Optimizing Neural Networks for Enhancing Air Traffic Safety

by

Geoffrey T. Cooney

Submitted to the Depart of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science and
Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 6, 2004

Copyright 2004 Geoffrey T Cooney. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Signature of Author . . .

Geoffrey T. Cooney
Department of Electrical Engineering and Computer Science
February 6, 2004

Certified by . . .

Dr. Amar Gupta
Co-Director, Productivity From Information Technology Initiative
Sloan School of Management
Thesis Supervisor

Accepted by . . .

Arthur C. Smith
Chairman, Department Committee on Graduate Theses
Abstract:

This thesis contains the process and results related to optimizing a neural network to predict future positions of airplanes in the vicinity of airports. These predicted positions are then used to calculate future separation distances between pairs of airplanes. The predicted values of the separation distance are used to ensure adequate distances between adjacent aircrafts in the air and, if necessary, to create early warning alarms to alert air traffic control tower personnel about planes that may pass too near each other in the immediate future.

The thesis covers three areas of work on this topic. The first section involves optimizing a neural network for Chicago O Hare Airport. The second is related to gathering data on the performance of this network in different scenarios. These data can be used to determine if the different days/runways have different characteristics. The final phase of this document describes how to generalize the process used to build an optimized neural network for Chicago O Hare airport in order to provide the capability to easily recreate the process for another airport.
Acknowledgements

I would like to thank all the people who helped and supported me during my thesis. Specifically, I would like to thank my fellow team members on the FAA project, Anuja Doshi and Yoshi Nakanishi without whose help I would never have been able to finish. I also like to thank everyone who worked on the project for me laid the groundwork for my own work.

My family and girlfriend, Sara Bissonnette, provided me with constant support throughout the entire process. I'd like to thank Sara especially for not getting too mad at me for always being on her laptop working.

Finally, I'd like to thank my supervisors throughout my time at MIT; Professor James Kirtley for constantly guiding me as both an undergraduate and a graduate; and Dr. Kent Lundberg for being patient with me in designing my AUP. Without Dr. Amar Gupta, I never would have finished this project on time. Thank you for your constant guidance and helping to keep me focused. I'd like to thank Dr. Rafael Palacios for always being there to bounce ideas off, helping to solve some of the more difficult problem, and generally holding my thesis to a high technical standard.
## Table of Contents

**Acknowledgements** ........................................................................................................ 3  

1 **Introduction** ....................................................................................................................... 7  

1.1 Overview of Work ............................................................................................................. 7  

1.2 Goal of Work .................................................................................................................... 7  

2 **Background** ....................................................................................................................... 8  

2.1 AMASS Overview .......................................................................................................... 8  

2.2 Neural Networks ............................................................................................................. 10  

2.3 Prior Work on Neural Networks in AMASS ....................................................................... 13  

2.3.1 Neural Networks for Generating Alarms ...................................................................... 13  

2.3.2 Neural Networks that predict separation distance ....................................................... 14  

3 **Related Work** ................................................................................................................... 16  

3.1 Runway Incursion Prevention Techniques ........................................................................ 17  

3.2 Aircraft Related Neural Networks .................................................................................. 20  

3.3 Time Series Applications of Neural Networks .................................................................... 22  

4 **Overview of Work** ............................................................................................................ 25  

4.1 Generating an Optimized Neural Network for Chicago O'Hare Airport .......................... 25  

4.2 Verifying Accuracy of Optimized Neural Network ......................................................... 26  

4.3 Generalizing Optimization Process .................................................................................. 27  

5 **Training Set** ......................................................................................................................... 28  

5.1 Training Set Events ......................................................................................................... 28  

5.2 Extraction of Events by Runway ...................................................................................... 31  

5.3 Validation of Extraction ................................................................................................. 36  

5.3.1 Visual Validation Tools ............................................................................................... 37  

5.3.2 Counting Validation Tools ......................................................................................... 38  

5.3.3 Results of Validation Tools ........................................................................................ 39  

5.4 Description of Final Extraction Algorithm ...................................................................... 43  

6 **Training the Neural Network** ............................................................................................. 45  

6.1 Neural Networks in Matlab ............................................................................................. 45  

6.2 Training Algorithm .......................................................................................................... 46  

6.3 Determining Size of Training Set .................................................................................... 47  

7 **Results** .................................................................................................................................. 49  

7.1 Methods for Evaluating the Network .............................................................................. 49  

7.2 Discussion of Evaluation Results ..................................................................................... 51  

7.3 Testing the Network on AMASS events ............................................................................. 60  

7.4 Situational Analysis .......................................................................................................... 62  

7.4.1 Landing/Takeoff Analysis ............................................................................................. 63  

7.4.2 Result Analysis by Runway ........................................................................................ 65  

8 **Generalizing the Process** .................................................................................................... 67  

8.1 Generalized Event Extraction ......................................................................................... 67  

8.1.1 Overview and Tool Design ......................................................................................... 67  

8.1.2 Tool Description .......................................................................................................... 68  

8.2 Generalized Network Training .......................................................................................... 70  

8.2.1 Overview and Tool Design ......................................................................................... 70  

8.2.2 Tool Description .......................................................................................................... 70
8.3  Generalized Results Analysis ................................................................. 71  
  8.3.1  Overview and Tool Design ............................................................. 71  
  8.3.2  Tool Description ................................................................. 72  
  8.4  GUI Form ......................................................................................... 73  
  8.4.1  Event Extraction GUI ............................................................... 74  
  8.4.2  Network Training GUI .............................................................. 76  
  8.4.3  Test Network GUI .................................................................... 77  

9  Conclusion ........................................................................................................... 79  

References .................................................................................................................... 81  

Appendix A: Extracted Event Counts ........................................................................... 84  
Appendix B: Trained Neural Networks ........................................................................ 86  
Appendix C: Data Extraction Toolset Matlab Code .................................................. 114  
Appendix D: Network Training Toolset Matlab Code .............................................. 123  
Appendix E: Network Testing Tools Matlab Code .................................................... 131  
Appendix F: Matlab GUI code .............................................................................. 142  
Appendix G: Graphs of Simulated Alarm Conditions ............................................ 158
List of Figures

Figure 1: Dataflow in AMASS System ......................................................... 9
Figure 2: Artificial Neuron ........................................................................ 11
Figure 3: Neural Network Layers ............................................................. 12
Figure 4: Position Prediction Neural Network ........................................... 29
Figure 5: Airport plot of ORD. Red Points bound the landing approach to runway 09L 33
Figure 6: Landing on Runway 27R. (DeltaT = 5s) ....................................... 34
Figure 7: Takeoff on Runway 09L. (DeltaT = 5s) ....................................... 35
Figure 8: AirportPlot graph of Extracted Takeoffs on Runway 09L. Events from runway 14L are incorrectly included. ........................................... 42
Figure 9: Event Extraction GUI .................................................................. 75
Figure 10: Network Training GUI .............................................................. 76
Figure 11: Test Network GUI ..................................................................... 78

List of Tables

Table 1: Extracted Landing Events ............................................................. 40
Table 2: Extracted Takeoff Events ............................................................. 40
Table 3: Extracted Landing Events(Average Speed over 100 fps) ............... 40
Table 4: Extracted Landing Events(Speed and Direction requirements) ...... 41
Table 5: statistical results from August 19th for the neural networks trained with all data from their respective days .................................................. 53
Table 6: Network Test Set Comparison ..................................................... 55
Table 7: Neural Network Prediction Statistics by Radar Type .................. 58
Table 8: Neural Network Prediction Statistics by Direction and Radar Type 59
Table 9: Landing/Takeoff Network comparison .......................................... 64
Table 10: Runway Network Analysis ........................................................ 66
1 Introduction

1.1 Overview of Work

This project focuses on implementing a neural network to improve predictions of future positions for airplanes operating near Chicago O Hare Airport. An airplane is considered to be operating near the airport if it is landing on, taking off from, or taxiing near any of the runways associated with the airport. Past projects, which are discussed in more detail below, have determined that future positional prediction can be improved through the use of two neural networks, one for airborne planes and one for ground planes.

This project attempts to use data mining techniques to extract appropriate training data for the neural networks. These data are used to optimally train the neural networks and demonstrate that a local maximum for prediction accuracy has been reached. Furthermore, data is presented on the accuracy of the predictive networks in predicting planes on specific runways and in specific scenarios. Ultimately, the techniques used to do this work for Chicago O Hare airport are collected in a toolset and presented in a generalized form so that they may be used to design optimal networks for other airports at a minimum of effort.

1.2 Goal of Work

The goal of this project is to generate a working neural network for Chicago O Hare airport as well as a well documented process and tools for generating similar neural networks for other airports. At the projects conclusion, the neural network will be fully implemented within the AMASS system and the accuracy of its predictions will be
fully documented. The specific goal of this documentation is to explain the development
of the neural network and to provide confidence in the performance of the neural
networks. The purpose of providing a well documented process with tools is to allow
future projects to easily extend the improvements provided by neural network predictions
to other airports.

2 Background

2.1 AMASS Overview

AMASS (Airport Movement Area Safety System) is an early-warning system
currently in use by numerous airports throughout the United States. It was created by
Northrop Grumman Corporation in response to increasing numbers of runway incursions
[1]. A runway incursion is defined by the US Federal Aviation Administration (FAA) as:

Any occurrence at an airport (with an operating control tower) involving
an aircraft, vehicle, person, or object on the ground that creates a
collision hazard or results in a loss of separation with an aircraft taking
off, intending to take off, landing, or intending to land. [2]

AMASS was created as a tool to be used as an add-on to the Airport Surface
Detection Equipment (ASDE-3). The ASDE-3 radar system tracks grounded airplanes
and other surface vehicles, providing air traffic controllers with a graphical display of all
vehicles on the airport surface. The Operator Display Unit (ODU) designed with the
ASDE-3 radar system may work independently of AMASS but is unable to give any
warnings for potential runway incursions without the assistance of AMASS. When
AMASS is used in conjunction with ASDE-3, the radar information from the ASDE-3
radar system is sent to the AMASS cabinet [3].

In addition to the surface positional data it receives from the ASDE-3 radar, the
AMASS cabinet also receives information about airborne planes from the Terminal
AutomFation Interface Unit (TAIU). The TAIU receives positional information about airborne planes from the Airport Surveillance Radar (ASR-9). It also receives data from the Automated Radar Terminal System (ARTS), which is used to tag the planes with their Ids [4].

![Dataflow in AMASS System](image)

**Figure 1: Dataflow in AMASS System**

The AMASS cabinet processes the ground and airborne data it receives from the ASDE-3 and ASR-9 radar systems, respectively, in order to determine potential runway incursions [3]. It displays the positional data it receives for all airplanes on the ODU. The AMASS cabinet generates alerts for planes that are close enough to result in a
runway incursion. It also uses a linear prediction model to predict future positions of planes and generate alerts for future predicted runway incursions. These alerts are displayed graphically on the ODU. Additionally, an aural message is generated in the control tower for all alerts [3]. It is important to note that AMASS does not have access to specific information about the planes it is tracking. Therefore, the prediction model does not take into account such factors as weight of the planes.

Figure 1 depicts the flow of information from the radar to the air traffic control tower. The upper left depicts the ASDE-3 radar. This sends data to the AMASS cabinet, shown in the upper middle. The lower left shows data flowing from the ASR-9 radar at the bottom left into the TAIU in the lower middle. The TAIU then feeds the data into the AMASS cabinet in the upper middle. The ASDE-3 and ASR-9 positional data is then fed along with alarm information into the ODU, depicted in the upper right. The ODU then displays this information for use by the control tower, shown in the lower right.

2.2 Neural Networks

Artificial Neural Networks (ANN) are mathematical models that attempt to emulate biological learning. The basic biological unit ANNs are based on is the neuron. The human brain is made up of hundreds of thousands of neurons. Together, these neurons account for our ability to learn from previous experiences. Neurons take in multiple electrochemical inputs; perform some function to combine these inputs; and output electrochemical outputs based on the results. Learning in biological neural networks involves modification of the synaptic connections between neurons [5].

Just as neurons are the basic units of the brain, artificial neurons are the basic units of ANNs. Artificial neurons are modeled as simplified neurons. Each artificial
neuron takes in multiple weighted inputs, sums these inputs, potentially puts them through a transfer function, and then generates outputs from the results. The biggest difference between artificial neurons and biological neurons lies in the input-output relationship. Artificial neurons typically have a linear relationship between inputs and outputs, while biological neurons typically have significantly more complex input-output relationship. Artificial neurons learn by modifying the weights related to each of its inputs [6].

![Artificial Neuron Diagram](image)

Figure 2: Artificial Neuron

Connections between biological neurons are numerous and complex. For ANNs, a simpler model for connecting neurons is required. ANNs give some structure to connections by placing nodes in layers. All of a neuron’s inputs either are inputs to the network or come from the layer immediately before it. All of its outputs either go to the layer immediately after the neuron or are outputs from the network [6]. All biological neurons can be grouped into three categories: neurons that take input from the body, neurons that send output signals to the body, and neurons that connect other neurons [5].
In artificial neurons, these different groups are modeled as the input layer, the output layer, and the hidden layers, respectively. Data entered into the system is first processed by neurons in the input layer. The outputs of the input layer neurons become the inputs to the next layer. The outputs of the neurons in this layer then become the inputs to the next layer. This process continues until the output layer is reached. The output layer then generates the outputs of the overall network.

![Neural Network Layers](image)

**Figure 3: Neural Network Layers**

The number of hidden layers is a feature specified by the designer of the neural network. Increasing the number of hidden layers generally increases both the complexity and the accuracy of the neural network. It is important to be careful not to add too many hidden layers to the neural network, however. If there are too many hidden layers, it is possible for the neural network to overfit to the training data [7]. Overfitting the data means that the network will become too attuned to specifics of the training data, resulting in it being ineffective on other data sets.

ANNs are being used in an increasing number of applications. They are effective at pattern matching, as well as classification problems. They are also effective on problems that involve complex systems or imperfect data, as they learn general
conclusions about large sets of data as opposed to generating a precise formula. Airport radar provides imperfect positional data for airplanes. Additionally, the motion of a plane approaching a runway can be complex and is affected by wind conditions of the airport, making neural networks a good solution for future position prediction.

2.3 Prior Work on Neural Networks in AMASS

A great deal of prior work has been done by previous members of the MIT project team on integrating neural networks into AMASS. This prior work laid the groundwork for this thesis project and is responsible for determining the type of neural network that would be most efficient within the AMASS system. Initial work was focused on directly predicting alarm situations using neural networks. This prediction method was discontinued, for reasons discussed below, in favor of using neural networks to predict future position.

2.3.1 Neural Networks for Generating Alarms

The initial idea on how to improve AMASS using neural networks involved designing a neural network that directly predicted alarm scenarios [8]. In other words, the network was trained to take inputs about the positions, headings, and velocities of planes and output a binary value which indicated whether or not the future positions of these planes generated an alarm condition. Multiple networks for modeling alarm conditions were trained. Both single-layer perceptron(SLP) and multi-layer perceptron(MLP) networks were trained to determine what type of network could best predict alarm conditions.

One problem with modeling alarm conditions is that there are numerous scenarios that can cause incursion events. However, it is possible to classify all incursion events
into one of a few categories. In order to be able to analyze alarms more efficiently, five distinct types of alarm classifications were defined. These scenarios (listed below) were defined by people working on neural networks and do not necessarily have a direct correlation to scenarios defined within the AMASS system.

- Lander Behind Lander (LL): This scenario occurs when a landing plane is still on the runway and is being approached by another plane landing on the same runway.
- Lander Behind Departure (LD): This scenario occurs when a plane is landing on a runway behind a plane that is taking off from the same runway.
- Intersecting Runways: This scenario occurs when two planes are approaching each other on different runways that physically cross each other.
- Land and Hold Short Operation (LAHSO): These scenarios occur when planes cross over an imaginary land that they are supposed to stop before.
- Taxi/Runway Intersections: These scenarios occur when two planes are approaching each other at a runway and taxiway intersection.

Since the attributes that cause an alarm vary for these different scenarios, distinct neural networks were created for each of these alarm conditions. The LL and LD scenarios were considered to be the most common ones. Therefore the initial networks were only tested on these scenarios. SLP neural networks trained for the LD and LL scenarios showed poor results. Testing additional neural networks demonstrated that the MLP neural network with one hidden layer performed well. All subsequent use of neural networks is based on the latter model.

2.3.2 Neural Networks that predict separation distance

Neural networks trained to predict performed very well on the available data. However, a problem arose due to the limited number of alarm scenarios available. The
small number of alarm scenarios available made it difficult to determine if the network was learning the true characteristics that lead to an alarm. There were insufficient alarm examples to properly test and validate the network's ability to correctly detect an alarm scenario. This led to the need for a new neural network that could be evaluated more thoroughly with the available data. For this reason, neural networks were developed to predict separation distance between two planes [8].

If the separation distance between the planes is predicted to dip below an acceptable level, then an alarm is triggered indirectly. Because this model predicts separation distance and not alarms, its performance can be tested using all the events available to us. Whether it throws an alarm or not is a consequence of the prediction values, not the primary performance metric.

The decision to predict separation distance instead of alarms was a key design decision; there were others too. Primary among these was the decision of how to calculate separation distance. The neural network could be designed to predict separation distance directly from two planes, predict future positions of the planes, or predict future velocities of the planes. The latter two options would require algorithms to calculate separation distance from the neural network outputs. A network that predicted separation distance directly required twice as many inputs and did not offer adequate improvement in accuracy over the other options. Further analysis led to the conclusion that taking velocity information as inputs and returning predicted future velocities would lead to more accurate results than the other options [9]. This also fits with logic, as different runways can have drastically different absolute positions, whereas the speed of the planes should change in a similar manner regardless of the position or size of the plane's
runway. At this time, it was also resolved by the design team that the same network would be used to determine both x and y velocities.

One further complexity arose during the design of the neural network. While the new network performed decently, it did not perform nearly as well as expected. It was initially surmised that the performance of the network could be improved by splitting the network into a low-speed neural network and a high-speed neural network. AMASS could be slightly modified to determine which network should be used based on the current speed of the plane. This resulted in an improvement in the network's performance, but the network still did not perform up to expectations. Using exploratory data mining tools, it was determined that airplane spatial and temporal behavior could be classified in two clusters depending on the radar system tracking the airplane [9]. As explained previously, surface planes are monitored with a different radar system than airborne planes. These radar systems have different attributes and idiosyncrasies. In order to take advantage of this, the final neural network also consists of two distinct neural networks. Instead of being split into a high-speed and a low-speed network, the final network is split into an ASR (airborne radar) and ASDE (ground radar) network. The ASR network is used for generating predictions when the plane is being monitored by the airborne radar and the ASDE network is used for generating predictions when the plane is being monitored by the ground network.

3 Related Work

Over the past decade, neural networks have been applied to a wide range of applications. A number of these applications are relevant to the development of neural networks for airplane position prediction. A few other neural network applications relate
to airplanes and aerodynamics. Also of interest are a number of neural network applications that deal with time series prediction.

3.1 Runway Incursion Prevention Techniques

Significant increases in both runway incursions and runway collisions in recent years have made reducing both the severity and frequency of runway incursions a hot topic for the FAA. As many tools and processes, AMASS among them, have been developed in recent years. These tools seek to increase the use of technology in airport movement as well as to improve methods employed to control airport movement.

In 2001, the FAA Subcommittee on Aviation held a hearing focused on Runway Incursions, Focusing on the Technology to Prevent Collisions [10]. The subcommittee reviewed the use of technology to prevent runway incursions and created a report detailing its findings. This report details the situations where the current AMASS system is and is not useful. The initial AMASS implementation was intended to monitor in the airport vicinity and identify any objects as planes taking off. Due to the ASDE-3 radar frequently tracking objects such as large puddles and AMASS's difficulty distinguishing departing planes from fast moving taxiing aircraft and other vehicles, the functionality of AMASS had to be limited to prevent excessive false alarms. The difficulty in determining which objects were departing planes was related to the definition of departure speed. Departure speed was defined by AMASS as 80 fps, a value which both AMASS and this thesis found insufficient to distinguish departing planes from other fast moving vehicles [10].

The AMASS system was modified to monitor only the runway group defined as active and conditions were implemented for monitoring objects on the active runways.
While this reduced the number of false alarms generated by AMASS to a reasonable level, it also limited the types of collisions AMASS could predict. The current implementation of AMASS is unable to predict collisions significantly ahead of time between a plane landing on or departing from a runway and a plane on another runway crossing its path. AMASS is also no longer able to monitor planes on closed runways. This renders it unable to detect if a plane is mistakenly taking off from a closed runway possibly leading to a collision with other planes that may be taxiing on that runway for a variety of reasons. The report did indicate that the average lead time of AMASS prior to collisions is 22 seconds; this is sufficient time for air traffic controllers to avert the collision [10]. Work currently being done by Anuja Doshi analyzing the neural network's ability to predict alarm events indicates that the neural network improves the lead time prior to collisions by an average of 2-3 seconds. Research is also being done into the neural network's ability to allow AMASS to predict further into the future without significantly increasing the number of false alarms [10].

In addition to reporting on the drawbacks and successes of AMASS's current implementation, the report details other technologies for preventing runway incursions. An improved version of the ASDE radar, ASDE-X is currently being developed. The ASDE-X radar receives information from multiple sources to determine aircraft positioning. It monitors planes with surface radar, transmitters and receivers that communicate with transponders on the aircraft and automatic dependent surveillance broadcast (ASD-B) which tracks aircraft through the Global Positioning System. The combination of these sources should result in a significantly more accurate radar system.
The remaining technologies discussed by the report focus on improving pilot information. Runway Status Lights (RWSL) are intended to keep pilots informed about the status of runways. They indicate when a runway is unsafe for use or crossing and are automatically controlled by surface radars. In-Cockpit Moving Map Displays are designed to replace paper maps. The display shows the airplane’s position on the airport map at all times and is intended to increase the pilot’s awareness of his own position. Finally, anti-blocking radios are intended to prevent pilots from losing communication with the control tower due to multiple pilots mistakenly using the same frequency [10].

A section of NASA, the NASA Aviation Safety Program, also works on making airplanes safer by decreasing runway incursions. This program has devised a Runway Incursion Prevention System (RIPS) to this end. While AMASS is designed to provide warnings to air traffic controllers, RIPS attempts to provide the pilots and airplane crew with warning indicating impending incursion events. RIPS is really a combination of a number of technologies. Under RIPS, each system is installed with a Heads-up Display (HUD). This display provides guidance to the pilot in the form of deceleration information, distance from hold short position, runway edge guides, and other useful guidance and information. The RIPS system utilizes an Electronic Moving Map (EMM). The EMM provides an image of the airport layout along with the plane’s position. It also displays the positions of other aircraft and vehicles and highlights dangerous areas. The RIPS system also the ability to generate alerts for the aircraft’s crew if the aircraft deviates from its assigned route or crosses a hold short line. In addition to its other attributes, the RIPS system has the ability to generate aural and visual alarms. Similar to AMASS, these alarms are generated by algorithms designed to process positional plane
information and predict future incursion events before they occur. The report suggests a couple of different algorithms for predicting incursions. The report also mentions that the report also has the ability to receive alarms from a ground-based system, such as AMASS. It recommends against this option, however, arguing that transmission delays and other factors cause the plane to receive the alerts slower than if they were generated on-board [11].

In addition to the AMASS program, the FAA is developing another system that monitors airport traffic using inductive loops. Inductive loops have been in use for monitoring highway traffic for decades. They operate using sensors embedded in the runway surface [12]. These sensors monitor traffic and could be developed to attempt to prevent runway collisions using signal processing neural networks. This technology’s capabilities are relatively limited compared to the ASDE radar system. However, planes can only be monitored in areas where sensors are placed. This renders the system ineffective for large airports and precludes monitoring of airborne data. Being unable to monitor airborne data means incursion events involving landing planes can not be predicted as early as with an ASDE system. Nonetheless, inductive loop systems are seen as a potentially viable alternative to ASDE systems smaller airports or airports that only have problems on certain localized runways [13].

3.2 Aircraft Related Neural Networks

One common use of neural networks is the tracking of airborne objects. There are a number of technical papers that deal with using neural networks in conjunction with radar tracking systems. One paper, published in the August 1998 edition of the International Journal of the Computer, the Internet and Management, outlines a back propagation
neural network designed to track airborne aircraft and predict their position one second in the future [14]. The neural network described in this paper is different from the AMASS neural network in a few key ways. The network in the paper makes predictions in three dimensions as opposed to two but predicts only one second into the future as opposed as 20 seconds which is the current standard used by AMASS. As with the AMASS neural network, the authors of the tracking network come to the conclusion that past positional differences serve as better neural network inputs and outputs than absolute positions. The high rate of change for absolute positions compared to that of positional difference is cited as the reason for this choice.

Another area where neural networks are being applied to airplane motion is for in-flight safety controllers. NASA is currently testing neural network software with the ability to determine if some error or malfunction is affecting an airplane’s flight. The network is installed on the plane along with numerous sensors that records factors such as speed and force on different sections of the plane. The trained neural network receives these readings and uses them to determine if the plane is flying according to a pre-defined flight pattern. If the network determines the plane is not flying according to its flight pattern, the network automatically adjusts the flight controls to compensate for the plane’s deviation from those flight plans [15].

The use of neural networks has even been expanded to the design and testing of aircraft. Charles Jorgenson of the Ames Research Center describes a method of using neural networks to help model the flow properties of an object. In the example of an airplane, the network would be modeling the aerodynamics characteristics of the aircraft. The method proposes training the neural network during wind tunnel tests. The trained
network could then be used to predict the characteristics of the plane for untested combinations of input variables. Jorgenson also proposes using the network to quickly find input characteristics that provide a desired output thus reducing expensive wind tunnel testing time [16].

3.3 Time Series Applications of Neural Networks

Time series applications of neural networks have been becoming increasingly prominent. Numerous papers have been published recently both on techniques for developing time-series neural networks and on specific time-series applications of neural networks. Neural networks have been developed to predict variables ranging from sales of a product to the market interest rate. Frank Thiesing and Oliver Vornburger developed a neural network to predict sales of items in a supermarket based on the volumes of past sales [17]. Based on the theory that sales of an item in the supermarket are effected not only by its own past sales but also by the past sales of other items, the network uses past weekly sales of all items as input and returns, and predicts sales for the upcoming week.

Additional work in using backpropagation neural networks for time-series prediction includes a paper on using neural networks to predict future performance of stocks on the Shanghai Stock Exchange [18]. This paper investigates the use of neural networks with a couple of different learning algorithms and initial weights in predicting future performance of stocks. Another article, published in Machine Learning, describes the use of neural networks in developing non-linear controllers for robotics [19].

It concludes that the neural network makes better predictions than an optimum autoregressive model and that the neural network is effective at noise filtration [20].

A paper from January 2002 investigates using neural networks to predict financial time series. More specifically, the paper looks at the use of negatively correlated neural network ensembles to make predictions. Neural network ensembles are a group of neural networks that are used in conjunction to solve a problem. The fact that they are negatively correlated means that each network's error function is penalized for giving outputs similar to other neural networks. This serves to develop networks that give distinct responses. The paper ultimately concludes that negatively correlated neural network ensembles forecast financial time series slightly more effectively than other standard neural network approaches [21].

Further papers holding relevance to this thesis pertain to time series prediction in general. As opposed to presenting specific applications of time series forecasting, these papers focus on general methods for making time series predictions. The reason there has been increasing interest in time series neural networks recently is their easy adaptability. Unlike most other time series models, neural networks do not need to be told the characteristics of the system. Given sufficient past data from the system, the neural network can train itself to make predictions. This makes neural networks extremely adaptable and useful tools for system identification. Papers published on generic time series neural network approach include a paper on extending the group method of data handling (GMDH) using polynomial harmonics [22]. GMDH is an algorithm that used to model neural networks. It is a beneficial method because the
structure of the network is determined computationally based on the data. This helps to remove any biases the designer may have for certain network designs [23].

Some other papers examine new models of neural networks for time series forecasting. One paper, published in the *Journal of Time Series Analysis*, presents a neural network with dynamic connections. The paper points out that many connections between nodes turn out to be unnecessary. It introduces the concept of inhibitor arcs, which are modeled after the brain's inhibitor synapses to account for this fact. The paper also builds excitatory arcs, modeled after excitatory synapses, into the neural network to model variable strengths at different times. These methods allow for the a more dynamic and easily updateable network [24].

Another paper attempts to analyze a new neural method's usefulness in predicting currency exchange rates. Typically, currency exchange rates have been predicted by multilayer perceptron or recurrent neural networks. This paper introduces a relatively new neural network model, KIII. KIII is a member of the K family of neural networks. The K family of neural networks was developed relatively recently. Their foundation in actual biological systems is stronger than that of standard neural networks. Each member of the K family of neural networks represents a different part of the nervous system. The KIII network used in this paper is modeled after the sensory cortex [25].
4 Overview of Work

4.1 Generating an Optimized Neural Network for Chicago O'hare Airport

Other members of the project team have created one neural network that utilized data from the ASR radar and another neural network that relies on ASDE radar; this was determined to be the best way to make accurate predictions of future separation distances. This thesis focuses on the implementation of such an approach for Chicago O'Hare airport. The development process involves: extracting relevant data from large log files; dividing these data into training, validation and test sets; training the neural network with the training and validation sets; and verifying the network's performance with the test set. The size of the data set is continuously increased until the neural network is optimized.

The raw data needed to train the neural network is contained in massive log files generated by AMASS. These log files are typically a day long. A script, written by Dr. Rafael Palacios, extracts relevant information from these files and places it into a 3-dimensional Matlab matrix. The first dimension represents the time in approximate seconds of the data. The data are sampled close to once a second but the actual time divisions are not exact due to the nature of the radar. Air traffic controllers tag planes with one of 256 identification numbers. The identifications are eventually re-used but not too close to cause confusion regarding two planes with the same identification number. The second dimension of the matrix represents this identification number. Using the first two dimensions of the 3-D matrix, it is possible to choose a specific plane at a specific time. The third dimension of the matrix is five elements long, with each element representing a specific piece of information about the specified plane and time. The first element indicates which type of radar is currently monitoring the plane. The second and
third elements are the $x$ and $y$ positions of the plane, respectively. The fourth and fifth elements are the $x$ and $y$ velocities of the plane.

The 3-D matrix discussed above provides all the information necessary to train the neural network. Unfortunately it needs to be processed further before the data are in a form that is useful for training the neural network. The neural network is intended to accurately predict future velocities of planes landing and taking off. The log file contains large amounts of null data where certain plane identifications are not being used or have not been specified. It also contains large portions of data on slow moving objects, which are not always even planes. At times, especially during rain storms, objects, such as puddles, are detected by the radar and registered with the AMASS system. It is necessary to break this log file into individual events in order to remove such extraneous data.

Splitting each day’s log files into its relevant takeoff and landing events provides the data needed for training the neural network. From here, it is simple to randomly divide these events between training, validation and test sets. The next step involves training the neural network. This is done through the neural network tools that exist within Matlab. This thesis focuses more on training the network with increasing amount of data until its prediction accuracy has reached a local minimum.

4.2 Verifying Accuracy of Optimized Neural Network

The second stage of this endeavor involves gathering data on the performance of the optimized neural network. The primary methods for comparing the performance of different training runs of the network will be the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). The values of these error measures will be compared to errors for the linear prediction model, as well. The goal of this error analysis is to
determine the degree of improvement between the linear prediction model and the neural network based model. Additionally, prediction errors will be calculated for different days and runways to determine the variation in the accuracy of the neural networks. These results will be analyzed to determine any potential recommendations on the most efficient use of the neural networks developed. It will also allow for a thorough analysis of the predictions in a variety of scenarios.

4.3 Generalizing Optimization Process

Once a fully developed, rigorously tested neural network is in place for Chicago O Hare airport, the process used to design that neural network will be generalized to a step-by-step process that can be applied to any airport. While some degree of human interaction and design will always be required to implement the neural network for a new airport, the goal is to automate as much of the process as possible and provide a well-documented process that can quickly and easily be applied by anyone with a basic understanding of the techniques involved. In other words, anyone with a basic user-level understanding of Matlab should be able to recreate the entire network optimization process with only the toolset and instructions provided by this project.

A major part of generalizing the optimization process lies in building a Matlab toolset that contains generic functions to perform the data mining, training, and testing work. These generalized functions will be developed from the specific Matlab scripts that were used to create an optimized neural network for Chicago O Hare airport. These functions will require only simple inputs, such as a log file or runway coordinates. Any data generated by these tools that is needed by other tools will be saved to a file specified
by the user. Similarly, tools that need previously calculated data would load such data from files specified by the user.

5 Training Set

The first step in building the optimized neural network is to generate a set of relevant events. This set is then be divided into training, validation, and test data. These different groups of data are used to train an optimized neural network and verify its accuracy. In order to generate a test set, it is necessary to define what an event of the test set consists of.

5.1 Training Set Events

The neural network, which was determined to produce the most accurate results, takes in instantaneous velocities over the last ten seconds as inputs; it outputs the distance the airplane will move up to thirty seconds in the future. This network is displayed in Figure 4. It requires a list of related velocities at one second intervals. Each list of velocities needs to be at least forty seconds long in order to provide maximum utility while training the network. The list of velocities is built from a matrix of positional data prior to the process of training the network. A matrix of related positional data defines a single training set event. The events are generated from the 3D matrix obtained from the AMASS log file. This 3D matrix was described in section 4.1.

This raises the question as to what a matrix of positional data should consist of. A simple idea would be to take all the positional data for each plane id over a set period of time. If a quarter of a day was the time length used, then each quarter day period would generate 256 distinct positional data sets, one for each plane id. There are a few problems with this method, however. The first problem arises from the fact that plane ids
are re-used. Therefore, in the specified time period, it is possible for the same id to be used for two or more distinct planes. The positional data for these two planes would be unrelated and should not be placed in the same set. The bigger challenge is that there are rarely more than 40 plane ids being used at any one time. This results in a frequent situation where there is no positional data for a particular plane id. Intervals of time with no data should not be included in the training set.

Another idea, which eliminates the two problems of just using plane ids for the positional data sets, is to consider any group of valid data for a given plane id as an event. If a plane id has \( n \) groups of valid data separated by periods without valid data, then it will generate \( n \) events. In other words, an event would be the total range of positional data from the radar readings for a single plane. The problem with defining an event in this way arises from the fact that the radar often picks up and tracks objects other than planes. Things such as trucks and other large objects are often detected and tracked by
the AMASS system. Taking every group of positional data as an event would include these situations. This would result in a training set that included events that are not characteristic of airplane motion and would be a detriment to the training of the network. Additionally, this method would include taxiing planes. Taxiing planes move either constantly in the same direction, thus making their velocity simple to predict, or can change velocity quickly since they are at low speeds, making it difficult to predict. Either way, slow moving planes are not nearly as useful for training the neural network as high-speed planes because we are especially interested in improving accuracy for high speed targets. This leads to the definition of an event used.

The final definition of an event that was chosen leaves only sets of positional data that are useful for training the network. An event is defined as any sequence of positional data that all relate to the landing or takeoff of a single plane. Since it is not possible for trucks or other runway obstructions tracked by AMASS to exhibit the characteristics of a plane taking off or landing, this definition results only in events that are useful in the training process. These are also the scenarios that result in the largest prediction errors and where the use of neural networks could most improve in prediction over a linear model. Linear models predict taxiing planes very well since they move at slow, near constant speeds. By training the neural network on high-speed planes, some accuracy will be lost on the prediction of future velocities for taxiing planes. On the other hand, prediction accuracy is greatly increased for high-speed planes. This is trade off is a beneficial one since high speed planes are significantly more likely to cause runway incursions than taxiing planes.
5.2 Extraction of Events by Runway

Now that training events are defined, the problem arises of how to separate these events from other data contained in the 3D matrix. Since planes involved in both take-offs and landings attain much higher speeds than any other objects monitored by AMASS, one way to extract these events would be to find all planes traveling above a certain velocity and take all the valid positional data before and after this time.

This method would theoretically give us all the landing and takeoff events and only those events. The downside is that it groups all takeoffs and landings on all the runways together. This makes it difficult to verify that the events were extracted properly. The difficulty involved in verifying this fact presents a formidable challenge since having invalid data could cause the neural network to be overtrained for a particular runway or situation. This could lead to a network that is unable to generalize the model for other runways and situations. The other drawback to extracting events by speed alone is that it eliminates the possibility of analyzing the performance of the network in various situations. In order to eliminate these issues, events were extracted separately by runway used and whether they involved takeoffs or landings.

The extraction of events by runway and separately for takeoffs and landings provides the ability to perform situational performance analysis on the neural network. In addition to gathering performance statistics on the overall performance of the neural network, performance splits for takeoffs and landings as well as splits by runway are possible. This provides the potential for research into the possible use of multiple networks for different situations.
With an idea of how events should be extracted, all that remains is developing a method to actually extract the desired events. There are two basic difficulties in developing this methodology. It is necessary to develop a method that can find out which planes are operating on a given runway and whether they are taking off or landing.

The 3D matrix contains data on the positions of the different planes around the airport. There is also a base image of Chicago O'Hare airport (Shown below in figure 5). A matlab script entitled AirportPlot.m written by Dr. Rafael Palacios takes a set of x and y coordinates given in the coordinates used by AMASS and converts them onto the airport image. This script can be used to determine coordinates to place a box around a section of a runway. Any planes that fall inside this box are assumed to be on the runway.
Figure 5: Airport plot of ORD. Red Points bound the landing approach to runway 09L.
Landing planes are the only objects tracked by ASR. This implies that they are the only planes that will ever be tracked outside the runway area of the airport. Therefore, the box to catch landings can be placed around the approach to a runway to catch all landing events. While this method works for most landings, there are some runways where the approach path overlaps with other runways. Additionally, takeoffs are only tracked by AMASS until they leave the ground. The takeoff zone often overlaps with takeoffs and landings from other runways as well as landings on the same runway but in the opposite direction.

Figure 6: Landing on Runway 27R. (DeltaT = 5s)
Figures 6 and 7 above show planes on runway 22L landing and taking off, respectively. The figures demonstrate a major difference between the two events. The red dots pictured are obtained by sampling the plane's position at regular time intervals (DeltaT = 1s). The implication being that when the dots are spaced far apart, the plane is moving at high speeds, whereas when the dots are clustered closely together the plane is taxiing or moving at low speeds. The landing plane is moving at high speeds at the start of the runway and taxiing at the end. The plane taking off acts conversely. It taxis at the start of the runway and enters high speeds toward the end. Checking for high speeds at the end of the runway, results in takeoff events while checking for high speeds at the start of the runway, returns landing events.

Determining at what end of a runway a plane moves at high speed provides a method for determining if an event is a takeoff or a landing event given a specific
runway. This still leaves a pair of problems. Firstly, there are runways that overlap. At these points, a test based solely on speed will give events for both runways. Also, many runways are used in both directions. For example, Runway 22L is the same runway as runway 04R but in the opposite direction. If wind conditions are different, the runway could be used in the other direction. Therefore, a test that checks for high speed planes at the end of 22L will pick up takeoffs on 22L and landings on 04R. Changing the check for speed to a check for a velocity solves both of these problems. The difference being that a velocity check takes speed and direction into account. Even with a built in error to account for the imperfection of the radar, checking for direction and magnitude eliminates side runways as well as planes moving in the opposite direction. This is because overlapping runways are never closer than 10 degrees to each other.

With the methodology for testing whether a plane is on the proper runway and is taking off or landing, one can build an algorithm to extract events. The algorithm looks through the 3D matrix and determines which planes fall inside a specified box and have the proper velocity. Then for every plane that meets the specified conditions, the set of valid data constitutes an event for that runway and landing/takeoff. The extraction process must then be repeated for a landing and a takeoff on every runway, changing only the positional and velocity requirements.

5.3 Validation of Extraction

Once the extraction algorithms have been applied, it is necessary to verify that they produced the expected results. Two types of tools were applied to verify this fact. There were visual tools and counting tools. The visual tools represented the events in a visual form so that the correctness could be verified by human observation. The counting
tools count the total number of events extracted on each runway for takeoffs and landings.

5.3.1 Visual Validation Tools

The visual validation tools apply the set of events to the background picture of Chicago O Hare Airport. The first of these tools is the same tool used to determine coordinates for the box placed around the runway (AirportPlot). Since this script plots a set of coordinates on the airport map and each event is a list of coordinates with the type of radar used, passing the event list to the script will cause all the points in all the events to be plotted on the airport map. Because of the large number of events plotted when the script is used in this way, this tool is not very useful for distinguishing between takeoffs and landings.

The AirportPlot tool is better used to determine if events from any runways other than the desired runway were extracted. If the event extraction ran perfectly, then the only areas that should have any data points are the desired runway, the approach to the runway, and taxiways that come directly off the runway. If there is a string of data points down any other runway, then events in that runway were incorrectly included.

Dr. Rafael Palacios also developed the other visual validation tool. This tool uses the same airport map as AirportPlot and is entitled AirportPlot4. The tool operates in a similarly fashion to AirportPlot in that it plots a set of coordinates. The difference is that instead of plotting all events simultaneously, it only plots one point at a time. This makes it useful to simulate the motion of the planes in the events. The tool also allows one to adapt the speed at which the points are changed in points per second. At a setting of 90 pps, the points are plotted slowly enough so that it is possible to visually distinguish
takeoff events from landing events but quickly enough that events take only a couple of seconds each. Even at only a couple of seconds per event, it can take a long time to go through some of the larger sets of runway. Therefore, this tool is more useful for checking a small number of events to make sure there is nothing significantly wrong. The previously discussed visual tool as well as the counting tool described below are more useful for checking large sets of data.

5.3.2 Counting Validation Tools

The subset of events of interest in a list of extracted events is separated by matrix rows consisting of all zeros. This provides a simple method to count the number of events extracted. Simply count the total number of zero rows. The count provides a rough estimate about whether the total number of landings and takeoffs in a day are approximately correct.

Another simple check, based on count, involves comparing the number of landings and takeoffs in a day. Since the total number of planes at the airport should remain close to constant, the total number of landings in a day should be close to the total number of takeoffs in a day. These tools are useful for determining whether the net extraction results are reasonable. If there are significantly more takeoffs than landings, for example, it is likely that either some of the takeoffs are counted twice or that some of the landing events were not extracted.

The analysis of the initial counts (discussed in further detail in section 5.3.3), indicated that there was a strong probability that there were either too many landings extracted or not enough takeoffs had been extracted. If there is a problem with takeoffs, it is because inadequate data were collected. Consequently, there is no sure way to check
if not enough takeoffs were extracted aside from implementing a new extraction algorithm. The opposite is true if there are too many landings extracted. This scenario can be easily detected by further analysis of the data.

The initial set of algorithms used to extract events did not take direction into account; they defined high-speed to be any speed over 80 feet per second (fps). Eighty fps is equivalent to about 54.5 miles per hour (mph). Keeping in mind that some fast moving motor vehicles may be moving faster than 55 mph, a modified counting script that only counted vehicles moving faster than 100 fps on average over a period 10 seconds was developed. Next, another modified counting script that took into account both the speed requirement and an additional requirement on the direction of the plane was developed.

5.3.3 Results of Validation Tools

The counting validation tools take less interaction than the visual validation tools. Once large amounts of data were extracted, the counting tools were applied on a portion of them to check the accuracy. The counting algorithm counted data for a single day. Due to processing power constraints, extractions were performed on periods of one quarter of a day at a time. The counting algorithm was run on the first three quarters of August 18, 2001. The results are displayed below in Tables 1 and 2. The columns of the table are runways specified by their runway number and R or L representing left or right, respectively. The rows of the table represented approximately six hour periods of the day. The first section of the day, specified by the tag a, starts at 00:00:00 MSE.
The counts on the extractions from the 18 hours chosen demonstrate a clear discrepancy between the number of landings and takeoffs extracted. There were 842 landing events extracted compared to only 609 takeoffs. There is still a possibility that the extractions worked properly. It is likely that at certain times during a day there are more planes coming into an airport than out of it. Despite the fact that the discrepancy seems too large to be entirely corrected by the last six hours of the day, the possibility exists. In order to determine if too many landings were extracted, the modified extraction counts were applied to the same set of landing events.
The declining event numbers in the tables above indicate that the extraction algorithm extracted too many landing events. The fact that increasing the speed requirement eliminates a number of events indicates that 80 fps is not an appropriate speed requirement. Further analysis into appropriate speed requirements revealed that 120 fps eliminates all slower moving objects without eliminating any valid landing events.

The further decrease in the number of events when an additional direction requirement was added to the count indicates that some events are being counted on multiple runways. This indicates the need for stricter direction requirements.

Many of the scripts used to extract the landing events tested only for speed and did not test for direction. The scripts were changed to check that the y velocity is greater than 120 times the cosine of the runway angle and the x velocity is greater than 120 times the sine of the runway angle.

The new scripts resulted in landing event counts identical to the counts in Table 4. This led to a total of 657 landings and 609 takeoffs. The difference between these two numbers seems significantly more reasonable than it was before. Now that the counts appear reasonable, the next step is to perform visual verification on some of the runways to determine if there are any other problems with the training set. The AirportPlot tool indicates that some event lists are still extracting some events from the wrong runways. This is most likely due to the test that is being used for direction. The test is not a strict

<table>
<thead>
<tr>
<th></th>
<th>09R</th>
<th>14L</th>
<th>14R</th>
<th>22R</th>
<th>22L</th>
<th>27R</th>
<th>27L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>08/18/01-a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>2</td>
<td>110</td>
<td>109</td>
<td>253</td>
</tr>
<tr>
<td>08/18/01-b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>08/18/01-c</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>162</td>
<td>47</td>
<td>30</td>
<td>117</td>
<td>404</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>194</td>
<td>49</td>
<td>140</td>
<td>226</td>
<td>657</td>
</tr>
</tbody>
</table>

Table 4: Extracted Landing Events (Speed and Direction requirements)
direction test. It merely assures that the plane is moving fast enough in the x or y directions. This is sufficient in most cases. The reason for this is that in order for a plane to be moving fast enough in the direction of the target runway, the portion of the plane's velocity that can be projected onto the target runway must be at least 120 fps. This implies that if the plane's runway is 30 degrees offset from the target runway, the plane's speed must be at least 138 fps when it meets the positional requirements.

Figure 8: AirportPlot graph of Extracted Takeoffs on Runway 09L. Events from runway 14L are incorrectly included.

Since there are still events from invalid runways being extracted (As shown in figure 8), it is necessary to make the directional test stricter. This task was accomplished by separating the test for speed and directional. The algorithm changed to test for absolute value of the velocity greater than 120 and to test for the direction of the plane to be within 5 degrees of the desired direction. This value was chosen because all airport runways must differ by at least 10 degrees. The new algorithm eliminated planes on incorrect runways.
The AirportPlot4 test verified that the extraction algorithm was extracting the events correctly. It did expose one additional problem with the extracted events, however. There are a number of events which meet the algorithm's criteria perfectly but for some reason do not make up entire events. For example, some runways have objects which appear to be approaching the runway for a landing but AMASS stops tracking once they hit the runway itself. It is difficult to tell why things such as this happen. It is possible that the plane is not properly passed between the two radars or that AMASS is accidentally tracking objects other than planes. Without knowing why these events occur, they should not be included in the training set. The primary difference between these events and actual events are the length of the valid data. The solution is to only allow events that contain a certain number of data points to be used as training events. It is difficult to tell before an event is extracted how many data points it will contain. It is simpler to extract all events that meet the requirements and then apply a filter to remove all the events that contain less than 40 data points. This final modification results in an algorithm that is able to extract the desired training set data. The final numbers of events extracted can be found in Appendix A.

5.4 Description of Final Extraction Algorithm

The event extraction algorithm takes three inputs. The first input is a 3D matrix built from the log files. The second input is a 2x2 position matrix where the first column contains x coordinates, the second column contains y-coordinates, the first row contains lower bounds, and the second row contains upper bounds. The final input to the extract event script is the angle of the runway in degrees oriented with zero degree pointing up in the y direction and ninety degrees pointing right in the x direction.
The script uses these inputs to build a 2-dimensional matrix. The first row of the matrix represents time in seconds and the second dimension of the matrix represents the plane id. This matrix maps to the first two dimensions of the 3D matrix. For each plane id and second in the 3D matrix, the script applies a Boolean test to the values contained in the third dimension of the 3D matrix. The 2-dimensional matrix stores the result of this test. The test returns true if the plane's position falls within the bounds given in the inputted position matrix, is moving faster than 120 feet per second (fps), and is moving within 5 degrees of the inputted direction.

The script then scrolls through the 2-dimensional matrix. For each plane id, it scrolls through each second until it finds a true in that column. The script then calls a script written by Dr. Rafael Palacios called ExtractEvent. ExtractEvent takes the 3D matrix, a plane id, a record number (time index), the number of seconds in the future, and the number of second in the past. ExtractEvent returns a matrix containing x positions, y positions, and radar types which constitute a single event. The event starts the number of seconds specified before the time given and extends to the number of seconds specified after. This event is then added to the list of events already extracted. The number of events extracted into the past and future depends on whether the event is a takeoff or landing. The time into the past and future is selected to be large enough to guarantee the full takeoff or landing is extracted without being so large as to incur large amounts of excess data. Takeoffs read 60 seconds into the past and 10 into the future, whereas landings do not read into the past and read 99 seconds into the future. The script then jumps forward in time and continues the process from there.
The script has one last step once it has built the matrix containing all the events. In order to remove events that may be incomplete, the algorithm counts the length of the valid data in every event. It then removes all events that are shorter than 40 seconds in length from the extracted events matrix. After this extraction is complete, the algorithm outputs the final extracted events matrix.

6 Training the Neural Network

After the training set has been built, the next step in the process is to train the actual network. Because there are two individual neural networks, the training set data must be assigned to the appropriate neural network. Once this has been accomplished, the training set for each network needs to be split up into training, validation and test cases. The training and validation sets are then passed to the matlab neural network training algorithm. Once the networks are trained, the test set is used to calculate performance statistics on the newly trained network.

6.1 Neural Networks in Matlab

Matlab has many built in functions that are capable of generating a wide range of neural networks. Both neural networks used to predict future plane position are built using the matlab function newff, which creates a feed-forward back-propagation network. The network is trained with two layers and ten hidden nodes. Both layers are configured with the transfer function purelin which is a purely linear transfer function. Additionally, the networks are both trained with a matlab back-propagation network training function named trainrp. Trainrp uses the resilient backpropagation algorithm(RPROP) to update the weights of the neural network. Both trainrp and purelin are parameters passed the function newff. More information on them can be found in Matlab’s documentation.
6.2 Training Algorithm

The neural network training procedures were largely developed by work completed prior to the work described in this paper [8]. Therefore, they will be only described here briefly. The matlab scripts used to train the neural networks can be found in appendix D.

The first step in the training algorithm is to split the training set into the appropriate pieces. A script entitled buildTS26 handles the task of taking the events and turning them into valid inputs and outputs for the neural network. BuildTS26 takes a list of events as output. For each event in the list, it turns the positional information into a list of speeds at each second. It builds this list by subtracting the plane’s position at each time from the plane’s position one second in the future. This results in a list of one-second averaged velocities. These velocity lists serve as inputs to the neural network. Since the neural network outputs are the distances that the airplane will advance in the following 5, 10, 15, 20, 25, and 30 seconds, this list is shifted by 5, 10, 15, 20, 25 and 30 seconds to form the list of ideal outputs for the neural network, as well. The same neural network is used to predict both x and y velocities. Therefore, the process just described is run on the x and y data separately to form independent x and y input/output pairs. The final step of BuildTS26 divides the input/output pairs into input/output pairs for the ASDE and ASR networks. This division is based on the indicator variable which indicates which radar was being used when each section of data was obtained. The reason the division does not occur earlier in the process is that most landing events start with ASR data and switch to ASDE data. Therefore, the ASR predictions are often tested
against the future ASDE positions. Dividing the inputs and outputs earlier would render some of the ASR test cases invalid.

Once the training set has been successfully divided by x or y coordinates, neural network and input/output, it is randomly divided into training, test and validation sets. The random process divides the data into fifty percent training data; twenty-five percent validation data and twenty-five percent test data. With the data prepared, the last part of the training algorithm only needs to invoke the correct matlab procedures. The algorithm calls the function newff to build the networks. It uses the matlab train function to train the neural networks and generates statistical data on the performance of the trained data as an output. Finally, the script concludes by providing the neural networks and the statistical data on their performance as outputs.

6.3 Determining Size of Training Set

It is important that an appropriate number of events are used to train the neural networks. If too few events are used, the network will not be fully trained and will be unable to make predictions as accurately as possible and there is some danger of the network becoming overfit. In other words, by training the network with too few events from the short time period available, there is a risk that the network will become overly attuned to the specific characteristics of the data set. If the data set is very big, it may have a bad balance of events. For example, if the data set contains 80% taxi events, the neural network may be unable to predict takeoff events accurately because the incidence of these errors was negligible during the optimization process. In addition, the acceleration of a plane in the process of taking off is impacted by the condition of the runway. If it has been raining and the runways are slick, the plane will take longer to
accelerate fully. Aside from predictable factors, such as this one, there could be any number of condition related factors that effect the motion of the plane. It is impossible to avoid having the network learn to account for these effects but if the network is over-trained, the results will become heavily fit to trends that may not extend to all times and conditions. The idea is to train the network enough that it learns the general trends that affect a plane taking off or landing at Chicago O Hare without giving it so much data that it can only effectively predict planes in its training set.

To achieve the goal of training the network to an optimal level without, the network was trained with sets of data of increasing sizes. Each network is then analyzed to determine an optimal network. The primary statistic used to evaluate the network is the prediction error. The prediction error gives a measure for how accurately the network is predicting future position on a set of events. The calculation of prediction errors is discussed in further detail in section 7.1. The prediction error for each of the neural networks is calculated on the same test set. Using the same test set to evaluate all the networks allows the prediction errors to be compared without having to worry if one error was build on a test set with events that were more difficult to predict. Additionally, the test set consists of events that were not used as training events in any of the networks. If the training set contained events used for training in some networks, those networks would have an advantage in predicting the event over other networks not trained with it. For this reason, the test set must be kept entirely separate from any of the training events.

The ultimate goal is to determine which network is the optimal one. As more events are added the networks training sets, the prediction error of these networks will continually improve. Once the network is trained sufficiently, adding more events will
likely still improve the prediction error. The improvements will not be very large, however. Once the rate of improvement drops, the network is sufficiently trained. Therefore, the optimal network is the first one with a prediction error that is close to the prediction errors of networks after it trained on more events.

7 Results

Given the ability to train multiple neural networks with varying size training sets, it is now necessary to analyze the prediction errors of these networks. Once the results are analyzed and an optimal neural network has been chosen, the performance of the network is compared to the performance of previous prediction methods. This thesis will also analyze the effect of an airplane's runway on prediction accuracy. This analysis will seek to determine if there is any benefit to specializing networks to specific groups of runways.

7.1 Methods for Evaluating the Network

The performance error of the network is based on the prediction errors of individual events and predictions. There are two feasible ways to evaluate the error of the overall network. The first way is to calculate the errors for each individual event. The average of these event errors then becomes the network error. Using this method, all events are treated as being equal. An event with a smaller number of samples is given the same weight as an event with a large number of samples. The other way to evaluate the error of the overall network is to average the errors of every prediction made. In this method, events with more samples are given more weight than events with less samples.

While it would seem logical to give each event equal weighting regardless of the sample size of the event, this is contrary to how the network is trained. The network is
trained based on the individual predictions contained within different events, not events as a whole. Evaluating the network on overall events as opposed to the individual predictions within those events does not accurately indicate how accurate the network is. If having too many predictions for a certain type of event becomes a concern, the events should be modified to all contain the same number of data samples.

Having defined the error of the network as the average of the event errors in the test set, it is necessary to define what constitutes the error of a single event. The error of a single event is calculated using two distinct methods. Both methods are based on the errors of the individual predictions in the event. The methods used are standard error prediction algorithms. The first is called Mean Absolute Error (MAE). Mean absolute error averages the individual prediction errors by summing all of them and dividing by the total number of predictions. The other method is Square Root of Mean Square Error (MSE) [26]. MSE sums the square of the prediction errors. It then divides this sum by the total number of predictions and takes the square root of the result. In MAE, the variance of the error does not come into effect. Using MSE, however, events with a large variance in individual prediction errors will have larger event errors. Both error estimation methods are useful for analysis of the neural networks. Therefore, both methods are used to calculate the event error independently. The equations for the two methods are shown below. Using these equations, the error of the future change in position can be calculated with ease by comparing the expected change to the actual change.
\[
MAE = \frac{|(X_{\text{predicted}} - X_{\text{output}})|}{N}
\]
\[
MSE = \sqrt{\frac{|(X_{\text{predicted}} - X_{\text{output}})^2|}{N}}
\]

### 7.2 Discussion of Evaluation Results

The training methods described in section 5 were used to train three neural networks. The first of these networks was trained solely with events from August 21st, 2001. The training set extracted from August 21st contained 538 events. The second neural network was trained on the data from August 21st as well as data from August 18th. This network was trained using a total of 2745 events. The final network was trained on data from August 18th, 21st, and 22nd. These three days from 2001 contain a total of 4642 events. The weight of the three trained networks can be seen in Appendix B. Additionally, there are more detailed statistics about the extracted events for August 18th, 21st, and 22nd in Appendix A.

A number of networks were analyzed while attempting to determine the optimal network. The initial three networks were trained on all the events from the network's respective training sets. The statistical results of these networks when used to predict events from August 19th, 2001 are shown in table 5. Surprisingly, the predictions appear to become worse as additional days of events are added to the training set. Analysis of the events contained in August 19th indicates that the distribution of landings and takeoffs on the various runways is very similar to that on the 21st. While predictive ability is supposed to be roughly independent from runway, it stands to reason that weather conditions effect the runways used to some degree. There is some basis to an argument
that a network trained on a day with a certain runway pattern would predict days with that same pattern more effectively than networks trained on days with other patterns. Therefore, it makes sense that the network trained only on August 21st is better at making predictions on the 19th than the other networks.

Runway patterns on the 19th provide some explanation as to why the network with the least data performed the best for the given test set. The large overall discrepancy in prediction accuracy seems to indicate that there is something else going on here. The large discrepancies seem to indicate that some amount of overfitting is accruing in networks two and three. The extracted events for August 21st only encompass a few hours of the day. The extracted events for the other three days contain data from the entire days. This means that network two, trained on the 18th and 21st, has approximately four times as many events from the 18th as it does from the 21st. Since the runways used are different on these days, it is entirely possible the weather conditions are different as well. Having a disproportionately low percentage of the data come from the 21st could potentially be causing planes flying under that pattern less weight in the neural network. Therefore while network one is most likely overfit to the pattern on the 21st, network two is possibly overfit to the pattern on the 18th. This would explain why network one predicts the 19th significantly better than networks two or three.
<table>
<thead>
<tr>
<th>Time</th>
<th>ASDE</th>
<th>X Data</th>
<th>Neural Network 1</th>
<th>Neural Network 2</th>
<th>Neural Network 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>5</td>
<td>43.0</td>
<td>104.7</td>
<td>47.2</td>
<td>72.4</td>
<td>49.3</td>
</tr>
<tr>
<td>10</td>
<td>128.4</td>
<td>134.4</td>
<td>138.2</td>
<td>200.2</td>
<td>143.8</td>
</tr>
<tr>
<td>15</td>
<td>273.9</td>
<td>189.8</td>
<td>293.0</td>
<td>406.5</td>
<td>308.0</td>
</tr>
<tr>
<td>20</td>
<td>478.1</td>
<td>261.9</td>
<td>519.3</td>
<td>700.8</td>
<td>536.9</td>
</tr>
<tr>
<td>25</td>
<td>747.6</td>
<td>368.1</td>
<td>814.6</td>
<td>1079.8</td>
<td>849.5</td>
</tr>
<tr>
<td>30</td>
<td>1070.7</td>
<td>504.0</td>
<td>1169.0</td>
<td>1535.6</td>
<td>1201.5</td>
</tr>
<tr>
<td></td>
<td>104.7</td>
<td>142.5</td>
<td>110.5</td>
<td>144.5</td>
<td>107.7</td>
</tr>
<tr>
<td>10</td>
<td>134.4</td>
<td>200.6</td>
<td>137.3</td>
<td>203.3</td>
<td>137.8</td>
</tr>
<tr>
<td>15</td>
<td>189.8</td>
<td>287.1</td>
<td>192.6</td>
<td>287.8</td>
<td>194.7</td>
</tr>
<tr>
<td>20</td>
<td>261.9</td>
<td>390.2</td>
<td>265.5</td>
<td>391.1</td>
<td>267.8</td>
</tr>
<tr>
<td>25</td>
<td>368.1</td>
<td>525.8</td>
<td>372.6</td>
<td>524.3</td>
<td>378.0</td>
</tr>
<tr>
<td>30</td>
<td>504.0</td>
<td>688.7</td>
<td>509.6</td>
<td>686.1</td>
<td>514.2</td>
</tr>
<tr>
<td></td>
<td>29.8</td>
<td>49.0</td>
<td>36.8</td>
<td>56.5</td>
<td>39.4</td>
</tr>
<tr>
<td>10</td>
<td>87.4</td>
<td>135.9</td>
<td>113.7</td>
<td>165.2</td>
<td>120.3</td>
</tr>
<tr>
<td>15</td>
<td>184.4</td>
<td>277.1</td>
<td>249.3</td>
<td>345.0</td>
<td>265.4</td>
</tr>
<tr>
<td>20</td>
<td>317.7</td>
<td>465.4</td>
<td>452.2</td>
<td>607.2</td>
<td>474.0</td>
</tr>
<tr>
<td>25</td>
<td>490.6</td>
<td>710.2</td>
<td>716.7</td>
<td>945.2</td>
<td>759.4</td>
</tr>
<tr>
<td>30</td>
<td>697.0</td>
<td>1007.0</td>
<td>1036.5</td>
<td>1353.9</td>
<td>1087.6</td>
</tr>
<tr>
<td></td>
<td>71.094</td>
<td>103.78</td>
<td>72.284</td>
<td>102.6</td>
<td>71.337</td>
</tr>
<tr>
<td>10</td>
<td>99.551</td>
<td>151.78</td>
<td>100.33</td>
<td>152.38</td>
<td>100.29</td>
</tr>
<tr>
<td>15</td>
<td>133.4</td>
<td>210.5</td>
<td>132.6</td>
<td>208.7</td>
<td>133.72</td>
</tr>
<tr>
<td>20</td>
<td>173.7</td>
<td>272.6</td>
<td>171.6</td>
<td>269.3</td>
<td>172.02</td>
</tr>
<tr>
<td>25</td>
<td>224.3</td>
<td>345.7</td>
<td>220.2</td>
<td>338.8</td>
<td>222.66</td>
</tr>
<tr>
<td>30</td>
<td>292.6</td>
<td>432.0</td>
<td>284.2</td>
<td>422.8</td>
<td>286.07</td>
</tr>
</tbody>
</table>

Table 5: Statistical results from August 19th for the neural networks trained with all data from their respective days.
In order to prevent the problem of overfitting, the rest of the networks were trained with an equal number of events from their respective days. Each day included in a training set, contributes 200 landings and 200 takeoffs to the training set. These landings and takeoffs are selected randomly from each day’s events. Another potential way to group events to prevent overfitting is to group them by runway and ensure each runway has sufficient representation in the test set. This idea contains some merit because the two most likely things to affect a plane’s motion are the current weather conditions or the plane’s runway. A training set that encompasses a sufficient number of days, however, should have an appropriate proportion of events from each runway. Additionally, the plane’s runway can be determined by its position and heading. If the runway is a mitigating factor in the plane’s motion, then separate neural networks should be developed for each runway. The effect of runway groups on neural networks is explored in section 7.4.2.

The training process sets aside 25% of the extracted data as a test set. This test set is used to generate performance statistics on the newly trained network. Since it is a subset of the extracted data, the test set is better used as an indicator as to how well the network has trained than as a way of comparing the performance of networks. The statistical performance of each network on its respective test set is shown in Table 6. The Mean Squared Error (MSE) and Mean Average Error (AE) are given for each of the three networks. The time parameter is the number of seconds into the future the network is attempting to predict.
<table>
<thead>
<tr>
<th></th>
<th>Neural Network 1</th>
<th>Neural Network 2</th>
<th>Neural Network 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>5 seconds</td>
<td>42.1</td>
<td>65.7</td>
<td>40.7</td>
</tr>
<tr>
<td>10 seconds</td>
<td>129.5</td>
<td>181.5</td>
<td>117.1</td>
</tr>
<tr>
<td>15 seconds</td>
<td>275.7</td>
<td>368.8</td>
<td>251.1</td>
</tr>
<tr>
<td>20 seconds</td>
<td>483.9</td>
<td>629.4</td>
<td>442.9</td>
</tr>
<tr>
<td>25 seconds</td>
<td>757.3</td>
<td>965.9</td>
<td>697.4</td>
</tr>
<tr>
<td>30 seconds</td>
<td>1081.6</td>
<td>1366.8</td>
<td>1004.3</td>
</tr>
</tbody>
</table>

Table 6: Network Test Set Comparison

Table 6 seems to indicate that the ASDE neural network performed better as the sample set was increased and the ASR network started to perform worse. The results of this table are misleading, however. The ASR network is trained solely on data read by the ASR radar. The ASR radar detects and tracks landing events and then passes the plane off to the ASDE radar once it hits the ground. Planes taking off are initially tracked by the ASDE radar system but once the plane leaves the ground the AMASS system stops monitoring it altogether. Therefore, all the data used to train and test the ASR neural network comes from the beginning of landing events.

Based on the way AMASS operates, when it first starts monitoring a plane, it does have any information about the plane prior to that time. Therefore, the change in the plane’s position over ten second is identical to the change in its position over eight
seconds and so on. This means that all ten of the inputs to the neural network are identical. This leads to imperfect inputs to the neural network, which in turn decreases the accuracy of the neural network prediction. While the identical inputs are slowly filtered out as the system receives more data about the plane, it takes ten seconds time for all ten of the network's inputs to become distinct. This presents a problem as many landing events have little data before the plane is passed to the ASDE radar. Therefore, the ASR data for many landing events consists almost entirely of incomplete neural network input.

The data in Table 6 also indicates that the ASDE predictions are more accurate than ASR predictions when predicting a short time into the future. For predictions farther in the future, the ASR network is more accurate. This occurrence results from the nature of the radar systems. The ASDE radar signal has more noise and is a lower quality signal than the ASDE radar signal. ASR's low sampling rate (deltaT = 4s) and distance from the targets it is tracking causes the noise in its signal. The noise causes short times to be imprecise. On the other hand, plane's being tracked by ASR typically move at very close to constant speeds. The roughly average speeds over time results in greater accuracy for predictions farther into the future.

The neural networks training sets contain data from three different days. An examination of the events on these days shows that on August 21st, the sole day used to train nn1, these events have a higher percentage of landings with long approaches and thus more ASR data was acquired as compared to other days. Longer approaches equate to a higher percentage of ASR data with ten valid inputs. They also lead to predictions that are more accurate. The test set for each neural network is built from its training data.
Since the training data used to train nn1 has a higher percentage of events with long approaches, it is logical that the neural network would predict more accurately on it. In other words, the decreasing performance of the ASR neural network may be more a factor of a decreasing percentage of long approaches in the test set and less an indicator of worse predictions. In order to determine if the network is actually performing worse, all three networks must be tested on the same test set. Building a universal test will also help to verify whether the ASDE neural network is increasing in accuracy as further data is added.

The universal test set contains events from all four days currently extracted. In order to prevent a bias towards any particular conditions, each day contributes 200 landings and 200 takeoffs to the test set. This also guarantees a fair balance between landing and takeoff events. The events in this test set were chosen randomly and completely independently of the training sets.

The uniform test set prediction statistics for the three neural networks are shown in Table 7. The same statistics are also presented in Table 8 with an additional parameter of x or y direction. The average errors and mean standard errors for both the ASDE and ASR networks in all three of the neural networks are very similar. None of the networks comparative average errors differ by more than about 5%. The mean standard errors are even closer than the average errors. None of these differences are statistically significant. This implies that the networks all performed approximately equally as well on the test set which in turn implies that all three networks have sufficient data in their training sets to be considered optimally trained.
<table>
<thead>
<tr>
<th></th>
<th>Neural Network 1</th>
<th>Neural Network 2</th>
<th>Neural Network 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>ASDE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>37.8</td>
<td>61.6</td>
<td>38.2</td>
</tr>
<tr>
<td>10 seconds</td>
<td>114.1</td>
<td>169.9</td>
<td>110.7</td>
</tr>
<tr>
<td>15 seconds</td>
<td>242.2</td>
<td>342.9</td>
<td>236.1</td>
</tr>
<tr>
<td>20 seconds</td>
<td>427.1</td>
<td>587.7</td>
<td>414.7</td>
</tr>
<tr>
<td>25 seconds</td>
<td>673.0</td>
<td>906.1</td>
<td>650.9</td>
</tr>
<tr>
<td>30 seconds</td>
<td>964.6</td>
<td>1286.3</td>
<td>936.8</td>
</tr>
<tr>
<td>ASR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>98.9</td>
<td>139.5</td>
<td>104.6</td>
</tr>
<tr>
<td>10 seconds</td>
<td>131.7</td>
<td>201.5</td>
<td>137.3</td>
</tr>
<tr>
<td>15 seconds</td>
<td>180.2</td>
<td>282.0</td>
<td>185.7</td>
</tr>
<tr>
<td>20 seconds</td>
<td>234.5</td>
<td>369.7</td>
<td>236.4</td>
</tr>
<tr>
<td>25 seconds</td>
<td>315.3</td>
<td>481.0</td>
<td>315.1</td>
</tr>
<tr>
<td>30 seconds</td>
<td>420.0</td>
<td>610.0</td>
<td>419.1</td>
</tr>
</tbody>
</table>

Table 7: Neural Network Prediction Statistics by Radar Type (August 19th)
<table>
<thead>
<tr>
<th></th>
<th>Neural Network 1</th>
<th></th>
<th>Neural Network 2</th>
<th></th>
<th>Neural Network 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>ASDE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>40.6</td>
<td>62.2</td>
<td>40.3</td>
<td>61.3</td>
<td>39.7</td>
<td>58.6</td>
</tr>
<tr>
<td>10 seconds</td>
<td>124.4</td>
<td>174.5</td>
<td>117.2</td>
<td>169.4</td>
<td>119.1</td>
<td>169.3</td>
</tr>
<tr>
<td>15 seconds</td>
<td>265.5</td>
<td>357.0</td>
<td>250.8</td>
<td>345.2</td>
<td>254.2</td>
<td>346.6</td>
</tr>
<tr>
<td>20 seconds</td>
<td>470.5</td>
<td>617.1</td>
<td>441.1</td>
<td>593.8</td>
<td>443.3</td>
<td>595.9</td>
</tr>
<tr>
<td>25 seconds</td>
<td>744.6</td>
<td>958.9</td>
<td>694.5</td>
<td>917.1</td>
<td>700.4</td>
<td>920.7</td>
</tr>
<tr>
<td>30 seconds</td>
<td>1069.6</td>
<td>1367.2</td>
<td>1000.7</td>
<td>1307.3</td>
<td>1005.2</td>
<td>1312.0</td>
</tr>
<tr>
<td>X Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>112.7</td>
<td>151.8</td>
<td>123.3</td>
<td>157.8</td>
<td>113.3</td>
<td>151.5</td>
</tr>
<tr>
<td>10 seconds</td>
<td>145.4</td>
<td>216.8</td>
<td>153.6</td>
<td>225.9</td>
<td>146.8</td>
<td>219.5</td>
</tr>
<tr>
<td>15 seconds</td>
<td>203.9</td>
<td>307.4</td>
<td>211.4</td>
<td>317.1</td>
<td>205.4</td>
<td>310.0</td>
</tr>
<tr>
<td>20 seconds</td>
<td>272.1</td>
<td>406.6</td>
<td>276.8</td>
<td>407.3</td>
<td>274.4</td>
<td>407.9</td>
</tr>
<tr>
<td>25 seconds</td>
<td>375.8</td>
<td>537.6</td>
<td>378.0</td>
<td>537.6</td>
<td>377.5</td>
<td>535.7</td>
</tr>
<tr>
<td>30 seconds</td>
<td>511.8</td>
<td>694.1</td>
<td>514.3</td>
<td>694.5</td>
<td>513.8</td>
<td>692.7</td>
</tr>
<tr>
<td>ASR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>35.0</td>
<td>61.1</td>
<td>36.1</td>
<td>61.7</td>
<td>35.8</td>
<td>59.2</td>
</tr>
<tr>
<td>10 seconds</td>
<td>103.7</td>
<td>165.3</td>
<td>104.2</td>
<td>169.4</td>
<td>107.4</td>
<td>169.6</td>
</tr>
<tr>
<td>15 seconds</td>
<td>218.9</td>
<td>328.2</td>
<td>221.3</td>
<td>337.0</td>
<td>229.6</td>
<td>340.7</td>
</tr>
<tr>
<td>20 seconds</td>
<td>383.7</td>
<td>556.8</td>
<td>388.2</td>
<td>572.9</td>
<td>401.0</td>
<td>583.0</td>
</tr>
<tr>
<td>25 seconds</td>
<td>601.4</td>
<td>850.0</td>
<td>607.3</td>
<td>875.2</td>
<td>632.3</td>
<td>891.7</td>
</tr>
<tr>
<td>30 seconds</td>
<td>859.5</td>
<td>1199.9</td>
<td>872.9</td>
<td>1237.1</td>
<td>906.7</td>
<td>1265.5</td>
</tr>
<tr>
<td>Y Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>85.086</td>
<td>126.08</td>
<td>85.868</td>
<td>123.15</td>
<td>83.104</td>
<td>122.18</td>
</tr>
<tr>
<td>10 seconds</td>
<td>118.09</td>
<td>184.99</td>
<td>121.03</td>
<td>188.6</td>
<td>117.14</td>
<td>184.92</td>
</tr>
<tr>
<td>15 seconds</td>
<td>156.4</td>
<td>254.2</td>
<td>160.0</td>
<td>260.6</td>
<td>154.62</td>
<td>253.49</td>
</tr>
<tr>
<td>20 seconds</td>
<td>196.9</td>
<td>328.7</td>
<td>196.1</td>
<td>324.8</td>
<td>196.84</td>
<td>327.22</td>
</tr>
<tr>
<td>25 seconds</td>
<td>254.8</td>
<td>416.7</td>
<td>252.1</td>
<td>414.2</td>
<td>249.07</td>
<td>411.43</td>
</tr>
<tr>
<td>30 seconds</td>
<td>328.2</td>
<td>512.4</td>
<td>323.9</td>
<td>508.5</td>
<td>319.69</td>
<td>505.72</td>
</tr>
</tbody>
</table>

Table 8: Neural Network Prediction Statistics by Direction and Radar Type (August 19th)
7.3 Testing the Network on AMASS events

Deriving performance statistics on the predictive accuracy of the neural network gives an indication of its performance relative to linear prediction as well as other networks. The ultimate goal of the network, however, is to predict alarm scenarios sooner and more accurately. A network that improves positional prediction should increase the accuracy and lead time of predictions. It is important to verify that it does in fact predict alarm conditions more accurately, however.

AMASS has a number of modes it can be run in. One of these modes runs the AMASS system as it normally would with the exception that plane positions are read from past AMASS log files instead of from the radars. By running AMASS on log files from past days, it is possible to simulate real alarm scenarios that have occurred in the past and compare AMASS linear predictions for the alarm to the neural network positions. Those results from those comparisons will not be discussed in this paper.

Due to the low number of alarm events contained in the AMASS log files available, it is difficult to perform direct rigorous analysis of the ability of the system to predict alarms. AMASS has a feature which allows for injection of planes taking off, taxiing, landing, or standing still into the AMASS system. This feature was taken advantage of in an attempt to artificially generate additional alarm scenarios. Event 1 was a plane landing approaching a stopped plane on near the end of the runway. No actual collision occurs but the landing plane comes very close to the stopped one before turning off the runway. The second event involves the same plane only the stopped plane has been moved to the front of the runway. Event 3 involves a plane trying to take off on a runway where another plane is taxiing. Event 4 consists of a plane trying to land on a runway
another plane is already taxiing on and Event 5 has a plane taking off with another plane already on its runway. There was also an attempt to create an alarm between a landing plane and a plane taxiing across the runway. Due to the workings of AMASS logic, however, the taxiing plane was not registered with the safety logic and thus the even did not generate any output.

Graphs of the results from these situations can be viewed in Appendix G. Essentially, these graphs show that the neural network performed slightly worse or about the same as the AMASS linear predictive model in all the situations. Initially, this would seem to be a cause for alarm. There are a number of factors to consider. Foremost is that the injected planes part of the AMASS system is designed more as a debugging feature for AMASS then as a system for detailed analysis. Therefore, it is extremely buggy and not fully reliable. AMASS uses a setting called OpConfig to specify which runway is being actively used and to help prevent false alarms on unused runways. If the OpConfig was set wrong when the artificial planes were injected into the system, the entire AMASS program would crash. Additionally, the neural network predictions for the injected planes were incorrect and clearly not generated using either the ASR or ASDE neural networks. Despite the fact that the predictions were re-generated using the appropriate networks, these errors occurring calls into question the reliability of the results.

Even assuming that the final results are accurate, there are still explanations for the poor performance of the neural network on injected events. One of the planes in all these events was either slow moving or stopped. The neural network is trained largely on high speed data because fast moving planes are harder to predict than slow moving planes. While having one slow moving plane alone does not fully account for performing
worse than the linear model, it still decreases the performance of the neural network versus the linear model. The other factor is that the artificially generated events move under perfect linear motion. Since they are not real events, they are not effected by the underlying patterns the neural network is trained to accept. In fact, the linear model should predict these events perfectly while the neural network should predict them worse than if they were real events. These factors combined help to explain why the neural network performs worse than the neural network on injected all events. Overall, the fact that injected events do not necessarily behave like real events and the lack of reliability of the injected plane system lead to the conclusion that injected events are not a reliable way to test the performance of the neural network.

7.4 Situational Analysis

The initial logic behind predicting plane position using neural networks stemmed from the hypothesis that a plane’s motion as it lands on or takes off from a runway is influenced by complex factors difficult for a pure physics model to completely encapsulate. These factors and consequently the plane’s motion may or may not be airport specific. When the neural network is trained with data from an airport, it learns the results of these complex factors but not what the factors actually are. This means that there is no information readily available on the conditions that modify aircraft’s patterns of motion.

While it is not possible to determine from the neural network what factors influence the plane’s motion, it is possible to try to analyze what situations modify this motion. This can be accomplished by comparing the performance of individual networks trained on different types of events to a network that includes all those events. For example, to
determine if a plane’s runway effects its motion, build separate neural networks for each runway and compare their overall performance to that of a network built using a combination of all the runways. If the results are similar than an aircraft’s runway does not effect its motion.

There are many elements that could change the factors that determine a plane’s motion. Aircraft may move differently based on what runway they are on, whether they are taking off or landing, or weather conditions. To truly determine the effect of weather conditions on a plane’s motion, specific information about weather conditions at the airport for the training dates as well as data from dates at different times of year would be required. A degree of intuition for what types weather affect taking off and landing would also be necessary. The AMASS data currently available is insufficient for such a task. There is sufficient data available, however, to determine if the runway or whether a plane is taking off or landing affect the aircraft’s motion. This thesis attempts to test the hypothesis that a plane’s motion is affected by its runway and status as a takeoff or landing.

7.4.1 Landing/Takeoff Analysis

The hypothesis that a plane’s status as a take off or landing affects its motion is based on the assumption that the complex factors that affect the plane’s motion are different when the plane is accelerating from when it is decelerating. This hypothesis will be tested using the ASDE neural network, only. This is due to the nature of AMASS in that takeoff events do not contain any ASR data. In order to perform situational analysis on the neural network, one of the three trained networks had to be chose to be compared to situational networks built from its own training set. Any of the networks
would have been fine to use. Neural Network 2 was chosen because its training set is the smallest that has representation from all the major runways.

<table>
<thead>
<tr>
<th></th>
<th>Land Network</th>
<th>Takeoff Network</th>
<th>Combined</th>
<th>Neural Network 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>X Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>32.9</td>
<td>49.3</td>
<td>71.3</td>
<td>91.2</td>
</tr>
<tr>
<td>10 seconds</td>
<td>79.9</td>
<td>114.2</td>
<td>241.0</td>
<td>297.5</td>
</tr>
<tr>
<td>15 seconds</td>
<td>137.7</td>
<td>192.6</td>
<td>537.7</td>
<td>640.5</td>
</tr>
<tr>
<td>20 seconds</td>
<td>197.4</td>
<td>271.8</td>
<td>978.9</td>
<td>1127.8</td>
</tr>
<tr>
<td>25 seconds</td>
<td>255.6</td>
<td>346.7</td>
<td>1573.5</td>
<td>1761.7</td>
</tr>
<tr>
<td>30 seconds</td>
<td>309.6</td>
<td>415.2</td>
<td>2287.1</td>
<td>2501.1</td>
</tr>
<tr>
<td>Y Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>27.1</td>
<td>47.0</td>
<td>71.3</td>
<td>91.2</td>
</tr>
<tr>
<td>10 seconds</td>
<td>64.6</td>
<td>102.6</td>
<td>241.0</td>
<td>297.5</td>
</tr>
<tr>
<td>15 seconds</td>
<td>111.6</td>
<td>167.5</td>
<td>537.7</td>
<td>640.5</td>
</tr>
<tr>
<td>20 seconds</td>
<td>161.9</td>
<td>234.6</td>
<td>978.9</td>
<td>1127.8</td>
</tr>
<tr>
<td>25 seconds</td>
<td>212.4</td>
<td>297.6</td>
<td>1573.5</td>
<td>1761.7</td>
</tr>
<tr>
<td>30 seconds</td>
<td>264.1</td>
<td>361.2</td>
<td>2287.1</td>
<td>2501.1</td>
</tr>
</tbody>
</table>

Table 9: Landing/Takeoff Network comparison

Table 9 lists the statistical comparison between networks trained on landings and takeoffs separately and the network trained on both collectively. The network combined is the combined statistics for the land network and the takeoff network. Neural Network 2 is the network trained with both takeoffs and landings. The analysis indicates that there is no benefit to training separate networks for landing and taking off events. The hypothesis that an airplane's status as a takeoff or landing affects its motion is shown to be false. Of interest is the large difference between landing errors and takeoff errors. The predictions for landing events were a factor of 10 more accurate than those for takeoffs. The fact that takeoffs are predicted with much lower accuracy indicates that the
network may have a difficult time determining when a plane is taking off and when it is just taxiing. Whether takeoffs actually are harder to predict or this is just a fluke case is difficult to determine without a more in depth analysis.

7.4.2 Result Analysis by Runway

Training specialized neural networks for different runways can be done a couple ways. Either each runway can be trained on a separate network or runways can be split into groups with a different network for each group. The goal of this thesis in regards to networks for different runways is not to implement an optimal runway based network system but merely to determine if there is merit to developing such a system. If a runway based network can improve performance, then an approach based on different networks for groups of related runways should improve performance as well. For this paper, the runways were divided into three groups. One group contained all the runways in the 40/220 degree directions. The others contain runways in the 90/270 and 140/320 degree directions, respectively.

The statistical data using multiple networks and a single network is shown below in Table 10. Similarly to the landing/takeoff analysis, using multiple networks for different runways does not appear to improve accuracy. An aircraft’s runway appears to have minimal or no correlation to the airplane’s motion. The results indicate that there is no reason to spend time developing multiple networks for different runways at the same airport.
<table>
<thead>
<tr>
<th></th>
<th>Ind. Network for Each Runway</th>
<th>Single Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>ASDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>40.08922</td>
<td>60.40356</td>
</tr>
<tr>
<td>10 seconds</td>
<td>120.7587</td>
<td>171.2492</td>
</tr>
<tr>
<td>15 seconds</td>
<td>258.0668</td>
<td>354.3787</td>
</tr>
<tr>
<td>20 seconds</td>
<td>464.1693</td>
<td>619.7556</td>
</tr>
<tr>
<td>25 seconds</td>
<td>733.0334</td>
<td>966.0107</td>
</tr>
<tr>
<td>30 seconds</td>
<td>1068.539</td>
<td>1393.033</td>
</tr>
<tr>
<td>ASR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>114.8291</td>
<td>148.7758</td>
</tr>
<tr>
<td>10 seconds</td>
<td>152.7262</td>
<td>216.4816</td>
</tr>
<tr>
<td>15 seconds</td>
<td>201.373</td>
<td>297.33</td>
</tr>
<tr>
<td>20 seconds</td>
<td>273.7381</td>
<td>399.3837</td>
</tr>
<tr>
<td>25 seconds</td>
<td>380.9887</td>
<td>535.1999</td>
</tr>
<tr>
<td>30 seconds</td>
<td>519.3812</td>
<td>698.477</td>
</tr>
<tr>
<td>ASDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>41.65142</td>
<td>66.67352</td>
</tr>
<tr>
<td>10 seconds</td>
<td>123.9512</td>
<td>187.7899</td>
</tr>
<tr>
<td>15 seconds</td>
<td>274.6635</td>
<td>403.4973</td>
</tr>
<tr>
<td>20 seconds</td>
<td>532.7069</td>
<td>763.4105</td>
</tr>
<tr>
<td>25 seconds</td>
<td>842.8242</td>
<td>1193.81</td>
</tr>
<tr>
<td>30 seconds</td>
<td>1233.68</td>
<td>1730.906</td>
</tr>
<tr>
<td>ASR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 seconds</td>
<td>145.6675</td>
<td>272.7407</td>
</tr>
<tr>
<td>10 seconds</td>
<td>152.9072</td>
<td>250.4695</td>
</tr>
<tr>
<td>15 seconds</td>
<td>288.6437</td>
<td>568.489</td>
</tr>
<tr>
<td>20 seconds</td>
<td>376.1264</td>
<td>750.0005</td>
</tr>
<tr>
<td>25 seconds</td>
<td>408.7145</td>
<td>763.5263</td>
</tr>
<tr>
<td>30 seconds</td>
<td>465.0717</td>
<td>821.8046</td>
</tr>
</tbody>
</table>

Table 10: Runway Network Analysis
8 Generalizing the Process

The process to generate and evaluate an optimized neural network has now been successfully implemented for Chicago O Hare(ORD) airport. The next step is to generalize this process so that it can be implemented to create an optimized neural network for any airport. The primary goal is to create a suite of tools that can extract events, train a network, evaluate a network and perform any other functions related to generating a network for airplane position prediction. This suite is designed to be sufficient to allow anyone with a user-level knowledge of matlab to generate an optimized neural network. Using the tools in the suite should not require any matlab programming. The tools in the suite are split up into three categories: event extraction tools are anything involved in extracting training events from the 3D matrix; network training tools are any tools related to using extracted events to train a network; results analysis tools are tools used to analyze and generate performance statistics on an already built neural network.

8.1 Generalized Event Extraction

8.1.1 Overview and Tool Design

The difficulty in generalizing event extraction lies in the fact that runways are placed in different locations at different airports. This makes it impossible to generalize the whole process. Instead the process of what makes a runway is generalized. The algorithm used is identical to the extraction algorithm described in section 4.4. Extracting events from a runway is based on the direction of the runway, whether the event is a landing or takeoff, and a bounding box on the plane’s position. These parameters must be determined by the user for each extraction and passed to the event
extraction function. Guidelines on how to generate these parameters can be found in section 7.1.2 under the description of the ExtractRunway tool.

The rest of the tools in the event extraction toolset were built to extract single events. Some of these tools were designed as tools to assist in extract runway events and others were designed to help in testing neural networks. Most of these tools are not necessary for designing an optimal neural network but may be useful for applications such as evaluating the performance of different networks for a single random event. Some of these tools would also be useful to someone who wishes to modify the current event extraction methodology.

8.1.2 Tool Description

ExtractRunwayEvents - Extracts events from the 3D matrix, of airplanes landing/taking of on an arbitrary runway. Selection of events is based on coordinates of the airplane, as well as speed and direction. This function calls FindEvent and ExtractEvent to obtain coordinates from the 3D matrix. Each event is stored in the tset matrix separated from the proceeding and following events by a row of zeros. Users of this function should determine the runway direction by multiplying ten times the runway number. Thus, runway 04L would be 40 degrees and runway 22R would be 220 degrees. Determining the placement of the bounding positional box is a little more difficult. For landing events, the box should be placed on the approach to the runway before the actual runway begins. It should be placed far enough away from the actual runway so that the box does not overlap the area where the plane is passed from the ASDE radar to ASR. For takeoff events, the bounding box should be placed towards the end of the runway. For long runways, the placement of this box is a somewhat tricky. It should be placed early
enough on the runway to pick up planes that may take off before the end of the runway but deep enough on the runway so that it does not extract landing events where the plane lands in the middle of the runway of the opposite direction. The positional information should be entered as a two-dimensional matrix of the form [lower_bound left_bound; upper_bound right_bound].

ExtractEvent — This function takes five inputs. Log3D is a 3D log matrix from matlab. Record and pid are the time index and plane id of the event. Sec_before and sec_after are the number of seconds before the event and the number of seconds after the event to be extracted. Extract event locates the appropriate event in the 3D matrix using the given record and pid. It then extracts data from record-sec_before to record+sec_after. When planes switch from being tracked by ASDE to ASR or vice-versa, their ids change. This function attempts to track changes from ASDE to ASR and vice-versa. If data is being extracted backward in time from an ASDE plane and valid data runs out, the function attempts to look for a plane with any id in the vicinity of the original plane and tracks this plane. The function does the same thing if valid ASR data runs out while tracking forward in time. The function returns sec_before+sec_after+1 seconds of data. Each second of data contains an x position, a y position, and a radar type in the first, second, and third columns, respectively.

FindEvent — FindEvent takes Log3D, a 3D log matrix, Xcoord, an x coordinate, and Ycoord, a y coordinate, as inputs. It returns a matrix, found, where each row contains an index, pid, and tid. Pids are the plane index in the 3D matrix and do not necessarily correspond to pids in AMASS. Tids are the target plane s ID as assigned by the ASR radar and do correspond to tids in AMASS. The matrix found is built from all the rows
of the 3D matrix that have an x position equal to Xcoord and a y position equal to Ycoord. Therefore, if there are n entries in the 3D matrix with x position equal to Xcoord and y position equal to Ycoord, found will have n-rows.

randEvent — This tool takes a list of extracted events as input. From this list, it randomly chooses one event. It returns that entire extracted event as an output, entitled randEvent.

ExtractCount — Extract Count performs a count on the number of events contained within a list of events. This can be used to determine the number of extractions and verify extracted data. The input events should be a list of events separated by rows of zeros. The output, count, is the count of the number of events.

8.2 Generalized Network Training

8.2.1 Overview and Tool Design

The goal of the network training toolset is to provide tools to facilitate the training of a new neural network given extracted events as inputs. This toolset contains tools that can perform the training process themselves as well as tools which could be used to develop a new training process. It also contains tools for combining multiple extracted events into a single group for the purpose of training.

8.2.2 Tool Description

TrainNN — This function is the primary function used to train a new neural network. It accepts as input a struct, events, that contains numerous event lists. These event lists are combined to form one large event list. The function then utilizes getTimeOuts, described below, to turn this event list into input/output pairs in training, validation, and test sets. TrainNN makes use of the matlab built in neural network functions to train the ASDE
and ASR networks and generate statistics on them. TrainNN returns the networks and
statistics in the variables ASDEnet, ASRnet, and stats.

getTrainOuts — The purpose of this function is to generate training, validation, and test
sets from a list of extracted events. The function calls BuildTS26 to generate ASDE and
ASR input/output pairs. It then randomly splits these input/output pairs into the different
sets. It places 50% of the data into the training set, 25% into the validation set, and 25%
into the test set. It the returns these data sets for each network.

BuildTS26 - BuildTS26 takes a list of events as input. This list of events should be
comprised of a series of related x-positions, y-positions, and an indicator specifying the
radar type. These events are used to build a matrix of valid inputs to the neural network,
consisting of positional change values. They are also used to generate expected outputs
from the neural networks which consists of positional change values, as well.

combineEvents — CombineEvents receives a struct containing an arbitrary number of
extracted events. It takes all of the extracted event lists in the struct and returns a single
list containing all the events. This function is very useful as the extracted events are
generally stored separately by runway while most functions that use these extracted
events require a single list of events.

8.3 Generalized Results Analysis

8.3.1 Overview and Tool Design

The result analysis toolset was built to provide tools for analysis of already
trained networks. It consist primarily of tools that have the ability to simulate a large
number of predictions using the neural network and generate performance statistics based
on these predictions and actual values. The other toolset contain all the tools necessary to
build a neural network for any airport. This toolset is not needed to build the actual network but is useful for verifying an already built neural network’s accuracy and for determining if the current network has been trained with sufficient data so as to be an optimal neural network. The network is considered optimized when adding additional data to the training set no longer improves the accuracy of the network significantly.

8.3.2 Tool Description

TestNN — TestNN generates average error and mean standard error for a neural network. As input, it takes a network, an input matrix with n input pairs, and an output matrix with n output sets. It simulates the neural network using the n inputs and then compares them to the desired n-outputs to generate the statistical data. This statistical data is the output of the function. TestNN is the function used by trainscript to generate the statistics on the newly generated networks.

TestNNCoords — TestNNCoords provides a tool that is able to take a single event and simulate the neural networks being used to predict that event in its entirety. As input, it takes a single event. TestNNCoords uses this event to build input/output pairs for the neural network. It then simulates all the inputs using the appropriate neural network (ASDE or ASR) and generates statistics based on these results. The function then outputs these statistics along with all the predicted and actual values. This function is particularly useful as it provides all the information necessary to plot the predicted values versus actual values over time.

TestNNRandomEvent — This function takes a list of events as input. From this list of inputs, it chooses a single random event using the tool RandomEvent. This event is then passed into TestNNCoords. The results of TestNNCoords are returned as the results of
TestNNRandomEvent. TestNNRandomEvent is useful for quickly and easily generating prediction information for a single random event.

*TestNNEvents* — TestNNEvents generates statistical analysis for the ASR and ASDE neural networks in both the x and y directions. The function utilizes BuildTS26 to generate input/output pairs for the ASDE and ASR in the two directions. It then calls testNN four times, one for each radar in each direction and generates statistics for their performance. These statistics become the output of TestNNCoors2.

*randomN* — This tool takes a list containing events and a number, n. It randomly selects n distinct events from the inputted event list and returns these n events in a new event list.

*removeOutliers* — This tool is used to filter out events that skew the performance of the network. This tool takes an ASDE neural network, and ASR neural network, and an event list as inputs. It uses the networks to predict each event in the event list. Any event with a maximum error above 10,000 feet is then removed from the event list. The resulting event list, which consists of events with maximum errors below 10,000 feet, is returned. This function should be used to filter outlier events from the data set. After it is run, the neural network should be retrained.

### 8.4 GUI Form

The toolset described above provides enough information for a matlab user to generate an optimized neural network for any airport and generate reports on the network's accuracy. Using these tools requires an understanding of a number of matlab data structures as well as some work coordinating multiple files. For example, if extractions are performed for ten different files, these extractions must all be collected together into a single matrix to train the network. To facilitate handling data from
multiple files and to further simplify the extraction process, simple GUI forms for the different steps of the process were developed. These forms handle tasks from developing valid extractions for the network to collecting extracted data from multiple files and using it to train a neural network.

8.4.1 Event Extraction GUI

The event extraction GUI coordinates the process of building a set of extractions and performing these extractions on multiple files. It provides the ability to add extractions and provide all the generic data relevant to each extraction. These extractions can be assigned names as well as be saved to or loaded from a file. Finally, the GUI provides the ability to run a set of extractions currently loaded on any 3D log file.

Figure 9 is a screenshot of the Extract Events GUI. The left side of the GUI contains a list of extractions. Any one of these extractions can be selected at a time using either the keypad or a mouse. The information for the currently selected extraction is displayed in the other fields of the form. The name, bounds, degree, and Takeoff/Landing fields all display information about the current extraction. All these fields are editable and update automatically. The extractions can be saved to a or loaded from a file with the save and load buttons, respectively. Both buttons provide a file dialog to choose a file for loading/saving.
The buttons along the bottom of the form provide functionality for modifying the list of extractions as well as running the extractions. The add runway adds a new extraction to the list with the default name `newEvent`. Delete runway deletes the currently selected runway from the list. Run extraction is more complex than the other two functions. Run extraction loads the 3D log contained with the file name and variable name specified in their respective text fields. Each extraction in the list is run on the 3D matrix using the parameters specified for that extraction. When the extraction process starts, the user is given a dialog box that can be used to choose a file to save the extractions to once they are complete. The completed extractions are all stored in a struct titled `extracted`. The results of each extraction are stored in this struct under the extractions specified name. If multiple extractions have the same name, they are named
event1, event2, event3, and so on. Upon completion of the extractions, the network training GUI is loaded with the recently extracted data.

### 8.4.2 Network Training GUI

The network training GUI provides functionality to combine multiple extracted events from multiple files. Extracted event lists can be saved and loaded. The individual extracted event lists can also be renamed. The ability to rename extracted event lists makes it simpler to combine files that contain the same extractions run on different log files.

![Network Training GUI](image)

**Figure 10: Network Training GUI**

The network training GUI (shown in Figure 10) has a look and feel similar to the event extraction GUI. There is a list of extracted event sets on the left of the form. The extracted event set currently selected in this list is displayed in the event name text box and can be also be renamed using this box. The form also displays a count of the total
number of events in the selected extracted event set. The save and load features are the same as for the extracted event GUI. The only difference is that while the extracted event GUI always over-rode the already defined extractions when a new set of extractions were loaded, over-riding is an option in the network training GUI. If the over-ride box is unchecked, then newly loaded events will be appended to the currently defined extracted event list. If an event in the list being loaded has the same name as an already defined event, it will be loaded with a 1 appended to the end of its name. This is necessary to prevent two extracted event sets with the same index. The delete button operates in the same manner as the one on the extracted event GUI. The train network button takes all the extracted event sets in the list and uses them to train the network. The user is prompted for a file to save the trained neural networks as well as the training statistics generated.

8.4.3 Test Network GUI

The test network GUI is not very different, in essence, from the other GUIs. Because the test set tools use events, the event listing, loading, saving, and deletion system is identical to the system used in the network training GUI. The test network GUI also features a neural network load and save feature which is used to load in the particular neural network being tested. The filename for the network currently loaded is displayed as a method of verifying that networks have loaded properly. The test network GUI is displayed below in Figure 11.
The buttons specific to this form are the test random event button and the test events button. The test events button combines all of the extracted events sets in the extracted events list and simulates them using the neural network. It then saves them to a file chosen by the user. Test random event chooses an event at random from the currently selected event set or, if the currently selected event set has zero events, it chooses an event at random from the combined list of all the loaded events. In other words, if LD27L (from Figure 11) was selected, an event from LD27L would be selected unless LD27L had no events, in which case, an event would be selected randomly from the combination of LD27R, LD27L, and LD22L. The randomly selected event is then sent to testNNEoords (see section 8.3) and the resulting output is saved to a file specified by the user.
9 Conclusion

The primary goal of this thesis was to build an optimal neural network, demonstrate that it had been optimized, and provide a generalized optimization process that can be recreated for other airports. These goals have all been achieved. Through an evolving process, a methodology has been developed for extracting runway events from a larger 3-dimensional log file. This process can be used to build training and test sets with events from multiple days. Using these training sets three neural networks were trained. None of the networks performed significantly better than the others. This indicates that enough data was used in all networks to generate an optimized network.

The processes used to train the neural network for Chicago O Hare was then generalized so it could be easily reapplied to another airport. A toolset consisting of all the matlab tools needed to extract events, train a network, and test a network was developed. This toolset allows anyone with a user-level understanding of matlab to develop an optimized neural network. Additionally, GUI forms were designed to provide a more interactive interface.

The trained networks were compared to other networks that were specialized to specific situations. The general network was compared to networks that had different neural networks for landing and takeoffs. It was also compared to specialized networks for each runway group. In both cases, it was determined that takeoff/landing status, and runway have minimal or no affect on airplane motion. The one factor that could not be properly investigated was weather. It is likely that the weather affects a plane’s motion when it is near the runways. Windier days should affect the motion of the plane and potentially the manner in which the pilot flies the plane. Using data from different times
of the year and from days with different types of weather to determine the affect of
weather on the plane’s motion could provide some valuable insight. Data from longer
time periods would also be useful for determining the accuracy of the network when
remove from the few day time period currently available. This data would also be useful
in determining how often, if at all, the network should be re-optimized.

Other potential future work involves further analysis of the network within AMASS. Anuja Doshi is currently working on analyzing the neural network’s performance on AMASS alarm events. Currently AMASS uses the 20 second look ahead. The neural network is demonstrably more accurate at the 20 second look ahead level. Future research could determine how much accuracy would be lost by switching to 25 or 30 second look ahead and make a recommendation on whether the decreased accuracy would offset the increased warning time on potential incursion events. The neural network could even be potentially expanded to look ahead farther than 30 seconds.
References


**Appendix A: Extracted Event Counts**

### August 18, 2001: Landings

<table>
<thead>
<tr>
<th></th>
<th>LD04R</th>
<th>LD09R</th>
<th>LD09L</th>
<th>LD14R</th>
<th>LD22R</th>
<th>LD22L</th>
<th>LD27R</th>
<th>LD27L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>081801a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>106</td>
<td>89</td>
<td>227</td>
</tr>
<tr>
<td>081801b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>081801c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>161</td>
<td>49</td>
<td>28</td>
<td>95</td>
<td>381</td>
</tr>
<tr>
<td>081801d</td>
<td>82</td>
<td>88</td>
<td>0</td>
<td>117</td>
<td>190</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td>520</td>
</tr>
<tr>
<td>Total</td>
<td>82</td>
<td>88</td>
<td>0</td>
<td>165</td>
<td>383</td>
<td>92</td>
<td>134</td>
<td>184</td>
<td>1128</td>
</tr>
</tbody>
</table>

### August 18, 2001: TakeOffs

<table>
<thead>
<tr>
<th></th>
<th>TO04L</th>
<th>TO09L</th>
<th>TO14R</th>
<th>TO14L</th>
<th>TO22L</th>
<th>TO27L</th>
<th>TO32R</th>
<th>TO32L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>081801a</td>
<td>61</td>
<td>8</td>
<td>31</td>
<td>410</td>
<td>162</td>
<td>52</td>
<td>328</td>
<td>1079</td>
<td>296</td>
</tr>
<tr>
<td>081801b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>081801c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>163</td>
<td>56</td>
<td>6</td>
<td>115</td>
<td>346</td>
</tr>
<tr>
<td>081801d</td>
<td>27</td>
<td>61</td>
<td>8</td>
<td>17</td>
<td>121</td>
<td>104</td>
<td>14</td>
<td>70</td>
<td>422</td>
</tr>
<tr>
<td>Total</td>
<td>27</td>
<td>61</td>
<td>8</td>
<td>31</td>
<td>410</td>
<td>162</td>
<td>52</td>
<td>328</td>
<td>1079</td>
</tr>
</tbody>
</table>

### August 19, 2001: Landings

<table>
<thead>
<tr>
<th></th>
<th>LD04R</th>
<th>LD09R</th>
<th>LD09L</th>
<th>LD14R</th>
<th>LD22R</th>
<th>LD22L</th>
<th>LD27R</th>
<th>LD27L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>082201a</td>
<td>43</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>74</td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>082201b</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>082201c</td>
<td>179</td>
<td>197</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>380</td>
</tr>
<tr>
<td>082201d</td>
<td>283</td>
<td>253</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>596</td>
</tr>
<tr>
<td>Total</td>
<td>505</td>
<td>493</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>74</td>
<td>1202</td>
</tr>
</tbody>
</table>

### August 19, 2001: TakeOffs

<table>
<thead>
<tr>
<th></th>
<th>TO04L</th>
<th>TO09L</th>
<th>TO14R</th>
<th>TO14L</th>
<th>TO22L</th>
<th>TO27L</th>
<th>TO32R</th>
<th>TO32L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>082201a</td>
<td>30</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>2</td>
<td>29</td>
<td>134</td>
<td>267</td>
</tr>
<tr>
<td>082201b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>082201c</td>
<td>80</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>96</td>
<td>295</td>
<td></td>
</tr>
<tr>
<td>082201d</td>
<td>112</td>
<td>178</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>27</td>
<td>187</td>
<td>508</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>222</td>
<td>342</td>
<td>0</td>
<td>4</td>
<td>24</td>
<td>5</td>
<td>59</td>
<td>420</td>
<td>1076</td>
</tr>
</tbody>
</table>
### August 21, 2001: Landings

<table>
<thead>
<tr>
<th></th>
<th>LD04R</th>
<th>LD09R</th>
<th>LD09L</th>
<th>LD14R</th>
<th>LD22R</th>
<th>LD22L</th>
<th>LD27R</th>
<th>LD27L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>94</td>
<td>121</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>248</td>
</tr>
</tbody>
</table>

### August 21, 2001: TakeOffs

<table>
<thead>
<tr>
<th></th>
<th>TOO4L</th>
<th>TOO9R</th>
<th>TOO9L</th>
<th>TO14R</th>
<th>TO14L</th>
<th>TO22L</th>
<th>TO32R</th>
<th>TO32L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>75</td>
<td>3</td>
<td>62</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>36</td>
<td>89</td>
<td>290</td>
</tr>
</tbody>
</table>

### August 22, 2001: Landings

<table>
<thead>
<tr>
<th></th>
<th>LD04R</th>
<th>LD09R</th>
<th>LD09L</th>
<th>LD14R</th>
<th>LD22R</th>
<th>LD22L</th>
<th>LD27R</th>
<th>LD27L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>082201a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>129</td>
<td>123</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>252</td>
</tr>
<tr>
<td>082201b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>082201c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>43</td>
<td>109</td>
<td>5</td>
<td>0</td>
<td>87</td>
<td>244</td>
</tr>
<tr>
<td>082201d</td>
<td>18</td>
<td>47</td>
<td>0</td>
<td>79</td>
<td>189</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>426</td>
</tr>
<tr>
<td>Total</td>
<td>18</td>
<td>47</td>
<td>0</td>
<td>293</td>
<td>421</td>
<td>5</td>
<td>0</td>
<td>180</td>
<td>964</td>
</tr>
</tbody>
</table>

### August 22, 2001: TakeOffs

<table>
<thead>
<tr>
<th></th>
<th>TOO4L</th>
<th>TOO9L</th>
<th>TO14R</th>
<th>TO14L</th>
<th>TO22L</th>
<th>TO27L</th>
<th>TO32R</th>
<th>TO32L</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>082201a</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>65</td>
<td>132</td>
<td>82</td>
<td>0</td>
<td>0</td>
<td>280</td>
</tr>
<tr>
<td>082201b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>082201c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>183</td>
<td>28</td>
<td>13</td>
<td>40</td>
<td>264</td>
</tr>
<tr>
<td>082201d</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>13</td>
<td>174</td>
<td>26</td>
<td>27</td>
<td>133</td>
<td>380</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>78</td>
<td>489</td>
<td>145</td>
<td>40</td>
<td>173</td>
<td>933</td>
</tr>
</tbody>
</table>
Appendix B: Trained Neural Networks

Neural Network 1 (trained on 08/21/01)

ASDNet

NN_BEGIN 3
NEURON 0 Input1 0.0
NEURON 0 Input2 0.0
NEURON 0 Input3 0.0
NEURON 0 Input4 0.0
NEURON 0 Input5 0.0
NEURON 0 Input6 0.0
NEURON 0 Input7 0.0
NEURON 0 Input8 0.0
NEURON 0 Input9 0.0
NEURON 0 Input10 0.0
NEURON 1 Hidden1 -6.056885028337
NEURON 1 Hidden2 8.760005138618
NEURON 1 Hidden3 2.516401548708
NEURON 1 Hidden4 0.832005689747
NEURON 1 Hidden5 -24.824157565520
NEURON 1 Hidden6 -1.437000499753
NEURON 1 Hidden7 2.403886799846
NEURON 1 Hidden8 -2.739024223796
NEURON 1 Hidden9 6.274495148707
NEURON 1 Hidden10 10.379975947663
NEURON 2 Output1 5.681530206801
NEURON 2 Output2 -6.711752969959
NEURON 2 Output3 4.966071093801
NEURON 2 Output4 4.26341853984
NEURON 2 Output5 31.244608595786
NEURON 2 Output6 69.226749447897
EDGE Input1 Hidden1 -0.56805015723
EDGE Input1 Hidden2 2.797361099792
EDGE Input1 Hidden3 2.362434724470
EDGE Input1 Hidden4 2.421796328568
EDGE Input1 Hidden5 -4.319317078937
EDGE Input1 Hidden6 -1.758479482123
EDGE Input1 Hidden7 2.807578733529
EDGE Input1 Hidden8 -1.716263098546
EDGE Input1 Hidden9 0.545863734635
EDGE Input1 Hidden10 1.105405066109
EDGE Input2 Hidden1 -0.681445079937
EDGE Input2 Hidden2 1.385538162919
EDGE Input2 Hidden3 1.674417446049

86
| EDGE Input2 Hidden4 | 0.840557037942 |
| EDGE Input2 Hidden5 | -2.80006375659 |
| EDGE Input2 Hidden6 | -0.992849687366 |
| EDGE Input2 Hidden7 | 1.368896646563 |
| EDGE Input2 Hidden8 | -1.583215825257 |
| EDGE Input2 Hidden9 | 0.308459333021 |
| EDGE Input2 Hidden10 | 0.537951844391 |
| EDGE Input3 Hidden1 | 0.315596164562 |
| EDGE Input3 Hidden2 | 0.777727955948 |
| EDGE Input3 Hidden3 | 1.145668159728 |
| EDGE Input3 Hidden4 | 0.857292997119 |
| EDGE Input3 Hidden5 | -1.178657181388 |
| EDGE Input3 Hidden6 | -0.320476463215 |
| EDGE Input3 Hidden7 | 0.606134322249 |
| EDGE Input3 Hidden8 | -1.208527626908 |
| EDGE Input3 Hidden9 | 0.344745463919 |
| EDGE Input3 Hidden10 | 0.857468604187 |
| EDGE Input4 Hidden1 | -0.382630041373 |
| EDGE Input4 Hidden2 | 0.777951359330 |
| EDGE Input4 Hidden3 | 1.064870046414 |
| EDGE Input4 Hidden4 | 0.682394719947 |
| EDGE Input4 Hidden5 | -1.435915513965 |
| EDGE Input4 Hidden6 | -0.665623241549 |
| EDGE Input4 Hidden7 | 0.664381965443 |
| EDGE Input4 Hidden8 | -0.470371090735 |
| EDGE Input4 Hidden9 | -0.369630934258 |
| EDGE Input4 Hidden10 | -0.602584853674 |
| EDGE Input5 Hidden1 | 0.411713169678 |
| EDGE Input5 Hidden2 | 0.018223799157 |
| EDGE Input5 Hidden3 | -0.939362769576 |
| EDGE Input5 Hidden4 | 0.474408378653 |
| EDGE Input5 Hidden5 | 2.696849986271 |
| EDGE Input5 Hidden6 | -0.22763751505 |
| EDGE Input5 Hidden7 | -0.678445488471 |
| EDGE Input5 Hidden8 | -0.169093329977 |
| EDGE Input5 Hidden9 | 0.066874093227 |
| EDGE Input5 Hidden10 | 0.517808262665 |
| EDGE Input6 Hidden1 | 1.109164181900 |
| EDGE Input6 Hidden2 | 0.606280189450 |
| EDGE Input6 Hidden3 | 0.194697770516 |
| EDGE Input6 Hidden4 | 0.926110903201 |
| EDGE Input6 Hidden5 | -1.100556259396 |
| EDGE Input6 Hidden6 | 0.988881633594 |
| EDGE Input6 Hidden7 | 1.164346888026 |
| EDGE Input6 Hidden8 | -1.218946168410 |
| EDGE Input6 Hidden9 | -1.001287865545 |
EDGE Input6 Hidden10 -1.052360671349
EDGE Input7 Hidden1 0.832478215237
EDGE Input7 Hidden2 -1.232325588096
EDGE Input7 Hidden3 0.458447272277
EDGE Input7 Hidden4 0.077365916468
EDGE Input7 Hidden5 0.545618175205
EDGE Input7 Hidden6 0.105788192787
EDGE Input7 Hidden7 0.584698144045
EDGE Input7 Hidden8 0.295385631379
EDGE Input7 Hidden9 -0.442718722937
EDGE Input7 Hidden10 0.137078253720
EDGE Input8 Hidden1 -0.302141236984
EDGE Input8 Hidden2 -0.956821959221
EDGE Input8 Hidden3 0.487751474726
EDGE Input8 Hidden4 0.656523954256
EDGE Input8 Hidden5 1.810414016993
EDGE Input8 Hidden6 -0.167177006979
EDGE Input8 Hidden7 0.321080927545
EDGE Input8 Hidden8 0.199464917843
EDGE Input8 Hidden9 -0.788542495980
EDGE Input8 Hidden10 -0.038020504200
EDGE Input9 Hidden1 0.966019405817
EDGE Input9 Hidden2 0.259770150443
EDGE Input9 Hidden3 -0.433662615400
EDGE Input9 Hidden4 0.561320076124
EDGE Input9 Hidden5 2.328285433636
EDGE Input9 Hidden6 0.297188967357
EDGE Input9 Hidden7 0.737684281952
EDGE Input9 Hidden8 -0.292135105208
EDGE Input9 Hidden9 -0.873397012951
EDGE Input9 Hidden10 -1.171406139783
EDGE Input10 Hidden1 -0.561790844810
EDGE Input10 Hidden2 -1.054816606991
EDGE Input10 Hidden3 -0.988813548228
EDGE Input10 Hidden4 0.410959562332
EDGE Input10 Hidden5 2.085091860974
EDGE Input10 Hidden6 0.827695732484
EDGE Input10 Hidden7 -2.652998897324
EDGE Input10 Hidden8 0.010657655743
EDGE Input10 Hidden9 -1.087719551164
EDGE Input10 Hidden10 0.002562137465
EDGE Hidden1 Output1 -0.248479886502
EDGE Hidden1 Output2 -1.011539788099
EDGE Hidden1 Output3 0.371056528508
EDGE Hidden1 Output4 -0.534631033120
EDGE Hidden1 Output5 -0.049871481343
<table>
<thead>
<tr>
<th>EDGE Hidden1 Output 6</th>
<th>-0.402485988574</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGE Hidden2 Output 1</td>
<td>0.430454787955</td>
</tr>
<tr>
<td>EDGE Hidden2 Output 2</td>
<td>0.001132293180</td>
</tr>
<tr>
<td>EDGE Hidden2 Output 3</td>
<td>-0.569709779598</td>
</tr>
<tr>
<td>EDGE Hidden2 Output 4</td>
<td>0.252402268666</td>
</tr>
<tr>
<td>EDGE Hidden2 Output 5</td>
<td>1.159579816436</td>
</tr>
<tr>
<td>EDGE Hidden2 Output 6</td>
<td>0.995808754250</td>
</tr>
<tr>
<td>EDGE Hidden3 Output 1</td>
<td>0.620923058695</td>
</tr>
<tr>
<td>EDGE Hidden3 Output 2</td>
<td>0.890222714268</td>
</tr>
<tr>
<td>EDGE Hidden3 Output 3</td>
<td>-0.416864013534</td>
</tr>
<tr>
<td>EDGE Hidden3 Output 4</td>
<td>1.090372108771</td>
</tr>
<tr>
<td>EDGE Hidden3 Output 5</td>
<td>1.057916060940</td>
</tr>
<tr>
<td>EDGE Hidden3 Output 6</td>
<td>1.404213551292</td>
</tr>
<tr>
<td>EDGE Hidden4 Output 1</td>
<td>-0.689657389993</td>
</tr>
<tr>
<td>EDGE Hidden4 Output 2</td>
<td>0.050283143252</td>
</tr>
<tr>
<td>EDGE Hidden4 Output 3</td>
<td>0.990365960289</td>
</tr>
<tr>
<td>EDGE Hidden4 Output 4</td>
<td>0.821593645117</td>
</tr>
<tr>
<td>EDGE Hidden4 Output 5</td>
<td>-0.460727052189</td>
</tr>
<tr>
<td>EDGE Hidden4 Output 6</td>
<td>0.840956749455</td>
</tr>
<tr>
<td>EDGE Hidden5 Output 1</td>
<td>0.277248218904</td>
</tr>
<tr>
<td>EDGE Hidden5 Output 2</td>
<td>-0.079383208999</td>
</tr>
<tr>
<td>EDGE Hidden5 Output 3</td>
<td>-1.153273619482</td>
</tr>
<tr>
<td>EDGE Hidden5 Output 4</td>
<td>-1.369004524062</td>
</tr>
<tr>
<td>EDGE Hidden5 Output 5</td>
<td>-2.029357278362</td>
</tr>
<tr>
<td>EDGE Hidden5 Output 6</td>
<td>-2.539001776192</td>
</tr>
<tr>
<td>EDGE Hidden6 Output 1</td>
<td>-0.922129181783</td>
</tr>
<tr>
<td>EDGE Hidden6 Output 2</td>
<td>0.320769256653</td>
</tr>
<tr>
<td>EDGE Hidden6 Output 3</td>
<td>-3.409261058620</td>
</tr>
<tr>
<td>EDGE Hidden6 Output 4</td>
<td>-1.133918151700</td>
</tr>
<tr>
<td>EDGE Hidden6 Output 5</td>
<td>-1.688499113016</td>
</tr>
<tr>
<td>EDGE Hidden6 Output 6</td>
<td>-1.832279919800</td>
</tr>
<tr>
<td>EDGE Hidden7 Output 1</td>
<td>-0.084689313466</td>
</tr>
<tr>
<td>EDGE Hidden7 Output 2</td>
<td>0.599307054784</td>
</tr>
<tr>
<td>EDGE Hidden7 Output 3</td>
<td>-0.088856348039</td>
</tr>
<tr>
<td>EDGE Hidden7 Output 4</td>
<td>0.181287447089</td>
</tr>
<tr>
<td>EDGE Hidden7 Output 5</td>
<td>0.635364464903</td>
</tr>
<tr>
<td>EDGE Hidden7 Output 6</td>
<td>0.450124713708</td>
</tr>
<tr>
<td>EDGE Hidden8 Output 1</td>
<td>-0.774942581616</td>
</tr>
<tr>
<td>EDGE Hidden8 Output 2</td>
<td>-0.736968809200</td>
</tr>
<tr>
<td>EDGE Hidden8 Output 3</td>
<td>-0.510585165694</td>
</tr>
<tr>
<td>EDGE Hidden8 Output 4</td>
<td>-0.422289170600</td>
</tr>
<tr>
<td>EDGE Hidden8 Output 5</td>
<td>-0.470517014807</td>
</tr>
<tr>
<td>EDGE Hidden8 Output 6</td>
<td>-0.402394051679</td>
</tr>
<tr>
<td>EDGE Hidden9 Output 1</td>
<td>-0.070907163194</td>
</tr>
<tr>
<td>EDGE Hidden9 Output 2</td>
<td>0.528033627996</td>
</tr>
<tr>
<td>EDGE Hidden9 Output 3</td>
<td>0.658199161307</td>
</tr>
</tbody>
</table>

89
EDGE Hidden9 Output4  1.147613818823
EDGE Hidden9 Output5  -0.267693748596
EDGE Hidden9 Output6  2.125968318204
EDGE Hidden10 Output1 -0.135188655065
EDGE Hidden10 Output2  0.946314354247
EDGE Hidden10 Output3  0.649592913891
EDGE Hidden10 Output4  2.324616761367
EDGE Hidden10 Output5  4.625746979269
EDGE Hidden10 Output6  4.472113508300
INPUT Input1 "DeltaInput 1"
INPUT Input2 "DeltaInput 2"
INPUT Input3 "DeltaInput 3"
INPUT Input4 "DeltaInput 4"
INPUT Input5 "DeltaInput 5"
INPUT Input6 "DeltaInput 6"
INPUT Input7 "DeltaInput 7"
INPUT Input8 "DeltaInput 8"
INPUT Input9 "DeltaInput 9"
INPUT Input10 "DeltaInput 10"
OUTPUT Output1 "Future Delta Distance 1"
OUTPUT Output2 "Future Delta Distance 2"
OUTPUT Output3 "Future Delta Distance 3"
OUTPUT Output4 "Future Delta Distance 4"
OUTPUT Output5 "Future Delta Distance 5"
OUTPUT Output6 "Future Delta Distance 6"

NN_END

ASRnet

NN_BEGIN 3
NEURON 0 Input1 0.0
NEURON 0 Input2 0.0
NEURON 0 Input3 0.0
NEURON 0 Input4 0.0
NEURON 0 Input5 0.0
NEURON 0 Input6 0.0
NEURON 0 Input7 0.0
NEURON 0 Input8 0.0
NEURON 0 Input9 0.0
NEURON 0 Input10 0.0
NEURON 1 Hidden1  -17.115244841726
NEURON 1 Hidden2   4.265436024286
NEURON 1 Hidden3  10.836839417243
NEURON 1 Hidden4  22.008104114427
NEURON 1 Hidden5  11.080019929008
NEURON 1 Hidden6  24.959270459364

90
| EDGE Input4 Hidden7 | 0.118991746970 |
| EDGE Input4 Hidden8 | 0.298275878912 |
| EDGE Input4 Hidden9 | 1.003193044815 |
| EDGE Input4 Hidden10 | 0.200304651043 |
| EDGE Input5 Hidden1 | 0.149127644817 |
| EDGE Input5 Hidden2 | 0.368474920769 |
| EDGE Input5 Hidden3 | 0.533933576939 |
| EDGE Input5 Hidden4 | 0.970952766041 |
| EDGE Input5 Hidden5 | -0.255491212254 |
| EDGE Input5 Hidden6 | 0.341419840199 |
| EDGE Input5 Hidden7 | -0.57813636430 |
| EDGE Input5 Hidden8 | 0.145086240835 |
| EDGE Input5 Hidden9 | 0.937750403361 |
| EDGE Input5 Hidden10 | 0.090426981779 |
| EDGE Input6 Hidden1 | -0.171690311186 |
| EDGE Input6 Hidden2 | 0.553090586299 |
| EDGE Input6 Hidden3 | 0.478521619185 |
| EDGE Input6 Hidden4 | 0.081296000145 |
| EDGE Input6 Hidden5 | 0.44219449177 |
| EDGE Input6 Hidden6 | 0.914014738414 |
| EDGE Input6 Hidden7 | -0.33753038056 |
| EDGE Input6 Hidden8 | -0.371835303924 |
| EDGE Input6 Hidden9 | -0.104016020509 |
| EDGE Input6 Hidden10 | -0.874934980988 |
| EDGE Input7 Hidden1 | -0.908250293542 |
| EDGE Input7 Hidden2 | -0.394718697017 |
| EDGE Input7 Hidden3 | 0.255667667095 |
| EDGE Input7 Hidden4 | 0.703569124663 |
| EDGE Input7 Hidden5 | 0.253474768528 |
| EDGE Input7 Hidden6 | 0.985638413237 |
| EDGE Input7 Hidden7 | 0.799145073131 |
| EDGE Input7 Hidden8 | -1.055133788738 |
| EDGE Input7 Hidden9 | 0.022049469804 |
| EDGE Input7 Hidden10 | -1.037309597553 |
| EDGE Input8 Hidden1 | -0.299146572202 |
| EDGE Input8 Hidden2 | 0.393991465953 |
| EDGE Input8 Hidden3 | 0.657533951630 |
| EDGE Input8 Hidden4 | -0.084397733954 |
| EDGE Input8 Hidden5 | 0.650319722429 |
| EDGE Input8 Hidden6 | 0.678504136348 |
| EDGE Input8 Hidden7 | -0.764720030968 |
| EDGE Input8 Hidden8 | 0.207461974824 |
| EDGE Input8 Hidden9 | 0.173905284710 |
| EDGE Input8 Hidden10 | -0.276585046454 |
| EDGE Input9 Hidden1 | -0.471227720476 |
| EDGE Input9 Hidden2 | 0.629060283225 |
| Input | Hidden1 Output1 | Hidden1 Output2 | Hidden1 Output3 | Hidden1 Output4 | Hidden1 Output5 | Hidden1 Output6 | Hidden2 Output1 | Hidden2 Output2 | Hidden2 Output3 | Hidden2 Output4 | Hidden2 Output5 | Hidden3 Output1 | Hidden3 Output2 | Hidden3 Output3 | Hidden3 Output4 | Hidden3 Output5 | Hidden3 Output6 | Hidden3 Output7 | Hidden3 Output8 | Hidden3 Output9 | Hidden3 Output10 | Hidden4 Output1 | Hidden4 Output2 | Hidden4 Output3 | Hidden4 Output4 | Hidden4 Output5 | Hidden4 Output6 | Hidden4 Output7 | Hidden4 Output8 | Hidden4 Output9 | Hidden5 Output1 | Hidden5 Output2 | Hidden5 Output3 | Hidden5 Output4 | Hidden5 Output5 | Hidden5 Output6 | Hidden5 Output7 | Hidden5 Output8 | Hidden5 Output9 | Hidden5 Output10 |
|-------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 9     | 0.123103470633 | 0.366084369289 | 0.445108732518 | -0.158627005440 | -0.536432326480 | -0.338328419225 | 0.941328357862 | -0.456739330267 | 0.747582851201 | 2.164709022367 | 0.162544771646 | 2.269720716938 | 0.106122295195 | 0.022394053647 | 1.120069978571 | 1.145529444264 | 1.590481277120 | 1.362565692035 | -0.702401195501 | -0.455103343922 | 0.267263679485 | 0.792634122037 | 0.389709578093 | 1.638906476208 | 0.83212658121 | 1.249771924365 | -0.456358988484 | 1.293178289892 |
EDGE Hidden5 Output5 0.199770209348
EDGE Hidden5 Output6 0.894792168463
EDGE Hidden6 Output1 0.493649081826
EDGE Hidden6 Output2 0.807559188258
EDGE Hidden6 Output3 0.497526366171
EDGE Hidden6 Output4 -0.276576997504
EDGE Hidden6 Output5 1.542272601087
EDGE Hidden6 Output6 -0.128192849003
EDGE Hidden7 Output1 0.695816411711
EDGE Hidden7 Output2 -1.012505191211
EDGE Hidden7 Output3 0.016142492467
EDGE Hidden7 Output4 0.841134970730
EDGE Hidden7 Output5 -0.024084694745
EDGE Hidden7 Output6 0.257935308933
EDGE Hidden8 Output1 -0.740013286472
EDGE Hidden8 Output2 -0.513137281772
EDGE Hidden8 Output3 0.136464411518
EDGE Hidden8 Output4 -0.632895756272
EDGE Hidden8 Output5 0.415729723549
EDGE Hidden8 Output6 -0.726907199074
EDGE Hidden9 Output1 1.234617309794
EDGE Hidden9 Output2 0.813839450400
EDGE Hidden9 Output3 0.075641472648
EDGE Hidden9 Output4 0.823053956729
EDGE Hidden9 Output5 0.298296196610
EDGE Hidden9 Output6 0.526034759798
EDGE Hidden10 Output1 0.496428368075
EDGE Hidden10 Output2 -1.382552938196
EDGE Hidden10 Output3 -1.190448220154
EDGE Hidden10 Output4 -0.301911371897
EDGE Hidden10 Output5 -0.239974877408
EDGE Hidden10 Output6 -0.766689464751

INPUT Input1 "DeltaInput 1"
INPUT Input2 "DeltaInput 2"
INPUT Input3 "DeltaInput 3"
INPUT Input4 "DeltaInput 4"
INPUT Input5 "DeltaInput 5"
INPUT Input6 "DeltaInput 6"
INPUT Input7 "DeltaInput 7"
INPUT Input8 "DeltaInput 8"
INPUT Input9 "DeltaInput 9"
INPUT Input10 "DeltaInput 10"

OUTPUT Output1 "Future Delta Distance 1"
OUTPUT Output2 "Future Delta Distance 2"
OUTPUT Output3 "Future Delta Distance 3"
OUTPUT Output4 "Future Delta Distance 4"
OUTPUT Output5 "Future Delta Distance 5"
OUTPUT Output6 "Future Delta Distance 6"
NN_END

**Neural Network 2**

ASDEnet

NN_BEGIN 3
NEURON 0 Input1 0.0
NEURON 0 Input2 0.0
NEURON 0 Input3 0.0
NEURON 0 Input4 0.0
NEURON 0 Input5 0.0
NEURON 0 Input6 0.0
NEURON 0 Input7 0.0
NEURON 0 Input8 0.0
NEURON 0 Input9 0.0
NEURON 0 Input10 0.0
NEURON 1 Hidden1 5.322876229348
NEURON 1 Hidden2 -6.302897525458
NEURON 1 Hidden3 -10.542978328167
NEURON 1 Hidden4 -14.805536415104
NEURON 1 Hidden5 2.561506830261
NEURON 1 Hidden6 2.662212006373
NEURON 1 Hidden7 14.409025553309
NEURON 1 Hidden8 17.752513796280
NEURON 1 Hidden9 -7.733798634021
NEURON 1 Hidden10 14.179780962280
NEURON 2 Output1 1.229383251315
NEURON 2 Output2 8.934821105026
NEURON 2 Output3 12.729539182140
NEURON 2 Output4 -3.612894889065
NEURON 2 Output5 -30.865724258230
NEURON 2 Output6 -54.545019732147

EDGES
EDGE Input1 Hidden1 -2.901490794834
EDGE Input1 Hidden2 0.996376843974
EDGE Input1 Hidden3 3.970348104426
EDGE Input1 Hidden4 2.602258723115
EDGE Input1 Hidden5 -1.132790181567
EDGE Input1 Hidden6 -1.806418681080
EDGE Input1 Hidden7 0.930833307324
EDGE Input1 Hidden8 -0.378925104959
EDGE Input1 Hidden9 1.114928547443
EDGE Input1 Hidden10 -1.682123402884
EDGE Input2 Hidden1 -2.520299067415
EDGE Input2 Hidden2  0.805674007318
EDGE Input2 Hidden3  4.234894388034
EDGE Input2 Hidden4  -0.490045011216
EDGE Input2 Hidden5  -0.544571068192
EDGE Input2 Hidden6  -1.019276457754
EDGE Input2 Hidden7  1.016635737897
EDGE Input2 Hidden8  -0.637878802606
EDGE Input2 Hidden9  0.609021981605
EDGE Input2 Hidden10 -1.187583362178
EDGE Input3 Hidden1  -1.611698504474
EDGE Input3 Hidden2  -0.582030329173
EDGE Input3 Hidden3  1.237981447821
EDGE Input3 Hidden4  0.174199332712
EDGE Input3 Hidden5  -1.001122025827
EDGE Input3 Hidden6  -1.183918650448
EDGE Input3 Hidden7  0.818768515256
EDGE Input3 Hidden8  -0.527043735405
EDGE Input3 Hidden9  -0.570647188511
EDGE Input3 Hidden10 0.229614949106
EDGE Input4 Hidden1  0.008555200437
EDGE Input4 Hidden2  0.611042403280
EDGE Input4 Hidden3  -0.560562436454
EDGE Input4 Hidden4  0.544393555004
EDGE Input4 Hidden5  0.307473277176
EDGE Input4 Hidden6  0.367377450682
EDGE Input4 Hidden7  -0.708698544559
EDGE Input4 Hidden8  -0.669823077995
EDGE Input4 Hidden9  0.702102678466
EDGE Input4 Hidden10 -0.262503243985
EDGE Input5 Hidden1  -0.298986433043
EDGE Input5 Hidden2  -1.675934243054
EDGE Input5 Hidden3  -1.618323643822
EDGE Input5 Hidden4  0.141991838330
EDGE Input5 Hidden5  -0.776007653385
EDGE Input5 Hidden6  -0.638708719253
EDGE Input5 Hidden7  -0.967400865891
EDGE Input5 Hidden8  1.011377226093
EDGE Input5 Hidden9  -0.447830107583
EDGE Input5 Hidden10 -0.878048563297
EDGE Input6 Hidden1  0.427386055319
EDGE Input6 Hidden2  0.410497809838
EDGE Input6 Hidden3  -0.653551801129
EDGE Input6 Hidden4  -0.194989594407
EDGE Input6 Hidden5  0.777641604215
EDGE Input6 Hidden6  0.653216152559
EDGE Input6 Hidden7  -0.441994805241
<table>
<thead>
<tr>
<th>Edge Input6 Hidden8</th>
<th>-0.317196976530</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Input6 Hidden9</td>
<td>0.717338567143</td>
</tr>
<tr>
<td>Edge Input6 Hidden10</td>
<td>0.42758483379</td>
</tr>
<tr>
<td>Edge Input7 Hidden1</td>
<td>0.755594004893</td>
</tr>
<tr>
<td>Edge Input7 Hidden2</td>
<td>0.143605453296</td>
</tr>
<tr>
<td>Edge Input7 Hidden3</td>
<td>0.716436350384</td>
</tr>
<tr>
<td>Edge Input7 Hidden4</td>
<td>-0.700920625756</td>
</tr>
<tr>
<td>Edge Input7 Hidden5</td>
<td>-1.070356368527</td>
</tr>
<tr>
<td>Edge Input7 Hidden6</td>
<td>0.158624680415</td>
</tr>
<tr>
<td>Edge Input7 Hidden7</td>
<td>0.404734590070</td>
</tr>
<tr>
<td>Edge Input7 Hidden8</td>
<td>-0.078947608705</td>
</tr>
<tr>
<td>Edge Input7 Hidden9</td>
<td>-0.516728401054</td>
</tr>
<tr>
<td>Edge Input7 Hidden10</td>
<td>1.267596586282</td>
</tr>
<tr>
<td>Edge Input8 Hidden1</td>
<td>0.681838455180</td>
</tr>
<tr>
<td>Edge Input8 Hidden2</td>
<td>-1.044928288841</td>
</tr>
<tr>
<td>Edge Input8 Hidden3</td>
<td>-0.571555566253</td>
</tr>
<tr>
<td>Edge Input8 Hidden4</td>
<td>-0.423080791032</td>
</tr>
<tr>
<td>Edge Input8 Hidden5</td>
<td>-0.540675017752</td>
</tr>
<tr>
<td>Edge Input8 Hidden6</td>
<td>0.877779791511</td>
</tr>
<tr>
<td>Edge Input8 Hidden7</td>
<td>0.972923694026</td>
</tr>
<tr>
<td>Edge Input8 Hidden8</td>
<td>-0.98106567442</td>
</tr>
<tr>
<td>Edge Input8 Hidden9</td>
<td>-0.700982419867</td>
</tr>
<tr>
<td>Edge Input8 Hidden10</td>
<td>0.707473211403</td>
</tr>
<tr>
<td>Edge Input9 Hidden1</td>
<td>0.268885430284</td>
</tr>
<tr>
<td>Edge Input9 Hidden2</td>
<td>0.286604672429</td>
</tr>
<tr>
<td>Edge Input9 Hidden3</td>
<td>-0.420225417166</td>
</tr>
<tr>
<td>Edge Input9 Hidden4</td>
<td>0.378477165129</td>
</tr>
<tr>
<td>Edge Input9 Hidden5</td>
<td>1.009989727269</td>
</tr>
<tr>
<td>Edge Input9 Hidden6</td>
<td>1.177467609747</td>
</tr>
<tr>
<td>Edge Input9 Hidden7</td>
<td>0.47702453740</td>
</tr>
<tr>
<td>Edge Input9 Hidden8</td>
<td>-0.265060782995</td>
</tr>
<tr>
<td>Edge Input9 Hidden9</td>
<td>-0.408320724401</td>
</tr>
<tr>
<td>Edge Input9 Hidden10</td>
<td>1.291751799921</td>
</tr>
<tr>
<td>Edge Input10 Hidden1</td>
<td>0.625014001212</td>
</tr>
<tr>
<td>Edge Input10 Hidden2</td>
<td>-0.525550469425</td>
</tr>
<tr>
<td>Edge Input10 Hidden3</td>
<td>-0.697428176924</td>
</tr>
<tr>
<td>Edge Input10 Hidden4</td>
<td>-0.985978699444</td>
</tr>
<tr>
<td>Edge Input10 Hidden5</td>
<td>0.977556681119</td>
</tr>
<tr>
<td>Edge Input10 Hidden6</td>
<td>-0.100528227791</td>
</tr>
<tr>
<td>Edge Input10 Hidden7</td>
<td>0.396649303127</td>
</tr>
<tr>
<td>Edge Input10 Hidden8</td>
<td>-0.389606169940</td>
</tr>
<tr>
<td>Edge Input10 Hidden9</td>
<td>0.441817726226</td>
</tr>
<tr>
<td>Edge Input10 Hidden10</td>
<td>1.602917648824</td>
</tr>
<tr>
<td>Edge Hidden1 Output1</td>
<td>-1.160170847301</td>
</tr>
<tr>
<td>Edge Hidden1 Output2</td>
<td>-0.72218002489</td>
</tr>
<tr>
<td>Edge Hidden1 Output3</td>
<td>-0.53473670520</td>
</tr>
<tr>
<td>Layer</td>
<td>Output 1</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>EDGE Hidden1</td>
<td>-0.796635081681</td>
</tr>
<tr>
<td>EDGE Hidden2</td>
<td>1.089727319709</td>
</tr>
<tr>
<td>EDGE Hidden3</td>
<td>0.444574393470</td>
</tr>
<tr>
<td>EDGE Hidden4</td>
<td>2.249340601678</td>
</tr>
<tr>
<td>EDGE Hidden5</td>
<td>0.842008944145</td>
</tr>
<tr>
<td>EDGE Hidden6</td>
<td>-0.731831887429</td>
</tr>
<tr>
<td>EDGE Hidden7</td>
<td>0.495640188661</td>
</tr>
<tr>
<td>EDGE Hidden8</td>
<td>0.377170225462</td>
</tr>
</tbody>
</table>
EDGE Hidden9 Output2  0.335424458503
EDGE Hidden9 Output3  1.358116571415
EDGE Hidden9 Output4  0.772227639404
EDGE Hidden9 Output5  2.076218420485
EDGE Hidden9 Output6  1.305446629809
EDGE Hidden10 Output1  0.366454536992
EDGE Hidden10 Output2  -0.06005273310
EDGE Hidden10 Output3  -0.718671585179
EDGE Hidden10 Output4  -1.678839469645
EDGE Hidden10 Output5  -2.385581180473
EDGE Hidden10 Output6  -3.558784517228

INPUT Input1 "DeltaInput 1"
INPUT Input2 "DeltaInput 2"
INPUT Input3 "DeltaInput 3"
INPUT Input4 "DeltaInput 4"
INPUT Input5 "DeltaInput 5"
INPUT Input6 "DeltaInput 6"
INPUT Input7 "DeltaInput 7"
INPUT Input8 "DeltaInput 8"
INPUT Input9 "DeltaInput 9"
INPUT Input10 "DeltaInput 10"

OUTPUT Output1 "Future Delta Distance 1"
OUTPUT Output2 "Future Delta Distance 2"
OUTPUT Output3 "Future Delta Distance 3"
OUTPUT Output4 "Future Delta Distance 4"
OUTPUT Output5 "Future Delta Distance 5"
OUTPUT Output6 "Future Delta Distance 6"

NN_END

ASRnet

NN_BEGIN 3
NEURON 0 Input1 0.0
NEURON 0 Input2 0.0
NEURON 0 Input3 0.0
NEURON 0 Input4 0.0
NEURON 0 Input5 0.0
NEURON 0 Input6 0.0
NEURON 0 Input7 0.0
NEURON 0 Input8 0.0
NEURON 0 Input9 0.0
NEURON 0 Input10 0.0
NEURON 1 Hidden1  -0.416768854362
NEURON 1 Hidden2  -1.431150113840
NEURON 1 Hidden3  0.343615259886
NEURON 1 Hidden4  9.688197432441
<p>| Neuron 1 Hidden5  | -15.639960797543 |
| Neuron 1 Hidden6  | 1.982487445494  |
| Neuron 1 Hidden7  | -0.845130345736 |
| Neuron 1 Hidden8  | -26.660791807317|
| Neuron 1 Hidden9  | -1.171423085716 |
| Neuron 1 Hidden10 | -30.885324847394|
| Neuron 2 Output1  | 0.961414137845  |
| Neuron 2 Output2  | -0.547159276792 |
| Neuron 2 Output3  | 0.970389817297  |
| Neuron 2 Output4  | 1.327203977976  |
| Neuron 2 Output5  | 13.832096146332 |
| Neuron 2 Output6  | 48.306024688326 |
| Edge Input1 Hidden1 | 0.128628569195 |
| Edge Input1 Hidden2 | 0.141933538270 |
| Edge Input1 Hidden3 | 0.033313753314 |
| Edge Input1 Hidden4 | 0.364181814539 |
| Edge Input1 Hidden5 | -0.803559267155|
| Edge Input1 Hidden6 | 0.751792718786 |
| Edge Input1 Hidden7 | -0.406105435780|
| Edge Input1 Hidden8 | -1.096692770239|
| Edge Input1 Hidden9 | -0.934203911592|
| Edge Input1 Hidden10| -0.631106819353|
| Edge Input2 Hidden1 | -0.676006042141|
| Edge Input2 Hidden2 | -0.728121663327|
| Edge Input2 Hidden3 | 1.076934982899 |
| Edge Input2 Hidden4 | 0.737055077521 |
| Edge Input2 Hidden5 | -0.347811522226 |
| Edge Input2 Hidden6 | -0.507672109865 |
| Edge Input2 Hidden7 | -0.953215599545 |
| Edge Input2 Hidden8 | 0.572171101256 |
| Edge Input2 Hidden9 | -0.221242106007|
| Edge Input2 Hidden10| -0.429916429830|
| Edge Input3 Hidden1 | 0.512348167187 |
| Edge Input3 Hidden2 | -1.120104341054 |
| Edge Input3 Hidden3 | 0.260543845686 |
| Edge Input3 Hidden4 | 1.270555549477 |
| Edge Input3 Hidden5 | -0.912037784451 |
| Edge Input3 Hidden6 | -0.298449955780|
| Edge Input3 Hidden7 | -1.337047206966|
| Edge Input3 Hidden8 | -0.345140925772|
| Edge Input3 Hidden9 | 0.142893001140 |
| Edge Input3 Hidden10| -0.577095090190 |
| Edge Input4 Hidden1 | 0.610854876559 |
| Edge Input4 Hidden2 | -0.800070176197 |
| Edge Input4 Hidden3 | -0.526483816415 |
| Edge Input4 Hidden4 | 0.673823363455 |</p>
<table>
<thead>
<tr>
<th>Input</th>
<th>Hidden1</th>
<th>Hidden2</th>
<th>Hidden3</th>
<th>Hidden4</th>
<th>Hidden5</th>
<th>Hidden6</th>
<th>Hidden7</th>
<th>Hidden8</th>
<th>Hidden9</th>
<th>Hidden10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input4</td>
<td>-0.985936511296</td>
<td>0.705884067479</td>
<td>-1.082542616549</td>
<td>0.332871785070</td>
<td>-0.515414554899</td>
<td>0.497028689264</td>
<td>0.060226139489</td>
<td>-0.880347855206</td>
<td>0.431328989373</td>
<td>0.441303012723</td>
</tr>
<tr>
<td>Input5</td>
<td>0.705884067479</td>
<td>-0.880347855206</td>
<td>0.431328989373</td>
<td>0.441303012723</td>
<td>-0.622518962087</td>
<td>0.004741217934</td>
<td>-1.425040914465</td>
<td>-0.007436631062</td>
<td>0.204199845973</td>
<td>-0.335359065201</td>
</tr>
<tr>
<td>Input6</td>
<td>-0.51166348604</td>
<td>0.015848497377</td>
<td>0.475898344235</td>
<td>1.387929202507</td>
<td>-0.618525021625</td>
<td>-0.387748113708</td>
<td>0.159894046730</td>
<td>-0.663431097140</td>
<td>-0.736041018974</td>
<td>0.702372795961</td>
</tr>
<tr>
<td>Input7</td>
<td>0.167250002657</td>
<td>-0.296521288310</td>
<td>0.276751566903</td>
<td>0.275807871294</td>
<td>-0.336542174014</td>
<td>0.784351950660</td>
<td>-0.715786502566</td>
<td>-0.852843229919</td>
<td>-0.856271789093</td>
<td>0.643184094418</td>
</tr>
<tr>
<td>Input8</td>
<td>0.345251376380</td>
<td>-0.209627081837</td>
<td>0.230918754082</td>
<td>0.948855866495</td>
<td>-1.415743855128</td>
<td>0.403656903767</td>
<td>0.027584697288</td>
<td>-0.862009331260</td>
<td>0.336507772160</td>
<td>-0.570568906001</td>
</tr>
<tr>
<td>Input9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

101
| EDGE Input9 Hidden1 | -1.000472553260 |
| EDGE Input9 Hidden2 | -1.367552994217 |
| EDGE Input9 Hidden3 | 0.321401088460 |
| EDGE Input9 Hidden4 | 0.541107243261 |
| EDGE Input9 Hidden5 | -0.835314210097 |
| EDGE Input9 Hidden6 | 0.909817156570 |
| EDGE Input9 Hidden7 | -0.518854484663 |
| EDGE Input9 Hidden8 | 0.595921155074 |
| EDGE Input9 Hidden9 | 0.957917797143 |
| EDGE Input9 Hidden10 | -0.554917746232 |
| EDGE Input10 Hidden1 | 0.327155304249 |
| EDGE Input10 Hidden2 | -0.182688182651 |
| EDGE Input10 Hidden3 | 0.212488935053 |
| EDGE Input10 Hidden4 | 1.320062280745 |
| EDGE Input10 Hidden5 | -0.380595259687 |
| EDGE Input10 Hidden6 | -0.036391522943 |
| EDGE Input10 Hidden7 | 0.196395437108 |
| EDGE Input10 Hidden8 | -0.764436503104 |
| EDGE Input10 Hidden9 | 0.315783139130 |
| EDGE Input10 Hidden10 | 0.694307756199 |
| EDGE Hidden1 Output1 | -1.239405915975 |
| EDGE Hidden1 Output2 | -0.739696857447 |
| EDGE Hidden1 Output3 | -0.585361500023 |
| EDGE Hidden1 Output4 | -0.986602357748 |
| EDGE Hidden1 Output5 | -1.606591227903 |
| EDGE Hidden1 Output6 | -1.890642130072 |
| EDGE Hidden2 Output1 | 0.955650729793 |
| EDGE Hidden2 Output2 | 0.898679938400 |
| EDGE Hidden2 Output3 | -0.040585394545 |
| EDGE Hidden2 Output4 | 0.947462243640 |
| EDGE Hidden2 Output5 | -0.753182736489 |
| EDGE Hidden2 Output6 | -0.760223060477 |
| EDGE Hidden3 Output1 | -0.628015332114 |
| EDGE Hidden3 Output2 | 0.933328058779 |
| EDGE Hidden3 Output3 | 0.576827139071 |
| EDGE Hidden3 Output4 | 0.469724559577 |
| EDGE Hidden3 Output5 | -0.074072132282 |
| EDGE Hidden3 Output6 | 0.076194149906 |
| EDGE Hidden4 Output1 | 0.758078006910 |
| EDGE Hidden4 Output2 | -0.549391509123 |
| EDGE Hidden4 Output3 | 0.499764042951 |
| EDGE Hidden4 Output4 | 1.331943536133 |
| EDGE Hidden4 Output5 | 0.932119609694 |
| EDGE Hidden4 Output6 | 0.597676192004 |
| EDGE Hidden5 Output1 | 0.232498957264 |
| EDGE Hidden5 Output2 | -0.950010976678 |
| EDGE Hidden5 Output3 | -1.257795773721 |
| EDGE Hidden5 Output4 | -0.827735519390 |
| EDGE Hidden5 Output5 | -0.389838789787 |
| EDGE Hidden5 Output6 | -0.995411400703 |
| EDGE Hidden6 Output1 | 0.227180537023 |
| EDGE Hidden6 Output2 | 0.759134424876 |
| EDGE Hidden6 Output3 | -0.563343481688 |
| EDGE Hidden6 Output4 | 0.703255157590 |
| EDGE Hidden6 Output5 | 0.754295484840 |
| EDGE Hidden6 Output6 | 0.088600198059 |
| EDGE Hidden7 Output1 | -1.212368181252 |
| EDGE Hidden7 Output2 | -1.260372587485 |
| EDGE Hidden7 Output3 | 0.142848981590 |
| EDGE Hidden7 Output4 | -0.699688272558 |
| EDGE Hidden7 Output5 | -0.541001718670 |
| EDGE Hidden7 Output6 | -0.760026053845 |
| EDGE Hidden8 Output1 | -0.045511203467 |
| EDGE Hidden8 Output2 | -0.354528314173 |
| EDGE Hidden8 Output3 | -0.361847812825 |
| EDGE Hidden8 Output4 | 0.329772319438 |
| EDGE Hidden8 Output5 | -0.684452967849 |
| EDGE Hidden8 Output6 | -1.160437440932 |
| EDGE Hidden9 Output1 | 0.280275531412 |
| EDGE Hidden9 Output2 | 0.451637219366 |
| EDGE Hidden9 Output3 | -0.615203555555 |
| EDGE Hidden9 Output4 | -0.931155798735 |
| EDGE Hidden9 Output5 | -1.309621257491 |
| EDGE Hidden9 Output6 | -1.420344275405 |
| EDGE Hidden10 Output1 | 0.220955095856 |
| EDGE Hidden10 Output2 | 0.593583423511 |
| EDGE Hidden10 Output3 | 0.059141474068 |
| EDGE Hidden10 Output4 | -0.518606142444 |
| EDGE Hidden10 Output5 | -0.603934262323 |
| EDGE Hidden10 Output6 | -0.584606569607 |

INPUT Input1 "DeltaInput 1"
INPUT Input2 "DeltaInput 2"
INPUT Input3 "DeltaInput 3"
INPUT Input4 "DeltaInput 4"
INPUT Input5 "DeltaInput 5"
INPUT Input6 "DeltaInput 6"
INPUT Input7 "DeltaInput 7"
INPUT Input8 "DeltaInput 8"
INPUT Input9 "DeltaInput 9"
INPUT Input10 "DeltaInput 10"
OUTPUT Output1 "Future Delta Distance 1"
OUTPUT Output2 "Future Delta Distance 2"
OUTPUT Output3 "Future Delta Distance 3"
OUTPUT Output4 "Future Delta Distance 4"
OUTPUT Output5 "Future Delta Distance 5"
OUTPUT Output6 "Future Delta Distance 6"

Neural Network 3

ASDEnet

NN_BEGIN 3
NEURON 0 Input1 0.0
NEURON 0 Input2 0.0
NEURON 0 Input3 0.0
NEURON 0 Input4 0.0
NEURON 0 Input5 0.0
NEURON 0 Input6 0.0
NEURON 0 Input7 0.0
NEURON 0 Input8 0.0
NEURON 0 Input9 0.0
NEURON 0 Input10 0.0
NEURON 1 Hidden1 -4.347895739047
NEURON 1 Hidden2 -3.147788387223
NEURON 1 Hidden3 -20.246718405560
NEURON 1 Hidden4 -47.238112262304
NEURON 1 Hidden5 13.552952855739
NEURON 1 Hidden6 27.521436903478
NEURON 1 Hidden7 13.152964876562
NEURON 1 Hidden8 -9.659712563765
NEURON 1 Hidden9 9.537411787504
NEURON 1 Hidden10 -6.804958291762
NEURON 2 Output1 1.995685253014
NEURON 2 Output2 12.666853093329
NEURON 2 Output3 -5.720004957781
NEURON 2 Output4 20.855351901883
NEURON 2 Output5 -22.711114971784
NEURON 2 Output6 -38.625348963332
EDGE Input1 Hidden1 0.936818110423
EDGE Input1 Hidden2 2.853653548909
EDGE Input1 Hidden3 3.903653249013
EDGE Input1 Hidden4 3.668390830534
EDGE Input1 Hidden5 -2.859730590814
EDGE Input1 Hidden6 -2.540246509498
EDGE Input1 Hidden7 -1.616230892390
EDGE Input1 Hidden8 -0.760303792374
EDGE Input1 Hidden9 -2.122960595768
<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Input</th>
<th>Hidden1</th>
<th>Hidden2</th>
<th>Hidden3</th>
<th>Hidden4</th>
<th>Hidden5</th>
<th>Hidden6</th>
<th>Hidden7</th>
<th>Hidden8</th>
<th>Hidden9</th>
<th>Hidden10</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGE Input1</td>
<td>0.977208754905</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>0.910658268515</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>1.200651167212</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>2.476711742193</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>0.850206151824</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>-2.033239910368</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>-1.130471913054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>-1.104171616167</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>-0.578104111806</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>-0.552219224534</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input2</td>
<td>0.689877101208</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>0.143495570350</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>0.868594814589</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>0.702453810026</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>1.002090334959</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>-1.006572524557</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>-0.459837699041</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>-1.833043957790</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>-0.540754358801</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>-0.577715817330</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input3</td>
<td>-0.294020660409</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.750226651955</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>1.444153509372</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>1.38592154324</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.287374367104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.695448821784</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.334076438928</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.28716305678</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.566044237373</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>0.430913147566</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input4</td>
<td>-0.131025259162</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.821227597895</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.342731995635</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>0.560922156401</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.900920168543</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>0.267584146496</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.673277729827</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.381954057963</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>0.045018918467</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.022580058701</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input5</td>
<td>-0.013985670699</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6</td>
<td>0.321493414379</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6</td>
<td>0.238843146805</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6</td>
<td>-0.780167310891</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6</td>
<td>-0.745204618903</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6</td>
<td>0.934934068501</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer Combination</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6 Hidden6</td>
<td>1.088136472753</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6 Hidden7</td>
<td>0.275860332829</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6 Hidden8</td>
<td>0.104109801083</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6 Hidden9</td>
<td>-0.925492947030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input6 Hidden10</td>
<td>-0.247901813576</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden1</td>
<td>-0.596181043037</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden2</td>
<td>-1.197810570378</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden3</td>
<td>-0.423834572492</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden4</td>
<td>-0.304380402157</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden5</td>
<td>-0.140450329399</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden6</td>
<td>0.316075263991</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden7</td>
<td>-1.109907712495</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden8</td>
<td>0.847255719768</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden9</td>
<td>1.167982636688</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input7 Hidden10</td>
<td>-0.445796311413</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden1</td>
<td>-0.010051829524</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden2</td>
<td>-0.964768695962</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden3</td>
<td>-1.254624973147</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden4</td>
<td>-1.923383546484</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden5</td>
<td>-0.556792455801</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden6</td>
<td>-0.196481598144</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden7</td>
<td>0.455025681412</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden8</td>
<td>-0.516142185792</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden9</td>
<td>1.918849954213</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input8 Hidden10</td>
<td>-0.185752411516</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden1</td>
<td>-0.397328503357</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden2</td>
<td>-0.615446529074</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden3</td>
<td>-0.685118382198</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden4</td>
<td>-2.082667616791</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden5</td>
<td>0.959122709584</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden6</td>
<td>0.588111842117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden7</td>
<td>0.797231603835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden8</td>
<td>0.157623508488</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden9</td>
<td>0.938267857406</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input9 Hidden10</td>
<td>0.686902109999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden1</td>
<td>-1.115085883106</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden2</td>
<td>-1.169104002727</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden3</td>
<td>0.194582226966</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden4</td>
<td>-1.917284257843</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden5</td>
<td>0.083956266555</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden6</td>
<td>-0.604076666048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden7</td>
<td>0.597843448648</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden8</td>
<td>-0.327196170198</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden9</td>
<td>1.252764636864</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Input10 Hidden10</td>
<td>-0.911760130251</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden1 Output1</td>
<td>0.886510023570</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Extracted Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden1 Output2</td>
<td>1.071754702498</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden1 Output3</td>
<td>0.932528408321</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden1 Output4</td>
<td>2.40357107766</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden1 Output5</td>
<td>1.19749029936</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden1 Output6</td>
<td>2.306374173381</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden2 Output1</td>
<td>0.495461139218</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden2 Output2</td>
<td>-0.011468956731</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden2 Output3</td>
<td>0.811233843037</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden2 Output4</td>
<td>0.085397777370</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden2 Output5</td>
<td>-0.154654145500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden2 Output6</td>
<td>0.511532465542</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden3 Output1</td>
<td>0.127480739068</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden3 Output2</td>
<td>0.193601959412</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden3 Output3</td>
<td>0.295734400764</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden3 Output4</td>
<td>1.430723772216</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden3 Output5</td>
<td>1.746308854964</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden3 Output6</td>
<td>0.772636681497</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden4 Output1</td>
<td>-0.385282193244</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden4 Output2</td>
<td>-0.341448021023</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden4 Output3</td>
<td>-0.112160314455</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden4 Output4</td>
<td>0.142593403439</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden4 Output5</td>
<td>1.289624891813</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden4 Output6</td>
<td>1.478898987135</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden5 Output1</td>
<td>0.423367572874</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden5 Output2</td>
<td>-0.142224450490</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden5 Output3</td>
<td>-1.745369575535</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden5 Output4</td>
<td>-2.397795495840</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden5 Output5</td>
<td>-2.938870843677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden5 Output6</td>
<td>-4.02472974352</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden6 Output1</td>
<td>-0.459943068300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden6 Output2</td>
<td>-0.906200507368</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden6 Output3</td>
<td>0.117874677587</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden6 Output4</td>
<td>-0.821794679262</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden6 Output5</td>
<td>0.574644707272</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden6 Output6</td>
<td>-1.327438640868</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden7 Output1</td>
<td>-0.821792342670</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden7 Output2</td>
<td>-0.920177742530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden7 Output3</td>
<td>-0.739450757010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden7 Output4</td>
<td>0.043350491974</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden7 Output5</td>
<td>-0.587287012382</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden7 Output6</td>
<td>-0.269834742881</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden8 Output1</td>
<td>0.099003913894</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden8 Output2</td>
<td>-0.660087750178</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden8 Output3</td>
<td>0.021564586148</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden8 Output4</td>
<td>1.041995985709</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE Hidden8 Output5</td>
<td>-0.71403000664</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

107
EDGE Hidden8 Output6 0.067045299038
EDGE Hidden9 Output1 0.064300222731
EDGE Hidden9 Output2 -0.700284628191
EDGE Hidden9 Output3 -0.660262892667
EDGE Hidden9 Output4 -0.901329673431
EDGE Hidden9 Output5 -1.197860273547
EDGE Hidden9 Output6 -1.597020179527
EDGE Hidden10 Output1 0.050129140837
EDGE Hidden10 Output2 0.561518019953
EDGE Hidden10 Output3 1.011058128213
EDGE Hidden10 Output4 1.260268177178
EDGE Hidden10 Output5 2.912964960150
EDGE Hidden10 Output6 3.572975729812
INPUT Input1 "DeltaInput 1"
INPUT Input2 "DeltaInput 2"
INPUT Input3 "DeltaInput 3"
INPUT Input4 "DeltaInput 4"
INPUT Input5 "DeltaInput 5"
INPUT Input6 "DeltaInput 6"
INPUT Input7 "DeltaInput 7"
INPUT Input8 "DeltaInput 8"
INPUT Input9 "DeltaInput 9"
INPUT Input10 "DeltaInput 10"
OUTPUT Output1 "Future Delta Distance 1"
OUTPUT Output2 "Future Delta Distance 2"
OUTPUT Output3 "Future Delta Distance 3"
OUTPUT Output4 "Future Delta Distance 4"
OUTPUT Output5 "Future Delta Distance 5"
OUTPUT Output6 "Future Delta Distance 6"
NN_END

ASRnet

NN_BEGIN 3
NEURON 0 Input1 0.0
NEURON 0 Input2 0.0
NEURON 0 Input3 0.0
NEURON 0 Input4 0.0
NEURON 0 Input5 0.0
NEURON 0 Input6 0.0
NEURON 0 Input7 0.0
NEURON 0 Input8 0.0
NEURON 0 Input9 0.0
NEURON 0 Input10 0.0
NEURON 1 Hidden1 1.936649323063
NEURON 1 Hidden2 -0.027400154943
<table>
<thead>
<tr>
<th>Connection</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGE Input4 Hidden3</td>
<td>0.617034426743</td>
</tr>
<tr>
<td>EDGE Input4 Hidden4</td>
<td>-0.270467935321</td>
</tr>
<tr>
<td>EDGE Input4 Hidden5</td>
<td>0.658383738898</td>
</tr>
<tr>
<td>EDGE Input4 Hidden6</td>
<td>0.559454297356</td>
</tr>
<tr>
<td>EDGE Input4 Hidden7</td>
<td>0.485326138228</td>
</tr>
<tr>
<td>EDGE Input4 Hidden8</td>
<td>0.219867465158</td>
</tr>
<tr>
<td>EDGE Input4 Hidden9</td>
<td>-1.222634653311</td>
</tr>
<tr>
<td>EDGE Input4 Hidden10</td>
<td>-0.178835819181</td>
</tr>
<tr>
<td>EDGE Input5 Hidden1</td>
<td>1.344246557527</td>
</tr>
<tr>
<td>EDGE Input5 Hidden2</td>
<td>-0.071230117145</td>
</tr>
<tr>
<td>EDGE Input5 Hidden3</td>
<td>-0.426615619443</td>
</tr>
<tr>
<td>EDGE Input5 Hidden4</td>
<td>-0.889935416657</td>
</tr>
<tr>
<td>EDGE Input5 Hidden5</td>
<td>0.217123023915</td>
</tr>
<tr>
<td>EDGE Input5 Hidden6</td>
<td>-0.984970094181</td>
</tr>
<tr>
<td>EDGE Input5 Hidden7</td>
<td>0.348669994506</td>
</tr>
<tr>
<td>EDGE Input5 Hidden8</td>
<td>0.61930564395</td>
</tr>
<tr>
<td>EDGE Input5 Hidden9</td>
<td>-0.880936352795</td>
</tr>
<tr>
<td>EDGE Input5 Hidden10</td>
<td>0.485553020335</td>
</tr>
<tr>
<td>EDGE Input6 Hidden1</td>
<td>1.193116474915</td>
</tr>
<tr>
<td>EDGE Input6 Hidden2</td>
<td>0.273681306837</td>
</tr>
<tr>
<td>EDGE Input6 Hidden3</td>
<td>0.321467868343</td>
</tr>
<tr>
<td>EDGE Input6 Hidden4</td>
<td>-0.816741936839</td>
</tr>
<tr>
<td>EDGE Input6 Hidden5</td>
<td>-0.640700399192</td>
</tr>
<tr>
<td>EDGE Input6 Hidden6</td>
<td>-0.867332037156</td>
</tr>
<tr>
<td>EDGE Input6 Hidden7</td>
<td>0.530382094408</td>
</tr>
<tr>
<td>EDGE Input6 Hidden8</td>
<td>0.909912927465</td>
</tr>
<tr>
<td>EDGE Input6 Hidden9</td>
<td>-0.439044248070</td>
</tr>
<tr>
<td>EDGE Input6 Hidden10</td>
<td>-0.215553240744</td>
</tr>
<tr>
<td>EDGE Input7 Hidden1</td>
<td>1.315438063770</td>
</tr>
<tr>
<td>EDGE Input7 Hidden2</td>
<td>0.581760253323</td>
</tr>
<tr>
<td>EDGE Input7 Hidden3</td>
<td>0.980323331167</td>
</tr>
<tr>
<td>EDGE Input7 Hidden4</td>
<td>-0.252709673692</td>
</tr>
<tr>
<td>EDGE Input7 Hidden5</td>
<td>-0.506513764916</td>
</tr>
<tr>
<td>EDGE Input7 Hidden6</td>
<td>-0.211261984867</td>
</tr>
<tr>
<td>EDGE Input7 Hidden7</td>
<td>-0.106081215854</td>
</tr>
<tr>
<td>EDGE Input7 Hidden8</td>
<td>0.105383212063</td>
</tr>
<tr>
<td>EDGE Input7 Hidden9</td>
<td>-0.362754809039</td>
</tr>
<tr>
<td>EDGE Input7 Hidden10</td>
<td>-0.686959501577</td>
</tr>
<tr>
<td>EDGE Input8 Hidden1</td>
<td>0.509491191944</td>
</tr>
<tr>
<td>EDGE Input8 Hidden2</td>
<td>0.984522023423</td>
</tr>
<tr>
<td>EDGE Input8 Hidden3</td>
<td>0.341985987845</td>
</tr>
<tr>
<td>EDGE Input8 Hidden4</td>
<td>-0.278718185222</td>
</tr>
<tr>
<td>EDGE Input8 Hidden5</td>
<td>-0.741362638121</td>
</tr>
<tr>
<td>EDGE Input8 Hidden6</td>
<td>-0.728342486320</td>
</tr>
<tr>
<td>EDGE Input8 Hidden7</td>
<td>-0.136593281436</td>
</tr>
<tr>
<td>EDGE Input8 Hidden8</td>
<td>-0.118578138563</td>
</tr>
<tr>
<td>EDGE Input8 Hidden9</td>
<td>-0.409015262281</td>
</tr>
<tr>
<td>EDGE Input8 Hidden10</td>
<td>-0.516343902328</td>
</tr>
<tr>
<td>EDGE Input9 Hidden1</td>
<td>0.567492315858</td>
</tr>
<tr>
<td>EDGE Input9 Hidden2</td>
<td>1.235833633573</td>
</tr>
<tr>
<td>EDGE Input9 Hidden3</td>
<td>0.867146044313</td>
</tr>
<tr>
<td>EDGE Input9 Hidden4</td>
<td>-0.863716201017</td>
</tr>
<tr>
<td>EDGE Input9 Hidden5</td>
<td>-1.288339140615</td>
</tr>
<tr>
<td>EDGE Input9 Hidden6</td>
<td>-0.058333271688</td>
</tr>
<tr>
<td>EDGE Input9 Hidden7</td>
<td>-0.494229644167</td>
</tr>
<tr>
<td>EDGE Input9 Hidden8</td>
<td>-0.637983428465</td>
</tr>
<tr>
<td>EDGE Input9 Hidden9</td>
<td>-0.344652101735</td>
</tr>
<tr>
<td>EDGE Input10 Hidden10</td>
<td>-0.512678457082</td>
</tr>
<tr>
<td>EDGE Input10 Hidden1</td>
<td>-0.509601889990</td>
</tr>
<tr>
<td>EDGE Input10 Hidden2</td>
<td>-0.424857749098</td>
</tr>
<tr>
<td>EDGE Input10 Hidden3</td>
<td>-0.251231735671</td>
</tr>
<tr>
<td>EDGE Input10 Hidden4</td>
<td>-0.392357539485</td>
</tr>
<tr>
<td>EDGE Input10 Hidden5</td>
<td>0.304898721954</td>
</tr>
<tr>
<td>EDGE Input10 Hidden6</td>
<td>-0.825179861379</td>
</tr>
<tr>
<td>EDGE Input10 Hidden7</td>
<td>0.773785114752</td>
</tr>
<tr>
<td>EDGE Input10 Hidden8</td>
<td>0.650405629596</td>
</tr>
<tr>
<td>EDGE Input10 Hidden9</td>
<td>-0.018122577619</td>
</tr>
<tr>
<td>EDGE Input10 Hidden10</td>
<td>-0.674036234969</td>
</tr>
<tr>
<td>EDGE Hidden1 Output1</td>
<td>0.104500487299</td>
</tr>
<tr>
<td>EDGE Hidden1 Output2</td>
<td>-0.284018303651</td>
</tr>
<tr>
<td>EDGE Hidden1 Output3</td>
<td>0.980767502670</td>
</tr>
<tr>
<td>EDGE Hidden1 Output4</td>
<td>0.620943338317</td>
</tr>
<tr>
<td>EDGE Hidden1 Output5</td>
<td>0.353516093629</td>
</tr>
<tr>
<td>EDGE Hidden1 Output6</td>
<td>0.932845277941</td>
</tr>
<tr>
<td>EDGE Hidden2 Output1</td>
<td>0.760224653807</td>
</tr>
<tr>
<td>EDGE Hidden2 Output2</td>
<td>0.055048509970</td>
</tr>
<tr>
<td>EDGE Hidden2 Output3</td>
<td>-0.443480672538</td>
</tr>
<tr>
<td>EDGE Hidden2 Output4</td>
<td>-0.046604708088</td>
</tr>
<tr>
<td>EDGE Hidden2 Output5</td>
<td>-0.142317043851</td>
</tr>
<tr>
<td>EDGE Hidden2 Output6</td>
<td>0.263065199858</td>
</tr>
<tr>
<td>EDGE Hidden3 Output1</td>
<td>0.364501822010</td>
</tr>
<tr>
<td>EDGE Hidden3 Output2</td>
<td>1.057406323592</td>
</tr>
<tr>
<td>EDGE Hidden3 Output3</td>
<td>0.476952063191</td>
</tr>
<tr>
<td>EDGE Hidden3 Output4</td>
<td>0.512825774717</td>
</tr>
<tr>
<td>EDGE Hidden3 Output5</td>
<td>0.267125407212</td>
</tr>
<tr>
<td>EDGE Hidden3 Output6</td>
<td>0.210175849247</td>
</tr>
<tr>
<td>EDGE Hidden4 Output1</td>
<td>0.168433695674</td>
</tr>
<tr>
<td>EDGE Hidden4 Output2</td>
<td>-0.152880554592</td>
</tr>
<tr>
<td>EDGE Hidden4 Output3</td>
<td>0.314236241022</td>
</tr>
<tr>
<td>EDGE Hidden4 Output4</td>
<td>-0.825478460360</td>
</tr>
<tr>
<td>EDGE Hidden4 Output5</td>
<td>-0.053363491571</td>
</tr>
<tr>
<td>EDGE Hidden4 Output6</td>
<td>-0.965151469575</td>
</tr>
<tr>
<td>EDGE Hidden5 Output1</td>
<td>0.401252631069</td>
</tr>
<tr>
<td>EDGE Hidden5 Output2</td>
<td>0.673845461667</td>
</tr>
<tr>
<td>EDGE Hidden5 Output3</td>
<td>-0.676896279171</td>
</tr>
<tr>
<td>EDGE Hidden5 Output4</td>
<td>-0.409481133538</td>
</tr>
<tr>
<td>EDGE Hidden5 Output5</td>
<td>-0.990141681070</td>
</tr>
<tr>
<td>EDGE Hidden5 Output6</td>
<td>-0.386515892678</td>
</tr>
<tr>
<td>EDGE Hidden6 Output1</td>
<td>0.020999899727</td>
</tr>
<tr>
<td>EDGE Hidden6 Output2</td>
<td>-1.144730928641</td>
</tr>
<tr>
<td>EDGE Hidden6 Output3</td>
<td>-1.025346970057</td>
</tr>
<tr>
<td>EDGE Hidden6 Output4</td>
<td>0.167214681327</td>
</tr>
<tr>
<td>EDGE Hidden6 Output5</td>
<td>-1.189642061992</td>
</tr>
<tr>
<td>EDGE Hidden6 Output6</td>
<td>-0.367058732925</td>
</tr>
<tr>
<td>EDGE Hidden7 Output1</td>
<td>0.576884833959</td>
</tr>
<tr>
<td>EDGE Hidden7 Output2</td>
<td>-0.801626465311</td>
</tr>
<tr>
<td>EDGE Hidden7 Output3</td>
<td>0.679120541300</td>
</tr>
<tr>
<td>EDGE Hidden7 Output4</td>
<td>-0.262572031469</td>
</tr>
<tr>
<td>EDGE Hidden7 Output5</td>
<td>0.779504232989</td>
</tr>
<tr>
<td>EDGE Hidden7 Output6</td>
<td>1.040425838769</td>
</tr>
<tr>
<td>EDGE Hidden8 Output1</td>
<td>0.446561249137</td>
</tr>
<tr>
<td>EDGE Hidden8 Output2</td>
<td>0.277957201203</td>
</tr>
<tr>
<td>EDGE Hidden8 Output3</td>
<td>-0.492442768093</td>
</tr>
<tr>
<td>EDGE Hidden8 Output4</td>
<td>0.665031428587</td>
</tr>
<tr>
<td>EDGE Hidden8 Output5</td>
<td>-0.309641925706</td>
</tr>
<tr>
<td>EDGE Hidden8 Output6</td>
<td>-0.208505661337</td>
</tr>
<tr>
<td>EDGE Hidden9 Output1</td>
<td>-0.168694210883</td>
</tr>
<tr>
<td>EDGE Hidden9 Output2</td>
<td>-0.878146266638</td>
</tr>
<tr>
<td>EDGE Hidden9 Output3</td>
<td>0.207995555377</td>
</tr>
<tr>
<td>EDGE Hidden9 Output4</td>
<td>-0.756707986407</td>
</tr>
<tr>
<td>EDGE Hidden9 Output5</td>
<td>-1.133795738934</td>
</tr>
<tr>
<td>EDGE Hidden9 Output6</td>
<td>-0.882829502670</td>
</tr>
<tr>
<td>EDGE Hidden10 Output1</td>
<td>-0.376350431377</td>
</tr>
<tr>
<td>EDGE Hidden10 Output2</td>
<td>-0.230306262922</td>
</tr>
<tr>
<td>EDGE Hidden10 Output3</td>
<td>-0.250547719270</td>
</tr>
<tr>
<td>EDGE Hidden10 Output4</td>
<td>-0.409320834736</td>
</tr>
<tr>
<td>EDGE Hidden10 Output5</td>
<td>-0.478471781240</td>
</tr>
<tr>
<td>EDGE Hidden10 Output6</td>
<td>-0.723574537582</td>
</tr>
</tbody>
</table>

**INPUT**
- Input1 "DeltaInput 1"
- Input2 "DeltaInput 2"
- Input3 "DeltaInput 3"
- Input4 "DeltaInput 4"
- Input5 "DeltaInput 5"
- Input6 "DeltaInput 6"
- Input7 "DeltaInput 7"
- Input8 "DeltaInput 8"
- Input9 "DeltaInput 9"
- Input10 "DeltaInput 10"
OUTPUT Output1 "Future Delta Distance 1"
OUTPUT Output2 "Future Delta Distance 2"
OUTPUT Output3 "Future Delta Distance 3"
OUTPUT Output4 "Future Delta Distance 4"
OUTPUT Output5 "Future Delta Distance 5"
OUTPUT Output6 "Future Delta Distance 6"
NN_END
Appendix C: Data Extraction Toolset Matlab Code

ExtractRunwayEvents

function tset=ExtractRunwayEvents(Log3D, pos, angle, land)

Log3D is the 3D matrix generated from AMASS log files. pos is 2x2 matrix giving x and y bounds
angle is the angle of the runway in degrees. land is 0 for takeoff event or 1 for landing event

Extract events from the 3D matrix, of airplanes landing/taking of on an arbitrary runway
Selection of events is based on coordinates of the airplane, as well as speed and direction
This function calls FindEvent and ExtractEvent to obtain coordinates from the 3D matrix. Each event comprises 100 samples and is stored in tset matrix separated by a row of zeros
Based on scripts for individual runways created by Rafael Palacios and modified by Yoshi Nakanishi

x = 180*atan(Log3D(:,:,4)./Log3D(:,:,5))/pi + 180*(Log3D(:,:,5)<0) + 360*(Log3D(:,:,5)>30 & Log3D(:,:,4)<0);

z=Log3D(:,:,2)<pos(2,1) & Log3D(:,:,2)>pos(1,1) & Log3D(:,:,3)>pos(1,2) & Log3D(:,:,3)<pos(2,2) ...
   & x(:,:,)<(angle+5) & x(:,:,)>(angle-5) & ((Log3D(:,:,4).^2+Log3D(:,:,5).^2).^0.5)>140;

tset1=[];
tset=[];
N=size(Log3D,1);

for k=1:256
i=1;
while (i<N)
   while (i<N & z(i,k)==0) i=i+1; end

   if z(i,k)==1
      if land == 0
         event=ExtractEvent(Log3D,i,k,60,10);
      else

      end
end

114
event=ExtractEvent(Log3D,i,k,0,99); %extract 100 samples
end
tset1=[tset1; [0, 0, 0]; event];
end

while (i<N & z(i,k)==1) i=i+100; end
end
fprintf('-->pid=%d\n',k);
end

%filter out events with low number of points
if not (isempty(tset1))
tset1 = tset1(find(not(isnan(tset1(:,3)))), :);

temp = find(tset1(:,3)==0);
for k=1:size(temp,1)
  if k == size(temp,1)
    if (size(tset1, 1) - temp(k)) > 40
      tset=[tset, tset1(temp(k):end, :)];
    end
  else
    if (temp(k+1) - temp(k)) > 40
      tset=[tset, tset1(temp(k):temp(k+1)-1, :)];
    end
  end
end
end
function XY=ExtractEvent(Log3D, record, pid, sec_before, sec_after)

% function XY=ExtractEvent(Log3D, record, pid, sec_before, sec_after)
% Extract X,Y coordinates of one event identified by record number and PID.
% This function extracts data from sec_before seconds before record to sec_after
% seconds after record.
% The event usually occurs when the aircraft is being tracked by ASDE. If no ASDE
% data is available while going backwards in time, this function looks for data
% coming from ASR for the same aircraft.
%
% function XY=ExtractEvent(Log3D, record, pid, sec_before, sec_after)
% Log3D: 3D matrix of the log file, as obtained with bigmat.m
% record, pid: Record number and PID to identify the event. Obtained with
% function FindEvent.m
% sec_before: Number of samples before the event we want to extract (if available)
% sec_after: Number of samples after the event we want to extract (if available)
% XY: returned matrix. First column=X coordinates, Second column=Y coordinates.
% The number of rows will be sec_before+1+sec_after, provided enough data is
% available
% Rafael Palacios (Feb/2003,Aug/2003)
%
% Zero Padding at the first and last rows to avoid overflow
Log3D(1,:,:)=zeros(size(Log3D(1,:,:)));
Log3D(end,:,:)=zeros(size(Log3D(1,:,:)));

fprintf('Begin PID=%d, TID=%d\n',pid,Log3D(record,pid,6));

% Create empty output matrix.
XY=zeros(sec_before+1+sec_after,3)*NaN;

% Current position
XY(sec_before+1,1:2)=Log3D(record,pid,2:3);
XY(sec_before+1,3)=Log3D(record,pid,1);

% ASDE tracking?
status=Log3D(record,pid,1);
if status==0
    fprintf('ERROR: PID=%d, TID=%d at record=%d has target_type=0 (No Data)\n', ...

116
pid, Log3D(record, pid, 6), record);
end
if status ~= 1
    fprintf('WARNING: PID=%d, TID=%d at record=%d has target_type=1 (Not ASDE)\n', ...
    pid, Log3D(record, pid, 6), record);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract data backwards in time
i = 1;
while Log3D(record-i, pid, 1) == status & i <= sec_before
    XY(sec_before+1-i, 1:2) = Log3D(record-i, pid, 2:3);
    XY(sec_before+1-i, 3) = Log3D(record-i, pid, 1);
    i = i + 1;
end
if i <= sec_before % we have run out of data
    if status == 1 % we were at ASDE, so lets find the closest aircraft at ASR
        rec = record-i;
        x = Log3D(rec+1, pid, 2); % current x
        y = Log3D(rec+1, pid, 3); % current y
        min_distance = 9999999;
        for k = find(Log3D(rec, :, 1) == 2)
            distance = sqrt((Log3D(rec, k, 2)-x)^2 + (Log3D(rec, k, 3)-y)^2);
            if distance < min_distance
                min_distance = distance;
                min_pid = k;
            end
        end
    end
end

% Checking results
if min_distance > 300 % minimum distance 300 ft
    fprintf('WARNING backwards: We were at ASDE, but the nearest aircraft is %f feet away\n', min_distance);
    fprintf('WARNING backwards: the beginning of the returned vector is not valid (NaN values)\n');
else
    % New PID, continue tracking backwards
    fprintf('Going backward we changed from PID=%d, TID=%d --> PID=%d, TID=%d, record=%d\n', ...
        pid, Log3D(record-i+1, pid, 6), min_pid, Log3D(record-i, min_pid, 6), rec);
    while Log3D(record-i, min_pid, 1) == 2 & i <= sec_before
        XY(sec_before+1-i, 1:2) = Log3D(record-i, min_pid, 2:3);
        XY(sec_before+1-i, 3) = Log3D(record-i, min_pid, 1);
        i = i + 1;
end

117
end
if i<=sec_before
    fprintf('WARNING backwards: We changed into ASR then we ran out of
data.
');
    fprintf('WARNING backwards: the beginning of the returned vector is not valid
(NaN values)
');
end
end
else
    fprintf('WARNING backwards: We were at ASR and we ran out of data.
');
    fprintf('WARNING backwards: the beginning of the returned vector is not valid (NaN
values)
');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract data forward in time
i=1;
if status==1 % we are at ASDE
    while Log3D(record+i,pid,1)==1 & i<=sec_after % while ASDE
        XY(sec_before+1+i,1:2)=Log3D(record+i,pid,2:3);
        XY(sec_before+1+i,3)=Log3D(record+i,pid,1);
        i=i+1;
    end
    if i<=sec_after
        fprintf('WARNING forward: We ran out of data while going forward in time.
');
        fprintf('WARNING forward: We were at ASDE from the beginning.
');
        fprintf('WARNING forward: the last part of the returned vector is not valid (NaN
values)
');
    end
else % we are at ASR, so in every step we need to check if ASDE data is available
    while Log3D(record+i,pid,1)==2 & i<=sec_after % while ASR
        XY(sec_before+1+i,1:2)=Log3D(record+i,pid,2:3);
        XY(sec_before+1+i,3)=Log3D(record+i,pid,1);
        i=i+1;
    end
    % Look for ASDE data
    rec=record+i;
    x=Log3D(rec,pid,2); % current x
    y=Log3D(rec,pid,3); % current y
    min_distance=9999999;
    for k=find(Log3D(rec,:,1)==1)
        distance=sqrt((Log3D(rec,k,2)-x)^2 + (Log3D(rec,k,3)-y)^2);
        if distance<min_distance
            min_distance=distance;
            min_pid=k;
        end
    end
end
end

% minimum distance to change from ASR to ASPE
if min_distance<300
    if min_distance<300
        fprintf('Going forward we changed from PID=%d, TID=%d --> PID=%d, TID=%d, record=%d\n', ...
            pid, Log3D(record+i-1,pid,6), min_pid, Log3D(record+i,min_pid,6), rec);
    end
    while Log3D(record+i,min_pid,1)==1 & i<=sec_after
        XY(sec_before+1+i,1:2)=Log3D(record+i,min_pid,2:3);
        XY(sec_before+1+i,3)=Log3D(record+i,min_pid,1);
        i=i+1;
    end
    if i<=sec_after
        fprintf('WARNING forward: We ran out of data while going forward in time.\n');
        fprintf('WARNING forward: We began at ASR, then changed to ASDE, then lost ...
        signal.\n');
        fprintf('WARNING forward: The last part of the returned vector is not valid (NaN values)\n');
        return;
    end
end
end
end
fprintf('End PID=%d\n', pid);
function found=FindEvent(Log3D, Xcoord, Ycoord)

% Finds an event within a 3D Log.
% Search is based on X and Y coordinates. This function returns an empty matrix
% if the event is not found, a 1-row matrix containing PID and index if the
% event is found once, or a n-row matrix if the event is found n times.
%
% function found=FindEvent(Log3D, Xcoord, Ycoord)
%
% Log3D: 3D matrix (Nx256x6) obtained from the Log file using bigmat.m
% Xcoord: X coordinate of the airplane
% Ycoord: Y coordinate of the airplane
% found: matrix containing index, PID, TID of the event.
% To find the time t of the event use: t(found(1,1))
% where t is the time vector returned by bigmat.m
%
% Warning. PIDs in the log file do not match PIDs within AMASS.
% Warning. Only TIDs are the same.
%
% Rafael Palacios Feb/2003
%
% k=1; found=[];
for pid=1:256
    ind=find(Log3D(:,pid,2)==Xcoord);
    ind2=find(Log3D(ind,pid,3)==Ycoord);
    for i=1:length(ind2)
        found(k,:)=ind(ind2(i)),pid,Log3D(ind(ind2(i)),pid,6));
        k=k+1;
    end
end
randomEvent

function randEvent = randomEvent(ExtractedData)

% Inputs - ExtractedData:
%      A list of takeoff an/or landing events extracted from a runway.
% Outputs - randEvent: matrix containing index, PID, TID of a random event
% chosen from ExtractedData.
% Implementation: Randomly chooses one event from a extractedData
% Developed by Geoff Cooney January 2004

temp = find(ExtractedData(:, 1)==0 & ExtractedData(:, 2)==0 & ExtractedData(:, 3)==0)
randInt = ceil((size(temp)-1)*rand(1));
randEvent = ExtractedData(temp(randInt)+1:temp(randInt+1)-1,:);
function count = ExtractCount(events)
    count = 0;
    if not(isempty(events))
        count = size(find(events(:,1)==0 & events(:,2) == 0 & events(:,3) == 0));
        if (events(1, :) == [0 0 0] & events(end, :) == [0 0 0])
            count = count -1;
        end
    end
end
Appendix D: Network Training Toolset Matlab Code

TrainNN

function net = trainNN(events)

% Created by Geoff Cooney January 2004
% Based strongly on training scripts created by Yoshi Nakanishi

[ASDETrin_all, ASDETrout_all, ASDEVin_all, ASDEVout_all, ASDETein_all, ASDETeout_all,...
     ASRTrin_all, ASRTrout_all, ASRVin_all, ASRVout_all, ASRTein_all, ASRTeout_all] = getTrainOuts(events);

NUM_HIDDEN_NODES = 10;
MAX_EPOCHS = 1000000;

fprintf('Training ASDE NN
');
ASDEnet =
    newff(minmax(ASDETrin_all),[NUM_HIDDEN_NODES,size(ASDETrout_all,1)],{'purelin','purelin'},'trainrp');
ASDEnet.trainParam.show=10;
ASDEnet.trainParam.epochs=MAX_EPOCHS;
V.P=ASDEVin_all;
V.T=ASDEVout_all;
tic;
ASDEnet = train(ASDEnet, ASDETrin_all, ASDETrout_all,[],[],V);
elapsedTime.trainingASDE=toc;

fprintf('Testing ASDE NN
');
stats.TestASDE=testNN(ASDEnet, ASDETein_all, ASDETeout_all);
stats.TestASDE.points = size(ASDETein_all, 2);
stats.TrainASDE=testNN(ASDEnet, ASDETrin_all, ASDETrout_all);
stats.TrainASDE.points = size(ASDETrin_all, 2);
stats.ValASDE=testNN(ASDEnet, ASDEVin_all, ASDEVout_all);
stats.ValASDE.points = size(ASDEVin_all, 2);

fprintf('Training ASR NN
');
ASRnet =
    newff(minmax(ASRTrin_all),[NUM_HIDDEN_NODES,size(ASRTrout_all,1)],{'purelin','purelin'},'trainrp');
ASRnet.trainParam.show=10;
ASRnet.trainParam.epochs=MAX_EPOCHS;
V.P=ASRVin_all;
V.T=ASRVout_all;
tic;
ASRnet = train(ASRnet, ASRTrin_all, ASRTrout_all,[],[],V);
elapsedTime.trainingASR = toc;

fprintf('Testing ASR NN\n');
stats.TestASR = testNN(ASRnet, ASRTein_all, ASRTeout_all);
stats.TestASR.points = size(ASRTein_all, 2);
stats.TrainASR = testNN(ASRnet, ASRTrin_all, ASRTrout_all);
stats.TrainASR.points = size(ASRTrin_all, 2);
stats.ValASR = testNN(ASRnet, ASRVin_all, ASRVout_all);
stats.ValASR.points = size(ASRVin_all, 2);

save 'C:\FAAproject\traindata-22b.mat' stats ASRnet ASDEnet;
net.stats = stats;
net.ASRnet = ASRnet;
net.ASDEnet = ASDEnet;
getTrainOuts

% Developed by Yoshi Nakanishi

function [ASDETrin, ASDETrout, ASDEVin, ASDEVout, ASDETein, ASDETeout,...
    ASRTrin, ASRTTrout, ASRVin, ASRVout, ASRTein, ASRTTeout] =
    getTrainOuts(XY)

if isempty(XY)
    ASRTrin = [];  
    ASRTrout = [];  
    ASRVin = [];  
    ASRVout = [];  
    ASRTein = [];  
    ASRTTeout = [];
else
    [ASDEinX, ASDEoutX, ASDEinY, ASDEoutY, ASRinX, ASRoutX, ASRinY,  
    ASRoutY, ind] = buildTS26(XY);
    [ASDETrin, ASDETrout, ASDEVin, ASDEVout, ASDETein,  
    ASDETeout] = randomOutputs(ASDEinX, ASDEoutX, ASDEinY, ASDEoutY);
    [ASRTrin, ASRTrout, ASRVin, ASRVout, ASRTein,  
    ASRTTeout] = randomOutputs(ASRinX, ASRoutX, ASRinY, ASRoutY);
end

    Trin = [];  
    Trout = [];  
    Vin = [];  
    Vout = [];  
    Tein = [];  
    Teout = [];
    a = size(inX, 2);
    if (a < 10)
        break;
    else
        x = rand(a, 1);
        [x, i] = sort(x);
    end
    TrinX = inX(:, i(1:a/2));
    TroutX = outX(:, i(1:a/2));
    VinX = inX(:, i(a/2:3*a/4:4*a));
    VoutX = outX(:, i(a/2:3*a/4:4*a));
    TeinX = inX(:, i(3*a/4:end));
    TeoutX = outX(:, i(3*a/4:end));
a=size(inY, 1);
x=rand(a, 1);
[x,i]=sort(x);
TrinY=inY(:,i(1:a/2));
TroutY=outY(:,i(1:a/2));
VinY=inY(:,i(a/2:3*a/4));
VoutY=outY(:,i(a/2:3*a/4));
TeinY=inY(:,i(3*a/4:end));
TeoutY=outY(:,i(3*a/4:end));

Trin=[TrinX TrinY];
Trout=[TroutX TroutY];
Vin=[VinX VinY];
Vout=[VoutX VoutY];
Tein=[TeinX TeinY];
Teout=[TeoutX TeoutY];

end
BuildTS26

function [ASDEinX, ASDEoutX, ASDEinY, ASDEoutY,...

% function [inX outX inY outY]=buildTS(coords)
%
% Function to build training sets from a vector of coordinates. Each event
% is separated by a row of [0, 0]
% coords is a two-column matrix.
% inX, inY are matrices of inputs to the NN (size 10xN)
% outX, outY are matrices of expected outputs (size 6xN)
%
% One can join X and Y information in a single training set by doing:
% in=[inX,inY];
% out=[outX,outY];
%
% Rafael Palacios (Sep/2003)
%
ind=[];
inX=[];
outX=[];
inY=[];
outY=[];

ok=find(isfinite(coords(:,1)));
coords=coords(ok,:);
if coords(1,:)==[0,0,0], coords=[[0,0,0];coords]; end
if coords(end,:)==[0,0,0], coords=[coords;[0,0,0]]; end

% check and remove NaN
counter=1;
tempo=zeros(size(coords, 1), size(coords, 2));
for i=1:size(coords,1)
    if(isnan(coords(i,1)) | isnan(coords(i,2)))
        continue;
    end
    tempo(counter,: ) = coords(i,:);
    counter = counter + 1;
end
coords = tempo(1:counter-1,:);

k=find(coords(:,1)==0 & coords(:,2)==0);
for i=1:length(k)-1
    if (k(i+1)-k(i) > 30)
        fprintf('Event %d
',i);
        eventX=coords(k(i)+1:k(i+1)-1,1);
        eventY=coords(k(i)+1:k(i+1)-1,2);
        indicator=coords(k(i)+1:k(i+1)-1,3);
        temp=indicator(2:end-30);
        % X coordinates first
        a=eventX;
        delta=a(2:end-30)-a(1:end-31);
        inputX=toeplitz(ones(10,1)*delta(1), delta); %builds input matrix flowing numbers
        outputX=...
            (a(2+5:end-30+5)-a(2:end-30))';
            (a(2+10:end-30+10)-a(2:end-30))';
            (a(2+15:end-30+15)-a(2:end-30))';
            (a(2+20:end-30+20)-a(2:end-30))';
            (a(2+25:end-30+25)-a(2:end-30))';
            (a(2+30:end-30+30)-a(2:end-30))' ];
        % Y coordinates now
        a=eventY;
        delta=a(2:end-30)-a(1:end-31);
        inputY=toeplitz(ones(10,1)*delta(1), delta); %builds input matrix flowing numbers
        outputY=...
            (a(2+5:end-30+5)-a(2:end-30))';
            (a(2+10:end-30+10)-a(2:end-30))';
            (a(2+15:end-30+15)-a(2:end-30))';
            (a(2+20:end-30+20)-a(2:end-30))';
            (a(2+25:end-30+25)-a(2:end-30))';
            (a(2+30:end-30+30)-a(2:end-30))' ];
        ind=[ind;temp];
        inX=[inX,inputX];
        outX=[outX,outputX];
        inY=[inY,inputY];
        outY=[outY,outputY];
    end
end

% Separate ASDE and ASR
% Allocate
ASDEinX = zeros(10,size(inX,2));
ASDEoutX = zeros(6,size(inX,2));
ASDEinY = zeros(10,size(inX,2));
ASDEoutY = zeros(6,size(inX,2));
umASDE = 0;
$\text{ASRinX} = \text{zeros}(10,\text{size(inX,2)});$  
$\text{ASRoutX} = \text{zeros}(6,\text{size(inX,2)});$  
$\text{ASRinY} = \text{zeros}(10,\text{size(inX,2)});$  
$\text{ASRoutY} = \text{zeros}(6,\text{size(inX,2)});$  
$\text{numASR} = 0;$  

for $i = 1: \text{size(ind,1)}$  
  if $(\text{ind}(i) == 0)$  
    fprintf('Error: target_type=0 (No Data)\n');  
  elseif $(\text{ind}(i) == 1)$  
    $\text{ASDEinX(:, numASDE+1)} = \text{inX(:,i)};$  
    $\text{ASDEoutX(:, numASDE+1)} = \text{outX(:,i)};$  
    $\text{ASDEinY(:, numASDE+1)} = \text{inY(:,i)};$  
    $\text{ASDEoutY(:, numASDE+1)} = \text{outY(:,i)};$  
    $\text{numASDE} = \text{numASDE+1};$  
  elseif $(\text{ind}(i) == 2)$  
    $\text{ASRinX(:, numASR+1)} = \text{inX(:,i)};$  
    $\text{ASRoutX(:, numASR+1)} = \text{outX(:,i)};$  
    $\text{ASRinY(:, numASR+1)} = \text{inY(:,i)};$  
    $\text{ASRoutY(:, numASR+1)} = \text{outY(:,i)};$  
    $\text{numASR} = \text{numASR+1};$  
  else  
    fprintf('Error: Something is really wrong\n');  
  end  
end  

% Truncate to size  
$\text{ASDEinX} = \text{ASDEinX(:,1:}\text{numASDE});$  
$\text{ASDEoutX} = \text{ASDEoutX(:,1:}\text{numASDE});$  
$\text{ASDEinY} = \text{ASDEinY(:,1:}\text{numASDE});$  
$\text{ASDEoutY} = \text{ASDEoutY(:,1:}\text{numASDE});$  

$\text{ASRinX} = \text{ASRinX(:,1:}\text{numASR});$  
$\text{ASRoutX} = \text{ASRoutX(:,1:}\text{numASR});$  
$\text{ASRinY} = \text{ASRinY(:,1:}\text{numASR});$  
$\text{ASRoutY} = \text{ASRoutY(:,1:}\text{numASR});$  
fprintf('\n');  
fprintf('ASDE: \d sets\n',numASDE);  
fprintf('ASR: \d sets\n',numASR);
combineEvents

% Developed by Geoff Cooney January 2004.
% takes a struct containing event lists and combines all the event lists into one single
% event list

function eventList = combineEvents(events)

eventList = [];

names = fieldnames(events);
for i = 1:size(names)
    eventList = [eventList; getfield(events, names{i})];
end
Appendix E: Network Testing Tools Matlab Code

TestNN

function [stats] = testNN(net, input, expected)

% function [stats] = testNN(net, input, expected)
%
% Gives statistics Absolute Mean Error (ae) and
% Square Root Absolute Mean Square Error (mse)
%
% net: trained neural network
% input: input to net with dimension consistent with net
% expected: expected output with dimension consistent with net
%
% edited from ynaka’s testNN_6out.m
% ntt: 5/8/03
%

simout = sim(net, input);

stats.ae=[];
stats.mse=[];
stats.max = [];
nTrain= size(expected,2);
stats.points = nTrain;

for i=1:size(expected,1);

    ae_test = sum(abs(expected(i,:)-simout(i,:)))/nTrain;

    stats.ae = [stats.ae;ae_test];

    mse_test = (sum((expected(i,:)-simout(i,:)).^2)/nTrain)^0.5;

    stats.mse = [stats.mse;mse_test];

    max_test = max(expected(i,:)-simout(i,:));

    stats.max = [stats.max; max_test];

end
TestNNCords

function [xstats, ystats, FutureDx, FutureDy, xOut, yOut, indicator] = testNNCords(ASDEnet, ASRnet, coords)

% Obtain TestSets

[xIn, yIn, xOut, yOut, indicator] = getTestSet(coords);
% [xIn, xOut, yIn, yOut, indicator] = buildTS(coords)

xIn = xIn';
yIn = yIn';
xOut = xOut';
yOut = yOut';

% %%%%%%%%%%%%%%%%%%%%%
% Simulate Data %
% %%%%%%%%%%%%%%%%%%%%%

FutureDx = zeros(size(xOut,1), size(xOut,2));
FutureDy = zeros(size(yOut,1), size(yOut,2));

for i = 1:size(indicator,1);
    if (indicator(i) == 1)
        FutureDx(:,i) = sim(ASDEnet, xIn(:,i));
        FutureDy(:,i) = sim(ASDEnet, yIn(:,i));
    elseif (indicator(i) == 2)
        FutureDx(:,i) = sim(ASRnet, xIn(:,i));
        FutureDy(:,i) = sim(ASRnet, yIn(:,i));
    else
        fprintf('Error: No clue what Radar\n');
    end
end

% %%%%%%%%%%%%%%%%%%%%%
% Test Data %
% %%%%%%%%%%%%%%%%%%%%%

long = size(xOut,2);
long2 = size(FutureDx,2);
xstats=[];
ystats=[];
for i = 0:5
    xmae = sum(abs(xOut(i+1,1:long-i*5)-FutureDx(i+1,1:long-i*5)))/((long-i*5);  
exstats(i+1,1) = xmae;
    xmse = (sum((xOut(i+1,1:long-i*5)-FutureDx(i+1,1:long-i*5)).^2)/((long-i*5)^0.5;
    xstats(i+1,2) = xmse;
    ymae = sum(abs(yOut(i+1,1:long-i*5)-FutureDy(i+1,1:long-i*5)))/((long-i*5);
    ystats(i+1,1) = ymae;
    ymse = (sum((yOut(i+1,1:long-i*5)-FutureDy(i+1,1:long-i*5)).^2)/((long-i*5)^0.5;
    ystats(i+1,2) = ymse;
    figure;
    plot(1:long-i*5,xOut(i+1,1:long-i*5), 1:long2, FutureDx(i+1,:));
    figure;
    plot(1:long-i*5,yOut(i+1,1:long-i*5), 1:long2, FutureDy(i+1,:));
end

% END

function [inX, outX, inY, outY, ind] = buildTS(coords)

% function [inX outX inY outY]=buildTS(coords)
% Function to build training sets from a vector of coordinates. Each event
% is separated by a row of [0, 0]
% coords   is a two-column matrix.
% inX, inY are matrices of inputs to the NN (size 10xN)
% outX, outY are matrices of expected outputs (size 6xN)
% One can join X and Y information in a single training set by doing:
% in=[inX,inY];
% out=[outX,outY];
% Rafael Palacios (Sep/2003)
% ind=[];
inX=[];
outX=[];
inY=[];
outY=[];

ok=find(isfinite(coords(:,1)));
coords=coords(ok,:);
if coords(1,:)~=[0,0,0], coords=[0,0,0]; coords; end
if coords(end,:)~=[0,0,0], coords=[coords,[0,0,0]]; end

k=find(coords(:,1)==0 & coords(:,2)==0);
for i=1:length(k)-1
    fprintf('Event %d
', i);
    eventX=coords(k(i)+1:k(i+1)-1,1);
    eventY=coords(k(i)+1:k(i+1)-1,2);
    indicator=coords(k(i)+1:k(i+1)-1,3);
    temp=indicator(2:end-30);
    % Indicator
    % X coordinates first
    a=eventX;
    delta=a(2:end-30)-a(1:end-31);
    inputX=toeplitz(ones(10,1)*delta(1), delta); % builds input matrix flowing numbers
    outputX=[...
    (a(2+5:end-30+5)-a(2:end-30))';
    (a(2+10:end-30+10)-a(2:end-30))';
    (a(2+15:end-30+15)-a(2:end-30))';
    (a(2+20:end-30+20)-a(2:end-30))';
    (a(2+25:end-30+25)-a(2:end-30))';
    (a(2+30:end-30+30)-a(2:end-30))' ];
    % Y coordinates now
    a=eventY;
    delta=a(2:end-30)-a(1:end-31);
    inputY=toeplitz(ones(10,1)*delta(1), delta); % builds input matrix flowing numbers
    outputY=[...
    (a(2+5:end-30+5)-a(2:end-30))';
    (a(2+10:end-30+10)-a(2:end-30))';
    (a(2+15:end-30+15)-a(2:end-30))';
    (a(2+20:end-30+20)-a(2:end-30))';
    (a(2+25:end-30+25)-a(2:end-30))';
    (a(2+30:end-30+30)-a(2:end-30))' ];
    ind=[ind; temp];
    inX=[inX,inputX];
    outX=[outX,outputX];
    inY=[inY,inputY];
    outY=[outY,outputY];
end

function [xInput, yInput, xOutput, yOutput, ind] = getTestSet(xyCoord)

    % xCoord=xyCoord(:,1);
    % yCoord=xyCoord(:,2);
% check and remove NaN
counter=1;
tempo=zeros(size(xyCoord, 1), size(xyCoord, 2));
for i= 1:size(xyCoord,1)
    if(isnan(xyCoord(i, 1)) | isnan(xyCoord(i,2))
        continue;
    end
    tempo(counter,:) = xyCoord(i,:);
    counter = counter+1;
end

xyCoord = tempo(1:counter-1,:);

len = size(xyCoord,1);
ind = xyCoord(2:len,3);

%%% Obtain Inputs %%

dx = xyCoord(2:len,2)-xyCoord(1:len-1,2);
dy = xyCoord(2:len,2)-xyCoord(1:len-1,2);

xInput = zeros(len-1,10);
yInput = zeros(len-1,10);

for i = 1:10
    xInput(i:len-1,i) = dx(1:len-i);
    yInput(i:len-1,i) = dy(1:len-i);
end

for i = 1:9
    xInput(i,i+1:10) = dx(1);
    yInput(i,i+1:10) = dy(1);
end

%%% Obtain Outputs %%

xOutput = zeros(len-5,6);
yOutput = zeros(len-5,6);

for i = 1:6
    for j = 1:len-i*5
        xOutput(j,i) = xyCoord(j+5*i,1)-xyCoord(j,1);
\[ y_{Output}(j,i) = xy_{Coord}(j+5*i,2)-xy_{Coord}(j,2); \]
end
end
TestNNRandomEvent

% Developed by Geoff Cooney January 2004
% Choose a random event and generates test statistics on it

function [xstats, ystats, FutureDx, FutureDy, xOut, yOut,
    indicator] = TestNNRandomEvent(ASDEnet, ASRnet, coords)

    testcoords = randEvent(coords);
    [xstats, ystats, FutureDx, FutureDy, xOut, yOut, indicator] = TestNNCoords(ASDEnet, ASRnet, testcoords);
TestNNEvents

% Developed by Geoff Cooney January 2004

function [xstats, ystats]=testNNEvents(ASDEnet, ASRnet, coords)

% break event list into individual events
if coords(1,:) == [0,0,0], coords=[0,0,0];coords]; end
if coords(end,:) == [0,0,0], coords=[coords;[0,0,0]]; end
    zeroRows = find(coords(:,1)==0 & coords(:,2)==0 & coords(:,3)==0);
    events = size(zeroRows,1);
sampleCount = 0;
[ASDEinX, ASDEoutX, ASDEinY, ASDEoutY, ASRinX, ASRoutX, ASRinY, ASRoutY, indicator] = buildTS26(coords);
    xstats1 = testNN(ASDEnet, ASDEinX, ASDEoutX);
    xstats2 = testNN(ASRnet, ASRinX, ASRoutX);
    ystats1 = testNN(ASDEnet, ASDEinY, ASDEoutY);
    ystats2 = testNN(ASRnet, ASRinY, ASRoutY);
    xstats.ae = [xstats1.ae, xstats2.ae];
    xstats.mse = [xstats1.mse, xstats2.mse];
    xstats.max = [xstats1.max, xstats2.max];
    xstats.points = [xstats1.points, xstats2.points];
    ystats.ae = [ystats1.ae, ystats2.ae];
    ystats.mse = [ystats1.mse, ystats2.mse];
    ystats.max = [ystats1.max, ystats2.max];
    ystats.points = [ystats1.points, ystats2.points];
randomN

% developed by Geoff Cooney January 2004
% takes an event list and an integer, n. Chooses n distinct random events from the event
% list

function endEvents = randomN(events, n)
    zeroRows = find(events(:,3)==0);
    temp = sort(ceil(rand(n,1)*(size(zeroRows, 1)-1)));
    endEvents = [];
    for i=1:size(temp,1)
        k = temp(i,1);
        endEvents = [endEvents; events(zeroRows(k):zeroRows(k+1)-1, :)];
    end


removeOutliers

% developed by Geoff Cooney January 2004
% takes in the neural networks and a list of extracted events and returns a list of events
% with any outliers(max prediction greater than 10000) from the event list

function extractedNew = removeOutliers(ASDEnet, ASRnet, extractedOld)
    extractedNew = [];
    if extractedOld(1,:) == [0 0 0]; extractedOld = [0 0 0; extractedOld]; end
    if extractedOld(end,:) == [0 0 0]; extractedOld = [extractedOld; 0 0 0]; end
    eventBreaks = find(extractedOld(:,1) == 0 & extractedOld(:,2) == 0 & extractedOld(:,3) == 0);
    for i = 1:size(eventBreaks,1)-1
        [xstats, ystats] = testNNEvents(ASDEnet, ASRnet, extractedOld(eventBreaks(i)+1:eventBreaks(i+1)-1, :));
        if not(isempty(xstats.max))
            if xstats.max(6, 1) < 10000 & ystats.max(6, 1) < 10000
                extractedNew = [extractedNew; extractedOld(eventBreaks(i):eventBreaks(i+1)-1, :)];
            end
        end
    end
end
Appendix F: Matlab GUI code

Extract Event GUI

function varargout = extracteventform(varargin)
% EXTRACTEVENTFORM Application M-file for networkDesigner.fig
% FIG = EXTRACTEVENTFORM launch networkDesigner GUI.
% EXTRACTEVENTFORM('callback_name', ...) invoke the named callback.
% Created by Geoff Cooney
% Last Modified by GUIDE v2.0 28-Jan-2004 16:03:26

if nargin == 0 % LAUNCH GUI
    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    set(fig, 'Name', 'Extract Events');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    if nargout > 0
        varargout{1} = fig;
    end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end

% ABOUT CALLBACKS:
% GUIDE automatically appends subfunction prototypes to this file, and
sets objects' callback properties to call them through the FEVAL switchyard above. This comment describes that mechanism.

Each callback subfunction declaration has the following form:

```
<SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
```

The subfunction name is composed using the object's Tag and the callback type separated by '_', e.g. 'slider2_Callback', 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.

H is the callback object's handle (obtained using GCBO).

EVENTDATA is empty, but reserved for future use.

HANDLES is a structure containing handles of components in GUI using tags as fieldnames, e.g. handles.figure1, handles.slider2. This structure is created at GUI startup using GUIHANDLES and stored in the figure's application data using GUIDATA. A copy of the structure is passed to each callback. You can store additional information in this structure at GUI startup, and you can change the structure during callbacks. Call guidata(h, handles) after changing your copy to replace the stored original so that subsequent callbacks see the updates. Type "help guihandles" and "help guidata" for more information.

VARARGIN contains any extra arguments you have passed to the callback. Specify the extra arguments by editing the callback property in the inspector. By default, GUIDE sets the property to:

```
<MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
```

Add any extra arguments after the last argument, before the final closing parenthesis.

```
% ------------------------------------------------------------------
function varargout = mnuFileCallback(h, eventdata, handles, varargin)

% ------------------------------------------------------------------
function varargout = mnuFileExitCallback(h, eventdata, handles, varargin)
    close force;

% ------------------------------------------------------------------
function varargout = listbox2_Callback(h, eventdata, handles, varargin)
    updateExtractionFields(handles)
```

143
% function varargout = butRunway_Callback(h, eventdata, handles, varargin)
set(handles.lstExtractions, 'string', [get(handles.lstExtractions, 'string'); cellstr('newEvent')]);
set(handles.lstExtractions, 'UserData', [get(handles.lstExtractions, 'UserData'); 0 0 0 0 0 0]);
updateExtractionFields(handles);

% function varargout = edit2_Callback(h, eventdata, handles, varargin)
updateExtractions(handles);

% function varargout = edit3_Callback(h, eventdata, handles, varargin)
updateExtractions(handles);

% function varargout = edit4_Callback(h, eventdata, handles, varargin)
updateExtractions(handles);

% function varargout = edit5_Callback(h, eventdata, handles, varargin)
updateExtractions(handles);

% function varargout = butDeleteRunwayCallback(h, eventdata, handles, varargin)
row = get(handles.lstExtractions, 'value');
tData = get(handles.lstExtractions, 'UserData');
tString = get(handles.lstExtractions, 'String');
tData(row, :) = [];
tString(row, :) = [];
set(handles.lstExtractions, 'UserData', tData);
set(handles.lstExtractions, 'String', tString);
if row > size(tString,1) & row > 1
    set(handles.lstExtractions, 'value', row-1);
end
if size(tString, 1) > 0
    updateExtractionFields(handles);
end

% function varargout = txtAngle_Callback(h, eventdata, handles, varargin)
updateExtractions(handles);
function varargout = txtExtractionName_Callback(h, eventdata, handles, varargin)
    updateExtractions(handles);

% --------------------------------------------------
function varargout = radLand_Callback(h, eventdata, handles, varargin)

    set(handles.radLand, 'value', 1);
    set(handles.radTakeOff, 'value', 0);
    updateExtractions(handles);

% --------------------------------------------------
function varargout = radTakeOff_Callback(h, eventdata, handles, varargin)

    set(handles.radLand, 'value', 0);
    set(handles.radTakeOff, 'value', 1);
    updateExtractions(handles);

function updateExtractions(handles)
    LowX = str2num(get(handles.txtLx, 'string'));
    LowY = str2num(get(handles.txtLy, 'string'));
    UpX = str2num(get(handles.txtUx, 'string'));
    UpY = str2num(get(handles.txtUy, 'string'));
    deg = str2num(get(handles.txtAngle, 'string'));
    takeOff = get(handles.radLand, 'value');
    tData = get(handles.lstExtractions, 'UserData');
    tString = get(handles.lstExtractions, 'String');
    tData(get(handles.lstExtractions, 'value'), :) = [LowX LowY UpX UpY deg takeOff];
    tString(get(handles.lstExtractions, 'value'), :) = cellstr(get(handles.txtExtractionName, 'string'));
    set(handles.lstExtractions, 'UserData', tData);
    set(handles.lstExtractions, 'String', tString);

function updateExtractionFields(handles)
    tData = get(handles.lstExtractions, 'UserData');
    curData = tData(get(handles.lstExtractions, 'value'), :);
    set(handles.txtLx, 'string', num2str(curData(1)));
    set(handles.txtLy, 'string', num2str(curData(2)));
    set(handles.txtUx, 'string', num2str(curData(3)));
    set(handles.txtUy, 'string', num2str(curData(4)));
    set(handles.txtAngle, 'string', num2str(curData(5)));
    set(handles.radLand, 'value', curData(6));
    set(handles.radTakeOff, 'value', not(curData(6)));
    tString = get(handles.lstExtractions, 'String');
    set(handles.txtExtractionName, 'string', tString{get(handles.lstExtractions, 'value')});

% --------------------------------------------------
function varargout = butSaveRunway_Callback(h, eventdata, handles, varargin)

    [file, path] = uiputfile("");
    if (file ~= 0)
        saveRunwayExtractions(strcat(path, file), get(handles.lstExtractions, 'String'),
                                get(handles.lstExtractions, 'UserData'));
    end

% ---------------------------------------------------------------------
function varargout = butLoadRunway_Callback(h, eventdata, handles, varargin)

    [file, path] = uigetfile("");
    if (path ~= 0)
        temp = load(strcat(path, file));
        set(handles.lstExtractions, 'String', temp.runwayNames);
        set(handles.lstExtractions, 'UserData', temp.extracted);
        set(handles.lstExtractions, 'value', 1);
    end

% ---------------------------------------------------------------------
function varargout = butExtract_Callback(h, eventdata, handles, varargin)

    Log3Dfile = get(handles.txtLog3DFile, 'String');
    Log3Dvar = get(handles.txtLog3DVar, 'String');
    tData = get(handles.lstExtractions, 'UserData');
    names = get(handles.lstExtractions, 'String');
    extracted = performExtractions(Log3Dfile, Log3Dvar, names, tData);
    [file, path] = uiputfile("");
    save(strcat(path, file), 'extracted');
    trainnnform(extracted);

% ---------------------------------------------------------------------
function varargout = txt3DLogFile_Callback(h, eventdata, handles, varargin)

% ---------------------------------------------------------------------
function varargout = txtLog3DVarCallback(h, eventdata, handles, varargin)
Network Training GUI

function varargout = trainNNForm(varargin)
% TRAINNNFORM Application M-file for trainNNForm.fig
% FIG = TRAINNNFORM launch trainNNForm GUI.
% TRAINNNFORM('callback_name', ...) invoke the named callback.

% created by Geoff Cooney january 2004

% Last Modified by GUIDE v2.0 28-Jan-2004 16:48:38

if nargin == 0 % LAUNCH GUI
    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    set(fig, 'Name', 'Train Network');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    if nargout > 0
        varargout{1} = fig;
    end
elseif isstruct(varargin{1})
    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    set(fig, 'Name', 'Train Network');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);
    set(handles.lstEvents, 'UserData', varargin{1});
    set(handles.lstEvents, 'String', fieldnames(varargin{1}));
    updateFields(handles);

    if nargout > 0
        varargout{1} = fig;
    end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL
        switchyard
            else
                feval(varargin{:}); % FEVAL switchyard
        end
        catch
            disp(lasterr);
        end
    end

    end

    % ABOUT CALLBACKS:
    % GUIDE automatically appends subfunction prototypes to this file, and
    % sets objects’ callback properties to call them through the FEVAL
    % switchyard above. This comment describes that mechanism.
    %
    % Each callback subfunction declaration has the following form:
    % <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
    %
    % The subfunction name is composed using the object’s Tag and the
    % callback type separated by ’_’, e.g. ’slider2_Callback’,
    % ’figure1_CloseRequestFcn’, ’axis1_ButtondownFcn’.
    %
    % H is the callback object’s handle (obtained using GCBO).
    %
    % EVENTDATA is empty, but reserved for future use.
    %
    % HANDLES is a structure containing handles of components in GUI using
    % tags as fieldnames, e.g. handles.figure1, handles.slider2. This
    % structure is created at GUI startup using GUIHANDLES and stored in
    % the figure’s application data using GUIDATA. A copy of the structure
    % is passed to each callback. You can store additional information in
    % this structure at GUI startup, and you can change the structure
    % during callbacks. Call guidata(h, handles) after changing your
    % copy to replace the stored original so that subsequent callbacks see
    % the updates. Type ”help guihandles” and ”help guidata” for more
    % information.
    %
    % VARARGIN contains any extra arguments you have passed to the
    % callback. Specify the extra arguments by editing the callback
    % property in the inspector. By default, GUIDE sets the property to:
function varargout = lstEvents_Callback(hObject, eventdata, handles, varargin)
    updateFields(handles);

function varargout = butTrain_Callback(hObject, eventdata, handles, varargin)
    extracted = get(handles.lstEvents, 'UserData');
    net = trainNN(combineEvents(extracted));
    [file, path] = uiputfile('');
    if (path ~= 0)
        save(strcat(path, file), 'net');
    end
    testnnform(net);

function varargout = butDelete_Callback(hObject, eventdata, handles, varargin)
    row = get(handles.lstEvents, 'value');
    tData = get(handles.lstEvents, 'UserData');
    tString = get(handles.lstEvents, 'String');
    tData = rmfield(tData, tString(row,:));
    tString(row,:) = [];
    set(handles.lstEvents, 'UserData', tData);
    set(handles.lstEvents, 'String', tString);
    if row > size(tString,1) & row > 1
        set(handles.lstEvents, 'value', row-1);
    end
    if size(tString, 1) > 0
        updateFields(handles);
    end

function varargout = edit1_Callback(hObject, eventdata, handles, varargin)

function varargout = pushbutton3_Callback(hObject, eventdata, handles, varargin)
% save
[file, path] = uiputfile('');
if (path ~= 0)
    extracted = get(handles.lstEvents, 'UserData');
    save(strcat(path, file), 'extracted');
end

% function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
[file, path] = uigetfile('');
if (path ~= 0)
    temp = load(strcat(path, file));
    loadEvents(handles, temp.extracted);
end

% function varargout = chkOverride_Callback(h, eventdata, handles, varargin)

% function varargout = txtEventName_Callback(h, eventdata, handles, varargin)
tString = get(handles.lstEvents, 'String');
newName = cellstr(get(handles.txtEventName, 'String'));
oldName = tString(get(handles.lstEvents, 'value'), :);
tString(get(handles.lstEvents, 'value'), :) = newName;
set(handles.lstEvents, 'String', tString);
tData = get(handles.lstEvents, 'UserData');
tData = setField(tData, newName{1}, getfield(tData, oldName{1}));
tData = rmfield(tData, oldName{1});
set(handles.lstEvents, 'UserData', tData);

% function varargout = txtEventCount_Callback(h, eventdata, handles, varargin)

% function updateFields(handles)
tString = get(handles.lstEvents, 'String');
curString = tString(get(handles.lstEvents, 'value'));
tData = get(handles.lstEvents, 'UserData');
curData = getField(tData, curString);
set(handles.txtEventName, 'String', curString);
set(handles.txtEventCount, 'String', ExtractCount(curData));

function loadEvents(handles, extracted)
    overwrite = get(handles.chkOverride, 'value');
    tData = get(handles.lstEvents, 'UserData');
    if (overwrite == 0 & ~isempty(tData))

        names = fieldnames(extracted);
        length = size(names, 1);
        for i = 1:length
            eventName = names{i};
            eventData = getField(extracted, eventName);
            while (isField(tData, eventName))
                eventName = strcat(eventName, '1');
            end
            tData = setField(tData, eventName, eventData);
        end
        set(handles.lstEvents, 'UserData', tData);
        set(handles.lstEvents, 'String', fieldnames(tData));
    else
        set(handles.lstEvents, 'UserData', extracted);
        set(handles.lstEvents, 'String', fieldnames(extracted));
    end
    updateFields(handles);
Network Testing GUI

function varargout = testNNForm(varargin)
% TESTNNFORM Application M-file for testNNForm.fig
% FIG = TESTNNFORM launch testNNForm GUI.
% TESTNNFORM(‘callback_name’,...) invoke the named callback.

% created by Geoff Cooney January 2004

% Last Modified by GUIDE v2.0 30-Jan-2004 00:10:45

if nargin == 0 % LAUNCH GUI
    fig = openfig(mfilename,’reuse’);

    % Use system color scheme for figure:
    set(fig,’Color’,get(0,’defaultUicontrolBackgroundColor’));
    set(fig,’Name’,’Test Network’);

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    if nargout > 0
        varargout{1} = fig;
    end
elseif isstruct(varargin{1})
    fig = openfig(mfilename,’reuse’);

    % Use system color scheme for figure:
    set(fig,’Color’,get(0,’defaultUicontrolBackgroundColor’));
    set(fig,’Name’,’Test Network’);

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    set(handles.lblNetFile, ’UserData’, varargin{1});
    set(handles.lblNetFile, ’String’, ’trainedNet’);

    if nargout > 0
        varargout{1} = fig;
    end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL
        switchyard
    end

    catch
        disp(lasterr);
    end

end

% ABOUT CALLBACKS:
% GUIDE automatically appends subfunction prototypes to this file, and
% sets objects’ callback properties to call them through the FEVAL
% switchyard above. This comment describes that mechanism.
% Each callback subfunction declaration has the following form:
% <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
% The subfunction name is composed using the object’s Tag and the
% callback type separated by ‘_’, e.g. ’slider2_Callback’,
% ’figure1_CloseRequestFcn’, ’axis1_ButtondownFcn’.
% H is the callback object’s handle (obtained using GCBO).
% EVENTDATA is empty, but reserved for future use.
% HANDLES is a structure containing handles of components in GUI using
% tags as fieldnames, e.g. handles.figure1, handles.slider2. This
% structure is created at GUI startup using GUIHANDLES and stored in
% the figure’s application data using GUIDATA. A copy of the structure
% is passed to each callback. You can store additional information in
% this structure at GUI startup, and you can change the structure
% during callbacks. Call guidata(h, handles) after changing your
% copy to replace the stored original so that subsequent callbacks see
% the updates. Type ”help guihandles” and ”help guidata” for more
% information.
% VARARGIN contains any extra arguments you have passed to the
% callback. Specify the extra arguments by editing the callback
% property in the inspector. By default, GUIDE sets the property to:
Add any extra arguments after the last argument, before the final closing parenthesis.

function varargout = listbox1_Callback(h, eventdata, handles, varargin)
    updateFields(handles);

function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
    if str2num(get(handles.txtEventCount, 'String')) == 0
        events = randomEvent(combineEvents(get(handles.lstEvents, 'UserData')));
    else
        tEvents = get(handles.lstEvents, 'UserData');
        events = randomEvent(getfield(tEvents, get(handles.txtEventName, 'String')));
    end
    net = get(handles.lblNetFile, 'UserData');
    [stats.xstats, stats.ystats, stats.FutureDx, stats.FutureDy, stats.xOut, stats.yOut, stats.indicator] = testNNCoords(net.ASDEnet, net.ASRnet, events);
    [file, path] = uiputfile('');
    if (path ~= 0)
        save(strcat(path, file), 'stats');
    end

function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
    events = combineEvents(get(handles.lstEvents, 'UserData'));
    net = get(handles.lblNetFile, 'UserData');
    [stats.xstats, stats.ystats] = testNNEvents(net.ASDEnet, net.ASRnet, events);
    [file, path] = uiputfile('');
    if (path ~= 0)
        save(strcat(path, file), 'stats');
    end

function varargout = edit1_Callback(h, eventdata, handles, varargin)

function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
    %save
    [file, path] = uiputfile('');
    if (path ~= 0)
        extracted = get(handles.lstEvents, 'UserData');
        save(strcat(path, file), 'extracted');
    end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
    [file, path] = uigetfile(');
    if (path ~= 0)
        temp = load(strcat(path, file));
        loadEvents(handles, temp.extracted);
    end

function varargout = checkbox1Callback(h, eventdata, handles, varargin)

function varargout = edit2_Callback(h, eventdata, handles, varargin)

function varargout = butDelete_Callback(h, eventdata, handles, varargin)
    row = get(handles.lstEvents, 'value');
    tData = get(handles.lstEvents, 'UserData');
    tString = get(handles.lstEvents, 'String');
    tData = rmfield(tData, tString(row, :));
    tString(row, :) = [];
    set(handles.lstEvents, 'UserData', tData);
    set(handles.lstEvents, 'String', tString);
    if row > size(tString,1) & row > 1
        set(handles.lstExtractions, 'value', row-1);
    end
    if size(tString, 1) > 0
        updateExtractionFields(handles);
    end

function varargout = edit5_Callback(h, eventdata, handles, varargin)
    tString = get(handles.lstEvents, 'String');
    newName = cellstr(get(handles.txtEventName, 'String'));
    oldName = tString(get(handles.lstEvents, 'value'), :);
    tString(get(handles.lstEvents, 'value'), :) = newName;
    set(handles.lstEvents, 'String', tString);
    tData = get(handles.lstEvents, 'UserData');
    tData = setField(tData, newName{1}, getField(tData, oldName{1}));
    tData = rmfield(tData, oldName{1});
    set(handles.lstEvents, 'UserData', tData);

function varargout = txtNetFilename_Callback(h, eventdata, handles, varargin)

end
function varargout = butNetLoad_Callback(h, eventdata, handles, varargin)
    [file, path] = uiputfile('');
    if (path ~= 0)
        load(strcat(path, file));
        set(handles.lblNetFile, 'UserData', net);
        set(handles.lblNetFile, 'String', file);
    end

% --------------------------------------------
function varargout = butNetSave_Callback(h, eventdata, handles, varargin)
    [file, path] = uiputfile('');
    if (path ~= 0)
        net = get(handles.lblNetFile, 'UserData');
        save(strcat(path, file), 'net');
    end

% -------------------------------
function varargout = lstStats_Callback(h, eventdata, handles, varargin)

% --------------------------------------------
function updateFields(handles)
    tString = get(handles.lstEvents, 'String');
    curString = tString{get(handles.lstEvents, 'value')};
    tData = get(handles.lstEvents, 'UserData');
    curData = getField(tData, curString);
    set(handles.txtEventName, 'String', curString);
    set(handles.txtEventCount, 'String', ExtractCount(curData));

% --------------------------------------------
function loadEvents(handles, extracted)
    overwrite = get(handles.chkOverride, 'value');
    tData = get(handles.lstEvents, 'UserData');
    if (overwrite == 0 & not(isempty(tData))

        names = fieldnames(extracted);
        length = size(names, 1);
        for i = 1:length
            eventName = names{i};
            eventData = getField(extracted, eventName);
            while (isField(tData, eventName))
                eventName = strcat(eventName, '1');
            end
            tData = setField(tData, eventName, eventData);
        end
set(handles.lstEvents, 'UserData', tData);
set(handles.lstEvents, 'String', fieldnames(tData));
else
    set(handles.lstEvents, 'UserData', extracted);
    set(handles.lstEvents, 'String', fieldnames(extracted));
end
updateFields(handles);

function varargout = txtStats_Callback(h, eventdata, handles, varargin)


Appendix G: Graphs of Simulated Alarm Conditions

Event 1: Lnd-Stp (end of runway)
Event 2: Lnd-Stp (Front of Runway)
Event 3: Dep-Taxi
Event 4: Lnd-Taxi
Event 5: Dep-Stp