A Neural Network Approach to Aircraft Performance Model Forecasting

Nicolas Vincent-Boulay

A Thesis in the Department of
Mechanical, Industrial and Aerospace Engineering (MIAE)

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Sciences in
Mechanical Engineering

Concordia University
Montreal, Quebec, Canada

May 2020

Signatures

This is to certify that the thesis prepared

By: Nicolas Vincent-Boulay

Entitled: A Neural Network Approach to Aircraft Performance Model Forecasting

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Sciences (Mechanical Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair

Dr. Brian Vermeire

_____ Examiner

Dr. Abdessamad Ben Hamza

_____ Examiner

Dr. Brian Vermeire

_____ Supervisor

Dr. Catharine Marsden

Approved by _____ _____

Dr. Waiz Ahmed Date

Chair of Department of Graduate Program Director

_____ _____

Dr. Mourad Debbabi Date

Dean of Faculty

Abstract

Performance models used in the aircraft development process are dependent on the assumptions and approximations associated with the engineering equations used to produce them. The design and implementation of these highly complex engineering models are typically associated with a longer development process. This study proposes a non-deterministic approach where machine learning techniques using Artificial Neural Networks are used to predict specific aircraft parameters using available data. The approach yields results that are independent of the equations used in conventional aircraft performance modeling methods and rely on stochastic data and its distribution to extract useful patterns. To test the viability of the approach, a case study is performed comparing a conventional performance model describing the takeoff ground roll distance with the values generated from a neural network using readily-available flight data. The neural network receives as input, and is trained using, aircraft performance parameters including atmospheric conditions (air temperature, air pressure, air density), performance characteristics (flap configuration, thrust setting, MTOW, etc.) and runway conditions (wet, dry, slope angle, etc.). The proposed predictive modeling approach can be tailored for use with a wider range of flight mission profiles such as climb, cruise, descent and landing.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Dr. Catharine Marsden, for her heartfelt dedication and support in the development of this work. Above all, I am grateful for her unyielding honesty and genuineness in every engineering discussion we have had the chance to have through the years we have known each other.

Thank you to Claude, Linda, Olivia and Claudia for the ever-present support and for lending me these precious moments where we explored the fascinating boundary between our two, often diverging, world views.

I would also like to thank the employees of the aerospace industrial partners from whom I've had the incredible opportunity to learn so much from. As so elegantly put by Albert Einstein: "The only source of knowledge is experience". Many thanks to NSERC (Natural Sciences and Engineering Research Council of Canada) and the NSERC Chair in Aerospace Design Engineering (NCADE) industrial partners for their financial support, and to Concordia University.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Symbols

Acronym/Symbol	Description
AEO	All Engines Operating
AFM	Aircraft Flight Manual
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASD	Accelerated Stop Distance
CNN	Convolutional Neural Network
D	Drag
DARPA	U.S. Defense Advanced Research Projects Agency
DL	Deep Learning
DoD	U.S. Department of Defense
ES	Expert System
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulation
ICAO	International Civil Aviation Organization
ISA	ICAO Standard Atmosphere
L	Lift
LTSM	Long Term Short Term Memory
LT	Logic Theorist
ML	Machine Learning
MAE	Mean Average Error
MAPE	Mean Average Percentage Error
MSE	Mean Squared Error
MSLE	Mean Squared Logarithmic Error
NN	Neural Network
NASA	National Aeronautics and Space Administration
OEI	One Engine Inoperative
OEM	Original Equipment Manufacturer
$p_{err,train}$ or $p_{err,test}$	Worst-case percentage error for training or testing
POH	Pilot's Operating Handbook
RNN	Recurrent Neural Network
T	Temperature or Thrust

TOD	Takeoff Distance
TOD_N or TOD_{AEO}	TOD with all engines operating
TOD_{N-1} or TOD_{OEI}	TOD with one engine inoperative
TOFL	Takeoff Field Length
TOR	Takeoff run
V_1	Takeoff decision point speed
V_2	Takeoff safety speed
V_3	Initial climbout speed, all engines operating
V_{EF}	Speed at engine failure
V_G	Ground speed
V_{LOF}	Speed at liftoff
V_R	Rotation speed
V_S	Stall speed
W	Weight
δ	Pressure ratio or relative pressure ($\delta = p/p_o$)
Θ	Temperature ratio or relative temperature ($\Theta = T/T_o$)
ρ	Density
ρ_o	Density at sea-level
σ	Density ratio or relative density ($\sigma = \rho / \rho_o$)

1. Introduction

1.1 Background on Deterministic and Non-Deterministic System Models

The modern aircraft design process has evolved to include systems of increasing complexity. To better understand these systems, engineering models are developed, where most of these models are said to be deterministic in nature. A deterministic model is a system in which all output parameters can be calculated from their relationship with other parameter values affecting the system and their initial conditions. In other words, it is a system for which all possible states are understood and, to some extent, are predictable. Many engineering problems can be solved using deterministic models. Deterministic methods have proven successful in applications where the mechanisms used to describe the full behavior of a system can be completely understood or understood enough to be able to describe a phenomenon with a minimum amount of acceptable error. Newton's laws of motion are examples of important deterministic models. When they are applied on a body, it is said that the future outcomes of the body can be predicted or determined by its present situation; and if they are applied on two identical bodies under the same conditions the outcome for both bodies will be same. In the field of aircraft design, many deterministic systems play fundamental contributing roles in the design process, such as models describing the motion of the aircraft in time and space; models tracking and predicting the state of onboard systems of the aircraft (i.e. fuel systems, electrical systems, environmental control systems); models monitoring and affecting the state of flight control systems; or models describing aerodynamic or structural loads on an aircraft's components. Although these models and their successful use are noteworthy, it must be understood that all deterministic models carry inherent limitations, some of which are listed below:

- It is possible to have not enough or no empirical data to support the development of the deterministic model.
- Predicted results can be outside the acceptable error margins when compared with empirical data.
- Highly complex deterministic models can be computationally expensive and require longer run times.
- Expert knowledge of the system being developed is always required in order to conceive a new deterministic model.
- If a system is too complex to be able to develop the deterministic equations defining its behavior, it can be impossible or extremely hard to develop this deterministic model.

These limitations contributed to the fact that *non-deterministic* models began to be investigated as a means of addressing some of the deterministic model limitations found in the aerospace industry [1, 2], where it

is typical to have a high complexity system with very large amounts of data for which expert knowledge is almost always required. A non-deterministic model describes a system for which the behavior of the system parameters is said to be stochastic and for which no deterministic relationships are possible. This type of modeling approach is based on probability and statistics, which introduce randomness in the models in such a way that the outcomes of the model can be viewed as probability distributions rather than unique values. Thus, non-deterministic methods can produce different outcomes after multiple runs for the same problem set and are consequently usually associated with parameter uncertainty intervals for point estimates and forecasts [3]. Depending on the problem at hand and the tools available to solve it, it can be practical to convert purely deterministic problems into non-deterministic problems by introducing stochastic variables [4]. Studies have been done looking at how predictive results vary based on what approach is used between a deterministic and a non-deterministic method for a same problem set [5, 6, 7, 8].

1.2 Artificial Intelligence in Aircraft Performance Modeling

Since the late 1990's, Artificial Intelligence (AI) has re-emerged as a popular field in the research community due to three factors [9]: 1) the widespread availability of data, 2) overcoming major hardware limitations enabling faster processing power and 3) resolving obstacles in the mathematical principles used in AI. Artificial intelligence applications have demonstrated the capability of dealing with complex problem sets in fields such as computer vision, natural language processing, game theory, robotics, control theory and machine learning, among others. Broadly speaking, these successes can be attributed to the capacity for Artificial Neural Networks (ANN), a specific type of AI, to deal with highly complex deterministic models with very large data sets.

The current work evaluates the potential of ANNs for modeling and prediction in the field of aircraft performance. Aircraft performance is an engineering discipline concerned with the analysis of the operational capabilities of aircraft with respect to specific performance maneuvers while satisfying certification requirements across the full flight envelope. It is a highly multidisciplinary field which integrates deterministic models from other fields (aerodynamics, structural, thermal, engine performance, icing, etc.) into comprehensive estimation and modeling tools in order to predict aircraft behavior. The development of these tools can be bounded by the deterministic model limitations listed in the previous section (particularly considering the non-linear nature of some the aircraft performance models), the very large amounts of data to contend with, the high dependency on expert knowledge in developing the models, and the possible model approximations resulting from the integration of models from other fields. The body of literature combining aircraft performance, and aerospace design in general, and artificial intelligence is still relatively small and this thesis investigates the relevance of developing ANNs for aircraft performance modeling.

1.3 Research Objective and Approach

The objective of this research project is to determine if ANNs can be used effectively as an alternative to current aircraft performance models. To achieve this objective, the following research approach is used:

1. A takeoff flight phase is selected as the relevant case study.
2. The advantages and disadvantages of existing deterministic models used in aircraft performance for this case study are investigated.
3. A methodology for developing ANNs for aircraft performance purposes is developed.
4. The methodology is applied to a dataset built using existing *deterministic takeoff models*.
5. The methodology is applied to a dataset built using *non-deterministic flight data*.
6. Both dataset results are compared and the practicality of using ANNs for predicting takeoff performance is assessed.

1.4 Scope

This thesis begins with a review of the relevant scientific literature in Chapter 2, with a focus on previous and current work in the field of machine learning applied to aerospace problems. The selected aircraft performance case study is described in Chapter 3. The methodology that was selected as a result of the literature survey is then described in Chapter 4. The results of the study are presented in Chapter 5, and concentrate on highlighting model efficiency differences between an existing deterministic model and the neural network model developed as a result of the research for aircraft performance forecasting. The analysis of these results will allow the objective formulation of an answer to the research question in the discussion of Chapter 6.

2. Literature Review

This chapter presents a review of the relevant scientific literature, with a focus on past and current work in the field of AI applied to aerospace problems. The literature review starts with general background knowledge on the field of AI in Section 2.1. Section 2.2 investigates research in AI specifically applied to aerospace applications. Conclusions drawn from the literature survey are covered in Section 2.3.

2.1 Overview of the Field of Artificial Intelligence

This section introduces relevant definitions and background information for non-experts in the field of AI, including a summary of the inception and historical progression of AI and machine learning.

2.1.1 Background and Definitions

Artificial Intelligence

Artificial intelligence can be broadly defined as a subfield of computer science focused on understanding and developing machines that are able to use some degree of intelligence and reasoning to attain a predefined goal. Machines that make use of AI are termed "intelligent agents". Russell and Norvig [10] divide common definitions of AI along two dimensions: definitions that measure an intelligent agent's success in terms of fidelity to *human performance* and definitions that measure success against an ideal performance measure, called *rationality*. The authors further divide each of these dimensions with definitions concerned with *thought processes and reasoning* and definitions concerned with *behavior, as in how to act or perform an action*. The field of AI has evolved to become extremely interdisciplinary, with roots originating in computer science, mathematics and information processing. Russell and Norvig explain that different scientific disciplines making use of AI focus on solving one of the four dimensions mentioned, which leads to different philosophies and approaches to developing AI solutions. For instance, the *cognitive modeling approach* is an approach which attempts to mimic human thought processes and reasoning. Allen Newell and Herbert Simon [11, 12], pioneers in the field of cognitive science, were concerned with comparing the thought processes of intelligent agents to that of human subjects solving the same problems, using computer science and experimental techniques from psychology to construct precise and testable theories of the human mind. Another approach to AI, which is of greater interest for the research that is the subject of this thesis, is the *rational agent approach*, which adopts the definitions of AI looking at *rationality* and *behavior*: How can we define an environment where an intelligent agent can perform actions in order to attain a mathematically defined ideal?

The rational agent is selected as the predominant approach for this research as it presents the following advantages over the other approaches:

1. It is not always correct to assume that the best performance metric is to compare an intelligent agent with human performance. The rationality approach may get inspiration from human performance in order to define its rational objective, but it does not assume that human-level performance is the ultimate ideal and is open to potentially better results than are humanly possible.
2. For some real-life scenarios in which intelligent agents may find themselves, there are cases where there exists no perfect thought process leading to a provably favorable rationality goal, but an action must still be taken. The rational agent approach is more practical than the "*thought processes and reasoning*" approaches as it is also trying to find the best thought process that leads to the best action, but only if this best thought process leads closer to the best rational goal.
3. Because the action leading to the rationality goal is mathematically defined, it is more amenable to scientific progress and improvement than studying human behaviour or thought. Furthermore, it can be tested to verify that the agents provably achieve their objectives.

Figure 1 shows an organizational chart of different AI disciplines based on end objective. The chart illustrates the multidisciplinary nature of AI applications.

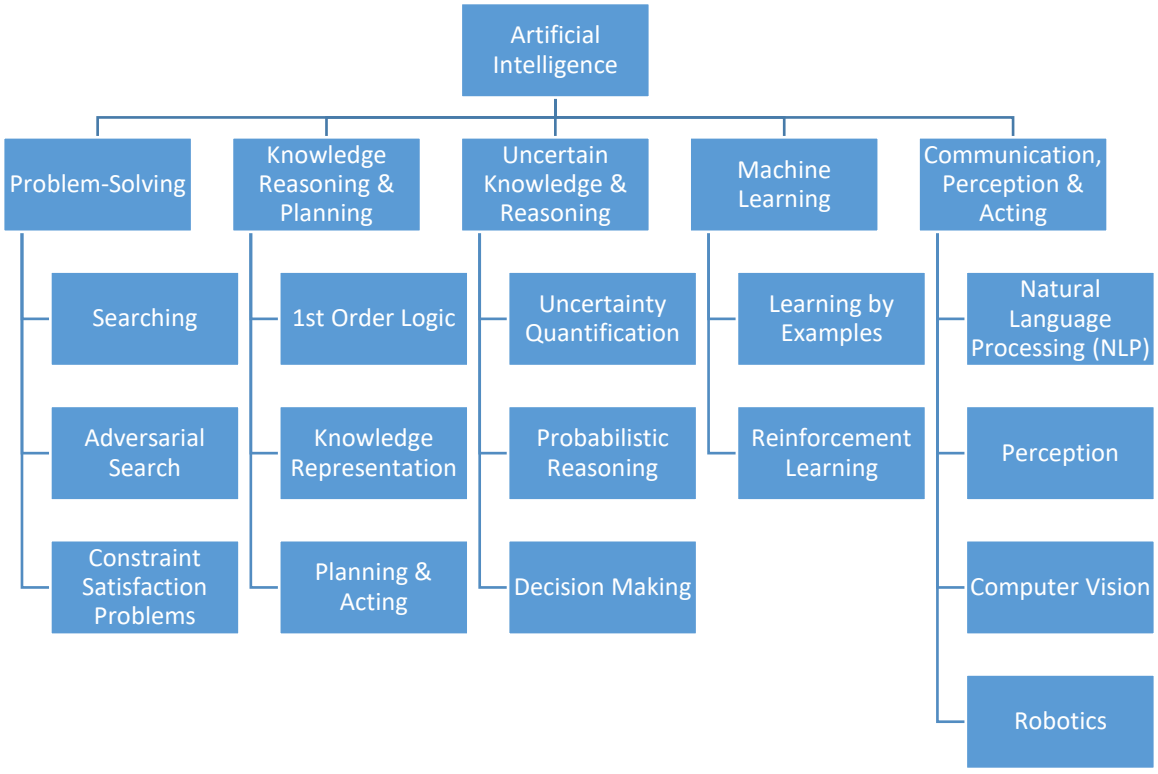


Figure 1 – Summary of some of the subfields of AI by objective

Machine Learning and Deep Learning

Machine learning (ML) is a subset of artificial intelligence which leverages fundamental concepts taken from human physiology and applies them to machine operation. These biological concepts enable machines to essentially "learn from experience". As a general rule, the more data they are presented with, the more they are able to extract complex patterns and, subsequently, develop elaborate knowledge bases [13]. Goodfellow et al. [9] describe ML as a concept where computers are able to learn from experience by understanding the world in terms of a hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts. This allows the machine to learn very high complexity concepts based on the hierarchical relationships they have with a set of simpler concepts. The main advantage of the ML methodology when compared to knowledge-dependent approaches like expert systems, for example, is that by gathering knowledge from computer experience, ML avoids the need for human operators to formally specify the knowledge that the computer needs, thus reducing the expert knowledge required from the designer.

Figure 2 provides a more detailed breakdown of Figure 1, with a focus on the machine learning subdisciplines. ANNs are seen as a type of machine learning that learns from examples in a supervised, unsupervised or semi-supervised manner, based on the desired application.

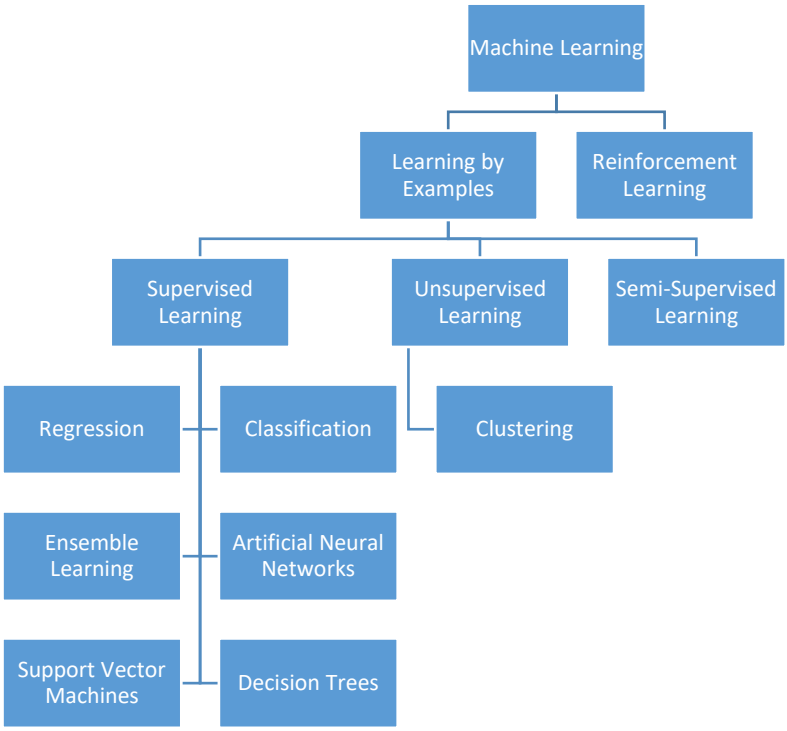


Figure 2 – Summary of the subfields of machine learning

Figure 3 shows the evolution of ML programs. The first column shows the original rule-based systems which must be hand designed. The second column represents a classical ML program, where the designer specifies features for which the computer is able to do feature learning (shaded boxes show programs that are able to learn from data). The designer workload is still significant, however, because they must specify each feature individually in order for the learning to take place. The third column depicts a representation learning program which is able to map links between features without having the designer explicitly specify each feature. This approach is characterized by a reduced workload for the designer, but may not always yield accurate results. The final column shows a Deep Learning (DL) program. DL programs use a more elaborate hierarchy of concept maps (which are said to be "deep"), enabling them to extract more complex patterns from simpler features. Most recent successes in AI can be directly attributed to advances made in DL applications.

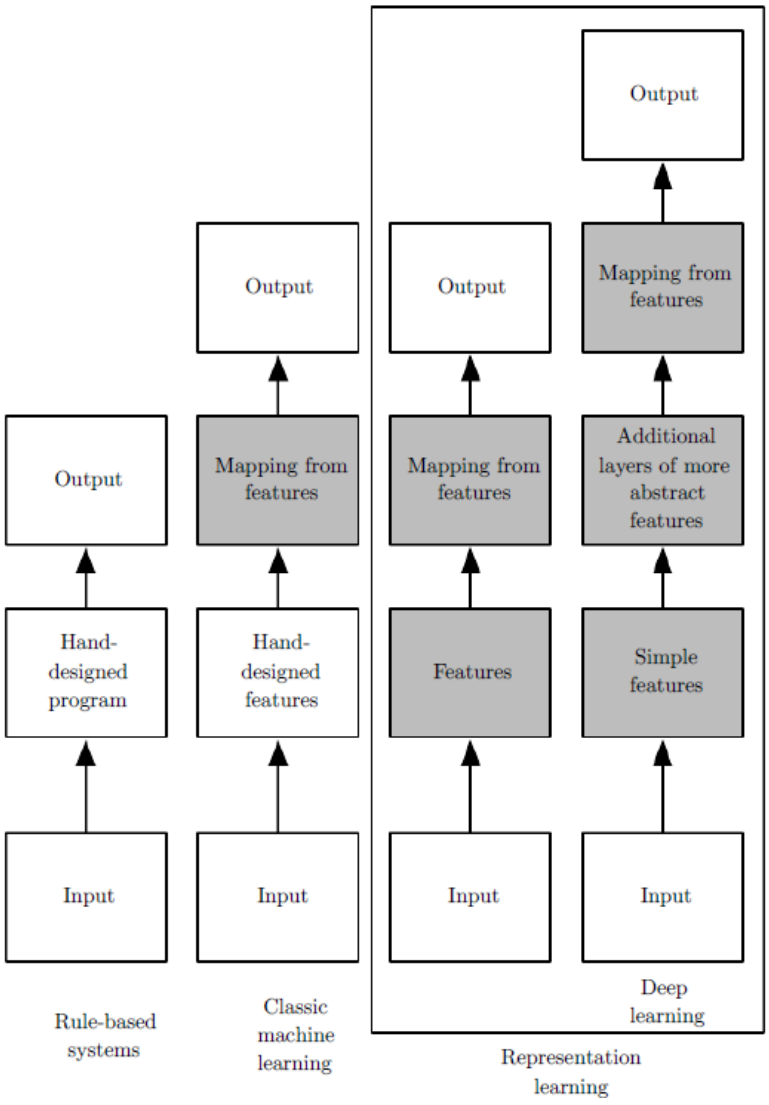


Figure 3 – How deep learning differs from classical machine learning [9]

Artificial Neural Networks

An artificial neural network (ANN) is a type of machine learning which is inspired by the human nervous system. In a biological neuron, part of the nervous system, the dendrites receive an input from other neurons, the cell body provides a decision based on the input, the axon translates the decision into the appropriate output format, and the axon terminal transmits the output to the next neuron as depicted in Figure 4A. Figure 4B shows the mathematical representation of the biological neuron, which can be used for the ANN. Figure 4C illustrates the synapse between neurons, which allows the simultaneous effect of multiple outputs to be received by other neurons as input. This phenomenon is replicated in Figure 4D, with the nodes of an ANN being interconnected into a structured network. These notions allow ANNs to be very capable in developing patterns for highly complex non-linear generalization systems, where a very large number of parameters are under study [9, 13]. Tasks where ML methodologies can be particularly useful include classification; classification with missing inputs; regression; transcription; machine translation; structured output; anomaly detection; synthesis and sampling; imputation of missing values; denoising; and density estimation or probability mass function estimation.

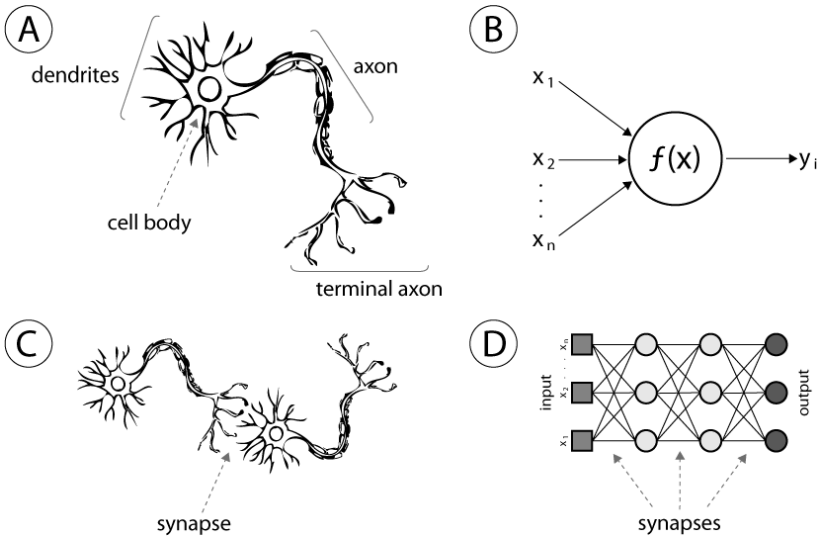


Figure 4 – Similarities between biological and artificial neural networks [14]

2.1.2 Inception of the Field of AI and its Major Historical Periods

One of the first works that led to the creation of the field of AI was written by Warren McCulloch and Walter Pitts in 1943 [15]. McCulloch and Pitts combined three concepts to form the basis for their theory: knowledge of the basic physiology and function of neurons in the human brain, a formal analysis of propositional logic from Russell and Whitehead [16], and Turing's theory of computation [17]. They

showed, for example, that any computable function could be calculated by some network of connected neurons, and that all the logical functions (AND, OR, NOT, IF, etc.) could be implemented by simple net structures. McCulloch and Pitts also suggested that suitably defined networks could learn. Donald Hebb (1949) [18] demonstrated a simple updating rule for modifying the connection strengths between neurons. In 1956 John McCarthy convinced a group of U.S. researchers from different universities to attend a two-month workshop at Dartmouth University, where they discussed artificial intelligence (this was the first use of the term "artificial intelligence"). The proposed study resulting from the workshop was the following:

"*The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together.*" (McCarthy et al., 1955)

Two researchers present at the meeting from Carnegie Tech, Allen Newell and Herbert Simon, demonstrated a reasoning program, the Logic Theorist (LT), which was able to prove most of the theorems found in Russell and Whitehead's *Principia Mathematica* [16]. The Dartmouth workshop did not lead to any new breakthroughs, but it did introduce all the major figures to each other and start the collaboration between them. For the next 20 years, the field would be dominated by these researchers.

The field of AI has matured significantly since its inception. Figure 5 shows a timeline of the major historical periods in the field of artificial intelligence. In the early 1960s there was much interest in developing intelligent systems and many promises were made concerning future capabilities of such systems (i.e. off the shelf generalization solutions). By the early 1970s, many of those promises had not been met, mainly due to limitations in hardware and access to data and AI research suffered. This period is referred as the "1st AI winter". From 1980 to 1987, the field experienced a renewed scientific interest as a result of the creation of Expert Systems (ES). ES are computer-based systems designed to emulate the problem-solving behavior of a human that is an expert on a specific topic. They are designed by first capturing the domain expert's knowledge and translating it in a format that can be understood by a computer program; and then by having the computer program use a reasoning logic to act upon this knowledge. For procedures that are well defined and already known, ES have been demonstrated to be accurate, but are time-consuming to develop and if situations arise were the systems operational envelope goes outside its prescribed procedures, the ES can no longer be trusted to yield accurate results. The AI industry boomed from a few million dollars in 1980 to billions of dollars by 1987 [10], and included hundreds of companies building expert systems, machine vision systems, robotics applications, and other specialized software and

hardware. Soon after came a period called the "2nd AI Winter," during which many companies failed to deliver on their promises and where the limitations of the ES approach were slowing down scientific progress. In the late 1980s, the back-propagation learning algorithm, which is a fundamental part of modern day ANNs, was reinvented into a more practical version which allowed the algorithm to be applied to many learning problems in computer science and psychology. Combined with Parallel Distributed Processing [19], the 1993-2011 period was marked by a re-emergence of AI applications. As of 2011, it was shown that with the availability of much bigger datasets it is possible to solve previously unsolvable problems including filtering images from the internet [20] and word-sense disambiguation used in translation research [21].

Figure 5 – Timeline of the major historical periods in the field of artificial intelligence

2.2 Artificial Intelligence in Aerospace

Given the main components and structure of AI systems, their evolution and inception, and their existing application areas, it is observed that many potential avenues can be taken when undertaking the development of an AI system. This section will review the avenues that have been chosen by practitioners in the aerospace sector as a means of implementing AI in existing systems and how these research areas have evolved with the advancements in the field of AI within the aerospace context.

2.2.1 Early Skepticism (1960 – 1980)

NASA began investigating uses for artificial intelligence in aerospace applications as early as 1965 [22]. The agency was mainly interested in looking at advances in the field of AI (pre-1965) and determining if applications could be found for the new technology specifically for the NASA objectives of the time, which were to design better control systems and gather and analyze data more efficiently and autonomously. An example of this is the research done on the F-8 Crusader aircraft, which was modified as a digital fly-by-wire testbed [23]. The aircraft used an adaptive control system that would iteratively adjust its model parameters in order to produce results in line with model outputs coming from onboard sensors.

Due to the unpredictable nature of neural networks and the field of AI still being in its infancy, the adoption of existing technologies for the aerospace sector was severely limited. Since innovation in the aerospace industry is subject to regulatory and safety assessment restrictions, novel technology adoption is typically done after proper certification processes and acceptable safety assessment plans are present, which was not

the case in the 1960's and 1970's. This led to most of the early aerospace-related adoption of NNs being for use in military fighter aircraft, where mission objectives are typically prioritized and civilian certification requirements are not applicable.

2.2.2 Appearance of Neural Network Research in Aerospace (1980 – 2000)

A notable overview of the field of neural networks and its possible applications for the aerospace sector is the 1989 Neural Network Study Report [24], sponsored by the Tactical Technology Office of the U.S. Defense Advanced Research Projects Agency (DARPA/TTO). The report was developed with the involvement of government, industry, and academic participants. The goals of the study were; to identify potential applications for neural networks in Department of Defense (DoD) systems, to determine the current neural network technology base; to identify technology requirements; and to identify a DoD program plan for the next five years. Some of the study's conclusions are summarized below:

1. The real strength of neural networks as a new form of computational approach comes from their ability to **self-adapt and learn from data-driven models**, in time showing potential for reduced need for application specific software.
2. Thanks to the development of advanced mathematical theories, new computer tools, and to a better understanding of neurobiology, **neural network research has matured greatly** since the perceptron of the 1950s and **recommends the scientific community divest more resources towards the study of NNs in aerospace applications**.
3. The **variety of problems addressed by neural networks is large**. Significant demonstrations of neural network capabilities in vision, speech, signal processing, and robotics were listed.
4. **Hardware capabilities are limiting the development** of important neural network applications.

In 1994, the Federal Aviation Administration (FAA) published research [25] reviewing and discussing issues related to the use of AI in aerospace technology. The research focused on three fields of AI: Expert systems, fuzzy logic, and neural networks. It is explained that ES and NNs on their own still have drawbacks (certification and functional) that are keeping them from being used more broadly in aerospace but, that when used together as integrated ES-NN systems (or fuzzy-NN systems), would generate considerable potential. The goal of integrating these two methods is to extract features in complex pattern recognition using NNs and using them in ES for reasoning, in time speeding up the process of developing ES and creating a system that can deal with more generalized problems than a single NN could solve. The FAA recommends use of such integrated systems in particular for space exploration, nuclear power, and military aerospace.

Around the mid-1990s, NN research in aerospace appeared in various domains including regression optimization [26, 27], aerodynamics modeling [28], adaptive control systems [29, 30], fault diagnosis [31] and anomaly detection [24]. This new wave of research was marked by demonstrated implementations of NNs as replacements or enhancements to existing methodologies in these respective aerospace domains. Faller and Schreck [31] present the case that NNs can act as a useful tool for solving real-life non-linear aerospace problems (with some examples), but that they must be combined with existing techniques in order to yield optimal results (for example, validating NN predictions using existing simulation tools).

2.2.3 Proliferation of Neural Network Scientific Interest (2000 – 2010)

In 2002, NASA developed a standard [32] for the verification and validation of neural networks for use in certified of aerospace systems, which resulted in addendums to the DO-178, which deals with guidelines for safety-critical software used in airborne systems. The focus at the time was on certifying adaptive flight control systems that use NNs. This standard provides a good basis for evaluating the proper development of any supervised neural net for use in an aerospace context. NASA has not yet provided a standard for unsupervised training neural nets.

The National Research Council of the National Academy of Sciences releases decadal surveys on major scientific research areas, which include current-state assessments and recommendations for government, academic and industrial entities. The latest aerospace survey, the 2006 Decadal Survey of Civil Aeronautics: Foundation for the Future [33], highlighted major areas of research that would most benefit civil aeronautics. Intelligent systems, of which NNs are a part, were identified as a recurring theme across *all* of the major areas of research.

Following the 2006 decadal survey, a proliferation of NN research began across a wide variety of aerospace applications. This was due to many factors including the increased availability of large datasets, new development tools enabling easier NN development, parallel computing, and better optimization algorithms. Advances in adaptive control systems included those able to compensate for system uncertainties [34], able to adapt to changes in flight conditions [35], possessing fault-tolerant abilities [36], and providing faster online training capabilities [36]. Advances in non-linear airflow analysis tools included systems capable of furthering the understanding of ice accretion models [37], unsteady aerodynamic models coupling multiple non-linear aerodynamics models (i.e. for aircraft in dynamic ground effect) [38], buffet pressure predictions [39], and the prediction of maneuver loads [40]. Advances in anomaly detection research included pattern recognition algorithms to identify anomalies for health management of aircraft gas turbine engines [41]. Advances in regression optimization of non-linear systems included rotorcraft vibration modeling for real-time applications [42], predicting aircraft cruise performance solely based on

available flight data [43], and simplified methods to estimate aircraft fuel consumption coupled with fast-time airspace simulation models like SIMMOD, TAAM (Total Airspace and Airport Model) or RAMS (Reorganized Analytical Modeling Systems) [44].

2.2.4 Appearance of Industry-Proven Implementations (2010 – Present)

Between 2014 and 2016, the National Research Council [45], the FAA [46] and NASA [47] released reports paving the way for developing methodologies for the verification of NNs in aerospace software. The goal of this research was to conduct a preliminary examination of what is necessary to provide sufficient assurance that an adaptive system is safely used in an aircraft product from a software perspective. These efforts resulted in recommendations for modifying the ARP-4754A, ARP-4761 and DO-178C guidelines and their supplements.

Since 2010, a new resurgence of NN research can be observed in the aerospace community, this time due to the arrival of big data, new development tools enabling non-experts to develop NNs with greater ease, even faster hardware allowing shorter computational times, and new and improved optimization algorithms. Advances in non-linear airflow analysis includes machine learning tools applied to hasten the numerical prediction of ice formation on local portions of aircraft flying in hazardous weather conditions for broad flight envelopes [48], novel approaches for developing real-time in-flight ice detection systems using computational aeroacoustics, Bayesian neural networks, and other Deep Neural Networks (i.e. Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN)) [49, 50], and the detection of airframe icing in a manner allowing system recommendations for reconfigurable control protecting aircraft from hazardous icing conditions [51]. Much research was also done in these years on anomaly detection or fault detection and diagnosis for aircraft: RNNs and Long Term Short Term Memory (LTSM) networks through semi-supervised or unsupervised learning are able to solve problems encountered by previous technology [52], used as tools for optimization of non-destructive testing [53], and engine fault detection using modified vision NN algorithms (CNN) (i.e. fatigue crack damage identification) [54]. Other advances in non-linear regression optimization research include the use of data fusion for airspace management and operations prediction tools [55] and assessing recommendations for reroutes [56, 57]. Some progress has also been made in the field of surrogate model generation using CNNs [58].

In 2019, Aero Montreal released a report [59] on the current-state of AI research and adoption in the aerospace industry. It is explained that Montreal, being the third largest aerospace hub by number of jobs in the world and at the same time being the largest AI hub, is the prime location for a more widespread adoption in the aerospace field, which will come with much resulting scientific progress. It is also stated

that such innovation can only happen with a high level of collaboration between the practitioners across the two sectors of aerospace and AI.

2.3 Gaps in the Literature

Major research areas emerging in NN-aerospace research including non-linear regression optimization, advanced airflow analysis, anomaly detection and health monitoring, adaptive control systems, and surrogate model generation. The survey of the literature that was undertaken as part of the study that is the subject of this thesis produces the following observations:

1. Non-linear regression optimization stands out as being the **most researched topic** with the **most demonstrated benefits** resulting from the use of NNs. Published research on the topic shows that, for highly non-linear problem sets where deterministic relationships are difficult to develop, NNs are generally able to solve the problems when presented with enough data.
2. Various NN types have been applied to a number of aerospace applications with varying levels of success. Recent research shows that certain types of NNs are better suited for very specific problem sets, and research in this area is ongoing. As a qualitative observation, **traditional feedforward backpropagation neural networks with two or more hidden layers have demonstrated the greatest potential at solving the widest variety of types of problem sets**, especially for non-linear regression.
3. **Dataset sizes have increased significantly through the years, resulting in demonstrated modeling prediction improvements**. This is a general observation and some cases remain where more data does not necessarily mean better model predictive accuracy, notably in cases dealing with very noisy data or with data populated with a large number of outliers.
4. **Development tools have evolved greatly** with respect to: 1) ease of use and practicality, 2) facility in integrating with existing systems (hardware and software).
5. Of all the literature found on NN applications in aerospace, **literature focused on aircraft performance is sparse** and is mostly concentrated on route [56], fuel [26, 44] or range [57, 60] optimization.

3. The Selected Aircraft Performance Case Study

For the purposes of the current research, the modeling of takeoff distance (TOD) is selected as a case study to determine if ANNs can be used to replace existing deterministic models. This chapter begins with the definition of the TOD used in this work, followed by a review of how current deterministic TOD models are being developed without the use of ANNs. The chapter concludes with an assessment of why this case study is relevant to the research topic and was selected as a result of the gaps identified in the literature review chapter.

3.1 Defining Takeoff Distance

Chapter 525 (Transport Category Aeroplanes) of Part V (Airworthiness Manual) of the Canadian Aviation Regulations (CARs) [61] describes the legally recognized definition of the takeoff distance as:

525.113 Take-off Distance and Take-off Run

(a) Take-off distance on a dry runway is the greater of:

(1) The horizontal distance along the take-off path from the start of the take-off to the point at which the aeroplane is 35 feet above the take-off surface, determined under 525.111; for a dry runway; or

(2) 115 percent of the horizontal distance along the take-off path, with all engines operating, from the start of the take-off to the point at which the aeroplane is 35 feet above the take-off surface, as determined by a procedure consistent with 525.111.

(b) Take-off distance on a wet runway is the greater of:

(1) The take-off distance on a dry runway determined in accordance with paragraph (a) of this section; or

(2) The horizontal distance along the take-off path from the start of the take-off to the point at which the aeroplane is 15 feet above the take-off surface, achieved in a manner consistent with the achievement of V_2 before reaching 35 feet above the take-off surface, determined under 525.111 for a wet runway.

(c) If the take-off distance does not include a clearway, the take-off run is equal to the take-off distance. If the take-off distance includes a clearway:

(1) The take-off run on a dry runway is the greater of:

(i) The horizontal distance along the take-off path from the start of the take-off to a point equidistant between the point at which V_{LOF} is reached and the point at which the aeroplane is 35 feet above the take-off surface, as determined under 525.111; for a dry runway; or

(ii) 115 percent of the horizontal distance along the take-off path, with all engines operating, from the start of the take-off to a point equidistant between the point at which V_{LOF} is reached and the point at which the aeroplane is 35 feet above the take-off surface, determined by a procedure consistent with 525.111.

(2) The take-off run on a wet runway is the greater of:

(i) The horizontal distance along the take-off path form the start of the take-off to the point at which the aeroplane is 15 feet above the take-off surface, achieved in a manner consistent with the achievement of V_2 before reaching 35 feet above the take-off surface, as determined under 525.111 for a wet runway; or

(ii) 115 percent of the horizontal distance along the take-off flight path, with all engines operating, from the start of the take-off to a point equidistant between the point at which V_{LOF} is reached and the point at which the aeroplane is 35 feet above the take-off surface, determined by a procedure consistent with 525.111.

From this definition, it is understood that the TOD can be different based on runway condition (dry or wet), if a clearway is present, and the end outcome of the takeoff (OEI, AEO, ASD). The takeoff flight path defined by the CARs standard 525.111, on which the TOD definition is based, is represented in Figure 6. The current research is only concerned with the take-off scenario up to 35 feet above ground level. Four possible scenarios exist: All Engine Operating Takeoff (AEO), One Engine Inoperative Takeoff (OEI), Accelerated Stop with AEO (ASD_{AEO}), and Acclerated Stop with OEI (ASD_{OEI}). The OEI and AEO both result in the aircraft taking off the runway, while the ASD_{AEO} and ASD_{OEI} both result in a rejected takeoff. Figure 7A and Figure 7B show representations of TOD for OEI and AEO scenarios under wet and dry conditions, denoted as TOD_{N-1} ad TOD_N respectively. Figure 7C and Figure 7D show the same scenarios if clearways are present. A clearway is an area beyond the paved runway, free of obstructions and under the control of the airport authorities. The length of the clearway may be included in the length of the takeoff distance available. ASD_{AEO} and ASD_{OEI} distances are represented in Figure 8A and Figure 8B for rejected takeoffs, denoted as ASD_{N-1} ad ASD_N respectively. Standards CAR 525.109 and 525.113 define the certified takeoff distances as functions of the distances shown in Figure 7 and Figure 8. Table 1 summarizes their definitions.

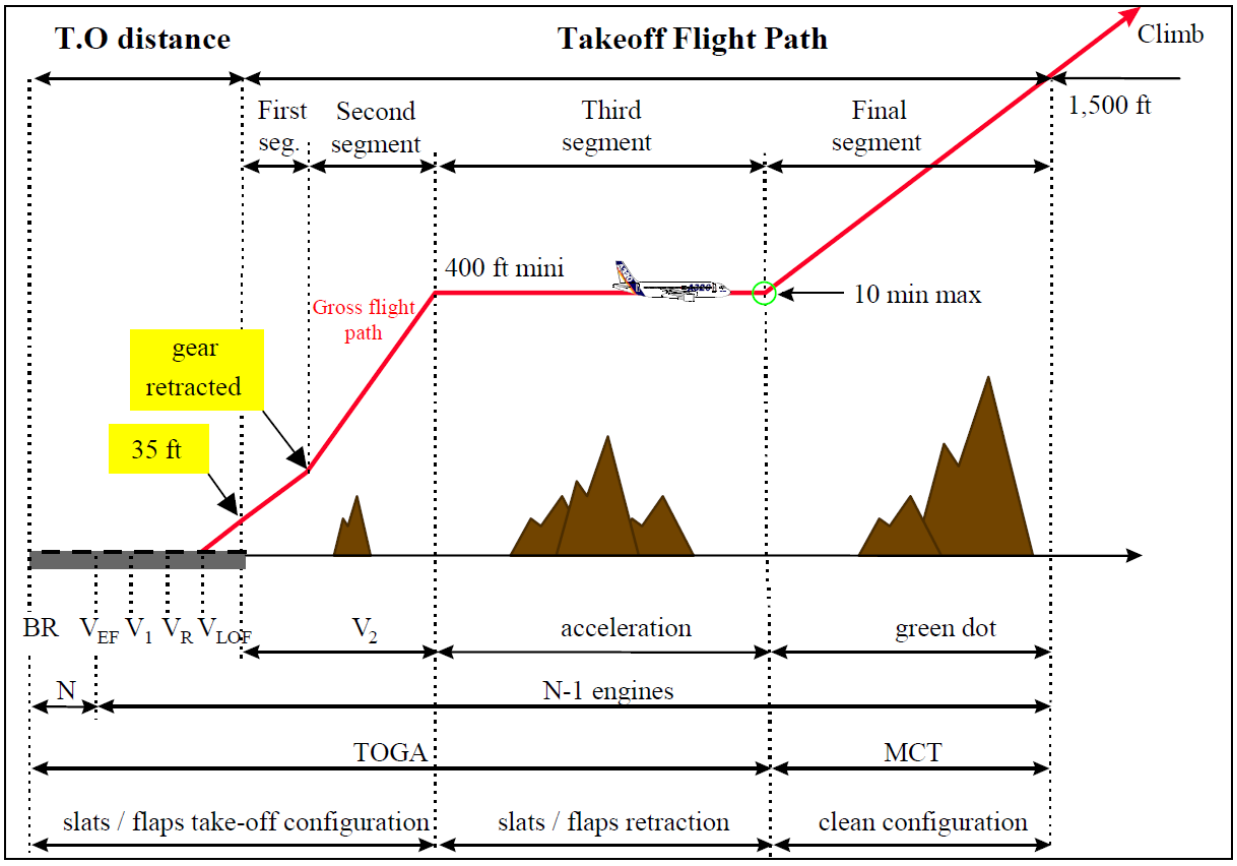


Figure 6 – Illustration of the takeoff flight path [62]

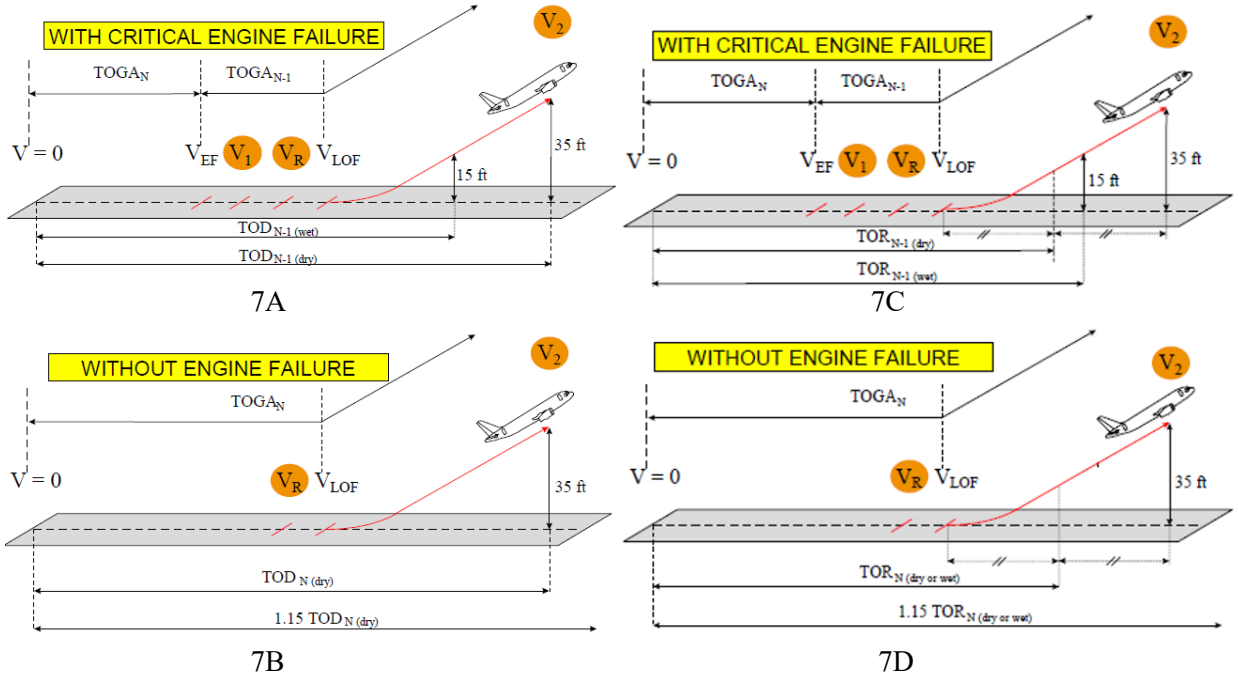


Figure 7 – Representation of the TOD scenario for OEI and AEO [62]

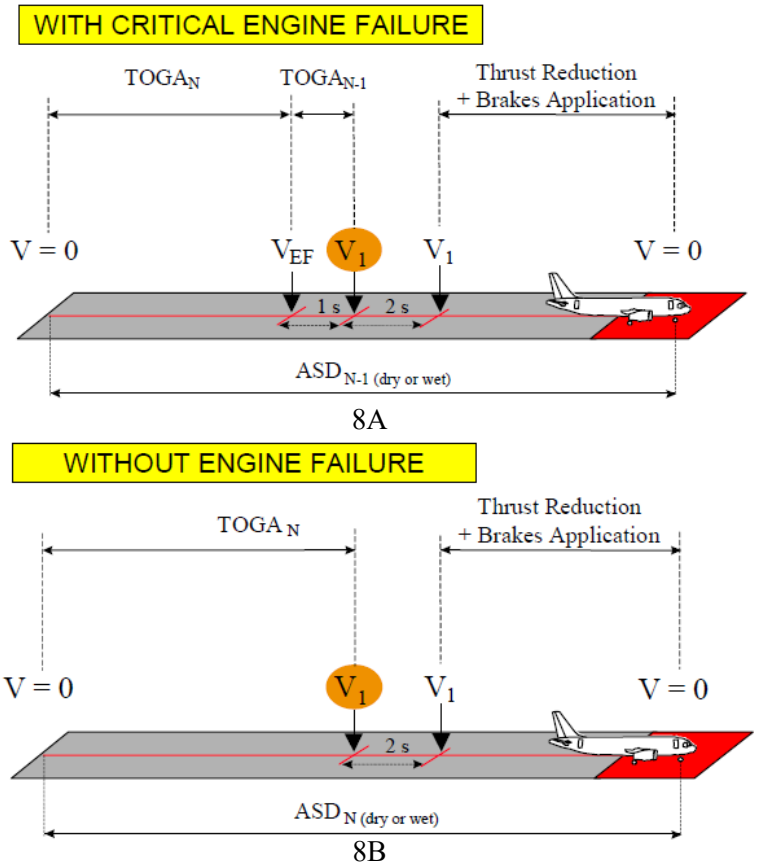


Figure 8 – Representation of the TOD scenario for ASD_{AEO} and ASD_{OEI} [62]

Table 1 – Takeoff distance definitions

Clearway	TOD Definitions	Regulation
No Clearway	$TOD_{dry} = \max(TOD_{N-1,dry}, 1.15TOD_{N,dry})$	CAR 525.113 / FAR 25.113 / CS 25.113
	$TOD_{wet} = \max(TOD_{dry}, TOD_{N-1,wet})$	
	$ASD_{dry} = \max(ASD_{N-1,dry}, ASD_{N,dry})$	CAR 525.109 / FAR 25.109 / CS 25.109
	$ASD_{wet} = \max(ASD_{dry}, ASD_{N-1,wet}, ASD_{N,wet})$	
With Clearway	$TOR_{dry} = \max(TOR_{N-1,dry}, 1.15TOR_{N,dry})$	CAR 525.113 / FAR 25.113 / CS 25.113
	$TOR_{wet} = \max(TOR_{N-1,wet}, 1.15TOR_{N-1,wet})$	

The different speeds encountered during takeoff (illustrated in Figure 7 and Figure 8) are defined in Table 2 and their limitations prescribed by regulations are summarized in Table 3.

Table 2 – Takeoff speed definitions

Takeoff Speed	Definition	Regulation
V_{EF}	V_{EF} is the calibrated airspeed at which the critical engine is assumed to fail. V_{EF} must be selected by the applicant, but may not be less than V_{MCG}.	CAR 525.107 FAR 25.107 CS 25.107
V_1	V1 is the maximum speed at which the crew can decide to reject the takeoff, and is ensured to stop the aircraft within the limits of the runway. The time between V_{EF} and V_1 is recognised as 1 second.	
V_R	V_R is the speed at which the pilot initiates the rotation, at the appropriate rate of about 3° per second.	
V_{LOF}	V_{LOF} is the calibrated airspeed at which the aeroplane first becomes airborne. Therefore, it is the speed at which the lift overcomes the weight.	
V_2	V_2 is the minimum climb speed that must be reached at a height of 35 feet above the runway surface, in case of an engine failure.	
V_{MBE}	Tire maximum absorption capacity speed during an extreme braking operation.	CAR 525.109 FAR 25.109 CS 25.109
V_{TIRE}	Maximum ground speed limited by tire centrifugal forces and heat elevation.	

Table 3 – Takeoff speed limitations

Limited Speed	Limitation		Regulation
V_{EF}	$V_{EF} \geq V_{MCG}$		CAR 525.107 / FAR 25.107 / CS 25.107
V_1	$V_{MCG} \leq V_{EF} \leq V_1$		CAR 525.107 / FAR 25.107 / CS 25.107
	$V_1 \leq V_{MBE}$		CAR 525.109 / FAR 25.109 / CS 25.109
V_R	$V_R \geq 1.05 V_{MCA}$		CAR 525.107 / FAR 25.107 / CS 25.107
V_{LOF}	Geometric	$V_{LOF} \geq \mathbf{1.05}\, V_{MU\,(N-1)}$ $V_{LOF} \geq 1.08\, V_{MU\,(N)}$	CAR 525.107/FAR 25.107/AC 25-7A
		$V_{LOF} \geq \mathbf{1.04}\, V_{MU\,(N-1)}$ $V_{LOF} \geq 1.08\, V_{MU\,(N)}$	CS 25.107
	Aerodynamic	$V_{LOF} \geq 1.05\, V_{MU\,(N-1)}$ $V_{LOF} \geq 1.10\, V_{MU\,(N)}$	CAR 525.107/FAR 25.107/CS 25.107
	Tire	$V_{LOF} \leq V_{TIRE}$	CAR 525.109/FAR 25.109/CS 25.109
V_2	$V_2 \geq 1.1\, V_{MCA}$		CAR 525.107 / FAR 25.107 / CS 25.107

3.2 The Calculation of Takeoff Distance

This section describes a general process for developing the deterministic model used to calculate takeoff distance. The detailed process for calculating takeoff distance was provided by an industry partner and contains proprietary data, which is not included as part of this thesis. The theoretical equations for takeoff distances can be found in most aircraft performance textbooks [63, 64, 65, 66]. The complex models used by OEMs (Original Equipment Manufacturers), although based on these equations, are modified and fitted with proprietary flight test and other data to account for phenomena not addressed in the more simplified theory. The calculation of TOD, as described in the referenced textbooks, is broken down in different segments that must be added together to give the final distance, as shown in Figure 9.

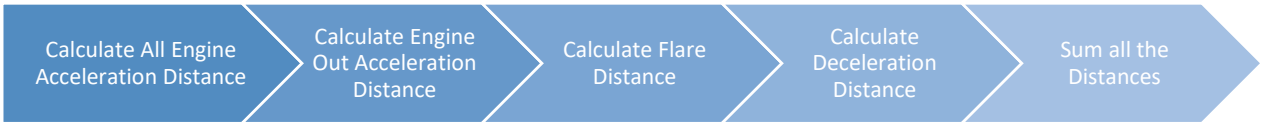


Figure 9 – TOD calculation process

Calculating the All Engine Acceleration Distance

The all engine acceleration distance extends from brake release to the point where rotation velocity, V_R, is achieved. A free body diagram of the forces acting on an aircraft during takeoff is shown in Figure 10. As the forces acting on the aircraft vary along the takeoff run, a step integration process can be used to calculate the acceleration changes, which can then be used to calculate the total distance traveled.

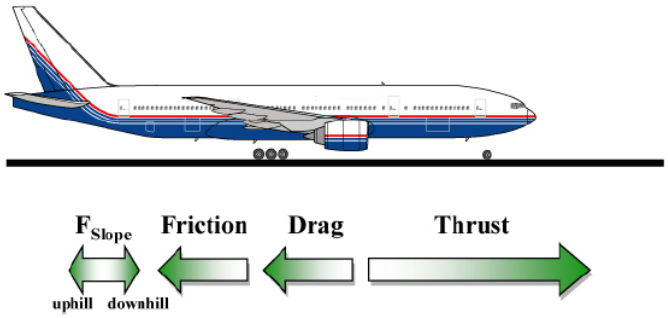


Figure 10 – Forces acting on an aircraft during takeoff [67]

Equation 1 shows the sum of the forces acting along the x-axis in the free body diagram.

$$\sum forces = T - D - \mu(W - L) - Wsin\varphi \qquad (1)\ [63, 64, 65, 66]$$

Where T is the thrust generated by the engines, D is the drag, μ is the runway rolling coefficient of friction, W is the aircraft weight at the start of the takeoff run, L is the lift generated by the lifting surfaces, and φ is the runway slope. Acceleration along the takeoff run is calculated using equation 2, where mass is expressed as aircraft weight over the Earth's gravity constant g.

$$a = \frac{\sum Forces}{Mass} = \frac{g}{W}[T - D - \mu(W - L) - Wsin\varphi] \qquad (2)\ [63, 64, 65, 66]$$

Equation 2 can be expressed in terms of its aerodynamic coefficients, as shown in equation 3.

$$a = \frac{g}{W}[T - \mu W - (C_D - \mu C_L)qS - W \sin\varphi] \qquad (3)\ [63, 64, 65, 66]$$

Where C_D is the drag coefficient, C_L is the lift coefficient, q is the dynamic pressure and S is the reference wing area. The dynamic pressure is obtained from equation 4.

$$q = 0.5\rho V^2 \qquad (4)\ [64]$$

The acceleration at different moments along the takeoff run (from V_0 to V_R) can be calculated using equations 3 and 4. The average velocity over a small change in velocity $\overline{V}$ is defined as:

$$\overline{V} = \frac{\Delta s}{\Delta t} \qquad (5)\ [64]$$

Where Δs is the incremental distance over the speed increment and Δt is the incremental time over the velocity increment. Change in velocity is related to acceleration by:

$$a = \frac{\Delta V}{\Delta t} \qquad (6)\ [64]$$

Where a is the acceleration and ΔV is the speed increment. The incremental distance between two points along the takeoff run can be obtained by combining equations 5 and 6.

$$\Delta s = \frac{\overline{V}\Delta V}{a} = \frac{\overline{V}\Delta V}{\frac{g}{W}[T - \mu W - (C_D - \mu C_L)(0.5\rho V^2)S - W\sin\varphi]} \quad (7)\ [64, 65]$$

The all engine acceleration distance is calculated as the sum of all the incremental distances Δs (equation 7) between V_0 and V_R. The distance can be obtained more accurately using integration of equation 7 as a function of speed.

Calculating the Engine Out Acceleration Distance

The engine out acceleration distance extends from engine failure (V_{EF}) to the rotation speed V_R. Right after the engine failure occurs, the engine progressively loses thrust, which affects the calculated distance. This is termed the engine spindown. The spindown factor is the ratio between the actual thrust produced by the engines during spindown divided by the thrust setting of the engine selected by the pilot in the cockpit. Performance charts showing the ratio of residual engine thrust over takeoff thrust and time from engine failure are used to determine the remaining thrust at any time from the engine failure event. The spindown factor is multiplied to the thrust values of equation 7 to get the actual thrust during engine failure. In order to use the proper value of engine spindown factor, which is a function of time, a step integration of the distance calculation from equation 7 must be undertaken with respect to time instead of speed. Using equation 7 with a spindown factor of 1 and the velocity at t = 0 (V_1), a first initial guess of acceleration can be found, which corresponds to the instantaneous acceleration. The speed after 1 second is found using the initial guess of acceleration. The instantaneous acceleration at 1 second is then calculated using equation 3. Having obtained the first two instantaneous accelerations, the first average acceleration at 1 second can be calculated. The new value of speed at 1 second can be recalculated using the average acceleration, and this process can be repeated until the value of airspeed converges to the final value.

Calculating the Flare Distance

The flare distance extends from initiation of rotation (V_R) to a height of 35 feet above ground level. For the AEO scenario the height of 35 feet above ground level corresponds to V_{35}, while for the OEI scenario the height of 35 feet above ground level corresponds to V_2 with an engine failed. During flight testing, experimental V_{35} and V_2 airspeed values are obtained for a variety of scenarios and conditions (i.e. for

different thrust to weight ratios). The flare distance for the AEO scenario can be calculated using equation 8 and the flare distance for the OEI scenario can be calculated using equation 9, where $\Delta t_{R\text{-}35}$ is the flare time from rotation to 35 feet.

$$S_{flare,AEO} = \frac{V_R + V_{35}}{2} \Delta t_{R-35} \qquad (8)\ [66]$$

$$S_{flare,OEI} = \frac{V_R + V_2}{2} \Delta t_{R-35} \qquad (9)\ [66]$$

Calculating the Deceleration Distance

The deceleration distance extends from V_1 and ends at the moment when the ground speed is zero. Regulations mandate that a 2 second recognition time must be considered at the V_1 speed. The throttle setting and brake design produce additional forces to consider in this scenario, which affect the value of the braking coefficient of friction μ_B. The deceleration scenario is depicted in the free body diagram of Figure 11.

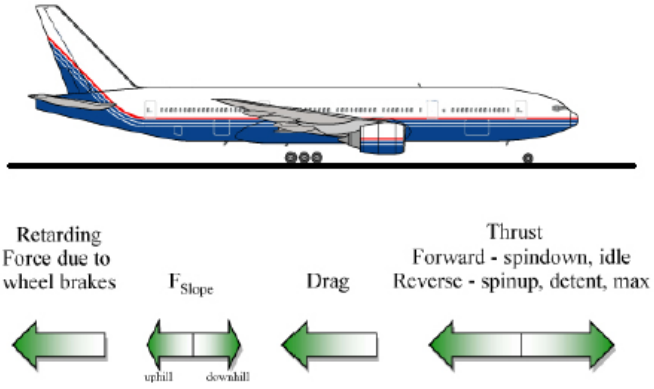


Figure 11 – Forces acting on an aircraft during a takeoff deceleration [67]

Equation 10 shows the sum of the forces acting along the x-axis in the free body diagram.

$$\sum forces = T - D - \mu_B(W - L) - Wsin\varphi \qquad (10)\ [63, 64, 65, 66]$$

Where T is the thrust generated by the engines, D is the drag, μ_B is the braking coefficient of friction, W is the aircraft weight at the start of the takeoff run, L is the lift generated by the lifting surfaces, and φ is the runway slope. Values of μ_B are experimentally obtained during flight tested or provided by the brake manufacturer. Deceleration along the takeoff run is calculated using equation 11, where mass is expressed as aircraft weight over the Earth's gravity constant g.

$$a = \frac{\sum Forces}{Mass} = \frac{g}{W}[T - D - \mu_B(W - L) - Wsin\varphi] \qquad (11)\ [63, 64, 65, 66]$$

Equation 11 can be expressed in terms of its aerodynamic coefficients, as shown in equation 12.

$$a = \frac{g}{W}[T - \mu_B W - (C_D - \mu_B C_L)qS - W\sin\varphi] \qquad (12)\ [63, 64, 65, 66]$$

Where C_D is the drag coefficient, C_L is the lift coefficient, q is the dynamic pressure and S is the reference wing area. The dynamic pressure is obtained from equation 4.

The deceleration from V_1 to $V_{full\ stop}$ can be calculated using equations 12 and 4. The incremental distance between two points along the deceleration segment can be obtained using equation 13.

$$\Delta s = \frac{\overline{V}\Delta V}{a} = \frac{\overline{V}\Delta V}{\frac{g}{W}[T - \mu W - (C_D - \mu_B C_L)(0.5\rho V^2)S - W\sin\varphi]} \quad (13)\ [64, 65]$$

The deceleration distance is calculated as the sum of all the incremental distances Δs (equation 13) between V_1 to $V_{full\ stop}$.

Summation of all of the Takeoff Distances

The final distances for each of the takeoff outcomes are calculated as follows:

$$TOD_{AEO} or\ TOD_N = \text{All Engine Acceleration Distance}$$

$$TOD_{OEI} or\ TOD_{N-1} = \text{All Engine Acceleration Distance} + \text{Engine Out Flare Distance}$$

$$ASD_{AEO} or\ ASD_N = \text{All Engine Acceleration Distance} + \text{Deceleration Distance}$$

$$ASD_{OEI} or\ ASD_{N-1} = \text{Engine Out Acceleration Distance} + \text{Deceleration Distance}$$

3.3 Assessment of the TOD Deterministic Model

The process used by OEMs to develop their proprietary deterministic models is complex because it integrates multiple models from different aerospace disciplines, deals with large amounts of data, requires expert knowledge to be developed, and is dependent on many external factors, which include:

- Airport atmospheric conditions and altitude (temperature, pressure, air density, wind speed and direction)
- Airport runway conditions (runway sediment accumulation, runway slope)
- Aircraft operational capabilities (available thrust, structurally limited takeoff speeds, V_R, C_L, etc.)
- Aircraft takeoff limitations from regulations (V_{MBE}, V_{TIRE})
- Aircraft configuration (ECS on/off, anti-ice on/off, flap setting, engine configuration)
- Weight and balance
- Pilot recognition time (time at V_1)

Considering the gaps identified in the literature and the assessment of the existing TOD deterministic model, the TOD is selected as the case study for this research because it is a high complexity non-linear regression optimization problem which is a function of a diverse set of parameters; requires much expert knowledge to develop; involves the processing of large amounts of data; and requires much development time and effort when done using deterministic models. Based on the literature survey, a traditional feedforward backpropagation neural network with two or more hidden layers will be tested using large datasets considering that this type of model has shown the most promise when applied to this type of problem. Two types of datasets were analyzed: 1) A deterministic dataset generated from an existing aircraft performance takeoff distance model and 2) A non-deterministic dataset consisting of empirical flight data from sensors onboard an aircraft. The selected research approach is explained in detail in Chapter 4.

4. Research Methodology

This chapter details the process followed in order to develop the different neural networks that were used in this study. Section 4.1 presents an overview of the chosen neural network development process. Section 4.2 defines clear requirements and objectives used to properly evaluate the performance of the NNs being developed. Section 4.3 covers the process of selecting and preprocessing the dataset that are used to develop the NNs. Two different selection and preprocessing methods were used for the deterministic and non-deterministic datasets. Section 4.4 details the development of the NN architecture, which includes defining key parameters constituting the foundation of the NN's mathematical structure. Once the architecture is defined, the NN can be trained and tested to evaluate its performance (Section 4.5). Finally, it is shown in Section 4.6 how a trained NN can be used in practical applications.

4.1 The Neural Network Development Process

Figure 12 illustrates the procedure that was used to develop the neural network application. The first step is to define the desired objectives and requirements for the NN. These are used to determine when to stop the optimization of the network. The next step is to analyze the dataset's properties (i.e. dataset distribution, size, file format, etc.) in order to appropriately select the optimal dataset properties. The dataset is then preprocessed to retain only the desired data properties and to facilitate data integration with the Python code environment. Based on the dataset's characteristics, a preliminary architecture can be defined, which is subsequently optimized using a topological study. The network is trained using the training dataset and tested using the testing dataset. If the network objectives are not met, the network architecture must be updated, and this process is iterated until the desired training and testing results are obtained. Once the network requirements are met, the weights matrix of the trained network are saved to a local file, which can then be used by a predictive tool application. Each step will be explained in more detail in the following sections.

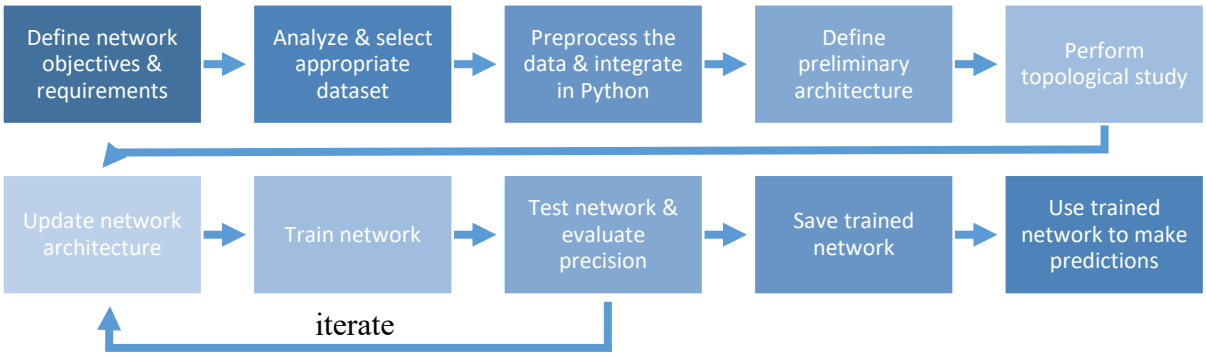


Figure 12 – Neural network development process

4.2 Network Objectives and Requirements Definition

The neural network's objectives and requirements depend on what it will be used for. For the calculation of TOD, data found in aircraft flight manuals (AFM) are typically acceptable when errors are lower or equal to 1 % of the flight test data. For the purposes of this research, the lowest possible error margins are desired. 20 hours have been arbitrarily selected as the maximum training and testing times. The time was chosen based on a conveniently long enough time for the NN to extract complex patterns of acceptable precision for the demonstrative purposes of this research. Since the tool making use of the trained NN could potentially be used onboard aircraft or be part of the development of software versions of AFMs, this tool must be able to compute TOD predictions very fast. It was set as a requirement for that time to be equal to or lower than 1 second. This research is used as a proof of concept, but it would be advantageous to develop it in a way that would also be adaptable to work with other aircraft performance scenarios than TOD (i.e. landing distance predictions). It must also be able to be integrated easily with existing technology (i.e. with a software AFM or with an Electronic Flight Bag). Table 4 summarizes the NN requirements.

Table 4 – Summary of the NN requirements and objectives

Neural Network Parameter	Requirements & Objectives
Maximum training and testing time	20 hours
Maximum predictive tool run time	1 sec
Network re-usability	Able to be re-used for alternate applications
Integration with existing technology	Able to integrate easily with existing technology
Outlier data points	Able to deal with outliers

4.3 Dataset Selection Process

The first of the two datasets used in the current research project was generated from a deterministic takeoff distance model based on the principles described in Section 3.2 and in accordance with Part 25 of the Federal Aviation Regulations (FARs) and Part V of the Canadian Aviation Regulations (CARs) (Airworthiness Standards for Transport Category Airplanes). The second dataset was obtained from NASA's DASHlink [68] (Discovery in Aeronautics Systems Health) initiative, a web-based tool for collaborative research in data mining and systems health. The primary goal of DASHlink is to disseminate information on the latest data mining and systems health algorithms, data and research. This dataset comes in the form of flight data collected from sensors onboard an unidentified aircraft (for legal purposes).

4.3.1 Deterministic Takeoff Distance Dataset

The first dataset's parameters are shown in Table 5. Based on the deterministic relationships describing the calculation of the TOD described in Section 3.3, it was decided that only the most influential factors affecting the TOD performance would be varied for development of the NN. Consequently, all other parameters involved in the TOD deterministic model were kept constant and are listed in the first column of Table 5. The parameters of interest are listed in the second column of Table 5 and are the aircraft weight, the pressure altitude, the temperature, the wind speed and the runway slope. Figure 13 shows the parameters as input and output values of the neural network, where the takeoff distance corresponds to the Takeoff Field Length (TOFL), which is the most constraining out of the TOD_{OEI}, TOD_{AEO}, and TOD_{ASD}. Figure 14 shows a sample of the raw deterministic TOD model used to generate the dataset. It is a text format file (.txt), which facilitates customization of the dataset as well as optimizes file size. This is an important choice as the customization will allow only the desired parameters to be selected for the NN or to modify the dataset to better suit the optimization of the NN architecture. The file size consideration is also important as this dataset can be very large in size (between 70 MB and 61 GB depending on what parameters and the number of test cases that are selected).

Table 5 – Parameters affecting TOD that can be obtained from the TOD deterministic model

Constant Input Parameters	Varying Input Parameters	Output Parameters
Aircraft & Engine Configuration	Weight	TOD_{OEI}
Flap Configuration	Pressure Altitude	TOD_{AEO}
Balanced V_1	Temperature	TOD_{ASD}
V_2	Wind Speed	TOFL
Runway Condition (Dry)	Runway Slope	
Thrust Setting		
Engine ECS (Off)		
Anti-Ice (Off)		
BTMS		
MMEL/CDL effects		

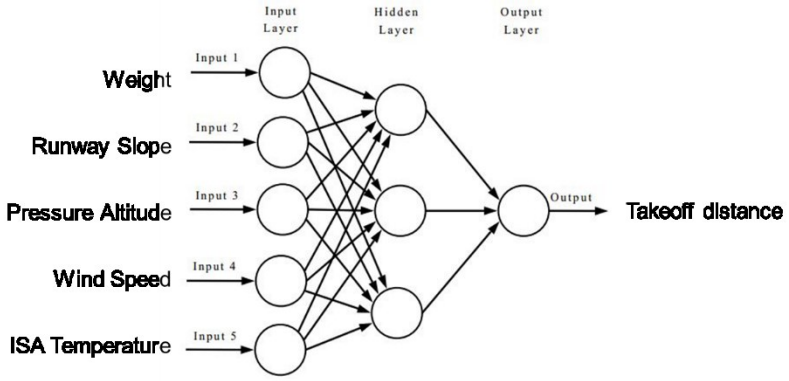


Figure 13 – Selected input and output values for the neural network

```
#TOFL
#Test Case =
$Rev$
#Validated Scenario = TOFL
#Aircraft =
#Engine =
caltyp = 6
perfms = 05
dtmtyp = 0
apdf =
config =
wgt = 70000/500/110000
flap = FLAP2
spd3t = Balanced
v2vstp = Minimum
btms = BTMS8
rlgto = No
thrpro = NORMAL
ecsto = Off
ai = Off
rycdn = Dry
rygrd = 0/0.1/2
alt = 0/1000/10000
wnd = 0
temtyp = DISA
temp = 0/5/40
mmel = NONE
```

wgt	rygrd	alt	temp	Status	toroei	todoei	toraeo	todaeo	asd	v1	vr	v
[lb]	[%]	[ft]	[deg C]	-	[ft]	[ft]	[ft]	[ft]	[ft]	[KIAS]	[KIAS]	[
70000	0	0	0	WRN	2308.149	2767.914	2573.012	2977.773	3350.566	108.8111	108.8065	1
70000	0	0	5	WRN	2338.541	2801.929	2606.698	3014.254	3397.776	108.7836	108.7778	1
70000	0	0	10	WRN	2368.79	2835.771	2640.274	3050.6	3444.712	108.7548	108.7502	1
70000	0	0	15	WRN	2398.947	2869.49	2673.707	3086.779	3492.771	108.7284	108.7226	1
70000	0	0	20	WRN	2455.873	2935.855	2729.904	3144.652	3552.658	108.6504	108.6446	1
70000	0	0	25	WRN	2587.907	3093.665	2848.214	3260.554	3644.336	108.4366	108.432	1
70000	0	0	30	WRN	2754.768	3291.703	2998.621	3407.292	3756.031	108.1813	108.1767	1
70000	0	0	35	WRN	2938.354	3503.884	3171.648	3577.058	3889.65	107.9436	107.9402	1
70000	0	0	40	WRN	3133.328	3725.791	3365.24	3773.111	4038.885	107.7259	107.7213	1
70000	0	1000	0	WRN	2403.872	2879.707	2666.899	3074.315	3447.128	108.672	108.6662	1
70000	0	1000	5	WRN	2435.834	2915.455	2702.326	3112.619	3496.182	108.6432	108.6398	1
70000	0	1000	10	WRN	2467.708	2951.083	2737.636	3150.782	3545.207	108.618	108.6122	1
70000	0	1000	15	WRN	2499.456	2986.553	2772.828	3188.799	3595.36	108.5928	108.587	1
70000	0	1000	20	WRN	2558.355	3054.922	2831.01	3248.523	3657.465	108.5171	108.5125	1
70000	0	1000	25	WRN	2693.971	3215.281	2955.121	3370.666	3755.38	108.3151	108.3092	1
70000	0	1000	30	WRN	2851.885	3401.978	3098.479	3510.804	3865.992	108.0909	108.0863	1
70000	0	1000	35	WRN	3043.637	3622.518	3280.615	3689.723	4008.152	107.8602	107.8556	1
70000	0	1000	40	WRN	3245.652	3851.324	3487.48	3904.069	4165.694	107.6497	107.645	1
70000	0	2000	0	WRN	2524.695	3020.941	2783.335	3192.478	3558.493	108.4991	108.4933	1
70000	0	2000	5	WRN	2558.62	3058.857	2820.82	3232.888	3609.929	108.4738	108.4692	1
70000	0	2000	10	WRN	2592.49	3096.675	2858.26	3273.23	3661.473	108.4486	108.4428	1
70000	0	2000	15	WRN	2626.143	3134.252	2895.49	3313.33	3713.423	108.4233	108.4187	1
70000	0	2000	20	WRN	2665.448	3178.519	2937.629	3358.09	3768.361	108.3909	108.3851	1
70000	0	2000	25	WRN	2805.101	3342.402	3067.197	3485.893	3872.333	108.1994	108.1936	1
70000	0	2000	30	WRN	2957.367	3520.892	3207.661	3623.734	3985.271	108.004	107.9994	1
70000	0	2000	35	WRN	3154.311	3746.762	3395.545	3808.433	4133.231	107.7792	107.7746	1
70000	0	2000	40	WRN	3365.311	3984.65	3617.616	4043.16	4300.237	107.5745	107.5711	1
70000	0	3000	0	WRN	2634.436	3147.635	2892.15	3303.941	3669.551	108.3668	108.361	1
70000	0	3000	5	WRN	2670.22	3187.568	2931.625	3346.391	3723.43	108.3415	108.3369	1
70000	0	3000	10	WRN	2705.857	3227.336	2970.982	3388.696	3777.041	108.3163	108.3116	1
70000	0	3000	15	WRN	2741.443	3267.02	3010.185	3430.819	3830.947	108.2934	108.2876	1
70000	0	3000	20	WRN	2776.937	3306.577	3049.344	3472.871	3885.03	108.2693	108.2647	1
70000	0	3000	25	WRN	2923.146	3476.899	3186.652	3608.528	3996.522	108.0849	108.0802	1
70000	0	3000	30	WRN	3075.915	3653.537	3331.122	3750.988	4116.481	107.9122	107.9064	1
70000	0	3000	35	WRN	3271.094	3877.7	3516.971	3933.691	4264.296	107.6981	107.6935	1

Figure 14 – Sample of the deterministic dataset[1]

Figure 15 shows how the deterministic dataset can be preprocessed and compressed to optimize run time and local storage. All unnecessary string characters are removed (i.e. spaces, character returns, paragraph returns) and only the five selected input parameters and the output TOFL value are kept.

test_clean.out

```
wgt,rygrd,alt,temp,wnd,Status,toroei,todoei,toraeo,todaeo,asd,v1,vr,v1/vr,v2,v2/vs,vfto,vlmcg,vlmbe,tofl
70000,0.75,10000,40,0,OK,4996.005999999999,5829.116,5246.835999999999,5763.339,5829.095,107.1892,107.9523,0.9929314,113.809,1.
70000,1.0,10000,40,0,OK,5048.715,5894.831,5311.043000000001,5830.55,5894.441999999999,107.6331,108.2382,0.99441,113.809,1.130(
70000,1.0,10000,40,10,OK,4705.554,5520.552,4974.458,5476.415,5520.609,106.9544,108.2382,0.9881389000000002,113.809,1.130003,15
70000,1.0,10000,40,30,OK,4209.304,4962.067,4334.139,4800.996,4962.07,107.2623,108.2382,0.9909843,113.809,1.130003,159.0119,10(
70000,1.25,9000,40,0,OK,4792.53,5616.888000000001,5040.909000000001,5547.549,5616.9619999999995,107.3916,107.9462,0.9948621,11
70000,1.25,9000,40,30,OK,4014.492,4746.379,4103.396,4558.048,4746.43,107.3639,107.9462,0.9946055,113.7895,1.129998,158.9217,1(
```

Figure 15 – Compressed deterministic dataset

[1] The font in the figure was intentionally sized to be unreadable due to proprietary reasons and is only used for illustrative purposes.

Figure 16 shows the distribution of each of the parameters from the dataset. The distribution for all input parameters is linear and the distribution for the output TOFL parameter is polynomial. The y-axis shows the amount of test cases generated from the TOD deterministic model and the x-axis shows the values of each test case generated. Values in the y-axis are scaled to allow for a better comparison. For some test cases, the TOD deterministic model cannot produce any values since they are physically impossible. For example, some test cases with DISA temperatures lower than -50 C are impossible for the aircraft to operate in and a reduction in generated number of test cases results from this fact.

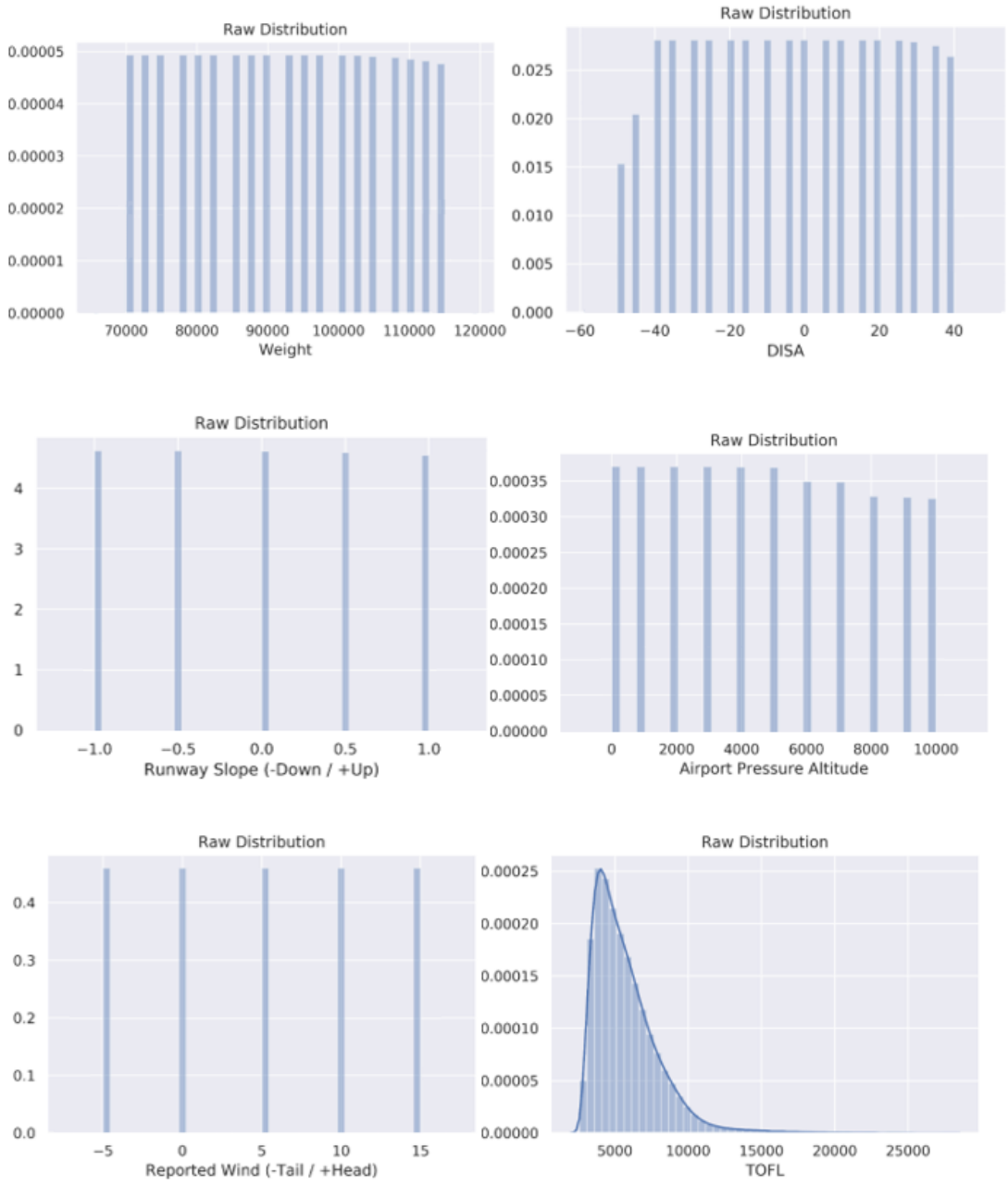


Figure 16 – Dataset distribution for individual input and output parameters

Three different preprocessing operations were tested: 1) the raw distribution without any modifications, 2) a normalized distribution, and 3) a standardized distribution. The raw distribution shows that the values of each parameter can vary greatly, which will affect the NN's prediction accuracy. The normalized and standardized distribution were generated to solve this problem. Normalization is selected as the best option using trial and error.

4.3.2 NASA DASHlink Non-Deterministic Dataset

The NASA DASHlink dataset is composed of aggregate flight recorded data, consisting of actual data recorded onboard a single type of regional jet operating in commercial service over a three-year period (2001-2004). NASA states the following about the flight data [68]: "While the files contain detailed aircraft dynamics, system performance, and other engineering parameters, they do not provide any information that can be traced to a particular airline or manufacturer. […] The appropriate parties have allowed NASA to provide the data to the general public for the purpose of evaluating and advancing data mining capabilities that can be used to promote aviation safety". The flight data provides an exhaustive list of parameters that are not required for this study. The parameters used for the current work can be found in Figure 17 and include the pressure altitude, fuel quantities, aircraft weight, total air temperature, wind speed, ground speed, altitude and Greenwich mean time. The figure also shows how each sensor value can be used to calculate aircraft total weight, distance traveled, the normalized wind speed, the elapsed time and takeoff distance; which are all values required for the calculation of TOD. The selected input and output parameters are the same as for the previous deterministic dataset shown in Figure 13 with the exception of runway slope, which was not possible to calculate due to lack of sensor accuracy.

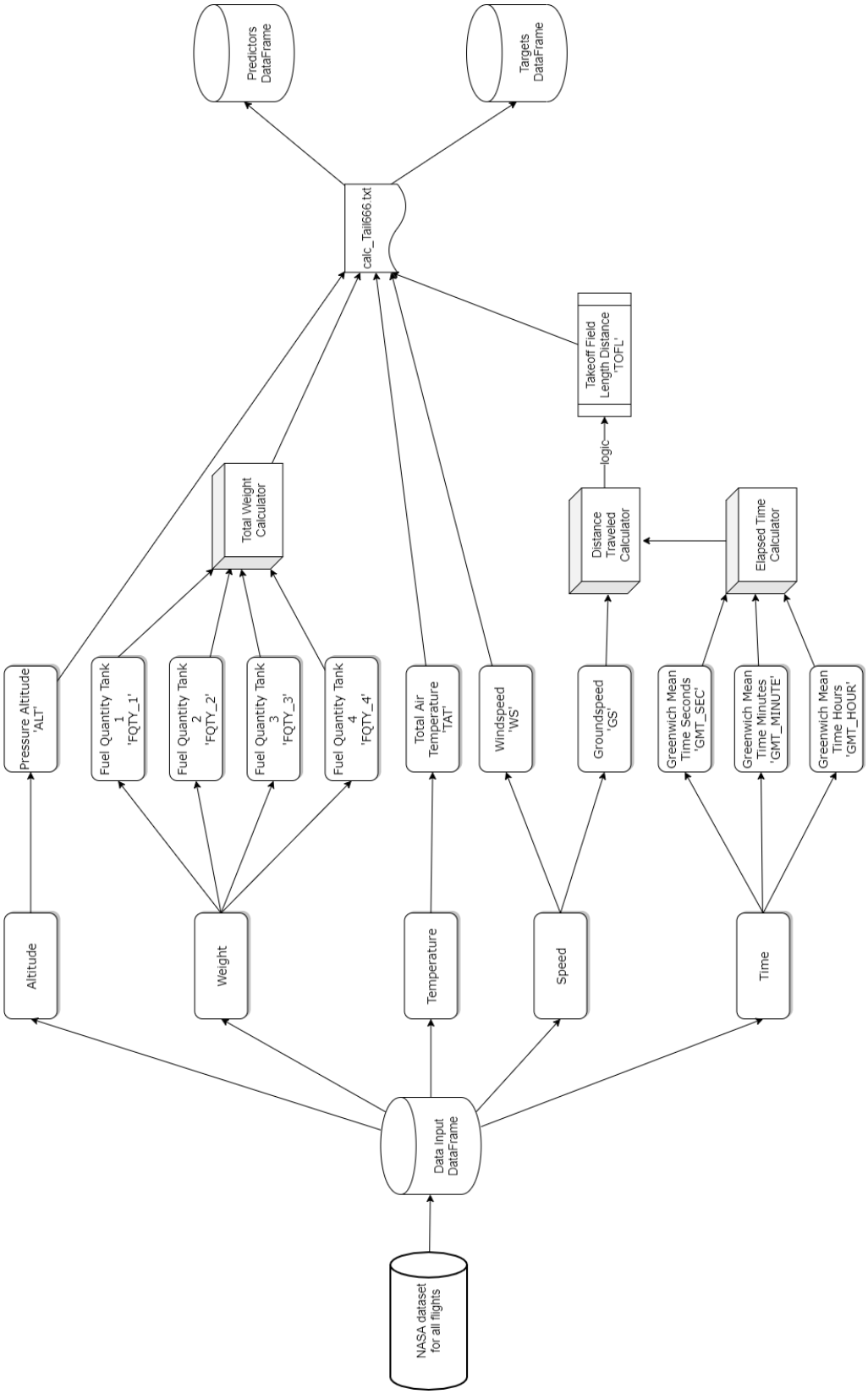


Figure 17 – Process of generating secondary data from the flight data

Upon further analysis of each input parameter of the original dataset's distribution as seen in Figure 18, distinct zones in the data distribution were identified for which not enough datapoints are available to build a NN capable of adequately generalizing patterns. This observation was validated through empirical trial and error. These problematic distribution zones are highlighted in red in Figure 18, and were removed from the datasets to be tested in the research. This data preprocessing approach is based on the following reasoning:

1. Some datapoints were physically unrealistic. These are attributed to sensor errors or inaccuracy. Examples include TOFL values lower than zero or aircraft weights more than ten times higher than other values, considering all values are meant to be attributable to the same type of regional jet.
2. Values that are not proportionately distributed. The flap drag and normalized wind speed distributions both show values that are not proportionately distributed as almost all datapoints are concentrated with a very small number of test cases. This significantly increase the difficulty in training a NN that generalizes well across all datapoints.
3. Zones of higher probability. The zones having the highest distributions, highlighted in green in Figure 18, were prioritized to facilitate generalization by reducing the amount of outlier datapoints.

The original dataset was separated into two distinct datasets: one consolidating data for a single aircraft and another for the entire fleet of twelve aircraft. The preprocessing approach discussed above was used to reduce the dataset using the limits shown in Table 6 and Table 7 respectively. Figure 19 and Figure 20 show the distributions of the two finalized datasets used in this study.

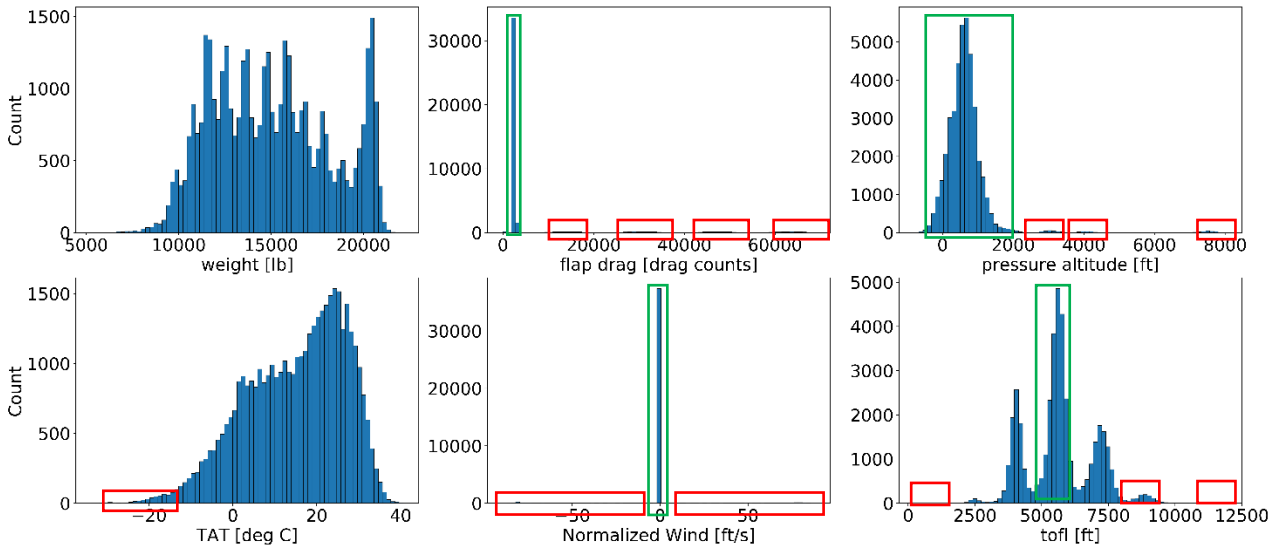


Figure 18 – Original dataset distribution for the fleet of 12 aircraft

Table 6 – Limits imposed on original dataset for 1 aircraft

Parameter	Limits
Weight	> 0 & < 23000
Flap Drag [drag counts]	> 2400 & < 2470
Pressure Altitude [feet]	> -800 & < 2400
Total Air Temperature [deg C]	> -10
Normalized Wind Speed [ft/s]	= 0
TOFL	> 0 & < 10000

Table 7 – Limits imposed on original dataset for the fleet of 12 aircraft

Parameter	Limits
Weight	> 0 & < 23000
Flap Drag [drag counts]	> 1800 & < 2600
Pressure Altitude [feet]	> -800 & < 2400
Total Air Temperature [deg C]	None
Normalized Wind Speed [ft/s]	= 0
TOFL	> 4700 & < 6500

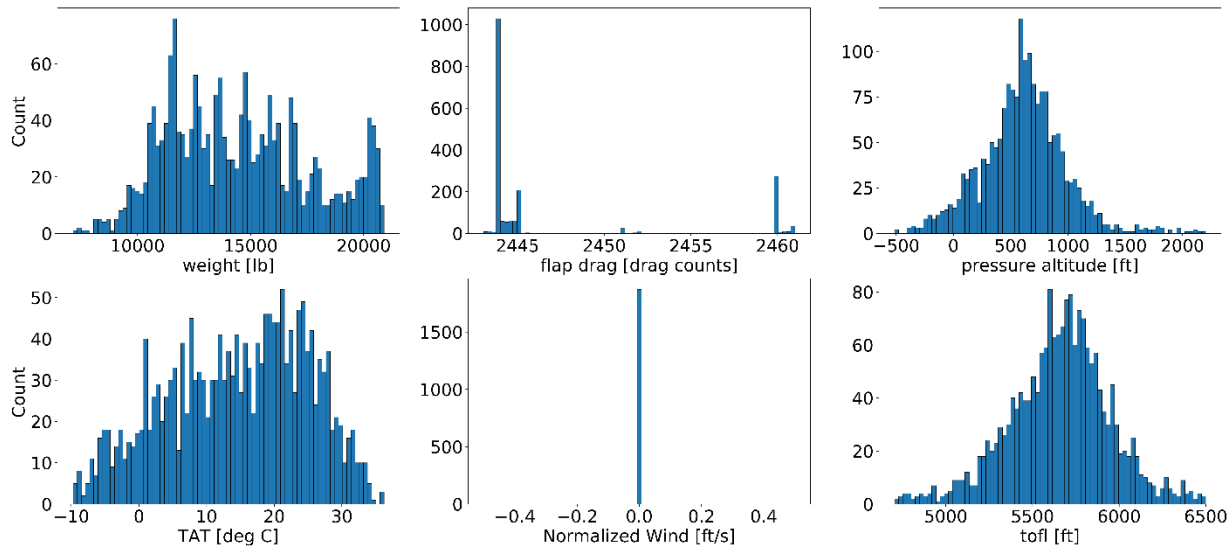


Figure 19 – Selected dataset distribution for 1 aircraft

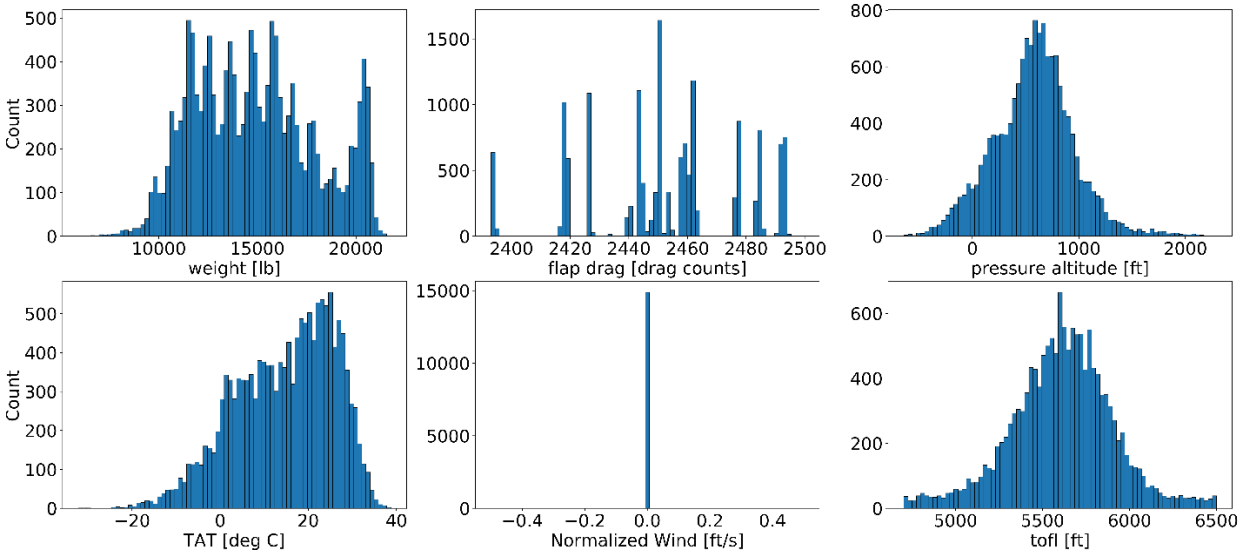


Figure 20 – Selected dataset distribution for the fleet of 12 aircraft

4.4 Neural Network Architecture Development

The development of the network architecture involves the definition of the architecture components listed below and depicted in Figure 21.

1. the optimization function;
2. the loss function;
3. the activation function;
4. the number of hidden layers;
5. the number of neurons per hidden layer; and

6. the maximum allowable number of epochs.

Layer 1 is the input layer and its number of neurons must be equal to the number of input values in the dataset. The last layer, Layer 4, is the output layer and its number of neurons must be equal to the number of output values in the dataset. Layers 2 and 3 are referred as 'hidden layers' and are used to propagate the information in a parallel manner towards the output solution. The more hidden layers a network has the 'deeper' a network is said to be. The term Deep Learning is generally used to describe a NN which has two or more hidden layers. Each neuron is constructed of an activation function, which decides when and if a neuron is used in the calculation, an optimization function, which is used to decide how to propagate the information on to the next neuron, and a loss function, which is the function the optimization function is trying to optimize. The combined objective of these components is to work in a manner that finds the weighted dot product of the value attributed to each neuron which gives the closest value to the desired output value. This is done by updating the weights and biases of each network connection. This type of 'feedforward backpropagation' neural network uses the backpropagation algorithm to update the weights and biases which will yield the best outcome solution. The number of times the weights and biases are updated and backpropagated down the network are called 'epochs', which can be viewed as a back-and-forth iteration. Selecting different types of functions and values for these architecture parameters significantly affects the network performance. General rules from the literature can be used [9, 13] in order to narrow down the search:

1. As a general observation, the deeper the network, the higher the generalization capability of the network. This comes with an increased need in computing power, longer run times, and may not necessarily always yield better results in cases where there is not enough data to extract patterns or the data is too noisy.
2. Certain types of optimization, loss and activation functions are more efficient for specific applications and datasets. These will be explained in more detail further in the current section.
3. The number of neurons per hidden layer and maximum allowable number of epochs are a function of the dataset size.

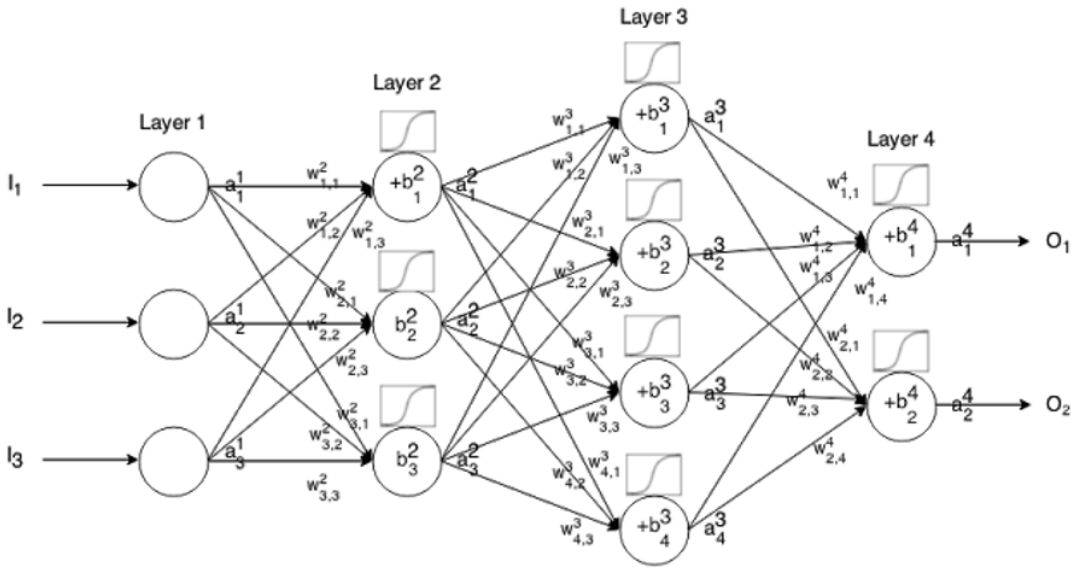


Figure 21 – Neural network architecture development process [69]

A number of different tools are available for the development of a neural network. NNs can be programmed directly in low-level of abstraction programming languages like C, but the associated workload and programming knowledge required is high. A number of different programming frameworks have been created in an effort to alleviate these issues. Programming frameworks have different purposes and orientations including industrial development, academic research, fast prototyping, and ease of model implementation. Machine learning frameworks include TensorFlow, Keras, PyTorch, Theano, Matlab, and Caffee, among others. Figure 22 shows a ranking of the most commonly used ML frameworks based on a study done by Jeff Hale [70, 71], which looked at ranking different deep learning frameworks with respect to various categories. The Keras API was selected for the current research based on the following advantages:

1. Efficiency in reducing cognitive load (i.e. consistent and simple APIs, minimizes number of user actions required for common use cases, and it provides clear and actionable feedback upon user error).
2. Useful for fast prototyping and experimentation.
3. Ease of implementation across a wide variety of products (iOS, android, browser, Google Cloud, Raspberry Pi etc.).
4. Widespread adoption and ease of access on supporting documentation. It is fully recognized as a front end to TensorFlow, which could be used in future research to further the optimization of the designed network.

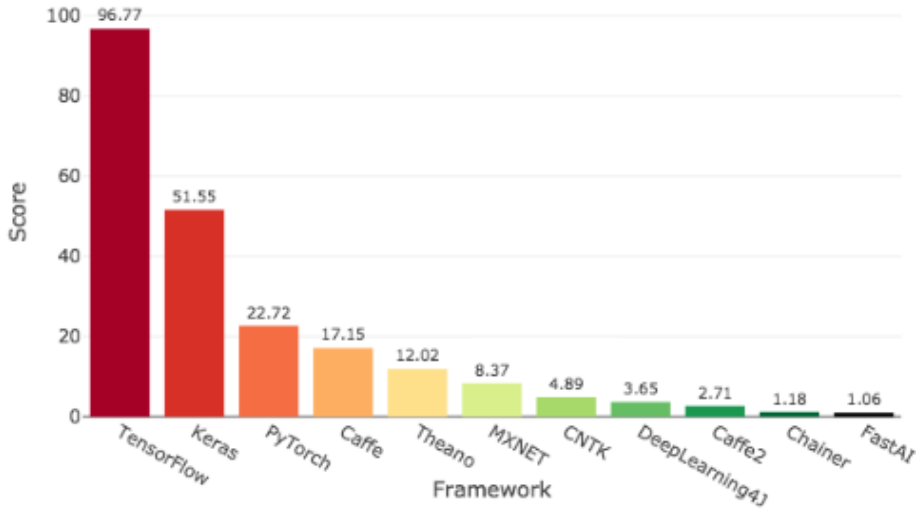


Figure 22 – Ranking of deep learning frameworks [70]

Figure 23 shows a classification of the parameters commonly used in ML that can be selected when defining a network architecture. All the functions present in the figure are available in Keras, and there is also an option to create customized functions.

Selecting the Relevant Loss Functions

The current work is interested in solving a regression optimization problem rather than a classification problem, which narrows down the architecture selection process. From the loss functions shown in Figure 23, the logcsoh, MAPE, MAE, MSE, and MSLE functions are relevant for a regression problem. Table 8 shows definitions for each loss function as well as their most used applications. Grover [71] explains that median is more robust to outliers than mean, which consequently makes MAE more robust to outliers than MSE. Furthermore, De Myttenaere et al. [72] explain that finding the best model under the MAPE is equivalent to doing weighted Mean Absolute Error (MAE) regression. For these reasons, MSLE and MAPE are used for NN testing.

Due to safety considerations associated with aerospace applications, the predictions generated by the NN must be conservative. Even though it is desirable to have the NN architecture that produces the lowest MSLE or MAPE values (which are a measure of the overall error for all test cases), it must be noted that the driving metric to determine the best model is the worst-case error for all possible test cases.

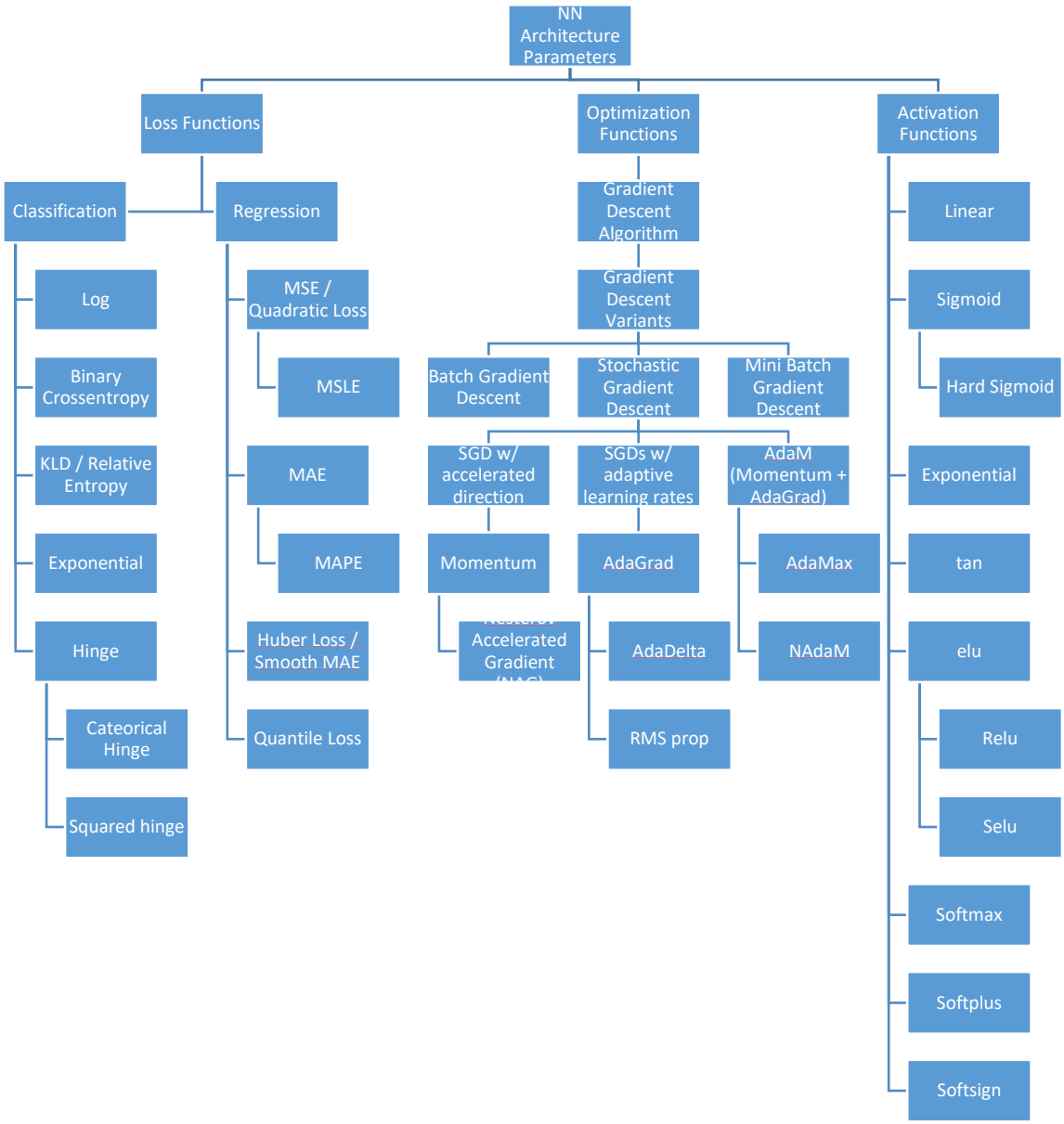


Figure 23 –-Commonly used NN architecture parameters

Table 8 – Loss function definitions and applications

Definition	Typical Application
$MSE = \frac{1}{n}\sum_{i=1}^{n}(actual_i - predicted_i)^2$	Well suited for datasets that have gaussian distributions.
$MSLE = \frac{1}{n}\sum_{i=1}^{n}[\log(actual_i) - \log(predicted_i)]^2$	Well suited for datasets that have high variance. It suffers from the problem of gradient and hessian for very large off-target predictions being constant.
$MAE = \frac{1}{n}\sum_{i=1}^{n}(\lvert actual_i\rvert - \lvert predicted_i\rvert)$	Well suited for datasets that have mostly gaussian distributions, but that contain outliers.
$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left\lvert\frac{actual_i - predicted_i}{actual_i}\right\rvert$	Same as MAE, but as a function of relative percentage error.

Selecting the Relevant Optimization Functions

For ML regression problems, the optimization functions shown in Figure 24 used are variants of gradient descent algorithms called stochastic gradient descents (SGD) [9, 73]. A gradient descent is an optimization algorithm used to minimize some loss function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In ML, gradient descent is used to update the weights of the NN. The SGD algorithm can be faced with two known problems [74]:

1. Local Minimum: SGDs have issues dealing with regression curves that are very steep. These can lead to local minimum problems, which traps the SGD algorithm in an incorrect local optimum solution. Momentum [75] was developed to solve this issue by accelerating the SGD solution towards a relevant direction.
2. Learning Rate Selection: Selecting the optimal learning rate is difficult as one that is too small can lead to a very slow convergence, while a one that is too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge. Some SGD variants make use of adaptive learning rates, which update depending on other parameters.

Table 9 shows a summary of the relevant SGD optimization functions. The Nadam (Nesterov-accelerated Adaptive Moment Estimation) optimization function was selected because it provides the most benefits in dealing with local minimum and for finding optimal learning rates.

Table 9 – Optimization functions and their applications

Optimization Function	Typical Applications
Nesterov Accelerated Gradient (NAG)	Helps accelerate SGD out of problematic local minimum solutions.
Adagrad	Adapts the learning rate to the frequency of occurrence of parameters. It is well-suited for dealing with sparse data.
Adadelta and RMSprop	An extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate.
Adam (Adaptive Moment Estimation) [76]	Adapts the learning rate by storing an exponentially decaying average of past squared gradients (like Adadelta and RMSprop), while also keeping an exponentially decaying average of past gradients (like momentum).
AdaMax	A variant of Adam useful for dealing with unstable learning norms of past gradient. Practical when dealing with larger norm values for past gradients.
Nadam (Nesterov-Accelerated Adaptive Moment Estimation)	Combines Adam and NAG capabilities.

Selecting the Relevant Activation Functions

The selection of the appropriate activation function depends on the application of the NN. Some are developed to give binary inputs (0 or 1), others to give values between 0 and 1 and others to give exact values. Goodfellow et al. [9] recommend using the ReLU (Rectified Linear Units) activation function for the input and hidden layers and linear or sigmoid functions for the output layer. If the ReLU function does not yield acceptable results, derivatives of the ReLU (eLU, SeLU) or Softmax, Softplus, and Softsign can be empirically tested.

Epochs

For the current research, the maximum number of allowed epochs is not considered a critical parameter for the architecture definition. Goodfellow et al. [9] explain that it is often more useful to define an "early stopping criteria", which, when triggered, will stop the training regardless of the number of epochs reached. The criteria used is: "if the last ten epochs have produced no improvement of mean average percentage error the training will stop". This criterion has the added benefit of saving computation time as the NN will not be training for further epochs which would in any case not produce better results.

Table 10 provides a matrix of the different combinations of NN architectures that were tested using the different datasets, the results of which will be provided in Chapter 5.

Table 10 – Selected architecture parameters to test using the datasets

Optimization Function	Loss Function	Activation Function	Nodes/Input Layer	Hidden Layers	Nodes/Hidden Layer
Adadelta	MAPE	ReLU	1000	1	10
Adamax	MSLE		3000	2	100
Nadam			5000	3	1000

4.5 Training and Testing a Neural Network

Training a NN involves using a dataset to update the weights of the NN in order to find the closest values to the output solution, while testing a NN uses the training weights to make predictions with a new dataset to validate the results. Testing is also a good validation method to experiment for cases that were not covered in the training dataset. Training and testing of a NN can be done using the same dataset and splitting it in two. Most regression problems use a validation split of 70 % for training and 30 % for testing. This ratio will be used for all the results presented in this thesis. The Keras application has the ability to generate graphs that can be used to evaluate the performance during training and testing such as the one shown in Figure 24. For each tested architecture in this study, these graphs were used to evaluate the NN performance. The y-axis plots the performance metric of choice while the x-axis shows the number of epochs, and thus, is a measure of the training time. In Figure 24 for example, the chosen metric is MAPE and is compared for training and testing values. The graph is an indication of: 1) How fast a metric can converge to a steady-state value and 2) the precision of the steady-state value.

A known problem that can occur when comparing training and testing results is overfitting or underfitting. Underfitted values are not able to extract the underlying pattern found in the dataset while overfitted values are able to extract this pattern too well as depicted in Figure 25. Underfitted values generally cannot converge to an acceptable solution while overfitted values do not allow accommodation for gaps between datapoints. This can result in poor model performance when testing values from a different dataset than in training. These issues can be mitigated by using a validation ratio close to the 30/70 and by providing enough data.

Appendix A provides an example of the code that can be used to train and test a NN using Keras in Python.

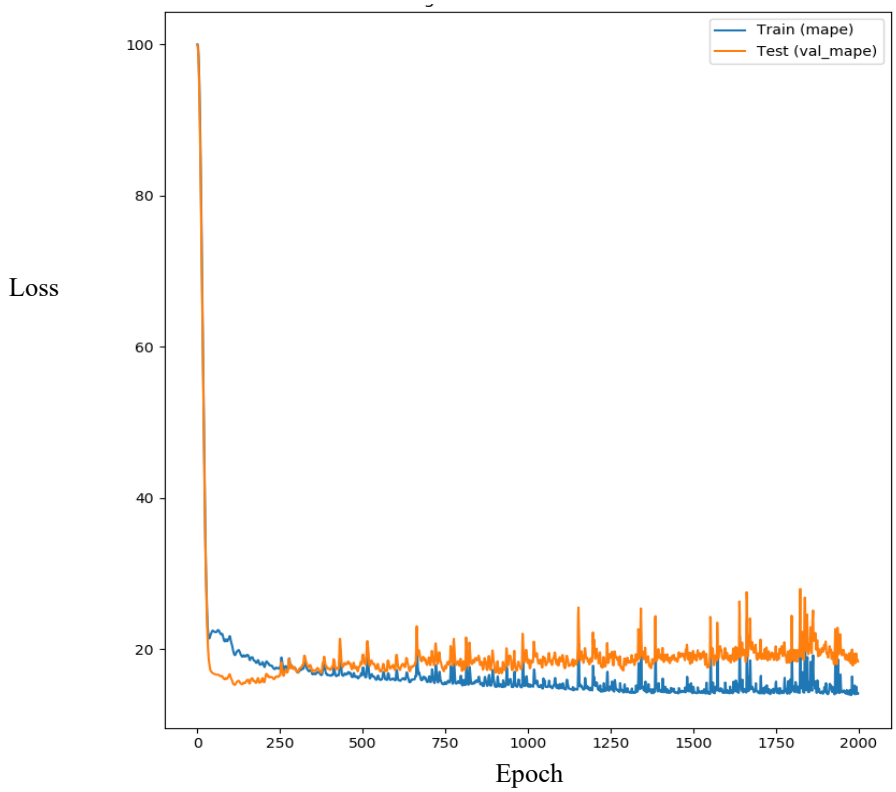


Figure 24 – Training and testing MAPE vs epoch

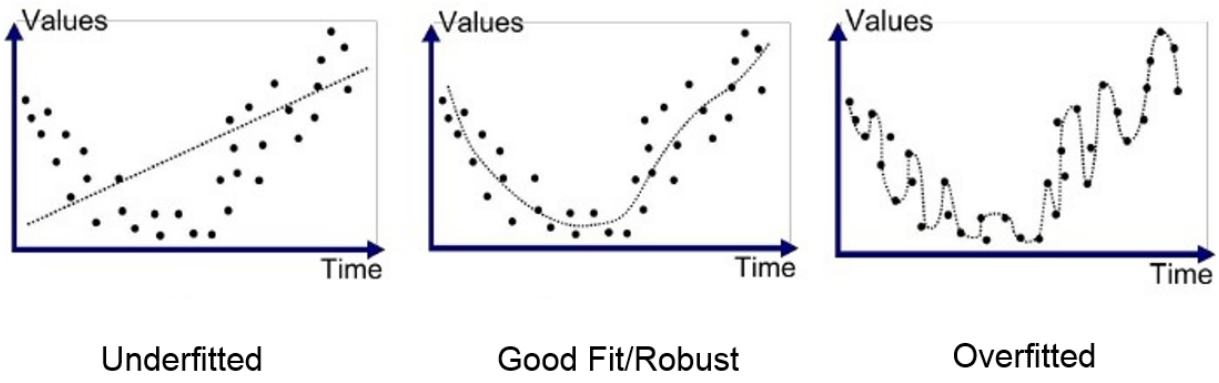


Figure 25 – Comparison of underfitting and overfitting [77]

4.6 Using the Trained Network to Make Predictions

Once the NN model is trained, a file is generated that contains only the trained weights of the network. A simple prediction tool program can be developed to give an output similar to the one of Figure 26. The user feeds the tool inputs for which they desire the tool to predict TOD for and the tool predicts TOD and compares the NN value with the actual value from the dataset it was trained on. The run time of the prediction tool is very fast as it only looks up the weights and calculates one output based on a set of given

input parameters. The format of the trained weights of the NN also facilitates integration with existing tools (FMS, EFB, software AFMs, etc.).

```
Weight [lb] = 93000
Runway Slope (-Down/+Up) [%] = 1.25
Airport Pressure Altitude [ft] = 5000
DISA [deg C] = -20
Reported Wind (-Tail/+Head) [kts] = 15
     Takeoff Field Length [ft] = 4889.172

data=
        wgt  rygrd     alt  temp   wnd      tofl
0  93000.0   1.25  5000.0 -20.0  15.0  4889.172

Actual value          = 4889.1720
Predicted by neural net = 4889.7251
Percentage error = 0.0113%
Mean difference [ft] = 0.5531
```

Figure 26 – Output of the tool making use of the trained NN

Figure 27 depicts the file structure of the Python programs used in this research and described in this section. A dataset is analyzed using a "dataset_properties.py" file. This is done to facilitate the data preprocessing phase when dealing with very large datasets. Most datasets will not be in the proper format or will need to be preprocessed before they can be used for the development of NNs by removing datapoints or modifying the dataset distribution. This is done using the "clean_out_file.py" file. Next the "neural_network_architecture_selection.py" file is used to train and test the network and evaluate its performance. This code facilitates experimentation in selecting the optimal NN architecture for each dataset. For example, it provides graphs such as the one shown in Figure 28 and summary tables similar to Table 12 for each tested architecture. Each trained network is saved in "saved_net.hdf5" file types, which are very memory efficient, and can then be used by "neural_network_tool.py" to make predictions based on user input.

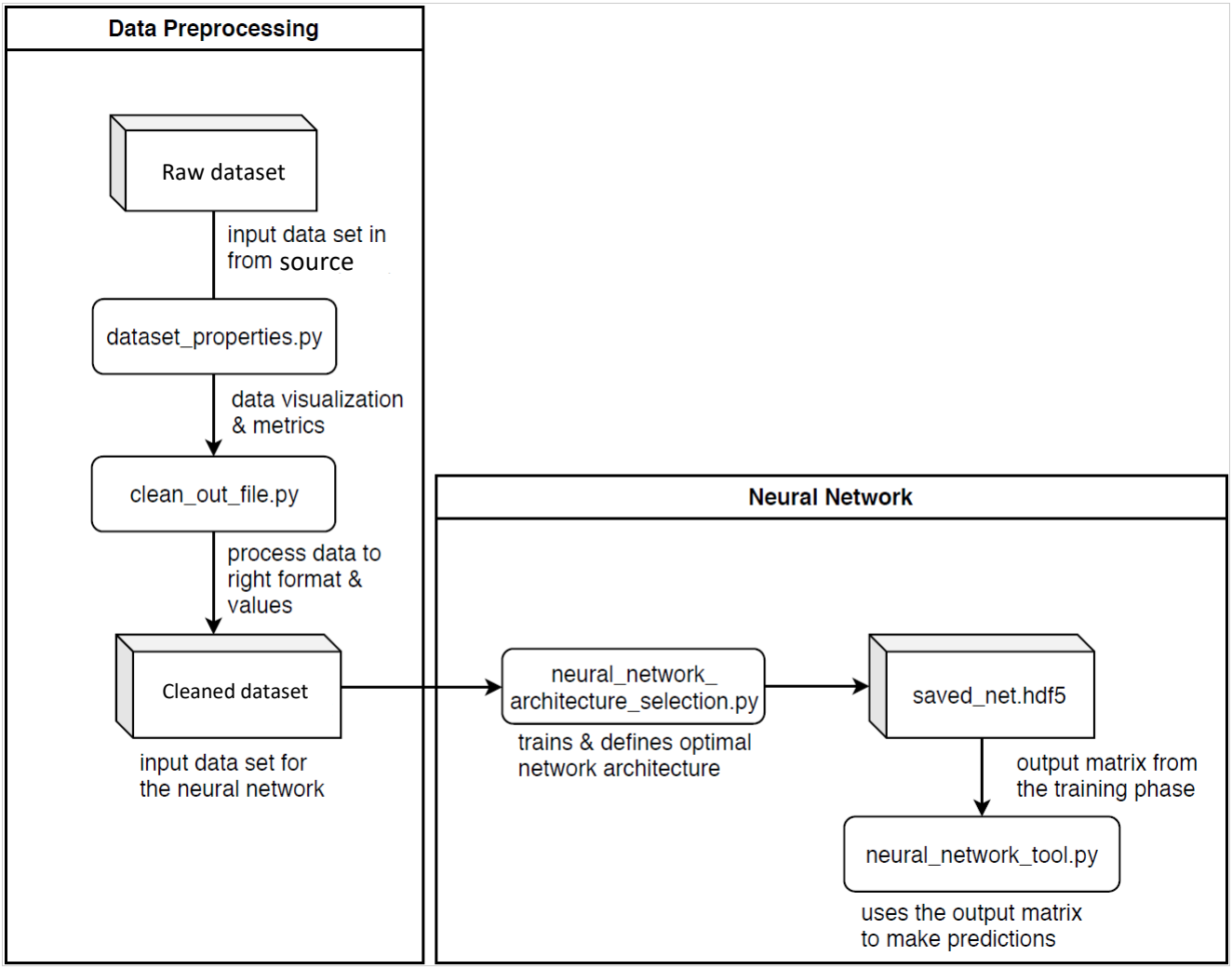


Figure 27 – Diagram of the code structure

5. Results

The following discussion of the results is organized according to which dataset was used to produce the neural network predictions. Section 5.1 reviews results found using datasets generated from the takeoff distance deterministic model described in Section 4.3.1. Section 5.2 reviews results found using the DASHlink dataset provided by NASA and described in Section 4.3.2.

5.1 Deterministic Takeoff Distance Dataset Results

For the NN developed using the deterministic TOD model dataset, three separate datasets were tested.

1. A small dataset totaling 1,000 test cases (dataset 1.1);
2. a medium sized dataset with the full scope of possible test cases totaling 358,722 test cases (dataset 1.2), and
3. a larger sized dataset with the full scope of possible test cases totaling 1,294,777 test cases (dataset 1.3).

Dataset 1.1 was used to provide a first estimate as to which optimization, loss and activation functions work best for the type of data used. A smaller dataset was selected in order to be able to run all available optimization, loss and activation functions efficiently.

Dataset 1.2 was used to evaluate the full scope of possible test cases that can be calculated by the deterministic model for the takeoff scenario. It is important to note that, since the dataset is based on the deterministic model's calculations, the interval between each test case value can be chosen, which affects the dataset distribution as seen in Table 11. The calculated values are also limited by the deterministic model's operating limitations. For example, the calculation might return errors or warnings if the calculated TOD is outside the physical limitations of the system or if the calculated value is for an aircraft operating in a dangerous zone. For the sake of convenience, the error and warning calculations were neglected in this study. This may also affect the dataset distribution since fewer test cases are present in zones where more errors or warnings occur.

Finally, dataset 1.3 was used to improve dataset 1.2's results and analyze the effect of increasing the size of the dataset on the results. Table 11 summarizes the dataset properties for datasets 1.1, 1.2, and 1.3.

Table 11 – TOD model dataset properties

Dataset	Input Parameters	Minimum Input	Step Size	Maximum Input	Unit
1.1	Weight	70000	500	73500	lb
	Runway Slope	-2	0.25	2	deg
	Pressure Altitude	0	1000	10000	feet
	ISA Temperature	-30	10	40	deg C
	Wind Speed	-10	10	30	knots
	Training Test Cases	700			
	Testing Test Cases	300			
	Total Test Cases	1000			
1.2	Weight	70000	500	110000	lb
	Runway Slope	-2	0.25	2	deg
	Pressure Altitude	0	1000	10000	feet
	ISA Temperature	-50	10	40	deg C
	Wind Speed	-10	10	30	knots
	Training Test Cases	251105			
	Testing Test Cases	107623			
	Total Test Cases	358722			
1.3	Weight	70000	500	110000	lb
	Runway Slope	-2	0.25	2	deg
	Pressure Altitude	0	1000	10000	feet
	ISA Temperature	-50	**5**	40	deg C
	Wind Speed	-10	**5**	30	knots
	Training Test Cases	906343			
	Testing Test Cases	388434			
	Total Test Cases	1294777			

5.1.1 Dataset 1.1 Results

The Python code is able to run multiple network architectures at once and presents the results in a manner that facilitates the review of each architecture. The 448 different architecture combinations of optimization, loss, and activation functions selected in Section 4.4 were all tested using Dataset 1.1. Figure 28 shows training and testing results for six NN architectures of interest. Model #1 and Model #2 are examples of training that encounters local minimum convergence, which leads the NN to make the same prediction for all test cases. Model #3 and Model #4 show results that are not trapped in a local minimum but that are much too high in percentage errors (around 99 %). Results for these models could be improved using more data but that approach was not undertaken because this thesis is focused on getting the best results for the least required amount of data. Model #5 and Model #6 demonstrate results which meet the criteria established as part of the research objectives. The resulting top ten model architectures obtained are summarized in Table 12, where the focus is on generating the lowest worst percentage error ($p_{err,train}$ and $p_{err,test)}$ while keeping the lowest mean average percentage error (MAPE) as a secondary priority. Upon evaluation of these results, the optimal optimization functions are adadelta, adagrad, adamax, adam, and nadam; the optimal loss functions are MSE, MAE and MAPE; and the optimal activation functions are all relu. These results validate the observations made in Section 4.4 as to which combination of network architecture is most efficient. As Nadam can deal with momentum best and MAPE gives the same results as MAE, it was chosen to use Nadam, MAPE and ReLU as the optimization, loss, and activation functions respectively for further testing with datasets 1.2 and 1.3.

Table 12 – Summary of the top 10 dataset 1.1 results based on best MAPE

model_id	opt_f	loss_f	actvn_f	MAPE[2]	p_err$_{train}$[3]	p_err$_{test}$[4]	err$_{train}$[5]	err$_{test}$[6]
194	adadelta	MSE	relu	1.89	9.76	9.12	474.71	467.76
193	adadelta	MSE	relu	1.99	9.19	7.83	420.44	401.91
146	adagrad	MAE	relu	2.17	12.60	11.68	449.55	450.67
338	adamax	MAE	relu	2.18	14.39	11.53	418.19	417.82
162	adagrad	MAPE	relu	2.19	11.83	12.17	472.37	472.85
354	adamax	MAPE	relu	2.19	12.68	11.60	458.08	459.06
290	adam	MAPE	relu	2.20	13.80	12.16	412.98	412.57
273	adam	MAE	relu	2.20	11.66	12.58	465.97	467.83
401	nadam	MAE	relu	2.20	15.36	11.86	380.36	380.26
417	nadam	MAPE	relu	2.20	13.46	11.91	423.09	422.36

[2] Mean average percentage error
[3] Worst percentage error for all test cases for the training results
[4] Worst percentage error for all test cases for the testing results
[5] Worst error for all test cases for the training results
[6] Worst error for all test cases for the testing results

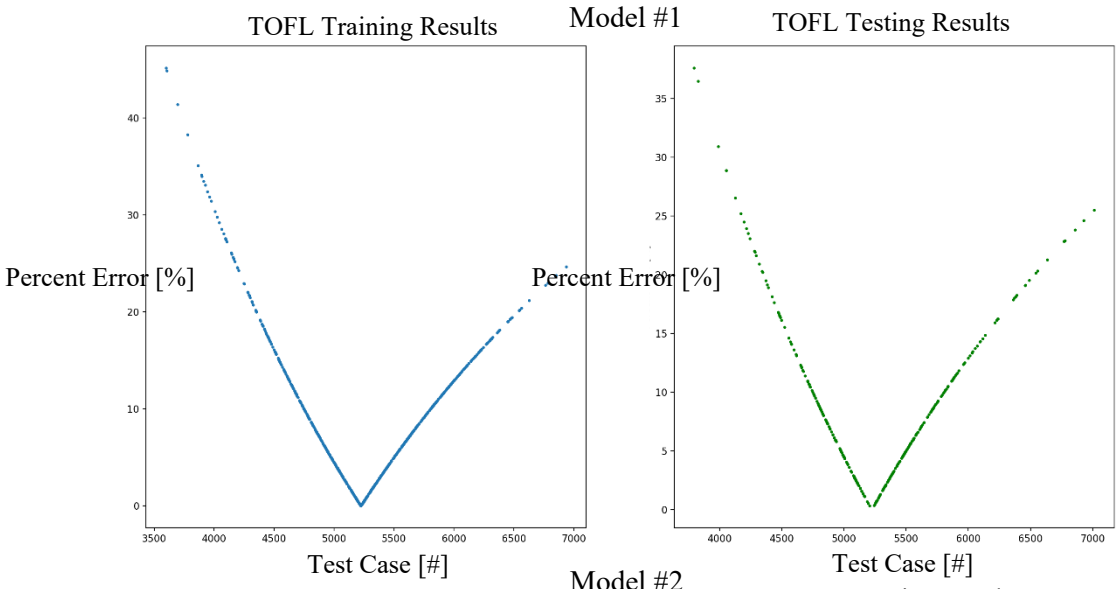
Table 13 – Summary of the top 10 dataset 1.1 results based on best p_err_{train}

model_id	opt_f	loss_f	actvn_f	MAPE	p_err_{train}	p_err_{test}	err_{train}	err_{test}
201	adadelta	MSE	softplus	2.29	8.64	9.23	355.13	346.82
193	adadelta	MSE	relu	1.99	9.19	7.83	420.44	401.91
118	rmsprop	MSLE	elu	2.38	9.43	9.29	479.96	467.16
409	nadam	MAE	softplus	2.31	9.43	9.89	314.10	307.21
73	rmsprop	MSE	softplus	2.79	9.68	10.16	380.75	373.39
194	adadelta	MSE	relu	1.89	9.76	9.12	474.71	467.76
297	adam	MAPE	softplus	2.70	9.84	10.05	400.33	400.43
120	rmsprop	MSLE	selu	2.48	9.87	9.57	502.54	490.83
391	nadam	MSE	selu	2.42	9.90	10.40	508.02	493.85
105	rmsprop	MAPE	softplus	2.70	9.91	10.10	393.16	394.20

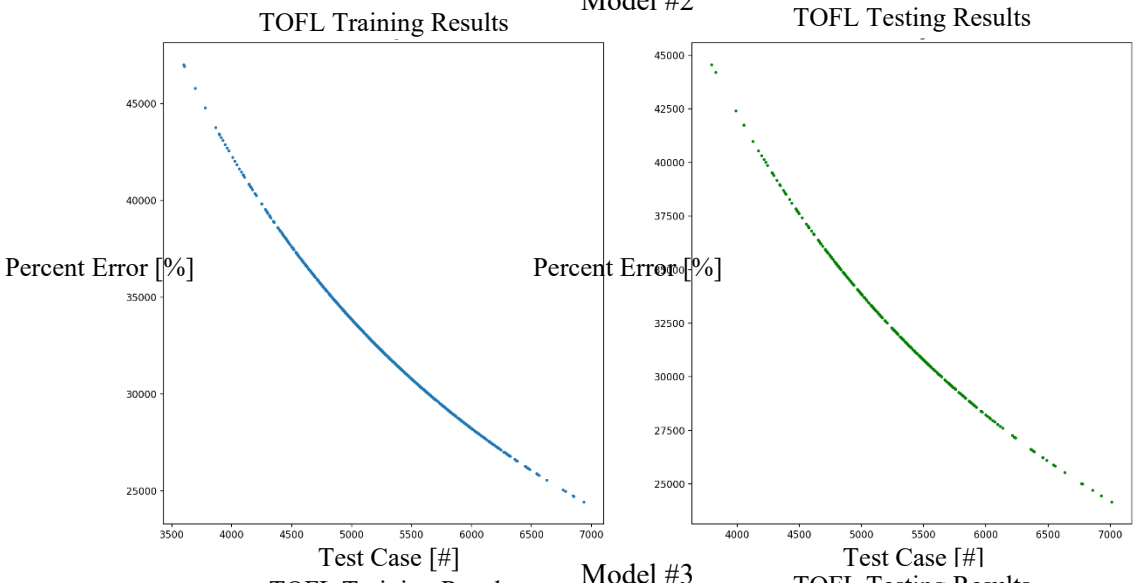
Table 14 – Summary of the top 10 dataset 1.1 results based on best p_err_{test}

model_id	opt_f	loss_f	actvn_f	MAPE	p_err_{train}	p_err_{test}	err_{train}	err_{test}
193	adadelta	MSE	relu	1.99	9.19	7.83	420.44	401.91
197	adadelta	MSE	elu	2.35	11.90	8.71	442.46	429.52
386	nadam	MSE	relu	2.32	14.05	8.95	433.57	418.17
66	rmsprop	MSE	relu	2.42	14.13	8.98	433.77	418.43
194	adadelta	MSE	relu	1.89	9.76	9.12	474.71	467.76
201	adadelta	MSE	softplus	2.29	8.64	9.23	355.13	346.82
118	rmsprop	MSLE	elu	2.38	9.43	9.29	479.96	467.16
70	rmsprop	MSE	elu	2.42	14.03	9.29	412.73	398.21
199	adadelta	MSE	selu	2.35	12.23	9.34	380.26	368.24
72	rmsprop	MSE	selu	2.54	11.76	9.54	502.59	489.45

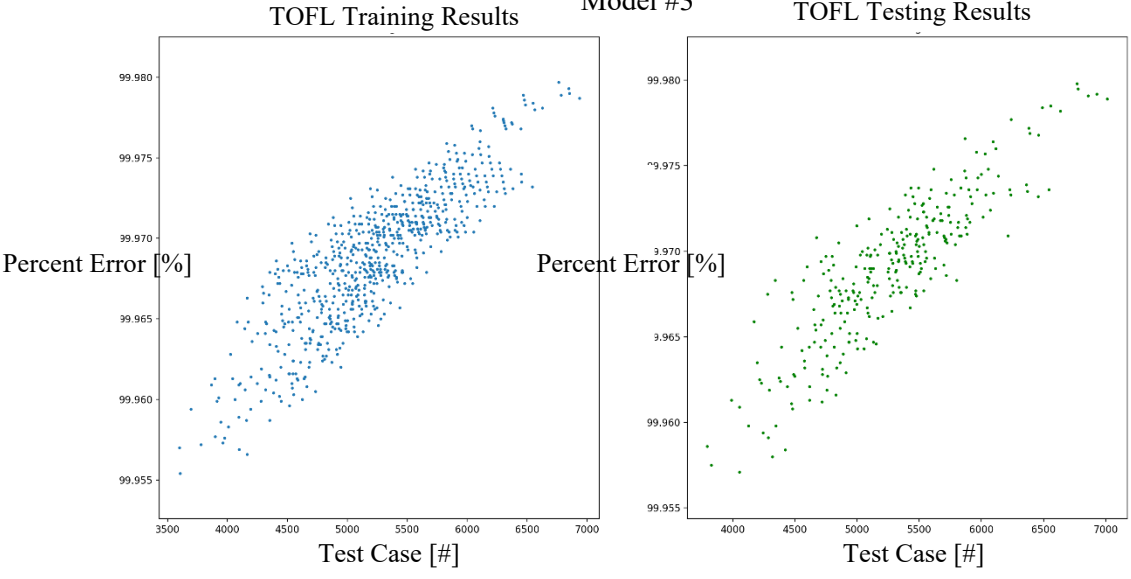
Model #1



Model #2



Model #3



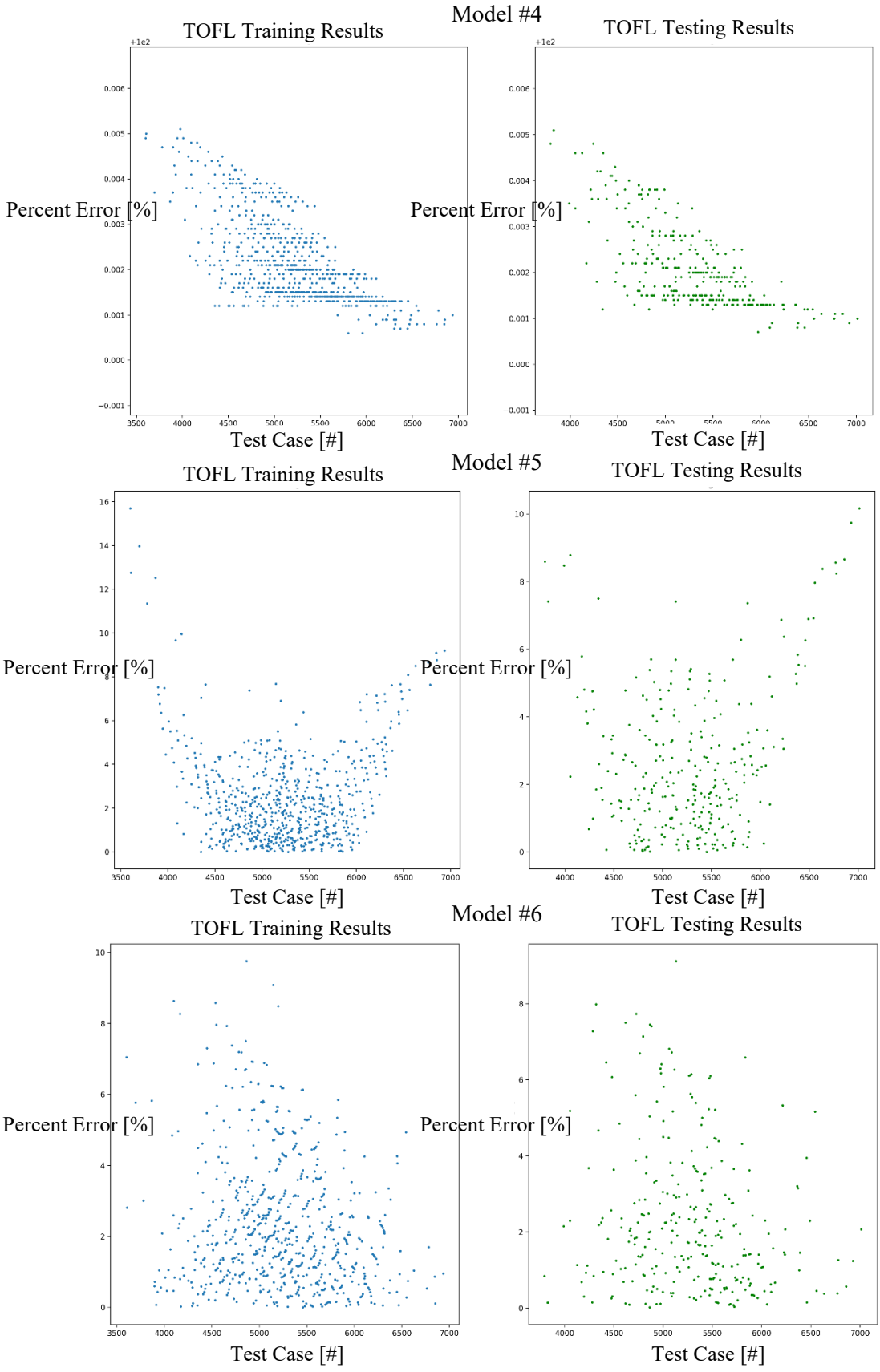


Figure 28 – Training and testing results for 6 different NN architectures

5.1.2 Dataset 1.2 Results

Dataset 1.2 was used to develop a NN model that can satisfy the NN requirements and objectives set out in Section 4.2, and which uses the optimization, loss, and activation functions found using dataset 1.1. To accomplish this, the effect of varying the following parameters was studied:

1. the number of nodes per input layer,
2. the number of hidden layers, and
3. the number of nodes per hidden layer.

For each varying parameter, the ranges that were tested are listed in Table 15 below.

Table 15 – Summary of the test ranges for models using dataset 1.2 values

Number of Nodes per Input Layer	1000, 3000, 5000
Number of Hidden Layers	1, 2, 3
Number of Nodes per Hidden Layer	10, 100, 1000

Table 16 shows the results of the best combination of the different architectures tested, and Figure 29 shows the error distribution for each test case for the training and testing dataset. The x-axis has 358,722 test cases (which encompass the full scope of possible TOD scenarios) and the y-axis has percentage errors between NN-predicted values and actual values calculated by the TOD deterministic model. For this model architecture, the MAPE is 0.04 % and worst percentage error is 1.29 %. The training time was 9.9 hours and when the trained NN is used to make predictions the run time is 0.03 seconds. When comparing the training and testing values in Figure 29, a slight underfitting of the testing data is observed. This could potentially be reduced using more data or a validation split higher on the testing side (i.e. 40 % for testing and 60 % for training). The results show that, for a given dataset, a larger number of nodes per input layer, number of hidden layers, and number of nodes per hidden layers give better model performance *up to a point where it no longer provides any improvement*. The following dataset 1.3 will test if adding more test cases will help push this bottleneck point and allow the use of a greater number of nodes per input layer, number of hidden layers, and number of nodes per hidden layer to reduce the worst-case error even lower.

Table 16 – Summary of the results for the optimal NN architecture using dataset 1.2

MAPE [%]	0.04
Worst-Case Error [%]	1.29
Train Time [hour]	9.9
Run Time [s]	0.03
Total Number of Test Cases	358722
Optimization Function	Nadam
Loss Function	MAPE
Activation Function	ReLU
Number of Nodes per Input Layer	3000
Number of Hidden Layers	2
Number of Nodes per Hidden Layer	100

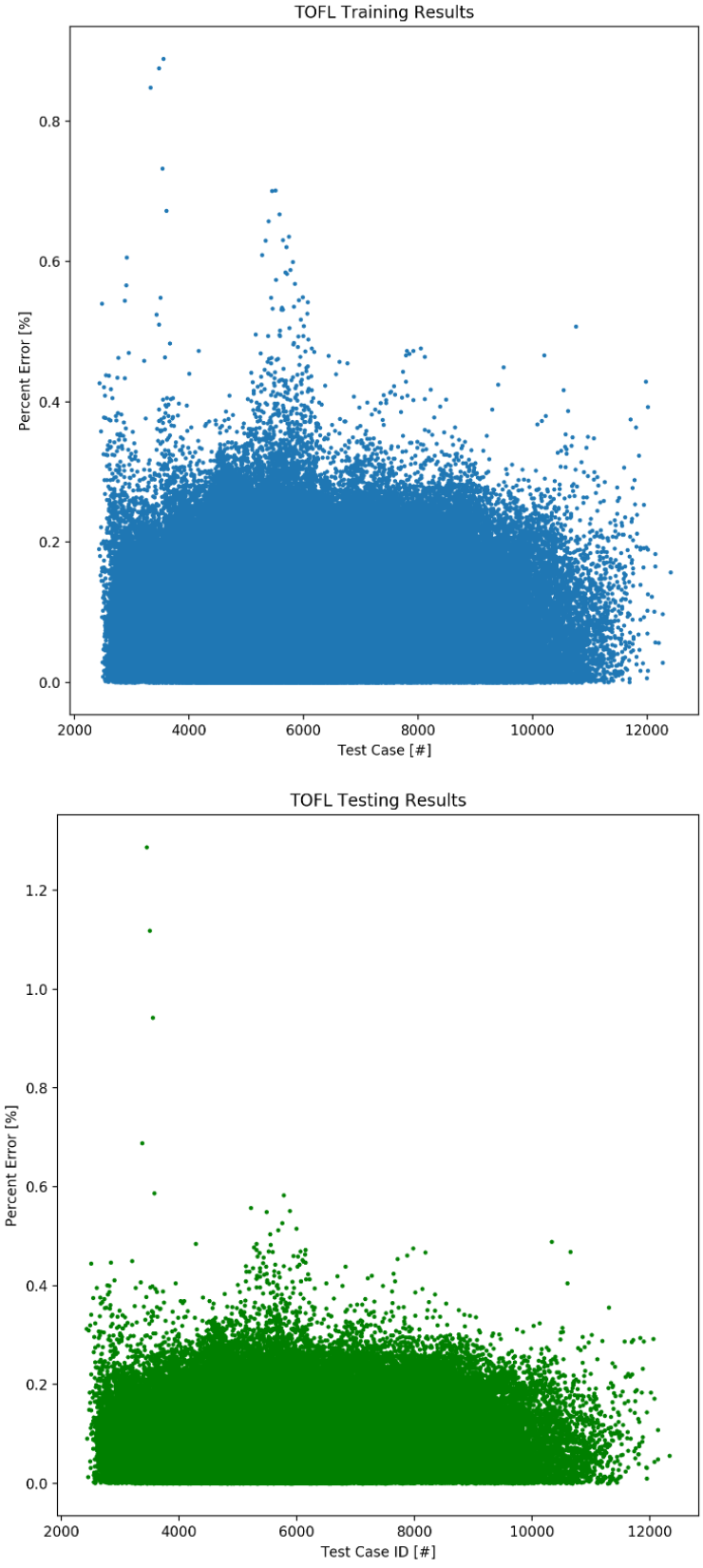


Figure 29 – Training and testing results for the optimal NN architecture using dataset 1.2

5.1.3 Dataset 1.3 Results

Dataset 1.3 was used to improve dataset 1.2's results and analyze the effect of increasing the size of the dataset on the results. The results from dataset 1.2 were analyzed and it was found that the test cases with the highest errors were values with high values of temperature and wind speed, as shown in Figure 30. Consequently, dataset 1.3 was produced using increased numbers of test cases for the temperature and wind parameters only. Table 11 lists the parameters used in dataset 1.3.

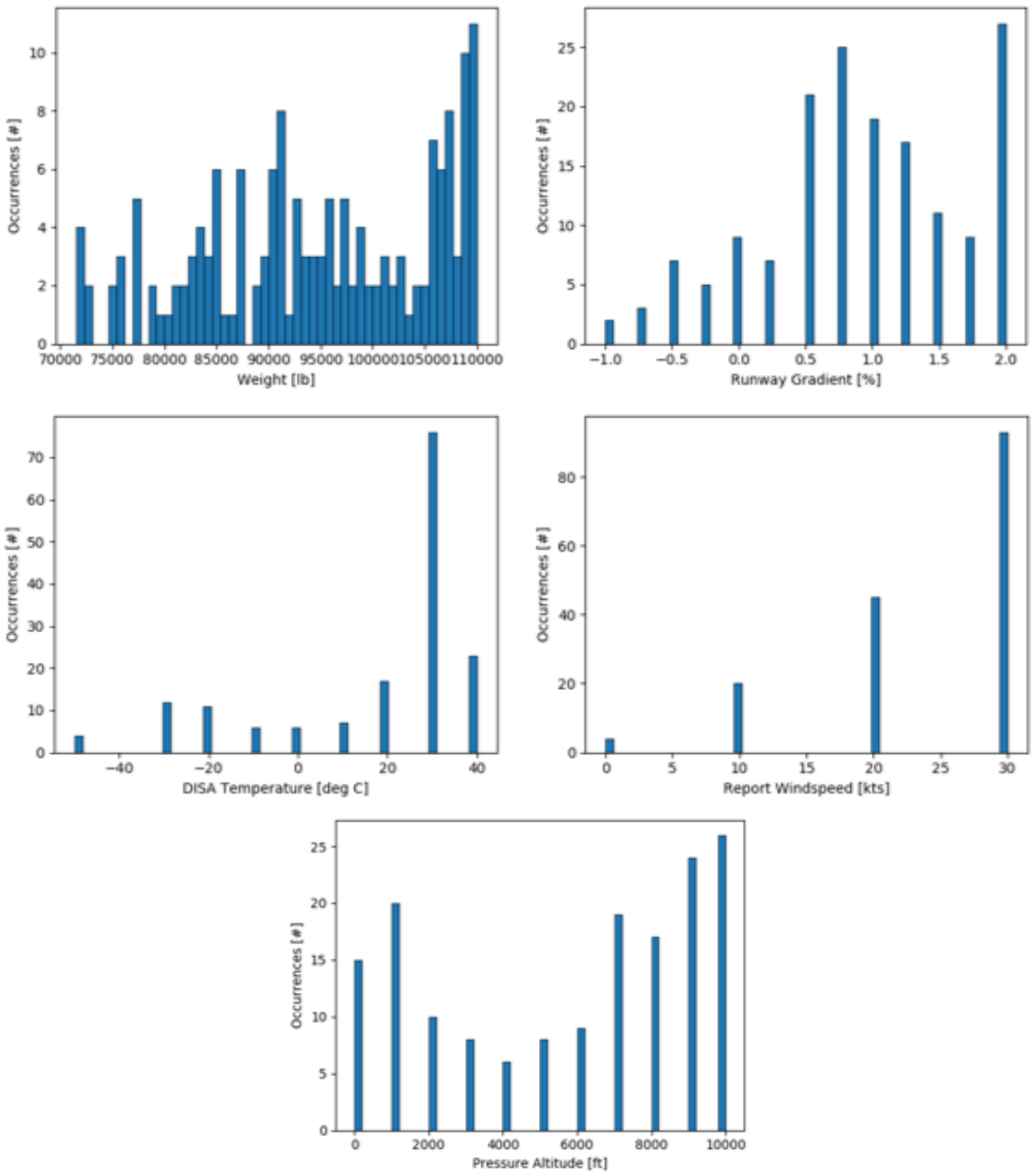


Figure 30 – Distribution of test cases with MAPE higher than 0.20 %

Tested models with dataset 1.3 did show improved predictions compared to the results from dataset 1.2.

Table 17 presents a summary of the NN architecture selected as the most appropriate for the dataset 1.3. This architecture has a lower number of hidden layers and nodes per layers than the best dataset 1.2 result, which leads to a slightly bigger MAPE of 0.05 %. Although this model has a higher MAPE, its value of worst-case percentage error as shown in Figure 31 is 0.61 %, which satisfies the NN requirements. The training time increased significantly from 9.9 hours to 19.4 hours but is still within the required 20 hours of training time. The increased dataset size also helped to solve the underfitting problem observed in the results obtained from dataset 1.2 as the predictions of Figure 31 generalize well to the testing results. Even though the dataset size has increased significantly, the run time to execute the prediction tool has not increased as both networks have similar structure (number of hidden layer and nodes).

Table 17 – Summary of the dataset 1.3 results

MAPE [%]	0.05
Worst-Case Error [%]	0.61
Train Time [hour]	19.4
Run Time [s]	0.02
Total Number of Test Cases	1294777
Optimization Function	Nadam
Loss Function	MAPE
Activation Function	ReLU
Number of Nodes per Input Layer	1000
Number of Hidden Layers	2
Number of Nodes per Hidden Layer	1000

Figure 31 – Training and testing results for the optimal NN architecture using dataset 1.3

5.2 NASA DASHlink Non-Deterministic Dataset Results

For the NN developed using the non-deterministic DASHlink dataset, two separate datasets were tested:

1. Dataset 2.1 uses flight data from a single aircraft and the limits imposed in Section 4.3.2 and detailed in Table 6; and
2. dataset 2.2 uses the full fleet of 12 aircraft and the limits shown in Table 7.

The reasoning behind separating the original dataset into two distinct datasets is to analyze if the NN can generalize takeoff performance predictions well enough to encompass the differences in performance between individual aircraft. The limits imposed on the original dataset, explained in Section 4.3.2, produce the dataset properties shown in Table 18, below. The number of total test cases for each dataset is considerably smaller than for the deterministic datasets 1.2 and 1.3, with 1,873 and 16,079 test cases respectively. Based on the value ranges of each dataset, it is observed that the flap drag and normalized wind speed parameters have very small ranges. This can negatively affect NN performance if the distribution of datapoints is concentrated in a localized area. For this reason, flap drag and normalized wind speed are removed as input parameters for the NN.

Table 18 – DASHlink dataset properties

Dataset	Input Parameters	Minimum	Maximum	Unit
2.1	Weight	7144	20904	lb
	Flap Drag	2443	2461	drag counts
	Pressure Altitude	-514	2202	feet
	TAT Temperature	-9	36	deg C
	Normalized Wind Speed	0	0	knots
	TOFL	4704	6499	feet
	Training Test Cases	1311		
	Testing Test Cases	562		
	Total Test Cases	1873		
2.2.	Weight	5904	21776	lb
	Flap Drag	2393	2596	deg
	Pressure Altitude	-643	2363	feet
	TAT Temperature	-32	40	deg C
	Normalized Wind Speed	0	0	knots
	TOFL	4702	6500	feet
	Training Test Cases	11255		
	Testing Test Cases	4824		
	Total Test Cases	16079		

5.2.1 Dataset 2.1 Results

The purpose for using dataset 2.1 is to develop a NN model that can satisfy the NN requirements and objectives from Section 4.2 while using readily available non-deterministic flight data for a single aircraft. Table 19 lists the architecture parameters that were tested in order to select the best NN model based on the established performance criteria. The test parameters were selected based on their ability for being applied to different datasets and dealing with high variance and data distribution. Table 20, Table 21, and Table 22 show the top 10 model architectures obtained based on best MAPE, best $p_{err,train}$, and best $p_{err,test}$. Results show that all tested optimization, loss, and activation functions produce results in the same ranges. One hidden layer demonstrates the best values of MAPE while two or more hidden layers give better $p_{err,train}$ and $p_{err,test}$ values. As the results are more constraining for the $p_{err,train}$ and $p_{err,test}$ values, the Table 21 and Table 22 results are prioritized as the most appropriate models. For the number of nodes, results vary, and no

general trend can be identified. Based on these results, the recommended NN architecture is model 153 (see Table 23 and Figure 32).

Table 19 – Summary of the test ranges for models using dataset 2.1 values

Optimization Function	adamax, adadelta, nadam
Loss Function	MAPE, MSLE
Activation Function	ReLU
Number of Nodes per Input Layer	1000, 3000, 5000
Number of Hidden Layers	1, 2, 3
Number of Nodes per Hidden Layer	10, 100, 1000

Table 20 – Summary of the top 10 dataset 2.1 results based on best MAPE

model	opt_f	loss_f	actvn_f	nb_hl	nd_pl	nd_il	MAPE [%]	p_err_{train} [%]	p_err_{test} [%]
3	adadelta	MAPE	relu	1	10	5000	3.81	22.36	21.10
1	adadelta	MAPE	relu	1	10	1000	3.81	21.64	20.22
2	adadelta	MAPE	relu	1	10	3000	3.81	22.53	21.02
87	adamax	MSLE	relu	1	100	5000	3.82	24.48	22.06
82	adamax	MSLE	relu	1	10	1000	3.82	22.65	20.16
145	nadam	MSLE	relu	2	10	1000	3.83	23.24	20.80
4	adadelta	MAPE	relu	1	100	1000	3.83	22.05	20.36
6	adadelta	MAPE	relu	1	100	5000	3.83	22.87	21.00
138	nadam	MSLE	relu	1	10	5000	3.83	23.07	20.67
58	adamax	MAPE	relu	1	100	1000	3.83	23.09	21.04

Table 21 – Summary of the top 10 dataset 2.1 results based on best p_err_{train}

model	opt_f	loss_f	actvn_f	nb_hl	nd_pl	nd_il	MAPE [%]	p_err_{train} [%]	p_err_{test} [%]
153	nadam	MSLE	relu	2	1000	5000	3.95	18.59	16.40
122	nadam	MAPE	relu	2	100	3000	4.27	18.76	16.47
67	adamax	MAPE	relu	2	100	1000	3.88	19.22	16.75
99	adamax	MSLE	relu	2	1000	5000	3.86	20.04	18.33
64	adamax	MAPE	relu	2	10	1000	4.12	20.24	19.19
114	nadam	MAPE	relu	1	100	5000	3.87	20.39	18.34
71	adamax	MAPE	relu	2	1000	3000	3.96	20.51	19.08
128	nadam	MAPE	relu	3	10	3000	3.86	20.56	18.60
79	adamax	MAPE	relu	3	1000	1000	3.94	20.69	18.63
10	adadelta	MAPE	relu	2	10	1000	3.88	20.75	18.71

Table 22 – Summary of the top 10 dataset 2.1 results based on best p_err$_{test}$

model	opt_f	loss_f	actvn_f	nb_hl	nd_pl	nd_il	MAPE [%]	p_err$_{train}$ [%]	p_err$_{test}$ [%]
153	nadam	MSLE	relu	2	1000	5000	3.95	18.59	16.40
122	nadam	MAPE	relu	2	100	3000	4.27	18.76	16.47
67	adamax	MAPE	relu	2	100	1000	3.88	19.22	16.75
99	adamax	MSLE	relu	2	1000	5000	3.86	20.04	18.33
114	nadam	MAPE	relu	1	100	5000	3.87	20.39	18.34
128	nadam	MAPE	relu	3	10	3000	3.86	20.56	18.60
79	adamax	MAPE	relu	3	1000	1000	3.94	20.69	18.63
10	adadelta	MAPE	relu	2	10	1000	3.88	20.75	18.71
118	nadam	MAPE	relu	2	10	1000	3.96	20.99	18.88
119	nadam	MAPE	relu	2	10	3000	4.09	20.94	18.94

The MAPE for the best NN model is of 3.95 % while the worst percentage error is 18.59 % for training and 16.40 % for testing. These values present the highest error values obtained in this research and possible explanations for these values are detailed in Chapter 6. Figure 32 illustrates the distribution of the NN model error for all trained test cases, where it is shown that most test case errors are lower than 10 % but that a small amount of test cases constrain the model's worst-case performance in the error range of 10 – 19 %.

Table 23 – Summary of the results for the optimal NN architecture using dataset 2.1

MAPE [%]	3.95
Worst-Case Error [%]	18.59
Train Time [hour]	0.48
Run Time [s]	0.09
Total Number of Test Cases	1873
Optimization Function	Nadam
Loss Function	MSLE
Activation Function	ReLU
Number of Nodes per Input Layer	5000
Number of Hidden Layers	2
Number of Nodes per Hidden Layer	1000

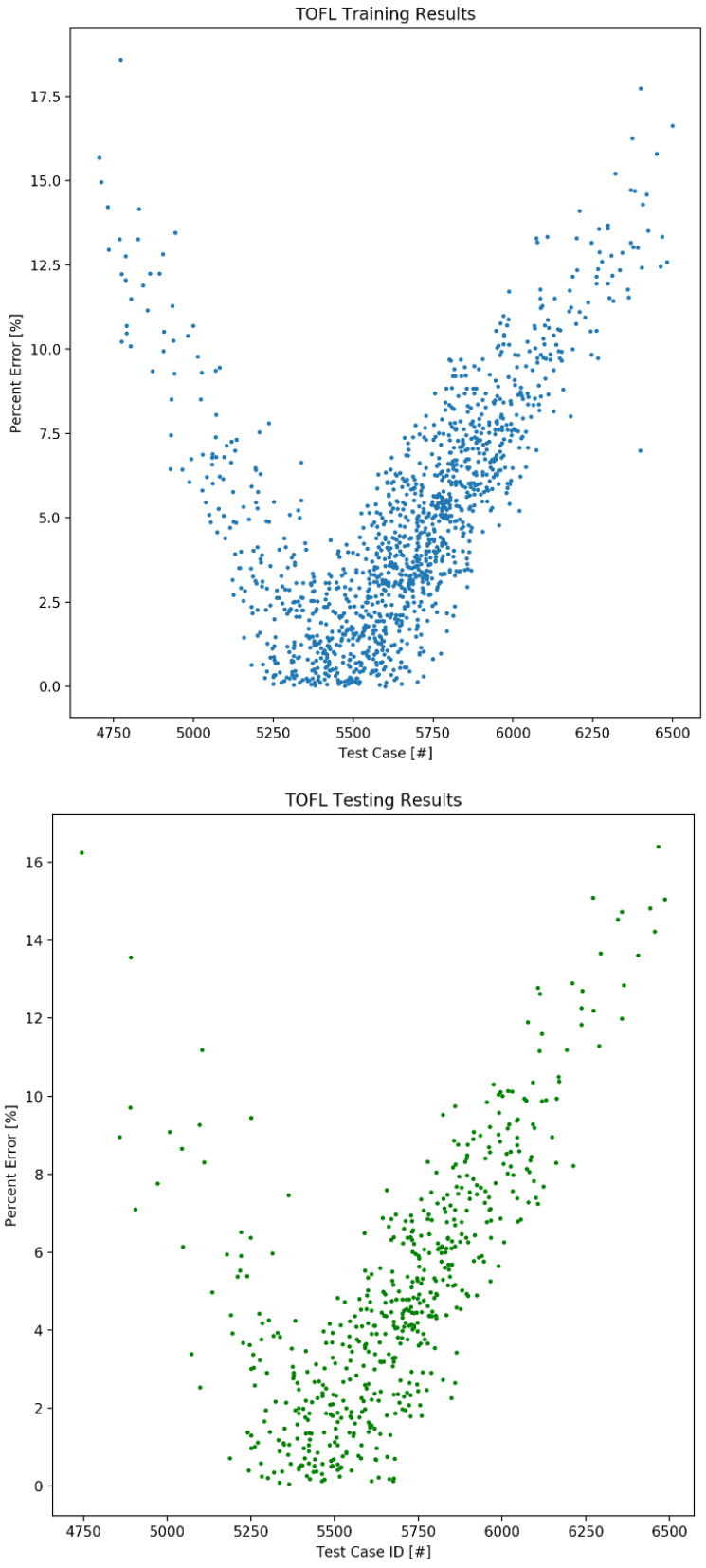


Figure 32 – Training and testing results for the optimal NN architecture using dataset 2.1

5.2.2 Dataset 2.2 Results

Dataset 2.2 was used to develop a NN model that can satisfy the NN requirements and objectives from Section 4.2 using the non-deterministic flight data for a fleet of aircraft of the same model rather than an individual aircraft as was the case for dataset 2.1. The model architectures tested are the same as for the dataset 2.1 models presented in Table 19, and Table 24, Table 25, and Table 26 show the top 10 model architectures obtained based on best MAPE, best $p_{err,train}$, and best $p_{err,test}$. Results show that the adadelta optimization function and the MAPE loss function are best at finding the lowest MAPE values. Higher numbers of hidden layers and nodes per layer generally demonstrate better $p_{err,train}$ and $p_{err,test}$ values. From the model architectures studied, those with the lowest worst-case training and testing errors are prioritized as the most appropriate models since this is the most conservative error as required for aerospace applications. Based on these results, the optimal NN architecture is model 106, detailed in Table 27 and Figure 33.

Table 24 – Summary of the top 10 dataset 2.2 results based on best MAPE

model	opt_f	loss_f	actvn_f	nb_hl	nd_pl	nd_il	MAPE [%]	p_err_{train} [%]	p_err_{test} [%]
20	adadelta	MAPE	relu	3	10	3000	3.82	22.98	22.17
4	adadelta	MAPE	relu	1	100	1000	3.83	21.43	21.28
1	adadelta	MAPE	relu	1	10	1000	3.84	21.28	21.27
24	adadelta	MAPE	relu	3	100	5000	3.84	20.31	19.67
10	adadelta	MAPE	relu	2	10	1000	3.84	22.31	21.79
82	adamax	MSLE	relu	1	10	1000	3.84	20.87	20.47
19	adadelta	MAPE	relu	3	10	1000	3.84	22.25	21.75
2	adadelta	MAPE	relu	1	10	3000	3.84	20.65	20.22
7	adadelta	MAPE	relu	1	1000	1000	3.84	22.53	21.83
3	adadelta	MAPE	relu	1	10	5000	3.85	21.82	21.74

Table 25 – Summary of the top 10 dataset 2.2 results based on best p_err_{train}

model	opt_f	loss_f	actvn_f	nb_hl	nd_pl	nd_il	MAPE [%]	p_err_{train} [%]	p_err_{test} [%]
106	adamax	MSLE	relu	3	1000	1000	4.00	18.50	18.00
66	adamax	MAPE	relu	2	10	5000	3.95	18.96	18.25
23	adadelta	MAPE	relu	3	100	3000	3.86	19.08	19.66
97	adamax	MSLE	relu	2	1000	1000	3.98	19.35	18.99
108	adamax	MSLE	relu	3	1000	5000	3.97	19.75	18.72
118	nadam	MAPE	relu	2	10	1000	3.90	19.77	19.39
62	adamax	MAPE	relu	1	1000	3000	3.88	19.84	19.17
127	nadam	MAPE	relu	3	10	1000	3.88	20.07	19.60
120	nadam	MAPE	relu	2	10	5000	3.93	20.08	19.24
99	adamax	MSLE	relu	2	1000	5000	3.96	20.15	20.01

Table 26 – Summary of the top 10 dataset 2.2 results based on best p_err_{test}

model	opt_f	loss_f	actvn_f	nb_hl	nd_pl	nd_il	MAPE [%]	p_err_{train} [%]	p_err_{test} [%]
106	adamax	MSLE	relu	3	1000	1000	4.00	18.50	18.00
66	adamax	MAPE	relu	2	10	5000	3.95	18.96	18.25
108	adamax	MSLE	relu	3	1000	5000	3.97	19.75	18.72
97	adamax	MSLE	relu	2	1000	1000	3.98	19.35	18.99
62	adamax	MAPE	relu	1	1000	3000	3.88	19.84	19.17
120	nadam	MAPE	relu	2	10	5000	3.93	20.08	19.24
118	nadam	MAPE	relu	2	10	1000	3.90	19.77	19.39
117	nadam	MAPE	relu	1	1000	5000	3.95	20.23	19.55
144	nadam	MSLE	relu	1	1000	5000	3.90	20.26	19.57
127	nadam	MAPE	relu	3	10	1000	3.88	20.07	19.60

The MAPE for the best NN model is 4.00 % while the worst percentage error is 18.50 % for training and 18.00 % for testing. Possible explanations for these values are detailed in the Chapter 6. Figure 33 shows the distribution of the NN model error for all trained test cases. Similar to the results of the previous dataset for a single aircraft, it may be observed that most test case errors are lower than 10 % but that a small amount of test cases constrain the model's worst-case performance in the error range of 10 – 18 %.

Table 27 – Summary of the results for the optimal NN architecture using dataset 2.2

MAPE [%]	4.00
Worst-Case Error [%]	18.50
Train Time [hour]	0.28
Run Time [s]	0.11
Total Number of Test Cases	16079
Optimization Function	adamax
Loss Function	MSLE
Activation Function	ReLU
Number of Nodes per Input Layer	1000
Number of Hidden Layers	3
Number of Nodes per Hidden Layer	1000

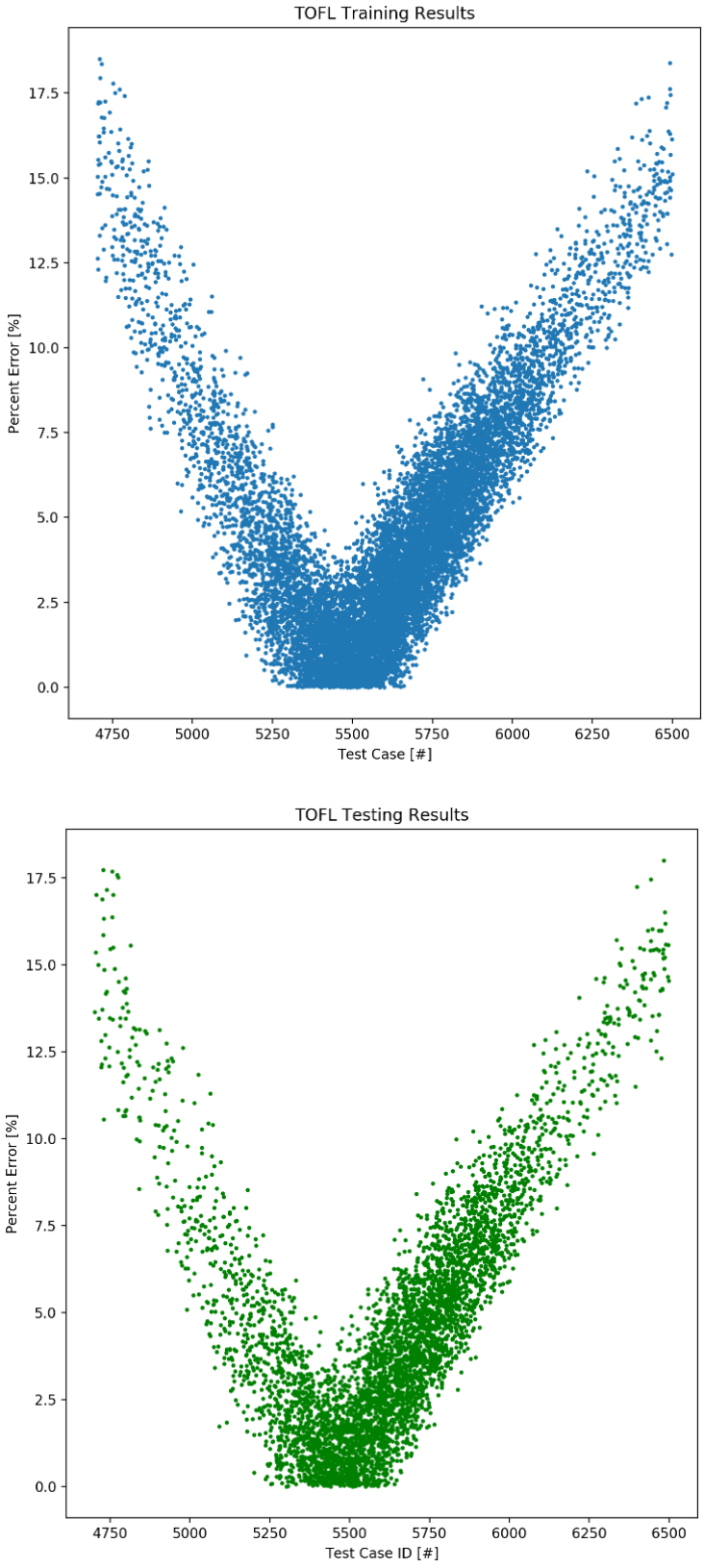


Figure 33 – Training and testing results for the optimal NN architecture using dataset 2.2

6. Discussion

6.1 Review of the Deterministic Dataset Results

The dataset 1.1 results were used to determine the most effective optimization, loss, and activation functions for the selected case study, which could then be used for a larger number of test cases in datasets 1.2 and 1.3 in order to encompass all TOD scenarios. Table 12, Table 13, and Table 14 show the best NN architectures to produce the lowest MAPE, worst-case percentage error for training ($p_{err,train}$), and worst-case percentage error for testing ($p_{err,test}$) respectively. After reviewing these results, it was observed that the ReLU activation function (and its derivatives elu and selu) consistently yielded the best results for all cases. Looking at the loss function results, MSE, MAE, and MAPE have best results for obtaining best MAPE and $p_{err,train}$ but MSE has the overall the best results as it ranks highest for MAPE, $p_{err,train}$, and $p_{err,test}$. For the optimization functions, adadelta and nadam demonstrated the best results for obtaining lowest values of MAPE, $p_{err,train}$, and $p_{err,test}$. These results validate the research listed in the Research Methodology Section on each architecture parameter. For this reason, it was decided to use these parameters for the architectures of datasets 1.2 and 1.3's architectures.

The dataset 1.2 results showed the impact of modifying the number of hidden layers and nodes per layers on the NN prediction accuracy for the complete envelope of possible test case ranges. Results showed that, for a given dataset, larger values of number of nodes per input layer, number of hidden layers, and number of nodes per hidden layers give better model performance *up to a point where it no longer provides any improvement*.

Dataset 1.3 was used to investigate if the point where adding more hidden layers and nodes no longer helps increase performance could be pushed further by adding more datapoints to the dataset. Results showed that it is possible to increase performance even further by adding more datapoints combined with higher values of hidden layers and nodes per layers. The observed drawback from this approach is the computational time and power required is increased and limited performance past a certain NN architecture and dataset size.

The optimal NN model developed using the dataset 1.3 values as seen in Figure 31 shows the final and best model to predict deterministic TOD values. Performance values from this model are listed in Table 17 and satisfy the requirements listed in Table 4.

6.2 Review of the Non-Deterministic Dataset Results

The NN models developed using the non-deterministic flight data from datasets 2.1 and 2.2 show that the provided data is enough to develop a NN model with MAPE errors lower than 4 % both for the performance

of a single aircraft as well as for a fleet of aircraft. On the other hand, these NN models had high values of worst-case percentage error (p_{err}), which were 18.59 % off from actual values. Comparing the values from the optimal NN model for the fleet of aircraft and the values for the single aircraft, with the bigger sized dataset of 2.2, a reduction in performance is observed when compared with the single aircraft results in dataset 2.1. This implies that the quality of the dataset is in question or that a better preprocessing operation would be required in future research.

The higher errors can be attributed to two main factors:

1. A dataset size that is too small
2. The presence of unknowns in the dataset for critical parameters

41,000 test cases were available from the original NASA DASHlink dataset. Following the preprocessing of the dataset, the resulting datasets were of 16079 test cases or smaller. It was experimentally found that, due to the two potential sources of errors just listed, the results from the non-deterministic datasets 2.1 and 2.2 show higher worst-case percentage errors than for the results from the deterministic datasets 1.2 and 1.3. Furthermore, as the NASA DASHlink dataset is open source, there were parameters considered critical to this study that were not specified such as the aircraft model, number of service hours, and the payload weight. This is important information since the dataset could potentially be combining different aircraft models with differing levels of operational capabilities, though with the provided data, it is not possible to separate these different cases. The same could be said for the number of service hours and payload weight. Another fact that could potentially explain these results is the degree of sensitivity of the flight data sensors. These were also not specified and if they were available for study they could possibly help explain outlier or erroneous datapoints which negatively influences the training results.

Nevertheless, the results obtained for the non-deterministic data show that a NN can be developed with acceptable MAPE for aircraft performance flight data. In future research, it would be worthwhile to investigate a similar study with a non-deterministic dataset that does not have the two limitations observed in the DASHlink dataset.

6.3 Comparing Deterministic and Non-Deterministic Results

Table 28 shows a summary of all the results obtained for the developed NNs, including deterministic and non-deterministic results. The results show that deterministic data can produce NN models that are much more robust than non-deterministic data. On the other hand, the greatest benefit of using non-deterministic data for this thesis is that it can allow the extraction of complex patterns for real-time empirical flight data, which may not be fully covered in the equivalent deterministic data.

The run time values show the performance of the prediction tool that makes use of the trained weights of the NN. An interesting discovery is that the run time is not a function of the dataset size but rather of the number of weights in the NN. Consequently, run time is a function of number of hidden layers and nodes per layer. This allows very large datasets to be used as effectively as smaller datasets using the prediction tool.

Table 28 – Comparison of deterministic and non-deterministic results

Dataset	MAPE [%]	Worst-Case Error [%]	Training Time [hour]	Run Time [s]	Total # Test Cases
1.1	1.8982	9.76	0.25	0.01	1000
1.2	0.0439	1.29	9.90	0.03	358722
1.3	0.0595	0.61	19.40	0.02	1294777
2.1	3.9452	18.59	0.48	0.09	1873
2.2	4.0030	18.50	0.28	0.11	16079

6.4 Summary of the Findings

Seven findings were determined from the results. Some can be associated with all ML regression optimization applications and some are more specific to the case study at hand in this research. These findings are presented in the present section.

Dealing with Sparsity Errors

Data sparsity is usually not desired as it means that information is missing that might be important to developing the pattern to the output solution. The chart on the left of Figure 34 is an example of data sparsity, where there is little data available for aircraft weighing less than 10,000 lb. Sparsity errors are a result of missing datapoints in the dataset and can lead to suboptimal NN performance. Allison et al. [78] review the problem and explains that when adding more data in the gaps, model accuracy increases. The results obtained from experimental models validate this statement. For the case of the deterministically generated datasets of this research, smaller increment sizes can be selected between each datapoint which allows to control the size and distribution of the dataset which best suits the needs of the designer. It is also possible to select the distribution and values of input parameters for which more data is required. The results of dataset 1.2 and 1.3 show that, when adding more datapoints for only the input parameters that had the highest errors, the model error can be reduced. For the case of the non-deterministic dataset, and all non-deterministic datasets in general, the provided dataset must be used or, if possible, more data can be acquired to solve the problem. Allison et al. further analyze different ML techniques that can help increase model performance when filling in the data sparsity when acquiring new data is not possible. Filtering techniques

exist to bridge gaps in the data (which are out of the scope of the current research) and different regularization methods exist to preprocess the data to yield better model performance. One such preprocessing operation that was used successfully in this research is the normalization of the dataset values.

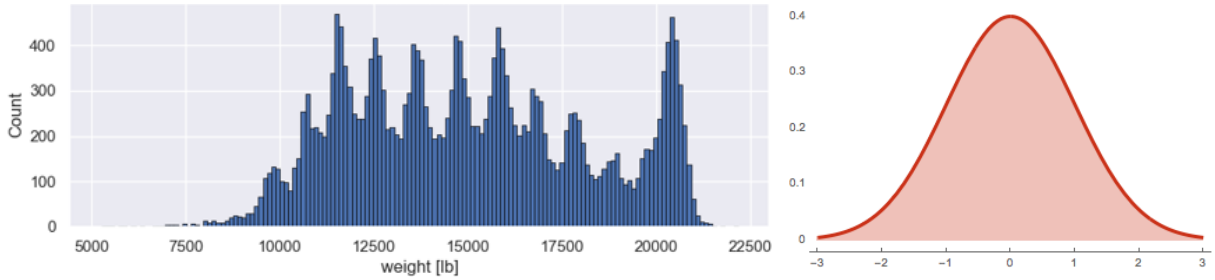


Figure 34 – Difference between a sparse and gaussian distribution

Generalization vs Specificity

One of the main hurdles when developing ML models is to be able to create NNs that are good at generalization. Figure 35 shows two different sample models. The model on the left (general model) is able to make predictions for a much wider scope of input parameters, but its worst-case percentage error is of 7.86 %. The model on the right (specific model) has a better worst-case percentage error of 0.92% but is only able to make predictions for a narrower scope of input parameters. The best-case model would be a model that is capable of making predictions for the range of input parameters of the general model with the precision of the specific model. In this research, this issue was solved by having multiple specific models with high accuracy that can be used depending on what range of input parameters and output parameters are of interest. Other methods exist to solve the problem. One method [27] uses an expert system with a gating logic which looks at multiple specific NNs in order to optimize the overall system's efficiency and accuracy. Another method [79] uses a type of NN called a Recurrent Neural Network (RNN) which can significantly improve the generalization performance of trained recurrent networks.

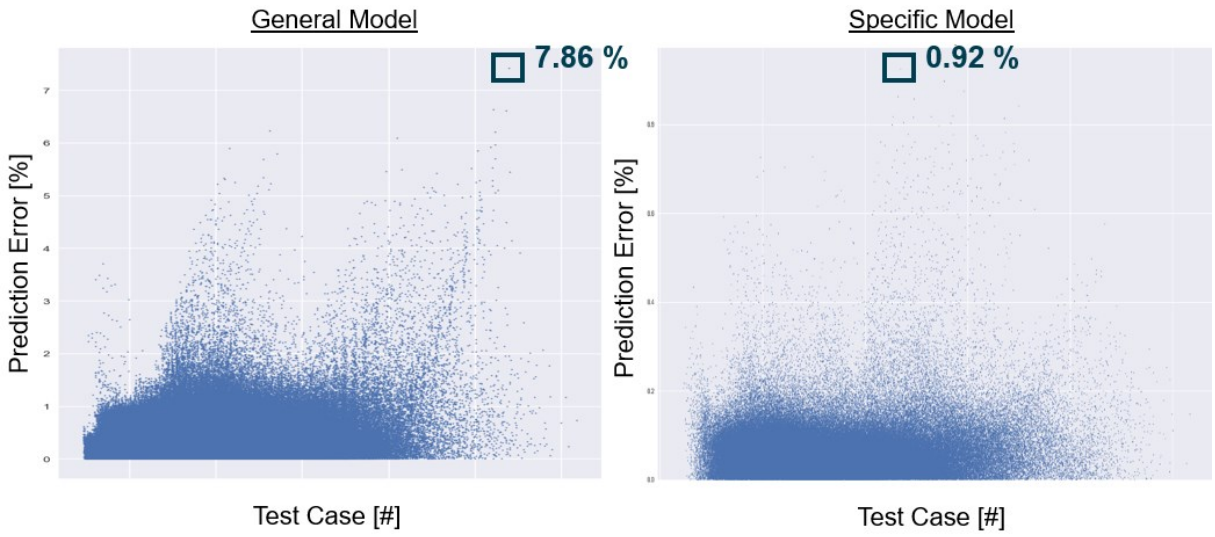


Figure 35 – Comparison of a general model and a specific model

No Free Lunch Theorem

The "no free lunch theorem" [80] states that no machine learning algorithm is universally any better than any other. This means that each implementation of ML methodologies to a real-world application problem must be optimized for this very specific problem and its data distribution. Wolpert [80] shows that generalization of one specific NN's results to another more general NN (with a large dataset) does not scale well. This explains why in general, when adding more data to increase model performance, it is exponentially more expensive with respect to the amount of data required and computational power. This observation can be a leading cause for reaching bottlenecks when developing ML models for novel applications. For the selected case study, it is observed that dataset 1.3 has an increased performance when compared with dataset 1.2 but its training time is over 19 hours instead of 9 hours.

Validating NN Results with Physical Phenomena

An important observation that was made in this research is that the physical interpretation of the NN values must always be validated for different reasons. The values produced by the NN could be validated simply to make sure that they are physically possible, to make sure they are desirable, or to discover new deterministic patterns. For all the deterministic datasets observed in this research, only values that did not give warning or error messages were used, which are an indication that the calculated TOD is close to operational limits or outside the physically possible values. For the non-deterministic datasets, it was explained in previous sections that much of the flight sensor data had to be omitted in the training data as the values were judged physically impossible. As TOD predictions that are shorter than the actual TOD values can result is dangerous situations, which are not desirable, all prediction values that are lower than

the actual solution must be processed in a way that can incorporate more safety into the model. Figure 36 shows one way to deal with this problem, by adding a safety margin to all values to shift them all upwards by a value that makes all predictions positive when compared with the actual values. This is the most conservative solution and can affect the network's prediction accuracy. A more sophisticated approach could also be used which could employ an expert system to decide what safety margin to apply based on each range of values. When using non-deterministic data, the NN results can be used to find new deterministic relations. This can be done by looking at the trained weight matrix of the NN and making interpretations. For example, the parameters having the highest weights could be used to update existing equations describing the physical system under study.

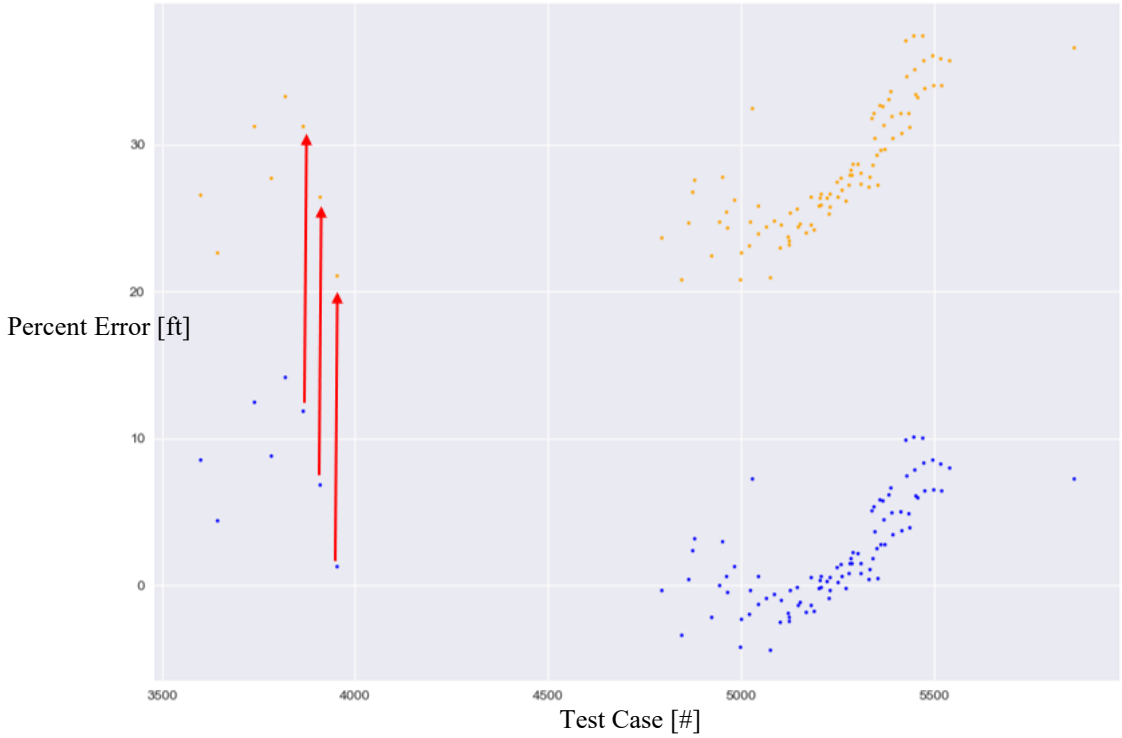


Figure 36 – NN model that adds a safety margin to all predictions

Selecting the Appropriate Performance Indicator

It was found that the selection of the appropriate performance metric used to evaluate the NN's success is case specific. Off the shelf performance metrics like MSE or MAE can be used, if relevant, but in most cases prior knowledge of the deterministic system under study is required as described in Section 3.2, in order to be able to choose an appropriate metric for the intended use. For example, in the case study, it is more relevant to look at worst-case percentage errors than MAPE, in order to be as conservative as possible across the whole scope of test cases.

How Much Expert Knowledge is Actually Required?

One of the main advantages of using machine learning is to reduce the need for expert-level knowledge in developing a model. For the selected case study, it was found that a basic understanding of the parameters affecting takeoff distance as well as the overall expected behavior of aircraft in takeoff scenarios were required to develop the NNs successfully. This background information was covered in Section 3. This shows that for the case study, ML does not *remove* the need for expert-level knowledge completely and that, at best, a basic understanding of the main drivers of the system under study are still required.

Using Deterministic Models to Generate ML Datasets

All the NN models that were developed using the deterministic datasets (datasets 1.1, 1.2, and 1.3) produced results which met the defined NN requirements and objectives. This can be generally attributed to the fact that for these datasets it was possible to control the dataset size and distribution in order to give better predictive accuracy. This shows that using existing deterministic models to generate ML datasets can be very practical, as they allow for more control of the dataset properties.

7. Conclusion

The objective of this research was to investigate if Artificial Neural Networks could be used effectively as an alternative to current aircraft performance models. Based on the results obtained from the deterministic dataset generated using an existing TOD model, it can be stated that the NN's results were in line with the existing model. For the optimal NN model, these results matched the existing TOD model values within 0.61 % or lower and had an overall average model error of 0.05 %. Training Neural Networks using data from an existing deterministic model proved to be very effective, mainly due to its ability to generate different dataset distributions and dataset sizes. For the NN results based on the non-deterministic flight data, it was also found that the NN could produce acceptable predictions. For the optimal NN model generated using the flight data, the average error for all predicted test cases demonstrated performance of 3.94 %, but the network was most constrained by outlier test cases in error ranges of 18.59 %. Reasons to explain the unacceptable results were reviewed and it is believed that with a similar dataset with an increased size and more detailed information on each data value, acceptable performance could potentially be obtained. Future research could investigate a more in-depth rationale to why these specific outlier test cases produce higher error values.

In future research, it would be of value to analyze if it would be possible to get performance *even more accurate than the existing deterministic model by using a more comprehensive non-deterministic dataset*. The NN results found using the deterministic TOD dataset showed almost no error, but even if this network

would be perfectly modeled, it could never give results better than the ones of the deterministic TOD values. This fact shows the value in being able to model non-deterministic data, since much of real-life empirical data is non-deterministic (i.e. flight data from onboard sensors).

Another avenue of interest would be to review the performance of existing NN models and apply them to the case study that is the subject of this thesis. This research focused on developing a proof of concept using a straightforward multiple layer feedforward backpropagation neural network. More sophisticated NN models exist that are tailored to solve specific problem sets. Scikit-Learn, a Python data processing library, provides a flowchart (see Figure 37) for selecting the appropriate NN model based on the ML problem at hand and the available dataset. Lasso, ElasticNet, SVR, Ridge Regression, or Ensemble Regressors could potentially be promising NNs for TOD prediction.

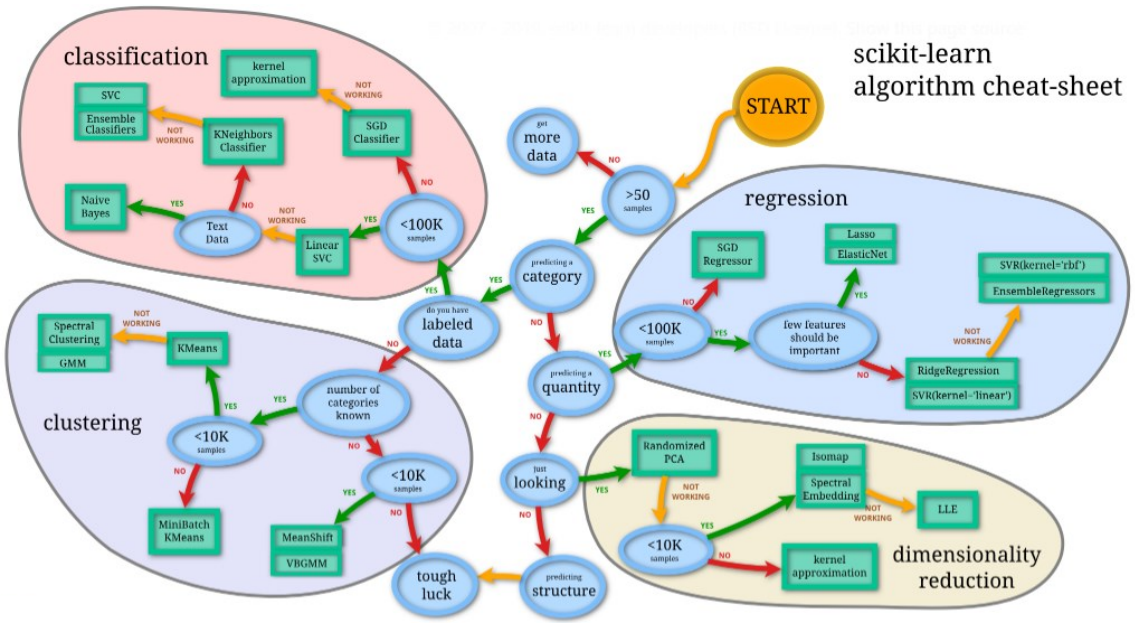


Figure 37 – Scikit-learn's cheat-sheet for selecting existing NN models [81]

8. References

[1] A. K. Noor, "Nondeterministic Approaches and Their Potential for Future Aerospace Systems," NASA Langley Research Center, Hampton, 2001.

[2] T. A. Cruse, "Non-Deterministic, Non-Traditional Methods (NDNTM)," NASA Glenn Research Center, Cleveland, 2001.

[3] S. J. Rey, "Mathematical Models in Geography," *International Encyclopedia of the Social & Behavioral Sciences (Second Edition),* vol. 14, no. 2, pp. 785-790, 2015.

[4] W. Schwarzacher, "Chapter 3 Deterministic models," *Developments in Sedimentology,* vol. 19, pp. 37-59, 2008.

[5] M. Urquidi-Macdonald and D. D. Macdonald, "Performance Comparison Between a Statistical Model, a Deterministic Model, and an Artificial Neural Network Model for Predicting Damage for Pitting Corrosion," *Journal of Research of the National Institute of Standards and Technology,* vol. 99, no. 4, pp. 495-504, 1994.

[6] D. N. Mavris and J. S. Schutte, "Application of Deterministic and Probabilistic System Design Methods and Enhancements of Conceptual Design Tools for ERA Project Final Report," Langley Research Center, Hampton, 2016.

[7] D. Gonze, J. Hallow and A. Goldbeter, "Deterministic versus Stochastic Models for Circadian Rhythms," *Journal of Biological Physics,* vol. 28k, pp. 637-653, 2002.

[8] R. M. Vogel, "Stochastic and Deterministic World Views," *Journal of Water Resources Planning and Management,* vol. 125, no. 6, pp. 311-313, 1999.

[9] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Montreal: MIT Press, 2015.

[10] S. Russell and P. Norvig, Artificial intelligence - A modern approach, Upper Saddle River, New Jersey: Pearson Education, Inc., 2010.

[11] S. Herbert and A. Newell, "Human problem solving: The state of the theory in 1970," *American Psychologist,* vol. 26, no. 2, pp. 145-159, 1971.

[12] G. W. Ernst, "GPS and Decision Making: An Overview," *Theoretical Approaches to Non-Numerical Problem Solving,* vol. 28, pp. 59-107, 1970.

[13] L. Fausett, Fundamentals of neural networks - Architectures, algorithms, and applications, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1994.

[14] I. Liljeqvist, "The Essence of Artificial Neural Networks," 20 March 2016. [Online]. Available: https://medium.com/@ivanliljeqvist/the-essence-of-artificial-neural-networks-5de300c995d6. [Accessed 18 September 2019].

[15] W. S. McCulloch and W. H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics,* vol. 5, no. 1, pp. 115-133, 1943.

[16] A. N. Whitehead and B. Russell, Principia Mathematica, Cambridge: Cambridge University Press, 1910.

[17] A. Turing, "Computing Machinery and Intelligence," *Mind,* vol. 59, no. 236, pp. 433-460, 1950.

[18] D. O. Hebb, The Organization of Behavior, Montreal: John Wiley & Sons, 1949.

[19] D. E. Rumelhart and J. L. M. G. E. hinton, "A General Framework for Parallel Distributed Processing," 1986.

[20] M. Baroni and A. Kilgarriff, "WebBootCaT: a web tool for instant corpora," *Computational Lexicography and Lexicology,* 2006.

[21] D. Yarowsky, "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods," in *33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, 1995.

[22] M. J. Pedelty, "A Review of the Field of Artificial Intelligence and its Possible Applications to NASA Objectives," School of Government and Public Administration, Washington, D.C., 1965.

[23] H. J. Dunn and R. C. Montgomery, "A Moving Window Parameter Adaptive Control System for the F8-DFBW Aircraft," *IEEE Transactions on Automatic Control,* Vols. AC-22, no. 5, pp. 788-795, 1977.

[24] Massachusetts Institute of Technology Lincoln Laboratory, "DARPA Neural Network Study Final Report," Tactical Technology Office of the U.S. Defense Advanced Research Projects Agency (DARPA/TTO), Lexington, MA, 1989.

[25] L. Harrison, P. Saunders and J. Janowitz, "Artificial Intelligence with Applications for Aircraft," Federal Aviation Administration (FAA), Springfield, Virginia, 1994.

[26] W. H. Cheung, *Neural Network Aided Aviation Fuel Consumption Modeling,* M.S. thesis, Blacksburg, Virginia: Virginia Polytechnic Institute and State University, 1997.

[27] D. L. Simon and T. W. Long, "Adaptive Optimization of Aircraft Engine Performance Using Neural Networks," NASA - U.S. Army Research Laboratory, Cleveland, OH, 1995.

[28] D. J. Linse and R. F. Stengel, "Identification of Aerodynamic Coefficients Using Computational Neural Networks," in *Aerospace Sciences Meeting (AIAA)*, Reno, NV, 1992.

[29] B. S. Kim and A. J. Calise, "Nonlinear Flight Control Using Neural Networks," *Journal of Guidance, Control and Dynamics,* vol. 20, no. 1, pp. 26-33, 1997.

[30] A. J. Calise and R. Rysdyk, "Nonlinear adaptive flight control using neural networks," *Control Systems,* vol. 18, no. 6, pp. 14-25, 1999.

[31] W. E. Failer and S. J. Schreck, "Neural Networks: Applications and Opportunities in Aeronautics," *Progress in Aerospace Science,* vol. 32, pp. 433-456, 1996.

[32] D. Mackall, S. Nelson and J. Schumann, "Verification and Validation of Neural Networks for Aerospace Systems," NASA Ames Research Center, Moffett Field, California, 2002.

[33] National Research Council, "Decadal Survey of Civil Aeronautics: Foundation for the Future," The National Academies Press, Washington, DC, 2006.

[34] A. Savran, R. Tasaltin and Y. Becerikli, "Intelligent adaptive nonlinear flight control for a high performance aircraft with neural networks," *ISA Transactions,* vol. 45, no. 2, pp. 225-247, 2006.

[35] U. J. Pesonen, "Adaptive Neural Network Inverse Controller for General Aviation Safety," *Journal of Guidance, Control, and Dynamics,* vol. 27, no. 3, 2004.

[36] T. Lee and Y. Kim, "Nonlinear Adaptive Flight Control Using Backstepping and Neural Networks Controller," *Journal of Guidance, Control and Dynamics,* vol. 24, no. 4, pp. 675-682, 2001.

[37] E. Ogretim, W. Huebsch and A. Shinn, "Aircraft Ice Accretion Prediction Based on Neural Networks," *Journal of Aircraft,* vol. 43, no. 1, pp. 233-240, 2006.

[38] C.-T. Weng, C. E. Lan and M. Guan, "Aerodynamic Analysis of a Jet Transport in Windshear Encounter During Landing," *Journal of Aircraft,* vol. 43, no. 2, pp. 419-427, 2006.

[39] O. Levinski, "Prediction of Buffet Loads Using Artificial Neural Networks," Defence Science & Technology Organisation (DSTO) Aeronautical and Maritime Research Laboratory, Victoria, Australia, 2001.

[40] D. Kim and K. Pechaud, "Improved Methodology for the Prediction of the Empennage Maneuver In-Flight Loads of a General Aviation Aircraft Using Neural Networks," Department of Transportation, Federal Aviation Administration, Springfield, VA, 2001.

[41] D. Tolani, M. Yasar, A. Ray and V. Yang, "Anomaly Detection in Aircraft Gas Turbine Engines," *Journal of Aerospace Computing, Information, and Communication,* vol. 3, no. 2, pp. 44-51, 2006.

[42] S. Kottapalli, "Neural-Network-Based Modeling of Rotorcraft Vibration for Real-Time Applications," in *AIAA Modeling and Simulation Technologies*, Denver, CO, 2000.

[43] H. S. Bruner, "The Analysis of Performance Flight Test Data Using a Neural Network," in *40th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, 2002.

[44] A. A. Trani, F. C. Wing-Ho, G. Schilling, H. Baik and A. Seshadri, "A Neural Network Model to Estimate Aircraft Fuel Consumption," in *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, Chicago, Illinois, 2004.

[45] National Research Council, "Autonomy Research for Civil Aviation: Toward a New Era of Flight," The National Academies Press, Washington, DC, 2014.

[46] C. Wilkinsin, J. Lynch, R. Bharadwaj and K. Woodham, "Verification of Adaptive Systems," Federal Aviation Administration (FAA), Springfield, Virginia, 2016.

[47] S. Bhattacharyya, D. Cofer, D. Musliner, J. Mueller and E. Engstr, "Certification Considerations for Adaptive Systems," NASA, Cedar Rapids, 2015.

[48] B. Arizmendi, T. Bellosta, A. del Val, G. Gori, J. Reis and M. Prazeres, "On Real-time Management of On-board Ice Protection Systems by means of Machine Learning," in *AIAA Aviation 2019*, Dallas, 2019.

[49] B. Zhou, "Towards a real-time in-flight ice detection system via computational aeroacoustics and bayesian neural networks," in *AIAA Aviation 2019*, Dallas, 2019.

[50] Y. Dong, "An application of Deep Neural Networks to the in-flight parameter identification for detection and characterization of aircraft icing," *Aerospace Science and Technology Journal,* vol. 77, no. 1, pp. 34-49, 2018.

[51] F. Caliskan and C. Hajiyev, "A review of in-flight detection and identification of aircraft icing and reconfigurable control," *Progress in Aerospace Sciences,* vol. 60, no. 1, pp. 12-34, 2013.

[52] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using recurrent neural networks (RNN)," in *2016 Integrated Communications Navigation and Surveillance Conference*, Fairfax, 2016.

[53] C. Sbarufatti and M. Giglio, "Performance Qualification of an On-Board Model-Based Diagnostic System for Fatigue Crack Monitoring," *Journal of the American Helicopter Society,* vol. 62, no. 4, pp. 1-10, 2017.

[54] X. Fu, H. Luo, S. Zhong and L. Lin, "Aircraft engine fault detection based on grouped convolutional denoising autoencoders," *Chinese Journal of Aeronautics,* vol. 32, no. 2, pp. 296-307, 2019.

[55] E. Mangortey, *Predicting The Occurrence Of Ground Delay Programs And Their Impact On Airport And Flight Operations,* M.S. thesis, Atlanta, Georgia: Georgia Institute of Technology, 2019.

[56] G. Dard, *Application of data fusion and machine learning to the analysis of the relevancy of recommended flight reroutes,* M.S. thesis, Atlanta, Georgia: Georgia Institute of Technology, 2019.

[57] C. E. V. Gallego, V. F. G. Comendador, F. J. S. Nieto, G. O. Imaz and R. M. A. Valdés, "Analysis of air traffic control operational impact on aircraft vertical profiles supported by machine learning," *Transportation Research Part C,* vol. 95, no. 1, pp. 883-903, 2018.

[58] X. Bertrand, F. Tost and S. Champagneux, "Wing Airfoil Pressure Calibration with Deep Learning," in *AIAA Aviation 2019*, Dallas, 2019.

[59] Boston Consulting Group, "Aerospace and AI - Bringing together Montreal's distinctive strengths," Aero Montreal, Montreal, 2019.

[60] D. McNally, "Dynamic Weather Routes: Two Years of Operational Testing at American Airlines," *AIAA Air Traffic Control Quarterly Journal,* vol. 23, no. 1, 2015.

[61] T. Canada, "Part V - Airworthiness Manual Chapter 525," Government of Canada, 26 09 2019. [Online]. Available: https://www.tc.gc.ca/en/transport-canada/corporate/acts-regulations/regulations/sor-96-433/part5-standards-525-sub-ab-1739.htm#525.113. [Accessed 1 1 1].

[62] Airbus, Getting to Grips with Aircraft Performance, Blagnac: Airbus Customer Services, 2002.

[63] M. Asselin, An Introduction to Aircraft Performance, Kingston, Ontario: American Institute of Aeronautics and Astronautics Inc. (AIAA), 1997.

[64] T. R. Yechout, Introduction to aircraft flight mechanics: Performance, static stability, dynamic stability, classical feedback control, and state-space foundations, Reston, Virginia: American Institute of Aeronautics and Astronautics Inc., 2014.

[65] G. Ruijgrok, Elements of Airplane Performance, Delft, Netherlands: Delft University Press, 1990.

[66] J. Williams, Aircraft Performance - Prediction Methods and Optimization, Paris, France: Advisory Group for Aerospace Research & Development, 1973.

[67] Performance Training Group, Jet Transport Performance Methods, Seattle: Boeing, 2009.

[68] NASA , "Sample Flight Data," 28 November 2018. [Online]. Available: https://c3.nasa.gov/dashlink/projects/85/.

[69] Maclobio, "CleanPNG," [Online]. Available: https://www.cleanpng.com/png-artificial-neural-network-neuron-biological-neural-1616821. [Accessed 10 December 2018].

[70] F. Chollet, "Keras Documentation," 19 September 2019. [Online]. Available: https://keras.io/why-use-keras/.

[71] P. Grover, "5 Regression Loss Functions All Machine Learners Should Know," Heartbeat, 5 June 2018. [Online]. Available: https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0.

[72] A. D. Myttenaere, B. Golden, B. L. G. and F. Rossi, "Mean Absolute Percentage Error for regression models," *Neurocomputing,* vol. 192, pp. 38-48, 2017.

[73] S. Ruder, "An overview of gradient descent optmization alogrithms," 19 January 2016. [Online]. Available: https://ruder.io/optimizing-gradient-descent/.

[74] R. S. Sutton, " Two problems with backpropagation and other steepest-descent learning procedures for networks," in *8th Annual Conference on Cognitive Science Society*, 1986.

[75] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks : The Official Journal of the International Neural Network Society,* vol. 12, no. 1, pp. 145-151, 1999.

[76] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2015.

[77] GreyAtom, [Online]. Available: https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjWrfSTh5_mAhXvct8KHS-2BREQjRx6BAgBEAQ&url=%2Furl%3Fsa%3Di%26source%3Dimages%26cd%3D%26ved%3D2ahUKEwid6e-Qh5_mAhWrVN8KHS4_CVkQjRx6BAgBEAQ%26url%3Dhttps%253A%252F%252Fmedium.com%252.

[78] B. Allison, D. Guthrie and L. Guthrie, "Another Look at the Data Sparsity Problem," in *International Conference on Text, Speech and Dialogue*, Berlin, 2006.

[79] C. Giles and C. Omlin, "Pruning recurrent neural networks for improved generalization performance," *IEEE Transactions on Neural Networks,* vol. 5, no. 5, pp. 848-851, 1994.

[80] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions of Evolutionary Computation,* vol. 1, no. 1, pp. 67-82, 1997.

[81] "Scikit-Learn Documentation," scikit-learn developers (BSD License), 2019. [Online]. Available: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html. [Accessed 28 November 2019].

[82] J. Roskam, "Part VII: Determination of stability, control and performance characteristics: FAR and military requirements," in *Airplane Design*, Ottawa, Kansas, Roskam Aviation and Engineering Corporation, 1988, p. 372.

[83] D. P. Raymer, "Chapter 17. Performance and Flight Mechanics," in *Aircraft Design: A Conceptual Approach*, Washington, DC, American Institute of Aeronautics and Astronautics, Inc. (AIAA), 1992, p. 46.

[84] A. Nuic, D. Poles and V. Mouillet, "BADA: An advanced aircraft performance model for present and future ATM systems," *Internationl Journal of Adaptative Control and Signal Processing,* pp. 850-866, 28 June 2010.

[85] A. Nuic, C. Poinsot and M.-G. Iagaru, "Advanced aircraft performance modeling for ATM: Enhancements to the BADA model," in *24th Digital Avionics System Conference*, Washington, D.C., 2005.

[86] R. D. Kimberlin, Flight Testing of Fixed-Wing Aircraft, American Institue of Aeronautics and Astronautics (AIAA).

[87] General Aviation Manufacturers Association, *Specification for Pilot's Operating Handbook,* Washington, D.C.: General Aviation Manufacturers Association, 1975.

[88] Federal Aviation Administration, *Pilot's Handbook of Aeronautical Knowledge,* Oaklahoma City: U.S. Department of Transportation, 2008.

[89] Federal Aviation Administration, *AC 25.1571-1 - Airplane Flight Manual,* U.S. Department of Transportation, 2012.

[90] J. Principe, N. Euliao and C. Lefeebvre, Neural and Adaptive Systems, Fundamental through Simulation, John Wiley, 2000.

[91] P. Wasserman, Neural Computing: Theory and Practice, New York: Van Nostrand Reinhold Co., 1989.

[92] J. Dayhoff, Neural Network Architectures: An Introduction, New York: Van Nostrand Reinhold Co., 1990.

[93] S. A. Oke, "A literature review on artificial intelligence," *International Journal of Information and Management Sciences,* vol. 19, no. 4, pp. 535-570, 2008.

[94] A. Megatroika, M. Galinium, A. Mahendra and N. Ruseno, "Aircraft Anomaly Detection Using Algorithmic Model and Data Model Trained on FOQA Data," Swiss German University and Aerotrack Pte Ltd, Jakarta Pusat, 2015.

[95] T. Brotherton and T. Johnson, "Anomaly Detection for Advanced Military Aircraft Using Neural Networks," IEEE, San Diego, 2001.

[96] Impact Technologies LLC and Air Force Research Laboratory, "Embedded Reasoning Supporting Aerospace IVHM," Impact Technologies LLC, Rochester, 2007.

[97] Z. Jiao, DongSun, Y. Shang, X. Liu and S. Wu, "A high efficiency aircraft anti-skid brake control with runway identification," *Aerospace Science and Technology,* vol. 91, no. 1, pp. 82-95, 2018.

[98] D. J. Lary, "Artificial Intelligence in Aerospace," in *Aerospace Technologies Advancements*, Vukovar, Intech, 2010, p. 492.

[99] National Academies of Sciences, Engineering, and Medicine, "Aeronautics 2050: Proceedings of a Workshop in Brief," The National Academies Press, Washington, DC, 2018.

[100] T. D. Sanger, *Optimal Unsupervised Learning in Feedforward Neural Networks,* M.S. thesis, Cambridge, MA: Massachusetts Institute of Technology, 1989.

[101] E. Torenbeek, Synthesis of subsonic airplane design, Rotterdam: Delft University Press, 1976.

[102] de Havilland Canada, "DHC-5 Pilot's Operating Handbook," 1974.

[103] F. S. Collins, M. Morgan and A. Patrinos, "The Human Genome Project: lessons from large-scale biology," *Science,* vol. 300, no. 5617, pp. 286-290, 2003.

[104] W. E. Faller and S. J. Schreck, "Neural networks: Applications and opportunities in aeronautics," *Progress in Aerospace Sciences,* vol. 32, no. 5, pp. 433-456, 1996.

[105] J. Hale, "Deep Learning Framework Power Scores 2018," Medium, 19 September 2018. [Online]. Available: https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a. [Accessed 19 September 2019].

9. Appendix A – Python Program

```
# ------------------------------------------------------------------------------
# Defining the neural network architecture
# ------------------------------------------------------------------------------

# hidden layer #1 properties
nodes_hlayer1 = 75
act_hlayer1 = 'relu'
# hidden layer #2 properties
nodes_hlayer2 = 100
act_hlayer2 = 'relu'
# hidden layer #3 properties
nodes_hlayer3 = 100
act_hlayer3 = 'relu'
# number of hidden layers (depth of net)
num_hidden_layers = 3

# model properties
model_opt = 'adam'
# model_loss = 'mae'  # https://keras.io/losses/
model_loss = 'msle'  # https://keras.io/losses/
model_metrics = 'mape'  # https://keras.io/metrics/

# train properties
validation_split = 0.3
num_epochs = 40
shuffle = 'True'

# define regression model & builds (compile) the neural net
def regression_model():
    model = Sequential()
    model.add(Dense(nodes_hlayer1, activation=act_hlayer1, input_shape=(n_cols,)))
    model.add(Dense(nodes_hlayer2, activation=act_hlayer2))
    model.add(Dense(nodes_hlayer3, activation=act_hlayer3))
    model.add(Dense(1))
    #keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=
    model.compile(optimizer=model_opt, loss=model_loss, metrics=[model_metrics])
    return model
```

```
# ----------------------------------------------------------------------------------------------------
# Train the neural network and save the weights & biases
# ----------------------------------------------------------------------------------------------------
csv_logger = CSVLogger('log.csv', append=True, separator=';')
# oldStdout = sys.stdout
# sys.stdout = open('logFile', 'w')

# train and test neural network
print("\nNetwork training results: \n")
# reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.001)
model = regression_model()  # build the models
history = model.fit(predictors_norm,
                    target,
                    validation_split=validation_split,
                    epochs=num_epochs,
                    verbose=2)
#                     shuffle=shuffle)
                    # callbacks=[reduce_lr])  # fit the model

# Saving/loading the network architecture (architecture + weights + optimizer state)
model.save(r'\net.hdf5')
```

10. Appendix B – Developed Neural Network Tools

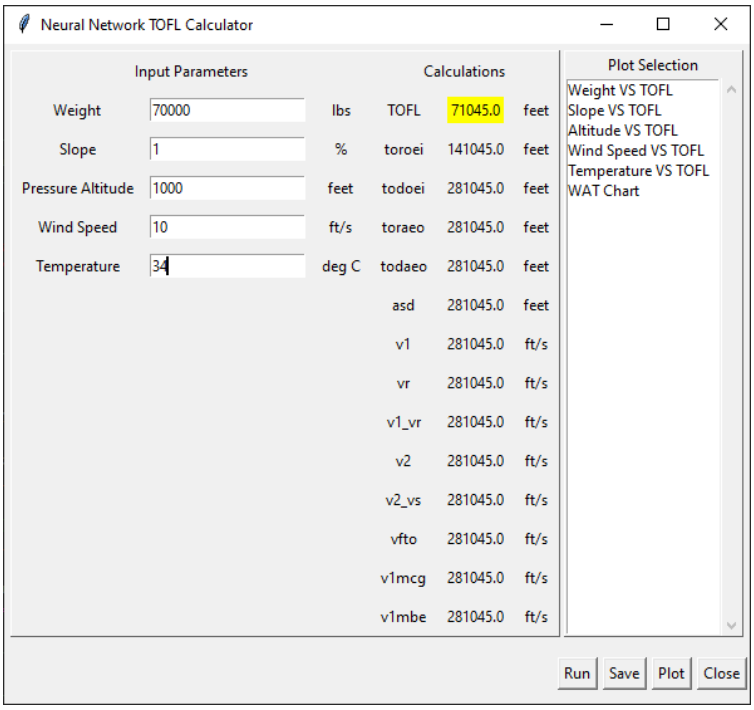


Figure 38 –TOFL prediction tool

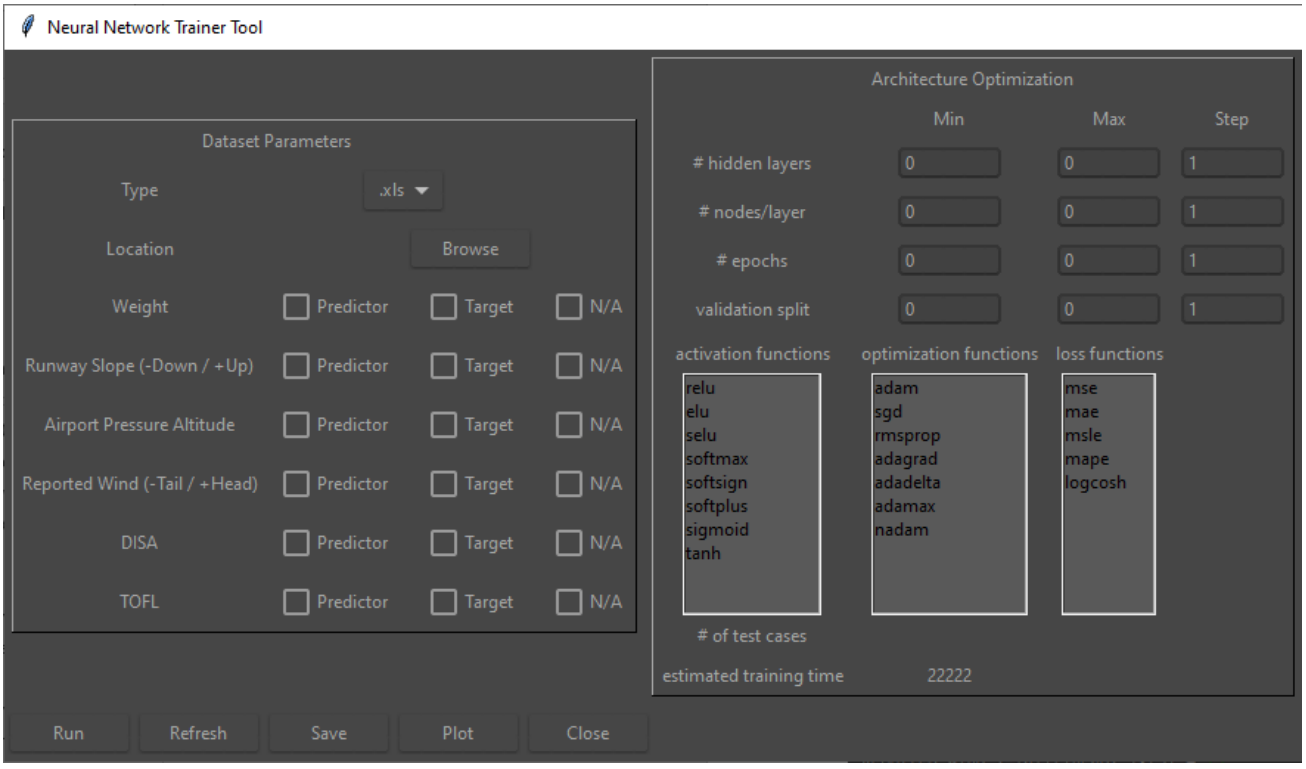


Figure 39 – NN architecture optimization tool