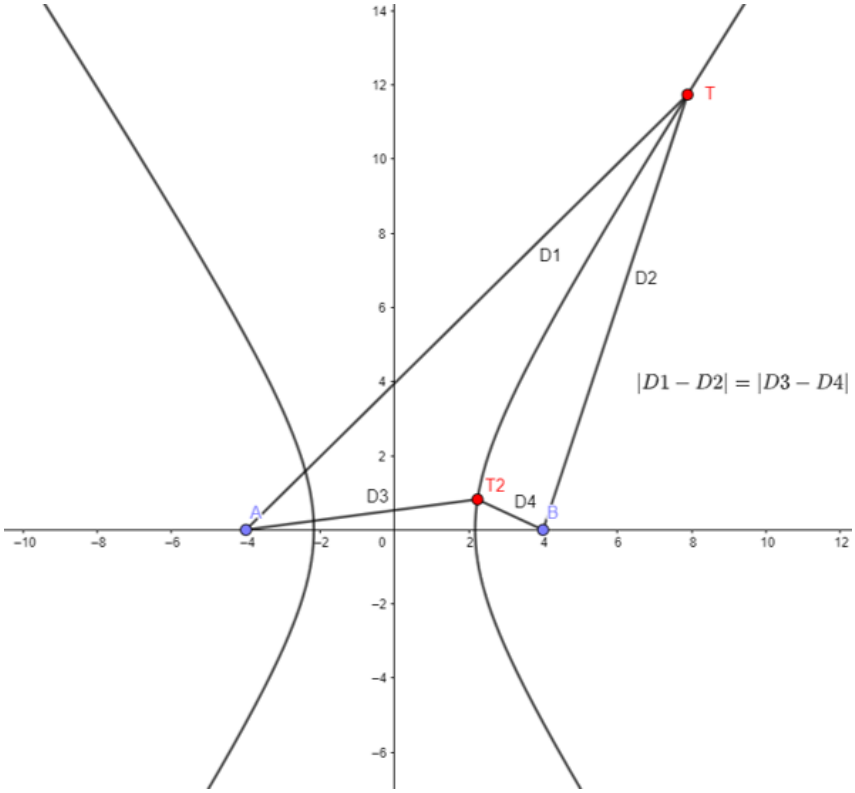

Figure 43: Construction of a hyperbola. the difference in distance between any point of the hyperbola and the two foci is constant throughout the hyperbola. Image created by the author.

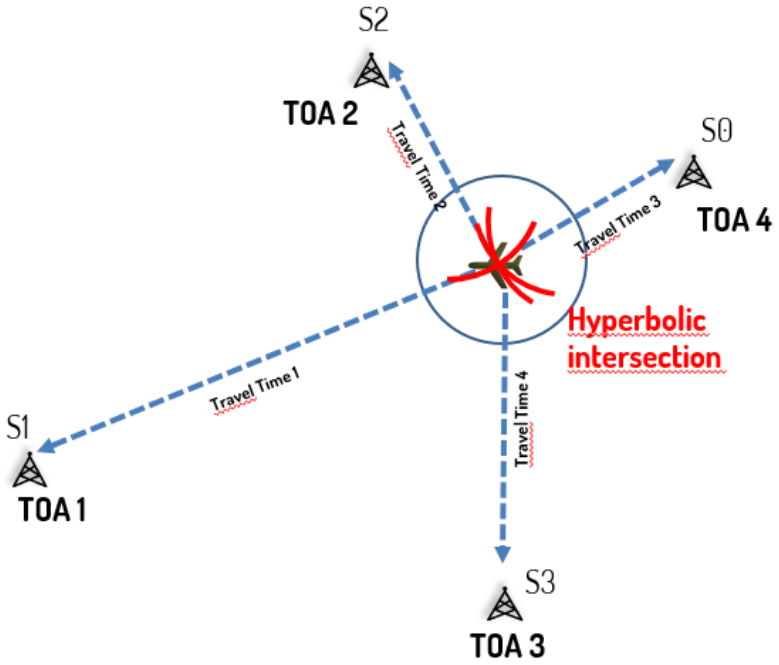

Figure 44: The multilateration principle of aircraft localization by the calculation of 3 TDOAs. Image created by the author.

path of the signal there is some deviation in the actual time it takes the signal to travel between the aircraft´s transmitter and the ground receiver). Finally, the geometry of the receiver's deployment influences the *PDOP* (Positional Dilution of Precision). PDOP consist of VDOP (Vertical Dilution of Precision), and HDOP (Horizontal Dilution of Precision). The σt determines the *dispersion* of the constructed hyperboloid. The PDOP determines the angle at which the hyperboloids intersect. It can be said that a PDOP is considered good if a small change in the aircraft position leads to a large change in TDOA, and PDOP can be considered bad if a large change in the aircraft´s position leads to a small change in TDOA. It is to be noted that the PDOP does not depend solely on the geometry of the receiving station's deployment, but also on the area of surveillance in relation to that specific deployment. The combination of the three sources of deviation between the actual and calculated position defines the *Area of Inaccuracy*. Area of inaccuracy is a physical 3D space that contains all the possible positions of the surveyed aircraft.[81][81]

3.5.3 Active Multilateration

So far the principles behind passive MLAT surveillance systems were discussed. MLAT surveillance systems however are usually deployed with one or more transmitters, that *actively* interrogate the Aircraft. This allows not only for at least some sort of control over the generated replies, but most importantly it allows for the application of *Multi-ranging* (elliptical) method. Multi-ranging is used for the reduction of the Area of Inaccuracy in measurements with high PDOP. Since the system knows the time of the interrogation transmission, and the time of the reply reception (TOA), it can construct an ellipsoid, on which the aircraft is located in 3D space. This ellipsoid can help reduce the Area of Inaccuracy quite drastically for such areas where the PDOP of the system is very high.[83]

3.6 AIR TRAFFIC SURVEILLANCE CONCLUSION

Every Air Traffic Surveillance system covers a different role in the overall ATC environment. This means that no single system can be chosen as the only source of the Air Traffic situation. Table 12 shows the different tasks and capabilities of the surveillance system with the type of aircraft that is being surveyed.

Air Traffic Surveillance system capabilities		
Surveillance system/Aircraft	Cooperative	Non-Cooperative
Dependent	ADS-B	-
Independent	SSR/MLAT	PSR

Table 12: Table of Air Traffic Surveillance systems capabilities.

To understand the figure 12, let's explain the labels used. Aircraft is considered Non-cooperative if it is not equipped with any form of transponder that would allow it to decode interrogations, or encode replies (unsolicited replies in the case of ADS-B). Aircraft that is capable of transmitting a response (or generating an ADS-B message), is considered a Cooperative. A surveillance system is considered Independent if the aircraft position is determined by the surveillance system itself. If the position information is provided by the surveyed aircraft, the surveillance system is

considered Dependent.

4 TRACK FILTERING

Tracking an aircraft is a task of updating the aircraft's position (or other information but for the scope of this thesis let's focus on position tracking) in time. By simply plotting the updated aircraft position provided by the chosen surveillance system (actually by the fusion of several surveillance systems but let's keep it simple for now), the accuracy of the track is solely dependent on the accuracy of each position determination. The accuracy of the track can be increased if the system considers the previous position and some other metrics that can be retrieved from the available data. Track filters are doing just that, and the filters that are most commonly utilized in Air traffic Surveillance are the focus content of this chapter.[84]

4.1 *g-h* FILTER

The *g-h* filter is also referred to as the α-β filter or the *f-g* filter. The *g-h* filtering algorithm is considered a steady-state two-dimensional Kalman filter. [85] In aircraft tracking problems, the implementation of a *g-h* filter allows for an increase in the positional accuracy and the accuracy of velocity measurement. To implement any filter into the tracking system, the system dynamic model has to be defined. The dynamic system model. An example of a system dynamic model with constant velocity is shown in 11. The dynamic system state equations are different for a system with linear or non-linear acceleration.[86]

$$x_{n+1} = x_n + \Delta N \dot{x_n} \tag{10}$$

$$\dot{x}_{n+1} = \dot{x}_n \tag{11}$$

Where:
x_{n+1} is the position at n+1
x_n is the position at n
ΔN is the time difference between the n and n+1
$\dot{x_n}$ is the velocity at time n
$\dot{x}_{n+1}$ is the velocity at time n+1

With the dynamic system model defined, the position and velocity prediction can be defined using the state extrapolation method. With the system state predictions calculated, it is possible to update the system state measurement taken at the time defined for the prediction using the equations for the updated system position 12 and updated system position. 13. These equations are called the α-β track update equations. Note that these are for a system dynamic model a constant velocity and linear position change.[86]

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + g(z_n - \hat{x}_{n,n-1}) \tag{12}$$

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n-1} + h(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}) \tag{13}$$

Where:
$\hat{x}_{n,n}$ is the estimated position of the dynamic system at n (after z_n)
$\hat{x}_{n,n-1}$ is the previous position estimate at time n (estimate made at n-1)
g is the factor dependent on the state measurement accuracy
h is the factor dependent on the velocity measurement accuracy
Z_n is the measurement at n
$\hat{\dot{x}}_{n,n}$ is the estimated velocity of the dynamic system at n (after z_n)
Δt is the time difference between measurements

In real-world applications, the g-h filter does not meet the requirements for tracking systems since it only deals with dynamic systems with constant velocity. It cant be expected the tracked aircraft will keep its heading and velocity constant for the entire duration of the flight. The g-h-k filter is more capable in the sense of track filtering dynamic systems with linear acceleration.[86] For such dynamic system, the state update equation in 14 15 16. The setup of the g-h-k parameters is crucial for the performance of the filtering algorithm.[86] The figure 45 show a flowchart of the estimation algorithm.

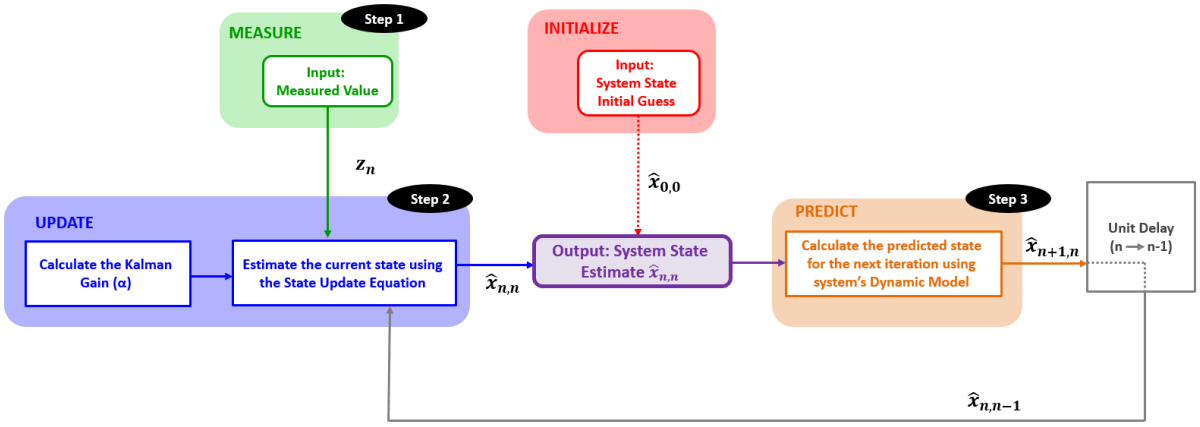

Figure 45: Flowchart explaining the estimation algorithm. [86]

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha(z_n - \hat{x}_{n,n-1}) \tag{14}$$

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n-1} + \beta(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}) \tag{15}$$

$$\hat{\ddot{x}}_{n,n} = \hat{\ddot{x}}_{n,n-1} + k(\frac{z_n - \hat{x}_{n,n-1}}{0.5\Delta t^2}) \tag{16}$$

Where:
$\hat{x}_{n,n}$ is the estimated position of the dynamic system at n (after z_n)
$\hat{x}_{n,n-1}$ is the previous position estimate at time n (estimate made at n-1)
g is the factor dependent on the state measurement accuracy
h is the factor dependent on the velocity measurement accuracy

k is the factor dependent on the acceleration measurement accuracy
Z_n is the measurement at n
$\hat{\dot{x}}_{n,n}$ is the estimated velocity of the dynamic system at n (after z_n)
Δt is the time difference between measurements
$\hat{\ddot{x}}_{n,n}$ is the estimated acceleration of the dynamic system at n (after z_n)

Even with the advantages of the *g-h-k* filter over the *g-h* filter, the real-life application is limited as well. Dynamics systems cant be expected to have either constant or linear acceleration for the entire track. Kalman filters however are more suitable for less predictable dynamic systems and are explained in the following section.[86]

4.2 KALMAN FILTER

Kalman filter is an iterative algorithm used to estimate system parameters (not unlike the *g-h* filter. Kalman filter is capable of providing accurate estimates even with the input data being inaccurate and/or noisy. Linear Kalman filter can be used for tracking of a dynamic linear system. For non-linear dynamic systems, there are other Kalman filters that can be employed[87]

Not unlike the *g-h-k* filter, the Kalman filter employs an extrapolation equation to describe the dynamic system model. The general equation is shown in 17.[86]

$$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n + w_n \tag{17}$$

Where:
$\hat{x}_{n+1,n}$ is the predicted state vector at n+1
$\hat{x}_{n,n}$ is the estimated system state vector at n
u_n is the control variable (measurable input to the system)
w_n is the process noise (input that can´t be measured)
F is the state transition matrix
G is the input transition matrix

The five Kalman filter equations explain the Kalman filter algorithm. The following equations show on the possible implementation of the Kalman filter for applications in state dynamic systems with constant velocity. The dynamic system state update equation is shown in 18. The system state and velocity extrapolation for a dynamic system with a constant velocity is shown in equations 19 and 20. The Kalman gain represents the estimation weight that is dynamically recalculated with each iteration. Kalman gain equation is shown in 21. Covariance update is also called the correction equation and is defined by the equation 22. For a dynamic state system with a constant acceleration, the covariance extrapolation is defined as 23. To gain e better understanding of the Kalman filter algorithm, figure 46 showcases the relationships in the algorithm.[86]

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n-1}) \tag{18}$$

Where:
$\hat{x}_{n,n}$ is the estimated system state vector at n
$\hat{x}_{n,n-1}$ is the estimated system state vector at the previous n
K_n is the Kalman Gain
z_n is the system state measurement at time n

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n}) \tag{19}$$

$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} \tag{20}$$

Where:
$\hat{x}_{n+1,n}$ is the predicted system state vector at n+1
$\hat{x}_{n,n}$ is the estimated system state vector at n
Δt is the time difference between measurements
$\hat{\dot{x}}_{n,n}$ is the estimate of velocity at n
$\hat{\dot{x}}_{n+1,n}$ is the prediction of velocity at n+1

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \tag{21}$$

Where:
K_n is the Kalman gain for the system state estimation at n
$p_{n,n-1}$ is the covariance update estimation at n-1
r_n is the measurement uncertainty

$$p_{n,n} = (1 - K_n)p_{n,n-1} \tag{22}$$

Where:
$p_{n,n}$ is the estimated covariance update at n
K_n is the Kalman gain at n
$p_{n,n-1}$ is the estimated covariance update at n-1

$$p_{n+1,n} = p_{n,n} \tag{23}$$

Where:
$p_{n+1,n}$ is the predicted covariance extrapolation at the n+1
$p_{n,n}$ is the estimated covariance update at n

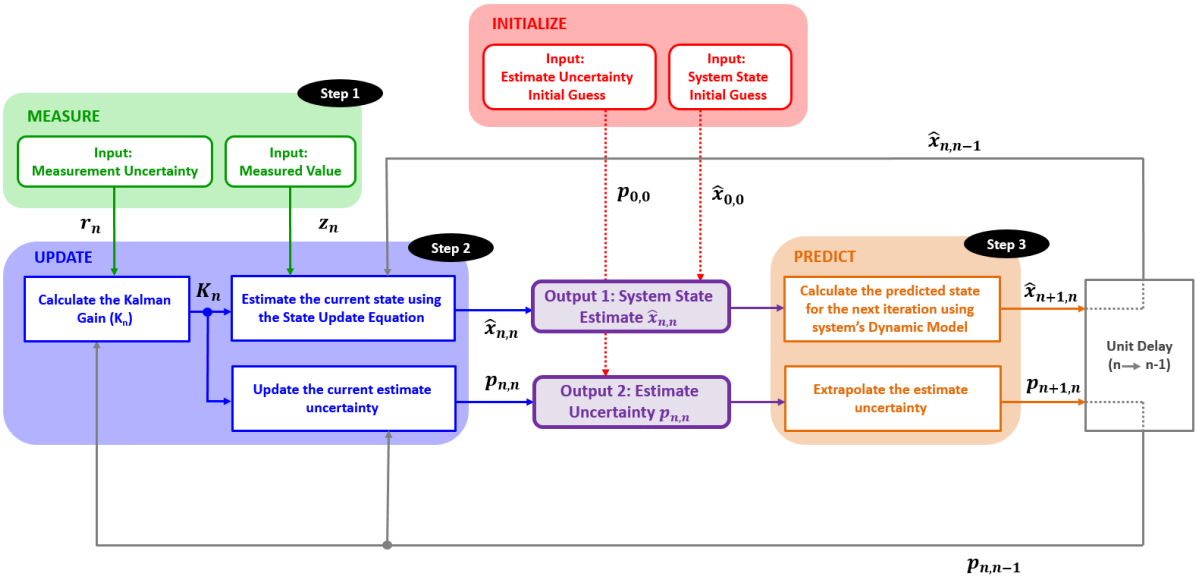

Figure 46: Flowchart explaining the Kalman filter algorithm.[86]

The Kalman Gain is an important part of the Kalman filter, it determines how much the prediction influences the system state estimate. In other words, if the measurement is labelled as low uncertainty, and the prediction is labelled with high uncertainty, the resulting Kalman gain would be High. This means the system state estimation will be more influenced by the measurement. Low Kalman gain is the result of a measurement with high uncertainty relative to the prediction uncertainty. In such case, the prediction will influence the final state estimation more.[86]

4.2.1 Extended Kalman Filter

A linear Kalman filter is limited by the need for a linear system. Extended Kalman Filter allows for the filtering of non-linear systems by a process called linearization. Linearization is a process of approximation of non-linear transition and measurement functions employing the Taylor Series Expansion. Linearization of non-linear function is done by a line tangent of the function in its mean value. When compared to the Simple Kalman Filter, the extended Kalman filter is less accurate, however, it allows for application in non-linear systems.[84][88]

4.3 TRACK FILTRATION CONCLUSION

With the understanding of different track filtering algorithms and approaches, the use cases for each are clearly different. The hypothesis proposed in this research discusses a track classification that would allow for an implementation of a system that would dynamically assign different filtration algorithms to the tracking algorithm based on the tracked aircraft behaviour. In other words, tracked aircraft with constant velocity and no change in the heading can be assigned with a linear filter, and aircraft with less predictable behaviour would be assigned with a track filter that can handle non-linear systems. This is argued to be beneficial in terms of the demands on computational power since less demanding aircraft tracks would be assigned with a filtration algorithm that has simpler calculations. The track classification methodology and execution are

closely examined in the following section.

5 METHODOLOGY

In this section, the methodology used for this research is presented. ADS-B is chosen as the source of surveillance data for the research due to its availability, however, note that other sources should be compatible with the methodology presented here. With the supervised ANN chosen as the machine learning algorithm, there is a need to define a data classification and labelling system. The first step however is data pre-processing which ensures that only data of certain quality are considered as input for the ANN. The methodology flowchart is shown in figure 47.

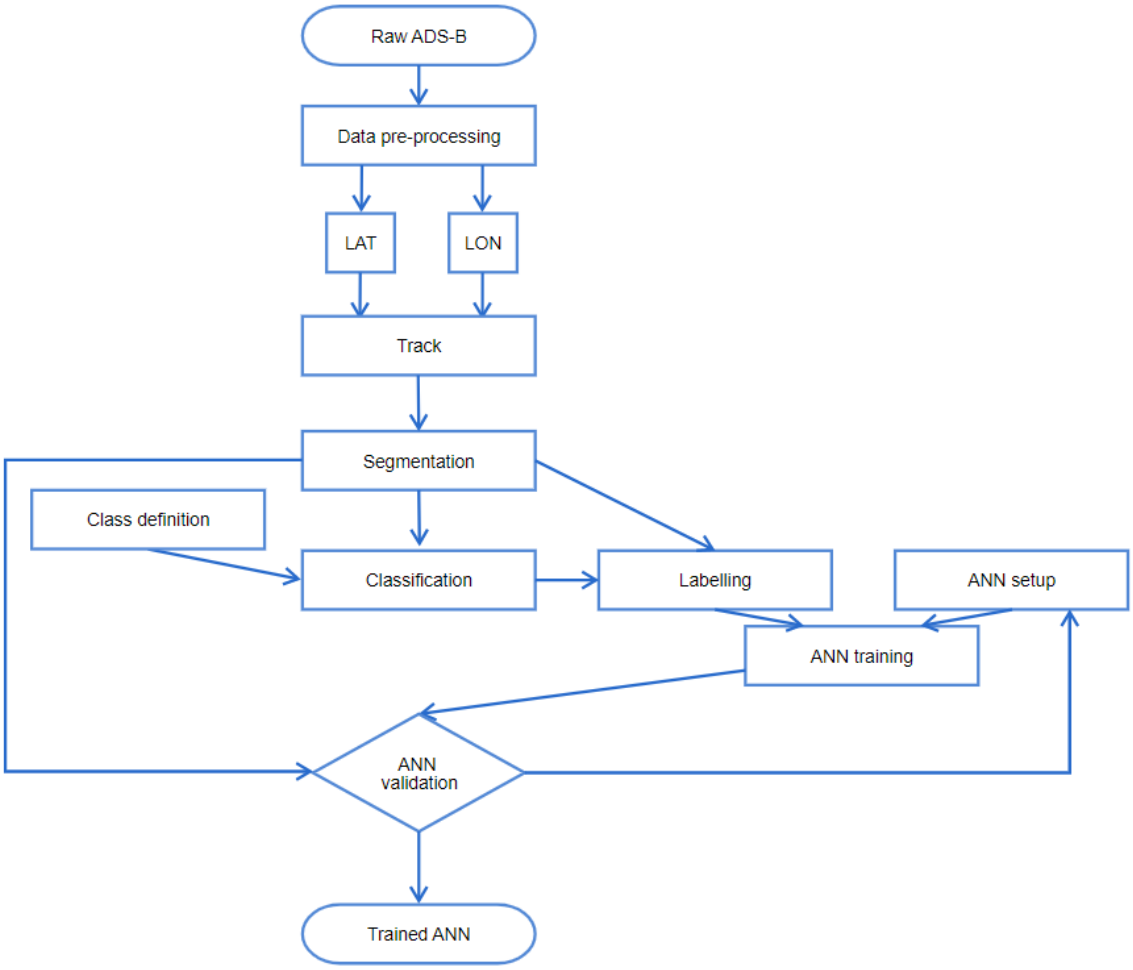

Figure 47: Flowchart of the thesis methodology.

Figure 47 depicts the methodology used in the research behind the thesis. Raw ADS-B data is firstly pre-processed to optimize the dataset for the task at hand. With the pre-processed dataset the track itself is then defined for the aircraft utilizing the positional information contained in the ADS-B messages. Tracks are then split into segments which will be then classified by a defined classification algorithm. The classification algorithm is proposed with the goal of classifying the segments based on the track behaviour. Classified segments are then labelled by the class they are a part of and are used as the training and testing datasets for the ANN designed for the goal of this research. Segments themselves are converted into images with normalized dimensions. This decision is made so that the input to the ANN is normalized being the image dimensions.

Another approach is providing the track segments as time series of 2D plots, however, this would require further data manipulation to ensure a certain level of normalization so that it can be used as input for the ANN. This additional data manipulation (interpolation of plots to ensure normalized update rate for example) could change some hidden data features that are present in the data if represented by an image. An MLP and CNN are proposed, trained and then compared to determine which is more suitable for the goal of this research.

To effectively train an ANN, the amount of data required is important. To be specific the more data is used for the training, the better the performance is to be expected from the ANN. This however means that the demands for computing power are great. The PC specs for this research are listed below:

- **Processor** - Intel(R) Core(TM) i5-10210U CPU @1.60GHz 2.11
- **RAM** - 8 GB (7.64 Usable)
- **Graphics** - Intel(R) UHD Graphics

This setup is not ideal for such an application due to the lack of a dedicated Graphics card. This does not affect the performance of the classification itself, but it does affect the time needed to train the ANN, and the time needed for the ANN to classify data.

5.1 DATA PRE-PROCESSING

Providing the ANN with well-prepared data is key to its efficient training. The raw ADS-B data is fed through several pre-processing algorithms before it is prepared to be fed into the ANN as input. The process of data pre-processing is examined in this section. A sample of the used ADS-B data is shown in figure 48.

index	addr	time	rawmsg	type_code	h_baro	h_gnss	mops	nucp	nucp_...	nacp	nacp_epu	segment_index	lat	lon
0	0a0046	1642411019.035	8d0a0046603920a01adbd22026b3	12	10250	nan	2	nan	nan	nan	nan	0	36.9381	3.28854
1	0a0046	1642411021.871	8d0a0046f82300030049b8193cf1	31	nan	nan	2	nan	nan	9	30	0	nan	nan
2	0a0046	1642411022.364	8d0a004660399437bed742bd40cf	12	10425	nan	2	nan	nan	nan	nan	0	36.94232	3.29029
3	0a0046	1642411023.453	8d0a00466039b0a10cdbfab1e843	12	10475	nan	2	nan	nan	nan	nan	0	36.94363	3.29087
4	0a0046	1642411024.606	8d0a00466039d0a13edc02be669c	12	10525	nan	2	nan	nan	nan	nan	0	36.94478	3.29134
5	0a0046	1642411025.424	8d0a00466039f0a176dc0bc4e163	12	10575	nan	2	nan	nan	nan	nan	0	36.94606	3.29187
6	0a0046	1642411026.582	8d0a0046603b043894d7655f4c14	12	10600	nan	2	nan	nan	nan	nan	0	36.9473	3.29238
7	0a0046	1642411026.625	8d0a0046f82300030049b8193cf1	31	nan	nan	2	nan	nan	9	30	0	nan	nan
8	0a0046	1642411027.443	8d0a0046603b20a1eadc1ec84a71	12	10650	nan	2	nan	nan	nan	nan	0	36.94872	3.29298
9	0a0046	1642411029.736	8d0a0046603b443942d782ddd79a	12	10700	nan	2	nan	nan	nan	nan	0	36.95135	3.29411
10	0a0046	1642411030.122	8d0a0046603b543962d78757cb6c	12	10725	nan	2	nan	nan	nan	nan	0	36.9521	3.29441
11	0a0046	1642411030.663	8d0a0046603b50a29adc3bcbc6c0	12	10725	nan	2	nan	nan	nan	nan	0	36.95274	3.29467
12	0a0046	1642411031.488	8d0a0046603b6439b6d7958c2b64	12	10750	nan	2	nan	nan	nan	nan	0	36.95405	3.29524

Figure 48: Sample of decoded ADS-B data before any processing for the ANN input.

This *raw* decoded ADS-B dataset contains a lot of data that are not needed for the track classification. Leaving only the key data greatly reduces the computational power needed for the later stages of pre-processing. Data deemed as necessary for the track segmentation and label definition are the aircraft's unique 24-bit ICAO address, timestamp, latitude and longitude. The rest of the data is filtered out. Furthermore, after calculating the time difference between all the rows in the dataset, some rows show an update rate that is not valid since their update rate is way too fast for an ADS-B system. Such rows are considered as noise (or false aircraft updates) and are filtered out from the dataset as well. Figure 49 shows the code used for the filtration of such rows.

```
def __remove_close_records(
    groups: List[DataFrame], threshold: float = 0.5
) -> List[DataFrame]:
    #discard rows that are too close to the previous row
    return [group[group["time"].diff() > threshold] for group in groups]
```

Figure 49: Code discarding rows from the dataset that have too small update time.

To calculate track flight dynamics indicators, the first step is to split the ADS-B data into segments which will be later labelled. The rules for the track segmentation are *heuristically* selected based on the reasonably expected correlation with the labelling logic and are as follows:

- single 24-bit ICAO address in one segment
- segments no longer than 15 seconds
- no less than 4 updates per segment

Having a single 24-bit ICAO address ensures that a single aircraft is present in a segment. The 15-second segment limit is chosen to minimise the change of the tracked aircraft behaviour in a single segment, which would make for a less clear classification. This would also result in a more demanding dataset for the ANN to classify. Limiting the lowest number of track updates with the distribution of plots within one segment. This condition ensures that the segment is not unbalanced with updates in time.

The dataset is then divided into segments using the code shown in figure 50, and the segments that do not comply with the rules listed above are discarded using the code shown in figure 51.

```
def __create_segments(
    groups: List[DataFrame],
    max_duration_limit: int = 30,
) -> List[DataFrame]:
    # for each segment calculate its index
    segments: List[DataFrame] = []

    for group in groups:
        # column with the time difference between rows (first row has time difference of 0)
        group["time_diff"] = group["time"].diff().fillna(0)
        # column with the cumulative sum of time differencec
        group["diff_cum"] = group["time_diff"].cumsum()
        # each row gets segment index added to the end
        group["segment"] = (group["diff_cum"] / max_duration_limit).astype(int)

        segments.extend(segment for (_, segment) in group.groupby(by="segment"))

    return segments
```

Figure 50: Code dividing the dataset into segments.

Having well-processed segments that comply with the defined rules is key for the definition of labels that are used for ANN learning. Inconsistency in the training dataset comprised of individual segments would greatly reduce the performance of the ANN not only while training, but also the label definition would be negatively affected resulting in undesirable and illogical ANN outputs.

```python
def __remove_unfit_segments(
    segments: List[DataFrame],
    min_duration_limit: int = 15,
    min_row_count_limit: int = 20,
) -> List[DataFrame]:
    # discard unfit segments
    ret: List[DataFrame] = []

    for segment in segments:
        # each segment has to have at least "min_row_count_limit" number of rowsúupdates (default value is 4)
        if len(segment.index) > min_row_count_limit:
            # take the first and last update and from "diff_cum" calculate the segment duration
            duration = segment["diff_cum"].iat[-1] - segment["diff_cum"].iat[0]
            # each segment has to be at least "min_duration_limit" long (default 15s)
            if duration > min_duration_limit:
                ret.append(segment)

    return ret
```

Figure 51: Code discarding segments that do not comply with the rules for valid segments.

5.2 TRACK CLASSIFICATION

This thesis focuses on track classification with the purpose of dynamic filtering applications. Track filters perform differently based on the "predictability" of tracked aircraft behaviour. Classification of tracked aircraft can be approached by a number of possible metrics which are discussed in this thesis.

5.2.1 Classification Metrics for Labelling

Defining what will be the classification based on is key for the labelling of the input data and for the ANN training performance. The classification of the track is intended to allow the ANN to classify segments for a dynamic filtering algorithm applied to a specific track. This means that the classification itself has to be correlated with the variables affecting the performance of the filter used. Simply put the track classification is based on how difficult it is to predict the behaviour of the track. There are endless possible indicators (and a combination of indicators) that can be retrieved from available data, that can be expected to have some correlation to the tracking behaviour. A selection of possible indicators is shown in this section. A trained ANN does not rely on the classification provided while being trained. While the ANN is being trained, it is only supplied with the label of the data that it is classifying, and it defines its own classification metric that is hidden from the user. These classification metrics defined by the ANN might not be the same as the ones defined by the user when training the ANN. This ensures that the ANN is capable of processing inputs that are not labelled by the user, and it still achieves the classification that is requested.

Tracked aircraft in upper *flight levels* can be argued to have more predictable behaviour than aircraft flying at lower flight levels. This assumption is based on the fact that most of the flight time spent in higher flight levels is without large and/or sudden changes in heading or altitude. Aircraft in lower flight levels can be expected to change their heading and altitude more unpredictably since it is expected that they are either about to land and are manoeuvring to prepare for an approach to an airport, or have just departed and are advancing into their planned flight path and altitude. With this in mind, the tracked aircraft's current altitude can be considered as an indicator (or better a part of a set of indicators) on which track classification can be based.

The *speed* of a tracked aircraft can also be considered as part of the set of indicators used for the track classification. Aircraft travelling at a higher speed can be considered less likely (and actually less capable) of sudden and/or large changes in heading or altitude. Subsequently, if the tracked target is determined to be travelling at a lower speed, it has the capability and is expected to proceed with more sudden and large changes in heading or altitude. The tracked aircraft *acceleration* is another flight performance indicator that can be used for the classification with a very similar application as the speed indicator described previously.

Flight plan of the tracked aircraft can be considered as a contributing factor to the classification algorithm as well. Taking into consideration the flight path and altitude given by the aircraft flight plan, the classification of such track would be quite straightforward, as long as the aircraft adhered to the flight plan. Any classification algorithm however can not rely solely on the

flight plan since there is no guarantee the aircraft will adhere to the flight plan due to changing conditions affecting the flight.

It can be argued that the *wake vortex category* can be another indicator of the aircraft's behaviour capability. Heavy aircraft do not have the same capability in terms of flight dynamics when compared to lighter aircraft. This assumption means that the wake vortex category can be considered one of the contributors to the classification algorithm.

Some sort of qualitative indicator about the tracked aircraft flight can be proposed that can be subsequently used as the classification backbone for the ANN and for the labelling algorithm. This research entertains several possible approaches to this method, which are closely examined in the following section.

5.2.2 Segment Classification for Labelling

One possible classification method explored while conducting the research is the minimal turn radius of the tracked aircraft flight within one segment. Figure 52 shows the logic behind this metric. Track segments are then represented by the smallest turn radius present in that segment. The turn radius is calculated for three subsequent plots, for all the groups of three in the segment excluding the first and last plot of the segment. This approach however is not ideal since it is very sensitive to noise present in the dataset, and the classification is deemed insufficient for the application of this research. The code processing of this approach is shown in the figure 53.

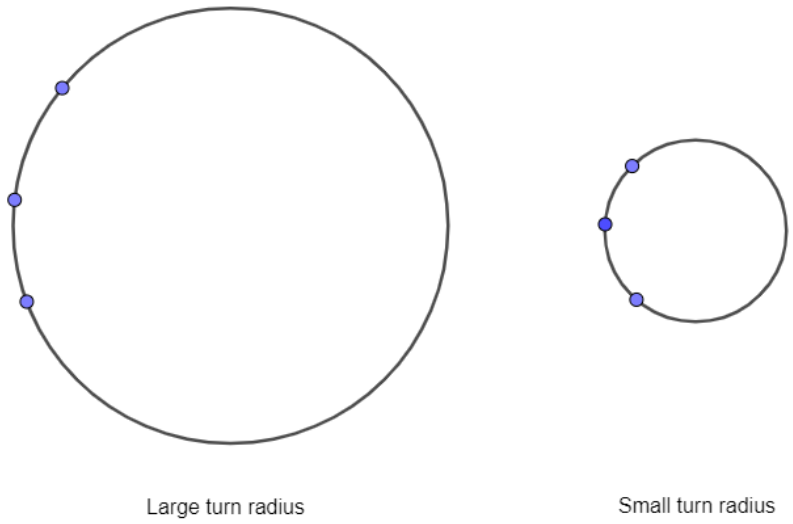

Figure 52: Figure showing the difference between a track with large turnradius, and a small turnradius.

The tracked aircraft track segment standard deviation from the shortest distance between the first and last plot (forming a linear function) is examined as one of the possible metrics for the track classification. The distribution of track segments can be seen in the histogram 54. Standard deviation is defined as the squared root of variance. Variance is defined as the squared differences from the mean. the formula for variance is shown in figure 24.

```
def findcircle (lat1, lon1, lat2, lon2, lat3, lon3):
    lat12 = lat1 - lat2;
    lat13 = lat1 - lat3;

    lon12 = lon1 - lon2;
    lon13 = lon1 - lon3;

    lon31 = lon3 - lon1;
    lon21 = lon2 - lon1;

    lat31 = lat3 - lat1;
    lat21 = lat2 - lat1;

    slat13 = pow(lat1, 2) - pow(lat3, 2);

    slon13 = pow(lon1, 2) - pow(lon3, 2);

    slat21 = pow(lat2, 2) - pow(lat1, 2);
    slon21 = pow(lon2, 2) - pow(lon1, 2);

    f = (((slat13) * (lat12) + (slon13) *
         (lat12) + (slat21) * (lat13) +
         (slon21) * (lat13)) // (2 *
         ((lon31) * (lat12) - (lon21) * (lat13))));

    g = (((slat13) * (lon12) + (slon13) * (lon12) +
         (slat21) * (lon13) + (slon21) * (lon13)) //
         (2 * ((lat31) * (lon12) - (lat21) * (lon13))));

    c = (-pow(lat1, 2) - pow(lon1, 2) -
        2 * g * lat1 - 2 * f * lon1);

    h = -g;
    k = -f;
    sqr_of_r = h * h + k * k - c;

    return np.sqrt(sqr_of_r);
```

Figure 53: Section of the code in charge of calculating the turnradius of three subsequent plots.

$$S^2 = \frac{\sum(x_i - \bar{x})}{n - 1} \tag{24}$$

Where:
S is the sample variance
x_i is the value of a single measurement
$\bar{x}T$ is the mean of all measurements
n is the number of measurements

This approach performed better than the turn radius as a metric approach, however, it is not selected as the classification metric for this research due to its poorer performance when compared to the following classification metric.

The best-performing metric for this research is considered to be the distance comparison between

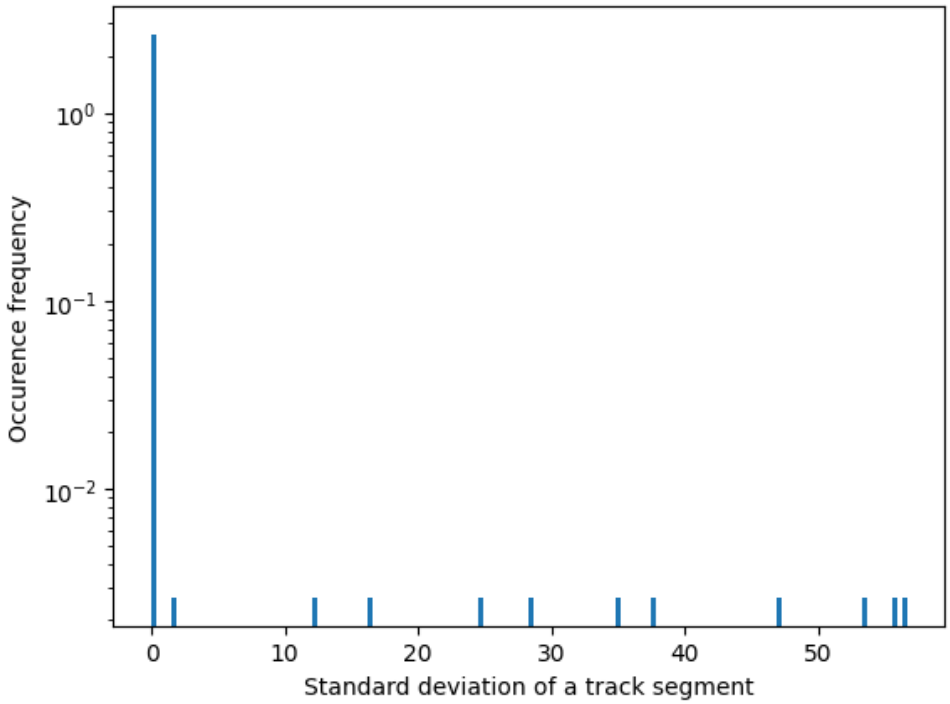

Figure 54: Histogram showing the frequency of standard deviation values. This histogram shows that most of the track segments are approaching a standard deviation of 0, meaning that the aircraft is flying straight, without a change in heading.

the tracks covered distance, and the shortest distance between the first and last plot of a segment. Figure 55 explains the logic behind this classification metric definition.

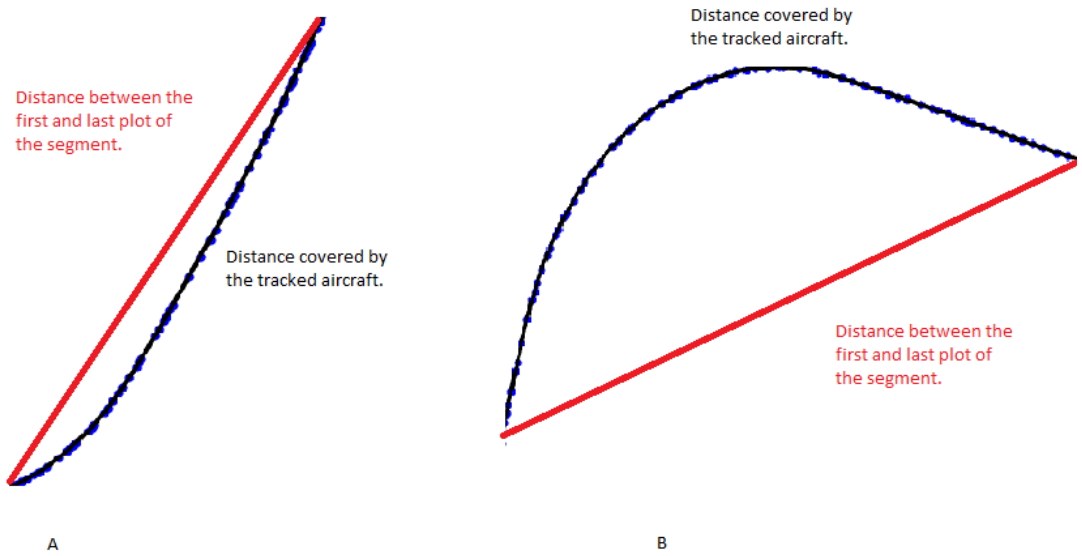

Figure 55: Logic between the selected classification metric. Track A is considered to be classified as easier to be tracked due to the difference in distance covered by the aircraft and the shortest distance between the first and last plot of the segment is lower, than the difference of the track B.

The classification of track segments is defined by the boundaries of the ratio between the distance covered by the aircraft and the shortest distance between the first and last plot of the segment. Boundaries set for the classification are set based on the distribution of ratios throughout the

dataset. The lower boundary is set to a ratio of 1.003, and the upper boundary is set to a ratio of 1.04. Segments with a ratio below the lower boundary of 1.003 are assigned to a class "0". Segments with a ratio between the lower boundary of 1.003 and upper boundary of 1.04 are assigned to a class "1", and the segments with a ratio above the upper boundary of 1.04 are assigned to a class "2". Code shown in the figure 56 deals with the shortest distance between the first and last plot of a segment, the ratio calculation between the shortest distance and the actual distance covered by the aircraft and with the segment classification based on the boundaries defined.

```
shortest_distance = calculate_distance(segment["lat"].iloc[0], segment["lon"].iloc[0], segment["lat"].iloc[-1], segment["lon"].iloc[-1])

ratio = distance/shortest_distance

label = 1

if ratio < 1.003:
    label = 0
if ratio > 1.04:
    label = 2
```

Figure 56: Code segment tasked with the shortest distance between the first plot and the last plot of the track segment calculation, the ratio calculation and with the segment classification.

Using this distance *ratio* classification metric, the track segments are divided into classes, each saved into separate folders representing the segment class as seen in the figure 57.

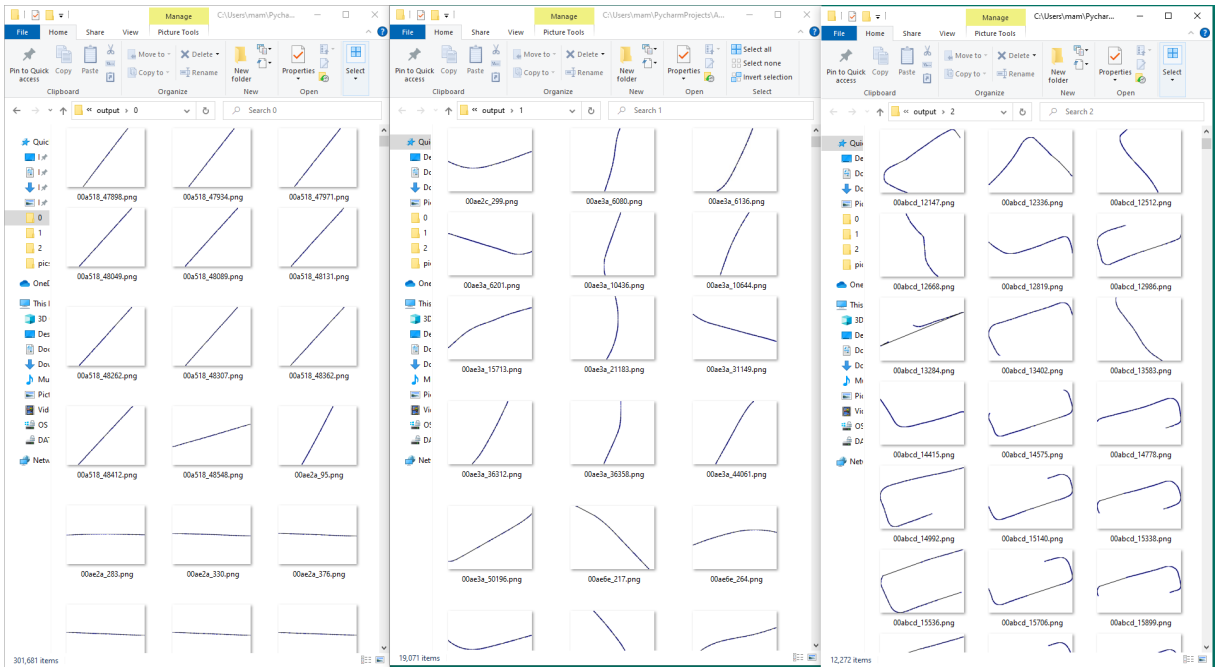

Figure 57: Figure showing the classes folders with segments loaded into the corresponding directory.

With the track segments sorted into their corresponding classes, the dataset is ready to be used for ANN training and testing.

5.3 ARTIFICIAL NEURAL NETWORK SETUP

A gentle introduction to artificial neural networks has already been presented in this thesis. The following part shows the application of selected neural networks to the problem at hand. There are two ANNs trained, tested and compared to each other so that there can be a conclusion made on which is more suitable for the goal of this thesis in mind.

5.3.1 Multi-Layer Perceptron Training Principles

MLP (Multi-Layer Perceptron) is a variant of an ANN that has all of its layers *fully connected*. This means that each neuron in a layer is connected to each layer of the preceding and the following layer in the neural network. There is a number of python libraries that can be utilized for the construction of an MLP. For this research, the Scikit Learn python library is used due to its relative ease of use, and amount of configuration provided to the user.[89]

The data flow throughout the neural network is divided into forward propagation, and backpropagation. The steps for forward propagation are explained in the figures below. The first step is multiplying the input value with the assigned weight of the path connecting the neurons, and then summing all those multiplied values together as shown in 25. Weight influences the value influence on the receiving neuron. The $\sum$ is also equal to the dot product x.w. [90]

$$\sum = (x_1 \times w_1) + (x_2 \times w_2) + ... + (x_n \times w_n) \tag{25}$$

Where:
x_i is the input value
w_i is the weight assigned to the path connecting the neurons
n is the number of connections between the neuron and the previous layer

The next step is adding the dataset bias to the sum of the values multiplied. Bias is necessary to fit the dataset to the activation function used by the neuron. The equation 26 shows the bias application.[90]

$$I = \sum + b \tag{26}$$

Where:
I is the input value adjusted by a bias
b is the bias
$\sum$ is the sum of values multiplied by weight

Third and the last step is to pass this I value through a selected activation function. Activation functions have a significant effect on the neural network performance since the *compress* the input values to a smaller scale easier manageable by the neural network. The example in 27 shows the application of a sigmoid application function.[90]

$$\hat{y} = \sigma(I) = \frac{1}{1 + e^{-I}} \tag{27}$$

Where:
$\hat{y}$ is the output value
σ is the sigmoid activation function
I is the input value adjusted by a bias

Backpropagation is used for the training of the MLP (or any other ANN). In simple terms, backpropagation is used to compute the gradient of the loss function of the MLP considering the weights. There are two main steps to backpropagation in ANN. The first step is to figure out how far is the estimate returned by the ANN from the actual desired values. Usually, the MSE (Mean Square Error) is used for the Loss function in regression problems, however, in the case of this research, a *Cross Entropy* loss function is used since it is dealing with a classification problem. Equation 28 shows how the Cross-Entropy loss function is calculated when there are more than 2 classes (non-binary classification).[91]

$$CE = -\sum_{c=1}^{\infty} y_{o,c} log(p_{o,c}) \tag{28}$$

Where:
M is the number of classes
c is the is the correct classification for o
o is the observation
y is the indicator for the "correctness" of c for o
p is the probability predicted of the o being the class c

The entire training set has to be accounted for for the loss function. The average of the loss function for the entire training set is called the Cost function $\boldsymbol{C}$.

To improve the performance of the ANN, the biases and weights are to be adjusted. This requires the knowledge of how the Cost function changes when the biases and weights are adjusted. The gradient is used just for that purpose. The equation 29 shows how the weight affects the Cost function, and equation 30 shows how the bias affects the cost function.[90]

$$\frac{\partial C}{\partial w_i} = \frac{2}{n} \times \sum (y - \hat{y}) \times \sigma(I) \times (1 - \sigma(I)) \times x_i \tag{29}$$

Where:
C is the Cost function
w is the weight
y is the actual value
$\hat{y}$ is the predicted value
I is the input value adjusted by a bias

x is the input value

$$\frac{\partial C}{\partial b} = \frac{2}{n} \times \sum (y - \hat{y}) \times \sigma(I) \times (1 - \sigma(I)) \tag{30}$$

Where:
C is the Cost function
b is the bias
y is the actual value
$\hat{y}$ is the predicted value
I is the input value adjusted by a bias
x is the input value
σ is the sigmoid activation function

With the knowledge of how the weights and biases affect the loss function (performance) of the ANN, it allows us to optimize the weights and biases to minimize the Loss function. Optimization is the search for the best set of weights and biases for the particular dataset classification. Gradient descent is the most widely used optimization algorithm and is also employed in our case. *Learning rate* α is a parameter that controls how much the optimization algorithm changes the values of weight and bias with each iteration. The equation 31 shows how the weight is adjusted to minimise the loss function, and the equation 32 shows how the bias is adjusted to minimise the loss function.[90]

$$w_i = w_i - (\alpha \times \frac{\partial C}{\partial w_i}) \tag{31}$$

Where:
C is the Cost function
w is the weight
α is the learning rate

$$b = b - (\alpha \times \frac{\partial C}{\partial b}) \tag{32}$$

Where:
C is the Cost function
b is the bias
α is the learning rate

5.3.2 MLP Setup

Using the Scikit Learn python library, the setup of the MLP is done using specified parameters. Some of the key parameters that can be adjusted to improve the MLP performance for a specific

dataset are described in this section.

The first step after loading the labelled dataset normalises the number of data in each class. This step ensures that there is no class *bias*. In our case, class "0" contains significantly more segments than class "1" and class "2". This would mean that the class "0" segments would be over-represented in the training set, and the trained MLP would tend to classify other classes as the class "0". The next step is splitting the dataset into a *training* and a *testing* dataset. For our purposes, the entire dataset is split into 80% training and 20% testing. Splitting the dataset into training and testing subsets prevents the occurrence called *overfitting*. Overfitting describes the case of having an ANN well-trained for the dataset that is used for the training of the ANN. By splitting the dataset, the system can be validated with dataset that it has not seen before.[92]

All segment image sizes are normalized, meaning there is no normalization needed at this point. Having images with the same size is important since the number of input layer neurons depends on the number of pixels in that image. Having a high-definition image means having a more complex MLP. This means minimising the number of pixels without removing key features from the image is important. The images in the dataset used do carry important features that can only be retrieved from an image with high resolution and would disappear if the pixel count was decreased. The resolution of images used as input for the MLP in this research is 640x480p.

The number of hidden layers affects the performance of the MLP. The number of hidden layers that are implemented should correspond with the dimensionality and complexity of the dataset. For the MLP examined the chosen number of hidden layers is 100. The solver used for the training of the CNN is chosen "adam" which is the stochastic gradient descent solver.

Selecting an activation function is crucial for MLP performance. Poorly selected activation functions used throughout the MLP can completely invalidate the output even with the rest of the MLP set up optimally. In the dataset used, the values of pixels don´t reach negative values, meaning the ReLU activation function can be applied to the hidden layers. The ReLU activation function is a linear function, meaning the input is mapped as the output. Another possible candidate for the activation function used in this research is the sigmoid function. The sigmoid function also allows for mapping just positive values (just like the ReLU activation function), however, it decreases the dataspace used, whereas the ReLU function does not. For the output layer, the ReLU is not suitable since the output should be a definitive class. For that reason, a Softmax activation function is applied in the output layers.

5.3.3 Convolutional Neural Network Training Principles

CNN's (Convolutional Neural Networks) have a different way of classification when compared to other ANNs. The biggest difference is that the layers are not always fully connected to the previous or following layers. CNN's layers are divided into three main types. The convolutional layer is tasked with the extraction of certain features from the input dataset. The neurons in a such layer are not connected to all the neurons from the previous layer, but only to a specified number of *neighbouring* neurons from the previous layer. *Kernels* are filters that are a part of a CNN, that are tasked with the extraction of features from the dataset. The math behind the convolution using a kernel is shown in the equation 33. To put it simply, the input data is split

into smaller sub-parts that are then convolved by the kernel. The output from a convolution layer is inherently location-dependent. This poses a problem in terms of the calculation power needed for a convolution neural network to conduct the classification. Figure 58 illustrates the CNN architecture. [93]

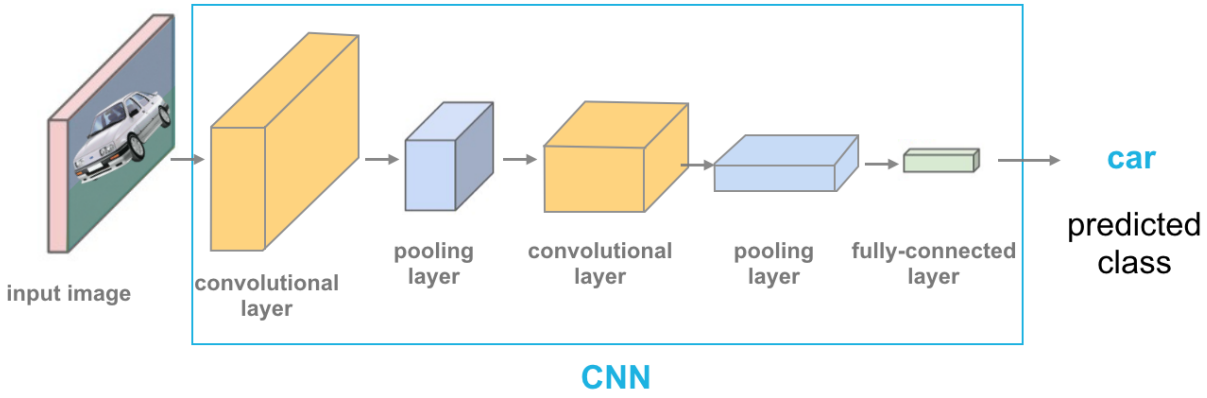

Figure 58: CNN diagram explaining the algorithm. [94]

$$Z = X * f \tag{33}$$

Where:
Z is the output from the Convolution
X is the input data
f is the filter (kernel)

Pooling layers are employed to solve this potential issue. Pooling is very similar to convolution in its principle. Pooling employs a filter that "travels" across the output from the convolutional layer, and the output depends on the pooling technique selected for the specific case, usually *Max Pooling* or *Average Pooling*. The pooling layer reduces the dimension of the output from the convolutional layer. The backpropagation of CNN is quite different. The Bias optimization is the same as with the MLP, however, there is also a need to optimize the filters that are used for the convolution. The principle however is the same as with the optimization in MLP, the optimization algorithm adjusts the values in the kernels so that the output matches the expected value of classification. For all the filters in the CNN, the equation for optimization using backpropagation is shown in 34. The equation for optimization of all the inputs in the CNN using backpropagation is shown in 35. [95] [96]

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{L}{\partial O_k} * \frac{\partial O_k}{\partial F_i} \tag{34}$$

Where:
F is the filter (kernel)
O is the convolution output

L is the loss function

$$\frac{\partial L}{\partial X_i} = \sum_{k=1}^{M} \frac{L}{\partial O_k} * \frac{\partial O_k}{\partial X_i} \tag{35}$$

Where:
X is the input to the convolutional layer
O is the convolution output
L is the loss function

5.3.4 CNN Setup

Setting up a CNN for the purpose of this research is considered more challenging when compared to the setup of MLP. The main reason is that unlike the MLP, in which all the hidden layers work in a similar fashion, the CNN has different types of Hidden layers. For the purpose of this research, the python library Tensorflow is used for the coding of the CNN.[97]

The final CNN setup is inspired by the setup used in [97], and is shown in figure 59. The dataset used for the training of the CNN is the same as for the MLP training.

There are two data splits explored for the training/testing data distribution, 80/20 and 70/30. Differences in CNN performance based on the data split are shown in a later section.

There are three convolution layers, all having a ReLU activation function assigned. The dense layer is the last, fully connected layer that is tasked with the final output. Re-scaling is used to convert the input images from colour pixels into greyscale so the features are more defined.

The optimizer used is "adam" which stands for stochastic gradient descent and the number of epochs is set to 10. The number of epochs is picked with regard to the computing power and time needed for the CNN training. It is possible to increase the number of epochs and achieve possibly higher accuracy, however, the increase in training time would not be justifiable.

The number of epochs representing the number of training iterations with the whole dataset being fed into the CNN is 10. The influence of adjusting the value of epochs is explored in a later section.

```
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 rescaling (Rescaling)       (None, 240, 320, 3)

 conv2d (Conv2D)             (None, 240, 320, 16)

 max_pooling2d (MaxPooling2D  (None, 120, 160, 16)
 )

 conv2d_1 (Conv2D)           (None, 120, 160, 32)

 max_pooling2d_1 (MaxPooling  (None, 60, 80, 32)
 2D)

 conv2d_2 (Conv2D)           (None, 60, 80, 64)

 max_pooling2d_2 (MaxPooling  (None, 30, 40, 64)
 2D)

 flatten (Flatten)           (None, 76800)

 dense (Dense)               (None, 128)

 dense_1 (Dense)             (None, 3)

=================================================================
```

Figure 59: The final CNN architecture used for the track classification.

5.4 ANN PERFORMANCE INDICATORS

To determine if the trained ANN performs to a degree that is acceptable, there are numerous indicators are used to evaluate ANN performance. Performance indicators, some of which are used to evaluate the ANNs examined in this research are explained in the following section.

Accuracy of the classification is a simple metric showing the ratio of such predictions that are correct to those that were incorrect. The calculation of accuracy is depicted in 36.[98] Training accuracy is the accuracy of the classification while training and validation accuracy is the accuracy calculated for the testing subset.

$$A = \frac{Cl_c}{Cl} \tag{36}$$

Where:
A is the Accuracy
Cl_c is the number of correct classifications
Cl is the number of classifications

Loss is a metric that represents the ANNs ability to fit the input data. Like the accuracy, the loss is also determined for the training and testing subset. High Loss represents the ANN´s high dependency on the dataset used for training. Low Loss means that the trained ANN performs well with data that it has not been presented with while training.

Precision is defined for binary classification problems. The calculation of Precision is shown in 38. Here, the terms *True Positive* and *False Positive* occur. True Positive Classification is a metric that shows the number of data that were classified as a specific class, and actually are part of that class. False Positive is an indicator that shows the number of data that were classified as a class, however, they do not belong to that class. [99]

$$Pr = \frac{TP}{TP + FP} \tag{37}$$

Where:
Pr is the Precision
TP is the number of True Positive classifications
FP is the number of False Positive classifications

Recall is an indicator that shows the ratio of positive classifications that were classified correctly. The recall calculation is shown in 38. There is a new term called False Negative present. A false negative count is used for the number of data that belongs to a specified class, however, is not classified as such. [99]

$$Re = \frac{TP}{TP + FN} \tag{38}$$

Where:
Re is the Recall
TP is the number of True Positive classifications
Fn is the number of False Negative classifications

ROC (receiver operating characteristic curve) is an indicator that is comprised of two other indicators, the True Positive rate and the False Positive Rate. The ROC curve is shown in figure 60. The goal of the classifier training is to maximise the AUC (area under the ORC curve). [100]

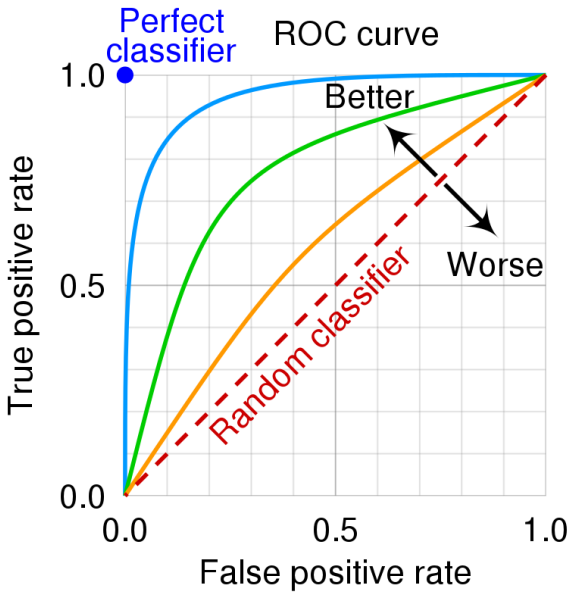

Figure 60: Different ROC curves for classifiers. The closer the curve to the upper left corner of the graph, the better the performance of the classifier. [101]

If the Precision and Recall are Combined, the performance metric called *F1 score* is the outcome. It is the harmonic mean (average) of precision and recall to be exact. F1 score is calculated using the formula 39. Classification will be awarded a high F1 score if both the Recall and Precision are high. On the other hand, if both Recall and Precision are low, the F1 score will be low as well. [102]

$$F1score = 2x\frac{Pr}{Re} \tag{39}$$

Where:
Re is the Recall
Pr is the precision

Another way to evaluate the performance of a classification algorithm is employing a so-called *Confusion Matrix*. It is a NxN matrix with the N standing for the number of classes. The matrix displays the classification performance, i.e. to which class data belongs, and where it is classified by the algorithm. The confusion matrix defined for this thesis is shown in figure 61. A confusion matrix is easily implemented for multi-class classification. [103]

		Predicted		
		0	1	2
Actual	0			
	1			
	2			

Figure 61: Confusion matrix with three classes used in this research.

5.5 MLP AND CNN PERFORMANCE COMPARISON

With two different ANN approaches used for the track classification segments, comparing the two shows which are more suitable for the given task. The performance difference is significant and is further explained in this section.

5.5.1 MLP Performance

MLP proved to be insufficient in the task of track classification. Even with several iterations with different parameters and training dataset sizes, the MLP performance of track classifications is not far from a system that would classify segments randomly (in terms of accuracy percentage). The influence of the training dataset size on the MLP performance is seen in the table 13. Three dataset sizes were tested, 300 segments with 100 per class, and 3000 segments with 1000 per class. And 30000 segments with 10000 per class. The confusion matrix of the proposed MLP is shown in figure 62. The distribution throughout the confusion matrix remained the same, regardless of the training dataset size or MLP parameters. The confusion matrix in figure 62 is created based on the MLP being trained with 1000 input track segments with an 80/20 ratio of training/testing dataset. The confusion matrix shows that the trained MLP classified most track segments as a class "0". This does not change with different dataset sizes or other MLP setup changes, such as the number of epochs or training/testing split ratio.

MLP performance based on the dataset size			
Number of segments	300	3000	30000
Validation Accuracy	36.663	37.522	36.996
Training Accuracy	42.456	36.550	37.256

Table 13: The validation accuracy of the trained MLP based on the size of the input dataset.

MLP confusion matrix				
		Predicted class		
		Class "0"	Class "1"	Class "2"
Actual class	Class "0"	956	3	41
	Class "1"	912	82	6
	Class "2"	802	173	25

Figure 62: Confusion matrix of the MLP trained with 3000 segments in the dataset.

Both the accuracy and confusion matrix did not change significantly with different MLP setups and dataset sizes. The conclusion is made that the MLP training and classification is not usable for the goal of the research, and the focus is shifted to CNN, which is expected to perform better.

5.5.2 CNN Performance

With the CNN setup presented in a previous section, the performance met the expectations even with the relatively small size of the training dataset. Performance only increased with the size of the training dataset. The influence of the training dataset size on the accuracy is examined later in this section. The performance of the CNN begins at a relatively high level even from the

first EPOCH and increases with each additional EPOCH until a certain number of epochs. The Performance increase with each epoch is shown in figure 65. Results of Training and Validation (testing) accuracy are shown in figure 63, and Results of Training and Validation (testing) loss are shown in figure 64.

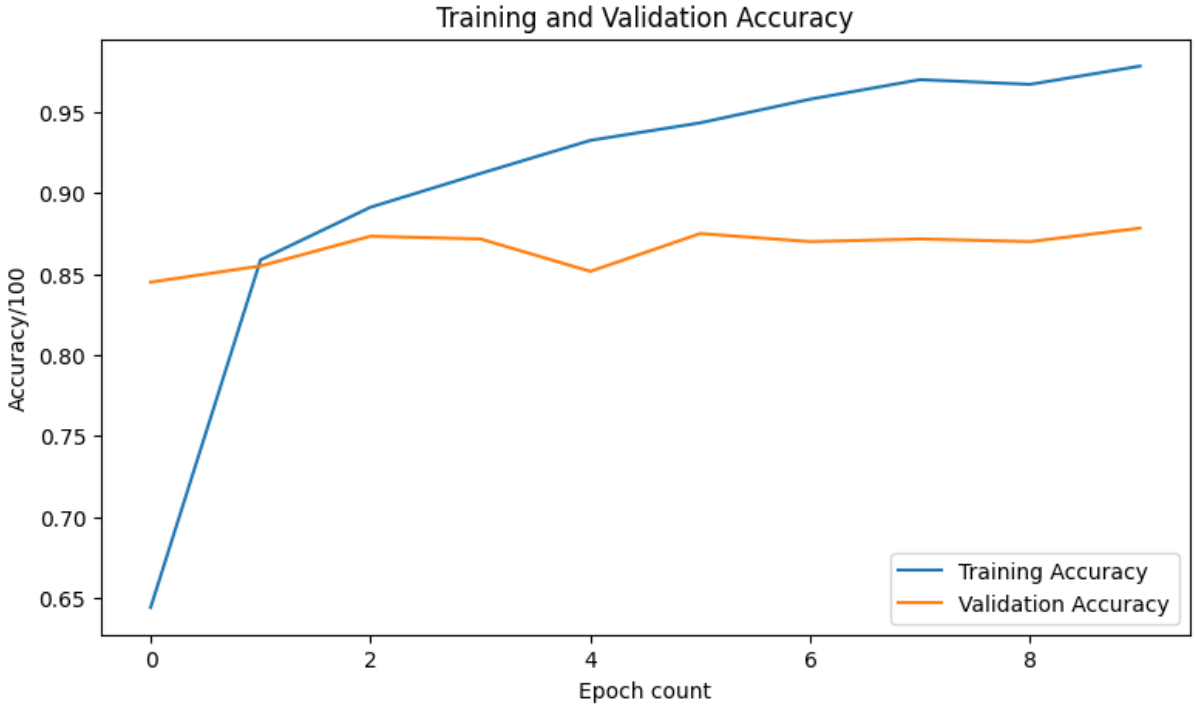

Figure 63: Accuracy of the training and validation set with the training/testing ration of 80/20.

Figure 64: Loss of the training and validation set with the training/testing ration of 80/20.

```
Epoch 1/10
75/75 [==============================] - 129s 2s/step - loss: 0.8578 - accuracy:
0.6750 - val_loss: 0.4309 - val_accuracy: 0.8400
Epoch 2/10
75/75 [==============================] - 108s 1s/step - loss: 0.3688 - accuracy:
0.8637 - val_loss: 0.3689 - val_accuracy: 0.8717
Epoch 3/10
75/75 [==============================] - 111s 1s/step - loss: 0.2805 - accuracy:
0.8933 - val_loss: 0.3967 - val_accuracy: 0.8733
Epoch 4/10
75/75 [==============================] - 124s 2s/step - loss: 0.2231 - accuracy:
0.9204 - val_loss: 0.4501 - val_accuracy: 0.8617
Epoch 5/10
75/75 [==============================] - 142s 2s/step - loss: 0.1662 - accuracy:
0.9446 - val_loss: 0.5013 - val_accuracy: 0.8650
Epoch 6/10
75/75 [==============================] - 132s 2s/step - loss: 0.1434 - accuracy:
0.9525 - val_loss: 0.4867 - val_accuracy: 0.8700
Epoch 7/10
75/75 [==============================] - 144s 2s/step - loss: 0.1301 - accuracy:
0.9571 - val_loss: 0.5321 - val_accuracy: 0.8733
Epoch 8/10
75/75 [==============================] - 127s 2s/step - loss: 0.0942 - accuracy:
0.9700 - val_loss: 0.5551 - val_accuracy: 0.8700
Epoch 9/10
75/75 [==============================] - 126s 2s/step - loss: 0.0783 - accuracy:
0.9787 - val_loss: 0.5778 - val_accuracy: 0.8750
Epoch 10/10
75/75 [==============================] - 139s 2s/step - loss: 0.0642 - accuracy:
0.9821 - val_loss: 0.5853 - val_accuracy: 0.8733
```

Figure 65: Accuracy development for each epoch with the training/testing ration of 80/20.

Another training iteration is conducted with an adjusted ratio of training/testing data distribution to 70/30. The results for training and validation (testing) accuracy are shown in figure 66. The result for training and validation (testing) loss are as shown in figure 67. The increase in accuracy with each epoch is shown in figure 68.

Results show that the Validation Loss deteriorates with the training/testing ratio setup as 70/30. The validation accuracy does not experience major improvements, the first distribution of 80/20 is considered more suitable for the CNN training.

To examine the influence of epochs on the classification capability, the EPOCH count is increased to 20. Results for training and validation(testing) accuracy are shown in figure 69. Results for training and validation (testing) loss are shown in figure 70.

Figure 69 shows the validation accuracy does not change significantly with the epoch count, and after epoch 20 it remains almost constant. Figure 70 shows that with the increasing epoch, the validation loss dramatically rises. This trend represents the tendency of overfitting the CNN to the dataset used for the training. The epoch count of 10 is more favourable for the goal of this research.

So far the track segment number is 3000, with 1000 track segments per class. As already mentioned, the size of the dataset used for the ANN training significantly influences the performance of the trained CNN. To illustrate, figure 71 shows the training and validation (testing) accuracy for a

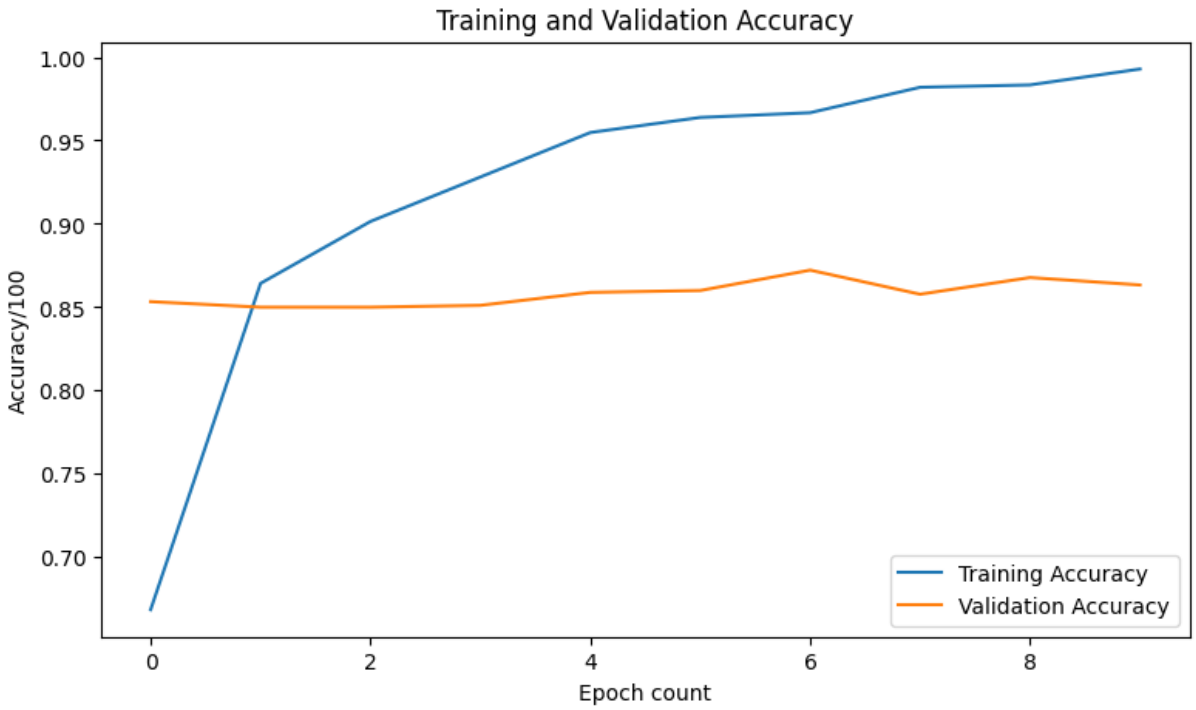

Figure 66: Accuracy of the training and validation set with the training/testing ration of 70/30.

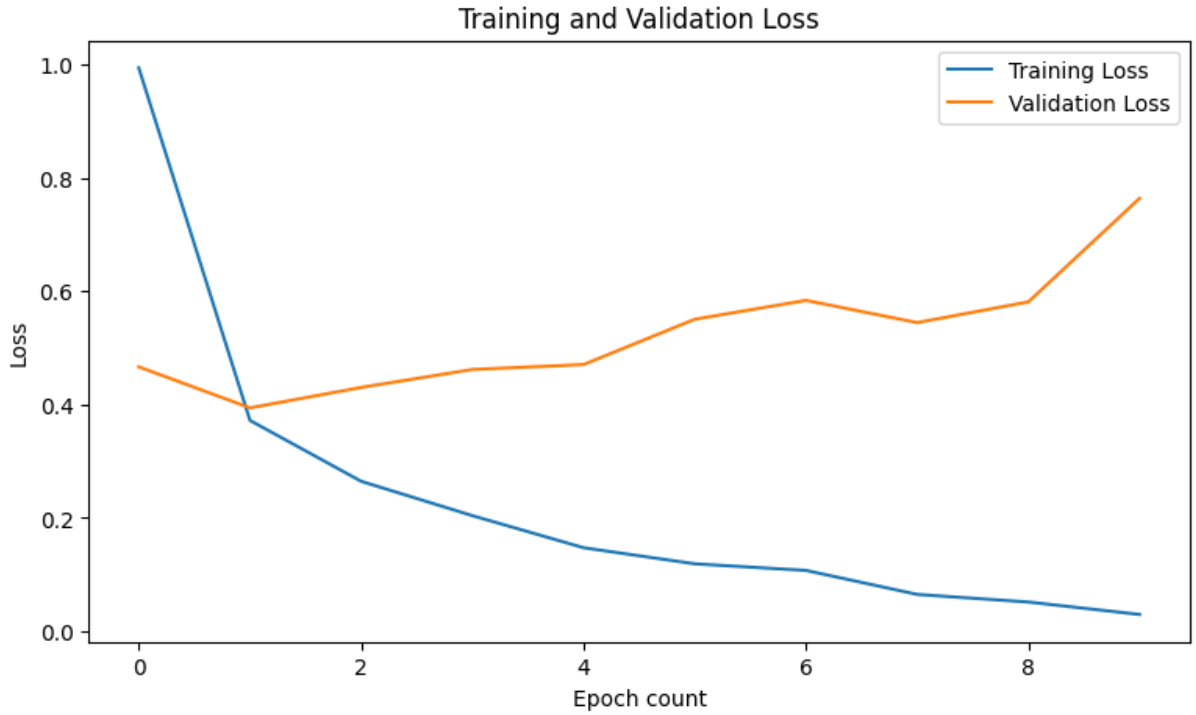

Figure 67: Loss of the training and validation set with the training/testing ration of 70/30.

CNN that is trained with 300 track segments, 100 per class. Figure 72 shows the training and validation (testing) loss for the same dataset.

As seen in the figure 71. The validation accuracy is negatively affected by the reduced size of the training and validation dataset. Figure 72 shows that the reduced dataset volume also negatively

```
Epoch 1/10
66/66 [==============================] - 92s 1s/step - loss: 0.9946 - accuracy: :
0.6686 - val_loss: 0.4669 - val_accuracy: 0.8533
Epoch 2/10
66/66 [==============================] - 93s 1s/step - loss: 0.3723 - accuracy: :
0.8643 - val_loss: 0.3944 - val_accuracy: 0.8500
Epoch 3/10
66/66 [==============================] - 94s 1s/step - loss: 0.2648 - accuracy: :
0.9014 - val_loss: 0.4305 - val_accuracy: 0.8500
Epoch 4/10
66/66 [==============================] - 95s 1s/step - loss: 0.2042 - accuracy: :
0.9281 - val_loss: 0.4623 - val_accuracy: 0.8511
Epoch 5/10
66/66 [==============================] - 100s 2s/step - loss: 0.1476 - accuracy:
0.9548 - val_loss: 0.4709 - val_accuracy: 0.8589
Epoch 6/10
66/66 [==============================] - 97s 1s/step - loss: 0.1192 - accuracy: :
0.9638 - val_loss: 0.5509 - val_accuracy: 0.8600
Epoch 7/10
66/66 [==============================] - 101s 2s/step - loss: 0.1075 - accuracy:
0.9667 - val_loss: 0.5842 - val_accuracy: 0.8722
Epoch 8/10
66/66 [==============================] - 93s 1s/step - loss: 0.0651 - accuracy: :
0.9819 - val_loss: 0.5449 - val_accuracy: 0.8578
Epoch 9/10
66/66 [==============================] - 91s 1s/step - loss: 0.0517 - accuracy: :
0.9833 - val_loss: 0.5816 - val_accuracy: 0.8678
Epoch 10/10
66/66 [==============================] - 90s 1s/step - loss: 0.0298 - accuracy: :
0.9929 - val_loss: 0.7645 - val_accuracy: 0.8633
```

Figure 68: Accuracy development for each epoch with the training/testing ration of 70/30.

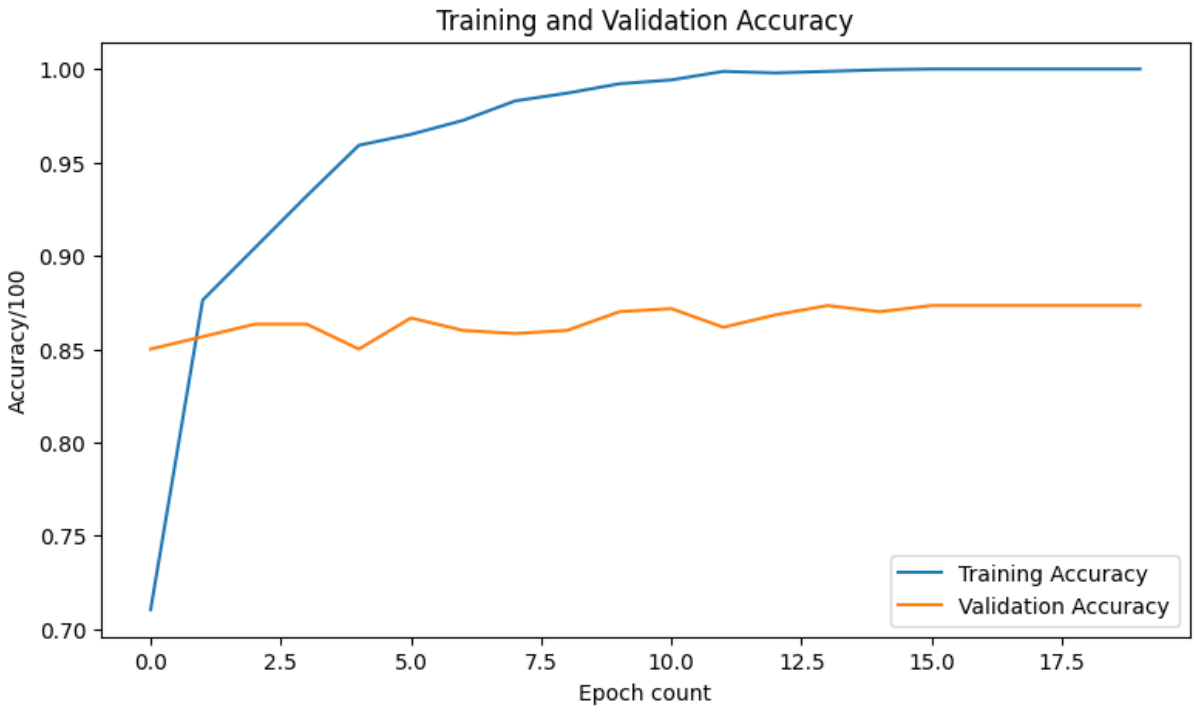

Figure 69: Accuracy of the training and validation set with the the epoch count of 20.

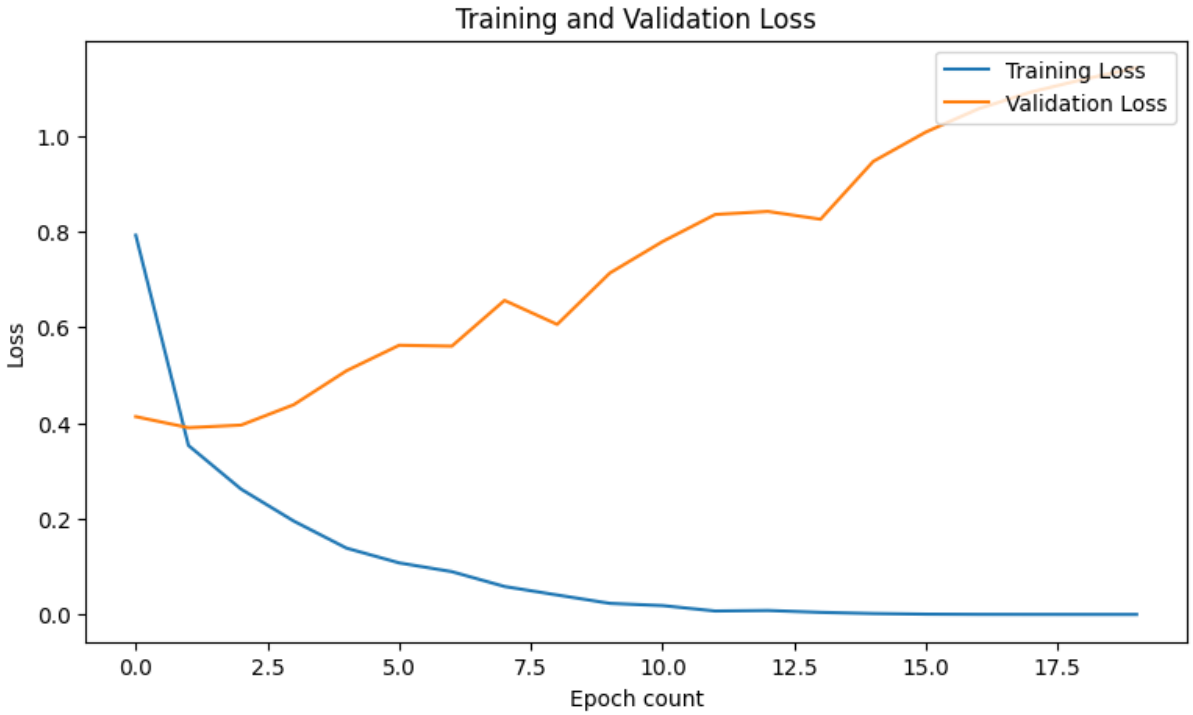

Figure 70: Loss of the training and validation set with the the epoch count of 20.

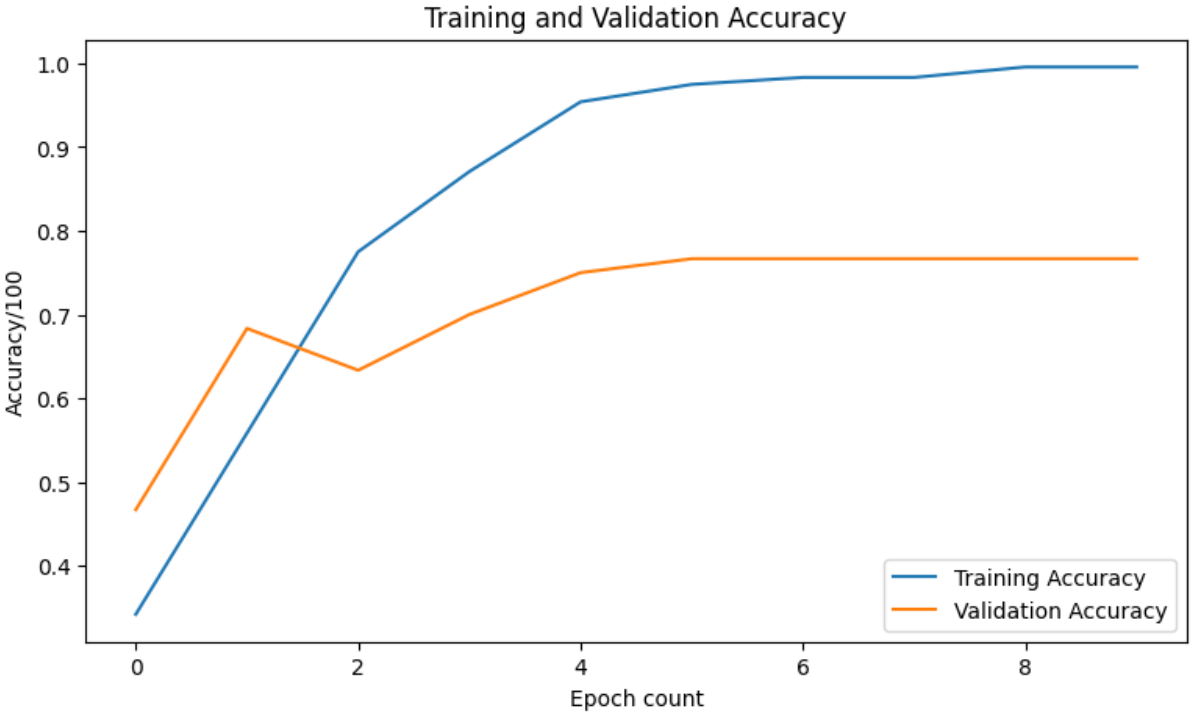

Figure 71: Accuracy of the training and validation set for track segment dataset size of 300.

affects validation loss. This makes the trained CNN not versatile to be able to classify data that is not used for the training.

The Confusion Matrix for the trained CNN with parameters defined based on the development explored above is shown in 73. To have 100% accuracy of the classification algorithm, only the

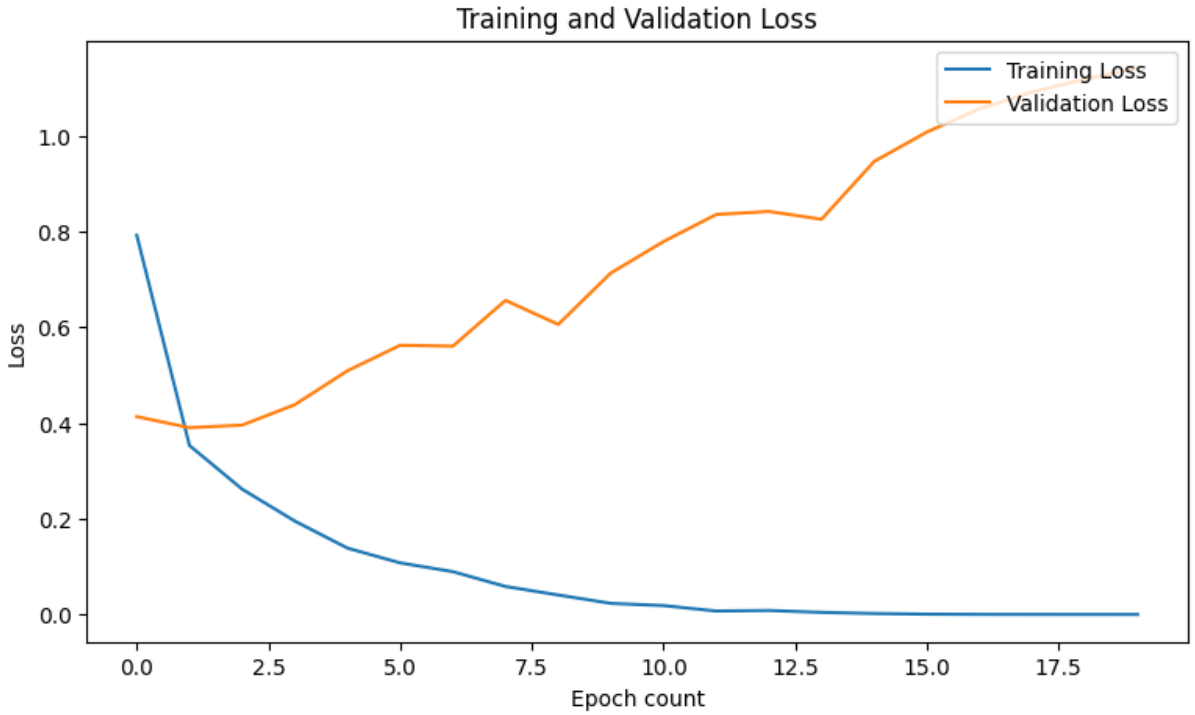

Figure 72: Loss of the training and validation set for track segment dataset size of 300.

diagonal of the confusion matrix has to be populated.

CNN confusion matrix				
		Predicted class		
		Class "0"	Class "1"	Class "2"
Actual class	Class "0"	861	83	56
	Class "1"	86	872	42
	Class "2"	28	93	879

Figure 73: Confusion matrix for the CNN.

CNN is considered to be the viable option for the goal of this research. It performed better in all areas of performance evaluation, and it achieved it without increasing the time required for the training and testing. The only drawback that can be considered for the CNN is the setup complexity of the CNN itself. Further improvements and limitations encountered while conducting the ANN development are explained in the discussion section.

6 DISCUSSION

The proposed CNN capable of track classification based on the motion dynamics does show high levels of performance. This section explores the limitations that prevent the proposed algorithm from achieving even higher performance. With the limitations defined, suggestions for further development of such algorithms are defined.

The most influential limitation defined when conducting this research is the computation power limitations. HW used for the ADS-B data manipulation and CNN training and validation can be considered as a lower-end specification when the HW´s performance is considered. The biggest drawback of the HW is the lack of a dedicated graphics card. Since the ANN is provided with track segments interpreted as images, a dedicated GPU would greatly reduce the time needed for the dataset to be propagated throughout the ANN. The time saving would allow for a larger dataset, which can be expected to improve the classification performance of the ANN.

This research considers the 2D positional data forming the dataset for track classification and for ANN training and validation. Extending the dimension of the dataset with other track information can be considered to further extend the capability of the classification. Such information can be the altitude of the track, the velocity of the track, the wake vortex category of the track and many more. Implementing this into further research would require more complex ANN and much more capable HW. Such ANN classification systems would not necessarily show higher performance in terms of accuracy, but they would allow for deeper integration into the track filtering algorithms.

Another approach for the data input would be utilizing the time series format instead of the image format used in this research. This would result in a more complex classification algorithm since the input layer dimension would have to be defined in such a way, that the lack of normalization caused by the time series format is accounted for. Such algorithms would require much more powerful HW to be trained and to be implemented.

7 CONCLUSION

The goal of developing an Artificial neural network capable of track classification with correlation to the track's motion dynamics is met. The proposed classification algorithm allows for the implementation of a system that would dynamically assign different track filtration algorithms based on the track flight behaviour.

The related field of Machine learning principles, aircraft surveillance techniques and track filtering algorithms are closely examined to provide the reader with an understanding of the environment for which this research is of value.

The process of ADS-B data manipulation covering the pre-processing and data filtration is explained. The track segmentation method suitable for the task of track classification is presented, and several track segment classification methods for labelling purposes are explored. Track segment classification based on the ratio of distance covered by the tracked aircraft and the shortest distance is chosen as the classification method based on which the labelling of the training and validation set for the ANN learning is conducted.

Two approaches to ANN algorithms are described and developed with the goal of track classification based on aircraft dynamics. Prepared labelled tracked segments are then used for the training of the proposed MLP and CNN.

Indicators used to evaluate the performance of classification are described and applied to both MLP and CNN. The results of the evaluations of the trained MLP and CNN are compared, with the CNN proving to be the optimal ANN algorithm for the track classification.

The results and possible further developments to improve the performance of the classification and extend the use of track classification by considering alternative classification methods are discussed.

List of Figures

List of Tables

List of Appendicies

References

1. AKINSOLA, J E T. Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends and Technology (IJCTT)*. 2017, vol. 48, pp. 128–138. Available from DOI: 10.14445/22312803/IJCTT-V48P126.
2. SINGH, Amanpreet; THAKUR, Narina; SHARMA, Aakanksha. A review of supervised machine learning algorithms. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016, pp. 1310–1315.
3. *2.1.1 Linear Classifiers - Machine Learning Notebook* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://sites.google.com/site/machinelearningnotebook2/classification/binary-classification/linear-classifiers.
4. YUAN, Guo-Xun; HO, Chia-Hua; LIN, Chih-Jen. Recent Advances of Large-Scale Linear Classification. *Proceedings of the IEEE* [online]. 2012, vol. 100, no. 9, pp. 2584–2603 [visited on 2022-11-26]. ISSN 0018-9219, ISSN 1558-2256. Available from DOI: 10.1109/JPROC.2012.2188013.
5. CYC. *Graphic showing 3 Hyperplanes in 2D. H3 doesn't separate the 2 classes. H1 does, with a small margin and H2 with the maximum margin.* [online]. 2008. [visited on 2022-11-27]. Available from: https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes.png.
6. WRIGHT, Raymond E.; ROSENBERG, Sheldon. Knowledge of text coherence and expository writing: A developmental study. *Journal of Educational Psychology* [online]. 1993, vol. 85, no. 1, pp. 152–158 [visited on 2022-11-26]. ISSN 1939-2176, ISSN 0022-0663. Available from DOI: 10.1037/0022-0663.85.1.152.
7. LAVALLEY, Michael P. Logistic Regression. *Circulation* [online]. 2008, vol. 117, no. 18, pp. 2395–2399 [visited on 2022-11-26]. ISSN 0009-7322, ISSN 1524-4539. Available from DOI: 10.1161/CIRCULATIONAHA.106.682658.
8. *Logistic Regression in Machine Learning - Javatpoint* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://www.javatpoint.com/logistic-regression-in-machine-learning.
9. HORNÝ, Michal. *Bayesian Networks*. Boston, 2014-04. Technical, 5. Available also from: https://www.bu.edu/sph/files/2014/05/bayesian-networks-final.pdf.
10. DEVROYE, Luc; GYÖRFI, László; LUGOSI, Gábor. *A probabilistic theory of pattern recognition*. New York: Springer, 1996. ISBN 978-0-387-94618-4.
11. MURPHY, Kevin P. Last updated October 24, 2006. [N.d.], p. 8.
12. HEARST, M.A.; DUMAIS, S.T.; OSUNA, E.; PLATT, J.; SCHOLKOPF, B. Support vector machines. *IEEE Intelligent Systems and their Applications*. 1998, vol. 13, no. 4, pp. 18–28. ISSN 2374-9423. Available from DOI: 10.1109/5254.708428. Conference Name: IEEE Intelligent Systems and their Applications.
13. WANG, Lipo. *Support Vector Machines: Theory and Applications*. Springer Science & Business Media, 2005. ISBN 978-3-540-24388-5. Google-Books-ID: uTzMPJjVjsMC.
14. *Support Vector Machine (SVM) Algorithm - Javatpoint* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.
15. *1.10. Decision Trees* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://scikit-learn/stable/modules/tree.html.
16. *Decision Tree* [online]. 2017. [visited on 2022-10-18]. Available from: https://www.geeksforgeeks.org/decision-tree/. Section: Advanced Computer Subject.
17. GUPTA, Prashant. *Decision Trees in Machine Learning* [online]. 2017. [visited on 2022-11-27]. Available from: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

18. YIU, Tony. *Understanding Random Forest* [online]. 2021. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.
19. SPEISER, Jaime Lynn; MILLER, Michael E.; TOOZE, Janet; IP, Edward. A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications* [online]. 2019, vol. 134, pp. 93–101 [visited on 2022-11-26]. ISSN 09574174. Available from DOI: 10.1016/j.eswa.2019.05.028.
20. GENTLEMAN, R.; CAREY, V. J. Unsupervised Machine Learning. In: *Bioconductor Case Studies* [online]. New York, NY: Springer New York, 2008, pp. 137–157 [visited on 2022-10-18]. ISBN 978-0-387-77239-4 978-0-387-77240-0. Available from DOI: 10.1007/978-0-387-77240-0_10.
21. USAMA, Muhammad; QADIR, Junaid; RAZA, Aunn; ARIF, Hunain; YAU, Kok-lim Alvin; ELKHATIB, Yehia; HUSSAIN, Amir; AL-FUQAHA, Ala. Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges. *IEEE Access*. 2019, vol. 7, pp. 65579–65615. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2019.2916648. Conference Name: IEEE Access.
22. MANNOR, Shie; JIN, Xin; HAN, Jiawei; JIN, Xin; HAN, Jiawei; JIN, Xin; HAN, Jiawei; ZHANG, Xinhua. K-Means Clustering. In: SAMMUT, Claude; WEBB, Geoffrey I. (eds.). *Encyclopedia of Machine Learning* [online]. Boston, MA: Springer US, 2011, pp. 563–564 [visited on 2022-10-18]. ISBN 978-0-387-30768-8 978-0-387-30164-8. Available from DOI: 10.1007/978-0-387-30164-8_425.
23. *K-Means Clustering Algorithm - Javatpoint* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning.
24. *ML — K-means++ Algorithm* [online]. 2019. [visited on 2022-11-27]. Available from: https://www.geeksforgeeks.org/ml-k-means-algorithm/. Section: Machine Learning.
25. FUERTES, T. *Hierarchical clustering, using it to invest Quantdare* [online]. 2016. [visited on 2022-11-27]. Available from: https://quantdare.com/hierarchical-clustering/.
26. PATLOLLA, Chaitanya Reddy. *Understanding the concept of Hierarchical clustering Technique* [online]. 2020. [visited on 2022-11-27]. Available from: https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec.
27. *Hierarchical Clustering in Machine Learning - Javatpoint* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://www.javatpoint.com/hierarchical-clustering-in-machine-learning.
28. *Understanding the concept of Hierarchical clustering Technique — by Chaitanya Reddy Patlolla — Towards Data Science* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec.
29. CARREIRA-PERPIÑÁN, Miguel Á. A review of mean-shift algorithms for clustering [online]. 2015 [visited on 2022-11-26]. Available from DOI: 10.48550/ARXIV.1503.00687. Publisher: arXiv Version Number: 1.
30. YUFENG. *Understanding Mean Shift Clustering and Implementation with Python* [online]. 2022. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/understanding-mean-shift-clustering-and-implementation-with-python-6d5809a2ac40.
31. KRIEGEL, Hans-Peter; KRÖGER, Peer; SANDER, Jörg; ZIMEK, Arthur. Density-based clustering. *WIREs Data Mining and Knowledge Discovery* [online]. 2011, vol. 1, no. 3, pp. 231–240 [visited on 2022-10-18]. ISSN 1942-4795. Available from DOI: 10.1002/widm.30. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.30.

32. *Density-based clustering in data minin - Javatpoint* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://www.javatpoint.com/density-based-clustering-in-data-mining.
33. BIRANT, Derya; KUT, Alp. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering* [online]. 2007, vol. 60, no. 1, pp. 208–221 [visited on 2022-10-18]. ISSN 0169023X. Available from DOI: 10.1016/j.datak.2006.01.013.
34. JAIN, A.K.; JIANCHANG MAO; MOHIUDDIN, K.M. Artificial neural networks: a tutorial. *Computer* [online]. 1996, vol. 29, no. 3, pp. 31–44 [visited on 2022-11-26]. ISSN 00189162. Available from DOI: 10.1109/2.485891.
35. HOPFIELD, J.J. Artificial neural networks. *IEEE Circuits and Devices Magazine* [online]. 1988, vol. 4, no. 5, pp. 3–10 [visited on 2022-11-26]. ISSN 8755-3996. Available from DOI: 10.1109/101.8118.
36. BRE, Facundo; GIMENEZ, Juan M.; FACHINOTTI, Víctor D. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings* [online]. 2018, vol. 158, pp. 1429–1441 [visited on 2022-11-27]. ISSN 03787788. Available from DOI: 10.1016/j.enbuild.2017.11.045.
37. PATTERSON, Cameron. *Managing a real/time massively/oarallel neural architecture.* 2012. PhD thesis.
38. NWANKPA, Chigozie; IJOMAH, Winifred; GACHAGAN, Anthony; MARSHALL, Stephen. Activation Functions: Comparison of trends in Practice and Research for Deep Learning [online]. 2018 [visited on 2022-11-26]. Available from DOI: 10.48550/ARXIV.1811.03378. Publisher: arXiv Version Number: 1.
39. SAMSON, Hasara. *Getting to know Activation Functions in Neural Networks.* [online]. 2020. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/getting-to-know-activation-functions-in-neural-networks-125405b67428.
40. SHARMA, SAGAR. *Activation Functions in Neural Networks* [online]. 2021. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.
41. *Hyperbolic Tangent Function - an overview — ScienceDirect Topics* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://www.sciencedirect.com/topics/mathematics/hyperbolic-tangent-function.
42. *A Gentle Introduction To Sigmoid Function* [online]. [N.d.]. [visited on 2022-10-18]. Available from: https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/.
43. SHARMA, SAGAR. *Activation Functions in Neural Networks* [online]. 2021. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.
44. SERENGIL, Sefik. *Hyperbolic Tangent as Neural Network Activation Function* [online]. 2017. [visited on 2022-10-18]. Available from: https://sefiks.com/2017/01/29/hyperbolic-tangent-as-neural-network-activation-function/.
45. BROWNLEE, Jason. *A Gentle Introduction to the Rectified Linear Unit (ReLU)* [online]. 2019. [visited on 2022-10-18]. Available from: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.
46. BROWNLEE, Jason. *How to Choose an Activation Function for Deep Learning* [online]. 2021. [visited on 2022-10-18]. Available from: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/.
47. BUSHAEV, Vitaly. *How do we 'train' neural networks ?* [online]. 2018. [visited on 2022-10-18]. Available from: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73.

48. BROWNLEE, Jason. *Loss and Loss Functions for Training Deep Learning Neural Networks* [online]. 2019. [visited on 2022-10-18]. Available from: `https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/`.
49. YATHISH, Vishal. *Loss Functions and Their Use In Neural Networks* [online]. 2022. [visited on 2022-10-18]. Available from: `https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9`.
50. *What is Gradient Descent? — IBM* [online]. [N.d.]. [visited on 2022-10-18]. Available from: `https://www.ibm.com/cloud/learn/gradient-descent`.
51. DU, Simon; LEE, Jason; LI, Haochuan; WANG, Liwei; ZHAI, Xiyu. Gradient Descent Finds Global Minima of Deep Neural Networks. In: *Proceedings of the 36th International Conference on Machine Learning* [online]. PMLR, 2019, pp. 1675–1685 [visited on 2022-10-18]. Available from: `https://proceedings.mlr.press/v97/du19c.html`. ISSN: 2640-3498.
52. SRINIVASAN, Aishwarya V. *Stochastic Gradient Descent — Clearly Explained !!* [online]. 2019. [visited on 2022-10-18]. Available from: `https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31`.
53. BOTTOU, Léon. Stochastic Gradient Descent Tricks. In: MONTAVON, Grégoire; ORR, Geneviève B.; MÜLLER, Klaus-Robert (eds.). *Neural Networks: Tricks of the Trade* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7700, pp. 421–436 [visited on 2022-10-18]. ISBN 978-3-642-35288-1 978-3-642-35289-8. Available from DOI: `10.1007/978-3-642-35289-8_25`. Series Title: Lecture Notes in Computer Science.
54. SCARFF, Brent. *Understanding Backpropagation* [online]. 2021. [visited on 2022-10-18]. Available from: `https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0`.
55. *A Step by Step Backpropagation Example – Matt Mazur* [online]. [N.d.]. [visited on 2022-10-18]. Available from: `https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/`.
56. ARC. *Derivative of the Sigmoid function* [online]. 2018. [visited on 2022-11-27]. Available from: `https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e`.
57. WANG, Chi-Feng. *The Vanishing Gradient Problem* [online]. 2019. [visited on 2022-10-18]. Available from: `https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484`.
58. *The Vanishing Gradient Problem. The Problem, Its Causes, Its... — by Chi-Feng Wang — Towards Data Science* [online]. [N.d.]. [visited on 2022-10-18]. Available from: `https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484`.
59. *Artificial Neural Network - an overview — ScienceDirect Topics* [online]. [N.d.]. [visited on 2022-10-18]. Available from: `https://www.sciencedirect.com/topics/earth-and-planetary-sciences/artificial-neural-network`.
60. NOLAN, Michael S. *Fundamentals of air traffic control*. 4th ed. Belmont, CA: Thomson–Brooks/Cole, 2004. ISBN 978-0-534-39375-5 978-0-534-39388-5.
61. *Daily Traffic Variation - States* [online]. [N.d.]. [visited on 2022-11-27]. Available from: `https://www.eurocontrol.int/Economics/DailyTrafficVariation-States.html`.
62. *James Clerk Maxwell — Biography & Facts — Britannica* [online]. [N.d.]. [visited on 2022-11-26]. Available from: `https://www.britannica.com/biography/James-Clerk-Maxwell`.
63. JAMES, R.J. A history of radar. *IEE Review* [online]. 1989, vol. 35, no. 9, p. 343 [visited on 2022-11-26]. ISSN 09535683. Available from DOI: `10.1049/ir:19890152`.
64. GUARNIERI, Massimo. The Early History of Radar [Historical. *IEEE Industrial Electronics Magazine* [online]. 2010, vol. 4, no. 3, pp. 36–42 [visited on 2022-11-26]. ISSN 1932-4529. Available from DOI: `10.1109/MIE.2010.937936`.

65. PAGE, R. The Early History of Radar. *Proceedings of the IRE* [online]. 1962, vol. 50, no. 5, pp. 1232–1236 [visited on 2022-11-26]. ISSN 0096-8390. Available from DOI: 10.1109/JRPROC.1962.288078.
66. BLANCHARD, Yves. Une histoire du radar en lien avec les mutations du système technique. 2019, vol. 2019, pp. 35–46.
67. *The Battle of Barking Creek* [online]. [N.d.]. [visited on 2022-11-26]. Available from: https://www.epoch-magazine.com/blackbattleofbarkingcreek.
68. SHAYLER, J. S. The Royal Navy and IFF — Identification Friend or Foe, 1935–45. In: KINGSLEY, F. A. (ed.). *The Development of Radar Equipments for the Royal Navy, 1935–45* [online]. London: Palgrave Macmillan UK, 1995, pp. 277–290 [visited on 2022-11-26]. ISBN 978-1-349-13459-5 978-1-349-13457-1. Available from DOI: 10.1007/978-1-349-13457-1_6.
69. *Surveillance Systems* [online]. [N.d.]. [visited on 2022-11-26]. Available from: https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap4_section_5.html.
70. *PRIMARY SURVEILLANCE RADAR* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://csgaerospace.cz/primary-surveillance-radar.
71. STEVENS, Michael C. *Secondary surveillance radar*. Boston: Artech House, 1988. Artech House radar library. ISBN 978-0-89006-292-0.
72. KOSE, Mehmet Cagri. *Mode A/C, Mode S and ADS-B, The Alphabet Soup of Secondary Surveillance* [online]. 2019. [visited on 2022-11-26]. Available from: https://medium.com/@mehmetcagrikose/mode-a-c-mode-s-and-ads-b-the-alphabet-soup-of-secondary-surveillance-1defcd35b2ab.
73. SUN, Junzi. The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals [online]. 2021 [visited on 2022-11-27]. Available from DOI: 10.34641/MG.11. Publisher: TU Delft OPEN.
74. MUSHTAQ, M. Tahir; BUTT, Faran Awais; MALIK, Ahmed. An overview of spectrum sensing in cognitive RADAR systems. In: *2014 IEEE Microwaves, Radar and Remote Sensing Symposium (MRRS)* [online]. Kiev, Ukraine: IEEE, 2014, pp. 115–118 [visited on 2022-11-27]. ISBN 978-1-4799-6073-6 978-1-4799-6072-9. Available from DOI: 10.1109/MRRS.2014.6956678.
75. *Mode S: An Air Traffic Control Data-Link Technology* [online]. [N.d.]. [visited on 2022-11-26]. Available from: http://web.mit.edu/6.933/www/Fall2000/mode-s/sidelobe.html.
76. WOLFF, Dipl-Ing (FH) Christian. *Radartutorial* [online]. [N.d.]. [visited on 2022-11-26]. Available from: https://www.radartutorial.eu/13.ssr/sr09.en.html. Publisher: Dipl.-Ing. (FH) Christian Wolff.
77. BODART, Jérôme. Mode S Surveillance Principle. [N.d.], p. 38.
78. *European Organisation for the Safety of Air Navigation (EUROCONTROL)* [online]. [visited on 2022-10-23]. Tech. rep. Koninklijke Brill NV. Available from DOI: 10.1163/1570-6664_iyb_SIM_org_39214. Type: dataset.
79. BEASLEY, Barry. Understanding-Mode-S-technology. [N.d.], p. 10.
80. SUN, Junzi; VU, Huy; ELLERBROEK, Joost; HOEKSTRA, Jacco M. pyModeS: Decoding Mode-S Surveillance Data for Open Air Transportation Research. *IEEE Transactions on Intelligent Transportation Systems* [online]. 2020, vol. 21, no. 7, pp. 2777–2786 [visited on 2022-10-23]. ISSN 1524-9050, ISSN 1558-0016. Available from DOI: 10.1109/TITS.2019.2914770.
81. *Multilateration - an overview (pdf) — ScienceDirect Topics* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://www.sciencedirect.com/topics/engineering/multilateration/pdf.
82. MICK, Travis. *A demonstration of TDOA multilateration* [online]. 2022. [visited on 2022-11-27]. Available from: https://lo.calho.st/posts/tdoa-multilateration/.

83. TURNER, Ron. Multilateration Technology Overview. [N.d.], p. 59.
84. BROOKNER, Eli. *Tracking and Kalman filtering made easy*. New York: Wiley, 1998. ISBN 978-0-471-18407-2.
85. VINAYKUMAR, Macharla; JATOTH, Ravi Kumar. Performance evaluation of Alpha-Beta and Kalman filter for object tracking. In: *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies* [online]. Ramanathapuram, India: IEEE, 2014, pp. 1369–1373 [visited on 2022-11-28]. ISBN 978-1-4799-3914-5 978-1-4799-3913-8. Available from DOI: 10.1109/ICACCCT.2014.7019323.
86. BECKER (WWW.KALMANFILTER.NET), Alex. *Online Kalman Filter Tutorial* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://www.kalmanfilter.net/.
87. BECKER (WWW.KALMANFILTER.NET), Alex. *Online Kalman Filter Tutorial* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://www.kalmanfilter.net/.
88. MONTELLA, Corey. The Kalman Filter and Related Algorithms: A Literature Review. 2011.
89. *1.17. Neural network models (supervised)* [online]. [N.d.]. [visited on 2022-11-27]. Available from: https://scikit-learn/stable/modules/neural_networks_supervised.html.
90. K, Dasaradh S. *A Gentle Introduction To Math Behind Neural Networks* [online]. 2020. [visited on 2022-11-28]. Available from: https://towardsdatascience.com/introduction-to-math-behind-neural-networks-e8b60dbbdeba.
91. *Loss Functions — ML Glossary documentation* [online]. [N.d.]. [visited on 2022-11-28]. Available from: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html.
92. PRIMUSA. *Answer to "What is the difference between the terms accuracy and validation accuracy"* [online]. 2018. [visited on 2022-11-29]. Available from: https://stackoverflow.com/a/51345413.
93. SINGH, Aishwarya. *Introduction to Neural Network— Convolutional Neural Network* [online]. 2020. [visited on 2022-11-28]. Available from: https://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network/.
94. CAMACHO, Cezanne. *Convolutional Neural Networks* [online]. [N.d.]. [visited on 2022-11-28]. Available from: https://cezannec.github.io/Convolutional_Neural_Networks/.
95. BROWNLEE, Jason. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks* [online]. 2019. [visited on 2022-11-28]. Available from: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.
96. SOLAI, Pavithra. *Convolutions and Backpropagations* [online]. 2018. [visited on 2022-11-28]. Available from: https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c.
97. *Convolutional Neural Network (CNN) — TensorFlow Core* [online]. [N.d.]. [visited on 2022-11-28]. Available from: https://www.tensorflow.org/tutorials/images/cnn.
98. *Classification: Accuracy — Machine Learning* [online]. [N.d.]. [visited on 2022-11-28]. Available from: https://developers.google.com/machine-learning/crash-course/classification/accuracy.
99. *Classification: Precision and Recall — Machine Learning — Google Developers* [online]. [N.d.]. [visited on 2022-11-28]. Available from: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.
100. *Classification: ROC Curve and AUC — Machine Learning — Google Developers* [online]. [N.d.]. [visited on 2022-11-28]. Available from: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.
101. *Receiver operating characteristic* [online]. 2022. [visited on 2022-11-28]. Available from: https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=1118010681. Page Version ID: 1118010681.

102. KORSTANJE, Joos. *The F1 score* [online]. 2021. [visited on 2022-11-28]. Available from: `https://towardsdatascience.com/the-f1-score-bec2bbc38aa6`.
103. NARKHEDE, Sarang. *Understanding Confusion Matrix* [online]. 2021. [visited on 2022-11-28]. Available from: `https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62`.