Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 1 Capitole (UT1 Capitole)*

Présentée et soutenue le *Date of defense (03/05/2022)* **by :**

Md Siddiqur RAHMAN

Deep Learning for En-Route Aircraft Conflict Resolution - two complementary approaches

---

JURY

M. Pierre GANÇARSKI	Université de Strasbourg	Rapporteur
Mme Maguelonne TEISSEIRE	UMR TETIS - INRAE	Rapporteure
M. Michael SCHULTZ	Universität der Bundeswehr München	Examinateur
M. Nicolas COUELLAN	ENAC	Examinateur
Mme Josiane MOTHE	IRIT	Directrice de thèse
M. Laurent LAPASSET	ENAC	Co-directeur de thèse

École doctorale et spécialité :
MITT : Image, Information, Hypermédia

Unité de Recherche :
SIG team at IRIT UMR5505 CNRS, INSPE et
DEVI team at l'Ecole Nationale de l'Aviation Civile (ENAC)
Université de Toulouse, Toulouse, France

Directeur(s) de Thèse :
Prof. Josiane MOTHE et *Eng. Laurent LAPASSET*

Rapporteurs :
Pierre Gançarski et *Maguelonne Teisseire*

Abstract

A situation is identified as a conflict when two or more aircraft fail to maintain a certain distance between them on their way. Earlier models to support air traffic controllers in solving conflicts were based on mathematical and statistical models. The recent successes of deep neuron network models in various domains have rekindled the research interest on automatic aircraft conflict resolution. Conflicts are solved by controllers at the en-route[1] level by giving orders to pilots to change the aircraft trajectory, based on the various aircraft positions and trajectories. In this thesis we propose two different ways of exploiting these data, considering either the trajectory data or the corresponding images of the trajectories.

The first model, CRMLnet, standing for Conflict Resolution Multi-label Neural Network, is a neural network model which output is a multi-label classification. This model takes the positioning trajectory parameters (time, latitude, longitude, altitude, and heading) of all the aircraft involved in the conflict as input and provides the heading changes for the aircraft at different angles as output. When compared to other machine learning models that use multiple single-label classifiers such as SVM[2], KNC[3], and LR[4], our CRMLnet achieves the best results with an accuracy of 98.72% and ROC of 0.999. This model is not appropriate to handle a variable number of aircraft involved.

On the contrary, our second model ACRnet, which stands for Aircraft Conflict Resolution Convolutional Neural Network, does not depend on the number of planes involved. For that model, we transformed the conflict scene into an image. This model is designed as a convolutional neural network and is also targeting multi-label classification. It achieves an accuracy of 99.16% on the training data and of 98.97% on the test data set for two aircraft. For both two and three aircraft, the accuracy is 99.05% (resp. 98.96%) on the training (resp. test) data set.

Keywords: Air traffic control • Convolutional neural network • Machine learning • Deep learning • Multi-label classification • En-route control.

[1] An aircraft reaches a certain altitude
[2] *Support Vector Machines*
[3] *K-Nearest Neighbor Classifier*
[4] *Logistic Regression*

Résumé

Une situation est considérée comme un conflit lorsque deux ou plusieurs avions ne parviennent pas à maintenir une certaine distance entre eux pendant leur trajet. Les modèles antérieurs destinés à aider les contrôleurs aériens à résoudre les conflits étaient basés sur des modèles mathématiques et statistiques. Les récents succès des modèles de réseaux de neurones profonds dans divers domaines ont relancé l'intérêt de la recherche sur la résolution automatique des conflits aériens. Les conflits sont résolus par les contrôleurs au niveau en-route en donnant des ordres aux pilotes pour modifier la trajectoire de l'avion, en fonction des différentes positions et trajectoires des avions. Dans cette thèse, nous proposons deux façons différentes d'exploiter ces données, en considérant soit les données de trajectoire, soit les images correspondantes des trajectoires.

Le premier modèle, CRMLnet[5], est un modèle de réseau de neurones dont la sortie est une classification multi-label. Ce modèle prend en entrée la trajectoire de positionnement paramètres (temps, latitude, longitude, altitude, et cap) de tous les avions impliqués dans le conflit et fournit en sortie les changements de cap des avions à différents angles. Comparé à d'autres modèles d'apprentissage automatique qui utilisent plusieurs classificateurs à étiquette unique, tels que SVM, KNC et LR, CRMLnet obtient les meilleurs résultats avec une précision de 98,72% et une mesure ROC de 0,999. Ce modèle n'est pas approprié pour traiter un nombre variable d'avions impliqués.

Le deuxième modèle, ACRnet[6], au contraire, ne dépend pas du nombre d'avions concernés. Pour ce modèle, nous avons transformé la scène de conflit aérien en une image. Ce modèle est conçu comme un réseau de neurones convolutionel vise également une classification multi-label. Il atteint une précision de 99,16% sur les données d'apprentissage et de 98,97% sur les données de test pour deux avions. Pour les cas avec deux et trois avions, la précision est de 99,05% (resp. 98,96%) sur les données d'entraînement (resp. de test).

Mots clés: Contrôle du trafic aérien • Réseau de neurones convolutif • Apprentissage automatique • Apprentissage profond • Classification multi-labels • Contrôle en-route.

[5] *Conflict Resolution Multi-label Neural Network*
[6] *Aircraft Conflict Resolution Convolutional Neural Network*

Acknowledgments

There are many people whose inspiration and contribution are absolutely undeniable and their tireless work, empathy to me, everything is the reason for my today.

It is a great pleasure to be able to express my deepest gratitude and appreciation to my thesis supervisor Prof. Josiane Mothe. From start to finish, you support me all the way through my entire Ph.D. journey. I am proud and grateful to you for the unwavering support. I would not have been able to accomplish this important chapter of my life without your unconditional assistance. Your guidance always led me in the right direction whenever I went wrong.

Also, I would like to thank my co-supervisor equally, Laurent Lapasset, for being a big part of my same journey. I am really glad to have a smiling and friendly supervisor like you who has always extended a big supporting hand to me to achieve my Ph.D. It is very commendable that you have managed a nice office room and all kinds of IT and technical support including a high-speed GPU for the implementation part of my research.

I want to give all the beautiful feelings to my mother from the bottom of my heart with all the emotions, love, gratitude to her who is the biggest inspiration of my life. My mother's foresight and prudence have contributed the most to make this beautiful day possible. I would like to keep the deepest thanks to my elder brother Abdur Razzak Ratan who has always supported me mentally and financially for my education. My unconditional love for him will last forever. I am very grateful to my two sisters Farida and Shirin and the other brothers Manik, Hiron, and Atik who always feel proud of me. Their generous love has always encouraged me.

I would like to thank the entire SIG team and team leader Olivier Teste, in parallel, I would also like to thank the IRIT lab for giving me the opportunity to do my Ph.D. there. I am really grateful to Zia, Reshma, Nathalie, and Faneva of the SIG team who have supported me and inspired me when I faced any challenge.

I would also like to thank all my colleagues and labmates of the entire DEVI team and head of research Patrick Senac at ENAC. I really cannot forget the support of some people and they are Hélène, Serge, Hasan, Sana, Diana, Ying, and Samuel. I would like to thank some of the professors, teachers and researchers who have directly or indirectly supported me all the way of my stay at ENAC: Stephane Puechmorel, Daniel Delahaye, Assia Hachichi, Marcel Mongeau, Xavier Pretat, and many others.

It is a genuine pleasure to express my deep gratitude to all of you.

Md Siddiqur Rhaman

Contents

[7] *Air Traffic Control*

[8] *Simulated Trajectory and Immediate Order*
[9] *Converted Trajectory to Image*

List of figures

[10] *Air Traffic Control Officer*

[11] *Multi-label Support Vector Machines*
[12] *Multi-label K-Nearest Neighbor Classifier*
[13] *Multi-label Logistic Regression*
[14] *Convolutional Neural Ntwork*

List of tables

Table of abbreviations and acronyms

Acc	*Accuracy*
ACRnet	*Aircraft Conflict Resolution Convolutional Neural Network*
ADS-B	*Automatic Dependent Surveillance-Broadcast*
Adam	*Adaptive Moment Estimation*
ATC	*Air Traffic Control*
ATCC	*Air Traffic Control Center*
ATCO	*Air Traffic Control Officer*
ATM	*Air Traffic Management*
auPR	*area under Precision Recall*
auROC	*area under Receiver Operating Characteristic*
CNN	*Convolutional Neural Ntwork*
CTI	*Converted Trajectory to Image*
CR-DNN	*Conflict Resolution Deep Neural Network*
CRMLnet	*Conflict Resolution Multi-label Neural Network*
CRMLnet$_{cv}$	*Conflict Resolution Multi-label Neural Network Cross Validation*
CRMLnet$_{ind}$	*Conflict Resolution Multi-label Neural Network Independent Test*
FPR	*False positive rate*
FNR	*False negative rate*
GPU	*Graphics Processing Unit*
ICAO	*International Civil Aviation Organization*
KNC	*K-Nearest Neighbor Classifier*
LR	*Logistic Regression*
LSTM	*Long Short-Term Memory*
MCC	*Mathew's Correlation Coefficient*

MILP	*Mixed-Integer Linear Programming*
MKNC	*Multi-label K-Nearest Neighbor Classifier*
MKNN$_{cv}$	*Multi-label K-Nearest Neighbor Cross Validation*
MKNN$_{ind}$	*Multi-label K-Nearest Neighbor Independent Test*
ML	*Machine Learning*
MLR	*Multi-label Logistic Regression*
MLR$_{cv}$	*Multi-label Logistic Regression Cross Validation*
MLR$_{ind}$	*Multi-label Logistic Regression Independent Test*
MSVM	*Multi-label Support Vector Machines*
MSVM$_{cv}$	*Multi-label Support Vector Machines Cross Validation*
MSVM$_{ind}$	*Multi-label Support Vector Machines Independent Test*
NN	*Neural Network*
PPV	*Positive predictive value*
RNN	*Recurrent Neural Network*
ReLU	*Rectified Linear Unit*
RMSprop	*Root Mean Square Propagation*
SGD	*Stochastic Gradient Descent*
STIO	*Simulated Trajectory and Immediate Order*
S_n	*Sensitivity*
S_p	*Specificity*
SVM	*Support Vector Machines*

Publications

Conference

1. Md Siddiqur Rahman, Laurent Lapasset and Josiane Mothe. (2022). **Multi-label Classification of Aircraft Heading Changes using Neural Network to Resolve Conflicts.** In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 3*, ISBN 978-989-758-547-0, ISSN 2184-433X, pages 403-411

2. Md Siddiqur Rahman, Laurent Lapasset and Josiane Mothe. (2021). **Aircraft Conflict Resolution using Convolutional Neural Network on Trajectory Image.** In *Proceedings of the 21st International Conference on Intelligent Systems Design and Applications*

3. Md Siddiqur Rahman. (2020). **Supervised machine learning model to help controllers solving aircraft conflicts.** In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pages 355–361. Springer, 2020.

Poster

1. Laurent Lapasset, Md Siddiqur Rahman, and Josiane Mothe (2020). **Solving aircraft conflicts: data resources.** In *1st International Conference on Cognitive Aircraft Systems (ICCAS 2020)*. p. 76 (2020).

Chapter 1

Introduction

Abstract.

When two or more aircraft come close to each other and are unable to maintain a certain distance (internationally specified) on their flight without modifying their route, it is called a conflict since there is a possible collision. Conflicts between aircraft are frequent and the corresponding air traffic control officer (ATCO) is responsible for guiding pilots to resolve them. Methods are used to assist ATCO, including some based on machine learning. Many of these methods perform rather well but have some limitations. For example, they provide a single solution to a conflict and do not provide an alternative one if the best one failed. In this thesis, we proposed two supervised multi-label machine learning classification models to solve aircraft conflicts. The first one, CRMLnet classifies the aircraft's possible heading decisions in a specific order and is based on trajectory data. Although, CRMLnet model provides multiple alternate resolutions for a single conflict, still, its major limitation is its dependence on the number of aircraft. We thus proposed a second model, ACRnet that uses images of the conflict scenes, which are converted from trajectory. This model is independent of the number of aircraft.

Contents

Two or more aircraft are considered in a conflict situation if they fall in a distance less than the *5 nautical miles horizontally or 1000 feet vertically* internationally defined distance when crossing each other [Kuchar 2000, Prandini 2000].

Figure 1.1 shows the region around an aircraft in which any other aircraft would be considered in conflict. Not only do conflicts can occur with other planes, but also with bad weather, military zones, etc. This area around an aircraft is called the *conflict volume* of that aircraft. The presence of any other aircraft in this volume is considered as a conflict.

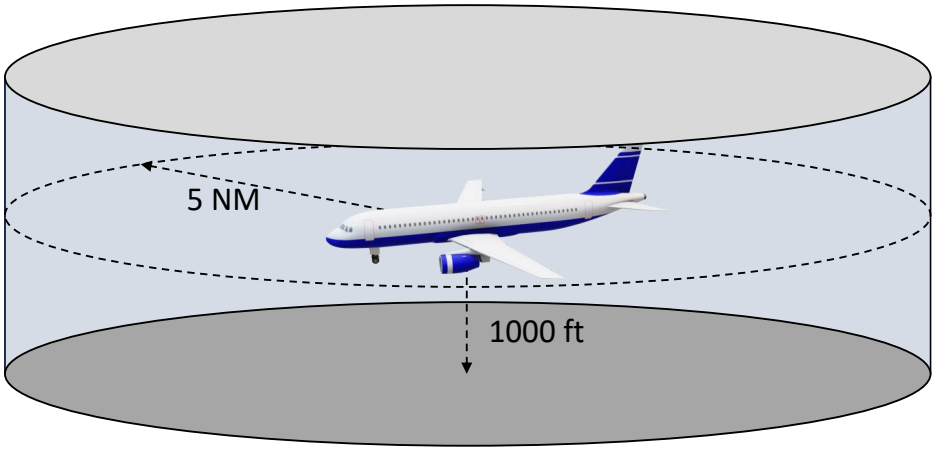


Figure 1.1: An aircraft is considered to be in conflict if it falls horizontally within 5 nm and/or vertically within 1000 feet. Two or more aircraft are in conflict if they overlap this region, called the conflict volume of that aircraft.

Conflicts are most common when aircraft are flying and these are the most frequent problems in the domain of air traffic management. Air traffic control officers (ATCOs) observe and resolve such conflicts. The world airspace is divided into sectors or small areas and specific ATCOs are assigned to each sector. Once a conflict occurs, the ATCO in that particular sector resolves the conflict as soon as possible. To avoid conflict situations, some rules and regulations have been developed [Mao 2001]. Conflict situations however are rapidly rising due to the increase of the number of aircraft. For example, Menéndez *et al.* [Prats Menéndez 2018] prepared the APACHE final project results report where they recorded 24 hours of summer and winter aircraft conflict in 2017-2018 for the FABEC (Functional Airspace Block Europe Central) airspace. A total of 938 conflicts were found for moderate traffic demand in summer and 602 for low traffic demand in winter. Thus, the most challenging task for ATCOs is to solve conflicts in real-time. Indeed, once a conflict is identified, within a short time, the ATCO must consider the environment of the conflict to make a quick decision to solve it. A conflict is resolved by guiding the pilot to change their initial route. ATCO considers different parameters such as the position of the aircraft (latitude, longitude, altitude), speed, destination, flight plan as well as other elements of the environment, for instance, weather, wind direction, military zones, etc.

1.1 Types of controller

There are different types of control systems available to control an aircraft based on its position. Different controllers operate at different stages of the control system.

(a) Apron controller: The apron controller is responsible for controlling the aircraft from engine start to taxiway. A pilot starts the aircraft engine with the permission of the apron controller and pushes the aircraft backward from the parking lot. The apron controller

sends a taxi to take the plane from the parking lot to the taxiway. The apron controller controls all the aircraft one by one through specific sequence maintenance.

(b) Ground controller: Once an aircraft enters the taxiway, the pilots concerned are under the control of the ground controller. The ground controller guides the aircraft from the taxiway to the next runway. From the different taxiways, the planes come to the runway with complete instructions from the ground controller. The main responsibility of the ground controller is to send the aircraft to the runway while maintaining the specific sequence in different aircraft.

(c) Local controller: Whenever an aircraft reaches the runway, the pilots call the local controller. The local controller at the control tower is then responsible for controlling the aircraft. This position of the aircraft is considered to be ready for take-off. The local controller controls the area up to 5 miles around the airport. Within this area, the pilot receives all kinds of information from the local controller including the weather updates. The local controller gives the take-off clearance. The local controller also controls the departure and landing of each aircraft considering the runway condition.

(d) Departure controller: The pilot starts the communication with the departure controller after the aircraft takes off and reaches a certain altitude. Each plane arrives at its specified altitude according to the departure controller's instructions. The departure controller considers the position of the other aircraft and guides the pilots until the aircraft reaches a safe altitude. The departure controller avoids collisions with other aircraft.

(e) Area controller: When an aircraft reaches a certain altitude (airway) under the direction of the departure controller, the pilot begins to communicate with the area controller. The airway area is divided into several sectors or small areas. There are many waypoints[1] to define the airway. A flight plan in the airway is one of the complete series of points of a flight where latitude, longitude, and altitude are the basic elements used to design it. Waypoints indicate the intermediate and intersection points used to design the flight plan. A minimum distance between the planes should be kept. Figure 1.1 shows the conflict zone around an aircraft which is 5 nm horizontally and 1000 ft vertically. An aircraft passes through several sectors on its way. When an aircraft enters a sector, the area controller of that sector begins to guide that aircraft. Our thesis focuses on the resolution of conflicts in the airway area, to help area controllers in their decisions.

(f) Approach controller: When an aircraft reaches its destination, it tunes to its own communication radio frequency to the approach controller. Then the approach controller provides the facility for landing, such as allocating the runway. One of the important responsibilities of the approach controller is to create a sequence between the incoming aircraft based on their priority and guide them for approaching. Finally, the approach controller transfers control to the local controller. Then considering the runway situation, the local controller gives permission for landing.

Table 1.1 shows the work sequence of an air traffic control in the control tower. At

[1]The waypoint is a fixed coordinate for specifying a single point on the Earth.

departure airports, for example, the controller sequence starts from the apron control. Then ground control, local control, departure control, and area control. On the other hand, the controlling sequence is opposite at the destination airport. The area control transfers the control to the approach control. Then local control, ground control, finally the aircraft is parked by the apron control. With the exception of departure and destination airports, an aircraft operates its flights through many sectors. Various area controllers are in charge of all sector controls. This attitude of the aircraft is called airway or en-route. Figure 3.3 describes the different phases of an aircraft. This thesis focuses on the resolution of aircraft conflict at the en-route level.

Table 1.1: Different types of controllers control an aircraft throughout its journey. Although the control procedures are the same at departure airports and destination airports, the order at destination airports is reversed. In between these two airports, the aircraft passes through different sectors at a certain altitude; the designated area controllers of those sectors guide the pilots.

Departure airport	Airway (en-route)	Destination airport
Apron control		Approach control
Ground control	Area control	Local control
Local control		Ground control
Departure control		Apron control

1.2 Types of conflict

According to Alonso-Ayuso *et al.*, there are three types of conflict based on the time distance between involved aircraft and the conflict point as follows [Alonso-Ayuso 2016b]:

(a) Short-range: the controller has 2-5 minutes to resolve the conflict; this is an emergency;

(b) Mid-range: the time distance is 5 to 20 minutes.

(c) Long-range: the time distance is 20 to 60 minutes.

In our research, we focused on area controllers also called en-route level (high altitude) and to mid-range conflicts. The resolution decisions are different for the different control levels and for the different ranges of conflict.

ATCOs make decisions by considering all the positional issues of all the aircraft involved to resolve the conflict. A conflict can be solved in many ways, but, mid-range conflicts at en-route are usually resolved by changing the heading direction of one or more aircraft. Depending on the conflict situation, the heading change can be left or right direction in any range.

ATC trajectory data is generally kept confidential and not publicly available. Thus, we simulate data for the area control level and for the mid-range conflicts.

1.3 Conflict resolution

Once a controller is alerted, they find out if a conflict is about to occur. The controller considers the position and movement of the planes involved in the conflict and guides the corresponding pilots to resolve the conflict. A conflict can also happen with bad weather, thunderstorms, military zones, etc. Since the position of the aircraft obtained by ATC radar is approximate, no model can guarantee 100% the conflict is resolved. Also, finding the heading resolution is highly combinational when the number of aircraft is high. According to Peyronne [Peyronne 2012], the conflict situation is sometimes considered as a non-deterministic polynomial-time (NP) hard problem when the number of aircraft is high. Thus, there are so many life risks involved that human (ATCO) interaction is not yet replaced by machines. However, assisting ATCOs in their work by suggesting the best possible conflict resolutions could be helpful. This is the challenge we tackle in our thesis.

In terms of modeling, we considered different categories of conflicts as follows:

(a) A conflict between two aircraft;

(b) A conflict that involves more than two aircraft but a fixed number of them;

(c) A conflict that involves more than two aircraft but an undefined number of them.

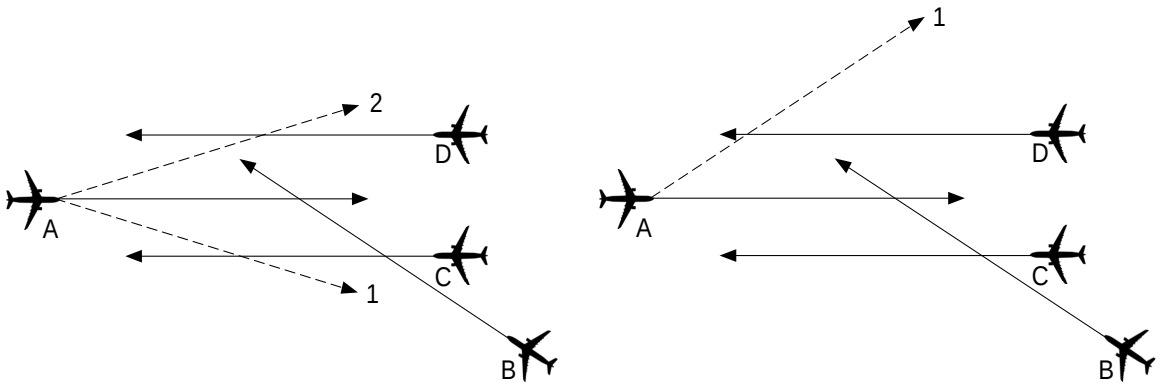
If a conflict occurs between two planes only, it can be resolved by considering all the possible resolutions by modifying the angle of the trajectory and choosing the best one. If a conflict occurs between more than two aircraft, then two cases can be distinguished:

(a) The conflict between the two planes will continue to be resolved until all conflicts are resolved (Figure 1.2 (a)). This is because a resolution can create a novel conflict;

(b) Another way is to find a global resolution considering the location of all the aircraft involved in the conflict (Figure 1.2 (b)).

Many models have been proposed on both approaches which we will discuss in detail in Chapter 2.

Figure 1.2 shows two different solutions considering the same scenario. Figure 1.2 (a) shows that there is a conflict between aircraft A and aircraft B. To solve this in a pairwise manner, we can first look for a solution between A and B. In this case, the right heading is searched for (1) or the left heading (2) 15 degrees. The conflict between aircraft A and B is resolved but later a new conflict is created with aircraft C or D. Although conflict can be resolved using this method, it takes a long time to find an overall solution after repeated attempts. In some cases, it is not possible to find a solution due to the high combination of the resolution between aircraft.



(a) Pairwise conflict resolution (b) Globally conflict resolution

Figure 1.2: Conflict between multiple aircraft can be resolved in two ways: (a) in pairs as long as there is no conflict; and (b) at once considering the overall scenario.

Conflict resolution is a little more complex considering the overall situation, especially applying conditional and rule-based mathematical models. If we look at Figure 1.2 (b), considering the overall situation, only the turn left 30 degrees heading of aircraft A can make the whole situation conflict-free. However, how long it takes to find the best resolution for a model depends on the type of model used. An additional challenge is when the number of aircraft involved is variable. Resolving the conflict between a variable number of aircraft is a challenging task. The problem is that most of the machine learning models have fixed input dimensions that cannot be changed in real-time. This means that a conflict resolution model has to consider pairwise resolution problems, input dimension problems, overall scenario, etc.

There are different models for aircraft conflict detection and resolution: Kuchar and Yang [Kuchar 2000] came up with a survey analysis of about 68 models. Their study provides a good overview of the early work on conflict identification and resolution. Most of the models initially proposed for conflict identification and resolution were based on mathematics, geometry, and probability [Havel 1989, Sridhar 1997, Ota 1998, Prandini 1999, Eby 1999].

1.4 Machine learning models to solve conflicts

Researchers are moving towards machine learning applications in almost all domains. Machine learning can handle any type of data. Machine learning models gain experience through training from examples and are then able to handle unknown samples. When the number of examples is small there is an advantage to increase the model training ability by artificially increasing the number of data through data augmentation.

Machine learning applications [Alam 2009, Jiang 2018, Srinivasamurthy 2018, Pham 2019a] including deep learning [Nanduri 2016, Brittain 2018, Brittain 2018, Wang 2019] have recently grown in the field of aircraft conflict avoidance. Over time, the application of machine learning has become popular in the different fields of ATMs including anomaly detection

[Das 2010, Nanduri 2016, Das 2011, Matthews 2013]. Details will be discussed later in Chapter 2.

The trajectory information such as the position and velocity of aircraft are approximate since they are updated through an ADS-B[2]. Our objective is to define a machine learning model that would learn the overall situation from examples and thus would be close to human performance.

In the contribution Chapter 4, we design a first model to solve conflicts when two aircraft are involved. It is based on a neural network trained with pairs of trajectories.

The position of each aircraft is changing at every moment and is defined by its latitude, longitude, altitude (See Figure 1.3). In this research, we do not consider the current position of the aircraft only but also 5 minutes of trajectory information that we call a 5-minute window. With regard to conflict resolution, many works use the current position of the aircraft from which the future position projection and the distance between the aircraft are calculated using speed, the angle between them, time, and many more parameters, possibly from different sources including on-board data [Prandini 1999, Prandini 2000, Pham 2019a, Kim 2016, Pham 2019b]. Since the positioning coordinates of each aircraft are approximated, we hypothesized that the 5-minute window information is useful for the model to learn the nature of the motion of the aircraft.

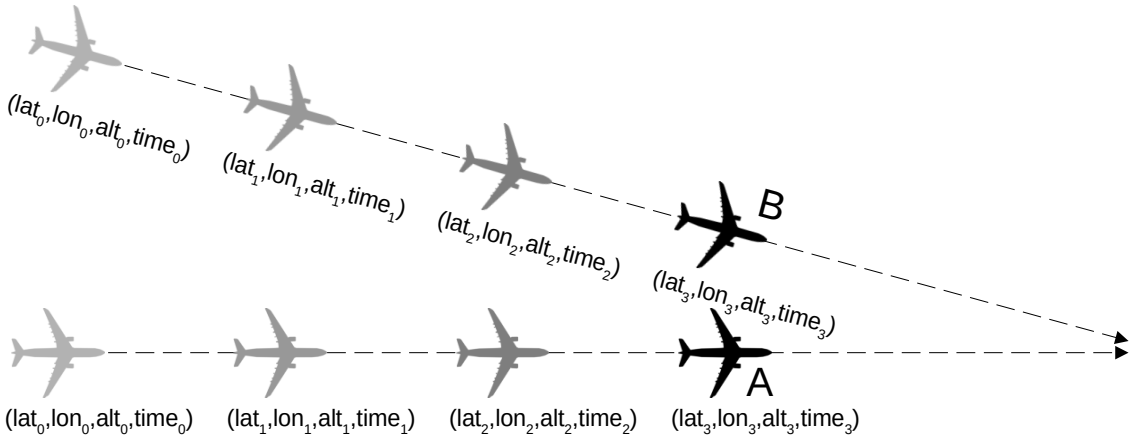


Figure 1.3: The position parameters of the plane change over time. For example, latitude, longitude, altitude, etc. change with time. So does the distance between aircraft.

Therefore, our model can learn the conflict environment from the movement of aircraft.

Another difference compared to related work is that we consider multiple output decisions for a given conflict. This is to give a controller multiple alternative solutions in hand where future risk avoidance will be much easier to get.

One of the main challenges of our research is data preparation. The ATC trajectory and controller immediate order data is not publicly available and there is no simulated data

[2] *Automatic Dependent Surveillance-Broadcast*

available. We thus created simulated data for this research that we made available to the research community. Chapter 3 provides the details related to data preparation.

In the case of conflicts that involves two aircraft, we first cast the conflict resolution problem into a multi-level classification one. We then developed a model that we called CRMLnet, which stands for Conflict Resolution Multi-label Neural Network. It is presented in Chapter 4.

One of the most important challenges of conflict resolution however is when there are more than two aircraft and when the number of aircraft can vary, which is the real world. The training and testing performance computation of state-of-the-art models increases with the number of involved aircraft. Moreover, most of the models, like our CRMLnet from Chapter 4 can handle only a given number of planes that are encoded by the model input of a fixed size. To face this challenge we designed an alternative model based on a convolutional neural network in Chapter 5 that we called ACRnet for Aircraft Conflict Resolution Convolutional Neural Network. We convert the trajectory data into image data so that the input dimension problem is overcome. This model will not only solve the conflict between the aircraft but could also be applied to the resolution of bad weather area conflict for example. The solution is predicted considering the overall scenario. We found that this method not only overcomes the problem of input dimensionality but also improves the prediction confidence of the model. The main purpose of Chapter 5 is not however to find the best image processing model but to easily overcome the challenge of existing sequence-based models through image processing with higher performance.

Since no model can guarantee a 100% resolution to the conflict and human life is involved, the purpose of this thesis is not to completely replace the controllers but to provide supportive tools to help them. Thus, the idea is to propose a model that will be closer to what a controller does, and therefore the decision will be more easily accepted as a support tool for collaborative decision making. Although many models have been proposed in the past, including neural network models, the limitation of almost all of them is that the models were trained based on the extraction of certain features using mathematical rules and conditions. In these models, only one resolution was provided for a fixed number of aircraft. The fact that this decision could fail is not considered and no alternative solution was suggested. In our thesis, we gradually demonstrate our idea of applying models based on neural networks in Chapter 4 and in Chapter 5. It resolves conflicts with good performance using raw features. We show it is possible to propose a generic convolutional neural network model for an arbitrary number of planes by converting images from trajectories.

The rest of this thesis is organized as follows: Chapter 2 covers important and recent related work. All data preparation used in this thesis is discussed in Chapter 3. Chapter 4 describes our first contribution to resolve the conflict between aircraft and details the CRMLnet model based on neural network where we used trajectory data. The model is evaluated on our simulated data. The second contribution is based on converting trajectory data into images. It leads to the ACRnet model based on a convolutional neural network and is presented in Chapter 5. Finally, Chapter 6 concludes this thesis and provides future directions.

Chapter 2

Literature review

Abstract.

Over the past few decades, many approaches were proposed to resolve aircraft conflicts. Most of the models initially proposed were based on mathematics. Advanced computer technology, such as computers with graphics processing unit (GPU) systems, simplifies complex and time-consuming computing such as machine learning. Machine learning models are now developed in the field of aircraft conflict resolution. In this chapter, we discuss some of the most important models and recent models that play an important role in aircraft conflict detection and resolution.

Contents

2.1 Introduction

The main purpose of conflict resolution models is to find the decisions where a safe distance is maintained between the aircraft. Different models for aircraft conflict detection and resolution have been published over the past few decades. Kuchar and Yang discussed most of the early studies in their paper [Kuchar 2000]. Most of the early models relied on mathematics, geometry, and probability [Havel 1989, Sridhar 1997, Ota 1998, Prandini 1999, Eby 1999]. Researchers focus now on the recent advances in machine learning. Although machine learning was invented a long time ago, it was not possible to build a high computational model of machine learning without advanced computers. Since this gap has recently been filled with GPU[1] facilities, the application of machine learning [Alam 2009, Jiang 2018, Srinivasamurthy 2018, Pham 2019a] including deep learning [Nanduri 2016, Brittain 2018, Brittain 2018, Wang 2019] have recently grown in the field of aircraft conflict avoidance.

[1] *Graphics Processing Unit*

This chapter introduces different methods from the last few decades. Conflict resolution methods are closely related to conflict detection methods. We discuss both in this chapter.

This chapter is structured as follows: Section 2.2 discusses some of the early research on aircraft conflict detection and resolution before machine learning was used. Section 2.3 discusses the different types of machine learning models for aircraft conflict detection and resolution; some of them include aircraft trajectory anomaly detection. Section 2.4 discusses the gaps in current research. We also explain the genesis of the models we propose.

2.2 Models before machine learning applications

Most of the earlier aircraft conflict detection and resolution models as well as some recent models are based on mathematics, geometry, and probability [Havel 1989, Sridhar 1997, Ota 1998, Prandini 1999, Eby 1999, Bilimoria 2000, Richards 2002, Alonso-Ayuso 2010, Alonso-Ayuso 2012, Alonso-Ayuso 2013, Feron 2013, Alonso-Ayuso 2016a].

Havel and Husarčík in 1989 discussed many important formulas to solve conflicts involving two aircraft. Their formulas have been used later in many studies [Havel 1989]. Their research was based entirely on theoretical formulas and the authors did not implement any model. According to their solution, the controller could only see the prediction of whether there was a conflict between two planes. For conflict resolution, controllers still had to make their own decision. The non solved issue in their approach is that if the number of aircraft exceeded two, the conflicts had to be resolved in pairs. A conflict resolution could then create a new conflict. The conflict in pairs had to be resolved till all the conflicts were resolved. This was time-consuming and in some cases, conflicts could not be resolved because of the many combinations of pairs. Systems receive multiple positioning information for each aircraft from different radar sources and calculates the estimated position which is thus approximate. Therefore, using Havel and Husarčík 's formulas to build an automatic model is risky because in their approach, the positioning coordinate values are assumed to be accurate.

Sridhar and Chatterji presented a comparison between three methods for conflict detection: (a) Euclidean distance-method, (b) Sorting-based algorithm, and (c) Accumulator-based algorithm [Sridhar 1997]. In the first method, the authors calculated the euclidean distance between each aircraft pair and compared it with the minimum separation distance to see if there was any conflict. The problem is that the computation had to be conducted $\frac{n(n-1)}{2}$ times for n aircraft. When the method is repeated for every time stamp, the computation is increasing with the number of aircraft. The authors presented two more methods to reduce the computation where they expressed the trajectory of each aircraft through a small bin within a 2D grid. Bins in the 2D grid were numbered sequentially. The trajectory of multiple aircraft was then plotted on this 2D grid through these bins. The bin numbers of all aircraft trajectories were then sorted in non-decreasing order in a single vector. If there is a conflict then the same number will be repeated in that vector after sorting. This means that where the bin number is repeated there is a conflict because the same bin belongs to multiple trajectories. In their third method, the whole grid was filled with initial zero instead

of sequential numbering. Then added one to where the aircraft trajectory is supposed to be, where the trajectory was first mapped. For example, when the trajectory of an aircraft is plotted, one is added to the initial zero value. If there is a conflict of that trajectory with another trajectory, then the value of the specific bin of the conflict is incremented again to become two. The value of that conflict bin is a minimum of two. After plotting the trajectories of the two aircraft, the value of the bin is two where the conflict has taken place and all the other trajectory values are one. Undoubtedly this is a really nice piece of work in terms of reducing time and computation to find the aircraft conflict, but there are some limitations. The authors' method only detected conflicts in between two planes when their routes cross each other based on a planned trajectories. Since time is not mentioned on their grid, it is possible that two aircraft trajectories crossed the same bin but not at the same time. In that case, there will be no conflict even though their sorting algorithm will show conflict. Whether or not two planes crossed at the same time is a very important issue that the authors missed. In addition, the authors assumed that all aircraft are following their initial planned route which it is not always the case. Although this work is not similar to ours, their idea of a 2D grid has inspired us to use the image conversion model in Chapter 5.

Ota *et al.* [Ota 1998] proposed a geometric model where the main idea of resolving the conflict was to find the geometric relationship of the aircraft and the expected threat. In this case, the relationship between the subject aircraft and each threat is calculated; they call this a "threat map". The solution considered each moving threat as a static threat. The authors solved conflicts using a set of rules. An example of such a rule is as follows: if two planes are detected as in face-to-face conflict and their speed is equal then both planes will turn to their right in the horizontal planar. However, the angle of the turn is not specified. Although it is not too complex to add, still, the issue of this model is the huge computing to do even if there is no conflict. This research has proposed equations for the threat map concept in which the movement of each aircraft is static, which does not correspond to reality. Also, vertical descent or climb is applicable in their resolution rules which is usually avoided in case of horizontal conflict in real cases.

Zeghal and Karim [Zeghal 1998] proposed a new formula to improve the force field method. The solution they called the force field method is based on the continuous change of velocity and position of each aircraft using relatively simple formulas of physics. In this approach, each plane is compared to a charged object so that each plane is continuously searching for its individual resolutions. Although this method seems very interesting, it searches for the resolution maneuver even though there is no conflict. Moreover, the position and velocity of aircraft are updated through an automatic dependent surveillance-broadcast (ADS-B) which is an approximate position. Thus it is risky to use mathematical formulas to make a sharp heading directional decision based on an approximate position. Besides, with the increasing number of aircraft, it will be challenging for this repeated approach to deal with real situations. Finally, this solution needs to be further considered before being applied in real-life because continuously searching for a resolution maneuver makes a simple heading change to a complex one which is unrealistic.

Prandini *et al.* [Prandini 1999] proposed a probabilistic framework for predicting a mid-

range conflict between two planes. The authors assumed that the two planes are moving horizontally at the same altitude level following their initial *flight plan*. Here, a flight plan is one of the complete series of waypoints that indicate the intermediate and intersection points that pilots use to design the initial flight plans before departure. The speed between each of the two waypoints is also considered. Using these pieces of information, the authors used some equations to project near future positions of both aircraft and predict whether there will be a conflict. They used a threshold value to determine what is the minimum probability score to be considered as a conflict. Since the position of the aircraft is approximate, this study predicts the future position of the aircraft using probability distribution rather than using precise geometric and mathematical equations. However, an algorithm is continuously estimating the probability distribution of the aircraft near future position that demands heavy computations. Since the probability distribution detects conflicts, it has to be calculated for each resolution direction. With more than two aircraft, that computation can be too huge for the real-time calculation.

Other related work consider free flights where a pilot can change his/her flight route in the mid-flight if s/he wants to. In reality, this is not yet possible. Warren [Warren 1997] applied performance analysis in three different situations to detect the conflicts for free routing: fixed threshold conflict detection, covariance method conflict detection, and conformance bound conflict detection. For the first method, the authors chose whether a situation is called conflict or not based on the closest point of approach (CPA). CPA was taken as 10 nm. In the second approach, they formulate an error ellipse based on covariance that is achieved by the error of surveillance, wind forecasting, and aircraft path following. The third approach is a commonly used conflict detection method for Advanced En Route ATC (AERA) [Brudnicki 1997]. The flight plan of an aircraft is monitored at every moment. If the actual flight plan is more inconsistent with the predicted flight plan, then the situation is considered as a conflict. This method is applied to every aircraft. All the three approaches are similar in performance but in some cases, fixed threshold based on out of CPA or inside CPA is doing better than the others.

Pallottino *et al.* [Feron 2013] proposed a method of conflict resolution where they applied mixed-integer linear programming (MILP[2]). Here, mixed-integer means that all the variables used to take conflict resolution decisions are integers and continuous mixed values. Richards and How in [Richards 2002] applied MILP for the first time to find the optimal trajectory waypoint so that conflict can be avoided. The variable in the decision to choose a waypoint is binary and since different coordinate variables are involved in the movement of an airplane and it is continuous, they called it a mixed-integer linear programming problem. Their models had different geometric constructions that assume that all pilots have the right to free flight. They resolved the conflict by changing the horizontal heading of the aircraft. No vertical resolutions are provided such as decent or climbing as they assumed that all aircraft are at the same altitude. Based on the euclidean distance between different planes, their method finds conflict-free trajectories by increasing or decreasing the left heading or right heading angle for each plane. Since they considered the concept of free-flight, a geometric equation is applied to each plane to find its own conflict-free heading degree angle. Although they

[2] *Mixed-Integer Linear Programming*

showed optimization so that increasing the number of aircraft requires less computing, the authors stated that their method is unsuitable for operational implementation. The concept of free-flight is very interesting because if for some reason the communication of the aircraft is cut off from the ground system, then the pilot themselves will be able to resolve the conflict. But since the concept of free-flight is still limited in theory, its application in practice is still a long way off because of the lack of advanced on-board systems that are not yet available.

Another application of MILP was proposed by Vela *et al.* in [Vela 2010] where they took both heading and speed changes decision to resolve the conflict for free flight. The authors claimed that their model provides a global solution by changing the heading and/or speed of one or more aircraft. But in reality, it is not so easy because every aircraft changes its speed or heading depending on its involvement in the future of any new conflict. In addition, it is not clear how their model will receive input when the number of aircraft is not fixed.

According to Alonso-Ayuso and Escudero [Alonso-Ayuso 2016a], three types of maneuvering are commonly applied for aircraft conflict resolution such as heading change, velocity change, and altitude change. The same author proposed different models that combined these three types of maneuvering to resolve the conflict between aircraft. For example, in [Alonso-Ayuso 2010] they applied the technique of changing velocity and changing altitude. They proposed another model in [Alonso-Ayuso 2012] that considered only the change in velocity. In that case, they took into account the acceleration change of the aircraft. The following year, they proposed two different models [Alonso-Ayuso 2013]: one with a change of altitude and the other with a change of velocity. All the models discussed so far by these authors used the mixed integer linear optimization (MILO) concept. Since the decision variables are mixed integers and continuous, the method is called MILO. Finally, the authors in [Alonso-Ayuso 2016a] proposed a method based on mixed-integer nonlinear optimization (MINO) that combined three maneuverings (heading change, velocity change, and altitude change). This is non-linear because they used continuous values for decision-making.

Carbone *et al.* [Carbone 2006] proposed a geometric algorithm for conflict resolution between two aircraft. In this case, one of the two aircraft is called the intruder; it usually does not play any role in conflict resolution. The other aircraft with which the intruder's conflict has been identified are used to resolve the conflict. The authors formed a sphere around the intruder using geometrical equations, and the future position of any other plane in that sphere detection was considered as a conflict. They applied three strategies for conflict resolution: lateral-directional control; longitudinal control; and speed control. Lateral-directional control means changing the direction of the heading whereas longitudinal control means climbing or descending vertically. And the other is to resolve the conflict through speed control. When one of these three strategies is applied, the others are considered constant. This means that multiple strategies are not applied simultaneously. All the calculations are done in real-time and the authors claimed the method is very fast. The limitation of the method however is that it is applied to two aircraft only. In a real-time situations where there are more than twp aircraft involved, the application of this method will be more challenging. Another issue is that it is not clear which of the three strategies is to be applied in which situation.

Durand *et al.* in [Durand 2020] proposed a visualization tool that helps the ATCOs to see

the conflict zones. To handle conflicts, ATCOs can use a 2D screen to plot future positions simply using a mouse pointer. Then, they can see different color line segments for conflict areas and the areas without conflict including the speed and headings. Finally, they can decide the heading to resolve the conflict. This research has provided a step forward for the controllers since they can see different conflict areas including uncertainty (bad weather). However, the limitation here is that the tool does not provide any specific decision like heading changes for the conflict resolution. ATCOs have to observe the scenario and take decisions by them-selves only. It is time-consuming in real-time to observe the conflicts by plotting different ways.

Zhao and Liu in [Zhao 2020] proposed another popular graph theory algorithm called A* search. This algorithm is based on the cost of the current node ($g(n)$) from the starting node and based on a heuristic function ($h(n)$) to find the shortest path that estimates the minimum cost from n to the goal node. The author generated a cost map based on the distance of Ownship (Figure 3.5) with all Intruders (Figure 3.5). Then they applied the A* search algorithm to get through this cost map at a low cost. Although graph theory plays a unique role in figuring out the shortest path and A* search is a popular algorithm in this case, if it is a static graph. One more thing to consider here is that when the number of planes is high, the use of graph theory is a matter of many combinations so that time is an important factor.

2.3 Machine learning models

Aircraft conflict resolution is of different levels (Section 1.1) and types (Section 1.2). In this section we will only discuss machine learning methods related to the mid-range conflicts at the en-route level. Although our thesis focuses on aircraft conflict resolution, we will also discuss machine learning applications for conflict detection as these two topics are related to each other.

With the availability of resources like data and high graphics computers, machine learning applications are used almost everywhere. A major advantage of the machine learning models is that such models acquire knowledge from real data which is much closer to human intelligence but faster than humans in real-time. Therefore, in the case of aircraft conflict resolution, it has been found that machine learning applications are able to make decisions very quickly, but since no model can offer a hundred percent guarantee, this type of models are preferred to assist rather than to replace humans to make quick decisions.

Over the past two decades, many machine learning methods were proposed to identify and/or resolve aircraft conflicts [Alam 2007, Alam 2009, Nanduri 2016, Jiang 2018, Srinivasamurthy 2018, Pham 2019a, Brittain 2018, Wang 2019].

One of the earliest machine learning applications, more specifically the application of data mining, was proposed by Alam *et al.* to detect aircraft conflicts for free flight, although they separated the concept into two different pieces of work [Alam 2007, Alam 2009]. In

their first study, [Alam 2007], they applied data mining technique through investigation and tried to find the patterns that would explain conflict detection algorithm failures. Later, in [Alam 2009], the same authors used this knowledge to construct a relationship between the failure of various conflict detection algorithms and these algorithms. They completed their research in three steps. In the first step, they applied various conflict detection algorithms to the simulated conflict scenarios. The second step was to find out the pattern of data with each algorithm and to create different rules. Finally, they applied an ensemble method based on all these rules to determine when an algorithm will be chosen for the final conflict detection. This means that the ensemble method allowed one conflict detection algorithm from different algorithms at a time to make a final decision based on the conflict situation and the rules already defined for each algorithm. Although this idea is limited to conflict detection only, it can be extended for conflict resolution. But the most important limitation here is that the authors have limited this algorithm to conflict detection for free-flight only. Therefore, the expansion of this idea will be applicable when the free-flight facility will be implemented in practice.

Detecting conflicts and resolving them are very close problems. This is because if a conflict detection model can be used to predict future conflicts, then by avoiding those routes, the conflict will be resolved. Jiang *et al.* [Jiang 2018] considered aircraft conflict detection as a binary classification problem. The authors believed that since the various parameters of each aircraft, such as tracking, navigation, weather, etc., are approximate parameters, it is better to consider the conflict detection problem as a probabilistic problem rather than a rule-based problem. In their paper, they considered the conflict detection problem as a binary classification problem and proposed a conflict detection model based on a support vector machine (SVM) classifier. As the input for this model, they extracted a new feature vector from the current positioning coordinate and velocity of the aircraft associated with the conflict, which they called the relative feature vector. They prepared a relative vector by calculating the distance between aircraft along the different axis (x, y, z), speed difference, etc. which was later used as input of the model. Finally, based on this input, their SVM model predicted future conflicts. Although this work has a lot of potential to extend this concept to resolve conflicts, the limitation of this idea is that the authors proposed the solution for free routing that is not yet available in real life.

Since it is difficult to get data related to aircraft conflict detection and resolution, very few supervised machine learning methods have been applied so far [Srinivasamurthy 2018, Kim 2016]. Srinivasamurthy *et al.* in [Srinivasamurthy 2018] proposed a semi-supervised model in their research that predicts the air traffic controller's voice command. Their model is a combination of the Deep Neural Network and the Hidden Markov model that updates itself iteratively with the speech and radar data. Although this research does not directly address the problem of aircraft conflict, if it is possible to fully predict ATCO's voice command, then it is possible to create a real-time solution by applying this same concept to resolve the aircraft conflict. This model can also help to store the radar data with voice command annotation and it will play a big role in preparing conflict resolution future models. Their research is still far from the expected real-time results because ATCO's voice commands have a lot of extra sounds (noise), there is still some work to be done to make this research applicable in

practice. That is why there is a lot of research which is carried on in the ATM field on speech recognition, and maybe we can use this concept in future models.

Kim *et al.* [Kim 2016] presented a performance analysis of two separated models to solve a conflict between two airplanes: a neural network-based and a SVM-based. Both models are fully supervised models. The SVM model combines 9 SVM, one per class label. Similarly, the neural network model is composed of 9 nodes in the output layers. This model gives an output vector of 9 class labels that are all zero except the most probable one. In the end, they have taken the highest probable one as the best action. Their dataset contains category-based resolutions such as vertical, horizontal, and speed control. For example, there are two resolutions for horizontal conflicts such as *Direct-to* and *Path stretch*. Here *Direct-to* means the resolution maneuver is to skip some initial waypoints (see Section 3.2.1) and go direct to the targeted waypoint whereas *Path stretch* is to add new waypoints to make the resolution more flexible. The model only predicts these categories. The limitation here is that there is no exact heading direction (horizontally) or climbing/descending level (vertically) to resolve the conflict. Still, ATCO needs to think about the resolutions before taking a decision.

Some reinforcement learning models were designed to resolve aircraft conflicts where deep neural network was used as an agent [Brittain 2018, Pham 2019a, Pham 2019b, Wang 2019]

Brittain and Wei in [Brittain 2018] applied a two-level agent-based (parent and child agent) deep reinforcement learning following a hierarchical network manner. For the parent agent, they applied a convolutional neural network to the images of the NASA Sector 33 game screen and select the most suitable route by applying all possible route combinations. In this case, the flight path is fixed. All aircraft are forced to use this limited flight paths which is inconsistent with reality. The parent agent can decide to change the direction of the aircraft only at any connecting waypoint (see waypoint in Section 3.2.1). The child agent adjusts the speed based on the combination of routes obtained from the parent agent where they have a list of six different speeds. In practice, resolving conflicts by changing the speed is avoided. Also, in reality, ATCO can change the heading of a specific aircraft at any angle to resolve a conflict while in [Brittain 2018] it is very limited.

Pham *et al.*[Pham 2019a, Pham 2019b] also applied the reinforcement learning method with the neural network-based agent using the aircraft conflict's trajectory data instead of using image data. The authors applied a single deep reinforcement network for two aircraft at the same altitude. The problem is then simplified to a two-dimension problem. The agent changes the heading of the aircraft in different directions for conflict resolution, each of which is called an action. Since the resolution action is not finite, the reinforcement agent resolves the conflict by applying an infinite number of actions. For each action, it gets rewards (rank) with either positive (successful) or negative (unsuccessful) feedback based on the quality of their conflicting resolution. Quality comes from selecting a set of features that is a kind of rule or condition. From this feedback, the model fits itself. The agent takes the current position, time, and current heading angle of the aircraft associated with the conflict as input and provides the new heading to resolve conflict. Wang *et al.* [Wang 2019] applied a similar approach with slight changes in the simulation of conflict scenario. For example, they defined an area (sector) with a radius of 60 nm. They considered a scene where there are some; then

a conflict is detected when a new aircraft enters this sector within 10 nm of entry. Then, the reinforcement model recommends heading change advisory action to resolve the conflict associated with this new aircraft. But this model also selects each action based on a reward function during model training.

Zhao and Liu [Zhao 2021] also applied reinforcement learning and CNN using image data where each image contains the current position of aircraft associated with the conflict. One of the major challenges in resolving aircraft conflicts is handling input dimensions. This is because most reinforcement learning models have been designed with the assumption that the number of aircraft is fixed. But in reality, it is not. These authors plotted an arbitrary number of aircraft within an image so that conflicts can be resolved even while the number of aircraft is variable. Therefore, it is possible to eliminate the dimensionality problem. However, a proper reward function is required for the best resolution of reinforcement learning and finding that function is the most challenging task because it depends on features like certain rules and conditions.

In general, all the models based on reinforcement learning are based on almost the same principle (rules and conditions to define reward function). The biggest challenge of applying the reinforcement learning method is to define a perfect reward function under different rules and conditions that validates each decision with different rewards. Because, it chooses the best decision based on this reward value.

Olive *et al.* [Olive 2018] applied an anomaly detection algorithm named auto-encoder that is based on a neural network to identify the anomalous from historical flight data. Here, anomaly means an irregular situation. Irregular situations usually occur when the initial flight plan is changed by a decision of the controller. When this model detects an anomaly, it means that the controller has taken some action for that trajectory. Although this method is not a direct solution for conflict resolution, it can be used to identify a situation for later data annotation.

2.4 Conclusion

Many models have been proposed to resolve aircraft conflicts. This chapter discusses some of the most important, recent, and popular studies within the last few decades.

The positions of aircraft are approximate. Therefore, it is risky to apply a condition-based mathematical model. Also, when the number of aircraft is high, there is a lot of run-time computation of the models and in many cases it is not possible to find a solution due to high combination.

Recently, the popularity of machine learning models has increased because this type of model takes a little more time for training but it responds very quickly. Since the machine learning model trains itself from examples, the performance in the case of resolving an unseen conflict scenario is very close to human.

In this chapter we have mentioned some machine learning models, all of them were designed based on the current positions of the aircraft. Model training is done through mathematical calculations for feature selection or feature extraction. This is risky with the aircraft's approximate position. Also, the reward function of the model based on reinforcement learning is designed with some rules and conditions.

As opposed to most of prior work, we trained our first neural network-based CRMLnet model using the last 5 minutes trajectory without any feature extraction. So, the model can learn the conflict scenario from the movement of the aircraft.

Secondly, to handle the arbitrary number of aircraft, we proposed another model based on a convolutional neural network, ACRnet, where we converted the whole conflict scenario into an image. Therefore, a conflict with an undefined number of aircraft can be resolved without changing the architecture of the model.

In both of our models, we provide multiple heading resolutions for a single conflict. Thus, ATCO can choose the best from multiple solutions so that future conflicts can be considered.

Chapter 3

Data related to aircraft conflicts

Abstract.

The first and foremost thing to consider in preparing a supervised model is a well-organized annotated dataset. Different data can be used to solve aircraft conflicts such as flight plans, trajectory data, controllers' immediate action orders and weather. Different data sources collect such data. However, since aircraft conflict data are sensitive and not publicly available, we had to create our own data sets. We used an open-source simulator named Blue Sky [Hoekstra 2016] to build a sequence-based trajectory data set. The data set consists of conflict trajectories and heading resolutions. While this resource fit some machine learning models, when there is a variable number of aircraft involved in the conflict, it can hardly be used in neural network based models that need a set dimension input. For this reason we developed a second data set which corresponds to the trajectory data transformed into images. Both datasets are delivered to the research community. They are freely accessible online at `https://independent.academia.edu/MDSIDDIQURRAHMAN9`. There is a total of 1,516 sequence data and a total of 1,656 image samples, of which 1,516 contain two aircraft and 140 contain three aircraft.

Contents

3.1 Introduction

The most crucial thing that is needed to build a strong prediction model is to have a good number of sample of annotated observations or data. Trajectory data is generally kept confidential and not publicly available. Thus, the most complex task is to synchronize the trajectories of the aircraft involved in a conflict and the immediate order issued by the ATCO to resolve the conflict. Another problem with real data is that there are not many variations on the conflicts. On the contrary, if we rely on simulated data, we can think of future conflicts and create variations of the conflicts, which may not be possible to easily collect from real data.

This chapter discussed how trajectory data can be built by simulating aircraft conflicts. We discuss in detail how to transform these trajectories into images. We also discuss why we converted trajectories into images and how we can benefit from this transformation.

This chapter begins with data that is primarily related to aviation collisions. Then the different sources of these data are discussed. Gradually discussing the different types of limitations, we started explaining more specific data that is mainly used in this research. Additionally, different types of data variations such as trajectory conversion into images and the benefits of using image data are discussed.

The main purpose behind creating this data was to create standard data for solving aircraft conflicts that could be used for machine learning applications. In addition, because this type of aviation data is not publicly available, we have made this data publicly available for future use by the research community.

Because the type of model to solve a conflict depends on the type of available data, the preparation of the data is actually very important. Most of the research in aircraft conflict resolution used trajectory data [Prandini 1999, Prandini 2000, Alonso-Ayuso 2013, Wandelt 2014]. In few studies, the controller's immediate orders were also combined with trajectory data to apply supervised models [Srinivasamurthy 2018, Kim 2016, Rahman 2020, Rahman 2022] including trajectory image data [Zhao 2021].

This chapter is organized as follows. Section 3.2 focuses on the types of data that are used to resolve an aircraft conflict. Then the different sources for these data are discussed in Section 3.3. Section 3.4 discusses more specifically data that we created for this research: trajectory data and trajectory conversion into images.

3.2 Types of data

There are four main types of data that are commonly used for conflict detection and resolution: (a) flight plans, (b) trajectory data, (c) ATCO's immediate actions, and (d) weather data. Although we did not use all these types of data in our thesis, it is important to be aware of all the possible data types.

3.2.1 Flight plan

A flight plan is one of the complete series of waypoints of a flight where latitude, longitude, and altitude are the basic elements used to distribute it. Waypoints indicate the intermediate and intersection points that pilots use to design the initial flight plans before departure. In other words, the waypoint is a fixed coordinate for specifying a single point on the Earth. Depending on where in the world, different technologies are used to create waypoints like radio beacons, buoys, satellites or control points. Figure 3.1 and Table 3.1 provide a sample flight plan from Toulouse city to Paris city in France where the identification number LFBO is for Toulouse airport and LFPO is for Paris airport. The identification numbers between LFBO and LFPO are the different waypoints of the planed route. A flight planner is responsible

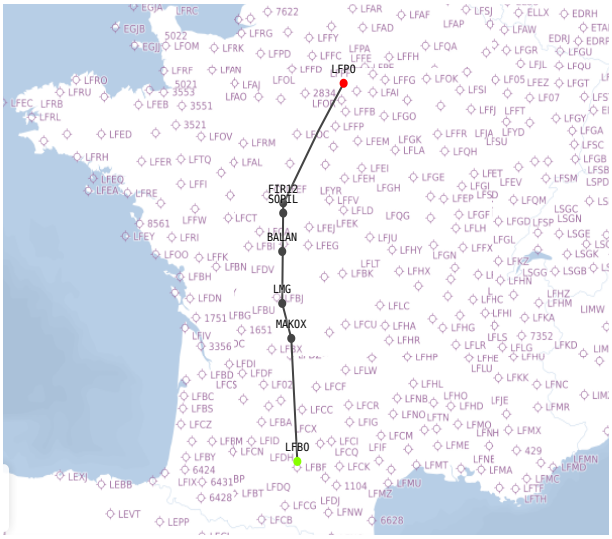


Figure 3.1: An example of a flight plan from Toulouse to Paris in France. The identification number for Toulouse airport is LFBO and the one for Paris airport is LFPO. There are several waypoints on the way from Toulouse to Paris such as MAKOX, LMG, BALAN, SOPIL FIR12 (see detail in Table 3.1).

Table 3.1: The flight plan from Toulouse to Paris in France contains a lot of information. 1^{st} column shows all the identification number of waypoints and 2^{nd} column shows the type of the point. LFBO and LFPO are the airports (APT). Also this table contains altitude (ft/m) and position (latitude/longitude).

ID	Type	Via	Altitude (ft/m)	Position (lat/lon)	Dist (leg/tot)	Name
LFBO	APT	-	0 / 0	43.62900 / 1.36397	- / 0 nm	Toulouse Blagnac
MAKOX	FIX	-	35,000 / 10,668	45.33280 / 1.23806	102 / 102 nm	
LMG	VOR	G36(AWY-LO)	35,000 / 10,668	45.81590 / 1.02558	30 / 132 nm	LIMOGES
BALAN	FIX	A34(AWY-LO)	31,300 / 9,540	46.51610 / 1.03333	42 / 174 nm	
SOPIL	FIX	A34(AWY-LO)	24,800 / 7,559	47.02580 / 1.05500	30 / 205 nm	
FIR12	FIX	A34(AWY-LO)	23,000 / 7,010	47.16670 / 1.05833	8 / 213 nm	
LFPO	APT	-	0 / 0	48.72630 / 2.36701	107 / 321 nm	Paris Orly

for designing a flight plan using waypoints before departure. According to the International

Civil Aviation Organization Doc 4444, a flight planner designs a flight plan by considering many parameters such as midair conflicts, amount of fuel, en route winds, airspace restriction, airport conditions and many more. This organization also defines an air route that is a fixed altitude corridor from a source location to a destination one. A pilot has to select a plan before the flight and follow the initial planned path unless asked to partially change it by the air traffic controllers. Flight plan data is very important for the implementation of the free flight concept. Another important aspect of the combination of flight plans and on-board data is that the pilot can continue the flight even if the aircraft is disconnected from the ground system for any reason.

3.2.2 Trajectory data

The location of any object on the Earth can be denoted by its coordinates. Similarly, each aircraft in the airspace is identified by the three basic coordinates latitude, longitude, altitude. These are the primary components of the aircraft coordinate system. Since an aircraft is a moving object in the sky, its 3D trajectory consists of a set of these coordinates. It is sometimes called the 4D trajectory where time is considered as the 4^{th} dimension [Wandelt 2014]. Although the flight plan of an aircraft is designed before it takes off, in reality, it is not possible to follow the flight plan completely in actual trajectory.

Figure 3.3 shows a view where the solid line represents a possible original flight plan and the dotted line represents a possible actual aircraft trajectory. The speed, the angle or the distance between two planes as well as many other parameters can be calculated from the elementary 4D components of trajectory data. From trajectory plans, one can calculate the

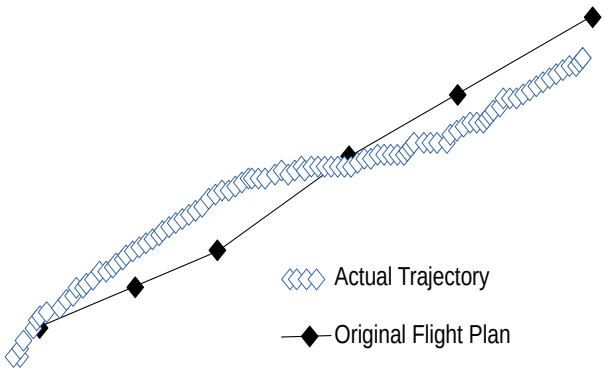


Figure 3.2: In reality, the actual trajectory differs from the original flight plan.

distance of any two airplanes, their speed, the angle between them. It is also possible to calculate the possible conflicts between them. Trajectory information is the first that air traffic controllers take into account for making any decision to resolve aircraft conflicts.

3.2.3 Air traffic controller's immediate action order

An immediate order is a voice communication between a controller and a pilot to guide them through conflict situations. Such an order aims at changing the aircraft trajectory to avoid a conflict situation. Pilots follow the ATCO's immediate order. The ATCO can choose different actions to properly guide the pilots through radio communication. According to Pavlinović *et al.* in [Pavlinović 2013], ATCOs' communication is defined in seven sections based on different aircraft phases (see Figure 3.3). Different ATCO operate at different phases. In our study, we

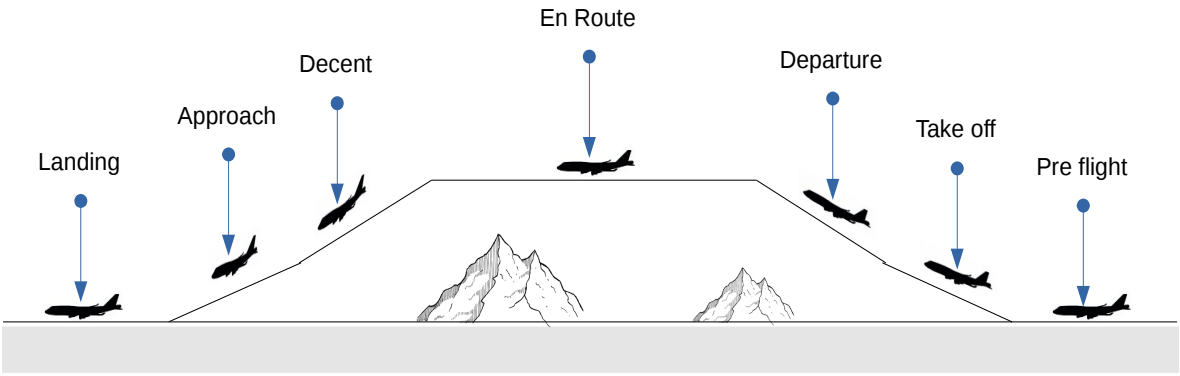


Figure 3.3: An aircraft has seven phases: landings, approaches, and descent, en route, pre-flight, take-off, and departure. En route is the top altitude level where the conflict resolutions are usually made by changing the heading direction of the aircraft.

consider the controllers' en route phase only where the altitude (height) of the aircraft usually remains unchanged. The common resolution maneuver is heading direction either turn left or right with a certain angle.

We also assume that the altitudes for all airplanes are set. Thus, the trajectory is simplified to a 2D instead of 3D trajectory. Therefore, the number of actions is limited and concern the aircraft heading (change left or right with any of the angles reported in Table 3.2) and its speed. ATCOs usually change the heading degree by a multiple of five (e.g. an immediate order could be "TURN LEFT 5^0 or TURN LEFT 10^0 as shown in Figure 3.5).

Table 3.2: The possible heading decision between 30 degrees left and 30 degrees right. The 2^{nd} row shows all the heading names (y_0 to y_{11}). The 3^{rd} shows each heading by changing 5 degrees up to 30 degrees on both the left and right sides. The 4^{th} row shows all the binary decision for each heading direction. Here '1' means it resolves the conflict while '0' means it does not.

	Left Headings						**Right Headings**					
Class label →	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}
Heading decision →	5^0	10^0	15^0	20^0	25^0	30^0	5^0	10^0	15^0	20^0	25^0	30^0
Binary decision →	0	0	1	1	1	1	0	0	0	0	0	0

3.2.4 Weather

There are many things that ATCOs keep in mind during conflict resolution. Weather is one of the them. The weather report consists mainly of wind speed, wind direction, thunderstorm, cloud, etc. An aircraft conflict may not only occur with another aircraft but also with bad weather. Since bad weather makes passengers uncomfortable and affects safe operation, the aircraft avoids it. Weather data is collected through various types of weather monitoring radars.

3.3 Data sources

In the previous section, different types of data have been described. This section discusses different sources of data. Basically three sources are available to collect such data: (a) open-source data, (b) radar data from ATC station, and (c) data simulation. Here, we describe them and explain why we had to develop simulated data which is one of the contributions of this thesis.

3.3.1 Open source data

Schäfer and Strohmeier *et al.* started OpenSky Network in 2012; it is one of the largest open-source aviation data source [Schäfer 2014].

The OpenSky Network mainly stores air traffic information around the world. It uses the ADS-B as a back-end technology to store the live data which is publicly available. The main functionality of the OpenSky Network is to provide public open access to ATC data. OpenSky Network collaborate with universities and government entities around the world. OpenSky Network uses around twelve tables in their database to store different ATC data including aircraft trajectories. The OpenSky Network also stores flight information such as flight paths that the aircraft has already followed. Among all the tables in the OpenSky Network database, one table named *state_vectors_data4* (see Table. 3.3) contains the basic trajectory information such as flight plan, aircraft heading change, callsign, and other information that is related to the trajectory. Table 3.3 includes trajectories only. The challenge to use such a data set is to synchronize the heading changes with the conflicts. Another issue is that there is no distinction in the heading changes between the ones that correspond to conflict situation solving and the ones that correspond to normal situations to follow the initial flight path. If the ATCO's orders were available, it would be possible to extract the trajectory from this table using *icao24, callsign* and *time*. Since the ATCO orders is sensitive, time-consuming and difficult to obtain, in our thesis we did not used this data.

The *state_vectors_data4* table contains about eighteen attributes for different data types. For example, each aircraft time is represented by *time* attributes. Some other common and important attributes are: *icao24* which is the unique aircraft identification number from

Table 3.3: Description of a table from OpenSky Network named *state_vectors_data4* that contains trajectory information for all the aircraft.

Name	Type
time	int
icao24	string
lat	dauble
lon	dauble
velocity	dauble
heading	dauble
vertrate	dauble
callsign	string
onground	boolean
alert	boolean
spi	boolean
squawk	string
baroaltitude	dauble
geoaltitude	dauble
lastposupdate	dauble
lascontact	dauble
serials	array<int>
hour	int

the ICAO[1], *lat, lon, velocity* which correspond to latitude, longitude, and ground speed per second, the *heading* parameter that contains all the changes in heading degrees clockwise from geographic north and *callsign* which is a broadcast identification number spread by the aircraft itself.

Although no conflict scenario data is provided by the OpenSky Network directly, we can download the synchronize historical dataset from OpenSky Network which is associated to a given ATCO's immediate order that includes the *time* and *icao24*.

3.3.2 Radar data from ATC station

ATC is a ground-based service advisory provider for aircraft collision avoidance. ATC's main job is to assist the pilot in flying with all kinds of information and advice from the ground. Depending on the country's system, the ATC also provides military defense assistance based on the capabilities of some countries. Although most modern aircraft have radar systems for aircraft collision avoidance, this is only possible when the aircraft comes very close (short-range conflict). Therefore, for mid-range or long-range collisions, the pilot needs the help of an ATC ground system. Typically all ATC stations store all information obtained from radar for future investigation. The primary thing that is kept in all that information is the trajectory of each aircraft. Here, trajectory refers to the positioning parameters of each aircraft such as latitude, longitude, altitude, heading direction, speed, etc. Generally, ATC stations store all aircraft location data in their database. Additionally, they store the immediate order from

[1] *International Civil Aviation Organization*

ATCOs. Each control station decides the type of data it stores.

In France, control stations use the IMAGE system to calculate the aircraft position [Hurter 2013]. The IMAGE system gathers data from all controller stations. The system receives multiple positioning information for each aircraft from different radar sources and calculates the estimated position. Although ATC stations store a variety of information, including trajectory data, this data is highly sensitive and not publicly accessible.

Alternatively, researchers simulate trajectory data for their research. One of the huge advantages of using simulated data is that one can create many variations of a conflict scenario that may not be available in real data in order to test the built models. A model can also then be trained with a variety of conflict data. For organizations were real data are available, the model can be tested with these data.

3.3.3 Simulated data

Data simulation is used to build a hypothetical set of mathematical, logical, and symbolic relationships between entities of interest in order to predict the system performance with a real-world process or simulation of system activity over time [Banks 2005].

This is specifically useful when there is no adequate real data. Using simulated data, a model can be trained by creating some hypothetical data to predict the real-world problem. Many studies on aircraft conflict detection and resolution used simulated data [Hu 1999, Eby 1999, Farley 2007, Pritchett 2017, Siqi 2018, Brittain 2018], including anomaly detection studies [Das 2010, Olive 2018].

Hu in *et al.* used the Brownian Motion [Mörters 2010] method to simulate random conflict scenarios in [Hu 1999]. This method was first applied to random movements in liquids or gas particles. Hu *et al.* used that concept considering each aircraft as a single gas particle and generate aircraft' movement using this method. In the case of conflict resolution, they resolved the conflict by changing the direction of the heading using mathematical formulas. They considered the current location of each aircraft only. In almost all the studies, data are simulated by generating position (latitude, longitude, altitude), time, speed, direction, etc. for a specific position. Brittain and Wei generated each conflict scenario with a video game simulator called NASA Sector 33, which shows the collision between different planes [Brittain 2018]. They used the game screen images and applied CNN to resolve the conflict.

Although some of the data mentioned above could have been used in our work, the biggest obstacle was that none of this data is publicly available. Moreover, as opposed to conventional data used in the related works considering only the current location of the aircraft, the data we generated consists in all the locations of an aircraft flying for 5 minutes; this has not been used before. This has the extra advantage that a model can learn the conflict environment from the aircraft movement without any other features.

3.4 Simulated data sets

In this research, for each conflict we prepared 5 minutes of trajectory data of all the aircraft involved in that conflict and the immediate order from the ATCO to resolve it. Such data is difficult to collect from ATC stations since there is the need to synchronized the trajectory data with the immediate order of ATCO. After having tried to get real data from ATC stations without any success, we have decided to simulate them and to create a new data set. We decided to open it to the research community.

We needed trajectory data that would contain not only a single position but multiple continuous positions to fit the model without feature extraction. As a result, our data set differs from the data used in the literature in that our data is not just a single position of the aircraft involved in the conflict. Rather, the trajectory of the last five minutes of the aircraft is considered. We call it a 5-minute window. The model can learn the collision environment from the movement of aircraft and can avoid the risk of a feature that is difficult or impossible to calculate.

We have created a dataset with multi-label annotations that means different possible solutions are given for each conflict sample. This is different from existing data set in which only the best solution is recorded. Our data set has thus a multi-class annotation. Indeed, we have considered multiple resolutions because resolving a conflict may provoke a new conflict. In that case, the multi-label solution will give ATCOs the opportunity for alternative resolutions.

As opposed to some previous studies [Prandini 1999, Prandini 2000, Kim 2016, Pham 2019a, Pham 2019b], this research mainly focuses on the raw data rather than feature extraction to avoid the risk of an unusable feature. Since the position of all the planes is approximate, it is risky to use the feature extraction under different conditions.

We explain the preparation of the two data sets we created : (a) simulated trajectory along with the ATCO's immediate orders (b) images obtained by the conversion of trajectories.

There are some limitations when using trajectory data. For example, the input dimension of a model depends on the number of aircraft changes. Since the input dimensions of a model cannot be changed at run-time, most trajectory-based machine learning models use a fixed number of aircraft. If we want to use recurrent neural network-based model such as long short-term memory (LSTM[2]), the computational complexity of the model increases with the number of aircraft. Since we use 5-minute trajectory data, it is more complicated than the other trajectories considering a single position.

The solution to a conflict where a variable number of aircraft are considered is a big challenge in this research. To overcome this challenge, inspired by [Zhao 2021, Brittain 2018], we convert sequence-based data into images. The advantage of converting trajectory sequences into images is that the size of an image does not depend on the number of planes plotted in

[2] *Long Short-Term Memory*

that image. If we included weather, restricted zones, etc., the size of the image would not change.

By converting simulated trajectories into images, we can also easily apply data augmentation to make the data sufficient for model training. As a result, if two or more aircraft are involved in a conflict, the model can be trained and tested without any changes in the input dimensions.

For both datasets, the output annotations are the same, which are the immediate orders of the ATCO for resolving the corresponding conflict. Sub-section 3.4.1 presents the simulated trajectories and ATCO's immediate orders while sub-section 3.4.2 presents the transformation of the trajectory data into images.

3.4.1 Dataset-STIO: simulated trajectory and ATCOs immediate orders

We generated the trajectory and controller's immediate order datasets using an open-source simulator named Blue Sky developed at TU Delft by Hoekstra and Ellerbroek [Hoekstra 2016]. Using this simulator, we made examples of conflict scenarios with ATCO's orders.

Table 3.4 reports a sample[3] data with a conflict situation between two airplanes. In this table, the first column reports time in seconds. Updating trajectory information varies depending on the radar refresh time. We simulate trajectory data that updates every 5 seconds. The next three columns in this table are aircraft A coordinates (latitude, longitude, and altitude). Similarly, the next three columns are aircraft B coordinates. The last column is the controller action. To populate this table, we played different scenarios with the conflict situations. Figure 3.4 shows an example of left-heading actions that are taken up to 30 degrees by a multiple of five to resolve the conflict between two aircraft (aircraft A and aircraft B).

Table 3.4: A sample trajectory data where two airplanes are going to have a conflict. The first column shows trajectory data updates every 5 seconds. The next eight columns are aircraft A coordinates (latitude, longitude, altitude, and heading) and aircraft B coordinates. The last column is the controller action.

Time in Sec	Lat (A)	Lon (A)	Alt (A)	H (A)	Lat (B)	Lon (B)	Alt (B)	H (B)	Controller's Action(s)
0	46.2499689	-2.8	7620	90	43.8	-6.19443506	7620	105	
5	46.23975835	-2.8	7620	90	43.8	-6.18028833	7620	105	
10	46.22954973	-2.8	7620	90	43.8	-6.16614426	7620	105	
15	46.2193411	-2.8	7620	90	43.8	-6.1520002	7620	105	
20	46.20913248	-2.8	7620	90	43.8	-6.13785613	7620	105	TURN
25	46.19892385	-2.8	7620	90	43.8	-6.12371206	7620	105	LEFT (A) 10^0
30	46.18871522	-2.8	7620	90	43.8	-6.10956799	7620	105	
...	...	...	...	...	...	...	...	...	
...	...	...	...	...	...	...	...	...	
290	45.65786669	-2.8	7620	90	43.8	-5.37407647	7620	105	
295	45.64765807	-2.8	7620	90	43.8	-5.3599324	7620	105	

[3]A sample corresponds to a conflict situation.

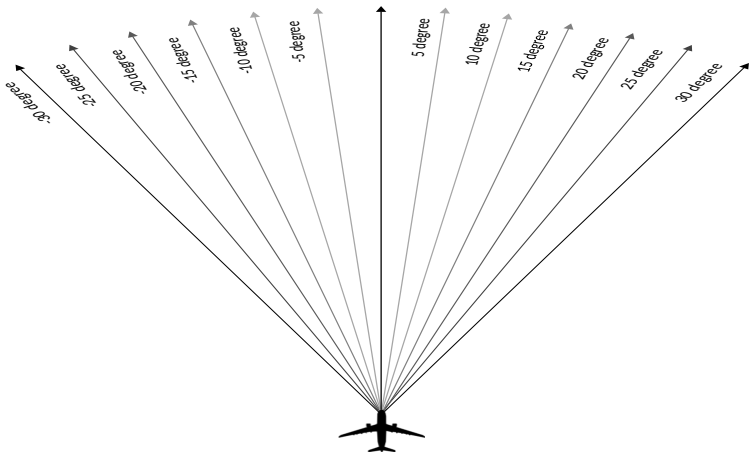


Figure 3.4: Possible heading resolution for each conflict sample data. Here the ATCO's order can be left/right heading actions up to 30 degrees by a multiple of five to resolve the conflict.

There are many advantages to create simulated data. First, we create sample scenarios. Second, many variations can be created.

According to our objectives, we generated different conflict scenarios for a pair of aircraft where a single instance contains every 5 seconds following a 5-minutes window of trajectory for each aircraft and the resolutions. Therefore, to create a conflict scenario, we consider two

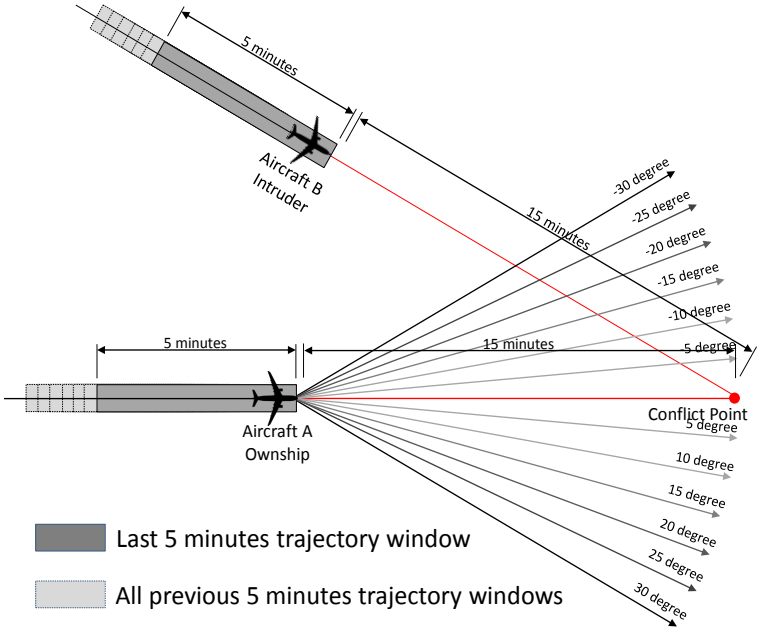


Figure 3.5: A conflict can be resolved within the heading range between left 30^0 and right 30^0. Here, the decisions are for the Ownship only. The Ownship refers to the aircraft following its initial flight plan.

planes in such a way they can create a conflict situation. Both aircraft's position is 20 minutes away from the conflict point. For both aircraft, we store 5 minutes of trajectory data that is just before the conflict detection. Therefore, after detecting the conflict we have 15 minutes to reach the conflict point. The reason we took the data just before collision detection is our model makes the resolution decision based on it. If we compare it to a real-life situation, whenever a conflict is detected, our model will apply prediction based on the data of the past 5-minutes of that time. The basic environmental parameters in our data are latitude, longitude, altitude, speed of both planes and angle between them.

Figure 3.5 shows a simulated scenario with possible resolutions (range: from left 30^0 to right 30^0).

It is not possible to record voice command in the simulator, we thus built and store trajectories along with the conflict situations and the text commands to simulate the ATCOs immediate orders.

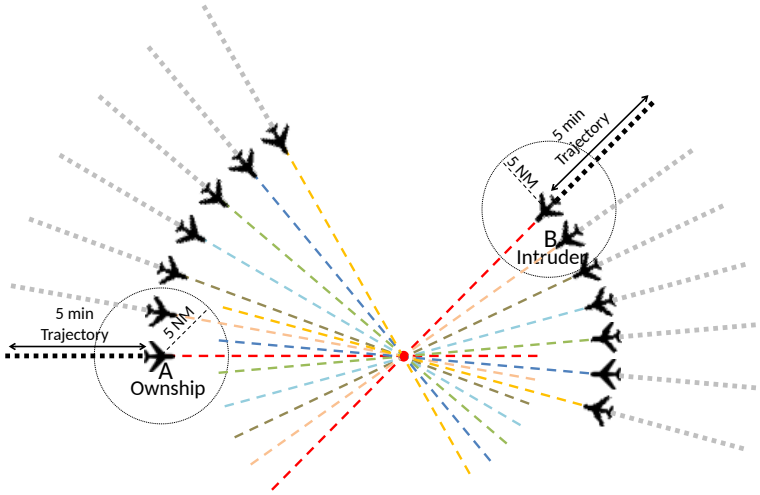


Figure 3.6: The rotation of the whole scene creates a new sample with the same annotation.

As shown in Figure 3.5 there are several appropriate solutions to solve the conflict and they are stored in our simulated data. So, for each conflict scenario, we saved multiple resolutions in binary decisions as shown in Table 3.2.

We applied different techniques to augment the data. For example, rotating a whole scenario does not change the decision; we also changed the speed by different values to create more samples. Figure 3.6 shows an example of rotation of the whole scenario. We only applied data rotation augmentation to increase the number of samples because shifting, zooming, flipping, etc. may affect the labeling. The rotation was made from 5^0 from 170^0. There are geometric formulas for rotating lines and we used them.

We also augmented the data set in another way: we split the two parts of each scenario in such a way that the time slot for one was at 0 second, 10 seconds, 20 seconds, up to 5 minutes. In the same way, the other one was for 5 seconds, 15 seconds, 25 seconds, up to 5

minutes. Even if we split one scenario into two scenarios, it will be two separate scenarios because it is a different time and coordinate position.

We have generated 1,516 sample scenarios and the corresponding valid commands to resolve each scenario. A scenario can be solved with different heading angles. Table 3.5 shows the number of categorical actions (2 solutions, 3 solutions, ..., 6 solutions) occurrences to resolve the conflict scenarios. The total conflict samples with two aircraft can be categorized based on the number of their solutions: [288, 2], [288, 3], [300, 4], [372, 5], and [6, 268] where the first value of each pair is the number of samples and the second value is the number of solutions. The data set consists of almost a similar number of samples in the different categories where the highest frequency is in the case of 5 solutions (see Table 3.5).

We also created 140 samples for the conflict between the three planes. Table 3.6 shows the total data with three planes.

It takes 10-20 minutes to create one sample.

Table 3.5: Number of samples based on their solutions, depending on the number of possible solutions. For instance, 288 samples of conflict situations have two solutions.

Number of conflict samples	288	288	300	372	268
Number of heading resolutions	2	3	4	5	6

Table 3.6: Number of samples with 3 aircraft (same notation as in Table 3.5).

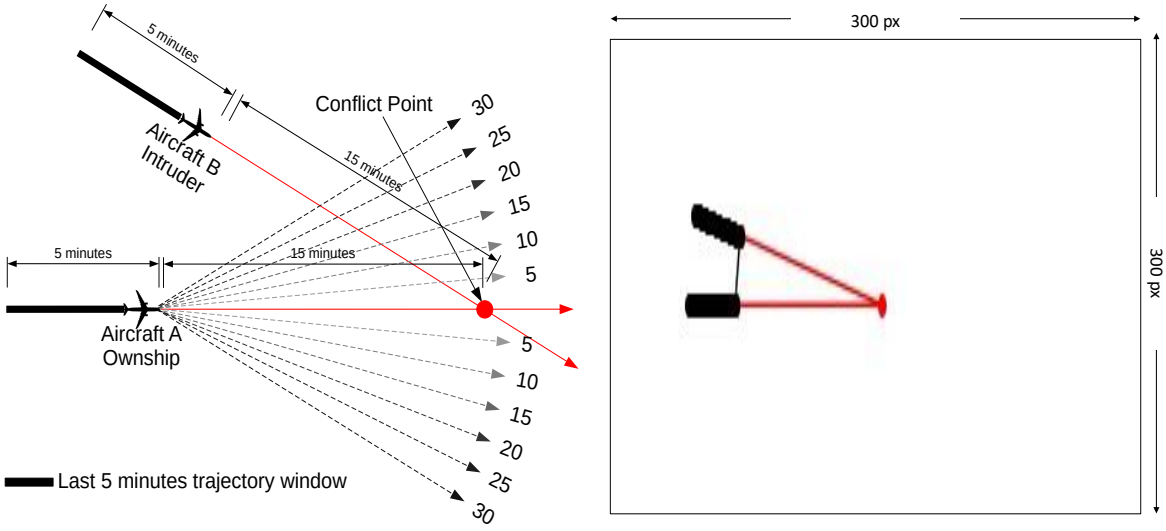
Number of conflict samples	4	4	4	56	72
Number of heading resolutions	1	2	3	4	5

3.4.2 Dataset-CTI: convert sequence data into images

This section explains how a trajectory sequence can be transformed into an image.

To create image data, we used the trajectory data that we described in the section 3.4.1. According to Figure 3.7 (a), each conflict sample of initial data contains the last 5-minute of the trajectory (a series of positions) for each aircraft associated with the conflict. Also, the resolution can be made for each conflict by changing the ownship's (see Figure 3.7 (a) aircraft A) heading from 30 degrees left to 30 degrees right by 5 degrees (total 12 heading directions). Each conflict is associated with multiple heading resolutions (multi-labels) for a single conflict.

In Figure 3.7, (b) shows the image converted from the trajectory data. It indicates the distance from the conflict point to the aircraft (red lines). The thin black line indicates the current distance between aircraft; it is the last point of the aircraft before the conflict is detected. It is possible to plot trajectories of multiple planes without changing the size of the



(a) Conflict scenario (b) Converted image

Figure 3.7: Aircraft conflict trajectory with heading resolutions and conversion into an image. (a) the black solid line just behind the plane represents the last 5 minutes of the trajectory before the conflict is detected. All dotted lines show possible heading changes to resolve the conflict (b) plots the positioning coordinate of the last 5 minutes where the black line is the current distance between aircraft and the red lines are the distance between the aircraft and the conflict point.

image. Thus, there will be no change in the input size if the number of aircraft changes at run time.

For example, Figure 3.7 shows the last 5-minute trajectories of the aircraft associated with the conflict. The image size is $300px \times 300px$ that is a fixed size.

Converting trajectory data into images not only solves the problem of input dimensionality but we also can apply image augmentation to increase the volume of the training data. Typically, image augmentation includes rotation, shifting, zooming, flipping, etc. In this thesis, we applied Python Keras built-in data augmentation technique. We applied rotation only. We could not apply some of the usual data augmentations because the resulting samples may be labeled differently. For example, in a conflict scene flipping will not be labeled the same class. In Figure 3.7 (a), this conflict can be resolve with left-heading 15^0, 20^0, 25^0, and 30^0. If we flip the whole scenario horizontally, then the resolutions will no longer be the left-heading, all will be the right-heading. In the case of data augmentation, Python Keras randomly augments (rotation in our case) the training data in each epoch. The model then trains and tests. There will be no data overlap between epochs. Python Keras uses different rotation angles to do this. The augmentation techniques are applied to training data only.

To transform the initial trajectory data into images, we plot the trajectory sequences of each conflict scene one by one. We uses *python matplotlib* [Ari 2014] library to transform all the trajectory sequences into images. We created a total of 1,656 image samples, of which

1,516 images contain two aircraft and 140 contain three aircraft.

Brittain and Wei used the image of NASA sector 33 game screen [Brittain 2018] for their model training. We could train our model with the image of the screen of the simulator. But in reality, if we take a snap of the ATC radar display screen, there is noise (pixels with unknown information) besides the trajectory, which may lead the model to a wrong decision to resolve the conflict. So, we just plotted the trajectory so that no other additional information comes into the image.

The benefits we get from converting trajectory data into images are mentioned below:

(a) The size of each of the sample data (image) remains the same although there is a variable number of trajectories plotted;

(b) The input dimension will remain unchanged even if weather restrictions areas were added in the future;

(c) Converting into an image means that not only we can classify conflict among aircraft, but also we can classify conflict between an aircraft and the restricted areas such as military zone, weather storming area, and so on;

(d) All the advantages of image classification can be applied to the data, such as using the CNN model and in that case, the model effectiveness can be increased by using data augmentation.

Chapter 4

Supervised machine learning and multi-label classification of aircraft heading changes

Abstract.

An aircraft conflict occurs when two or more aircraft cross at a certain distance. Aircraft heading changes are the common resolution at the en-route level (high altitude). One or more alternative heading changes are possible to resolve a conflict. We consider this problem as a multi-label classification problem. We developed a multi-label classification model which provides multiple heading suggestions for a given conflict. This model we named CRMLnet is based on the use of a multi-layer neural network ans uses trajectory data. It classifies all possible heading resolution in a multi-label classification manner. When compared to other machine learning models that use multiple single-label classifiers such as SVM, KNC, and LR, our CRMLnet achieves the best results with an accuracy of 98.72% and ROC of 0.999. We used the simulated data set presented in Chapter 3.

Contents

4.1 Introduction

In the domain of air traffic, two or more planes are considered as in a conflict situation when their trajectories cross each other in certain circumstances of distance at the same time [Kuchar 2000]. A controller considers various types of information to solve a conflict. The most common and preliminary information is the coordinate position of the involved aircraft.

The resolution of the aircraft conflict varies at different altitude levels. Figure 3.3 shows the seven usual phases of an aircraft. The conflicts have different resolutions. Air traffic controllers usually resolve conflict by changing the heading direction of the aircraft at this level. For example, the heading direction of flight of an aircraft is changed to a certain degree to the right or to the left. Multiple heading resolutions are possible to resolve a conflict. This is why we considered the problem as a multi-label classification problem.

Decisions to solve conflicts are made manually in real-time and consist of changing aircraft trajectories to maintain a safe distance between planes. When a conflict is identified, the ATCO has to make a quick decision about the best possible solution using his/her knowledge and experience. ATCOs have to take into account all the aircraft flight parameters such as its speed, positioning coordinate, destination, flight plan, its environment, weather, wind direction, military zone, etc. and the other flights.

Because a conflict resolution can be multi-labeled, in this chapter, we propose a multi-level classification model of conflict resolution based on a multi-layer neural network we named CRMLnet. It aims to help the controllers in their decision to provide the different possible heading advisories.

In this chapter, we consider the mid-range conflict only. We consider the 5-minutes of involved aircraft trajectory data just before the conflict is detected. In the real-life situations, an ATCO is automatically alerted whenever such a conflict situation arises; they then decide to change flights and gives the pilot the order regarding the flight change via radio communication.

Related work on air traffic conflicts using trajectories considers the aircraft current position. Then different mathematical calculation like future position projection, speed, distance between the involved aircraft, etc are used. Many parameters sometimes from different sources such on-board data are needed. As opposed to that, we have created a dataset where each sample contains trajectories of the last 5 minutes for each aircraft (See Section 3.4.1). Therefore, our model can learn the conflict environment from the movement of aircraft. We do not

need any features extraction such as calculating the distance between them, projecting the future position, etc.

In this chapter, we propose a model based on a neural network that aims to assist the ATCOs with multiple solutions to resolve a conflict. We designed the NN[1]-based multi-label classification model, develop and evaluate it. Besides, we developed other models based on multiple single-label classifiers (support vector machine SVM, logistic regression LR, and K-nearest neighbor classifier KNC) where we used separate classifiers for each output class-label solution and compared them with the neural network-based model.

4.2 From traditional machine learning to neural network for conflict resolution

We designed and developed a ML[2]-based model that will predict the actions for any new scenario using different kinds of information that an ATCO takes into account although we focus on sequence-based trajectory data.

We can think of many ways to resolve a conflict situation. It can be cast into a *ranking problem* where conflict resolution actions can be ranked; the top one being the most appropriate one. The advantage of this solution is that constraints can be easily added considering variables such as delay, proximity to destination, and flight time as used by Archibald *et al.* [Archibald 2008]. The problem of aircraft conflicts can also be considered as:

(a) A single-label or binary classification problem where the classifier simply classifies whether it is solvable or not;

(b) A multi-class classification problem where the classifier selects only the best one from multiple resolutions;

(c) A multi-label classification where the selection of resolution will be one or more for a single conflict.

We proposed a multi-label classification model. The multi-label resolution is more applicable in real life because a controller will have multiple alternative solutions in hand where it will be much easier to avoid risk. ATCO can take one of the solutions thinking of the other aircraft's, which are not involved in the conflict, position to avoid additional future conflicts.

Subsequent sections discuss classification methods.

[1] *Neural Network*
[2] *Machine Learning*

4.2.1 Single-label or binary classification

A single-label or binary classification is a two states classification. This type of classification is the most commonly used classification, especially when it comes to data sample identification. Figure 4.1, for example, shows the image identification of cats and dogs. The class of image data is labeled with 0 or 1. Here, 0 means cat and 1 means dog. Binary classification is also called single-label classification.

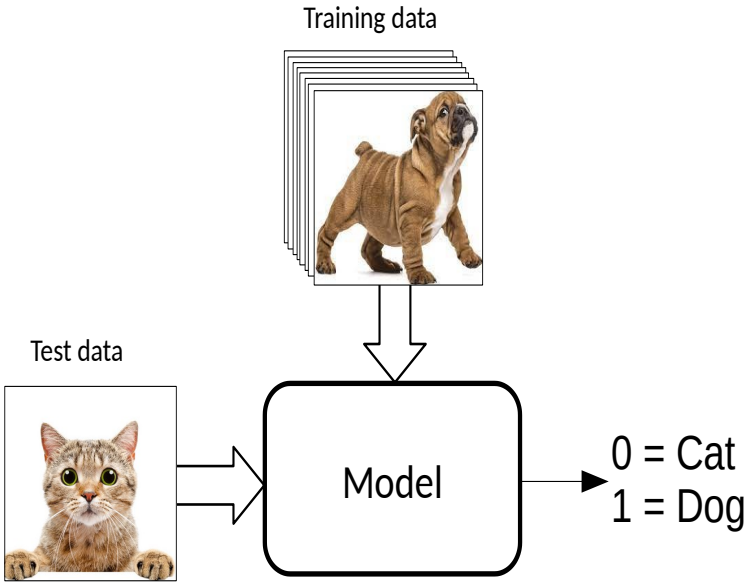


Figure 4.1: A binary classification output can be either 0 or 1. Here 0 means cat and 1 means dog ; the output is either cat or dog.

4.2.2 Multi-class classification

When a dataset is annotated by more than two class labels, it is called multi-class.

For example, Figure 4.2 illustrates a model trained on images which are numbers between '0' to '9'. The output of this type of model is the one selected according to the highest probability. One thing to note here is an image belongs to a single-digit. A special kind of encoding is used to select the output called one-hot encoding [Harris 2010]. One-hot encoding output is presented by a group of bits, where only one bit is possible to be '1', all other bits are '0'. Also, the sum of the probabilities of all the outputs is '1'.

4.2.3 Multi-label classification

According to the early discussion, there are different types of data classification: single label, multi-class, and multi-label. In all these classifications, multi-label classification is a bit

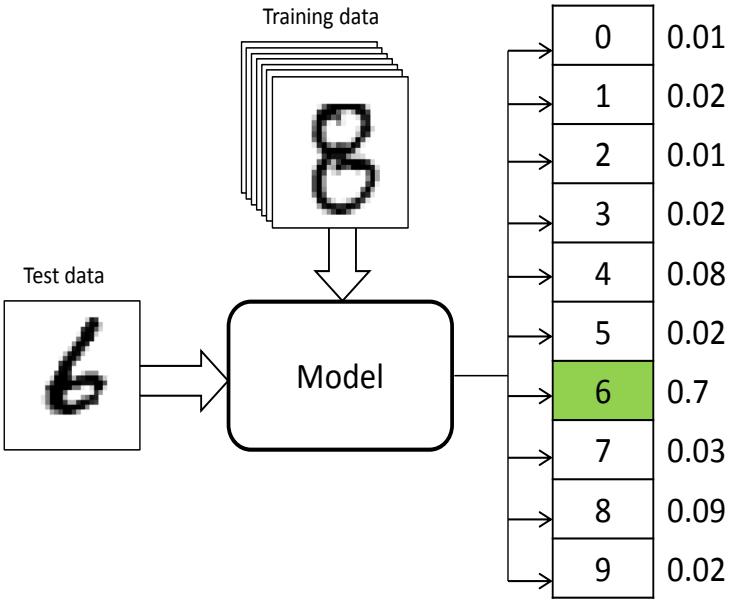


Figure 4.2: A multi-class classification output can be one of more than two class labels. This figure shows a multi-class classification model trained with the images of decimal numbers. The model gives one of '0' to '9' as output which is the most probable one. The multi-class classification model gives a unique output .

different. In multi-label classification, each sample corresponds to a set of labels $Y \subseteq L$.

When each sample corresponds to a single-label ($|L| > 1$), it corresponds to a single-label classification problem. It is called a binary classification problem if $|L| = 2$. If $|L| > 2$, then it is called a multi-class or a single-label multi-class classification problem.

A multi-label classification is slightly different from the multi-class classification in terms of input and output. The main classification difference between multi-class and multi-label is that multi-class is a single-label problem of classification into one of more than two classes whereas there is no limit to how many labels a sample can have in a multi-label classification. It not only gives the best probable one as an output, like multi-class, but it also gives all the possible outputs using an individual *sigmoid* activation function for each class label. Figure 4.3 shows that the model provides an individual probability between '0' and '1' in the output for each class label. Here, there is no such one-hot encoding as a multi-class that the sum of the probability scores will be '1'. The dataset is also different from the dataset used for the multi-class classification models.

In the case of a multi-class classification, Figure 4.2 shows that each sample image contains only one class label information whereas Figure 4.3 shows that each sample image may contain more than one class of information for multi-label classification. This means that in the case of multi-label classification, a model is trained in such a way that the datasets are annotated with multiple labels. Thus, multi-label classification output does not simply depend on the best probable one. As one-hot encoding is not used in the output, this means that the probability score is provided separately for each class label.

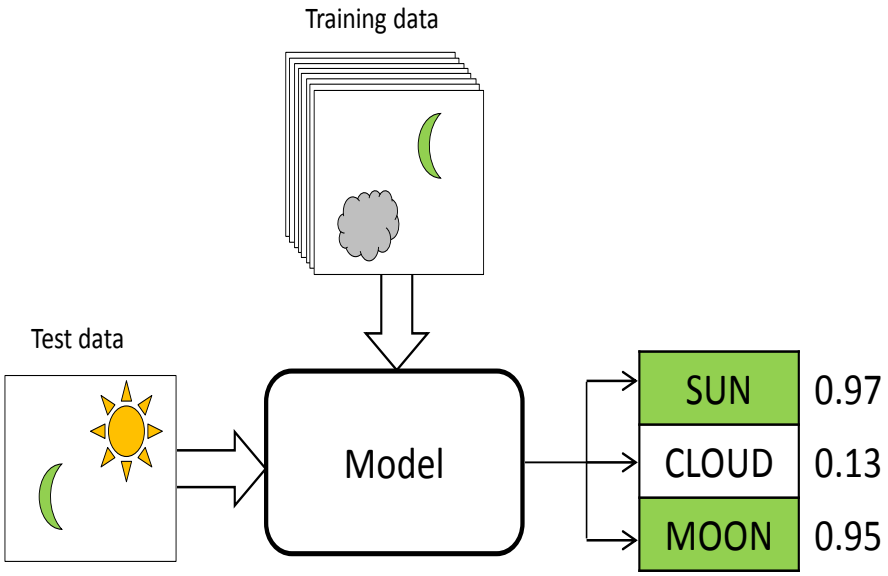


Figure 4.3: The figure shows that a sample image can contain multiple objects. Multiple outputs can be true. For example, this model is trained with images that contain three objects: the sun, the moon, and clouds. The test image contains both the sun and the moon, so the output is for the sun and the moon.

Figure 4.3 shows an example of multi-label classification. In this dataset, each sample image contains objects: the sun, the moon, and the cloud. One single image can contain multiple objects. Here, the test image contains both the sun and the moon. Thus, the output of that test image should give a high probability for those two class labels. It is noted that if the output probability is 0.5 or more for any object, then it is considered as true otherwise it is considered as false.

4.2.4 Classification algorithm

Different algorithms can be used for multi-label classification problems. And a multi-label classification problem can be defined in one or more single-label classification problems [Tsoumakas 2007]. In this chapter, we present four different machine learning classification algorithms to design models for classifying multi-labeled datasets: Support Vector Machine (SVM), K-Nearest Neighbor Classifier (KNC), Logistic Regression (LR), and Neural Network (NN) and how they can be applied to aircraft conflicts.

4.2.4.1 Support Vector Machine

SVM is one of the most widely used and highly popular supervised algorithms [Cortes 1995]. The main reason for the popularity of SVM is that this algorithm gives a high performance for binary classification based on training examples. On the other hand, it also works well for

non-linear classification in addition to linear classification. It uses kernels that actually map the input vector to a high dimensional feature space. SVM performance depends on a good separation of the training data by an hyperplane (also called functional margin). The more separation there is, the lower the generalization error. The basic formulation of a Linear SVM is: suppose $(\vec{x}_0, y_0)$,...., $(\vec{x}_n, y_n)$ where n number of training examples, $\vec{x}_i$ is the input vector, and y_i is the corresponding class-label. Here, each $\vec{x}_i$ is a vector of p-dimension where SVM tries to tune the hyperplane with a maximum margin so that the training data of the group $y_i = 1$ is well separated from the group of $y_i = 0$.

Now if we think about each heading change separately those individually stand on a binary state. For example, either a particular heading change can solve the conflict or not. In such case, we can use one individual SVM for each heading change where all the individual SVMs perform binary classification.

4.2.4.2 K-Nearest Neighbor Classifier

KNC is a classifying algorithm that classifies training samples based on closest examples. KNC is one of the most basic and simple classification algorithms. In this algorithm, $\boldsymbol{K}$ refers to the number of nearest neighbors that the KNC classifier uses to classify. After that, it classifies the new sample based on the majority vote of its $\boldsymbol{K}$ neighbors. A neighbor is chosen by using a distance function. The common distance functions are *Euclidean*, *Manhattan*, and *Minkowshki* and the equations are as follows:

$$d_{Eucliean} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{4.1}$$

$$d_{Manhattan} = \sum_{i=1}^{n}|p_i - q_i| \tag{4.2}$$

$$d_{Minkowski} = \left(\sum_{i=1}^{n}(|p_i - q_i|)^m\right)^{1/m} \tag{4.3}$$

Here all the distance function formulas calculate the distance between two points p_i and q_i. For the *Euclidean* distance, it is always 2-norm distance while for the *Minkowshki*, it is defined by a variable m. Depending on the dataset, any of them can be used. Since KNC is a supervised learning classifier, the output category of each data for training is already known. Now suppose a new sample comes and it needs to be classified. KNC uses its distance function to calculate the distance of the new sample from the already known category and find the $\boldsymbol{K}$ number nearest neighbor. The new sample is identified for the category that has the highest number of among $\boldsymbol{K}$ neighbors. The value of $\boldsymbol{K}$ is selected by hyper-parameter tuning.

Since each of the heading decisions resolving a conflict is a binary classification, we can apply KNC separately for each heading decision. So for each heading decision, the total data is divided into two categories, for example, category '0' and category '1'. Then we can apply

KNC individually for each heading decision as a binary classification. So, a KNC will be set up to determine if a heading decision can resolve a particular conflict situation. Here '0' means cannot resolve the conflict and '1' means it can. Similarly, we can apply KNC for rest of the heading. Finally, we will get all the heading decision separately for a single conflict.

4.2.4.3 Logistic Regression

LR is another machine learning technique that actually comes from the field of statistics. Both linear regression and logistic regression algorithms have the same goal but the difference is that the output prediction result of the logistic regression goes through a non-linear function called a logistic function. Sometimes it is also called a *sigmoid* function. This function can express any real value into a value between '0' and '1', but they are never exactly equal to that. The following equations are used for the hypothesis of logistic regression:

$$h_\theta(x) = g(\theta^T x) \quad \text{where} \quad g(z) = \frac{1}{1+e^{-z}} \tag{4.4}$$

Here $h_\theta(x)$ is the hypothesis of the logistic regression where θ is the hyper-parameter, x is the input, and the function $g(z) = \frac{1}{1+e^{-z}}$ is the *sigmoid* function. So if any real value goes through this *sigmoid* function, it becomes a number between '0' and '1' that is $0 < h_\theta(x) < 1$. We can use complex parameters to create more complex decision boundaries. There are many types of regression algorithms such as linear, polynomial, non-polynomial, multiple, logistic, etc. Only logistic regression is discussed here as it is widely used for binary classification.

We can apply this binary LR classifier to classify each of the heading decision individually while the model based on multiple LR resolving a single conflict. So for each heading direction, dedicated LR provides binary decision as an output separately using the same input trajectory.

4.2.4.4 Neural Network

A NN is a network of simple and strongly interconnected elements called nodes or neurons or perceptrons. Each node is a single computational unit. One or more weighted inputs are connected to each node for the computing process and it goes to the output through a non-linearity function called the activation function. The nodes are organized in different layers in a meaningful way. Usually, a NN is organized in several layers: input layer, hidden layer(s), and output layer.

- Input layer: This layer takes the input. The size of this layer is equal to the number of input variables.
- Hidden layer(s): They are the intermediate layers between the input layer and the output layer. There can be one or more layers. The number of nodes is often chosen empirically.

- Output layer: This is the last layer that contains the end results. The number of nodes equals the number of class-labels.

The basic notation of a NN is as follows:

$$f(x) = W^T X + b \quad \text{where} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ : \\ w_n \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} x_1 \\ x_2 \\ : \\ x_n \end{bmatrix} \tag{4.5}$$

Here X is the input vector that contains all the input features and W is the corresponding weight vector. The basic formula $f(x) = W^T X + b$ calculates a product of input feature (X) with a transpose of weight (W) matrix and then includes a bias (b). Equation 4.6 uses a linear function in the case of n variables.

$$f(x) = b + w_1.x_1 + w_2.x_2 + w_3.x_3 + ... + w_n.x_n \tag{4.6}$$

Here, we have presented different types of classifiers. In the following sub-sections we discuss on how the problem of aircraft conflict resolution can be formulated to apply these models.

4.2.5 Problem formulation

We discussed in Section 4.2 that we will apply multi-label classification. This section details the mathematical formulation.

Let us consider, a single-label classification of n instances. A dataset D can be composed as $(\vec{X}_0, y_0)$, $(\vec{X}_1, y_1)$, $(\vec{X}_2, y_2)$, ..., $(\vec{X}_n, y_n)$ where $\vec{X}$ represents each input feature vector and y represents the corresponding class-label. For multi-label classification, each input feature $\vec{X}$ has a subset of labels $\vec{Y} \subseteq \vec{L}$ where $\vec{L}$ is multiple labels and $\vec{Y}$ is a subset of $\vec{L}$. Therefore, the dataset D is composed of multi-label classification of n instances $(\vec{X}_0, \vec{Y}_0)$, $(\vec{X}_1, \vec{Y}_1)$, $(\vec{X}_2, \vec{Y}_2)$, ..., $(\vec{X}_n, \vec{Y}_n)$. In our case, each $\vec{Y} = [y_0, y_1, y_2, ..., y_{11}]$. This means, each conflict sample $\vec{X}$ corresponds to a set of resolutions $\vec{Y} = [y_0, y_1, y_2, ..., y_{11}]$ class labels (12 heading resolutions).

Table 4.1 shows an example of the different conflict samples and the corresponding multiple class labels. In this table, there are n conflict samples and each sample contains a pair of coordinates (A,B) for m times. Our simulated data contains multiple heading decision for a single conflict, therefore, Table 4.1 (b) shows the multiple class labels from y_1 to y_{11} where each label corresponds to a heading decision with a certain degree angle.

In the next section, we explain how NN can be used to solve aircraft conflicts.

Table 4.1: An example of the binary decisions for multi-label classification. The left side of the table shows all the conflicting samples where each sample contains the positioning coordinate of the involved aircraft. For example, (t_0,A,B) is a coordinate of aircraft A and aircraft B at time t_0. The right side of the table shows the multiple class labels from y_0 to y_{11} where each label corresponds to the same sample input. y_i equals 1 if the corresponding heading solves the conflict, 0 otherwise.

	$\vec{X}$			
$\vec{X}_0 \rightarrow$	(t_0,A,B),	(t_1,A,B),	.....,	(t_m,A,B)
$\vec{X}_1 \rightarrow$	(t_0,A,B),	(t_1,A,B),	.....,	(t_m,A,B)
:	:	:	.....,	:
:	:	:	.....,	:
$\vec{X}_n \rightarrow$	(t_0,A,B),	(t_1,A,B),	.....,	(t_m,A,B)

(a) Input samples

	$\vec{Y}$			
$\vec{Y}_0 \rightarrow$	y_0	y_1	.....	y_{11}
$\vec{Y}_1 \rightarrow$	y_0	y_1	.....	y_{11}
	:	:	.....	:
	:	:	.....	:
$\vec{Y}_n \rightarrow$	y_0	y_1	.....	y_{11}

(b) Output headings

4.2.6 Preliminary Neural Network Model

Since supervised algorithm training and testing require a pair of input-outputs for each sample data, we discuss here how our data is suitable for supervised learning. We explained in Chapter 3 that when a conflict occurs, the position, direction, speed, altitude, and many more parameters of each aircraft are stored by the ATCC[3]. The ATCC also stores the immediate order given by the ATCO to resolve the conflict. Since both the aircraft and the ATCOs immediate order data are stored in an ATCC, supervised learning algorithms can be used.

Figure 4.4 illustrates an example of how a neural network can be used to avoid collisions between two planes. This figure shows the current values of the four parameters of the two planes, latitude, longitude, speed, and direction, are shown as examples. Now, the discussion continues with the explanation of this figure. How our data can be fitted with a neural network will be discussed in detail.

Figure 4.4 shows when two planes are in conflict, their current positioning coordinates, speed, and direction are given to a neural network as input. On the other hand, different heading modification decisions are given as the output of the network.

We suggested the training will not be according to the current position but according to all the positions of the aircraft for the last 5 minutes, that we call last 5-minute window. The advantage of using a 5-minute window trajectory is the neural network model will be able to understand the environment of that conflict by changing the position of the involved aircraft without extracting any features.

There are some important factors in the output on which we determine the different classification types. As shown in Figure 4.4 there are m outputs and the types of classification depend on which or how many of them will be selected as real output. We can select the best possible output using a *softmax* activation function at the output layer to select the best probable heading action. It would be multi-class classification. There is a risk that, if for

[3] *Air Traffic Control Center*

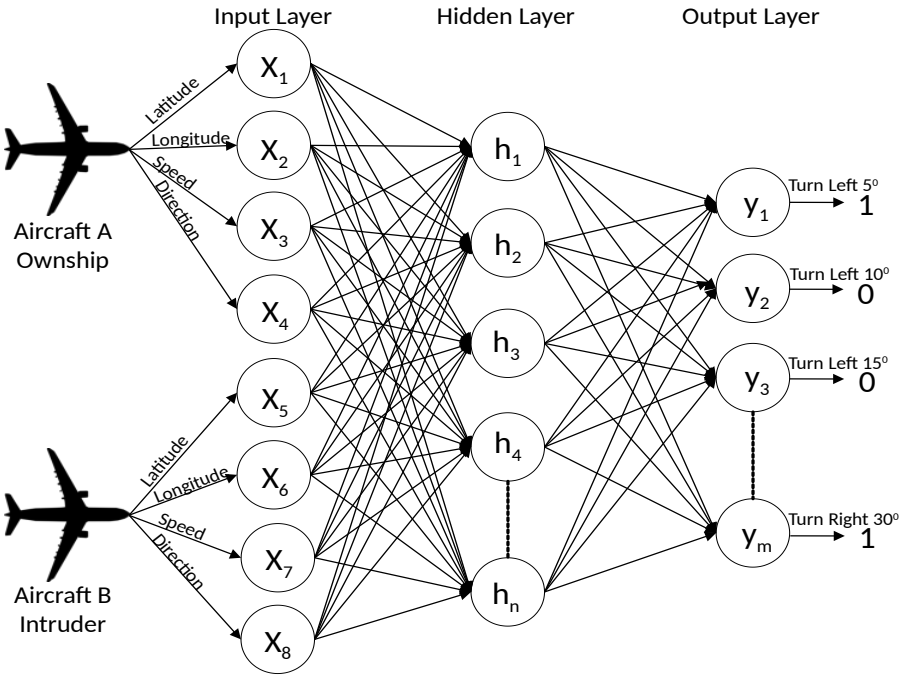


Figure 4.4: A supervised neural network to resolve conflicts between a pair of planes. This figure shows the different parameters of the two aircraft given to a neural network as input and the immediate order of ATCO as the output. There are eight nodes in the input layer; the input nodes are given as two planes' latitude, longitude, speed, and direction. There is a hidden layer with n nodes. The output layer consists of m nodes that encode the possible combinations (both aircraft can turn together to right/left) of heading between aircraft A and aircraft B.

some reasons the best solution fails, then the ATCO will not have alternative options. On the other hand, if we use the *sigmoid* activation function, we get the probability output of each heading action separately. In that case, we get one binary decision output for each heading action. Since one or more outputs are available for a conflict scenario, it is a multi-label classification. By adding various constraints to the output of multi-label classification, the output heading resolutions can be ranked (see Section 4.2). *Softmax* activation function and *sigmoid* activation function will be discussed later.

Our preliminary model was designed to find the best solution. We considered the conflict resolution problem as a multi-class classification problem using a *softmax* activation function. For instance, from Table 3.2, the resolutions for a single conflict illustrated in Figure 3.5 are 15 degrees, 20 degree, 25 degree, and 30 degree in *Turn Left aircraft A*. In this example, the *softmax* activation function chose the best one.

Figure 4.4 shows the current positions of the two aircraft. Figure 4.5 shows how to train with their last 5-minute positioning coordinates.

There are 9 features in an input trajectory, 4 (latitude, longitude, altitude, heading direction) for the two trajectories plus timestamp (see Table 3.4). Since we stored the two airplanes location every 5 seconds, for 5 minutes we have 60 values; that makes 540 (9*60) input fea-

tures. Therefore, the input layer of our model contains 540 nodes. The number of hidden layers and their number of nodes are chosen using different parameter search algorithms.

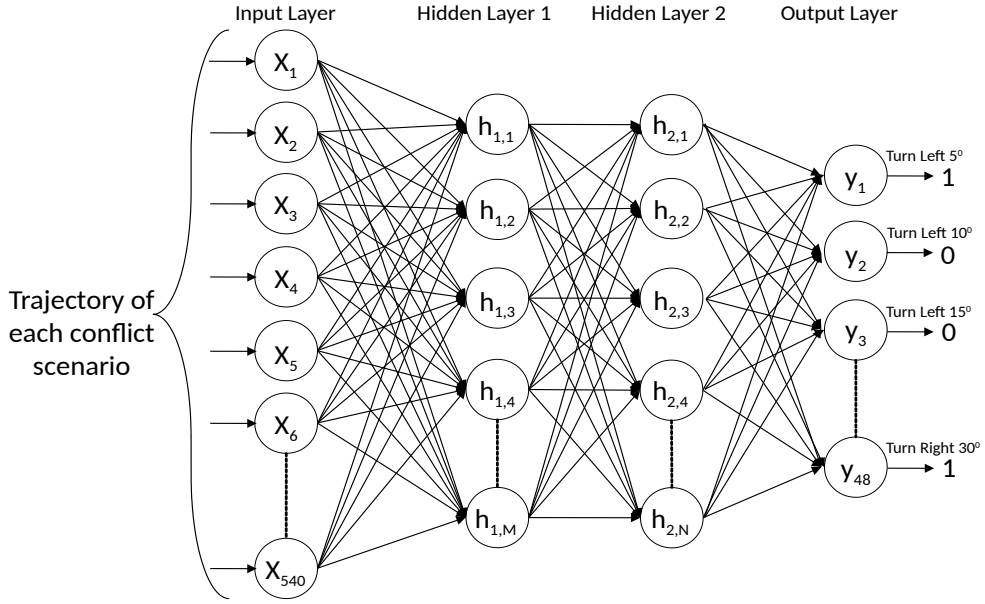


Figure 4.5: Preliminary model based on a neural network to classify aircraft heading decisions. There are 9 features in an input trajectory, 4 (latitude, longitude, altitude, heading direction) for the two trajectories plus timestamp. We stored the two airplanes location every 5 seconds, for 5 minutes we thus have 60 values; that makes 540 (9*60) input features. The output layer contains 48 nodes as the number of possible actions. Hidden layers are in between.

The output layer contains 48 nodes as the number of possible actions, the heading changes of any aircraft associated with the conflict. Figure 4.6 shows all the heading decisions for two aircraft. Each aircraft can make 12 heading decisions, which means 24 decisions in total. Again, if both aircraft take the left or the right turn or opposite direction turn together, then 24 more decisions are made. This is a total of 48 decisions. For example, in Figure 4.6, if aircraft A and aircraft B rotate -30 degrees together, it is possible to resolve the conflict. This type of the decision, although rare, can be taken in case of emergency. For each action (node) the neural network (Figure 4.5) will provide the best probable action as an output that is selected by the *softmax* activation function.

We have discussed how to fit a conflict situation into a neural network. In this preliminary model that we did not implemented, either the direction of one or of the two aircraft are changed for conflict resolution. We will see that one aircraft heading will remain unchanged in the implemented models. The output heading decision (class label) will be thus reduced to twelve. This is closer to the real world where conflicts are resolved by simply changing the heading of one aircraft.

Our initial model was not appropriate for more than two aircraft because both the number of inputs and output would increase because the possible heading resolution of each aircraft would then be considered. In real life, there are two types of aircraft involved in a conflict

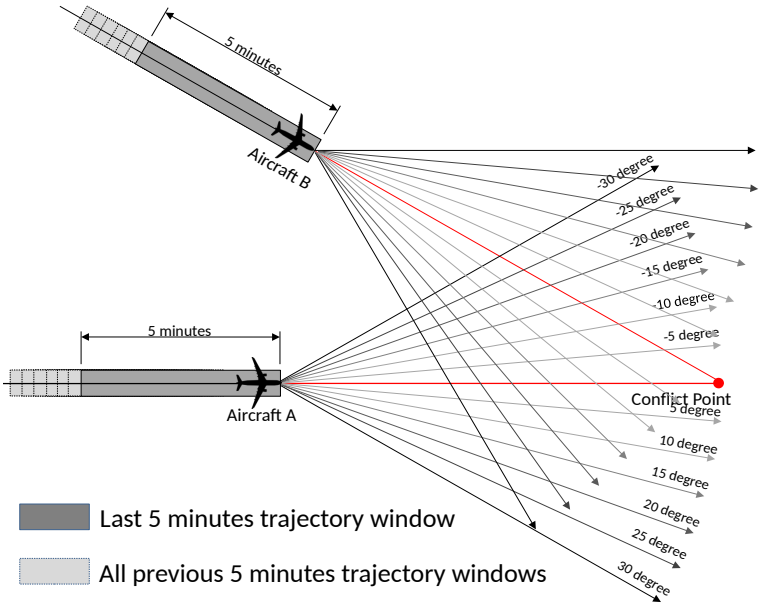


Figure 4.6: A conflict can be resolved by any combination of heading decisions. This is an example of a conflict scenario with 5-minutes of trajectory for the involved aircraft with 12 heading resolution decision of each individual. Individual aircraft can turn left or right. Also, they can turn together right or left or even can turn in opposite directions.

[Carbone 2006] (see Figure 4.8) (a) ownship and (b) intruder. Conflicts are usually resolved by taking the decision for the ownership. Because the intruder is the plane that has already changed its direction and it is the cause of the conflict. We did not implemented this initial idea.

Considering the real-life situation of how conflicts are actually resolved, we propose the CRMLnet model in the next section where we consider heading resolution of only one ownship as output. This model extends Kim et al.' related work [Kim 2016] (a) to provide more specific heading change(s) (b) possible binary solutions (c) to use 5 minute positioning coordinates for each aircraft. The binary decision means that the specific action may or may not resolve the conflict. Thus, one of the strong point of the CRMLnet model is it will suggest optional multiple actions.

4.3 CRMLnet: Conflict resolution multi-label neural network model

In this section, we propose a multi-label classification model, CRMLnet. The initial model aimed to find the best solution between the possible headings of the two aircraft using a *softmax* activation function. For the CRMLnet model we propose multiple alternatives heading resolution of one aircraft (ownship) using a *sigmoid* activation function. Trajectory parameters were used as input for both models but in the case of CRMLnet, the input dimensions

were halved which will be discussed later.

Our CRMLnet model is not a deep neural network since it has a single hidden layer.

Kim *et al.* applied the same idea but for the multi-class classification [Kim 2016]. The authors chose the best probable one as output. They also applied a neural network for multi-class (see Section 4.2.2) classification. In our case, we applied the same concept to the multi-label classification case using a single architecture based on a neural network. As we have discussed in Section 4.2.4, a multi-label classification can also be solved as multiple single-label classifications, we also implemented three other models using multiple single-label algorithms. Figure 4.7 shows the model based on a neural network and Figure 4.9 shows the model based on multiple single-label classifiers. We also compared CRMLnet model and the models based on multiple single-label classifiers.

He and Xia proposed a single neural network architecture with separate logistic functions at the output layer for multi-label classification of text emotion [He 2018]. The authors showed that a single network can perform better for multi-label classification than multiple individual networks. In a single network, all neurons are interconnected to each other, thus, all output decisions are based on sharing information with each other. On the other hand, Baker and Korhonen mentioned two disadvantages of using separate binary classifiers for multi-label classification : first, it is assumed that class-labels are independent, although this is not true in all cases; second, it is relatively expensive to compute because the classifiers are computing separately while using the same input [Baker 2017] .

We developed a model for conflict resolution using multi-label classification based on neural network that we call CRMLnet. Figure 4.7 shows our CRMLnet model. Figure 4.8 shows that in the CRMLnet model the conflict is resolved by changing the heading decision of one aircraft only, which is what happens in the real world. Compared to our preliminary model presented in Figure 4.5, the number of nodes in the output layer is reduced to 12.

The input parameter of our preliminary model is 540. Since there are not many conflict samples in the initial data, so, we split it into two parts each scenario in such a way that the time slot for the first part is at 0 second, 10 seconds, 20 seconds, up to 5 minutes. The other part is for 5 seconds, 15 seconds, 25 seconds, up to 5 minutes. Each part produces a new scenario (See Section 3.4.1 and Table 3.4). In the Table 3.4, we consider all the gray rows as one sample and the other as another sample. Since we store 5-minutes (5×60 seconds $= 300$ seconds) of trajectory following a 10-second change for each aircraft, we have the same parameters at each 10-seconds but the values change with respect to time. This means we store the features repeatedly for 30 (300 seconds $\div$ 10 seconds $= 30$) times with different values. The angle (α) between two planes remains unchanged. Thus, we have 9 input features that are repeated 30 times every 10-seconds: *time*, *latitude (aircraft A)*, *longitude (aircraft A)*, *altitude (aircraft A)*, *heading (aircraft A)*, *latitude (aircraft B)*, *longitude (aircraft B)*, *altitude (aircraft B)*, *heading (aircraft B)*. Overall, we have 271 (1 (angle) + 9×30 (repeated parameters) $= 271$) total input features. For that reason, the input layer of our neural network model is composed of 271 nodes.

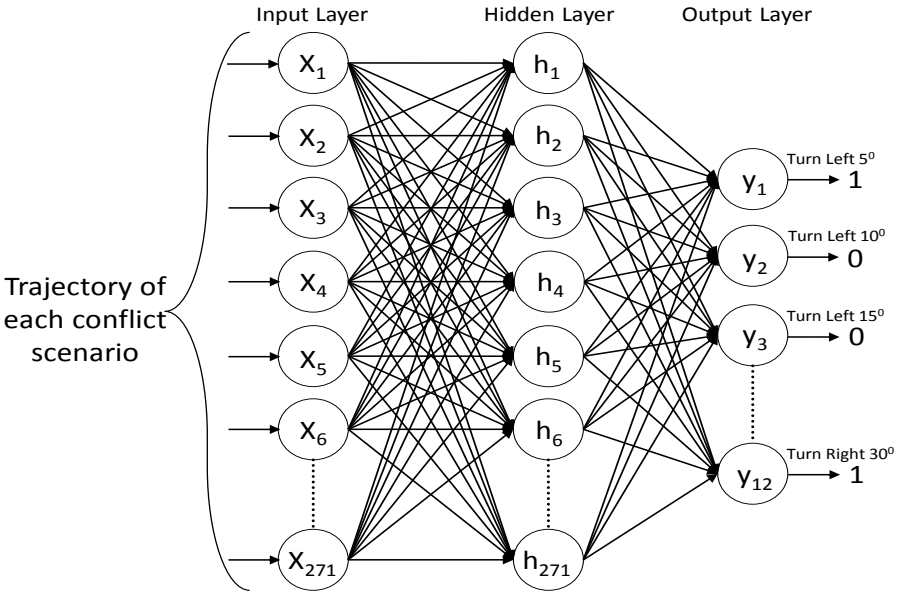


Figure 4.7: CRMLnet: conflict resolution multi-label neural network model. The input layer consists of 271 nodes for 5-minute trajectory parameters of a pair of aircraft. There is one hidden layer with the same number of nodes as the input layer. The output layer has 12 output nodes for immediate heading actions range from left 30^0 to right 30^0.

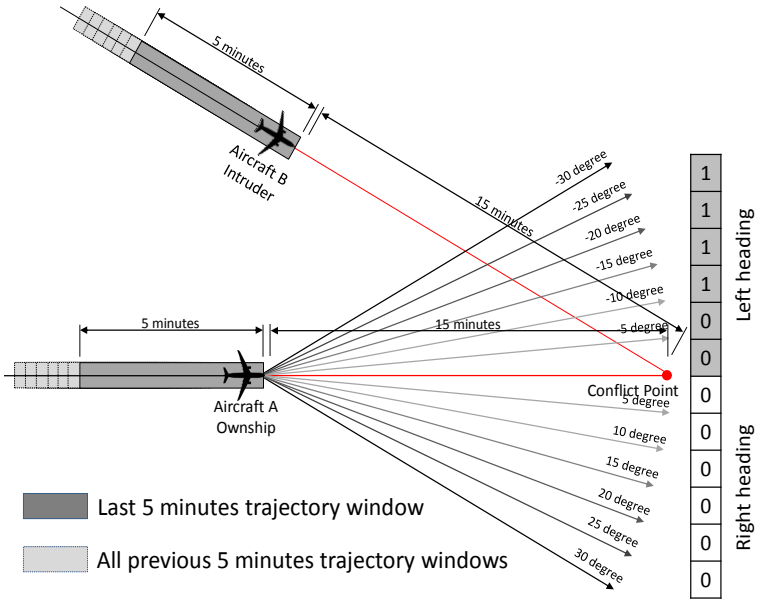


Figure 4.8: A conflict can be resolved by changing heading direction of one aircraft. Aircraft A can change its heading between left 30^0 and right 30^0 to solve the conflict while the heading of aircraft B remains unchanged. The column vector on the right shows the binary decision for this sample. Here "0" means the decision is not able to resolve the conflict whereas "1" means it can.

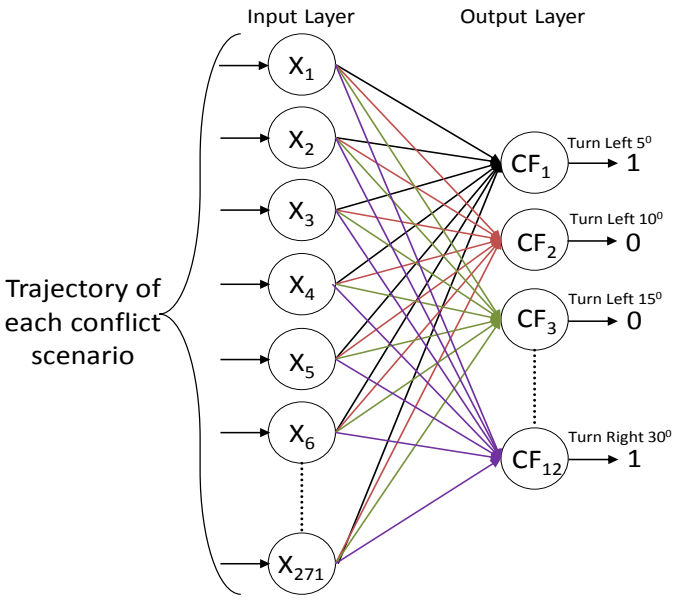


Figure 4.9: A multi-label classification architecture using multiple single-label classifiers. The input layer consists of 5-minute trajectory parameters that is the same number of inputs as in Figure 4.7. The output layer contains multiple but the same classifier. For example, CF_1, CF_2, ..., CF_{12} are replaced by any single-label binary classifier. All the classifications are independent of each other.

Additional hidden layers are needed , specifically when the problem or data is not linearly separable. For example, Yanling *et al.* showed that it is not possible to solve a logical XOR problem using a regular single layer neural network [Yanling 2002]. Therefore, the authors suggested to use the multiple layer perceptrons (nodes) to make the model non-linearity separability. Each layer of the neural network causes an additional transformation of the input and increases the non-linearity separability, therefore the model can fit itself better with the data. Our initial model was based on multiple hidden layers. After applying hyper-parameter search details (See Section 4.5.1) we limited the model to one hidden layer to avoid increasing the loss and decreasing the accuracy. Indeed, more hidden layers are more likely to increase overfitting than to increase learning ability because of the large number of neurons [Panchal 2011], specifically when the number of training examples is not huge.

The number of nodes in the hidden layer is equal to the number of nodes in the input layer.

We used 12 nodes for the output layer because we have 12 heading actions or class-label for one aircraft as shown in Table 3.2.

All the layers are dense that we used in this model. The difference between a linear layer and a dense layer is that an activation function ($y = f(W^T X + b)$) is used for the end result in the case of a dense layer. However, the general formula for both linear and dense layers is the same ($W^T X + b$).

Different activation functions can be used depending on the task. We used two types of activation functions in our model. We used a ReLU[4] activation function at the hidden layer. At the output layer, we used a *sigmoid* activation function. ReLU is used to avoid negative values by taking the maximum one between the input value of the neuron and a zero. ReLU uses the following formula: $\phi(x) = max(0, x)$ where x is the input value of the neuron. However, according to the Figure 4.10, if for any reason an input becomes negative, it becomes positive through this activation function.

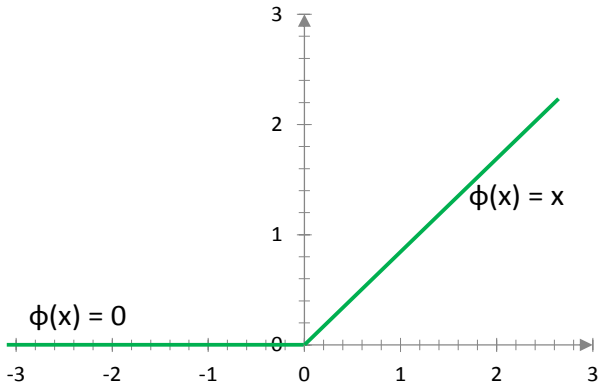


Figure 4.10: The ReLU activation function avoids making the value negative. The green line along the x-axis in this graph shows that when a value goes below zero it is set to zero using the function $\phi(x) = max(0, x)$.

Since our goal is to use multi-label classification of all the headings in Table 3.2, our model should provide the probability of each heading action individually. To meet this need, we used a *sigmoid* activation function for each output neuron. The functionality of a *sigmoid* activation function is to provide the probability score between '0' and '1' using the following mathematical formula: $f(x) = \frac{1}{1+e^{-x}}$. Figure 4.11 shows how a *sigmoid* activation function works. If any value of $f(x)$ is below 0.5, then the output of the *sigmoid* function will be 0, otherwise the output will be 1. Finally, the output of all nodes goes through the individual *sigmoid* function. The output prediction of each heading decision is either '0' or '1'.

We also implemented the three models we have discussed in Section 4.2.4 (SVM, KNC, and LR).

Our dataset contains trajectory data and there are different types of values inside this data and they come in different ranges. As a result, the impact of the larger values is greater than the impact of the smaller values. Therefore, to get rid of this problem, before applying any of the machine learning models, we also perform a standard scaling method on the data. To scale a peculiar feature, for instance, i-th feature x_i which ranges from low_i to up_i, the following equation is used:

$$x_i := low_i + (up_i - low_i)\frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{4.7}$$

[4] *Rectified Linear Unit*

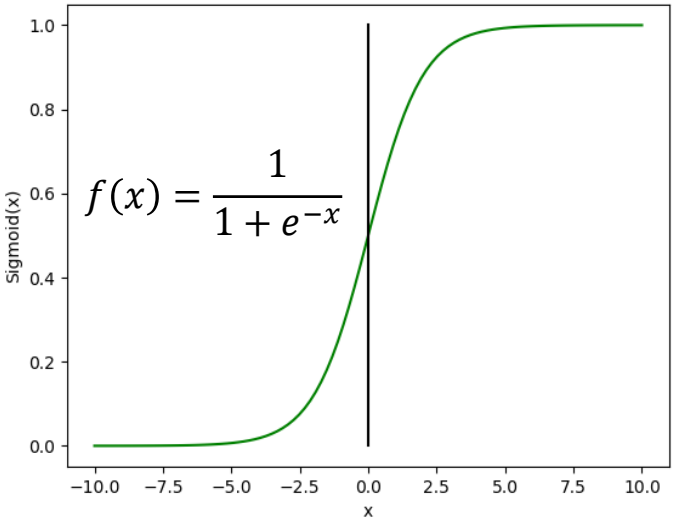


Figure 4.11: A *sigmoid* function makes the incoming value either '0' or '1'.

Commonly used ranges are [-1, +1] and [0, 1]. In our case, we used the first one: [-1, +1] because our data has both positive and negative values. Figure 4.12 shows an overview of the training and testing procedure of the CRMLnet model. The following section discusses it further.

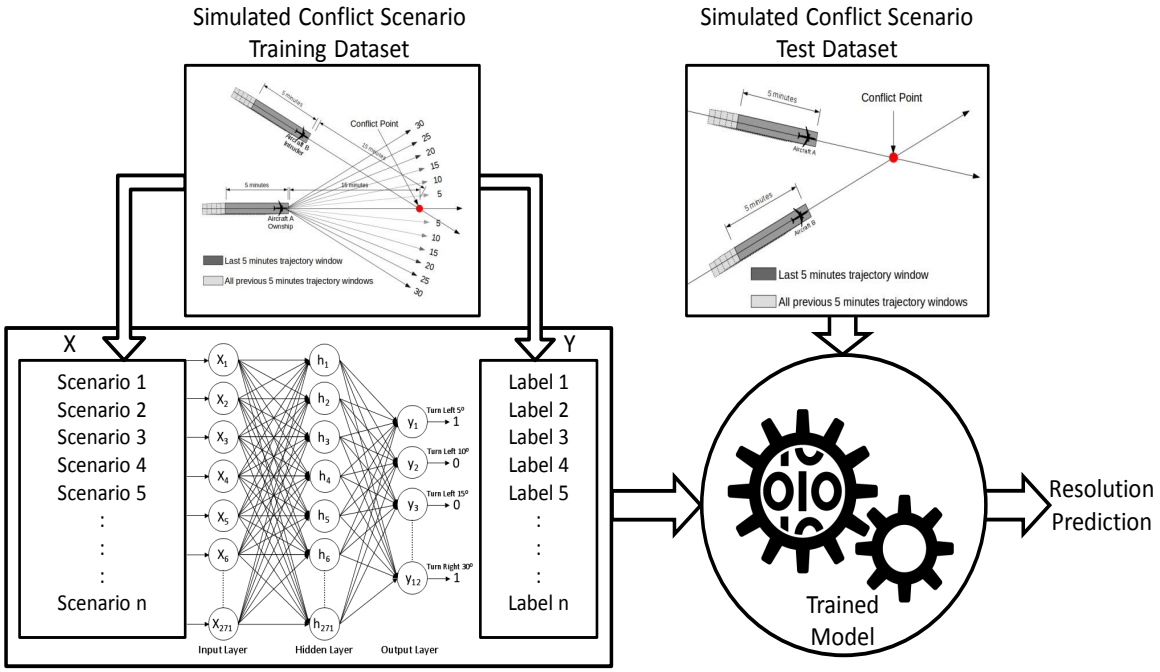


Figure 4.12: An overview of the CRMLnet model and the training/testing procedure we used. Here, the model is trained with conflict scenarios and output decisions. Once the training is over, it is tested with the unseen conflict sample. We evaluate the same model in two different ways (cross-validation and independent test).

4.4 Evaluation

Our CRMLnet model and the models based on SVM, KNC, and LR were evaluated on the simulated dataset presented in Section 3.4.1.

Comparing different algorithms requires performance measurement and sampling methods.

We use several metrics to evaluate performance namely *Acc*[5], *auROC*[6] curve , *auPR*[7] curve, F_1 score, ${S_n}^8$, ${S_p}^9$, and *MCC*[10]. These measures are also used to compare the different models. These metrics are defined in the following equations:

$$\text{Accuracy } (Acc) = \frac{TN + TP}{TP + FN + TN + FP} \tag{4.8}$$

Here, TP is the total number of correctly classified positive examples, TN is the total number of correctly classified negative examples, FP is the total number of incorrectly classified positive examples, and FN is the total number of incorrectly classified negative examples. The range of accuracy is in between 0% to 100%.

Area under the receiver operating characteristic curve *auROC* and area under the precision recall curve *auPR* are crucial measures. They express the strength of the underlying classification regardless of the selected threshold.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4.9}$$

$$\text{where } precision = \frac{TP}{TP + FP} \text{ and } recall = \frac{TP}{TP + FN} \tag{4.10}$$

One of the most used performance measures for machine learning models is the F1 score. F_1-score is the harmonic mean of *precision* and *recall* because the right part of the Equation 4.9 shows that F_1 considers FP and FN equally.

Also, Equation 4.10 shows that the only difference between *precision* and *recall* is a portion of the denominator which is FP and FN. Here, *precision* is the number of true positive (TP) over the number of true positive (TP) plus the number of false positive (FP). *Recall* is also known as Sensitivity (S_n).

[5] *Accuracy*
[6] *area under Receiver Operating Characteristic*
[7] *area under Precision Recall*
[8] *Sensitivity*
[9] *Specificity*
[10] *Mathew's Correlation Coefficient*

$$\text{Sensitivity } (S_n) = \frac{TP}{TP + FN} \tag{4.11}$$

$$\text{Specificity } (S_p) = \frac{TN}{TN + FP} \tag{4.12}$$

Similarly, S_p is the true negative rate or number of correctly classified negative instances over the total number of negative examples.

Matthews correlation coefficient (MCC), Equation 4.13, is one of the very important and more complex performance measurements for binary classification. The scoring range of MCC is between -1 and 1. MCC output is maximum if FP = FN = 0, TP $\neq$ 0 and TN $\neq$ 0. That would occur when there is no incorrectly classified sample. MCC drops to -1 if TP = TN = 0, FP $\neq$ 0, and FN $\neq$ 0, which occurs when the model does not correctly classify a single sample.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.13}$$

There are several types of sampling methods to test and validate a classification algorithm: k-fold cross-validation [Kohavi 1995], independent test sets, jackknife tests, etc.

We used k-fold cross-validation and independent test sets. With k-fold cross-validation method the model can train itself even if the volume of the data is low, avoiding overfitting. In training sessions, k-fold cross-validation separates the dataset in k small partitions (k-fold). The classifier then continues training and testing for k times where k is defined after several tests. Each time, it takes (k-1) subsets for training and the remaining subset for testing. It proceeds likewise for each testing subset. All performance measures are taken for each fold and the end result is their average.

Another sampling method, the independent test set, was also used where the dataset is divided into three subsets. For example, 60% of the total data is for training, 20% for validation, and 20% for testing purposes. This sampling method is called independent test set because the test set is kept completely separate from the training set.

During the training, validation and test, each pair of dataset contains a trajectory vector (Table 3.4) and the corresponding multi-label heading resolution decisions (Table 3.2) vector $(\vec{X}_0, \vec{Y}_0)$, $(\vec{X}_1, \vec{Y}_1)$, $(\vec{X}_2, \vec{Y}_2)$, ..., $(\vec{X}_n, \vec{Y}_n)$ that already discussed in Section 4.2.5.

4.5 Result and discussion

This section discusses all the experimental results and analysis. We used python programming language, python Keras library, and python sci-kit learn library to implement our model.

4.5.1 Hyper-parameters search algorithm

The performance of a machine learning model highly depends on the selection of its hyper-parameters. On the other hand, selecting hyper-parameters is also quite difficult because of calculating the permutations of the parameters. There are algorithms to find the best possible parameter values. Random Search [Bergstra 2012] is one of the popular and widely used algorithms to find the most influential parameters. Grid Search [LaValle 2004] is an another commonly used algorithm to find hyper-parameters but this algorithm searches all combinations of hyper-parameter subsets. According to [Bergstra 2012], in the case of high dimensional models such as neural networks, it is very time-consuming to apply grid search to find hyper-parameters. Thus, we applied it on our CRMLnet model to find the learning rate, number of hidden layers, number of nodes in each hidden layer, optimizer, etc. After the parameters selection and different experiments, we tuned our model with Adam[11] as an optimizer. The Adam optimizer is a combined version of two widely used optimizers: RMSprop[12] and SGD[13] with momentum [Ruder 2016, Bottou 2012].

4.5.2 Results

Before running any validation procedure, we shuffled all the data and split them into a train, validation, and test sets. We applied k-fold cross-validation where we used k=5, k=10, and k=20. We trained our model using these three values for k but we got the highest accuracy and the lowest loss for $k = 10$. We thus set k=10 for the cross-validation.

Figure 4.13 plots the train and validation loss for 10-fold cross-validation. Figure 4.15 (a) shows all these loss curves are overlapping, thus their results are close one to the other. This means the results are quite robust and do not depend on the data split under consideration.

We also plot the accuracy of the train and validation on Figure 4.14 while Figure 4.15 (b) shows all these accuracy curves overlapping together. In all figures, the red curves represent the loss and accuracy during training while the green curves represent the loss and accuracy during the validation. We applied our CRMLnet model on different numbers of epochs. Here, 1 epoch means the complete forward and backward pass of input features during training. After many experiments, we set our model to 100 epochs. The curves on Figure 4.13, Figure 4.15 (a), Figure 4.14, and Figure 4.15 (b) are for 10-fold cross-validation with 100 epochs. Based on the plotted results, we see that the loss and accuracy are almost the same during training and validation.

So, according to Figure 4.13, Figure 4.15 (a), Figure 4.14, and Figure 4.15 (b), up to 100 epochs our model does not overfit; no variance problem occurs either. The average validation (test loss in this case) loss in all cases is around 0.05, which is low. The lower loss a model has, the better its performance.

[11] *Adaptive Moment Estimation*
[12] *Root Mean Square Propagation*
[13] *Stochastic Gradient Descent*

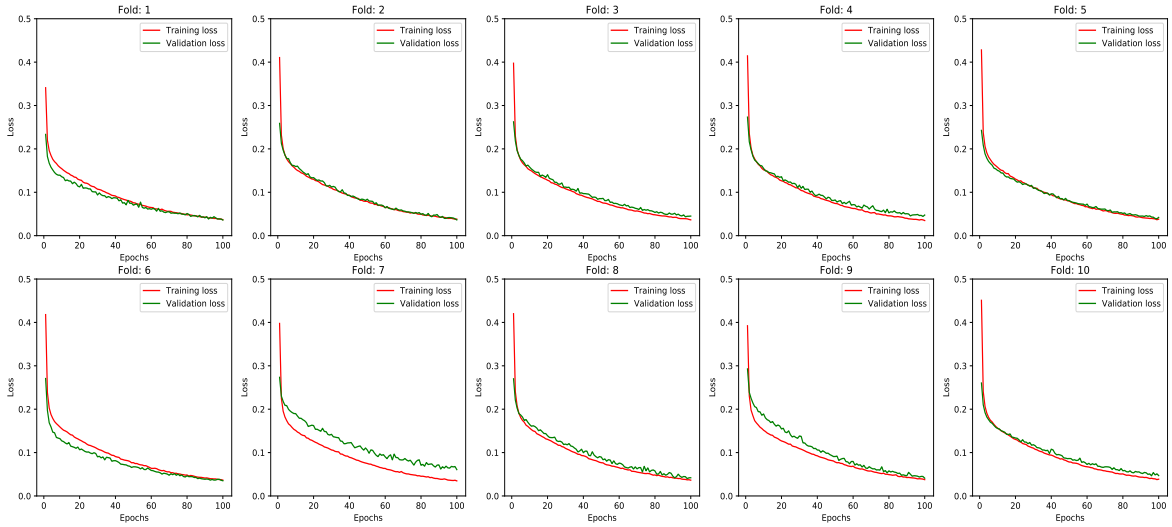


Figure 4.13: Up to 100 epochs, CRMLnet performs well and does not overfit when considering cross-validation. All the training and validation losses are almost similar and decreased to 0.5 (very low). Here, the horizontal axis represents the number of epoch and the vertical axis is the loss.

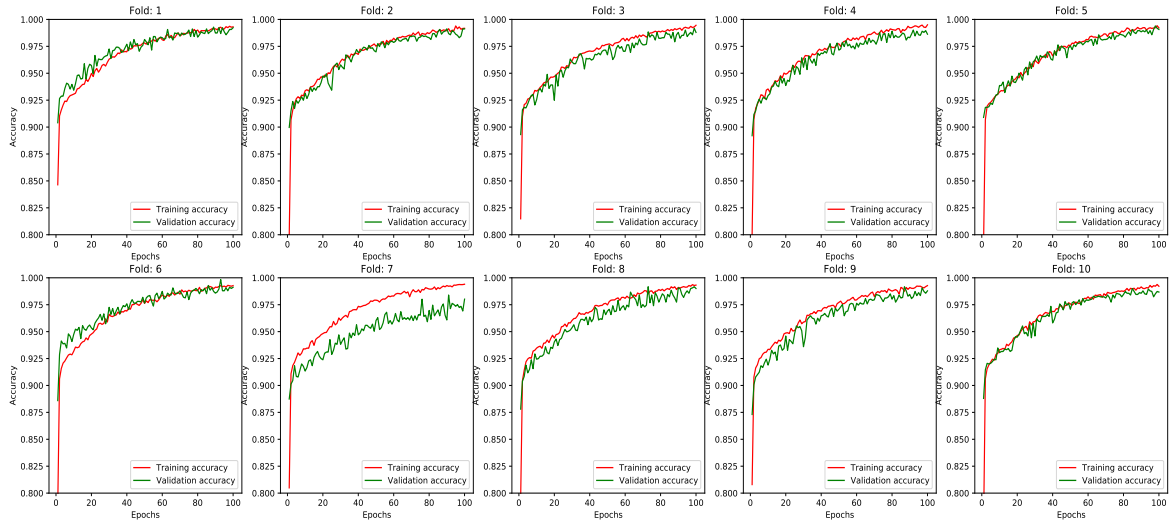


Figure 4.14: Up to 100 epochs, CRMLnet performs well and does not overtfit when considering cross-validation. The training and validation accuracy curves are mostly smoothly overlapped in each plot with a score of around 98% which is a very high score although there are a few curves with some fluctuations. Here, the horizontal axis represents the number of epochs and the vertical axis is the accuracy.

We also used the independent test set. The total dataset is divided into three subsets: training set is 60% of the total data, test set is 20% and validation set is 20%. Figure 4.16 (a) shows the training and validation loss while the CRMLnet model is applied with independent test set data sampling. Figure 4.16 (b) shows the accuracy of the using the independent test set data sampling method.

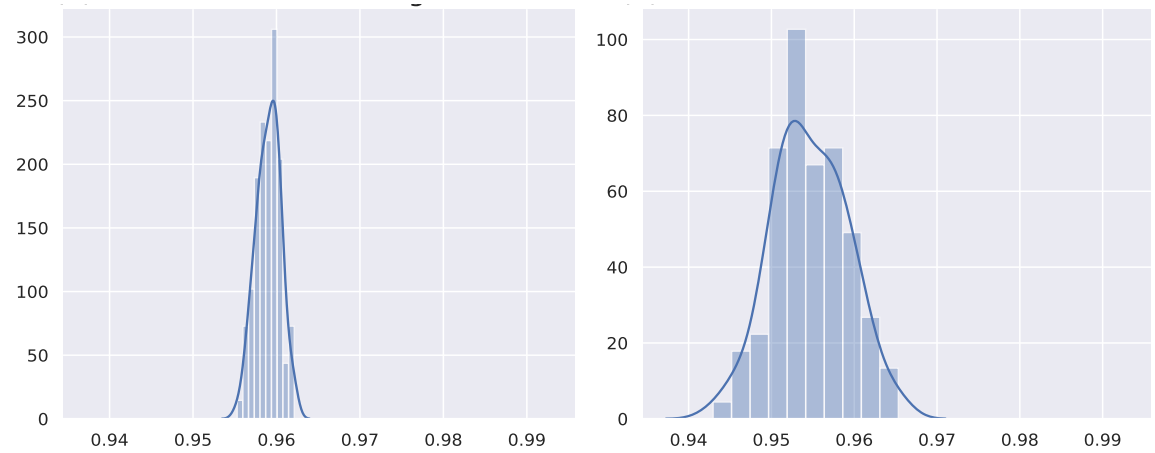
(a) Loss (b) Accuracy

Figure 4.15: All the losses and accuracies of CRMLnet are plotted together up to 100 epochs. (a) shows that all the losses are almost overlapping. Although there is some fluctuation in accuracy (b), overall CRMLnet performs well.

(a) Loss (b) Accuracy

Figure 4.16: The accuracy and loss of CRMLnet are also well up to 100 epochs applying independent test set. The other comments made for Figure 4.15 also hold here.

We run the CRMLnet model 100 times with independent test sets by shuffling the datasets each time and reported all the accuracy values distribution. Figure 4.17 (a) shows the distribution of training accuracy values while (b) shows the distribution of validation accuracy. There is not much difference between these two distributions, which means that our data preparation and the model used have neither overfitting problems nor variance problems even while using independent test set data sampling.



(a) Distribution of training accuracy (b) Distribution of validation accuracy

Figure 4.17: After running the independent test set 100 times the distribution of accuracy shows CRMLnet performs well. Here, the horizontal axis of both (a) & (b) shows the number of samples while the vertical axis of both (a) & (b) shows the accuracy. Although, (a) the distribution of training accuracy looks a little bit better than (b) the distribution of validation accuracy, still, there is not much difference between them.

Accuracy is a common performance measure. Accuracy on our CRMLnet model is around 98.72% for 10-fold cross-validation (designated as CRMLnet$_{cv}$[14]). It is around 97.79% independent test set (designated as CRMLnet$_{ind}$[15]). This means the performance in both cases is generalized while 10-fold cross validation shows better accuracy than independent test set. Subsequent paragraphs show some more measurements for our CRMLnet as well as for the other architectures with multiple single-label classifiers.

Table 4.2: CRMLnet is much better than the other classifiers when using cross-validation (CRMLnet$_{cv}$). Here, the 1_{st} column is the classifier. The next columns are : Accuracy (*Acc*), area under receiver operating characteristic curve (*auROC*), area under precision-recall curve (*auPR*), Specificity(S$_p$), Sensitivity (S$_n$), Mathew's Correlation Coefficient (*MCC*), and F$_1$-score.

Classifiers	*Acc*	*auROC*	*auPR*	S_p	S_n	*MCC*	F_1
CRMLnet$_{cv}$	**98.72%**	**0.999**	**0.998**	**99.11%**	**97.94%**	**0.971**	**0.981**
MSVM$_{cv}$	91.66%	0.953	0.934	94.24%	86.54%	0.812	0.793
MKNC$_{cv}$	95.45%	0.979	0.958	96.68%	93.01%	0.898	0.921
MLR$_{cv}$	90.96%	0.863	0.818	93.29%	86.36%	0.797	0.785

In addition to accuracy, we have made some other measurements: *auROC*, *auPR*, S_p, S_n, *MCC*, and F_1 score (See Section 4.4). We perform all these measures on CRMLnet using both 10-fold cross-validation and independent test set (See Table 4.2 and Table 4.3 first line).

We also applied other popular machine learning classifiers that we discussed in Sec-

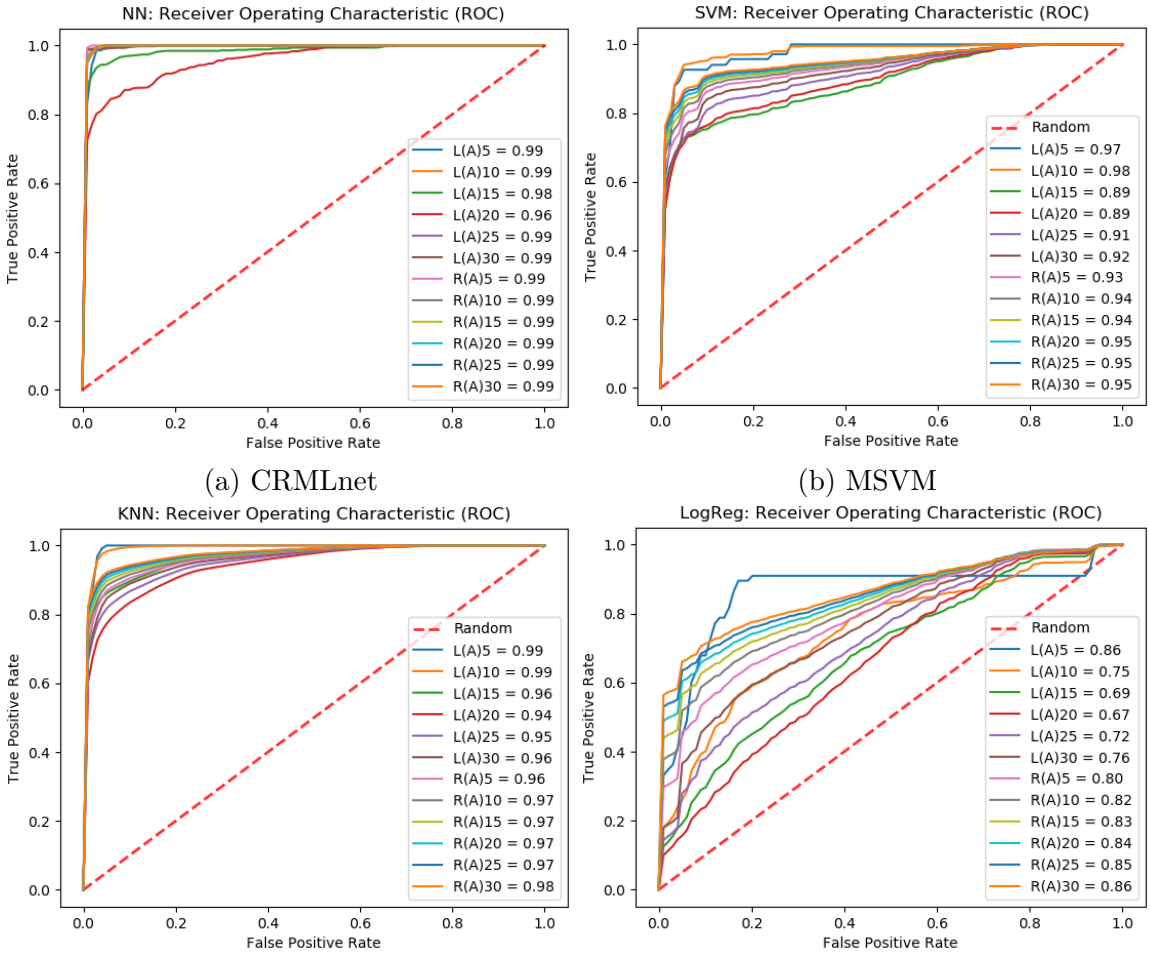
[14] *Conflict Resolution Multi-label Neural Network Cross Validation*
[15] *Conflict Resolution Multi-label Neural Network Independent Test*

Table 4.3: CRMLnet is also much better than the other classifiers when using independent test set (CRMLnet$_{ind}$). The columns are the same as in Table 4.2

Classifiers	*Acc*	*auROC*	*auPR*	S_p	S_n	*MCC*	F_1
CRMLnet$_{ind}$	**97.79%**	**0.997**	**0.995**	**97.93%**	**97.36%**	**0.952**	**0.968**
MSVM$_{ind}$	91.47%	0.944	0.899	94.30%	85.89%	0.808	0.768
MKNC$_{ind}$	93.00%	0.931	0.895	95.14%	88.78%	0.843	0.884
MLR$_{ind}$	90.97%	0.842	0.789	93.63%	85.73%	0.797	0.785

tion 4.2.4 and Figure 4.9. We compared their performance with the CRMLnet model and show our model is much better in performance than the other machine learning classifiers. As we explained in Section 4.2.4, a multi-label classification problem can be defined as a multiple single-label classification problem. Therefore, we have applied multiple separate single-label classifiers using the same classifier for each single class-label. For example, in Table 3.2 or Figure 4.8, we have twelve distinct class-labels (horizontal heading direction) and for each class-label, we applied a single-label classifier to classify them individually. All individual classifiers perform a binary classification to predict the corresponding heading changes whether it solves the conflict or not. So, all the individual classifiers use the same input features. For example, Figure 4.9 shows a general architectural view of a multi-label classification model using a single-label classifier where all the $CF(CF_1, CF_2, ..., CF_{12})$ can be replaced by any of one single-label classifier (SVM, KNC, or LR). Thus, we designed three different architectures for SVM, KNC, and LR (designated as MSVM, MKNC, and MLR) and applied them on the same dataset using both sampling methods: 10-fold cross-validation and independent test set.

Finally, all the results discussed in Section 4.4 of the different models are represented in the Table 4.2 for cross-validation sequentially as follows: CRMLnet$_{cv}$, MSVM$_{cv}$, MKNC$_{cv}$, MLR$_{cv}$. Table 4.3 also presents the results using independent test set validation. Table 4.2 and Table 4.3 reveal that all the models did better for cross-validation than the independent test set. A closer look at the results show that our CRMLnet model for both cross-validation and independent test set are much better than the other models based on a single-label classifier. CRMLnet model did well not only for accuracy but also for all other scoring discussed in Section 4.4. Notable among these scores are F_1 and MCC, which have recently been used to compare almost all machine learning models. Although numerical results are often important, many complex things are easier to understand if they are visually presented. Therefore, in Figure 4.18, we represent the ROC curve of individual class-label (twelve heading directions from Table 3.2) for all the methods with 10-fold cross-validation: (a) Neural Network-based model CRMLnet, (b) Multiple Support Vector Machine based model MSVM, (c) Multiple K-Nearest Neighbor Classifier based model MKNC, and (d) Multiple Logistic Regression based model MLR. If we look at the ROC curve in Figure 4.18, we can see that the model based on a neural network CRMLnet$_{cv}$ is performing better than the other models for all class labels. We can see ROC highly fluctuate on the other models (MSVM$_{cv}$, MKNC$_{cv}$, and MLR$_{cv}$) while it does not for our CRMLnet$_{cv}$ model.



(a) CRMLnet (b) MSVM

(c) MKNC (d) MLR

Figure 4.18: Overall CRMLnet is much better considering ROC performance than the other models. Here, each figure shows the ROC of individual heading decision in Table 3.2: (a) Neural Network-based model CRMLnet performance; (b) Multiple Support Vector Machine based model MSVM; (c) Multiple K-Nearest Neighbor Classifier based model MKNC, and (d) Multiple Logistic Regression based model MLR.

4.6 Conclusion

Almost all the conflict resolution related work has been done based on a single position of the aircraft involved in the conflict while in this chapter we defined and evaluated a model based on a series of 5-minute continuous positions of each involved aircraft. Thanks to this data representation, the model can be trained considering the conflict environment with the 5-minute of trajectory. This solution also preserves us to calculate additional features that are prone to errors.

We cast the problem into a multi-label classification problem where several solutions may be possible.

In this chapter, we developed a neural network model (CRMLnet) that we evaluate against more traditional classifiers. We show that when two aircraft are in conflict, our model performs better than the other classifiers.

Our preliminary model presented in Section 4.2.6 was published at ADBIS[16], TPDL[17] & EDA[18] joint conferences 2020 [Rahman 2020].

The CRMLnet model was published in 2022 at the 14th International Conference on Agents and Artificial Intelligence [Rahman 2022].

The CRMLnet is implemented using the Python programming language and TensorFlow's Keras library. Complete code attached to Appendix A.

Although our CRMLnet model is valuable, this model cannot handle conflicts where a variable number of aircraft can be involved, which is the real word case. Indeed, each aircraft trajectory corresponds to a number of input and the input layer of our model is set for two aircraft only.

This difficult problem is solved in the next chapter.

[16] 24th European Conference on Advances in Databases and Information Systems
[17] 24th International Conference on Theory and Practice of Digital Libraries
[18] 16th EDA days on Business Intelligence & Big Data

Chapter 5

Aircraft Conflict Resolution using Convolutional Neural Network on Trajectory Images

Abstract.

Resolving aircraft conflicts using neural network models on trajectory data is not possible for conflicts that imply a variable number of aircraft because the input size of the model is fixed while the input data is not. To solve this challenge, we transformed the trajectory data into images which size does not depend on the number of planes. We developed a multi-label conflict resolution model that we named ACRnet. It is based on a convolutional neural network to classify the obtained images. ACRnet model achieves an accuracy of 99.16% on the training data and of 98.97% on the test data set for two aircraft. For both two and three aircraft, the accuracy is 99.05% (resp. 98.96%) on the training (resp. test) data set.

Contents

5.1 Introduction

In Chapter 4, we developed a model based on a series of 5-minute continuous positions for two aircraft. While this approach could be adapted for another number of aircraft, this number

has to be fixed because the model -the number of neurons on the input layer- depends on the number of aircraft. Indeed, because our first model is based on trajectory data related to positions, there is a number of input for each aircraft. The total number of input depends on the number of aircraft while the neural network model has a fix number of input. This situation is not representative of real world situations. Rather, usually, each time a conflict occurs, there may be a different number of aircraft involved.

As the input size of a NN model remains constant and cannot be changed in real-time, it is an input dimensionality problem. For example, if we need k parameters for an aircraft to be considered during a conflict resolution, then there will be $n \times k$ parameters for n aircraft.The number of inputs of the model depends on the number of aircraft.

Brittain and Wei [Brittain 2021] applied recurrent neural network (RNN[1])-based LSTM model. The computation of their model depends on the number of aircraft when using LSTM. Since the number of input aircraft can be arbitrary and the trajectory data of each aircraft is 5 minutes, in our case, it is very complicated to fit it in any recurrent-based model. Brittain and Wei only considered the current position of the aircraft while Zhao and Liu [Zhao 2021] converted trajectories into image data to handle a variable number of aircraft in real-time using reinforcement learning. Finding the perfect reward function when using reinforcement is challenging.

If this input dependency could be eliminated by any means, we could build a model so that the model does not depend on the number of aircraft. One of the challenges here is that if we solve this problem by changing the model, it will become model-dependent. On the other hand, if we represent the data in such a way that the data is applicable to the input of all models, then the acceptability of that solution will be much better than the changing of the model architecture. So in the case of a new representation of our data, we must keep in mind that the input dimension of the model should be independent of the number of aircraft.

Here, inheriting ideas from Zhao and Liu [Zhao 2021], we converted trajectory data into image data. We detail this process in Section 3.4.1.

The benefits of converting trajectory data into images are given below:

(a) The trajectory of a variable number of planes can be plotted without resizing the image. Thus, it is possible to apply the same machine learning model without any change in the input size of the model. It solves the input dimensionality problem for the model;

(b) The complexity of the computation will not change even if the number of planes changes, which is not the case with non-image data;

(c) We can easily apply the image data augmentation technique to increase the training sample and convolutional neural network (CNN) can be applied with data augmentation;

(d) Not only can conflicts be resolved between aircraft but also between aircraft and other airspaces such as weather, military zones, etc.

[1] *Recurrent Neural Network*

Like in the previous chapter, to reflect the fact a conflict can be solved in different ways, we annotated each image with multi-labels, each corresponds to a possible solution.

We developed and evaluate a CNN with multi-label classification that we call ACRnet: aircraft conflict resolution convolutional neural network.

This chapter is organized as follows. Section 5.2 discusses the model architectures. Section 5.3 presents the evaluation framework. Section 5.4 presents the results and comparisons between different models. Finally, Section 5.5 concludes this chapter.

5.2 ACRnet: Aircraft conflict resolution CNN model

We converted trajectory into image data where the initial trajectory is the one we presented in Section 3.4.1 in Chapter 3.

Like in the previous chapter and model, there are 12 class labels (Figure 4.6: $-30^0, -25^0, ...-5^0$ and $+5^0, +10^0, ... + 30^0$) as output for each input conflict situation. Each conflict sample is annotated by one or more class labels and thus a multi-label classification-based model is the most appropriate.

5.2.1 Model selection

Since our image data are labeled with multi-label classes, we decided to use a model based on a convolutional neural network where the outputs are multi-labeled. There are many existing models based on convolutional neural networks for image classification. We applied two widely used models: VGG16 [Simonyan 2014] and ResNet [He 2016]. Although these models are very popular for image classification, our data (1,656 images) may not be sufficient for these models due to a large number of layers and the number of nodes in these architectures. Thus, since the performance of a neural network-based model depends on its hyper-parameters, we created a model by applying different combinations of the hyper-parameters.

The hyper-parameters of a neural network are: the number of hidden layers, the number of nodes in each layer, the activation function, etc. Optimizing hyper-parameters is costly. Grid search [LaValle 2004, Bergstra 2012] and random search [Bergstra 2012] algorithms can help to find the best hyper-parameter values. The Keras team has recently developed KerasTuner [O'Malley 2019] for deep learning or convolutional neural network for hyper-parameter selection which is widely used. We used KerasTuner for our CNN-based model to find the hyper-parameters. KerasTuner comes with the combination of Bayesian optimization [Frazier 2018] and random search [Bergstra 2012]. After applying KerasTuner, we found our ACRnet model architecture that consists of five layers: input, output, and three hidden layers. In addition to these layers, some other layers are used such as dropout layer, dense layer, max pooling, etc. which are discussed later.

We only converted the input trajectory to the image data and made no changes to the output class label (See Section 3.4.2). Thus, the output layer consists of 12 nodes. Finally, we consider a small architecture to propose a model for our dataset, we call it *aircraft conflict resolution convolutional neural network* (ACRnet) model (see Figure 5.1). It might not have been appropriate to use VGG16 or ResNet model due to the large number of nodes and layers. The model we chose, based on our hyper-parameters search, is much smaller compared to these architectures.

5.2.2 ACRnet model based on images

We defined the ACRnet model after applying model selection. Because our data is not very large, using more conventional CNN models with many layers and nodes (neurons) increases the risk of overfitting. We thus designed our model with a lower number of layers and nodes. Figure 5.1 shows the architecture of the model.

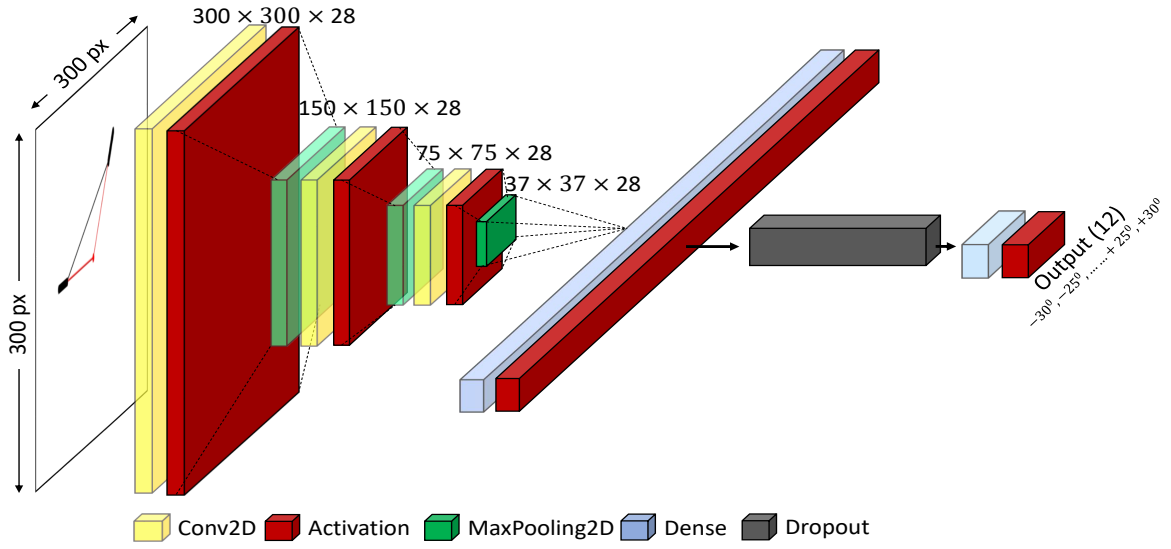


Figure 5.1: ACRnet: Aircraft conflict resolution CNN model. The size of the first convolutional layer (Conv2D) is 300 × 300 with 28 nodes (filters) as the image size is $300px \times 300px$. This model contains 3 hidden layers and each hidden layer (Conv2D) of this model has 28 nodes (filters). The activation function is ReLU except for the output layer which uses sigmoid. Finally, there are 12 nodes in the output layer.

Figure 5.1 shows our ACRnet model. This model has three hidden convolutional layers (Conv2D: two dimensional convolutional layers) and 28 nodes (filters) in each layer. The size of each image specified for this model is 300 × 300 and the number of nodes or filters at the input layer is 28. We found the best scores for 28 filters when searching for hyperparameters, so we kept this constant for all the layers.

Just after each convolutional layer there is an activation layer. We chose the widely used Rectified Linear Units (ReLU) [Agarap 2018] activation function, for all the hidden layers.

In Figure 5.1, all activation layers are represented by red blocks. Usually, the input image with high dimension is reduced through the use of *MaxPooling* layers. In this model, several *MaxPooling* layers of size 2 × 2 are used to reduce the dimension of each Conv2D layer by half. *MaxPooling* is a 2D grid that travels on Conv2D to create a new Conv2D grid. Since the size of the *MaxPooling* grid is 2 × 2, the size of the new Conv2D is half that of the previous Conv2D. Because *MaxPooling* takes only the maximum value from 2 × 2 grid to build the new grid. But, there will be no change in the size of the filter. For instance, the Conv2D size in the first layer is 300 × 300 and the number of filters is 28. So, the total parameters are 300 × 300 × 28. The Conv2D size is reduced from 300 × 300 to 150 × 150 in the second layer because of the *MaxPooling*.

After the final convolutional layer, there is a *Dense* layer. It is also called a fully connected layer because it is flat and densely connected to the previous and/or next layer. It converts the final Conv2D into a 1D vector. Generally, the function of this layer is to decide on the next final output using the important features reduced by the convolutional layers. We also use a ReLU activation function in this layer. Right after this dense activation layer (ReLU), there is a *Dropout* layer. Typically, *Dropout* layer is used to exclude less important information. In this case, it is necessary to specify how much information will be reduced. We set it to 50%. Thus, this layer reduces information from *Dense* layers by 50%. In this way, the *Dropout* layer is used to forward the important information in making the final decision. Finally, there are 12 nodes in the output layer that provide binary decisions for 12 heading directions (-30^0, -25^0, ..., $+25^0$, $+30^0$) described in Figure 3.2 using the sigmoid activation function. Since all the outputs are separate binary classes (0 or 1), we use an activation layer with sigmoid activation function just after the output layer. All the outputs are either '0' or '1'. Whenever a new test conflict image is given after model training, the model gives the binary output of 12 headings. Training and testing are discussed in detail in the next section.

5.3 Evaluation framework

To evaluate the performance of the ACRnet model, we reused the performance measurements discussed in Section 4.4: (*Acc*, *auROC* curve , *auPR* curve, F_1 score, Sensitivity S_n , Specificity S_p , and *MCC*). We add two new measures: False positive rate (FPR[2]) and False negative rate (FNR[3]).

Since all the heading decision individual class levels are binary classes, the performances are evaluated from the confusion matrix.

$$\text{False positive rate (FPR)} = \frac{FP}{FP + TN} \tag{5.1}$$

[2] *False positive rate*
[3] *False negative rate*

$$\text{False negative rate (FNR)} = \frac{FN}{FN + TP} \tag{5.2}$$

False positive rate (FPR) is the percentage of false positive (FP) prediction among total number of negative ground truth (FP + TN). Similarly, false negative rate (FNR) is the proportion of false negative (FN) prediction among total number of positive ground truth (FN + TP).

A convolutional neural network (CNN) model can be evaluated in many ways. There are many facts that we can consider for the CNN-based model in parallel with model selection and hyper-parameter selection to improve performance. Some of them are selecting the correct

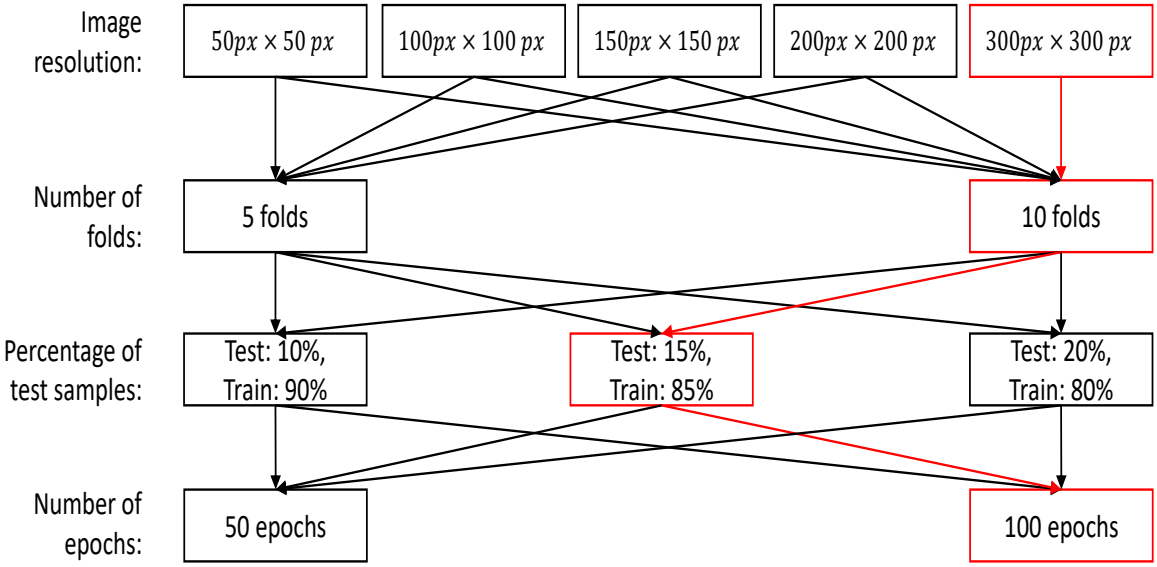


Figure 5.2: Presents the combination of different parameters related to our model training. A set of image resolutions is present in the first row of this figure. Second row, there are two options, either k=5 or k=10, to chose the number of folds while the third row shows the three different percentages of train and test data. Finally, the fourth row presents the two choices for the number of epochs during: 50 epochs or 100 epochs.

resolution of the input images, selecting the correct amount of train and test data, selecting the number of epochs during training, and many more.

We considered these issues in this chapter. Since changing the image resolution has an impact on the performance of the CNN-based model, we generated images with five different resolutions: (a) $50px \times 50px$; (b) $100px \times 100px$; (c) $150px \times 150px$; (d) $200px \times 200px$; and (e) $300px \times 300px$. We used k-fold cross-validation [Kohavi 1995] as the data sampling method where we applied both k=5 and k=10.

We applied cross-validation thrice, completely separating three different amounts of test and train data. Before starting the procedure each time, we shuffled the total data and separated them into test and train. Thus, we trained and tested the model three times in

different amount of test data. The amount of test data was taken as 10%, 15%, and 20%. Figure 5.2 shows the different combinations of selecting image resolution, number of folds, percentage of test data, and the number of epochs for the model training. Here, 1 epoch means a complete cycle of training data passes through the model network. For each combination, we analysed the performance mentioned earlier in this section.

The ACRnet model performs best when the parameters are taken as the red path in Figure 5.2: image resolution $300px \times 300px$, k-fold cross-validation where k=10, the amount of test data is 15% while 85% is for training, and the number of epochs is 100.

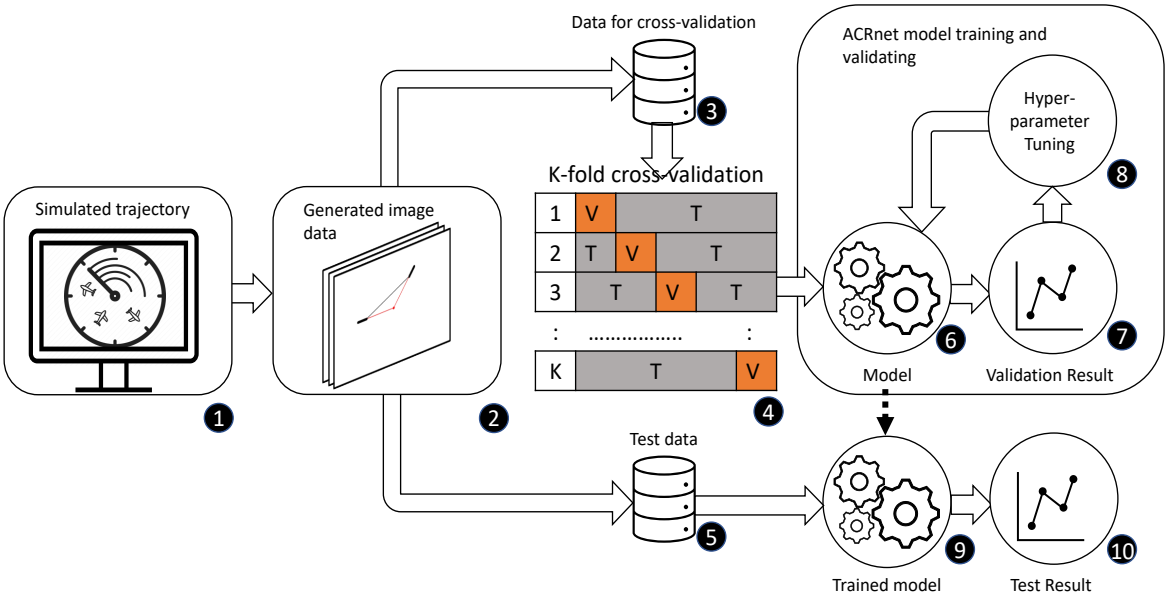


Figure 5.3: Shows the block diagram of overall training, validating, and testing procedure for ACRnet model. The procedure starts from ① where a simulator produces trajectories of conflict scenario. ② is the converted images from the trajectories. The image data set is divided into two parts: ③ data for training (85%) with cross-validatoin & ⑤ test data (15%). The training data in ③ uses for the k-fold cross-validation in ④. The model in ⑥ is trained and validated k times using the k-1 parts of data for training and 1 part for the validation. Based on validation results in ⑦, all the hyper-parameters are updated with the new weights. Finally, the trained model in ⑨ is tested with test data in ⑤ and shows the results in ⑩.

Figure 5.3 describes a complete block diagram of the workflow including the model training, validation, and test. We briefly describe each of them below. At the very beginning of the process, ① in Figure 5.3 shows a lightweight open-source simulator named Blue Sky developed at TU Delft by Hoekstra and Ellerbroek [Hoekstra 2016] was used to simulate conflict between aircraft. Then block ② shows that all the trajectories are transformed into images. The python library *python matplotlib* [Ari 2014] was used. Block ③ and ⑤ are two separate parts of image data. Here, block ③ shows the training data (85% of total data) that uses for cross-validation while block ⑤ shows the 15% of the total data that uses for testing purposes. The block ④ illustrates the complete loop of the k-fold cross-validation

taking data from block ③. In this step, the training data is further divided into k parts. Each time, the model in block ⑥ is trained on k-1 parts of data (indicated by "T" at ④) and 1 part to validate it (indicated by "V" at ④). Based on the validation results at ⑦, the hyper-parameters are tuned in block ⑧ with new weights (modify the initial or the previous weight values). It continues the steps ⑥, ⑦, and ⑧, taking different training and validation data from ④ until the k folds (k=10) are complete. Block ⑨ shows the final trained model that is ready to be tested on unseen data and will provide the results, output heading in our case, in block ⑩.

Since a CNN model requires a lot of data to be trained and because our data is not so much, we used the most common image data augmentation technique. We discussed it in Section 3.4.2. We only applied the rotation feature of the image data augmentation. We used Python Keras for data augmentation which works as follows: first the training data is randomly augmented and the model is trained with that data; then the mode is validated with validation data. In this case, the validation data is not augmented. Random data augmentation is applied for each epoch. Thus, the model is trained with the flavor of new data at each epoch. This augmentation process has the advantage of using a small number of images to get a large number of image data performance as it gets a new set of augmented image data in each epoch. This is very difficult to do when using trajectory data because we have only done manual trajectory augmentation once while Keras automatically augment the image in each epoch.

5.4 Results and Discussion

This section discusses the results we found for the ACRnet and compares our models with other models. There are different types of programming languages and each language has different libraries for machine learning algorithms. Since python is simple in coding compare to other programming language, we use python to implement all the models in this thesis. For instance, python Keras library was used to build our CNN model and python scikit-learn library was used to find different results discussed in Section 5.3.

The performance of a neural network model depends on many things such as accurate annotation of data, choosing the right model, selecting the hyper-parameters using model selection procedure (Section 5.2.1), model training with the appropriate data sampling method, and many more things. We have presented the choices we made in Section 5.2. We first compare our ACRnet model and the CRMLnet model presented in Chapter 4 where we used image data for ACRnet and trajectory data forCRMLnet model. Data annotation is the same for both models. All the results generated for both models use 10-fold cross-validation with 100 epochs: 15% of the data is allocated for testing the model. We applied the same number of folds, same percentage of data for training, validation and test.

Since we have used Keras library to implement both ACRnet and CRMLnet models; it offers many features including training accuracy and validation accuracy curve plot. Also, we can use similar features to plot training and validation loss. This is a widely used plots for

visualizing or comparing the performances of models based on CNN. It is really necessary to understand whether there is any overfitting or variance problem in the model.

Figure 5.4 shows the loss function and Figure 5.5 shows the accuracy function for the CRMLnet model. As we have trained and validated both models (ACRnet and CRMLnet) using 10-fold cross-validation, we should have 10 different results for them. Figure 5.4 and Figure 5.5 plot the loss and accuracy for all the 10-fold independently. In both cases, training results are represented in red color curves while validation results are plotted in green color curves. Here, the x-axis represents the number of epochs used to train the model (up to 100). The y-axis in Figure 5.4 shows the loss in between the range of 0.0 and 0.5 while in Figure 5.5 shows the accuracy between 75% and 100%.

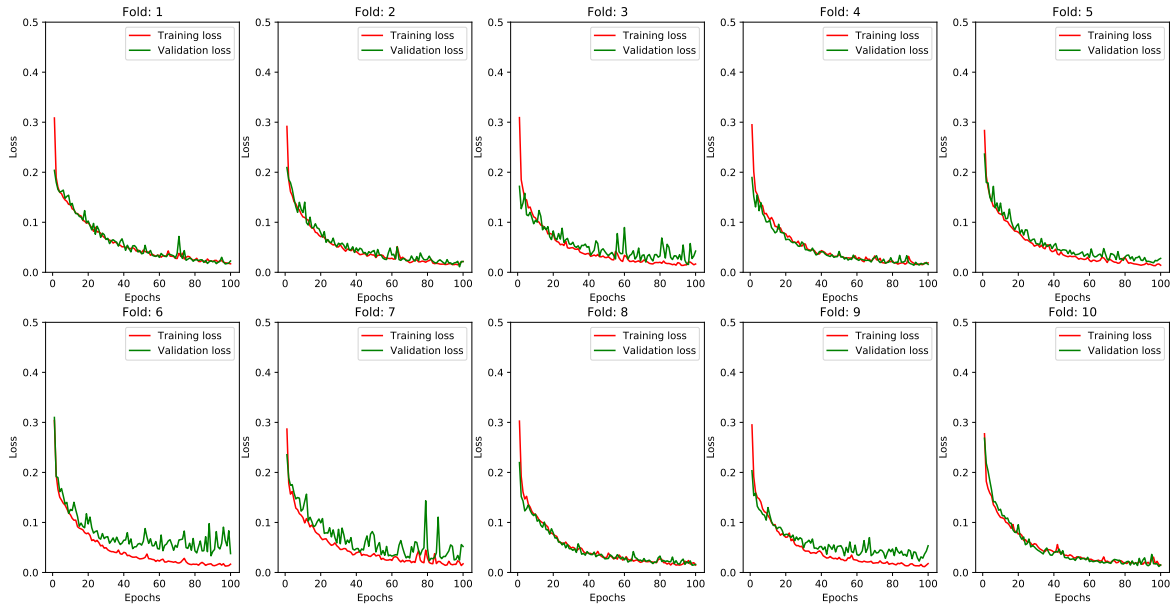


Figure 5.4: Up to 100 epochs, the loss of CRMLnet decreases appropriately. The lower the loss of a model, the better the performance of that model. This figure, presents the training and validation loss changes of the CRMLnet model with respect to the epochs using 10-fold cross-validation. Here separate sub-plots are presented for 10 results. The x-axis shows the number of epochs between 0 and 100 while the y-axis represents the loss between 0.0 and 0.5.

We applied the same procedure for the ACRnet model to visualize the accuracy function and loss function. Figure 5.6 and Figure 5.7 represent all the subplot of the loss and the accuracy functions for the ACRnet model using 10-fold cross-validation. We can find out the difference between the two models. The performance of the CRMLnet model looks a little better than the ACRnet model. However, to understand more clearly the difference in performance between the two models, we need to look not only at the visual graph of the loss and accuracy functions but also at some other performance measures. Thus, all the performance measures discussed in Section 5.3 were applied.

To compare the models, we applied 10-fold cross-validation and recorded the average

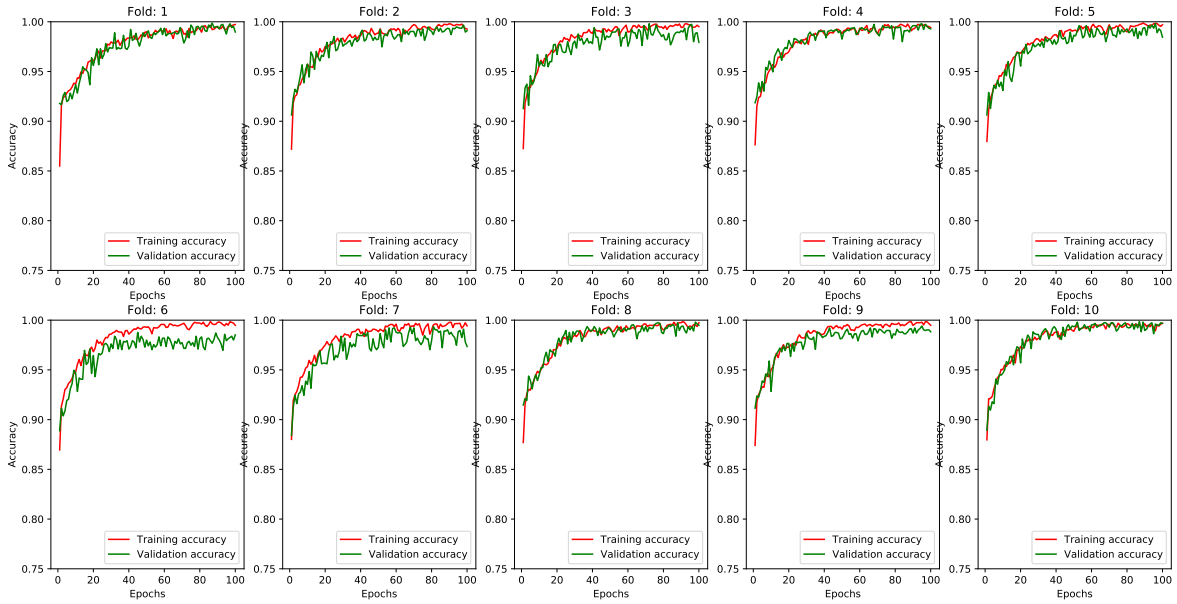


Figure 5.5: Up to 100 epochs, the accuracy of CRMLnet is increased upwards. This figure reads like Figure 5.4 for accuracy which is represented between 75% and 100%.

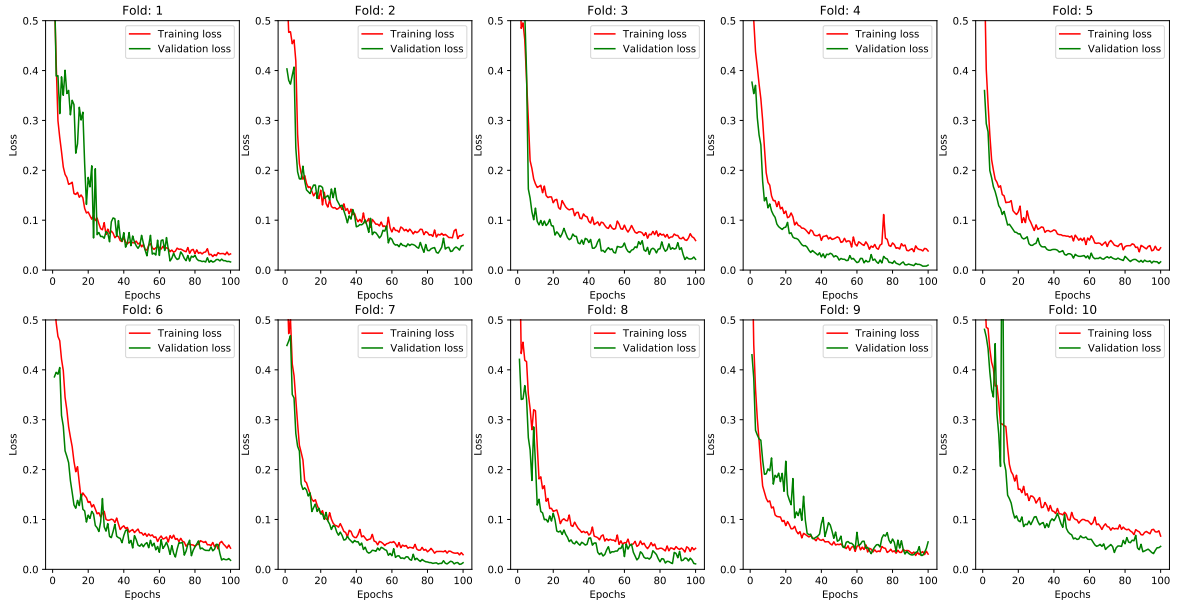


Figure 5.6: Up to 100 epochs, the loss of ACRnet decreases appropriately. The lower the loss of a model, the better the performance of that model. Training and validation loss changes of the ACRnet model (y-axis) with respect to the epochs (x-axis) using 10-fold cross-validation. Separate sub-plots are presented for 10 results.

results for the 10 measurements. For example, all the results on validation data in Table 5.1 are an average of results on 10-fold. The 1^{st} column is the name of the model. The 2^{2nd} column shows the test accuracy of the models. All subsequent columns are: accuracy (Acc), area

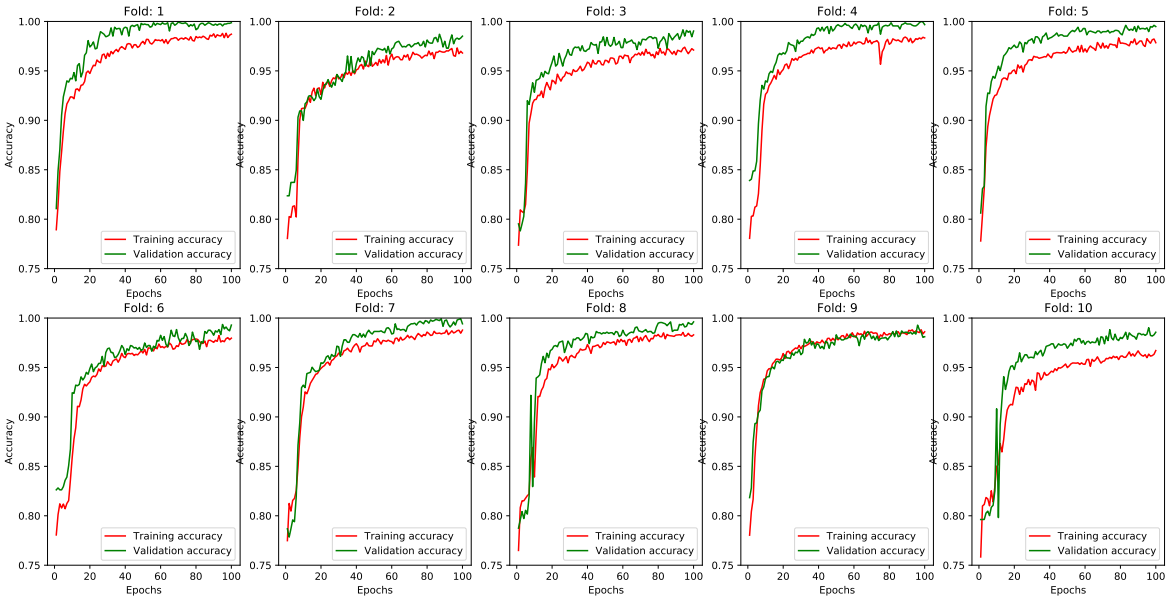


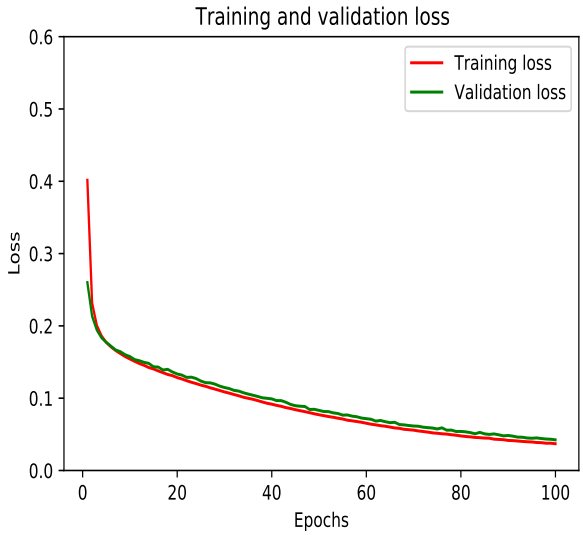
Figure 5.7: Up to 100 epochs, the accuracy of ACRnet increases appropriately. Same as Figure 5.6 where y-axis is accuracy.

under receiver operating characteristic curve (auROC), area under precision-recall (auPR), specificity(S_p), sensitivity (S_n), positive predictive value (PPV), false negative rate (FNR), false positive rate (FPR), Mathew's correlation coefficient (MCC), and F_1 score. We can observe in Figures 5.4, 5.5, 5.6, and 5.7 that CRMLnet is slightly better than ACRnet. But in almost all the cases, Table 5.1 shows that ACRnet is better than CRMLnet including for accuracy on the test with completely unseen data. We stated in the Section 5.1 that our target is not only to handle the variable number of aircraft but also to increase the performance of the model. For this reason, we used data augmentation. In the next discussion, we will see some more results for both models.

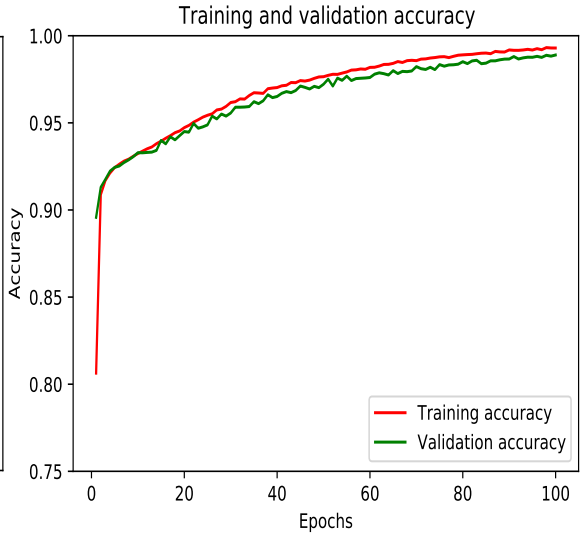
Figure 5.8 (a) and Figure 5.8 (b) show the loss and the accuracy curves for the CRMLnet model when using 10-fold cross-validation.

Figure 5.8 (c) and Figure 5.8 (d) show the loss and the accuracy curves for the ACRnet model. In both cases, the x-axis represents the number of epochs used to train the model up to 100. The y-axes in Figure 5.8 (a) and (c) show the losses in between the range of 0.0 and 0.6 and Figure 5.8 (b) and (d) show the accuracy. In parallel with the visual loss and accuracy,

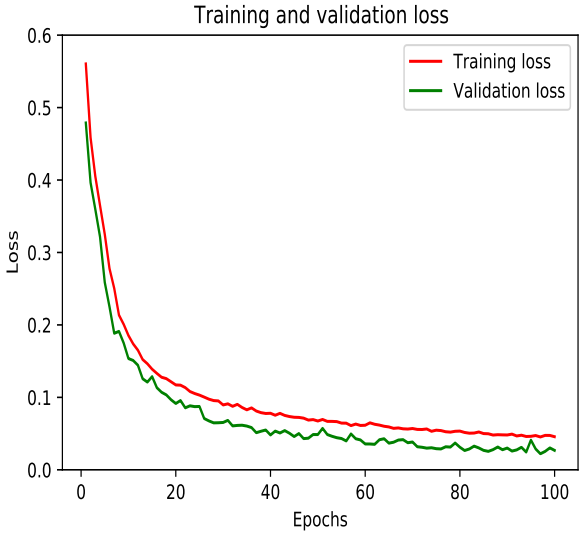
Table 5.1 reports the test and validation results of the models using 10-fold cross-validation. In this table, Block 1 shows the ACRnet model scores for two different datasets. ACRnet_2 are the scores using image data containing two aircraft while ACRnet_3 are the scores when using image containing two or three aircraft. For both two and three aircraft, the ACRnet model architecture remained unchanged. Block 2 represents the performance scores for the CRMLnet model using trajectory data while Block 3 shows the scores for VGG16 [Simonyan 2014] and



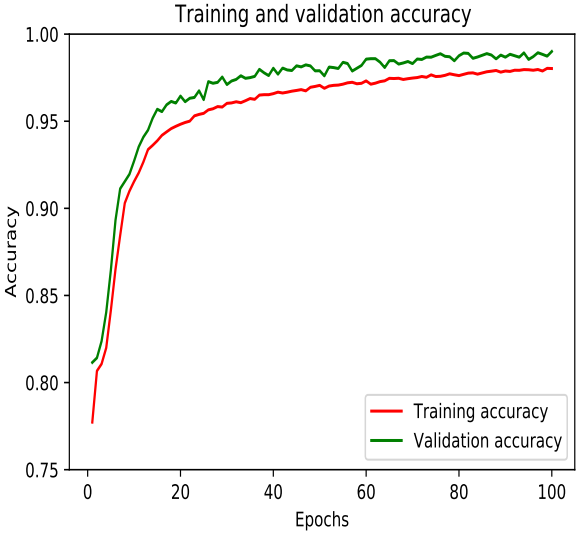
(a) CRMLnet: loss



(b) CRMLnet: accuracy



(c) ACRnet: loss



(d) ACRnet: accuracy

Figure 5.8: Up to 100 epochs both CRMLnet and ACRnet models performed well without any overfitting. This figure shows the average loss and accuracy for CRMLnet and ACRnet models. The comparison between loss (a) & (c) and accuracy (b) & (d) shows that the performances of both models are very close to each other while CRMLnet is a little better than ACRnet when comparing the training and validation curves.

ResNet [He 2016] using the same image data as ACRnet. According to loss and accuracy functions in Figure 5.8, CRMLnet looks a little better than ACRnet but the scores, especially the highlighted scores, in Table 5.1 show that ACRnet is much better than the others. So, using image data not only we handle a variable number of aircraft but also we increase the performance of the model.

Table 5.1 Block 3 shows the scores of VGG16 and ResNet; they do not perform well

compare to ACRnet. This may be because of their many layers. If there were more samples, the performance of those two models may be better. The main purpose of this chapter was not to find the best image processing model but to easily overcome the challenge of existing sequence-based models through image processing with higher performance.

Table 5.1: ACRnet performs much better than CRMLnet, VGG16, and ResNet. Here, the 1^{st} column corresponds to the model name. The 2^{2nd} column is the accuracy on test data of the models. The subsequent columns are: accuracy (Acc), area under receiver operating characteristic curve (auROC), area under precision-recall (auPR), specificity(S_p), sensitivity (S_n), positive predictive value (PPV), false negative rate (FNR), false positive rate (FPR), Mathew's correlation coefficient (MCC), and F_1 score. Block 1 shows the ACRnet model score for two aircraft ($ACRnet_2$) and mixed (two and three) aircraft ($ACRnet_3$). The highlighted scores (Block 1) are the most significant where ACRnet is much better than CRMLnet (Block 2), VGG16 (Block 3), ResNet(Block 3).

Block 1	Test	Validation									
Model	Acc	Acc	auROC	auPR	S_p	S_n	PPV	FNR	FPR	MCC	F_1
$ACRnet_2$	**98.97%**	**99.16%**	**0.999**	0.999	99.41%	98.66%	98.82%	1.34%	0.59%	**0.981**	**0.987**
$ACRnet_3$	**98.96%**	**99.05%**	**1.000**	0.999	99.20%	98.78%	98.63%	1.22%	0.80%	**0.980**	**0.987**
Block 2											
CRMLnet	96.38%	98.76%	0.999	0.999	99.20%	97.87%	98.40%	2.13%	0.80%	0.972	0.981
Block 3											
VGG16	79.97%	80.93%	0.771	0.607	88.93%	65.05%	74.78%	34.95%	11.07%	0.561	0.694
ResNet	92.79%	91.34%	0.973	0.951	92.30%	89.44%	85.42%	10.56%	7.70%	0.809	0.874

One of the most challenging issues in multi-label classification is to properly evaluate its performance. In this chapter, in addition to the overall performance, we present the model performance of each individual class label (Table 3.2) on the test data. Since the output of both models is multiple binary classification, we produce separate test results for each individual class label. In this case, we can make a comparison between the results of each individual classification from CRMLnet and ACRnet. For example, both models classify the aircraft's heading directions (Table 3.2) for conflict resolution, and there are twelve heading directions for each model. Table 5.2 presents the results on the test data for the CRMLnet model for the individual twelve heading directions. Similarly, Table 5.3 presents the results for the ACRnet model. The comparison of the two tables show that in almost all the cases, ACRnet performs much better than CRMLnet. Although both models have an individual heading's accuracy above 90%, the overall individual heading prediction score of ACRnet model is much better than the one of CRMLnet model. The performance of a model cannot be completely determined by only measuring accuracy and/or ROC curve. We show some other performance measures. For example, Table 5.2 reports 100% accuracy using the CRMLnet model for the Right 5^0 direction and F1 score is 0.800. On the other hand, Table 5.3 reports 90.79% accuracy using the ACRnet model for the Left 20^0 direction when the F1 score is 1. Since recently, F1 score measurements were used in many machine learning fields to evaluate a model because of its harmonic mean property (see Section 4.4 Equation 4.9), we also report

F1 scores to show the difference in performance between CRMLnet and ACRnet models.

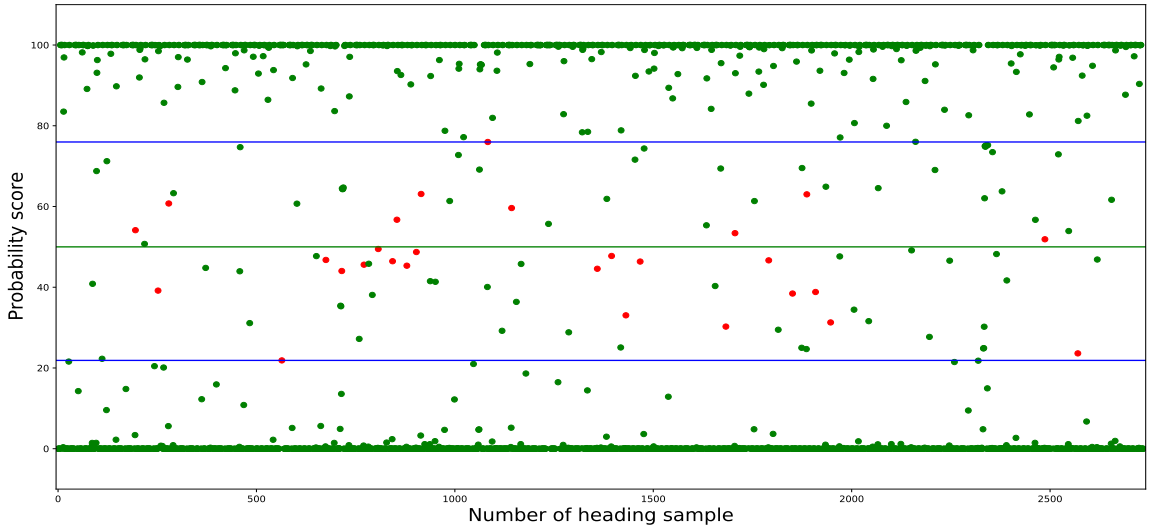
Table 5.2: Individual class label (heading) prediction results of the CMRLnet model on test data. Here, the 1^{st} column is the individual heading direction. All subsequent columns are: accuracy (Acc), area under receiver operating characteristic curve (auROC), area under precision-recall (auPR), specificity(S_p), sensitivity (S_n), positive predictive value (PPV), false negative rate (FNR), false positive rate (FPR), Mathew's correlation coefficient (MCC), and F_1 score.

Heading	Acc	auROC	auPR	S_p	S_n	PPV	FNR	FPR	MCC	F_1
Left 5^0	98.25%	1.000	1.000	99.54%	70.00%	87.50%	30.00%	0.46%	0.774	1.000
Left 10^0	95.18%	0.938	0.800	97.52%	89.55%	93.75%	10.45%	2.48%	0.883	0.889
Left 15^0	90.35%	0.833	0.750	86.21%	94.64%	86.89%	5.36%	13.79%	0.810	0.800
Left 20^0	87.72%	1.000	1.000	83.78%	89.61%	92.00%	10.39%	16.22%	0.724	1.000
Left 25^0	97.37%	1.000	1.000	86.49%	99.48%	97.44%	0.52%	13.51%	0.901	1.000
Left 30^0	97.37%	1.000	1.000	86.49%	99.48%	97.44%	0.52%	13.51%	0.901	1.000
Right 5^0	**100.00%**	0.833	0.750	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	**0.800**
Right 10^0	98.68%	0.417	0.472	99.48%	94.44%	97.14%	5.56%	0.52%	0.950	0.222
Right 15^0	98.25%	1.000	1.000	99.48%	91.89%	97.14%	8.11%	0.52%	0.935	1.000
Right 20^0	97.81%	1.000	1.000	99.48%	89.19%	97.06%	10.81%	0.52%	0.918	1.000
Right 25^0	97.37%	1.000	1.000	99.48%	86.49%	96.97%	13.51%	0.52%	0.901	1.000
Right 30^0	98.25%	0.950	0.667	99.48%	91.89%	97.14%	8.11%	0.52%	0.935	0.800

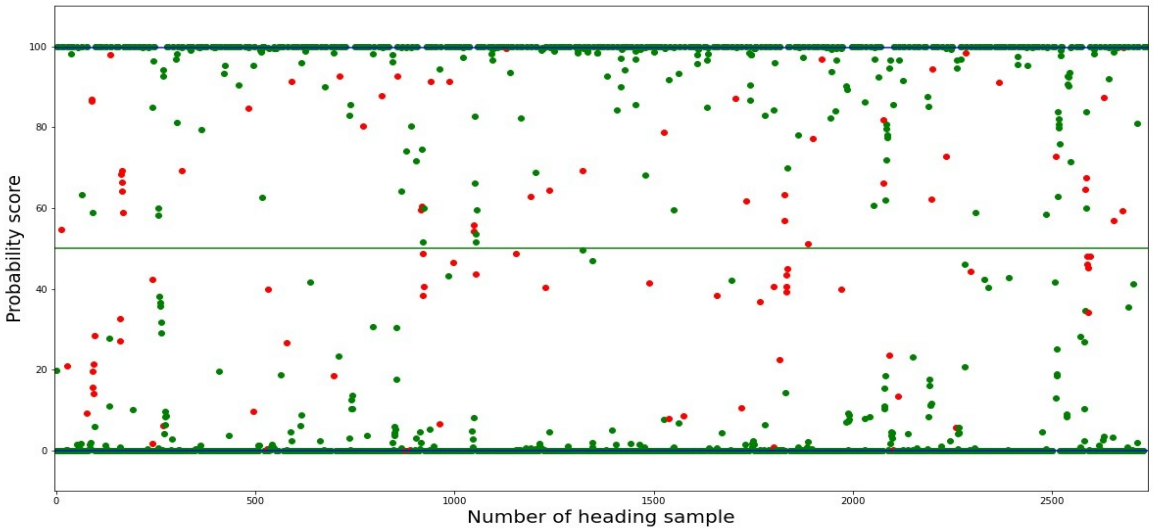
Table 5.3: Individual class label (heading) prediction results of the ACRnet model on test data. The columns are the same as in Table 5.2

Heading	Acc	auROC	auPR	S_p	S_n	PPV	FNR	FPR	MCC	F_1
Left 5^0	99.12%	1.000	1.000	100.00%	80.00%	100.00%	20.00%	0.00%	0.890	1.000
Left 10^0	99.56%	1.000	1.000	100.00%	98.55%	100.00%	1.45%	0.00%	0.990	1.000
Left 15^0	98.68%	1.000	1.000	98.31%	99.09%	98.20%	0.91%	1.69%	0.974	1.000
Left 20^0	**90.79%**	1.000	1.000	89.86%	91.19%	95.39%	8.81%	10.14%	0.790	**1.000**
Left 25^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000
Left 30^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000
Right 5^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000
Right 10^0	99.56%	1.000	1.000	100.00%	97.14%	100.00%	2.86%	0.00%	0.983	1.000
Right 15^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000
Right 20^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000
Right 25^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000
Right 30^0	100.00%	1.000	1.000	100.00%	100.00%	100.00%	0.00%	0.00%	1.000	1.000

We further tested ACRnet and CRMLnet to see the probability score of each heading decision and to understand how image data helps in increasing performance. Figure 5.9 (a) and (b) plot the predicted probability scores of all the individual heading resolutions for ACRnet and CRMLnet. Here, it is an excellent comparison between the ACRnet and CRMLnet models because Figure 5.9 (a) and (b) show the predicted probability scores of all the class labels separately for all the conflict scenarios. For example, each conflict sample is annotated with multiple heading directions, and therefore all individual heading degrees are classified by both ACRnet and CRMLnet models based on their probability scores. In this case, both ACRnet and CRMLnet models have followed the same strategy to classify each



(a) ACRnet: predicted probability



(b) CRMLnet: predicted probability

Figure 5.9: ACRnet is more confident in predicting using image data than CRMLnet with trajectory data. Here x-axes in both (a) and (b) present the total number of heading decisions (288 (test conflict sample) $\times$ 12 (heading degree) = 2736) and the y-axis is the probability between 0% and 100%. All the dots above the green line in the middle (threshold = 50%) are the positive class ('1') and below are the negative class ('0'). All red dots are incorrectly classified that are bounded by the blue lines while green dots are correctly classified. The distance between blue lines (incorrectly classified boundaries) is shorter for ACRnet than it is for CRMLnet. The blue lines are overlapping on the 0% and 100% scoreline for CRMLnet. It means there are some incorrectly classified samples that are closed to 0% and 100%. Thus, ACRnet is more confident than CRMLnet because the shorter the distance between the blue lines, the more confident the model is.

output heading degree. The green horizontal line in the middle of both figures (a) and (b)

indicates 50% of probability score line exactly. All positive classes ('1') are above the green horizontal line and all negative classes ('0') are below that line. The green dots represent the samples that are correctly classified while the red is for misclassified. All misclassified samples are fallen inside the blue line boundaries. For both (a) and (b) in Figure 5.9, x-axis presents the total number of heading degrees (288 (test conflict sample) $\times$ 12 (heading degree) = 2736) and the y-axis presents the probability score of each heading resolution between 0% and 100%. The blue boundary (misclassified) for CRMLnet is overlapped on the 0% scoreline line and 100% scoreline, so, it is invisible in Figure 5.9 (b) while it is clearly visible for ACRnet in Figure 5.9 (a). This means that CRMLnet classifies many samples incorrectly but with a high confidence. Thus, ACRnet is more confident than CRMLnet because the shorter the distance between the blue lines, the more confident the model is.

5.5 Conclusion

This research aims to provide a generalized model to resolve a conflict where a variable number of aircraft are involved. Different aspects were considered: (a) create image data from trajectory sequences; (b) apply augmentation technique to increase the number of training image data ; and (c) find a CNN-based model to classify these images. We defined a relatively small CNN-based model, ACRnet, and found an accuracy of 98.97% and 98.96% for the conflict resolution classification of two and mixed (both two and three) aircraft. The work in this chapter has been published in [Rahman 2021].

We compared our ACRnet model with the CRMLnet model (Chapter 4) using 10-fold cross-validation where the input data of ACRnet are images and of CRMLnet are trajectory sequences. We also compared ACRnet with two other widely used CNN-base models: VGG16 [Simonyan 2014] and ResNet [He 2016]. Overall, ACRnet performs much better than the other models.

The main purpose of this research was not to find the best image processing model, but rather to show that the use of image data not only overcomes the input dimension problem but also increases the performance of the model. An interesting aspect of using image data is that it can be used not only for conflict between planes but can also with other information without changing the image size such as weather, thunderstorms, military zones, etc. Our future research will include that information for better resolution of aircraft conflicts as well as more than 3 aircraft.

Chapter 6

Conclusion and future work

Aircraft conflict resolution is a routine task of the air traffic management system and is going to be a challenge in the future as the number of aircraft is increasing . The goal of our research is to propose a model that would incorporate machine learning techniques to develop an intelligence system for helping humans resolving aircraft conflicts. In this thesis, we considered two widely used and popular methods of machine learning, especially deep learning. Our main challenge was to adapt the conflict environment to the machine learning approach so that our model can make appropriate output decisions.

In the models we proposed and in related work, there are many important factors that need to be considered such as multiple alternative solutions to the same conflict, model input dimensionality problem, easier way to include weather information, etc. The application of machine learning in conflict resolution is at an early stage. Almost all sectors however are more likely to be assisted by machine learning in the future because its performance is close to humans' but also faster than humans.

In this thesis, our main contributions are:

(a) Our first contribution in Chapter 3 was to create a dataset. Indeed, data on aircraft conflicts is not publicly available and difficult to collect from the ATC centers. Simulated data are not available either. We created a total of 1,516 trajectory sequence data and a total of 1,656 image samples, of which 1,516 contain two aircraft and 140 contain three aircraft. The data sets are freely accessible online at `https://independent.academia.edu/MDSIDDIQURRAHMAN9`.

(b) We designed, trained and tested our first CRMLnet model in Chapter 4 using 5 minutes of trajectory sequence of each aircraft involved in the conflict and provided multiple heading resolutions.

(c) Since the input dimension of the machine learning model cannot be changed in real-time, we have converted every conflict scenario into an image to overcome this dimensionality problem. We then applied our second model in Chapter 5 based on a convolutional neural network, ACRnet, to classify these image data. The model could be applied to conflicts implying a variable number of aircraft without changing its architecture.

With regard to (b), the purpose is to classify the heading resolutions following a multi-label classification principle. Here, multi-label means multiple solutions for a single conflict.

Most of the models that have been proposed so far for aircraft conflict resolution are based on the current position of the aircraft involved in the conflict. As opposed to this, we designed the model using the last 5 minutes of trajectory data, which provides more context for the decision. For evaluation purpose, we also designed three models where a single-label classifier was used for each heading decision. These models are based on SVM, KNC, and LR. We found that our CRMLnet model performs better than these models based on single-label classifier. After training and testing, CRMLnet obtained 98.72% of accuracy using 10-fold cross-validation and 97.79% with independent test set. Although this model performed very well, the limitation is that it resolves conflicts following a pairwise approach. This model could be used to resolve conflicts for more than two aircraft, but it would be necessary to change the input dimensions of the model. This is not possible at run time.

There is a limitation to using trajectory-based models to manage input dimensionality. There are models such as recurrent neural network (RNN), long short term memory (LSTM), etc. to handle a variable number of input. In that case, the computation of these models depend on the number of inputs. In our case, it is also difficult to develop such a model because we are using 5 minutes of the trajectory of each aircraft instead of using a single point.

Thus, we proposed another model based on a convolutional neural network, ACRnet (contribution (c)), where we converted the whole conflict scene into an image, so that the model can take an image as its input. As a result, if we plot an arbitrary number of planes in the image, there will be no change in the model architecture. Therefore, the model can resolve conflicts with a variable number of aircraft without any change in input dimension. We compared our ACRnet model with two widely used image processing models: VGG16 and ResNet. We also compared ACRnet and CRMLnet using the same data. ACRnet performs better than the other models on the various measures we used. ACRnet model achieves an accuracy of 99.16% on the training data and of 98.97% on the test data set for two aircraft. For both two and three aircraft, the accuracy is 99.05% (resp. 98.96%) on the training (resp. test) data set. It is worth mentioning that the purpose of our research was not to find the best image processing model but to show that the input dimensionality problem can be resolved with the image-based model. We found that using images not only perform better but also more robustly than using a trajectory sequence.

The further advantage of using image-based models is that we can incorporate uncertainties such as bad weather or restricted military zones into the picture. The same model could then be applied without changing the model architecture.

For other future work, there are many strategies to improve image-based models by improving image quality. Also, we could include speed and heading information within the image using different transformation techniques. Finally, we could try to improve the image processing model by changing the NN architecture.

Appendix A

CRMLnet model implementation

Step by step implementation of CRMLnet full code using python programming.

Show the TensorFlow and Python version

```
import tensorflow as tf
print("TensorFlow version: ", tf.__version__)
from platform import python_version
print("Python version: ", python_version())
```

Select the specific GPU. We select GPU No: 0

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="0"
```

Import necessary libraries

```
import keras
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Input, Dense, Dropout, \
Activation
from tensorflow.keras.optimizers import SGD, Adamax, RMSprop
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, log_loss, \
classification_report, confusion_matrix, roc_auc_score, \
roc_curve,auc, average_precision_score, f1_score
```

No random state, always start from 1

```
np.random.seed(seed=1)
```

Load dataset CSV file from the local hard-disk

```
Dataset = '/local-directory/Dataset.csv'
plot_path = "/local-directory/output/"
D = pd.read_csv(Dataset)
print("Number of total entries: ",len(D))
n_columns = len(D.columns)
print("Number of total features: ", n_columns-12)
print("Number of total class labels: ", n_columns-271)
D.head()
```

Standard scaling

```
scale = StandardScaler()
```

For splitting data into 10-Fold and shuffling them where the random state is 100

```
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=100)
```

Batch size

```
batch_size = 28
```

Number of epoch

```
e = 100
```

Neural network model building function

```
def build_model(X, y):
    inputs = Input(shape=(X.shape[1],))
    outputs = Dense(271, activation='relu')(inputs)
    predictions = Dense(y.shape[1], activation='sigmoid')(outputs)
    model = Model(inputs=inputs, outputs=predictions)
    return model
```

Adaptive Moment Estimation (Adam) optimizer

```
from tensorflow.keras.optimizers import Adam
optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
```

Split the dataset into feature X and class label y

```
X = D.iloc[:,:-12].values
y = D.iloc[:,-12:].values
```

Shuffling X and y together

```
from sklearn.utils import shuffle
X, y = shuffle(X, y, random_state=200)
```

Split X and y into train and test dataset where the amount of test data is 15%

```
XX, x_test, yy, y_test = train_test_split(X, y, random_state = 1, \
test_size=0.15, shuffle=True)
XX = scale.fit_transform(XX)
```

Libraries and variables for output results

```
from sklearn.metrics import confusion_matrix, \
roc_auc_score, average_precision_score
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from scipy import interp
CM = np.zeros((2, 2), dtype=int)
accuracy = []
auroc = []
aupr = []
F1 = []
tprs = []
aucs = []
acc = 0.0
fold = 1
mean_fpr = np.linspace(0, 1, 100)
history_dict = []
```

10-fold cross-validation training and validation

```
count = 1
plt.figure(figsize=(8,6))
for train_index, val_index in cv.split(XX,yy[:,0]):
    print("********* Fold {} **********".format(count))
    count += 1
    x_train = XX[train_index]
    x_val = XX[val_index]

    y_train = yy[train_index]
    y_val = yy[val_index]
```

```
    # Model build
    model = build_model(XX, yy)
    model.compile(loss='binary_crossentropy', \
    optimizer=optimizer, metrics=['acc'])
    history = model.fit(x_train, y_train, epochs=e, \
    batch_size=batch_size, validation_data=(x_val, y_val))
    history_dict.append(history.history)

    # For prediction,
    y_proba = model.predict(x_val).ravel()


    y_pred = np.where(y_proba >= 0.5, 1, 0)


    test = y_val.flatten()
    fpr_keras, tpr_keras, thresholds_keras = \
    roc_curve(test, y_proba)
    auc_keras = auc(fpr_keras, tpr_keras)
    #plt.plot(fpr_keras, tpr_keras, \
    label='Keras (area = {:.3f})'.format(auc_keras))
    fpr, tpr, thresholds = roc_curve(test, y_proba)
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)


    auroc.append(roc_auc_score(y_true=test, y_score=y_proba))
    aupr.append(average_precision_score(y_true=test, \
    y_score=y_proba))
    accuracy.append(accuracy_score(y_pred=y_pred, y_true=test))
    F1.append(f1_score(y_pred=y_pred, y_true=test))
    CM += confusion_matrix(y_pred=y_pred, y_true=test)
    fold += 1
```

To display different result obtained from the confusion matrix

```
TN, FP, FN, TP = CM.ravel()

print('| Acc | auROC | auPR | Sp  | Sn  | PPV | FNR | \
FPR | MCC | F1  |')
```

```
print('|%.2f' % (np.mean(accuracy)*100)\
+'|%.3f' % (np.mean(auroc))\
+'|%.3f' % (np.mean(aupr)) \
+'|%.2f'% ((TN / (TN + FP))*100)\
+'|%.2f' % ((TP / (TP + FN))*100) \
+'|%.2f' % ((TP / (TP + FP))*100) \
+'|%.2f' % ((FN / (FN + TP))*100) \
+'|%.2f' % ((FP / (FP + TN))*100) \

+ '|%.3f' % ((TP*TN-FP*FN)/(np.sqrt((TP+FP)*(TP+FN)*(TN+FP)* \
(TN+FN)))) + '|%.3f|' % (np.mean(F1)))
```

Bibliography

[Agarap 2018] Abien Fred Agarap. *Deep learning using rectified linear units (relu).* arXiv preprint arXiv:1803.08375, 2018.

[Alam 2007] Sameer Alam, Kamran Shafi, Hussein A Abbass et Michael Barlow. *Evolving air traffic scenarios for the evaluation of conflict detection models.* In Proc. 6th Eurocontrol Innovative Research Workshop, Eurocontrol Experiment Research Center, 2007.

[Alam 2009] Sameer Alam, Kamran Shafi, Hussein A Abbass et Michael Barlow. *An ensemble approach for conflict detection in free flight by data mining.* Transportation research part C: emerging technologies, vol. 17, no. 3, pages 298–317, 2009.

[Alonso-Ayuso 2010] Antonio Alonso-Ayuso, Laureano F Escudero et F Javier Martín-Campo. *Collision avoidance in air traffic management: A mixed-integer linear optimization approach.* IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 1, pages 47–57, 2010.

[Alonso-Ayuso 2012] Antonio Alonso-Ayuso, Laureano F Escudero et F Javier Martín-Campo. *A mixed 0–1 nonlinear optimization model and algorithmic approach for the collision avoidance in ATM: Velocity changes through a time horizon.* Computers & Operations Research, vol. 39, no. 12, pages 3136–3146, 2012.

[Alonso-Ayuso 2013] Antonio Alonso-Ayuso, Laureano F Escudero, Pablo Olaso et Celeste Pizarro. *Conflict avoidance: 0-1 linear models for conflict detection & resolution.* Top, vol. 21, no. 3, pages 485–504, 2013.

[Alonso-Ayuso 2016a] Antonio Alonso-Ayuso, Laureano F Escudero et F Javier Martín-Campo. *An exact multi-objective mixed integer nonlinear optimization approach for aircraft conflict resolution.* Top, vol. 24, no. 2, pages 381–408, 2016.

[Alonso-Ayuso 2016b] Antonio Alonso-Ayuso, Laureano F Escudero et F Javier Martín-Campo. *Multiobjective optimization for aircraft conflict resolution. A metaheuristic approach.* European Journal of Operational Research, vol. 248, no. 2, pages 691–702, 2016.

[Archibald 2008] James K Archibald, Jared C Hill, Nicholas A Jepsen, Wynn C Stirling et Richard L Frost. *A satisficing approach to aircraft conflict resolution.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 38, no. 4, pages 510–521, 2008.

[Ari 2014] Niyazi Ari et Makhamadsulton Ustazhanov. *Matplotlib in python.* In 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), pages 1–6. IEEE, 2014.

[Baker 2017] Simon Baker et Anna-Leena Korhonen. *Initializing neural networks for hierarchical multi-label text classification.* Association for Computational Linguistics, 2017.

[Banks 2005] Jerry Banks. Discrete event system simulation. Pearson Education India, 2005.

[Bergstra 2012] James Bergstra et Yoshua Bengio. *Random search for hyper-parameter optimization.* The Journal of Machine Learning Research, vol. 13, no. 1, pages 281–305, 2012.

[Bilimoria 2000] Karl Bilimoria. *A geometric optimization approach to aircraft conflict resolution.* In 18th Applied aerodynamics conference, page 4265, 2000.

[Bottou 2012] Léon Bottou. *Stochastic gradient descent tricks.* In Neural networks: Tricks of the trade, pages 421–436. Springer, 2012.

[Brittain 2018] Marc Brittain et Peng Wei. *Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning.* In Proceedings of the International Conference for Research in Air Transportation, 2018.

[Brittain 2021] Marc W Brittain et Peng Wei. *One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory.* In AIAA Scitech 2021 Forum, page 1952, 2021.

[Brudnicki 1997] DJ Brudnicki et AL McFarland. *User Request Evaluation Tool (URET) conflict probe performance and benefits assessment.* In Proc. USA/Europe ATM Seminar. Eurocontrol, 1997.

[Carbone 2006] Ciro Carbone, Umberto Ciniglio, Federico Corraro et Salvatore Luongo. *A novel 3D geometric algorithm for aircraft autonomous collision avoidance.* In Proceedings of the 45th IEEE Conference on Decision and Control, pages 1580–1585. IEEE, 2006.

[Cortes 1995] Corinna Cortes et Vladimir Vapnik. *Support-vector networks.* Machine learning, vol. 20, no. 3, pages 273–297, 1995.

[Das 2010] Santanu Das, Bryan L Matthews, Ashok N Srivastava et Nikunj C Oza. *Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study.* In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 47–56. ACM, 2010.

[Das 2011] Santanu Das, Bryan L Matthews et Robert Lawrence. *Fleet level anomaly detection of aviation safety data.* In 2011 IEEE Conference on Prognostics and Health Management, pages 1–10. IEEE, 2011.

[Durand 2020] Nicolas Durand, Jean-Baptiste Gotteland, Nadine Matton, Léa Bortolotti et Margot Sandt. *Understanding and overcoming horizontal separation complexity in air traffic control: an expert/novice comparison.* Cognition, Technology & Work, pages 1–16, 2020.

[Eby 1999] Martin S Eby et Wallace E Kelly. *Free flight separation assurance using distributed algorithms.* In 1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403), volume 2, pages 429–441. IEEE, 1999.

[Farley 2007] Todd Farley et Heinz Erzberger. *Fast-time simulation evaluation of a conflict resolution algorithm under high air traffic demand.* In 7th USA/Europe ATM 2007 R&D Seminar, 2007.

[Feron 2013] Eric Feron, Antonio Bicchi et Lucia Pallottino. *Mixed integer programming for aircraft conflict resolution.* In AIAA guidance, navigation, and control conference and exhibit, page 4295, 2013.

[Frazier 2018] Peter I Frazier. *A tutorial on Bayesian optimization.* arXiv preprint arXiv:1807.02811, 2018.

[Harris 2010] David Harris et Sarah Harris. Digital design and computer architecture. Morgan Kaufmann, 2010.

[Havel 1989] Karel Havel et Jaroslav Husarčík. *A theory of the tactical conflict prediction of a pair of aircraft.* The Journal of Navigation, vol. 42, no. 3, pages 417–429, 1989.

[He 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. *Deep residual learning for image recognition.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[He 2018] Huihui He et Rui Xia. *Joint binary neural network for multi-label learning with applications to emotion classification.* In CCF International Conference on Natural Language Processing and Chinese Computing, pages 250–259. Springer, 2018.

[Hoekstra 2016] Jacco M Hoekstra et Joost Ellerbroek. *Bluesky atc simulator project: an open data and open source approach.* In Proceedings of the 7th International Conference on Research in Air Transportation, pages 1–8. FAA/Eurocontrol USA/Europe, 2016.

[Hu 1999] Jianghai Hu, John Lygeros, Maria Prandini et Shankar Sastry. *Aircraft conflict prediction and resolution using Brownian Motion.* In Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304), volume 3, pages 2438–2443. IEEE, 1999.

[Hurter 2013] C. Hurter, G. Andrienko, N. Andrienko, R.H. Güting et M. Sakr. Air traffic analysis, pages 240–258. Cambridge University Press, 2013.

[Jiang 2018] Xu-rui Jiang, Xiang-xi Wen, Ming-gong Wu, Ze-kun Wang et Xi Qiu. *A SVM approach of aircraft conflict detection in free flight.* Journal of Advanced Transportation, vol. 2018, 2018.

[Kim 2016] Kwangyeon Kim, Inseok Hwang et Bong-Jun Yang. *Classification of Conflict Resolution Methods using Data-Mining Techniques.* In AIAA, page 4075, 2016.

[Kohavi 1995] Ron Kohavi*et al.* *A study of cross-validation and bootstrap for accuracy estimation and model selection.* In Ijcai, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[Kuchar 2000] James K Kuchar et Lee C Yang. *A review of conflict detection and resolution modeling methods.* IEEE Trans. on intelligent transportation syst., vol. 1, no. 4, pages 179–189, 2000.

[LaValle 2004] Steven M LaValle, Michael S Branicky et Stephen R Lindemann. *On the relationship between classical grid search and probabilistic roadmaps.* The International Journal of Robotics Research, vol. 23, no. 7-8, pages 673–692, 2004.

[Mao 2001] Zhi-Hong Mao, Eric Feron et Karl Bilimoria. *Stability and performance of intersecting aircraft flows under decentralized conflict avoidance rules.* IEEE Trans. on Intelligent Transportation Syst., vol. 2, no. 2, pages 101–109, 2001.

[Matthews 2013] Bryan Matthews, Santanu Das, Kanishka Bhaduri, Kamalika Das, Rodney Martin et Nikunj Oza. *Discovering anomalous aviation safety events using scalable data mining algorithms.* Journal of Aerospace Information Systems, vol. 10, no. 10, pages 467–475, 2013.

[Mörters 2010] Peter Mörters et Yuval Peres. Brownian motion, volume 30. Cambridge University Press, 2010.

[Nanduri 2016] Anvardh Nanduri et Lance Sherry. *Anomaly detection in aircraft data using Recurrent Neural Networks (RNN).* In 2016 Integrated Communications Navigation and Surveillance (ICNS), pages 5C2–1. IEEE, 2016.

[Olive 2018] Xavier Olive, Jeremy Grignard, Thomas Dubot et Julie Saint-Lot. *Detecting Controllers' Actions in Past Mode S Data by Autoencoder-Based Anomaly Detection.* In SID 2018, 8th SESAR Innovation Days, 2018.

[O'Malley 2019] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi*et al.* *KerasTuner.* `https://github.com/keras-team/keras-tuner`, 2019.

[Ota 1998] Tomoki Ota, M Nagati et Dong-Chan Lee. *Aircraft collision avoidance trajectory generation.* In Guidance, Navigation, and Control Conference and Exhibit, page 4241, 1998.

[Panchal 2011] Gaurang Panchal, Amit Ganatra, YP Kosta et Devyani Panchal. *Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers.* International Journal of Computer Theory and Engineering, vol. 3, no. 2, pages 332–337, 2011.

[Pavlinović 2013] Mira Pavlinović, Damir Boras et Ivana Francetić. *First Steps in Designing Air Traffic Control Communication Language Technology System-Compiling Spoken Corpus of Radiotelephony Communication.* International Journal of Computers and Communications, vol. 7, no. 3, page 73, 2013.

[Peyronne 2012] Clément Peyronne. *Modélisation mathématique et résolution automatique de conflits par algorithmes génétiques et par optimisation locale continue.* PhD thesis, Université Paul Sabatier-Toulouse III, 2012.

[Pham 2019a] Duc-Thinh Pham, Ngoc Phu Tran, Sameer Alam, Vu Duong et Daniel Delahaye. *A Machine Learning Approach for Conflict Resolution in Dense Traffic Scenarios with Uncertainties.* 2019.

[Pham 2019b] Duc-Thinh Pham, Ngoc Phu Trant, Sim Kuan Goh, Sameer Alam et Vu Duong. *Reinforcement Learning for Two-Aircraft Conflict Resolution in the Presence of Uncertainty.* In 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF), pages 1–6. IEEE, 2019.

[Prandini 1999] Maria Prandini, John Lygeros, Arnab Nilim et Shankar Sastry. *A probabilistic framework for aircraft conflict detection.* In Guidance, Navigation, and Control Conference and Exhibit, page 4144, 1999.

[Prandini 2000] Maria Prandini, Jianghai Hu, John Lygeros et Shankar Sastry. *A probabilistic approach to aircraft conflict detection.* IEEE Transactions on intelligent transportation systems, vol. 1, no. 4, pages 199–220, 2000.

[Prats Menéndez 2018] Xavier Prats Menéndez, Ignacio Agüi, Fedja Netjasov, Goran Pavlovic et Andrija Vidosavljevic. *APACHE-Final project results report.* 2018.

[Pritchett 2017] Amy R Pritchett et Antoine Genton. *Negotiated decentralized aircraft conflict resolution.* IEEE transactions on intelligent transportation systems, vol. 19, no. 1, pages 81–91, 2017.

[Rahman 2020] Md Siddiqur Rahman. *Supervised machine learning model to help controllers solving aircraft conflicts.* In ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium, pages 355–361. Springer, 2020.

[Rahman 2021] Md Siddiqur Rahman, Laurent Lapasset et Josiane Mothe. *Aircraft Conflict Resolution using Convolutional Neural Network on Trajectory Image.* In 21st international conference intelligent systems design and applications. Springer, 2021.

[Rahman 2022] Md Siddiqur Rahman, Laurent Lapasset et Josiane Mothe. *Multi-label Classification of Aircraft Heading Changes Using Neural Network to Resolve Conflicts.* arXiv preprint arXiv:2109.04767, 2022.

[Richards 2002] Arthur Richards et Jonathan P How. *Aircraft trajectory planning with collision avoidance using mixed integer linear programming.* In Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301), volume 3, pages 1936–1941. IEEE, 2002.

[Ruder 2016] Sebastian Ruder. *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.

[Schäfer 2014] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic et Matthias Wilhelm. *Bringing up OpenSky: A large-scale ADS-B sensor network for research.* In Proceedings of the 13th international symposium on Information processing in sensor networks, pages 83–94. IEEE Press, 2014.

[Simonyan 2014] Karen Simonyan et Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.

[Siqi 2018] HAO Siqi, Shaowu Cheng et Yaping Zhang. *A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation.* Chinese Journal of Aeronautics, vol. 31, no. 7, pages 1579–1593, 2018.

[Sridhar 1997] B Sridhar et GB Chatterji. *Computationally efficient conflict detection methods for air traffic management.* In Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041), volume 2, pages 1126–1130. IEEE, 1997.

[Srinivasamurthy 2018] Ajay Srinivasamurthy, Petr Motlicek, Mittul Singh, Youssef Oualil, Matthias Kleinert, Heiko Ehr et Hartmut Helmke. *Iterative Learning of Speech Recognition Models for Air Traffic Control.* In Interspeech, pages 3519–3523, 2018.

[Tsoumakas 2007] Grigorios Tsoumakas et Ioannis Katakis. *Multi-label classification: An overview.* International Journal of Data Warehousing and Mining (IJDWM), vol. 3, no. 3, pages 1–13, 2007.

[Vela 2010] Adan E Vela, Senay Solak, John-Paul B Clarke, William E Singhose, Earl R Barnes et Ellis L Johnson. *Near real-time fuel-optimal en route conflict resolution.* IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 4, pages 826–837, 2010.

[Wandelt 2014] Sebastian Wandelt et Xiaoqian Sun. *Efficient compression of 4D-trajectory data in air traffic management.* IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pages 844–853, 2014.

[Wang 2019] Zhuang Wang, Hui Li, Junfeng Wang et Feng Shen. *Deep reinforcement learning based conflict detection and resolution in air traffic control.* IET Intelligent Transport Systems, vol. 13, no. 6, pages 1041–1047, 2019.

[Warren 1997] Anthony Warren. *Medium term conflict detection for free routing: Operational concepts and requirements analysis.* In 16th DASC. AIAA/IEEE Digital Avionics Systems Conference. Reflections to the Future. Proceedings, volume 2, pages 9–3. IEEE, 1997.

[Yanling 2002] Zhao Yanling, Deng Bimin et Wang Zhanrong. *Analysis and study of perceptron to solve XOR problem.* In The 2nd International Workshop on Autonomous Decentralized System, 2002., pages 168–173. IEEE, 2002.

[Zeghal 1998] Karim Zeghal. *A review of different approaches based on force fields for airborne conflict resolution.* In Guidance, Navigation, and Control Conference and Exhibit, page 4240, 1998.

[Zhao 2020] Peng Zhao et Yongming Liu. *Efficient Multiple Aircraft Conflict Resolution Using A* Algorithm and Indexing Method.* In AIAA AVIATION 2020 FORUM, page 2918, 2020.

[Zhao 2021] Peng Zhao et Yongming Liu. *Physics Informed Deep Reinforcement Learning for Aircraft Conflict Resolution.* IEEE Transactions on Intelligent Transportation Systems, 2021.