2020-06

CLASSIFYING ADS-B TRAJECTORY SHAPES USING A DENSE FEED-FORWARD NEURAL NETWORK

Gingrass, Colton J.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CLASSIFYING ADS-B TRAJECTORY SHAPES USING A DENSE FEED-FORWARD NEURAL NETWORK

by

Colton J. Gingrass

June 2020

Thesis Advisor: Devaushi I. Singham
Second Reader: Michael P. Atkinson

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CLASSIFYING ADS-B TRAJECTORY SHAPES USING A DENSE FEED-FORWARD NEURAL NETWORK	**5. FUNDING NUMBERS**
6. AUTHOR(S) Colton J. Gingrass	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

There is a recent abundance of flight trajectory data due to Automatic Dependent Surveillance-Broadcast (ADS-B) becoming a prevalent and required aviation traffic control system. Motivated by incidents like the September 11 attacks, the Department of Defense and civilian intelligence agencies have taken a renewed interest in being able to quickly flag and act on flight pattern behavior that is considered outside the norm. Due to the large volume of daily flights in the United States alone, it is almost impossible for human operators to monitor and analyze individual flights for anomalous behavior. The Department of Defense and civilian intelligence agencies stand to gain increased capability and capacity if given the ability to analyze and flag unusual flight trajectories in a matter of seconds. Anomalous behavior in many cases is determined by the overall shape of the flight pattern. This thesis uses calculated shape features to classify nine pre-determined categories of ADS-B flight trajectories using a Deep Sequential Neural Network. With a data set of 11,303 human-classified tracks, the network has performed with an overall accuracy of 71% and a categorical average F1 score of 0.33 on a validation set. It has also performed with 70% accuracy and a categorical average F1 score of 0.25 on a ten-fold cross validation. The proposed method shows promise in being able to select unusual shapes from straight trajectories and in some cases may be able to classify them.

14. SUBJECT TERMS flight, classification, neural networks, machine learning, operations research			**15. NUMBER OF PAGES** 75
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

CLASSIFYING ADS-B TRAJECTORY SHAPES USING A DENSE FEED-FORWARD NEURAL NETWORK

Colton J. Gingrass
Ensign, United States Navy
BS, United States Naval Academy, 2019

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN APPLIED SCIENCE
(OPERATIONS RESEARCH)**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2020**

Approved by: Devaushi I. Singham
Advisor

Michael P. Atkinson
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

There is a recent abundance of flight trajectory data due to Automatic Dependent Surveillance-Broadcast (ADS-B) becoming a prevalent and required aviation traffic control system. Motivated by incidents like the September 11, 2001, attacks, the Department of Defense and civilian intelligence agencies have taken a renewed interest in being able to quickly flag and act on flight pattern behavior that is considered outside the norm. Due to the large volume of daily flights in the United States alone, it is almost impossible for human operators to monitor and analyze individual flights for anomalous behavior. The Department of Defense and civilian intelligence agencies stand to gain increased capability and capacity if given the ability to analyze and flag unusual flight trajectories in a matter of seconds. Anomalous behavior in many cases is determined by the overall shape of the flight pattern. This thesis uses calculated shape features to classify nine pre-determined categories of ADS-B flight trajectories using a Deep Sequential Neural Network. With a data set of 11,303 human-classified tracks, the network has performed with an overall accuracy of 71% and a categorical average F1 score of 0.33 on a validation set. It has also performed with 70% accuracy and a categorical average F1 score of 0.25 on a ten-fold cross validation. The proposed method shows promise in being able to select unusual shapes from straight trajectories and in some cases may be able to classify them.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AIS	Automatic Identification System
ADS-B	Automatic Dependent Surveillance-Broadcast
CNN	Convolutional Neural Network
CV	Cross Validation
FAA	Federal Aviation Administration
MLP	Multi-Layered Perceptron
LSTM	Long Short Term Memory
NOLH	Nearly Orthogonal Latin Hypercube
ReLU	Rectified Linear Unit
ROC	Receiver-Operator Curve
RNN	Recurrent Neural Network

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Military and civilian intelligence organizations have taken a renewed interest in being able to identify and track potential hazards in domestic and international skies for the purpose of preventing events similar to the September 11th attacks on the United States. As of 1 January 2020, the FAA has standardized and enforced the use of Automatic Dependent Surveillance-Broadcast (ADS-B) on all aircraft in controlled airspace. This fairly novel technology relays a state vector from every ADS-B equipped aircraft that includes information such as the aircraft's vertical and horizontal position, speed, and other attributes.

The large amount of data produced and recorded by this system can be used to establish baseline criteria for flights that deviate from normal air traffic control patterns. These anomalous flights could be potential hazards to other aircraft in the airspace or in the case of an extreme event, a hazard to thousands of civilians or military personnel. With thousands of flights occurring every day, it is nearly impossible for the individual human operator to sort, visualize, analyze, and report anomalous flights at a reasonable rate. This has motivated the need for an automated classification ability that aids in identifying and flagging anomalies.

This research aims to identify anomalies by observing the two-dimensional shape of flight patterns and calculate key mathematical attributes believed to define these shapes. After visualizing over 15,000 graphed ADS-B tracks, we develop a classification system containing nine different categories that identify flight tracks by their shape and significant visual attributes. Using these nine categories as selection criteria, two human operators classified a total of 10,920 tracks. Over 7000 other tracks did not fit into these categories and were not included in the data set. Characteristic features of each track, including bearing changes and curvature, are calculated to be used in predictive modeling.

Using these features as input, we design and create a deep sequential neural network that classifies each track into one of the nine identified shape categories. Neural networks are complex mathematical models, loosely modeled after the human brain's neurons, that can be used to predict, classify, and approximate mathematical functions, images, and movement. This research describes this mathematical tool and walks through the basic framework, development, and application to the classification task at hand.

To measure the success of this network, the model classified a fraction of the data set that it had not previously seen. The model was evaluated on its ability to correctly classify each track according to its ground truth label. The network performed with an overall accuracy of 71% and a categorical average F1 score of 0.33 on a validation set. It has also performed with 70% accuracy and a categorical average F1 score of 0.25 on a ten-fold cross validation. The results of this research demonstrate that the created model demonstrates an enhanced ability to distinguish between linear and non-linear flight trajectories. The proposed method also shows promise in being able to classify unusual shapes.

Acknowledgments

I would like to thank my advisors, Dashi Singham and Mike Atkinson, and the Operations Research Department for the guidance, responsiveness, and knowledge they provided to make this document and my degree at NPS a reality.

I would also like to thank my family for their unwavering support in my efforts, dreams, and goals. I would not be even close to where I am today without them.

Finally, I'd like to thank my friends who have supported me like brothers and sisters along the way. My life would not be as wild, fun, and free without them.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

With the goal of maximizing time and manpower saved by the Department of Defense and civilian intelligence agencies, this thesis explores the field of machine learning as it has been applied to trajectory prediction, shape analysis, and automated classification. Using information and lessons learned from existing literature, we build a multi-label classifier neural network and apply it to a database of Automatic Dependent Surveillance-Broadcast (ADS-B) flight tracks classified by human operators.

Section 1.1 addresses the background behind studying and classifying ADS-B data. Section 1.2 builds the motivation for continuing research in the area. Section 1.3 discusses our contribution to the field of classifying anomalous flight patterns using machine learning programs. Finally, Section 1.4 outlines the framework for the rest of this thesis.

1.1 Background

Since the September 11th attacks on the United States, military and civilian intelligence organizations have taken a renewed interest in being able to identify and track potential hazards in domestic and international skies. Since then, airspace has become an even more crowded and dynamic environment with the advent of drone technology and a continually evolving commercial airline industry. Automatic dependent surveillance-broadcast (ADS-B) is a fairly novel technology that allows aircraft to broadcast their position, speed, and altitude–among other things–in real time with precision (FAA 2020). As of 1 January 2020, the Federal Aviation Administration (FAA) has standardized and enforced the use of ADS-B on all aircraft in controlled airspace. This has introduced massive troves of data that can be used to establish baseline criteria for flights that deviate from normal air traffic control patterns. These anomalous flights could be potential hazards to other aircraft in the airspace or in the case of an extreme event, a hazard to thousands of civilians or military personnel (Li 2013). With thousands of flights occurring everyday in the United States, the task of analyzing and tracking anomalous flights "presents an insurmountable challenge for an individual human operator" and the signal analysis field "stands to gain a great deal from computational automation" (Sillito and Fisher 2008). Thus, the ability to classify these flight patterns as anomalous and bring them to the attention of an aviation supervisor or intelligence analyst would increase readiness and the speed of crisis analysis and response.

1.2 Motivation

With the massive amount of ADS-B data made available in the last decade, a large research interest has taken hold in mathematical and statistical fields. Authors are interested in the defining features of movement for shape analysis (Prati et al. 2008), the ability to detect possible cyber attacks on ADS-B systems (Akerman et al. 2019), and being able to precisely and accurately predict the 4-D kinematic movements and trajectories of aircraft (Liu and Hansen 2018). To tackle these problems, academia has taken advantage of the current available computing power to employ machine learning techniques such as neural networks, random forests, clustering, and support vector machines, as well as more traditional statistical techniques such as principle component analysis.

While this data is readily available, it is not necessarily "clean" in a statistical sense. Many flights broadcast their information in sporadic time intervals due to broadcast strength, weather patterns, and other external factors. Also, ADS-B ground stations may not receive broadcasts in full or at all due to other external factors. These issues make ADS-B data very noisy and occasionally sparse. In order to begin modeling or making justified inferences, the data must be pre-processed, smoothed, and cleaned. We have developed algorithms to process each raw ADS-B track into an interpolated vector of 20 coordinate pairs. From these coordinates, we are able to calculate features such as curvature radius, change in bearing, and distance traveled.

The Department of Defense and the civilian intelligence sector have also gained interest in ADS-B prediction and classification. These organizations would increase their analytical and actionable intelligence network capacity by being able to sort through millions of flight tracks without a human operator. Given a classification algorithm with reliable accuracy and precision, thousands of flight tracks could be classified in minutes. The categories deemed anomalous could be forwarded to an analyst for further investigation. This would significantly reduce the burden on the individual intelligence operator.

The task of classification does not prove to be a simple problem for machine learning. Broadcast dropout and signal continuity have been a legitimate concern with ADS-B (Semke et al. 2017). Due to this possible variability in the initial data before pre-processing, the noise generated from smoothing and interpolating coordinates into a standardized vector, and the noise generated by human error in classifying flight tracks, high variability may exist in the data set used in this thesis. Achieving significant accuracy with a neural network would provide a good baseline for further exploration of classification of anomalous flight behavior in the machine learning field.

1.3 Our Contribution

With algorithms we have developed, we convert raw ADS-B tracks into multiple files containing vectors of each track's calculated features. Among these features are latitude, longitude, curvature, change in bearing, distance traveled, and a binary feature indicating if the track finishes in close proximity to its starting position. We then develop a classification system based on observed shapes and patterns in the graphed coordinate output of thousands of flight tracks. A database of 10,920 tracks was then divided into a 60/20/20% train/test/validation split. Using inspiration from Prati et al. and Calderara et al., we isolated the bearing changes and curvature statistics in each track. Using solely these features for each track's input vector, we developed a deep multi-layer perceptron (or Deep Sequential Neural Network) with the objective of classifying each shape in its respective ground truth category.

The network performed with an overall accuracy of 71%, classifying straight tracks, single loops, and round trip flights particularly well. Straight tracks, single loops, and round trip flights had 0.86, 0.38, and 0.46 as their respective F1 scores. Parabolic trajectories, switchbacks, and figure eights performed poorly compared to other categories with F1 scores of 0.00, 0.17, and 0.14 respectively. In order to explore the robustness of the particular network, we applied it to the same data set, but only differentiating between linear and non-linear track shapes. This performed with an overall accuracy of 77%. The model evaluated straight tracks with a F1 score of 0.85 and non-linear tracks with a F1-score of 0.55. Further analysis demonstrates that this particular network may be over fitting on the data and is not properly tuned for binary classification.

1.4 Thesis Outline

Chapter 2 reviews relevant literature and will briefly introduce the concept of a neural network, shape analysis, and past employments of machine learning in air traffic analysis. Chapter 3 will explore sequential neural networks in depth and discuss the methods, tools, and mathematics used to build and tune the final neural network used. Chapter 4 contains analysis of the algorithm's performance on the validation set, a 10-fold cross-validation, and a binary data set. The analysis also contains an in-depth look at how the network performed on each track type. Chapter 5 will conclude this thesis with a suggested way forward for further developing the model, recommendations for alternative neural network schemes, and possible improvements in data classification.

CHAPTER 2: Literature Review

Section 2.1 explores the background of ADS-B and its use in air traffic control. Section 2.2 discusses the different techniques used in predicting and classifying the motion of trajectories. Then we will review different features and techniques used to describe motion, shape, and trajectory in Section 2.3.

2.1 Automatic Dependent Surveillance-Broadcast Background

Traditional air traffic control relies on ground radar systems that send intermittent pings to develop situational awareness in the sky. In the last decade, air traffic control technology has developed further and traffic density in controlled airspace has also grown. Both manned and unmanned aircraft have begun to occupy the same airspace. With increased traffic density comes new challenges in safety, situational awareness, and collision avoidance. Automatic Dependent Surveillance-Broadband (ADS-B) has developed in the recent decade as an alternate air traffic control surveillance system (Zhang and Qiao 2008). The on board system periodically broadcasts a state vector with the aircraft's horizontal and vertical position (Semke et al. 2017). An ADS-B module also receives the same state vector information from other aircraft and any messages relayed from a ground station. With information from these external sources, ADS-B develops the pilot's awareness to proximate aircraft (Semke et al. 2017). The system has not only been integrated into the FAA's air traffic control system but also as of 2020, general aircraft are required to broadcast the signal, making aviation safer, more efficient, and more advanced (FAA 2020). This data produced from individual aircraft is collected in massive amounts. This large data set allows for in-depth, multifaceted analyses similar to the ones discussed below.

2.2 Flight Trajectory Prediction and Anomaly Detection

2.2.1 Neural Networks

Put simply, neural networks are a machine learning tool with deep mathematical underpinning that are modeled loosely on the human brain's neurons. One of the many reasons neural networks have become fashionable in the statistical realm is their ability to overcome some of the limitations of the linear model. Linear models are fit efficiently in closed form or with convex optimization. Neural networks thrive in being able to apply a nonlinear transformation to inputs via a mapping function. This allows neural networks to benefit from being able to model complex non-linear functions. With this function applied throughout multiple layers, the input can be represented in a different dimension with different features. This change in dimensionality can be seen in the algorithm developed in this thesis. We take a vector of 36 inputs and transform it into a vector with nine outputs using a series of nonlinear transformations (Goodfellow et al. 2016).

Typically neural networks are given a certain task such as prediction or classification. The model's input data is initially divided into train, test, and validation sets. Typically training sets make up the largest proportion of the total data. This is the data that the model will repeatedly cycle through and adjust its internal parameters. Once adjusted, the model will be evaluated on the test set. Depending on its performance, the model will be adjusted further until considered ready for validation. The validation set remains completely unseen by the network until its ready for the validation phase. The model's true performance and ability to predict on unseen data is demonstrated when predicting on the validation set. This determines the model's true worth in the context of the problem.

In 1999, the idea of using neural networks to predict aircraft trajectory was explored by Yann Le Fablec. With the limited computational power available at the time, Fablec used a single hidden layer feed-forward neural network. He concluded that neural networks outperformed existing parametric methods (Yann Le Fablec 1999). With the development of Tensor Flow and Keras, the application of neural networks has leaped to an unprecedented level. Academics have applied feed-forward, recurrent, long short term memory (LSTM) (Shi et al. 2018) and convolutional neural networks (CNN) (Akerman et al. 2019) on aircraft trajectories and ADS-B data for prediction and classification.

Recurrent Neural Network (RNN)

Recurrent neural networks are a special subset of neural networks which add a temporal dimension to the model. The aim is to exploit the sequential nature of a certain input vector in order to make a better prediction.

Recurrent neural networks have fallen into favor in prediction spaces that are inherently tied to sequences. Liu et al. (2016) argues that traditional RNN models have issues modeling continuous time intervals and distance. They designed a “Spacial Temporal” RNN which uses time specific and distance specific transition matrices to improve prediction. Liu et al.’s method has proven effective in prediction but it has not been tested in the classification realm. This effective use of neural networks provides a promising foundation for the use of a RNN or simple Multi-Layered Perceptron (MLP) with trajectory data.

RNNs have been used for classification using kinematic and trajectory data recently. Baekkegaard et al. (2018) uses Automatic Identification System (AIS) data from vessels in Danish waters to create a RNN that classifies five types of ships. This provided more accurate results than a random forest classifier using the same data set.

Long Short Term Memory (LSTM) Networks

Shi et al. (2018) explore the concept of applying a LSTM to flight trajectory prediction. A LSTM network is a subset of the RNN that has a complex structure which keeps track of characteristics of a sequence over arbitrary time intervals. It employs gates which filter past input vector information throughout the prediction process. Their technique also employs sliding windows which helps improve prediction precision for adjacent points in the input vector.

Akerman et al. (2019) couples the Convolutional Neural Network (CNN) with a specific LSTM encoder in order to detect adversarial ADS-B broadcasts. This appears to be another one of few recent forays into using neural networks to classify ADS-B data. CNNs have a distinct advantage of not needing engineered or predetermined features when processing data (Goodfellow et al. 2016). While this proves to be an advantage from a pre-processing perspective, it becomes a detriment due to the computational effort and time needed for a CNN to determine these features. While this computational effort can be overcome by parallelized operations using modern computer graphics processing units, it takes specific hardware not currently at our disposal for this thesis (Goodfellow et al. 2016).

While the convolutional and LSTM networks demonstrate a lot of pre-processing advantages, they need a lot of computational power and time. In a situation with crowded airspace and not a lot of time, these networks may be outperformed by a more simple approach with engineered features and a less complex network design.

2.3 Trajectory Feature Analysis

Shapes are composed of a series of angles and curves that differentiate them from other objects. For example, a square is composed of four 90° angles with four equilateral sides. This differs from a right triangle which is composed of three angles summing to 180° and three sides with lengths determined by the Pythagorean theorem. If given one of these facts, one could attempt to predict a certain shape. This is the fundamental idea behind shape analysis. Given a series of features such as angles, one may have enough information to make an informed prediction of a shape.

In the case of ADS-B, we study the shapes formed by the trajectory of a moving aircraft in two dimensions. Tao et al. (2004) identifies that linear functions are inadequate when used to predict the movement of an object along a curvature over time. And even though "piece-wise line segments can approximate curves, they cannot be effectively applied for prediction, especially in the distant future"(Tao et al. 2004). This thought process laid groundwork for mathematicians and information scientists to extract specific features from non-linear paths. For example, Nadeem Anjum (2008) uses "multiple feature spaces to obtain higher degrees of descriptiveness" in order to predict human trajectories in video with clustering techniques. These include velocity, directional distance, horizontal and vertical trajectories, and accelerations. Junejo et al. (2004) explores these same features while adding path curvature to increase modeling accuracy of human movements along a path.

Inspiration for track characteristics used in the following neural networks was drawn from Prati et al.'s use of bearing change in trajectory analysis. Prati et al. propose the ability to model the shape of a trajectory based on a sequence of angles. They use a mixture of von Mises distributions for trajectory analysis because it is particularly useful for the statistical inference of angular data. The von Mises distribution is also referred to as the circular distribution (Prati et al. 2008). The authors use iterative clustering in order to classify certain shapes. Calderara et al. (2011) further explored this method and applied it specifically to human foot traffic trajectories in video analysis. The successful use of circular statistics and angular data reinforces our intuition to use bearing changes and curvature statistics in the following networks.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Deep Sequential Neural Network Development

This chapter begins by exploring the data and features used in our model in Section 3.1. We then take an extended look into the basics of deep sequential neural networks and the hyperparameters used to tune their predictive capacity in Section 3.2. Section 3.3 finishes by describing the search for model framework, specifications, and hyperparameter values.

3.1 Input Data Overview

3.1.1 Classification System

The data used in this thesis is a collection of raw ADS-B tracks that have been converted and interpolated into multiple vectors of length 20 containing latitude, longitude, change in bearing, curvature, distance traveled, and a binary feature indicating if the track finishes in close proximity to its starting position. We will explore the features used with great detail in Section 3.1.2. In order to establish ground truth labels, we combed through thousands of plotted tracks and isolated different common shapes and patterns. We settled on a classification system with nine categories described in Table 3.1. Visually, these shapes are similar to the examples pulled from the data in Figures 3.1, 3.2, and 3.3.

Table 3.1. Classification System

Class Number	Shape	Class Number	Shape
1	Straight	6	Out and Back
2	Straight with Slight Deviations	7	Switchback Pattern
3	Single Loop/Circle	8	Figure 8
4	Parabola	9	Sinusoidal Shape
5	Multiple Loops		

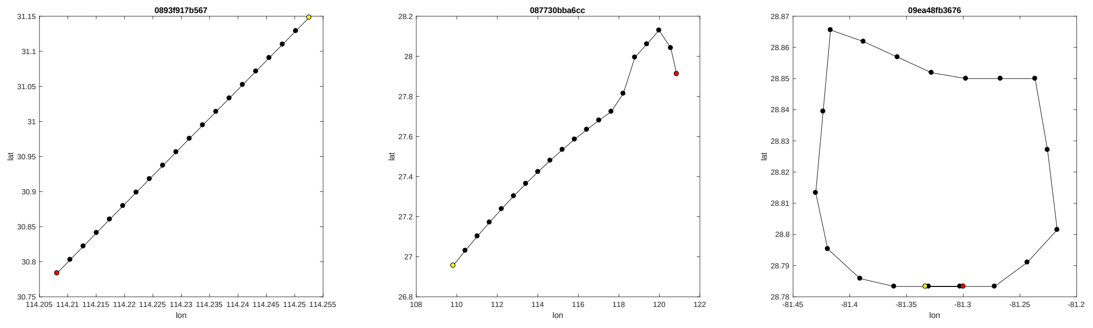


Figure 3.1. Types 1 (straight), 2 (detour), and 3 (loop)

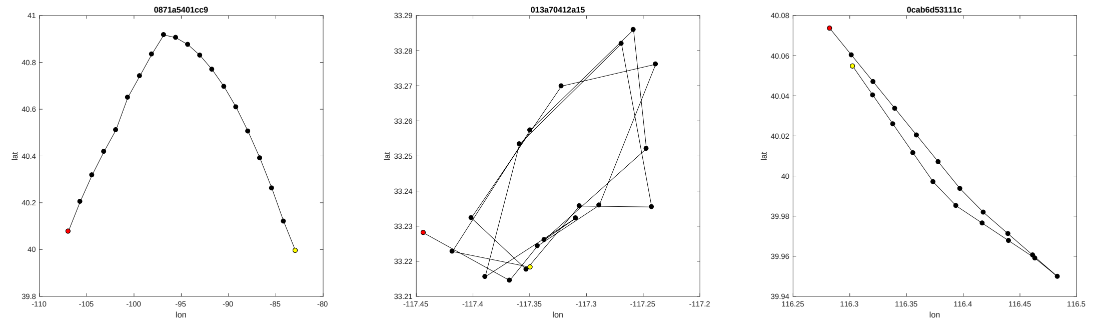


Figure 3.2. Types 4 (parabola), 5 (multi-loop), and 6 (out and back)

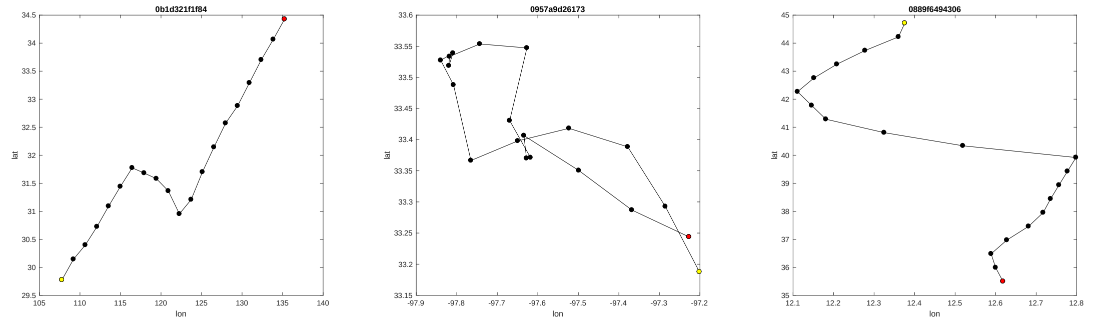
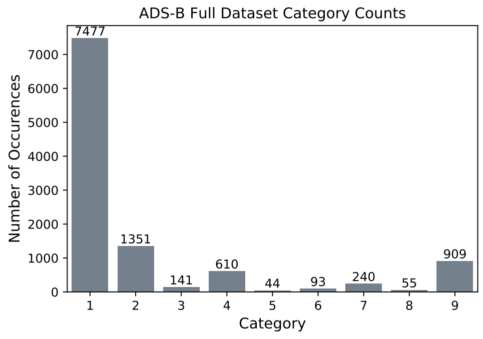


Figure 3.3. Types 7 (switchback), 8 (figure eight), and 9 (sinusoidal)

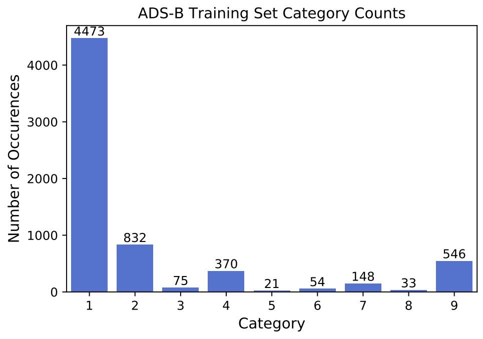
Tracks classified as type 1 are linear with little deviation. Tracks with no sharp turns and slight curvature or bow are still classified as type 1. Type 2 tracks have a largely linear trajectory, but they contain small deviations and turns at the beginning or end of the track. Type 3 tracks appear circular in nature. Any track that continuously changes bearing and ends up at or close to its starting point, making a circular shape, is classified as type 3. Type 4 is a parabolic shape with gradual turns that effectively make the track non-linear. Any shape parabolic in nature is classified as type 4. Tracks that continually change bearing around a certain center point are classified as type 5. We refer to this as the ‘multi-loop’ track. Trajectories that end up in close proximity to their beginning point that do not appear circular in nature are classified as type 6. These are also referred to as ‘out and back’ tracks. Type 7 or the ‘lightning’ track models a trajectory that takes multiple sharp turns in the middle of the track. Type 8 is appropriately labeled as any trajectory appearing in the shape of a figure eight. Trajectories that move back and forth in a zig-zag pattern and in the shape of a sinusoid are classified as type 9. This type was originally divided into three separate categories, but due very few samples of different variations, the trajectories were combined under one label.

After establishing these ground truth labels, two sets of approximately 600 tracks were classified by two people using the ground truth labels. The two operators’ results were then compared to each other to establish a human error metric. This would create an upper bound our model’s accuracy. The average difference between two operator’s classifications was approximately 21.5%.

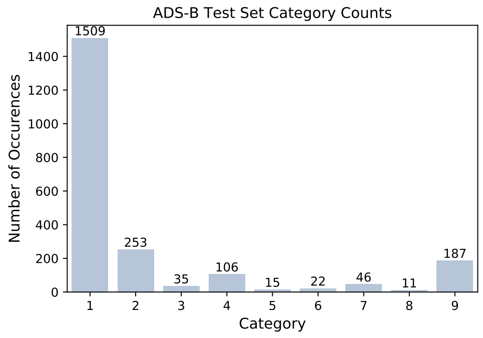
Over 18,000 tracks were graphed and classified, but many did not fit the predetermined categories described in Table 3.1. These were assigned a default label and were not included in the final data set. The final data set includes 10,920 tracks that have been labeled by two operators, each labeling approximately half of the set individually and independently. The data was split evenly into a 60/20/20% split for training, testing, and validation respectively. Figure 3.4 shows the breakdown of the data set by class.



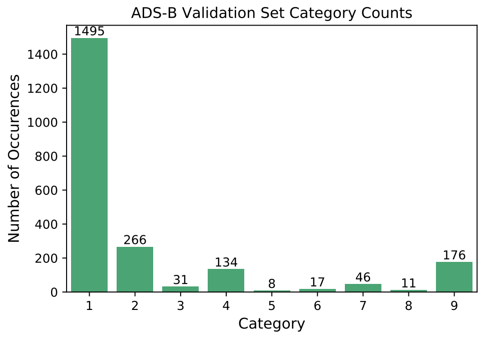
(a) Total Set



(b) Training Set



(c) Test Set



(d) Validation Set

Figure 3.4. Train/Test/Validation Split

Even with a volume of over 10,000 tracks, the data set has a large imbalance among the categories. Type 1 dominates, while types 5 and 8 have very little representation.

3.1.2 Features

As stated above, this data set includes features including latitude, longitude, change in bearing, curvature radius, round trip, and distance. Using conclusions from Prati et al. (2008) and Calderara et al. (2011)'s work on shape analysis, we have decided to use both curvature radius and change in bearing as the input features for each track. The network will not have access to distance, latitude, longitude, round trip, or distance. We believe these features to have the most explanatory power in classifying the categories displayed above, because they do not concentrate on the location or size of the track. The features are normalized to focus on the shape of the track.

Change in Bearing

Change in bearing calculates the two dimensional change in direction between a pair of X and Y coordinates at times t and $t+1$, which are (X_t, Y_t), and $(X_{t+1}, Y_{t+1}$. Formula (3.1) was used to calculate the series of bearing changes for each input vector. The subscripts of X and Y indicate their time sequence in the track.

$$\mathbf{Bearing1} = \mathbf{arctan2}(X_{t+1} - X_t, Y_{t+1} - Y_t)$$

$$\mathbf{Bearing2} = \mathbf{arctan2}(X_{t+2} - X_{t+1}, Y_{t+2} - Y_{t+1})$$

$$\mathbf{\Delta Bearing} = \mathbf{Bearing2} - \mathbf{Bearing1} \tag{3.1}$$

Values for this change can be both positive and negative. The magnitude of the change conveys the pertinent information of the feature. An entry with a high absolute value indicates that the aircraft made a significant change in direction. A series of low magnitude entries indicates a relatively straight path. However, a series of significant magnitude entries throughout the vector may indicate the track is moving in a circular pattern. When examined in a macro context, the series and magnitude of this feature indicate a lot about a shape, making it a potentially useful classification feature.

Curvature

Curvature can be calculated in multiple ways in both closed and open form. When given a series of points, curvature can be calculated using a three point approximation method. In the algorithm used to calculate features for our data set, the curvature radius was calculated using the three point approximation method. This method is simple and does not take much information to calculate, but it can provide a lot of inferential power about a specific track. The radius is calculated with Formula (3.2). Given three successive points, one can form a triangle by connecting the dots. Variables a,b, and c denote the lengths of the three sides of a triangle made by these points. The area of the same triangle is represented by S.

$$\mathbf{R} = \frac{\mathbf{abc}}{\mathbf{4S}} \tag{3.2}$$

This radius size can tell us a lot about the track. Figure 3.5 demonstrates this information effectively. With a small radius, the curvature of the three points is high. This indicates a sharper turn than a curvature statistic with a large radius. The multiplicity and sharpness of these curvatures in a single input vector will hopefully provide viable information for the neural network to train. For example, a vector with multiple small radius turns may indicate a shape with a jagged shape such as type 7 or type 9. A vector with a series of solely large radii would most certainly indicate a straight path.

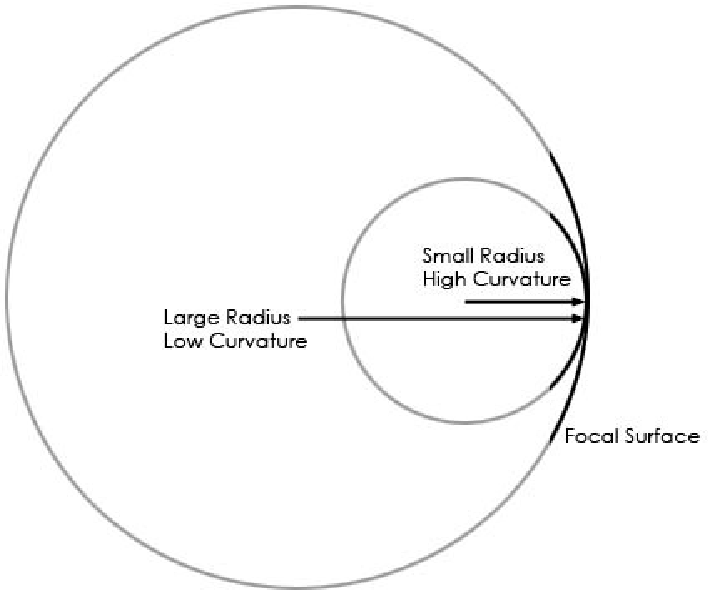


Figure 3.5. Curvature Representation Source: Starizona (2020).

3.2 Sequential Neural Network Overview

The increasingly widespread use of neural networks has shown promising results in prediction. However, the classification of kinematic data is fairly novel. We have decided to apply similar methods to those discussed in Chapter 2 in the form of feed-forward or sequential neural networks with the Keras API (Chollet et al. 2015). These are also referred to as Multi-Layered Perceptrons (MLP).

In order to establish a common language moving forward, the following terms are defined from Goodfellow et al. (2016):

Input Layer:

The initial vector of data given to the network.

Hidden Layer:

A layer not visible to external systems that contains, weight, bias, and applies the activation function, feeding forward to the next layer.

Output Layer:

The final resulting vector of the algorithm.

Width:

The dimensionality of a layer; the total number of neurons in a layer.

Units/Neurons:

A representation of a vector to scalar function that receives input from a previous layer and computes its own activation value to output.

Depth:

Overall number of layers or length of the network chain.

Activation Function:

Function that computes hidden unit values with inputs from the past layer.

Loss Function:

Function that calculates model error.

Knowledge of these terms will be essential in order to understand the structure and nuances of training, testing, and tuning a neural network model.

These feed forward networks represent a composition of multiple functions in a chain-like form made up of an input layer, hidden layers, and an output layer. During network training, the algorithm applies a set of functional mappings that ultimately dictate the neural network outputs (Goodfellow et al. 2016). The sequential networks described below are comprised of a series of dense, fully connected layers that resemble the graphical scheme obtained from Ognjanovski (2019) in Figure 3.6.

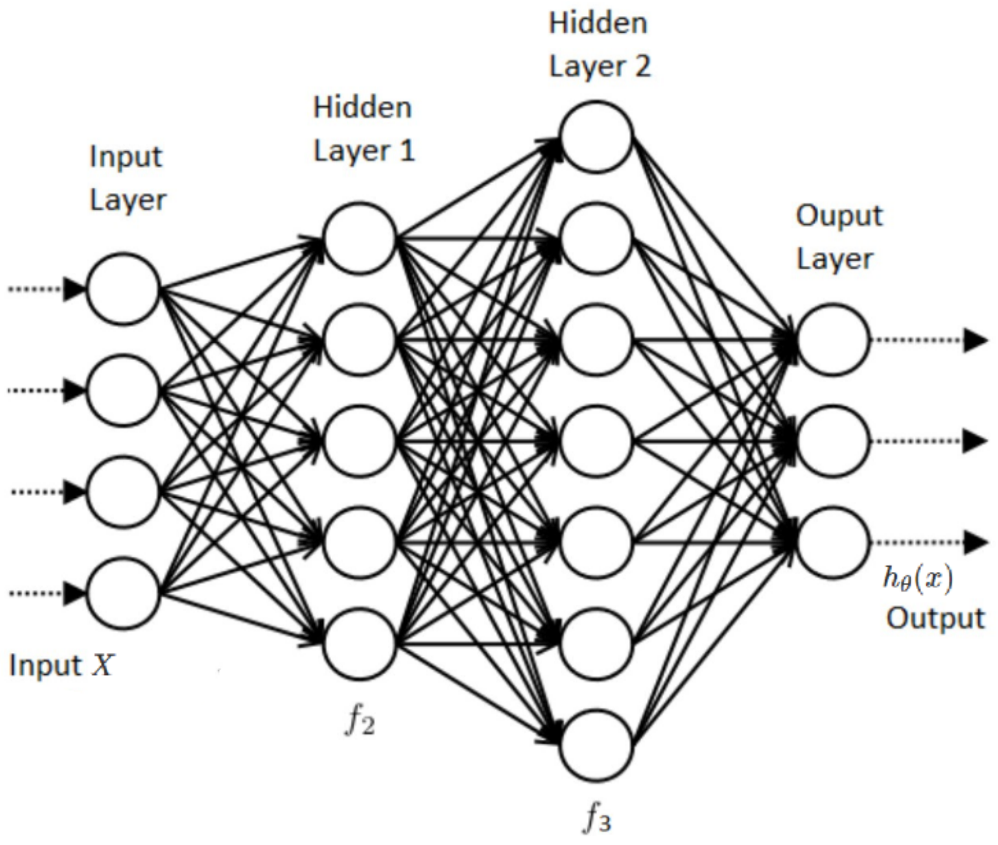


Figure 3.6. Dense Sequential Neural Network Diagram
Source: Ognjanovski (2019).

In the case of classifying either a binary outcome or a set of categorical outcomes, the output layer's function and width change. For binary classification, the output layer is width two; for multi-category classification, the output layer width corresponds to the classification labels. The output layer activation functions for the models used will produce probabilities of potential outcomes (ie., classes) (Chollet et al. 2015). Further details of each model will be discussed below.

3.2.1 Hyperparameter Tuning

Hyperparameters are algorithmic settings set before the training process. Goodfellow et al. states that "some of these hyperparameters affect the time and memory cost of running the algorithm. Some of these hyperparameters affect the quality of the model recovered by the training process and its ability to infer correct results when deployed on new inputs" (Goodfellow et al. 2016). Tuning hyperparameters follows a similar paradigm to the statistical relationship between train/test error. In machine learning, this error is called generalization error. Generalization error is the measure of how accurately a model will perform on an unseen or "general" data set. When the optimal or near optimal hyperparameter is found, the generalization error will be low and the predictive capacity of the model will be optimal or near optimal. If the hyperparameter is set non-optimally, there is a chance for reduced capacity or high generalization error. Figure 3.7 is a useful representation of the concept.

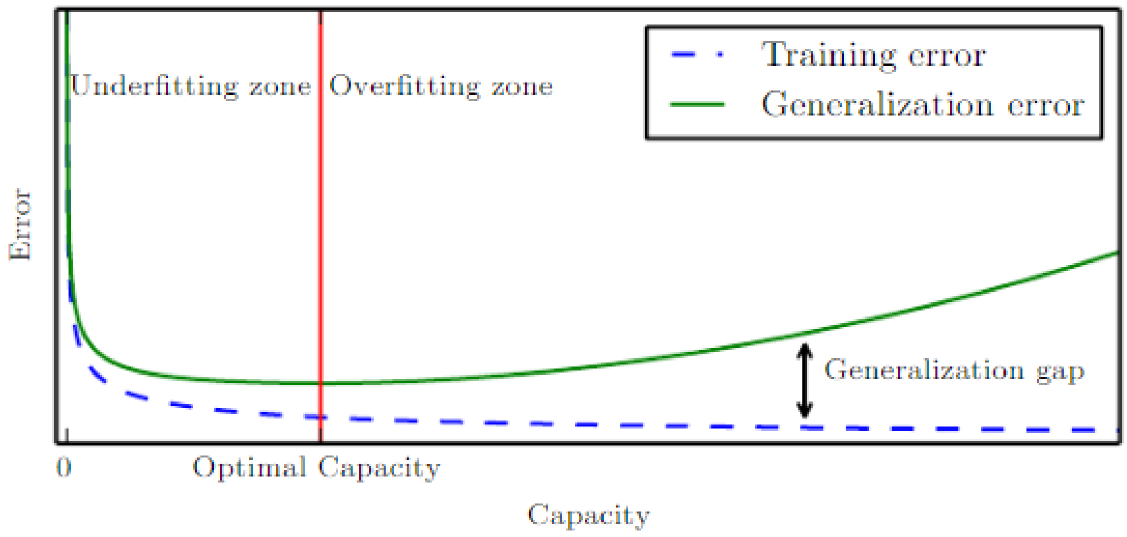


Figure 3.7. Training/Generalization Error Paradigm
Source: Goodfellow et al. (2016).

In order to gain the most predictive power in the following models, we conducted a Nearly Orthogonal Latin Hypercube (NOLH) experimental design to explore the number of hidden layers necessary and then a series of grid searches for particular hyperparameters that affected the speed and capacity of the network. The follow-on subsections explore significant hyperparameters and model settings that were tuned for the multi-level classifier.

Hidden Layers

Hidden layers are placed between the input and output layers. Each applies an activation function to transform the input of each layer into an output. The activation function of each hidden layer in this network will be a Rectified Linear Unit (ReLU). This is the typical activation function applied to feed forward neural networks that preserves many linear properties, making the output easier to optimize and the model faster to train. The network then determines the weights and biases applied to each output in a gradient descent optimization process. The number of hidden layers in a network determines the depth of the network. The number of neurons in those layers determines the width of each layer. The depth and width of each layer have a significant impact on the output and predictive capacity of a neural network (Goodfellow et al. 2016).

Loss Functions and Error

To understand the train, test, and validation error calculated by this algorithm, we must understand the loss function that governs that error. Many different loss functions can be applied to a neural network, but specific loss functions are superior to others depending on the task of the network. In this multi-category classifier, we used categorical cross-entropy loss which pairs very well with the softmax activation function due to their shared use of probabilistic output. This loss is calculated using the following function (Peltarion 2020):

$$L(y, \hat{p}) = -\sum_{i=1}^{9} \left(y * \log\left(\hat{p}_i\right)\right). \tag{3.3}$$

In (3.3), $\hat{p}_i$ represents the predicted probability of a certain track being in category i. The variable, y, represents the true label of that individual prediction. For example, if a certain track is predicted to be type 5 with a probability of 0.95, and the true label is 5, the loss will be extremely low. If the true label is not five, the loss would be relatively large.

Learning Rate

The learning rate affects the step size of the gradient descent algorithm used in the training of the neural network. While outside the scope of this thesis, gradient descent plays an enormous role in the effective capacity of machine learning algorithms (Goodfellow et al.

2016). If the learning rate is too high, training error can increase instead of decrease. When too low, the training time becomes slower and the training error may become stuck at a sub-optimal value. For this reason, learning rate is typically considered the most important hyperparameter to tune.

Dropout Rate

Dropout is a technique used to combat overtraining of a neural network. At a pre-selected rate, the model will randomly disable a neuron in each layer in the network along with its incoming and outgoing edges at each update during training (Srivastava 2013). The hyperparameter in this case is the rate at which the random dropout occurs. This happens independently and randomly for each hidden neuron and training iteration. Having too high of a dropout rate has the potential to negatively impact predictive capacity and having too low of a dropout may result in no effect on the training of the model. Overall, research has shown that dropout significantly helps prevent overfitting (Chollet et al. 2015).

L2 Regularization

L2 Regularization is another method used to prevent overfitting. While dropout is random, this method enforces a targeted penalty. This penalty value, λ, is the hyperparameter that is tuned. If a certain node's output is weighted too high, the layer output can depend on this node too much. This could not only affect the overall training of the model, but also the weight given to certain entries in the input layer. Assigning an appropriate penalty value will reduce these overbearing weights. Due to the noise level of the classified ADS-B data, some tracks have inconsistencies at the beginning and end of the track vector. This regularization helps prevent those inconsistencies from manifesting in the model (van Laarhoven 2017).

Batch Size and Epochs

Batch size and epochs are two related parameters that involve model training iterations. Batch size is the number of samples that the model trains on before updating its internal parameters (Brownlee 2018). At the end of each batch, the model calculates loss and updates the optimization algorithm. Each update is adjusting the network in order to find the optimal weights of the neural network. The batch size allows the estimated optimal weight to be updated based on a gradient calculation from a batch vice the whole set of data. This decreases training time. The epoch parameter defines the number of times the network will iterate over the entire data set (Brownlee 2018). For example, a model with 768 input vectors, a batch size of 64, and 100 epochs would train 100 times on the data set. During each training iteration, the model would update its internal parameters 12 times, because the 768 input vectors would be divided into 12 batches of 64 input vectors.

3.3 Multi-Label Classifier

The multi-label classifier was built in Python using the Keras machine learning platform. The network takes a vector of bearing changes and curvature radii as input and after a series of dense feed-forward layers, produces a vector of nine probabilities. The nine values correspond to that particular track's probability of being in a certain category. This is done using the softmax activation function paired with categorical cross entropy. (Zhang and Sabuncu 2018).

With this task of classifying multiple categories, the depth of network needed was initially unknown. In attempt to maximize the model's capacity and minimize generalization error as seen in Figure 3.7, we began a search for the proper number of layers in the network, also known as depth. First, we used a NOLH experimental design to refine the search area for the number of layers to be included the model. Once we refined the search and chose the number of layers appropriate for this model, we employed 3-fold cross-validated line and grid searches for optimal hyperparameter values to include layer width, learning rate, dropout rate, L2 regularization, batch size, and number of epochs.

To refine the search parameters for the depth of this network, we used a NOLH experimental design (Sanchez 2011) with the ranges of layers and hyperparameters in table 3.2.

Table 3.2. Exploratory NOLH Response Surface Ranges

Factor	Minimum	Maximum
Layers	3	20
Neurons	10	100
Epochs	100	1000
Batch Size	64	512
Dropout Rate	0	0.2

Using a NOLH as a way to explore the response surface saves a large amount of time. If were to exhaustively search the above factors in increments of five, it would take 4,644,864 replications. Even at one second per replication, that would not compute results in a timely fashion. Instead, this NOLH resulted in 17 experiments with different settings for each of the factors above. The design results in a response surface that is adequately explored. Epochs, batch size, and dropout rate are important factors to consider in the model training. These were selected to be included in the NOLH because they have a high effect on the networks training speed and fit. If we were to only explore layers and neurons at one setting of batch size and epochs, we would find the layer and neuron quantities optimal for that setting alone. Adding these parameters allows us to more fully explore the respone surface and predictive capacity of each combination of layers and neurons. The scatterplot in Figure 3.8 represents a multidimensional look at the response surface area covered.

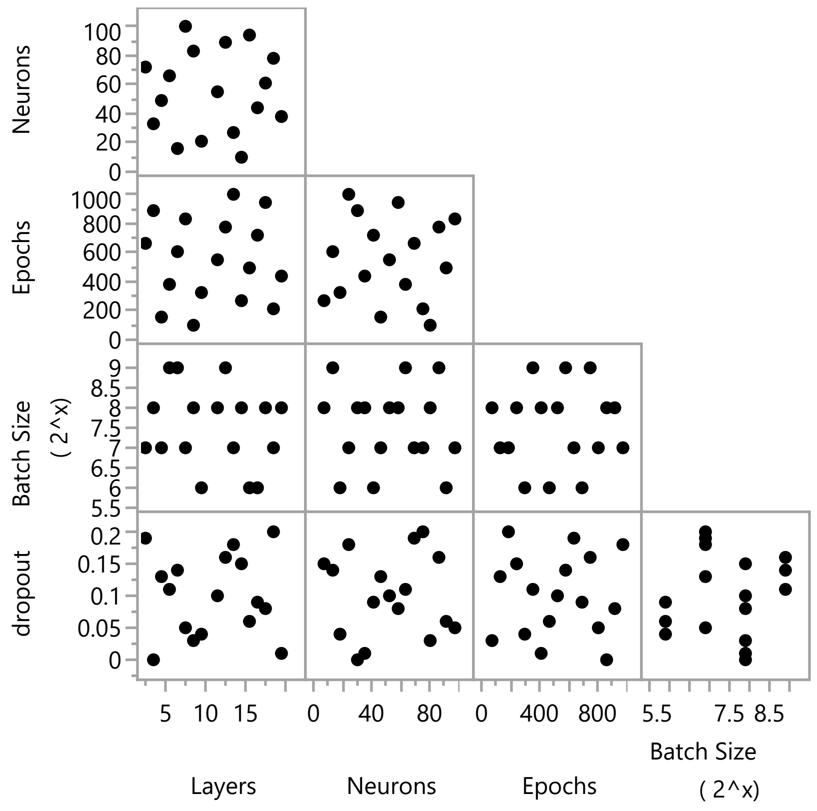


Figure 3.8. Exploratory NOLH Response Surface Scatterplot Matrix

From the 17 experiments conducted, we selected the refined settings to conduct another experiment based on the train accuracy, test accuracy, categorical F1, macro-F1, and weighted F1 scores. These metrics are defined in Chapter 4. The highest performing models appeared to exist with a depth between three and 19 and a width of 20 to 200 neurons. With these thresholds in mind, we conducted another NOLH design with the ranges described in Table 3.3.

Table 3.3. Refined NOLH Response Surface Ranges

Factor	Minimum	Maximum
Layers	3	10
Neurons	20	200
Dropout Rate	0	0.2

The width of the layers was increased in order to explore the possibility of picking up more attributes of the data set in the shallower model. The visualization in Figure 3.9 represents the refined NOLH Response surface.

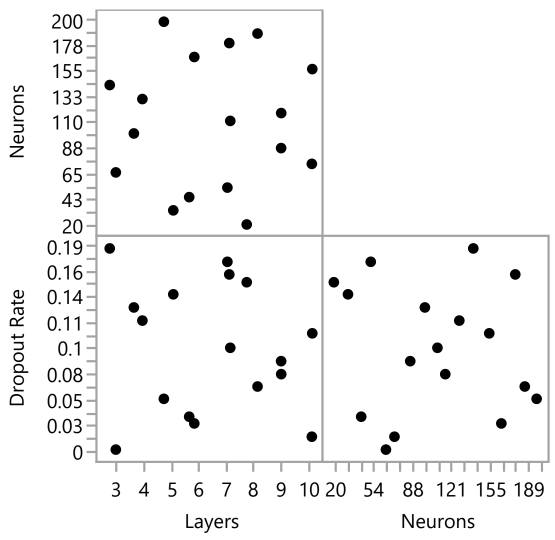


Figure 3.9. Refined NOLH Response Surface Scatterplot Matrix

We see that the response surface of the network is covered fairly well, but there may be some gaps between four and six layers combined with 65 to 155 neurons. Using the highest train and test accuracy, categorical, macro, and weighted F1 scores, we conclude that the best candidates for the final model are four layers with 133 neurons and a dropout of 0.11 and six layers with 43 neurons and a dropout of 0.04. The model with six layers maintained a slightly better train and test error as well, but at the cost of a longer training time. The improved train test error was considered very marginal. For that reason, the simpler, four layer model was chosen for further evaluation.

3.3.1 Model Tuning with *K*-Fold Cross Validated Grid Search

Before, we discussed that NOLHs are computationally efficient, but come at the expense of potential search gaps and perfect orthogonality in design. While these are useful with a large number of different parameters, they do not always examine the complete response surface. When there are three or fewer parameters to explore, exhaustive grid search with a factorial design can become a computationally feasible practice (Goodfellow et al. 2016).

K-Fold cross validated grid search is a method of model tuning which "trains a model for every joint specification of hyperparameter values in the Cartesian product of the set of values [given] for each hyperparameter" (Goodfellow et al. 2016). The data set is randomly allocated into k train and test sets and gathers each iteration's training and testing score. The average train and test error for each parameter setting over the k folds is recorded and the grid search algorithm selects the best performing mean test error as the optimal parameter settings (Buitinck et al. 2013). The benefit of this techniques lies in its ability to exhaustively search the response surface of the algorithm. Its drawback lies in the time it takes to execute that exhaustive search. For example, a search for two parameters, each set at 5 different levels and a three-fold cross validation, would have 75 total replications. If each replication took approximately a minute, this would take 75 minutes to complete. One can see how this time could easily expand exponentially with more parameters, more levels, and more folds.

Layer Widths

We conducted the first three-fold CV grid search for the width of each hidden layer. Three folds were chosen because of the computational intensity of searching over five sets. The search was conducted over the range of 50 to 250 in increments of 50, for a total of 5 levels. This search occurred independently at each hidden layer. With 625 replications per fold, the network was trained total of 1,875 times. The optimal train/test values were found when layer two had a width of 100, layer three had a width of 200, layer four had a width of 150, and layer five had width of 100.

Learning Rate

After finding the optimal layer widths, we conducted a three-fold cross validated line search for the learning rate. A line search is still cross validated but done by itself and independently of any other parameters. The search was conducted on the scale from 10^{-1} to 10^{-6}. This initial search yielded 10^{-3} as the most optimal value. From there, we refined the search from 0.001 to 0.1 in 10 linearly spaced increments. The final learning rate was set to the optimal value found at 0.009.

Dropout Rate

We conducted a three-fold cross validated linear search on dropout rate. Table 3.4 below describes the search parameters. The line search found that the rate 0.015 had the lowest train/test error values among its alternatives. Since the value was at neither extremum of the search bounds, we infer that the search took place within a reasonable range (Goodfellow et al. 2016).

Table 3.4. Dropout Rate Search Parameters

Dropout Rate	0.001	0.0025	0.005	0.0075	0.01	0.0125	0.015	0.0175	0.02

L2 Regularization

For the L2 λ parameter, the three-fold cross validated line search found 0.03 as the optimal hyperparameter value. The search was conducted over the range in Table 3.5.

Table 3.5. L2 Regularization Search Parameters

λ Parameter	0.01	0.02	0.03	0.04	0.05

Batch Size and Epoch

The search for batch size and epochs was conducted jointly with a grid design because they both make a large impact on model training time (Brownlee 2018). The three-fold cross validated grid search resulted in a total of 120 runs, stemming from the five batch size search parameters and the eight epoch search parameters. The optimal combination found based on overall mean model test accuracy was 256 samples per batch and 120 epochs. The full search range can be found below in Table 3.6.

Table 3.6. Batch Size and Epoch Search Parameters

Batch Size	64	128	256	512	1024			
Epoch	40	60	75	100	120	160	180	200

3.3.2 Final Model

In summary of Section 3.3.1, this section will state and briefly discuss the final model selected. The total model contains six layers–an input layer, four hidden layers, and an output layer. While the NOLH experiments yielded a model with six hidden layers, we chose the four hidden layer model. The difference in accuracy and F1 score between the two was marginal. Choosing a smaller model would take less time to train with fewer layers and also take less time to tune with exhaustive grid searches. The grid search for layer width alone was reduced from a space tuning six layers with five levels to four layers with five levels. When calculated, this is becomes the stark reduction from 15,625 replications per fold to 625 replications per fold, making an exhaustive search computationally possible. This exhaustive search yielded layer widths for the four hidden layers found in Table 3.7.

Table 3.7. Final Model Layer Widths

Layer #	Width
1	36
2	100
3	200
4	150
5	100
6	9

The model settings found in Table 3.8 are largely necessary for categorical classification in the Keras API (Chollet et al. 2015). These choices are discussed in the very beginning of Section 3.3.

Table 3.8. Final Model Settings

Model Settings	Type
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Hidden Layer Activation	ReLu
Output Layer Activation	Softmax

Finally, Section 3.3.1 also discusses the procedures used to find the tuned hyperparameter values in Table 3.9. These were found using k-fold cross validated linear and grid searches. Choosing these values will maximize this specific algorithm's potential for correctly classifying the shapes of ADS-B tracks according to our classification system.

Table 3.9. Hyperparameter Values

Hyperparameter	Value
Learning Rate	0.009
Dropout Rate	0.015
L-2 Regularization	0.03
Batch Size	256
Epoch	120

The application and results of the multi-category classifier discussed in this section will be explored in detail in the Chapter 4.

CHAPTER 4: Network Performance Results and Analysis

The following section explores the results of the classifier discussed in Section 3. First we will explore the different performance metrics used to evaluate the classifier. After that, we will discuss the classifiers effectiveness on unseen validation data and its mean performance on the full data set using cross validation. Finally, we will explore the robustness of the model by applying it to a binary classification task on the same data set.

4.1 Key Performance Metrics

Accuracy

Classification accuracy is defined by Equation (4.1). In the case of binary prediction, true positive denotes the model's correct prediction of a certain sample belonging to the positive class. A true negative refers to a correct prediction of a certain sample belonging to the negative class. When a sample is predicted as positive, but is actually negative, this is a false positive. When a sample is predicted as negative, but is actually positive, this is a false negative. In the context of multiple classes, the sum of the diagonal of the confusion matrix is equal to the numerator of Equation (4.1) while the denominator remains unchanged:

$$\text{Accuracy} = \frac{\text{True Positive } + \text{ True Negative}}{\text{Total Samples Predicted}}. \tag{4.1}$$

In the following section, we will be reporting the network's accuracy, but we will not put a lot of emphasis or weight behind that simple number. Accuracy falls short in a number of attributes. The metric assumes equal misclassification cost for false positive and false negative errors. This can be costly in a realistic system, because one has the possibility of being much more costly than the other. Accuracy also inherently assumes that "the class distribution is known for the target environment" (Provost et al. 1998). In this classification problem, the data set is highly skewed with a disproportionate amount of type 1 labels as seen in Figure 3.4. We will emphasize the use of precision, recall, and the F1-score, because accuracy results may be optimistically misleading to the untrained eye (Davis and Goadrich 2006).

Precision and Recall

In a classification task with multiple classes and a large skew in class distribution as seen in Figure 3.4, precision and recall give more insight into an algorithm's performance than accuracy and a Receiver-Operator Curve (ROC) (Davis and Goadrich 2006). Precision and recall are defined in Equations (4.2) and (4.3) below (Buitinck et al. 2013):

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive } + \text{False Positive}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive } + \text{False Negative}}. \tag{4.3}$$

These metrics stem from the information retrieval field (Davis and Goadrich 2006). Precision is a measure of result relevancy. Within the context of our problem, of the tracks classified as type 5, what proportion of tracks were actually type 5? Recall is a measure of how many truly relevant results are returned. Of the tracks that were type 7, what proportion of tracks were classified as type 7?

A high precision results in a low false positive rate and a high recall results in a low false negative rate. When both metrics are high, the classifier is returning accurate results and a majority of all positive results. Ideally, a classifier would have both high precision and high recall, but this is rarely the case in practice. Due to the classification threshold of a model, a trade off between precision and recall tends to exists. There exists points in the precision-recall curve where increasing one may decrease the other and vice-versa (Buitinck et al. 2013). It becomes difficult to attempt to maximize both as well as determine which one takes priority over the other. For that reason the F1 score was created.

F1 Score

F1 score is defined as the harmonic mean of precision and recall. It reaches its best value at one and its worst value at zero while equally weighting both precision and recall (Buitinck et al. 2013). This allows us to maximize one value instead of two and easily interpret the value's effect on predictive capacity. That is, a higher value indicates more overall predictive capacity:

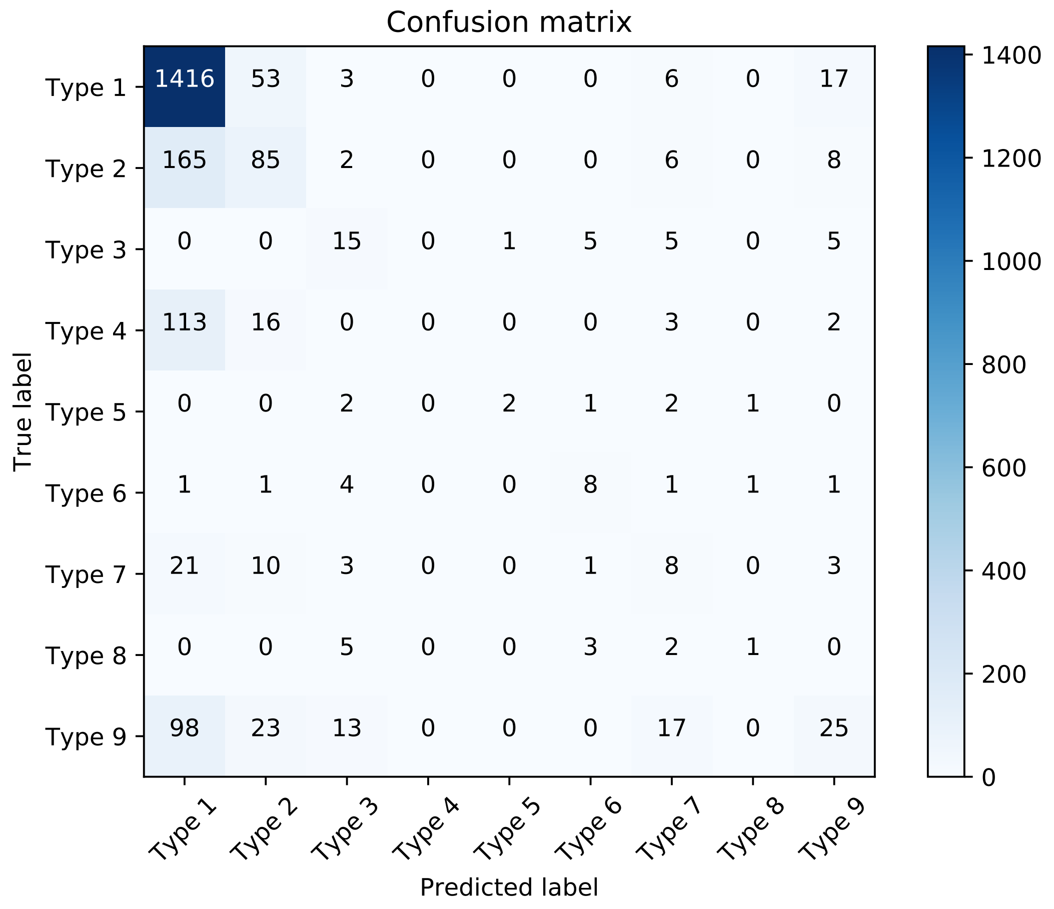
$$\text{F1 Score} = 2 * \frac{\text{Precision} \ * \ \text{Recall}}{\text{Precision} \ + \ \text{Recall}}. \tag{4.4}$$

For each class, we calculated the precision, recall, and F1 score for each class. We also calculated these metrics' macro averages and weighted averages. The macro average is the unweighted mean of each metric across all the classes. This does not take label imbalance into account. The weighted average calculates the weighted average score across all labels according to the number of true instances of that label (Buitinck et al. 2013).

4.2 Multi-Label Classifier Results–Validation Set

4.2.1 Results and Analysis

The multi-label classifier trained and tuned using the training and test sets. In this section, we evaluate the model on the validation set. This set is composed of 20% of the entire data set, totaling 2184 tracks and has never been seen or used by the model up until this point. The category distribution of this set largely mirrors the breakdown of the entire data set. This can be seen in Figure 3.4. The confusion matrix in Figure 4.1 illustrates the predictions made by the network.



	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	Type 8	Type 9
Type 1	1416	53	3	0	0	0	6	0	17
Type 2	165	85	2	0	0	0	6	0	8
Type 3	0	0	15	0	1	5	5	0	5
Type 4	113	16	0	0	0	0	3	0	2
Type 5	0	0	2	0	2	1	2	1	0
Type 6	1	1	4	0	0	8	1	1	1
Type 7	21	10	3	0	0	1	8	0	3
Type 8	0	0	5	0	0	3	2	1	0
Type 9	98	23	13	0	0	0	17	0	25

Figure 4.1. Multi-Label Classification Confusion Matrix–Validation Set

Numbers in the matrix are shaded according to their magnitude; lighter blue is small and darker blue is large. The elements along the diagonal of this matrix are the correctly classified samples. Type 1, 2, 3, and 6 initially appear to have significant true positives. The more detailed categorical breakdown is summarized below.

Table 4.1. Categorical Classification Statistics–Validation Set

	Precision	Recall	F1 Score	Samples
Type 1	0.78	0.95	0.86	1495
Type 2	0.45	0.32	0.37	266
Type 3	0.32	0.48	0.38	31
Type 4	0	0	0	134
Type 5	0.67	0.25	0.36	8
Type 6	0.44	0.47	0.46	17
Type 7	0.16	0.17	0.17	46
Type 8	0.33	0.09	0.14	11
Type 9	0.41	0.14	0.21	176

Type 1 (Straight Line)
Type 1 tracks were the most abundant in the data set and this distribution is mirrored in the validation set. It was also the most simple track. From the precision, we see that of the 1814 results predicted as type 1, 78% of them or 1416 were actually type 1. The recall indicates that of 1495 that were actually type 1, 1416 or 95% were classified as type 1. This indicates that the network is largely successful in predicting items in type 1, but it had a higher misclassification rate on tracks that were not actually type 1. The F1 score of 0.86 for this category is high compared to the rest of the labels. This indicates the network was most successful in classifying tracks of this type.

Type 2 (Straight with Slight Deviation)
Type 2 tracks are the second largest in quantity among the distribution, but the algorithm performed worse compared to type 1 results. Of the 188 tracks predicted to be type 2, 85 or 45% of those tracks were correctly classified. Given the recall statistic, we see that 32% of the results that were actually type 2 were correctly classified as type 2. These two statistics result in a F1 score of 0.37. The network misclassified type 2 tracks as type 1 most frequently. We believe this to be an artifact of the two categories' similarities. Type 2

tracks are typically straight with small deviations towards the beginning and end. An input vector with a majority of low bearing changes and high radius curvatures could be more easily predicted as a straight line than other shapes.

Type 3 (Single Loop/Circle)
Tracks of type 3 are low in representation among the distribution of the data set. However, the network performed with a precision of 32%. That is, of the 47 samples predicted to be type 3, 15 of them were actually type 3. The recall of 48% means that of the 31 tracks that were actually type 3, 15 of them were correctly classified. The network did not misclassify any type 3 tracks as type 1 or 2. This could be due to the relatively consistent curvature radii. The relatively high F1 score of 0.38 for such few samples implies that the network may possibly perform better in this category if given more samples.

Type 4 (Parabola)
This network was unable to correctly classify any type 4 tracks. As a result, precision, recall, and the F1 score were zero. Of the 134 samples, 113 were predicted to be type 1. Upon visual inspection, type 4 tracks take a parabolic shape. A small series of bearing changes and a larger series of curvature radii could be predicted as a straight line instead of a parabola. This issue could also stem from the misclassification of type 1 and type 4 tracks by the human operators. One operator could have stricter or more lax standards for what he or she considers a parabola or a line that is essentially ‘straight’.

Type 5 (Multiple Loops)
Type 5 tracks had the lowest representation among all categories. However the network performed surprisingly well considering it had 38 samples in the train/test sets and nine in the validation set. Of the three tracks predicted to be type 5, two were actually type 5, giving the class 67% precision. Of the eight labels that were actually type 5, two were predicted to be type 5 giving the class’ recall a score of 25%. Due to such a low sample size, little can be inferred about the characteristics of the shape. The network misclassified most type 5 labels as type 3 or type 7. Type 3 is characteristically similar to type 5. Both are loops. Type 7 is a more peculiar misclassification. The shapes do not seem to share

many characteristics in bearing change or curvature radius. More samples are needed to further explore this network's ability to classify type 5 tracks.

Type 6 (Out and Back)

The 'out and back' track shape which is classified as type 6 saw a precision of 44% and recall of 47%. Of the 17 tracks that were actually type 6, the network primarily misclassified these as type 3. Both types contain a series of bearing changes that eventually return them within a close proximity of their starting point. The 'out and back' track does not take on a circular shape so its curvature radii should be lower than tracks such with circular characteristics such as the loop, multi-loop, and figure eight. In order to better explore the network's ability to classify this track, more recorded tracks of this type would help.

Type 7 (Switchback/ Lightning)

Type 7 tracks were not accurately classified by this network. In the context of this problem, the probability of randomly guessing a track's shape is $\frac{1}{9}$ or 11%. Of the 46 samples that were type 7, only eight were correctly classified. Of the 50 tracks that were predicted to be type 7, only eight were actually type 7. The network predicted 21 of the 46 samples to be type 1 and 10 samples to be type 2. Types one, two, and seven have the common attribute of being approximately straight for most of the track. This results in a series of low bearing changes and high radius curvatures. Due to the very few sharp turns in a type 7 track, the network may predict the track to be a straight track with slight deviations (type 2) or a fully straight track (type 1), attributing the high magnitude bearing changes to noise. Since classes seven and two share many similar features and class seven has such a low sample size, there may be benefits to splitting the class and reclassifying the tracks as either type 1 or type 2 tracks.

Type 8 (Figure Eight)

Of the three tracks the algorithm predicted to be type 8, one was actually type 8, giving the category a 33% in precision. While this may initially appear like a significant value, one must look at it with the context of how many samples the network attempted to classify as type 8. Due to such a low sample size, this precision statistic does not say much about the network's ability to properly classify figure eight tracks. This manifests in the recall statistic, where one out of the 11 tracks that were actually type 8, only 2 were correctly classified. Most misclassifications occurred with type 3 and type 6. Type 8 shares very similar characteristics with both tracky types. Type 3 contains a series of moderately sized curvature radii giving it a circular shape. A figure eight contains two circular shapes, possibly leading to the misclassification. Type 8 also contains a series of bearing changes that lead it into close proximity to its starting point. This similarity with the out and back track (type 6) could be an underlying reason for the misclassification.

Type 9 (Sinusoid)
Track type 9 was the third most prominent track type in the distribution. Of the 176 tracks available in the validation set, the network correctly classified 25. The network misclassified a majority of type 9 tracks as type 1 and type 2. This result could be similar to the results we saw with type 4 tracks. Since the sinusoidal shape can see a long series of low bearing changes, particularly in the middle, it could be classified as straight or straight with small deviations at the beginning and end.

4.2.2 Summary Statistics Analysis–Validation Set

Table 4.2 summarizes the precision, recall, and F1 score across all track types previously discussed. The macro-averages represent the unweighted mean of each metric across all the classes. The weighted averages represent the mean scores across all types, weighted by the number of samples in each category. The overall accuracy, defined in Equation (4.1), is not shown in the table below, but was calculated to be 71%. This means that 71% of the entire data set predicted was correctly classified. This does not take class imbalance into account. Since a large amount of correct classifications stem from the type 1 label which also has the largest representation, the accuracy metric is skewed.

Table 4.2. Multi Category Classifier Summary Metric Table–Validation Set

Metric	Macro-Average	Weighted-Average
Precision	0.40	0.64
Recall	0.32	0.71
F1 Score	0.33	0.66

The average precision and recall scores tend to be hard to interpret over multiple classes. Because of this, the F1 score becomes the main statistic of interest when exploring averages. Unweighted, the average F1 score among the nine categories is 0.33. When weighted by the number of samples per class, the F1 score sees a drastic increase to 0.66. This is similar to accuracy because the high sample size of type 1 skews the weighted F1 score.

4.3 Multi-Label Classifier Results–10-Fold Cross Validation

After testing the network on the validation set, we examined what its performance may look like on average using Cross Validation (CV). The full data set was split randomly and evenly into 10 folds. Those 10 folds were further divided into train and test sets where the model was trained and evaluated on each set respectively. The confusion matrix in Figure 4.2 illustrates the results of the cross validated predictions.

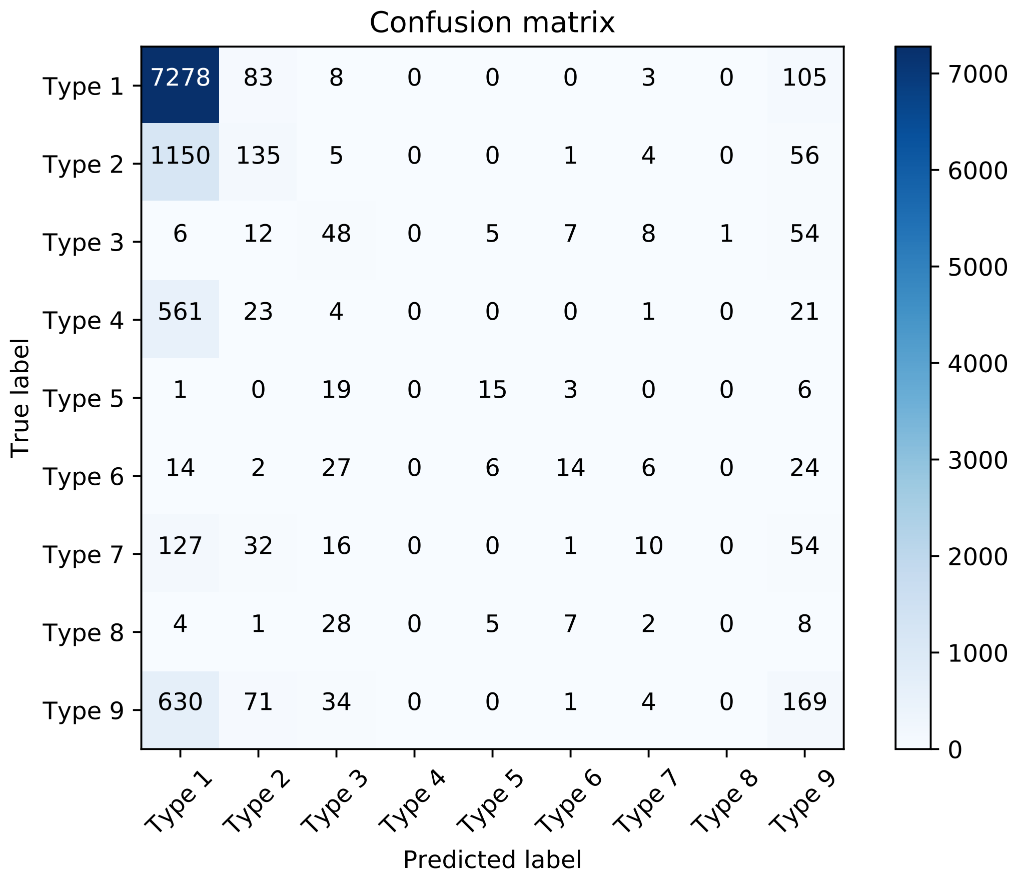


Figure 4.2. Multi-Label Classification Confusion Matrix–10-Fold CV

A categorical summary table similar to the validation results was also produced in Table 4.3.

Table 4.3. Categorical Classification Statistics–10-Fold CV

	Precision	Recall	F1 Score	Samples
Type 1	0.74	0.97	0.84	7477
Type 2	0.38	0.10	0.16	1351
Type 3	0.25	0.34	0.29	141
Type 4	0.00	0.00	0.00	610
Type 5	0.48	0.34	0.40	44
Type 6	0.41	0.15	0.22	93
Type 7	0.26	0.04	0.07	240
Type 8	0.00	0.00	0.00	55
Type 9	0.34	0.19	0.24	909

The 10-fold cross validation saw similar results to the validation set predictions in types 1, 4, 5, and 9. Significant differences occurred in types 2, 3, 6, 7, and 8 which will be discussed below.

Type 2 (Straight with Slight Deviation)
Type 2 saw a slight decrease in precision and a large decrease in recall from 0.32 to 0.10. This translates to a very high false negative rate. That is, many tracks whose ground truth label was type 2 were classified as types other than two. The trend of type 2 also parallels the validation results in the common misclassification to type 1.

Type 3 (Single Loop/Circle)
'Single-loop' tracks showed promising results in the validation set results, but the class' F1 score decreased by 0.09 indicating a drop in both precision and recall. Results from the confusion matrix suggest that the majority of misclassifications were type 9. This result initially seems strange, but upon further consideration it's very plausible. Type 3 tracks are usually broken up into two different types of loops. Some contain multiple low magnitude bearing changes with the occasional turn, making it a square loop and some

contain moderate change in bearing throughout the track making it more circular. Type 9 or the "sinusoidal" shape usual contains a curve or moderate change in bearing with a small radius that transitions to a linear trajectory with another moderate change in bearing with a small radius at the end. The only difference between the sinusoid and a loop is the direction of the turn.

Type 6 (Out and Back)

Type 6 saw a significant drop in recall similar to type 2 in the cross validated results. The recall dropped from 0.47 to 0.15. This drastic increase in the false negative rate may be explained by the increase in exposure to the entire set of type 6 tracks. The network misclassified 27 of 93 results as class 3. Types 3 and 6 share the unique characteristic of ending in close proximity to where they began. This manifests in both types as a series of bearing changes and curvature radii that direct the trajectory back to its origin. Its clear that this could cause a misclassification.

Type 7 (Switchback/Lightning Pattern)

The cross validated output for type 7 was very different from the validation results. The precision starkly increased from 0.16 to 0.26 and the recall significantly dropped from 0.17 to 0.04. Of the total results that were classified as type 7, 26% of them were correct. However, of the actual tracks that were labeled type 7, only 4% were correctly classified. Most were classified as type 1. This also happened in the validation results, but the increase in misclassifications was not proportional. This could be due to the fact that type 7 tracks are largely straight. The low bearing change and high curvature radius may outweigh its other features in the model's prediction.

Type 8 (Figure Eight)

In the cross validation, type 8 was never correctly classified. Of the 55 samples, 28 were classified as type 3. This misclassification makes sense. Both type 3 and type 8 are very circular in nature. Type 8 just contains two circles vice one. Regardless, both shapes contain a series of bearing changes and curvature radii that result in near complete circumnavigation.

The statistics tabulated in Table 4.4 demonstrate an overall decline from the validation results in Table 4.2. The accuracy of the network differed by 1%, averaging at 70% overall.

Since the data set was significantly skewed in distribution, this 70% should be observed with scrutiny.

Table 4.4. Multi-Label Classifier Summary Metric Table–10-Fold CV

Metric	Macro-Average	Weighted-Average
Precision	0.32	0.60
Recall	0.24	0.70
F1 Score	0.25	0.63

The average overall precision and recall both decreased by 8% while the weighted metrics remained relatively the same. Similar to accuracy, these results are skewed by class distribution. The F1 score of the cross validation also dropped 0.08 points to 0.25 due to the 8% decrease in precision and recall.

This disparity between the validation set results and the 10-fold cross validated results can be explained through the size of the train and test sets. The validation set makes up 20% of the full data set. The model was trained using 60% of the full data set which gave it a larger ability to train on each class. When the data set was divided into 10 sets and further divided into train and test sets, the model may not have been able to adequately train on all types due to the skewed distribution. For example, the data set only contains 55 samples of type 8 tracks. If divided into 10 sets, each set will contain five or six samples on average to use for training and testing.

4.4 Binary Classifier Results

In order to explore the robustness of this particular model, we modified the data set to create a binary classification problem. The data was split into two label types. Type 1 is the same as before–an approximately straight track. Type 0 encompasses every other track type (i.e., Types 2 through 9). In order to accommodate the altered input, the network needed to be slightly modified. The output layer was changed from softmax to sigmoid and the loss function was changed to binary cross entropy. The output layer still produces probabilistic categorical prediction, but in a binary context. The loss function is still mathematically similar and works over a set of two categories as opposed to nine. Results from this test indicate this specific model's ability to distinguish between an approximately linear and non-linear track. The evaluation was conducted using five fold cross validation.

Table 4.5 breaks down the precision, recall, and F1 score for each track type. The computed accuracy of the classifier on the binary data was 77%.

Table 4.5. Binary Classification Report

Track Type	Precision	Recall	F1 Score
Type 0	0.73	0.44	0.55
Type 1	0.78	0.92	0.85

The confusion matrix in Figure 4.3 illustrates a categorical breakdown of the classifier's results. Of the 7477 actual type 1 tracks, 6915 were correctly classified and 562 were false negatives. This yields a false negative rate of 7.5%. Of the 3442 tracks that were actually type 0, 1508 of them were classified correctly and 1934 were false positives. This yields a 56.2% false positive rate. If our goal is to successfully indicate if an aircraft is flying in non-linear path, this classifier performs rather poorly.

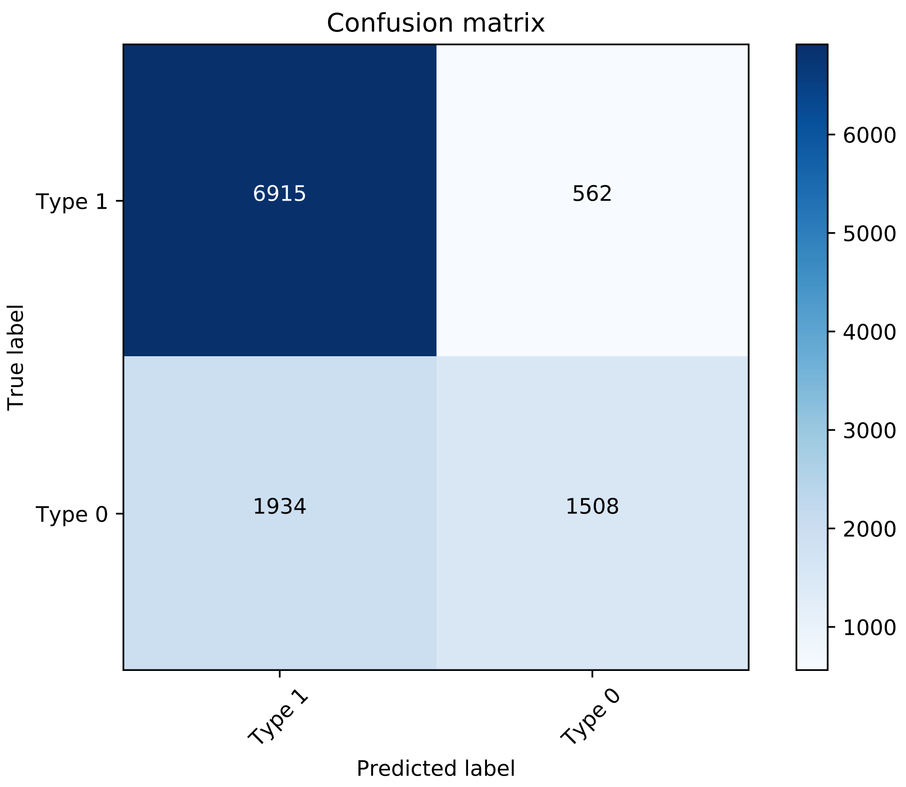


Figure 4.3. Binary Classification Confusion Matrix

In Figure 4.4, we plot the five-fold cross-validated receiver-operator curve. The curve has a mean area under the curve of 0.66. This outperforms a 50% chance guess, as denoted by the chance line, by 16%.

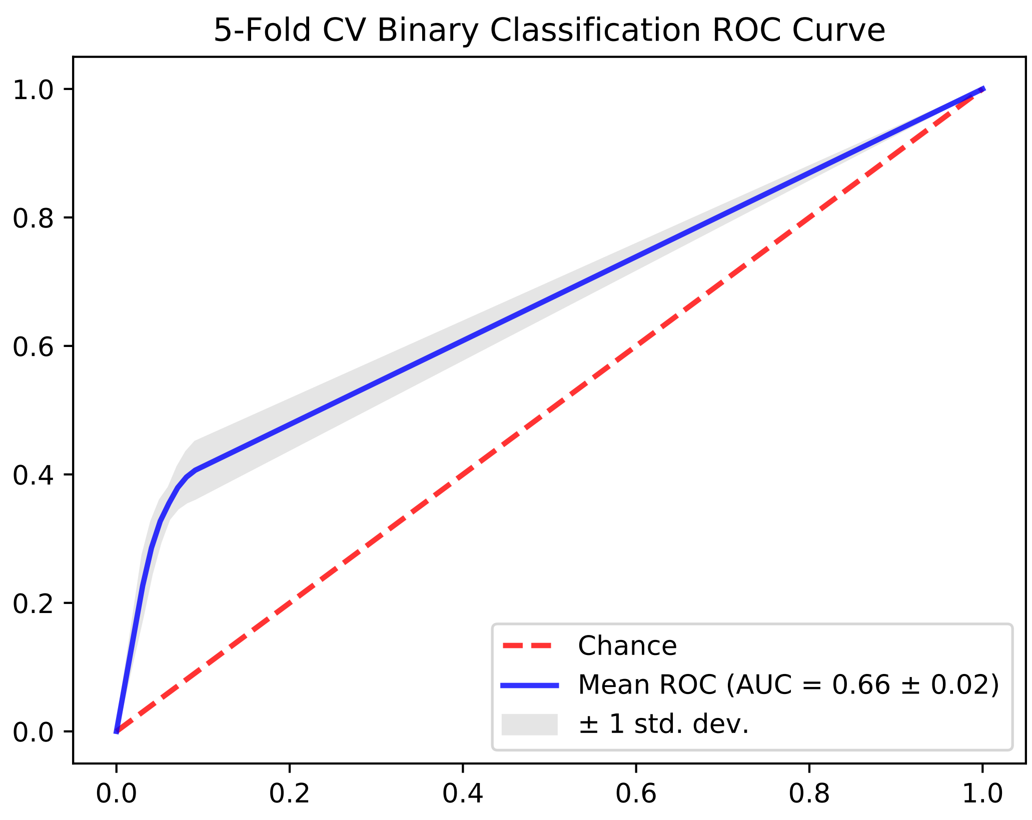


Figure 4.4. Binary Classification Receiver-Operator Curve

The confidence interval width of this ROC curve indicates that this particular model may be overfitting on the binary data. We determined with experimental design that this task needed four layers of different widths to classify a track into nine categories. When the solution space is severely reduced to two track types, the model may not need as many layers because the problem becomes much simpler. This overfitting and high false positive rate indicate that this model is not optimally constructed for use with binary data and should be built and tuned with this specific task in mind.

CHAPTER 5: Conclusion and Future Work

5.1 Conclusion

Using a classification algorithm with a high predictive capacity to flag anomalous flight patterns could not only save thousands of man hours, but also save many lives in the right circumstances. Given the volume of traffic in today's airspace, it is impossible to monitor every flight and determine its potential to be a threat. Because of this, the Department of Defense and civilian intelligence agencies could benefit greatly if given the capacity to automate certain areas of signals intelligence.

Using algorithms we have developed to sort, clean, and visually display thousands of raw ADS-B tracks, we were able to visually classify 10,920 flight trajectories into nine shape categories. After researching existing classification and prediction methods in Chapter 2, we were able to analyze and calculate key features, such as change in bearing and curvature radius, existing in each flight trajectory.

We then developed a multi-label classifier using Python and the Keras machine learning platform. Specifically, we built a dense feed-forward neural network. In order to properly develop the model, we employed the NOLH to explore the number of layers necessary for classification. The network with four hidden layers then underwent an extensive hyper-parameter tuning process which included two full-factorial cross-validated grid searches and three cross-validated line searches. The final model in Section 3.3.2 was evaluated for classification accuracy using a validation set and 10-fold cross validation. The network's robustness was also evaluated on a binary classification exercise.

The model was evaluated on the validation set using metrics explained in Section 4.1. The network performed with an overall accuracy of 71%. Predictive capacity in Types 1, 3, and 6 appeared promising with F1 scores of 0.86, 0.38, and 0.46. Types 4, 7, 8, and 9 demonstrated poor predictive capacity with F1 scores of 0.00, 0.17, 0.14, and 0.21. The overall unweighted average F1 score was 0.33 with a weighted F1 score of 0.66.

To further explore the model's predictive capacity, we conducted 10-fold cross validation on the full data set. Results from this exercises yielded performance metrics that were significantly lower than the validation set. Large decreases in F1 scores manifested in track types 2, 3, 6, 7, and 8, leading to an overall drop in performance averages. The cross validation yielded a macro-average F1 score of 0.25, a weighted F1 score of 0.63 and an accuracy of 70%. We attempted to explain these variations in the results by speculating the division of the data into ten train and test splits was too small to adequately train the classifier.

Finally, the model was applied to the same data set with a different classification system. Types 2 through 9 were merged into Type 0 with the goal of applying the model to linear/non-linear binary classification. The model performed with a false negative rate of 7.5% and a false positive rate of 56.2%. In the context of this classification, a false negative would represent an aircraft trajectory which is classified as Type 0 or nonlinear, but was actually Type 1 and linear. Assuming type 1 is a non-threatening flight pattern, this has a very low cost impact. On the other hand, a false positive in this context indicates an flight track that is actually nonlinear, but has been classified as linear. Using the previous assumption of prediction cost, this false positive has an extremely high cost impact, making the network questionable for use on binary targets.

While this dense feed-forward neural network contains multiple shortcomings in predictive capacity, it establishes foundational support for further research into the classification of ADS-B trajectory shapes using neural networks. All classes except Types 4, 7, and 8 maintained predictive capacity above the chance threshold (11.1%). Types 1, 3, 5, and 6 showed promise with the highest F1 scores in both validation set and cross validation results.

5.2 Future Work

The results outlined in Chapter 4 demonstrate significant shortfalls in the classifications of certain shapes shown in Figures 3.1, 3.2, and 3.3. Since some of these share similar characteristics, there are a few ways this classification method could be improved. More category specific data, especially for track types 5, 6, and 8, could be collected to further increase predictive capacity support during the model training.

While room for improvement in the model will always exist, this classification problem may benefit greatly by specifying and codifying the visual characteristics necessary for a track to be deemed one of the labeled categories. Many tracks contain attributes that make them questionably belong to two or more categories. For example, strict criteria could be imposed to better define single loop, multi-loop, and figure eight tracks. These all contain circular characteristics which the classifier was unable to distinguish, leading to multiple misclassifications in this subset. Stricter visual differences such as loop size, number of loops, and proximity to starting point could be determined to better define each class.

Alternatively, more class labels could be added in order to identify more unique variations of each shape that appear frequently. This enlarged response space could increase the specificity of the model and make each label type more distinguished from others. While good in theory, this practice would require a large expansion of the data set. As seen with type 8 (figure eight) and type 5 (multi-loop), nearly 50 instances of each were seen in a data set of over 11,000. This does not include the samples that did not belong to any class. The network would also require enough support in each of the additional categories to successfully cross validate.

This body of research would also benefit from exploring multiple methods for predicting and classifying ADS-B data. The deep sequential neural network outlined in this study used the input vectors with no regard for their temporal position in the vector and only used the calculated features of bearing change and curvature radius. More features such as distance traveled, raw coordinates, and distance between start and end point could be possibly useful inputs to a neural network. As discussed in Chapter 2, RNNs may be more suitable to the temporal nature of flight trajectory. CNNs were also discussed in Chapter 2. While more computationally intensive, a CNN could use images of each track in order to develop visual characteristics that best distinguish each track from another. A RNN or CNN that explores the classification ability of shapes and anomaly detection could be a significant addition to this body of work.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Akerman S, Habler E, Shabtai A (2019) VizADS-B: Analyzing sequences of ADS-B images using explainable convolutional LSTM encoder-decoder to detect cyber attacks. Cornell University, https://arxiv.org/abs/1906.07921.

Baekkegaard S, Blixenkrone-Moller J, Larsen JJ, Jochumsen L (2018) Target classification using kinematic data and a recurrent neural network. *2018 19th International Radar Symposium (IRS)*, 1–10.

Brownlee J (2018) *What is the difference between a batch and an epoch in a neural network?* Machine Learning Mastery, Vermont Victoria, Australia.

Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G (2013) API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Calderara S, Prati A, Cucchiara R (2011) Mixtures of Von Mises distributions for people trajectory shape analysis. *IEEE Transactions on Circuits and Systems for Video Technology* 21(4):457–471.

Chollet F, et al. (2015) Keras. https://keras.io.

Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning*, 233–240, ICML '06 (Association for Computing Machinery, New York, NY, USA), ISBN 1595933832, URL http://dx.doi.org/10.1145/1143844.1143874.

FAA (2020) Automatic dependent surveillance-broadcast (ADS-B). FAA, Washington, DC.

Goodfellow I, Bengio Y, Courville A (2016) *DeepsLearning* (MIT Press), http://www.deeplearningbook.org.

Junejo IN, Javed O, Shah M (2004) Multi feature path modeling for video surveillance. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, 716–719 (IEEE).

Li L (2013) *Anomaly detection in airline routine operations using flight data recorder data*. Ph.D. thesis, Massachusetts Institute of Technology.

Liu Q, Wu S, Wang L, Tan T (2016) *Predicting the next location: A recurrent model with spatial and temporal contexts* (AAAI Publications).

Liu Y, Hansen M (2018) Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach. Cornell University, https://arxiv.org/abs/1812.11670.

Nadeem Anjum AC (2008) Multifeature object trajectory clustering for video analysis. *IEEE Transactions on Circuits and Systems for Video Technology* 18(2):1555-1564.

Ognjanovski G (2019) Everything you need to know about neural networks and backpropagation-machine learning made easy. Towards Data Science, San Francisco, CA.

Peltarion (2020) AI glossary: Deep learning definitions. Peltarion platform.

Prati A, Calderara S, Cucchiara R (2008) Using circular statistics for trajectory shape analysis. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8 (IEEE).

Provost F, Fawcett T, Kohavi R (1998) The case against accuracy estimation while comparing induction algorithms. *Proceedings of the Fifteenth ICML Conference,* 445-453.

Sanchez SM (2011) NOLH designs spreadsheet. URL http://harvest.nps.edu.

Semke W, Allen N, Tabassum A, McCrink M, Moallemi M, Snyder K, Arnold E, Stott D, Wing MG (2017) Analysis of radar and ADS-B influences on aircraft detect and avoid (daa) systems. *Aerospace* 4(3):49.

Shi Z, Xu M, Pan Q, Yan B, Zhang H (2018) LSTM-based flight trajectory prediction. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (IEEE).

Sillito RR, Fisher RB (2008) Semi-supervised learning for anomalous trajectory detection. *BMVC*, Volume 1, 035–1.

Srivastava N (2013) Improving neural networks with dropout. *University of Toronto* 182(566):7.

Starizona (2020) Field curvature. Starizona, Tucson, AZ.

Tao Y, Faloutsos C, Papadias D, Liu B (2004) Prediction and indexing of moving objects with unknown motion patterns. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 611–622.

van Laarhoven T (2017) L2 regularization versus batch and weight normalization. Cornell University, https://arxiv.org/abs/1706.05350.

Yann Le Fablec JMA (1999) Using neural networks to predict aircraft trajectories. Technical report, Ecole Nationale de l'Aviation Civile, Cedex, France.

Zhang Y, Qiao J (2008) ADS-B radar system. U.S. Patent 7,414,567.

Zhang Z, Sabuncu M (2018) Generalized cross entropy loss for training deep neural networks with noisy labels. Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, eds., *Advances in Neural Information Processing Systems 31*, 8778–8788 (Curran Associates Inc.).

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California