AUTOMATIC TARGET RECOGNITION USING NONLINEAR AUTOREGRESSIVE NEURAL NETWORKS

THESIS

Marc R. Ward, Captain, USAF

AFIT-ENS-14-M-33

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-14-M-33

AUTOMATIC TARGET RECOGNITION USING NONLINEAR AUTOREGRESSIVE NEURAL NETWORKS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Analysis

Marc R. Ward, BS

Captain, USAF

March 2014

AFIT-ENS-14-M-33

AUTOMATIC TARGET RECOGNITION USING NONLINEAR AUTOREGRESSIVE NEURAL NETWORKS

Marc R. Ward, BS
Captain, USAF

Approved:

_____//signed// _____	10 March 2014
Kenneth W. Bauer Jr, Ph.D. (Chairman)	Date
_____//signed// _____	10 March 2014
John O. Miller, Ph.D. (Member)	Date
_____//signed// _____	10 March 2014
David M. Ryer, Lt. Col., Ph.D. (Member)	Date

Abstract

Accurate combat identification is critical to military interactions. Laser radar for vehicle identification is a rapidly developing field that could possibly assist in combat identification by providing information about operating characteristics of a particular vehicle based on measured vibrations. This research focuses on simulated laser radar data collected from mounted vibrometers on idling vehicles. An approach to identify vehicles using nonlinear autoregressive neural networks for classification is developed and employed. The resulting algorithm combines the trained neural networks across three dimensions of vibration readings. This method offers improved performance over literature in successfully identifying a vehicle through vibration measurements alone.

Dedicated to my wife and our wonderful children

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Kenneth Bauer Jr., for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated.

Marc R. Ward, Capt

TABLE OF CONTENTS

Page

List of Figures

List of Tables

AUTOMATIC TARGET RECOGNITION USING NONLINEAR AUTOREGRESSIVE NEURAL NETWORKS

I. Introduction

Background

Laser radar (LADAR) for vehicle identification is a rapidly developing field that could possibly assist in combat identification by providing information about operating characteristics of a particular vehicle based on measured vibrations. Research has been focused on the identifying critical features based upon the frequency domain characteristics in order to obtain a classification. This research expands on the body of knowledge and provides an alternative approach rather than the classic Principle Component Analysis (PCA).

Problem Statement

The ability to quickly and accurately identify objects on the battlefield is an essential ability for warfigthers. Remote LADAR vibrometry remains an emerging field of study for classification purposes. Target identification from vibrometry data could enable operators to distinguish between unique vibrations signatures. Figure 1 illustrates one possible implementation of LADAR vibrometry for automatic target recognition (ATR).

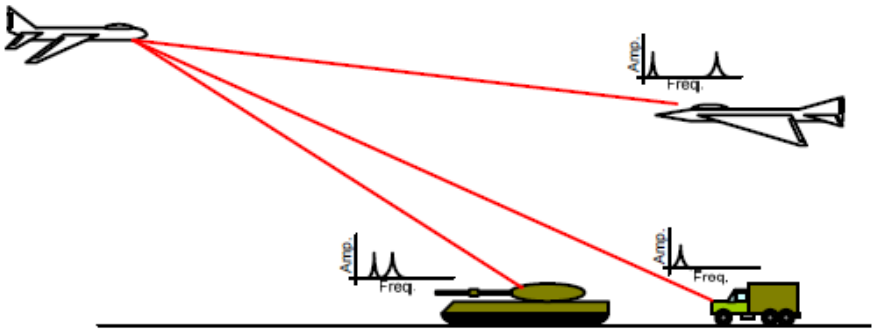


Figure 1 Laser Radar Automatic Target Recognition, taken from (Jameson, 2007)

Each vehicle vibrates different depending on the type of engine, the engine state, and characteristics of the vehicle's body. Herein, three dimensional simulated LADAR vibrational data will be used. Trained autoregressive neural networks using time-series vibrations will be developed and employed for quick and accurate prediction of the objects of interest with results exceeding previous work in this field.

Objectives

The objective of this thesis research was to create a classification algorithm using neural networks to identify vehicle types based upon simulated laser vibrometry data.

Hypothesis

The hypothesis of this research is entailed in a proof of concept that a time series input could be used by neural networks to distinguish between previously known vehicles.

Assumptions

There are not many assumptions necessary in moving forward with solutions for this problem. The main underlying assumption was the simulated laser vibrometry data collected via accelerometers during data collection mimicked the actual signal a LADAR system would retrieve on the same vehicle. Another assumption was all sensors collecting data during a particular run on a specific vehicle were started and stopped at simultaneous time periods.

Limitations

This research is limited to three specified vehicles of interest. More vehicles could be added to the algorithm with more data but that would require many of the steps followed to be re-accomplished.

Implications

This proof of concept demonstrates it would be possible to perform automatic target recognition using the time series data from vibrations obtained while a vehicle of interest is operating. Although other techniques have produced significant results, the algorithm developed here adds to the body of knowledge moving forward.

Format

This thesis is divided as such, Chapter II examines prior literature in this area, Chapter III explains the methodology used to develop the classification algorithm, Chapter IV lists the results obtained by the final model, and Chapter V summarizes this research and provides insight into future research areas.

II. Literature Review

This chapter examines prior work in artificial neural networks (ANN), as applied to time series, non-time series data, and LADAR vibrometry for automatic target recognition (ATR). Time series and non-time series of data pose separate problems to the analyst and must be processed differently. With respect to time series data there is a large body of work specifically focused on stock and commodity market prices (Kaastra & Boyd, 1996) with a goal of discovering non-linear relationships via ANNs which might provide an operational advantage over competitors. For categorical input data, discovering differences and/or patterns amongst the samples in order to predict the class of a new sample continues to be the main focus. ANNs have been applied to many fields including business (Young, Bihl, & Weckman, July 2013) (Kuo, Chen, & Hwang, 2001) (Azcarraga, Hsieh, & Setiono, 2008) (Phillips, Phillips, & Hurrell, 2013), politics (Beck, King, & Zeng, 2000), medicine diagnosis (Burke, et al., 1997) (Lisboa, 2002) (Temurtas, Yumusak, & Temurtas, 2009) (Skidmore, 1991) (Laine, Bauer, Lanning, Russell, & Wilson, 2002) (Ubeyli, 2009), insurance (Speights, Brodsky, & Chudova, 1999), ecosystem modeling (Young, et al., 2011), and sports statistical analysis (Young & Weckman, 2008) (Loeffelholz, Bednar, & Bauer, 2009), among other fields as noted by Paliwal and Kumar (2009) and Zhang (2000).

In addition to an overview of ANNs, the literature review analyzes the current state of ATR research. ATR, in a military context, exists to identify enemy, friendly, and neutral objects (tanks/buildings/personnel) in order to limit fratricide and increase combat effectiveness. Various techniques attempt to correctly identify objects of interest but this research reviews current state of the art ATR performed using LADAR vibrometry.

Data Background

Time Series

A vector of past observations from a specific time interval is an example of a time series. For example, monthly stock prices from 2000 through 2012 would provide 13 years of monthly stock prices, or 156 values organized in sequence from oldest time point to newest time point. To be consistent, time series data vectors are generally collected at equal time intervals between observations.

ANN Introduction

Biological nervous systems inspired artificial neural networks, with the overarching goal of ANNs to map a relationship between specific inputs to a particular target output (Young et al., July 2013). To accomplish this, an ANN consists of inputs, outputs, hidden layers, and hidden layer nodes and the connections between these layers and nodes operating in parallel, see Figure 2

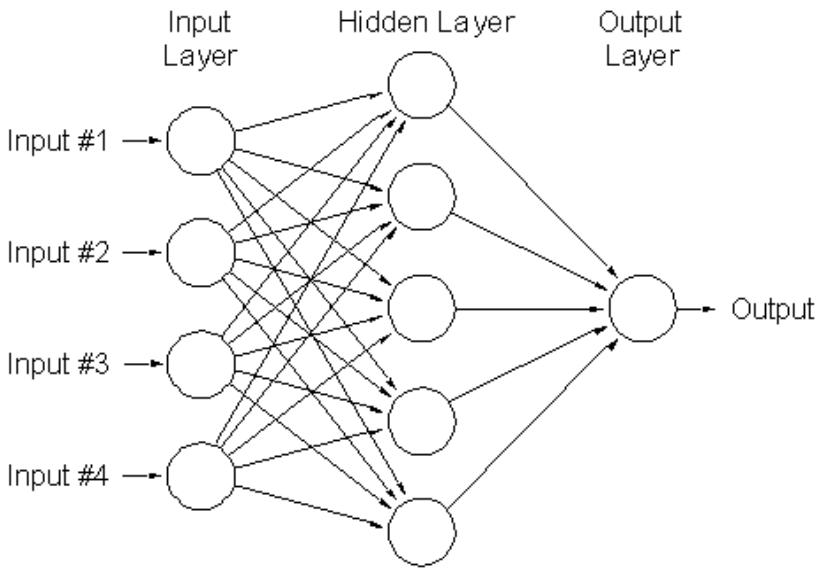


Figure 2 Standard Artificial Neural Network, taken from (Ivry & Michal, 2013)

Building a General ANN

ANNs are supervised classification methods which involve a user configuring and training a given ANN to identify an input pattern as a member of a predefined class or output (Jain, Duin, & Mao, 2000). Computers train a neural network in order to perform a function by adjusting the values of the connections between elements. This flexibility allows ANNs to perform complex functions in fields to include pattern recognition, identification, and classification.

The ANN training stage minimizes the error of the classified outputs by changing the connection weights through the process of backpropagation, which is one method of training, until it has reached a global minimum; occasionally, the minimum is a local one and hence changes in the development should be made. Network architecture plays an important role for neural network classification performance; the optimal topology will depend upon the problem at hand. With an understanding of the problem, selecting the number of hidden layers, units, and feedback connections can be incorporated into the network architecture (Duda, Hart, & Stork, 2001). Training a network can be described as moving down an error surface which takes place by weight adjustments during the learning phase. The standard backpropagation network, proposed by the PDP group (McClelland & Rumelhart, 1988), employs the steepest descent algorithm for adjusting the weights. Once trained, the ANN can predict the classification of a new sample with an inherent error rate. Selecting and adjusting the complexity of the network remains an issue in the use of neural network techniques. Kolmogorov and Hecht-Nielsen posited the sufficiency of one hidden layer for properly posed problems (Young et al., July 2013). However, while most problems can be solved using one hidden layer, complicated non-linear and/or non-separable problems may require multiple hidden layers (Young et al., July

2013). Conversely, the training data cannot be learned adequately if too few parameters are implemented (Duda et al., 2001).

In ANN model development, the learning process helps to identify the optimal weights to assign at each layer and interaction between nodes. At first, the data is divided into training, testing, and sequestered validation sets; various heuristics exist for allocating data into these groups (Young et al., July 2013). One issue to avoid is possibly training and testing over the same observations (Zhang, 2007); therefore, the cross validation methods are common. Training data presents the first signals into the net which are passed through to determine the output at the output layer. At this point, the output is compared with the target values and any difference corresponds to an error (Duda et al., 2001). The network's goal is to minimize the error between the target values and calculated outputs; until some threshold of error is obtained the ANN will iterate this process and adjust the network's weights after each step. Two independently selected subsets of the training data which were removed before learning are used to perform validation and testing. The validation set decides when to stop the training; the test set is examined after model building and used to evaluate the performance of the network.

ANN Pattern Recognition

A two-layer feed forward network remains the standard network used for pattern recognition. A sigmoid transfer function connects the nodes in both the layers of the ANN. To process through a network the data is multiplied by the connections weights added to a given bias then sent through a sigmoid transfer function before being sent to the next layer. The process repeats itself in the next layer to produce an output. The user defines the number of first layer (generally defined as the hidden layer) nodes and the number of second layer (output layer) nodes equals the number of classification states. Figure 2 depicts a standard network with four

inputs, five first layer nodes, and one output layer node, a general neural structure as first described by McCulloch and Pitts in 1943 (McCulloch & Pitts, 1943). ANNs can also take different forms than the two-layer feed forward network described above, multiple hidden layers are permissible but usually not optimal (Young et al., July 2013), additionally feedback and feedforward aspects are possible. A recurrent ANN, Figure 3, differentiates itself from the feed forward network in Figure 2 because it has at least one feedback loop contained in its structure. In other words, the output of one state provides an input into another state.

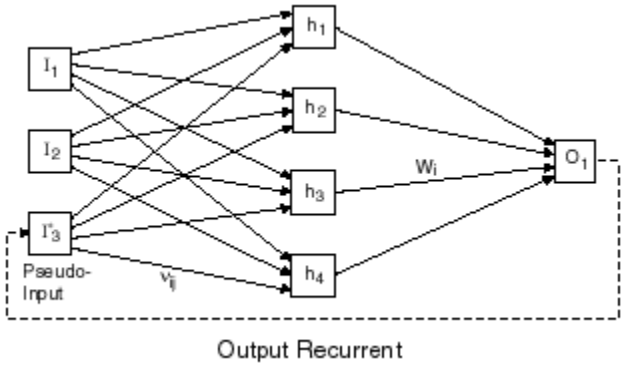


Figure 3 Recurrent Neural Network, taken from (O'Brien, 2012)

The recurrent ANN results in a nonlinear dynamic behavior because the dependent structure of the neurons (Haykin, 1994). Recurrent neural networks provide the ability to model non-linear dependencies and can be used with time series data sets to enable feedback loops. These feedback loops help during the training phase where the network learns from its mistakes while optimizing the performance. In their most general form, researchers have found greatest use of recurrent networks in time series prediction and are effective in learning time-dependent signals whose structure varies over short periods (Duda et al., 2001).

ANN Classification

System state classification continues to remain an important facet of human nature; whether classifying a road as safe or dangerous or a business decision as high or low risk, we must use the information at hand and make a decision. Generally, these decisions are made in context of prior knowledge which aids in the classification. The Bayesian methodology of using prior outcomes to calculate a probability of some outcome plays a role in these decisions. In engineering and mathematics, traditional statistical classification works well when the underlying assumptions are met (data independence, distribution assumptions, linearity, and etc.); however, issues arise when a problem is non-linear or fails to meet an underlying assumption. ANNs do not make distribution assumptions about data and can provide a flexible tool to tackle these more difficult problems.

ANNs are data driven self-adaptive methods. They can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model (Zhang, 2000). The self-adaptive nature of neural networks allows them to fit into any size and shape hole as long as it is provided with enough data points to adapt to. Without having to fit any of the standard classic assumptions of normality or independence this methodology separates itself into a different playing field.

Despite ANN's ability to perform either classification or prediction, Zhang (2000) claims that classification research remains the most researched topic of ANNs. The research contained in this paper furthers Zhang's claim and the body of work surrounding neural networks and state classification.

Feature Extraction versus Feature Selection

The degree of difficulty of the classification problem depends on the variability in the feature values for objects in the same category relative to the difference between feature values in different categories. The variability of feature values for objects in the same category may be due to complexity or due to noise (Duda et al., 2001). Research has focused on feature or input selection attempting to determine the most important variables to inject into the model and discard noisy features (Verikas & Bacauskiene, 2002). One technique to conduct feature selection includes the processing of the resulting signal-to-noise ratio (SNR) provided by the features selected. A feature with the lowest SNR measure will be discarded before the neural net is re-trained. This process repeats itself, removing features in a step-wise fashion until a significant fraction of the classification error is calculated. At which point the most recently removed feature would be re-instated and the final list of features would be solidified (Bauer, Alsing, & Greene, 2000). The ANN SNR feature selection approach performs favorably compared to other backward selection methods for ANNs (Verikas & Bacauskiene, 2002), and is applied to current research problems (Ubeyli, 2009) (Ubeyli, 2008) (Bihl & Bauer, to be submitted: 2014).

The ability of a neural network to correctly classify objects into their true classes is measured by the classification error rate. Minimizing the percentage of new objects incorrectly assigned (assigned to the wrong category) remains the goal in ANN creation and design. Another technique includes minimizing the risk or total expected cost of misclassification. This can be incorporated in the training phase of the neural network if the cost of false positives and false negatives are known.

Given the recent economic crisis experienced and the harm caused the global assets, market practitioners studied the ability to predict future asset prices. One paper, (Pacelli, Bevilacqua, & Azzollini, 2011) aimed to analyze the ability of ANNs to predict the behavior in a highly liquid (high efficiency) market, the Euro/US dollar exchange rates. Pacelli et al. (2011) assumed two hypotheses when conducting their research. The first hypothesis assumed that the process of pricing in financial markets was not random; if this proved to be invalid, then they demonstrated that no model could predict prices. Their second hypothesis stated the degree of information efficiency of the financial markets is not strong or semi-strong; if the second hypothesis could be proved invalid then all relevant information is instantly incorporated into the pricing of financial products yielding the act of predicting unnecessary (Pacelli et al., 2011). At first, Pacelli et al.'s (2011) list of input variables included over forty possible financial data sets which could provide predictive ability; they eliminated any variables which were collected only monthly, and among the daily collected variables, any variable with a Pearson correlation coefficient with another variable above a threshold was removed. These two criteria left only seven input variables for the neural network. Once they standardized their data and performed trial and error to determine a viable network topology the researchers arrived at a conclusive result. Through analysis of the data it is possible to say that the ANN model developed can predict the trend to three days of Euro/USD exchange rate. This predictive ability from the ANN demonstrates both their assumptions were valid. Their analysis provided evidence to support their hypotheses that the processes of pricing in financial markets are determined by interaction between actors and relationships between variables of a nonlinear nature (Pacelli et al., 2011).

ANN Time Series Prediction

ANNs can also be used for prediction; for this research, we are interested in predicting time series values to predict future values or impute missing values. Similar to pattern recognition neural networks, time series ANNs can either predict the next value or a set of future values. Different structures for time series ANNs exist and implemented in various ways given a specific problem and its underlying data. In one type of time series problem, a user predicts future values of a time series $y(t)$ from past values of that time series and past values of a second time series $x(t)$. This ANN is referred to as a nonlinear autoregressive with exogenous input (NARX) network (NARXNet) (Beale, Hagan, & Demuth, 2013). A NARXNet follows the mathematical formulation:

$$y^*(t) = f(y(t-1), ..., y(t-d), x(t-1), ..., x(t-d)) \quad (1)$$

where y^* is the predicted value at time t, x is an exogenous variable's value at a given time, f represents the neural network, t is the time of data collection, and d is (Beale et al., 2013).

In operation, a trained NARXNet will predict future values of a stock, based on such economic variables as company earnings and trading volatility. When created to represent dynamic systems, a NARXNet can also execute system classification. Another time series problem is similar to the NARXNet and involves two series, but without information of previous values of $y(t)$ (Beale et al., 2013). This input-output model can be written as:

$$y(t) = f(x(t-1), ..., x(t-d)) \quad (2)$$

Compared to the input-output model, the NARXNet which "will provide better predictions than input-output model, because it uses the additional information contained in the previous values of $y(t)$" (Beale et al., 2013). However, there may be some applications in which the previous

values of $y(t)$ would not be available. Those are the only cases where you would want to use the input-output model instead of the NARXNet.

A third type of time series problem involves only one series, the future values of a time series *being* predicted using only past values of itself. Referred to as nonlinear autoregressive network (NARNet) this prediction technique can be written as where d is the lag desired:

$$y(t) = f(y(t-1), ..., y(t-d)). \quad 3$$

Figure 4 depicts the NARNet's structure where a predetermined number of time periods are used as an input for the network to predict the next time step value Figure 4.

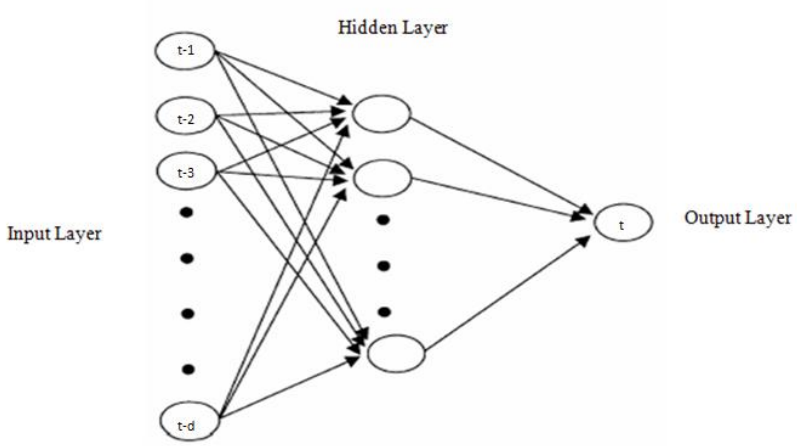


Figure 4 Nonlinear Autoregressive Neural Network

Long term Forecasting

Forecasters are not simply interested in a one period ahead prediction but a long-term prediction of time series. There are several choices to build long-term prediction models, the *direct* and the *recursive* prediction strategies (Sorjamaa, Hao, Reyhani, Ji, & Lendasse, 2007). However, long-term prediction faces increasing uncertainties from various sources, including the lack of information about a system's current state.

The *recursive* strategy appears to be the most intuitive as it views the predicted values as known data to predict the next ones; for example, the fifth predicted values will use the first through fourth predicted values as well as the known values of the data series to predict the sixth predicted value. The accuracy of this strategy deteriorates significantly when the number of predicted values exceeds the number of inputs (as you move farther from truth data).

The *direct* model, on the other hand, does not contain this degradation as it will produce the predicted values at the same time instead of iteratively. The direct model increases the complexity but more accurate results are achieved (Sorjamaa et al., 2007).

Nonlinear Autoregressive with Exogenous Input Network

The NARXNet methodology uses the time series of interest as a main input as well as other user selected seemingly unrelated data streams to forecast future data points. When applied to time series prediction, the NARXNet is designed as a feedforward time delay neural network (TDNN) without any feedback loop (Haykin, 1994).

Researchers implement NARXNets in various fields to achieve predictions of future values; Lee and Chang (2009) employed a NARXNet for studying the thermodynamics in a pulsating heat pipe (PHP), a type of cooling device which contains unsteady flow oscillations formed by the passing non-uniform distributions of vapor plugs and liquid slugs. A NARXNet used to represent discrete time multi-variable non-linear stochastic systems is derived from the neural network, as shown in Figure 5.

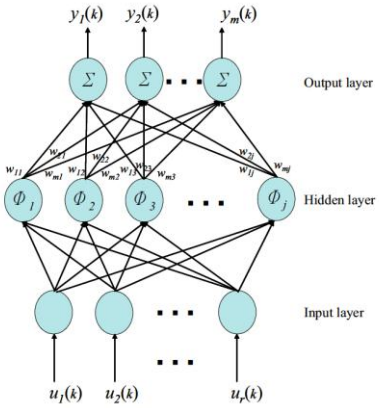


Figure 5 NARXNet, taken from (Lee & Chang, 2009)

The NARXNet consists of an input layer with n nodes, a hidden layer with m neurons and an output layer with *j* nodes. Each of the input nodes is connected to all the neurons in the hidden layer with different weights, and each of the hidden layer nodes is connected to the output node through different weights as well. For instance, the *m*-th output node is connected to all

nodes in the hidden layer with different weighting. The NARXNet model can be expressed by equation 1 above. At the input layer, the input values are not restricted to single values but, in the time series case, a vector of past values of a predetermined length. Varying the length of the input vectors allows the system to achieve the best performance. As stated previously the predicted values of the time series will drift farther from the actual values and induce more error as the input vectors' contain less values than the desired output.

Lee and Chang (2009) conducted numerous experimental designs before they reached their desired NARXNet model. The designs can vary significantly as they decided what variables and time series to include or exclude. In conclusion Lee and Chang (2009) were satisfied with their approach and believed they proved the NARXNet approach could establish appropriate models for time series successfully.

ANN Time Series Implementation

As discussed above, many researchers use time series networks to predict and forecast future values of the time series. In fact, the US government in 1989 "embarked on a five-year, multi-million dollar program for neural network research, but financial services organizations have been the principal sponsors of research in neural network applications" (Trippi & DeSieno, 1992). Researchers in Germany attempted to implement a NARXNet to predict future price movements of natural gas, Busse et al. (2012) discovered that "the best performance could be achieved selecting only five input factors (the temperature forecast four days ahead, the natural gas spot prices of the three major hubs and the exchange rate USD/EUR". The number of lag periods to include along with the inclusion or exclusion of external data sets will result in the NARXNet's structure and its performance. However, every researcher must tailor their network to the problem at hand and vary the parameters to increase performance and minimize error.

Unfortunately, this process requires many iterations accompanied with patience and determination.

Two studies, Surkan and Singleton (1991) and Odom and Sharda (1991), compared ANN models to multivariate discriminate analysis (MDA) models and found significant results. Surkan and Singleton (1991) discovered that an ANN outperformed their MDA model for bond ratings, with the ANN providing 88% correct classification compared with at most 57% by MDA. In a separate analysis, Odom and Sharda (1991) created both ANN and MDA models for predicting corporate bankruptcy probabilities, with the ANN being over 20% more accurate than the MDA model. Additional comparisons of ANNs to other classification methods which show benefits to ANNs, include Kurt et al. (2008), Dreiseitl and Ohno-Machado (2002), and Manel et al. (1999).

Soman (2008) examined implementing NARXNet in his thesis at Rutgers University to forecast future values of currency trades. Through Mathworks' MatLab® software, this method iterated, varied, and optimized the structure of the time series neural network to provide the desired output. Soman (2008) thereby created a model which could adapt to current information by selecting amongst multiple trained NARXNets to produce an optimal prediction; this research discovered that an adaptive strategy with multiple NARXNets performed better than a static NARXNet and standard implementation of technical indicators (linear regression, relative strength index, etc).

Other published research proposed using neural networks to forecast the behavior of multivariate time series. Chakraborty et al. (1992) modeled flour prices over an eight year period for the cities of Buffalo, Minneapolis and Kansas City via a neural network and compared their results to a standard linear statistical model. Chakraborty et al. (1992) implemented

various techniques to include create separate models for all three cities, one model using data from all three cities, and different experimentation with lag output predictions. Regarding the lag output predictions, a multi-lag network used predictions of the network to predict the next time series value versus a one-lag model which only used the actual values to predict the next data point. Improved performance resulted from the combined modeling approach for the presented data. The researchers stated that the separate modeling gives poorer results than combined modeling because each series carries information valuable not only for prediction of its own future values but also for those of the other two series and the combined modeling training set contains three times as many observations as are available for each single modeling training set. They claimed success in training the networks to learn the price curve for each of the modeled cities, and therefore could make accurate price predictions. Results indicated that the neural network approach led to better predictions than the classic statistical model implemented (Chakraborty, Mehrotra, Mohan, & Ranka, 1992).

Time series neural networks and ANNs in general are flexible frameworks for modeling a wide range of nonlinear problems. Zhang (2003) states "one significant advantage of the ANN models over other classes of nonlinear model is that ANNs are universal approximators which can approximate a large class of functions with a high degree of accuracy". In fact, Zhang (2003) implemented a hybrid approach to forecast future values of a time series combining both a non-traditional Nonlinear Autoregressive (NAR) ANN and a more traditional autoregressive integrated moving average (ARIMA) to produce better results than the models produced individually. Zhang (2003) concluded that when the linear ARIMA and the NARNet were fused they captured a greater degree of the relationship in the time series data.

In another example, Chow and Leung (1996) studied the ability to forecast the electric load based on weather compensation; hypothesizing that a NAR neural network could classify the nonlinear time-series and provide accurate forecast over time for the Hong Kong Island electric load profile. This weather compensation neural network proved to accurately predict the change of electric load consumption on day ahead. This methodology calculated more accurate load forecast with a 0.9% reduction in forecast error (Chow & Leung, 1996).

ANN Ensembles

Ensembles are combinations of classifiers. ANN ensembles ensure one is not limited by one neural network and its pre-determined structure; ensemble methods have therefore been devised in ways to fuse multiple ANNs together. Through this approach the inherent uncertainty in on network can be limited by combining the output with other networks. An ensemble, or classifier fusion, provides a flexible way to link multiple networks. Various ensemble constructions include multiple network architectures, same architecture trained with different algorithms, different initial random weights, or even different classifiers. Researchers have also suggested the combination of neural networks with traditional statistical classifiers. Kuncheva et al. (2003) show that the majority vote with dependent classifiers can potentially offer a dramatic improvement both over independent classifiers and over the individual accuracy of one ANN. Leap et al. (2008) demonstrated several fusion techniques were robust to correlation when they controlled for the level of correlation at various levels and that fusion always performed no worse than the worst classifier. Turnquist (2011) examined classifier fusion for hyperspectral imagery data. Although the mentioned fusion methods are parallel in nature, where multiple classifier outputs are fused, series fusion is possible as examined by Friesen et al. (2013) where the output of one classifier became the input of another. The research performed with ensemble

neural networks suggest that the researcher should not be satisfied with a structure until various methods are attempted and these models are brought together because, similar to human nature, diversity brings strength.

ANN with Vibrometry Data

Of most interest to this thesis, the current research into implementing ANNs from vibrometry sensors. An emerging technique, using the vibrations as they bounce off an object in order to classify that object into a particular state has seen some interesting progress. One area where researchers from the Georgia Institute of Technology have implemented this technique is with classifying electric utility poles as healthy or in need of repair (Stack, Harley, Springer, & Mahaffey, 2003). Wooden electric utility poles span the United States and transport the electric power long distances from their source to the customer. As time passes, the wooden poles, which comprise the majority of poles in the transmission and distribution network, will need to be replaced. In order to determine if the pole has structural deficiencies and requires replacement, Stack et al. (2003) suggested using a helicopter equipped with acoustic equipment to measure the vibrations received from the telephone poles as the helicopter flew by. The data set would then be passed through a trained neural network to classify the pole has healthy or deficient. The researchers discovered that this technique could save both time and money when compared to the traditional, man-hour intensive, process to climb, inspect, and redo. They have patented (Stack, 2003) their research and plan to implement their strategy across the expansive United States electric network.

In another approach to combine vibrometry for measurement with neural networks for processing, Castellini and Revei (2000) proposed a methodology to detect, localize, and characterize defects in mechanical structures. Using scanning laser Doppler Vibrometry

(SLDV), Castellini and Revei (2000) offers a non-intrusive technique to explore and evaluate the object under investigation. Castellini and Revei's (2000) methodology proved to be efficient to recognize defects and determine their depth in composite materials. Not only applicable to metallurgic structures, Turkish researchers, Turkoglu et al. (2003) processed Doppler sonographic signals measured during patient heart tests to determine if any heart valve diseases were present. The performance of this developed system proved to have a correct classification rate of 94% for abnormal and normal subjects.

ATR State of Affairs

Academic researchers who attempt to successfully classify systems overlap with military strategists wanting to identify targets. ATR complements both fields of study and presents an important evolving area of study for all concerned. Recent research in this field entails the processing and disposition of hyper-spectral images (HSI) (Smetek, 2007).

Many agencies have undertaken the initiative to explore ATR using simulated vibrometry data obtained from vehicles (Dierking, Heitkamp, Roth, & Armstrong, 2012). For the past decade various research studies have attempted to process and understand the data obtained from accelerometers during controlled experiments. Multiple avenues were explored, to include in-house government research and externally funded academic or contractor teams (Dierking et al., 2012).

These research teams reviewed two types of problems, identifying between vehicles and distinguishing engine types. The latter problem was solved with a high degree of precision by multiple teams using both probabilistic neural networks (PNN) and FFNNs. The three-class problem, identifying between three different vehicles, was attacked from various angles but none could reach the same level of precision as the engine classification problem. The techniques that

were explored included power spectral densities, Multi Angle-Centered Discrete Fractional Fourier Transform (MA-CDFRFT), and PCA (Dierking et al., 2012). The PCA analysis was implemented with hope of producing a dimensionality reduction before a Feed-Forward Neural Network (FFNN) would classify the targets. While the training data set resulted in close to perfect classification ability, when a test data set entered the equation the results dropped significantly (Dierking et al., 2012).

Crider and Kangas (2012) investigated ATR through four distinct, but related areas. Each approach has indicated that preliminary results being able to discriminate vehicle types with similar classes. The analysis has been made based on small datasets, and the performance under field conditions has not been investigated. Multiple groups noted data shortages as a limitation (Crider & Kangas, 2012). Although type discrimination has been anecdotally demonstrated, a significant amount of rework remains before a reliable & robust ATR system is realized.

Literature Review Summary

A number of techniques have been implemented to enable ATR on simulated laser vibrometry data. Time series neural networks were identified as an unexplored technique using the same data source as previous research. Of the various types of time series neural networks presented above, the nonlinear autoregressive neural network proved to be the best structure to produce the best results. The research presented in this paper focuses on the training, optimizing and implementation of NARNets to enable successful ATR.

III. Methodology

This section discusses the development of the neural network models using the vibrometry data to predict the source based solely on the time series data collected by accelerometers.

Scope and Data Description

The Air Force Research Laboratory's Sensor Directorate recently collected vibration data on three separate vehicles in various states of operation (Roth, 2013). There were three vehicles of interest, labeled: A, B, and C. Each vehicle had multiple spatially distributed accelerometers set-up during numerous replications to collect the vibration feedback from each vehicle. Each sensor provided a measurement for a specific axis (either x, y or z) for the vehicle during the sample run. The observations were collected at 10 KHz for 60 seconds which provided a time series stream of 600,000 points. During each run, multiple sensors monitored the vehicle from the front and the rear (appropriately designated), this resulted in a simultaneous collection of multiple sensor observations on multiple axes. For example, during run #13 on vehicle A, eleven sensors were fixed to various parts on the vehicle's front and were distributed as such: three accelerometers on the x-axis (sensors 9, 19 and 20), four on the y-axis (11, 13, 22, and 23), and four on the z-axis (10, 14, 21, and 24). This resulted in 48 different combinations of sensors which, when one sensor from each axis was combined, would provide a complete vibration reading. Table 1 lists the sensors which collected data at different locations during the experimental runs. Note, not every run had an identical sensor set-up and no data was available from the front of vehicle B in the x-axis. In all there are 212 data sets from different sensors and run numbers, this provided the data necessary produce the final classification algorithm.

Table 1 Data Sets

A

AF		
x-axis	y-axis	z-axis
1_15	1_11	1_10
1_19	1_13	1_14
1_20	1_22	1_21
1_9	1_23	1_24
		1_8
4_19	4_11	4_10
4_20	4_13	4_14
4_9	4_22	4_21
	4_23	
7_19	7_11	7_10
7_20	7_13	7_14
7_9	7_22	7_21
	7_23	7_24
13_19	13_11	13_10
13_20	13_13	13_14
13_9	13_22	13_21
	13_23	13_24
14_19	14_11	14_10
14_20	14_13	14_14
14_9	14_22	14_21
	14_23	14_24

AR		
x-axis	y-axis	z-axis
10_12	10_13	10_14
10_17	10_20	10_18
	10_23	10_21
		10_22
11_12	11_13	11_14
11_17	11_20	11_18
	11_23	11_21
		11_22
1_12	1_13	1_14
1_17	1_16	1_18
	1_20	1_21
	1_23	1_22
		1_24
4_11	4_10	4_14
4_12	4_13	4_18
	4_19	4_22
	4_20	4_9
	4_23	

B

BF		
x-axis	y-axis	z-axis
	11_23	11_20
		11_22
		14_21
		14_22
	15_23	15_21
		15_22
	1_23	1_20
		1_21
		1_22
	4_23	4_20
		4_21
		4_22

BR		
x-axis	y-axis	z-axis
10_17	10_19	10_14
10_23		10_20
		10_21
		10_22
		10_24
1_17	1_19	1_14
1_23		1_20
		1_21
		1_22
		1_24
4_17	4_19	4_14
4_23		4_20
		4_21
		4_22
		4_24
13_17	13_13	13_14
13_23	13_19	13_21
14_17	14_19	14_14
14_23		14_20
		14_21
		14_22
		14_24

C

CF		
x-axis	y-axis	z-axis
15_13	15_11	15_10
15_15	15_12	15_14
15_19	15_17	15_16
	15_23	15_18
		15_20
		15_21
		15_22
		15_24
18_13	18_11	18_10
18_15	18_12	18_14
18_19	18_17	18_16
	18_23	18_18
		18_20
		18_24
1_13	1_11	1_10
1_15	1_12	1_14
1_19	1_17	1_16
	1_23	1_18
		1_20
		1_21
		1_22
		1_24
4_13	4_11	4_10
4_15	4_12	4_14
4_19	4_17	4_16
	4_23	4_18
		4_20
		4_21
		4_22
		4_24

CR		
x-axis	y-axis	z-axis
2_24	2_16	2_17
		2_18

Data Nomenclature

In order to follow the process of neural network training and model description it is imperative to first introduce the data's nomenclature. There were three vehicle models which were used to collect the vibration data. On each vehicle were a number of sensors spread onto different parts of the vehicles. A number of test runs were used to collect the vibrations under a single engine condition, stationary and idle, while each sensor collected data for one axis. To ensure correct syntax throughout the collection process each data set was provided a unique designator. Table 2 lists the data designators and provides an example of the data nomenclature by model (A, B, or C), location (F for front or R for rear), run number (ordinal), and sensor number (ordinal). The example describes the type of vehicle, A, from run number 13 and sensor 9, which was collected on the vehicle's front in the x-axis.

Table 2 Data Nomenclature

Vehicle	Model	Location	Run #	Sensor #
	A	F (Front)	Multiple	Multiple
	B	R (Rear)		
	C			

Example: AF_13_9

Classification Algorithm Methodology

With the data provided, Table 1, the goal was to make a quick and accurate vehicle classifier model. Given the universe of three vehicles, the prediction would be based upon the minimum reconstruction error (Mean-Squared Error (MSE)) from the best neural networks from each vehicle across the three axes. MSE was defined to be the average of each target value

minus the network predicted value squared. Equation 5 depicts the calculation of NARNet (as seen in Figure 4 performance using the MSE value.

$$MSE = \frac{\sum_{1}^{n} (t_n - y(t_n))^2}{n} \qquad 4$$

Where $y(t_n)$ *is the nth NARNet predicted value and*

t_n *is the corresponding target value*

In order to make a prediction given a data stream it is necessary to train time series autoregressive neural networks to each sensor across the x, y and z-axis for each vehicle. With the resulting neural networks, one can then determine the best three neural network combinations, for a given axis, that produces the best true-positive rate across the data sets. The true-positive rate is defined to be the sum, across all three vehicles, of the fraction of correctly identified vehicles given a known data source of all data sets of interest. This can be pictured in Figure 6 as the sum of the green boxes. This example demonstrates a true positive rate of 5.45.

Data Set	"C"	"B"	"C"
AF_1_15	0.9	0	0.1
AF_1_19	0.8	0.1	0.1
BF_14_21	0	1	0
BF_14_22	0	0.75	0.25
CF_15_13	0	0	1
CF_15_15	0	0	1

Figure 6 True Positive Rate Defined

A seven step process, depicted in Figure 7, was followed in order to train, optimize, and validate the best model which ensured the highest true positive rate obtainable.

Step 1: entailed the data collection. This step was performed outside of the scope of this research project.

Step 2: consisted of data processing. The accelerometer sensor data from each run was processed into data vectors. This step was performed by AFRL/RYY.

Step 3: consisted of the neural network training. Networks were trained to all the testing data sets with various neural structures.

Step 4: consisted of finding the neural structure which had the best performance to the data set it was trained to with the goal of finding the best neural network for a given data set.

Step 5 was an optimization step and found the combination of neural networks across a single axis that resulted in the best true positive rate.

Step 6: Once the best networks were identified, algorithm verification was a spot check to ensure the classification algorithm worked sufficiently.

Step 7: used the best combination of networks discovered during step 5 and implemented the majority voting rule on the validation segments of the validation data sets.

Figure 7 depicts the seven step process, the corresponding data sets and the segments of those data sets used in the associated step.

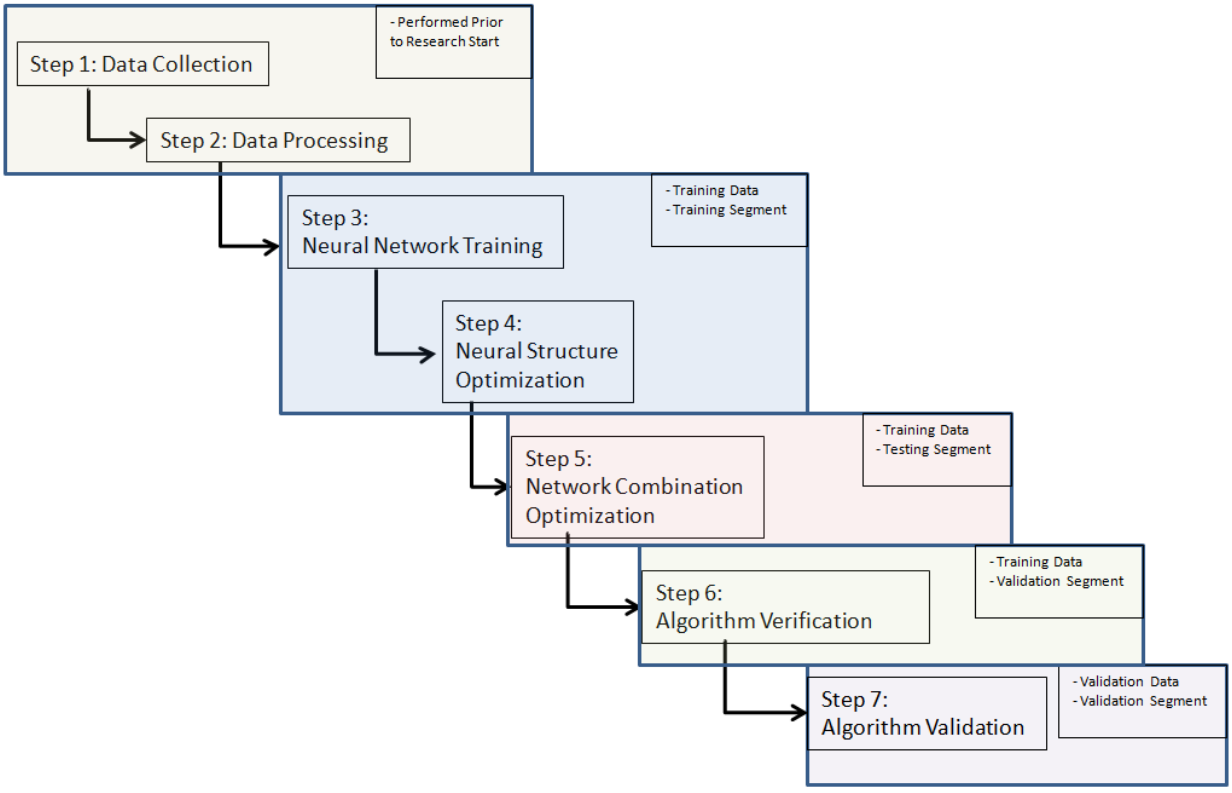


Figure 7 Research Methodology

Certain run numbers from each vehicle were separated for use as validation data when the neural networks were trained. The data sets removed, including all corresponding sensors, were run numbers 13 and 14 from the front of vehicle A, run numbers 1 and 4 from the rear of vehicle A, run numbers 1 and 4 from the front of vehicle B, run numbers 13 and 14 from the rear of vehicle B, and run number 4 from the front of vehicle C. Figure 8 shows the data sets used for training versus the data sets which were removed and used solely for validation.

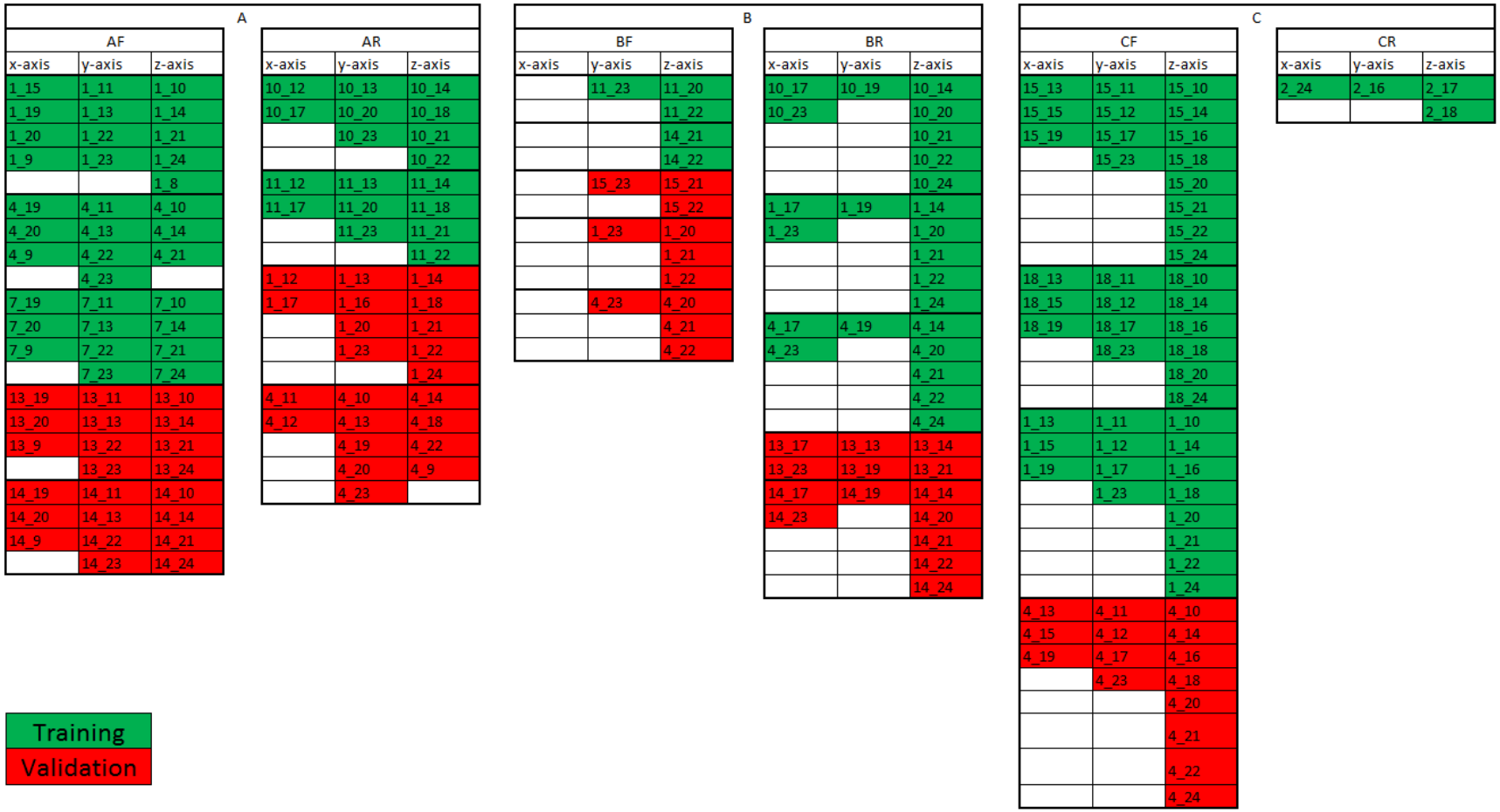


Figure 8 Training and Validation Data Sets

Neural Network Training

It was necessary to train multiple neural structures for each data set to determine the best set of characteristics which provided the lowest MSE (performance) for each data set. Only a small section of the data was examined for the training, testing and validation, of the ANNs. To be analogous to an operational environment, it was ensured that the data used to train and test the ANNs were taken from observations occurring before the model validation set. This length of data, rows 5,000 - 7,000, pulled from the 600,000 array was only 0.33% of the complete vector. Figure 9 depicts the various data segments of data used to train the neural networks, test for optimal neural network combinations and validate the resulting algorithm. As noted, vehicle B was only of length 300,000; therefore, the model limited the validation run up to point 300,000 to accommodate the all data sets.

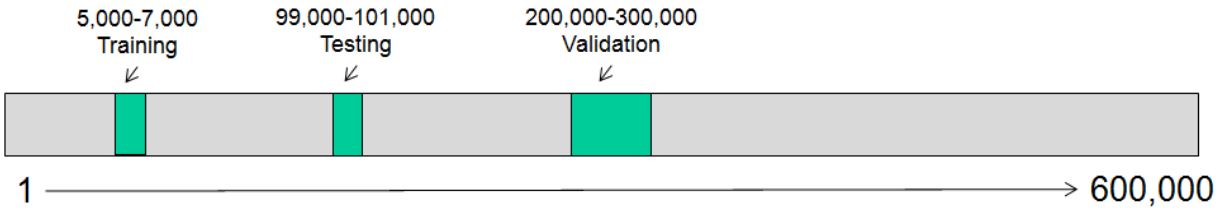


Figure 9 Data Set Segmentation

Neural Structure Optimization

For the training section of data, a network was trained with 5 different hidden nodes (5, 6, 7, 8, and 9) and 9 various lag lengths, (50- 90, by 5s). During the optimization routine, a neural network was trained to the data section given the hidden nodes and a lag length. Once trained the MSE was compared to previous neural networks from the same data set. The number of trained neural networks for each data set numbered 45 (5 x 9), each independent of one another consisting of different hidden nodes with different weights and various lag sizes. The neural

network structure which produced the best performance was assigned to that data set. This process was repeated for all the sensors, for each vehicle (front and rear) and each axis.

Across all axes and vehicles, 128 data sets (34 from the AF, 18 from the AR, 5 from the BF, 24 from the BR, 43 from the CF, and 4 from the CR) were used for training neural networks. During the neural structure optimization routine 5,760 (128 x 45) neural networks were trained. According to Step 1 of Figure 7, only the best performing network from each data set, as calculated by the MSE, remained, resulting in 128 neural networks that were pushed forward into step 3.

Figure 10 depicts the process of training neural networks for one data set, sensor 15 which monitored the front of vehicle C during run #1. As described above, the resulting neural networks were compared to neural networks with different characteristics (lag length, hidden nodes) trained to the same run number, sensor number and axis. The neural network with the best performance was assigned to that data set. In this example the neural network trained using 6 hidden nodes and a lag length of 50 was the best structure for the data set CF_1_15.

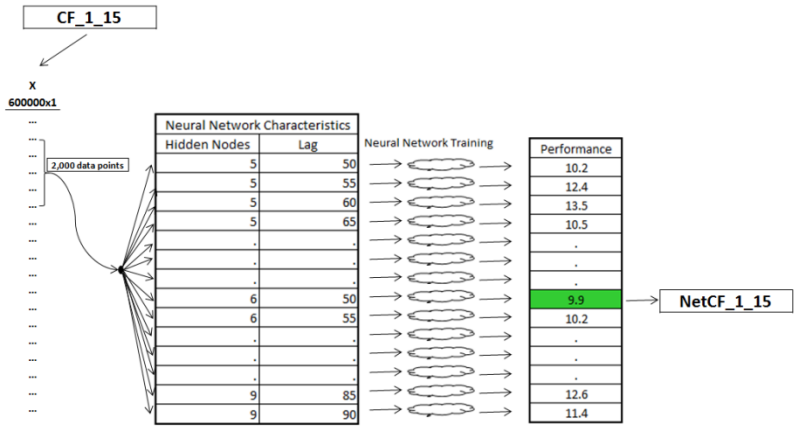


Figure 10 Neural Network Training

Network Combination Optimization

Step 3 of Figure 7 identifies the combination of networks that provide the best true-positive rate within each axis from a vehicle's location. For this procedure the three axes were separated and the front neural networks were treated as different sets than the neural networks trained to the vehicles' rear. With 128 networks, the goal was to down-select and only push forward the best 18 networks (9 front and 9 rear trained networks, 3 per axis). For example, we will examine the x-axis networks trained to the rear data sets. Figure 11 shows the highlighted area of interest.

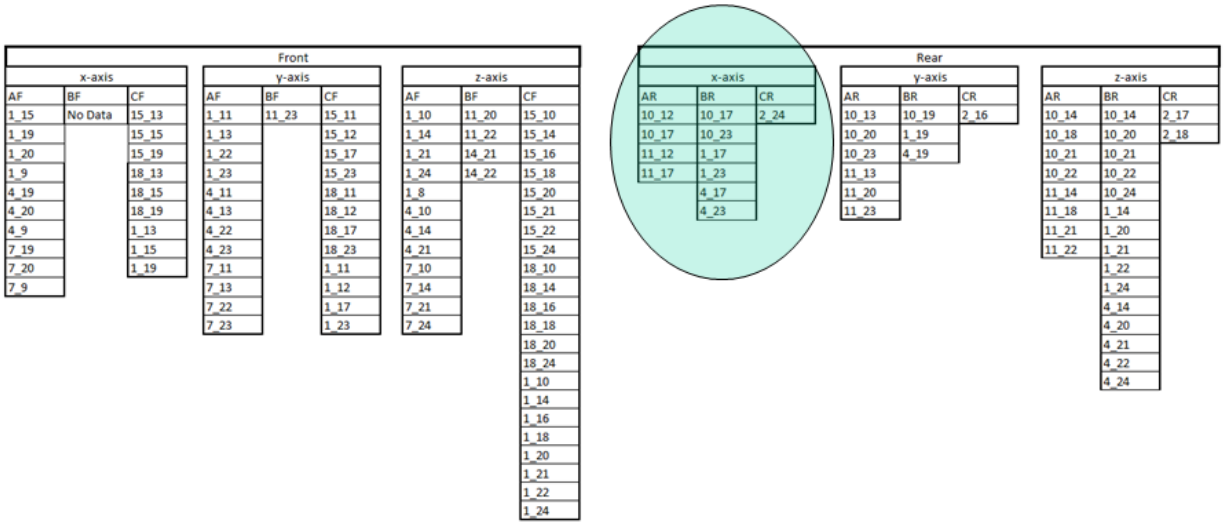
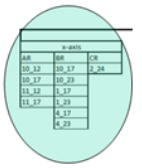


Figure 11 B Rear Training Data

From the Rear data sets in the x-axis there were 4 networks from the rear of vehicle A, 6 networks from the rear of vehicle B and 1 network from the rear of vehicle C. This resulted in 24 unique permutations of networks that must be tested. Figure 12 shows the labeling of networks and the resulting permutations.



Data	Neural Networks
AR	
10_12	NetAR_10_12
10_17	NetAR_10_17
11_12	NetAR_11_12
11_17	NetAR_11_17
BR	
10_17	NetBR_10_17
10_23	NetBR_10_23
1_17	NetBR_1_17
1_23	NetBR_1_23
4_17	NetBR_4_17
4_23	NetBR_4_23
CR	
2_24	NetCR_2_24

Network Combinations			
1	NetAR_10_12	NetBR_10_17	NetCR_2_24
2	NetAR_10_12	NetBR_10_23	NetCR_2_24
3	NetAR_10_12	NetBR_1_17	NetCR_2_24
4	NetAR_10_12	NetBR_1_23	NetCR_2_24
5	NetAR_10_12	NetBR_4_17	NetCR_2_24
6	NetAR_10_12	NetBR_4_23	NetCR_2_24
7	NetAR_10_17	NetBR_10_17	NetCR_2_24
8	NetAR_10_17	NetBR_10_23	NetCR_2_24
9	NetAR_10_17	NetBR_1_17	NetCR_2_24
10	NetAR_10_17	NetBR_1_23	NetCR_2_24
11	NetAR_10_17	NetBR_4_17	NetCR_2_24
12	NetAR_10_17	NetBR_4_23	NetCR_2_24
13	NetAR_11_12	NetBR_10_17	NetCR_2_24
14	NetAR_11_12	NetBR_10_23	NetCR_2_24
15	NetAR_11_12	NetBR_1_17	NetCR_2_24
16	NetAR_11_12	NetBR_1_23	NetCR_2_24
17	NetAR_11_12	NetBR_4_17	NetCR_2_24
18	NetAR_11_12	NetBR_4_23	NetCR_2_24
19	NetAR_11_17	NetBR_10_17	NetCR_2_24
20	NetAR_11_17	NetBR_10_23	NetCR_2_24
21	NetAR_11_17	NetBR_1_17	NetCR_2_24
22	NetAR_11_17	NetBR_1_23	NetCR_2_24
23	NetAR_11_17	NetBR_4_17	NetCR_2_24
24	NetAR_11_17	NetBR_4_23	NetCR_2_24

Figure 12 Neural Network Combinations by Axis

To find the best neural network combination which resulted in the highest true positive rate two sections from the testing segment, as seen in Figure 9, of length 1,000 each, from one of the corresponding data sets, were passed through the combinations of neural networks. The section of data was passed through the three neural networks all attempted to reconstruct the vehicles vibrational data provided their own, independent, characteristics (hidden nodes/weights/lag). Of the three neural networks the one which resulted in the lowest MSE was deemed to be the winner and the vehicle would be classified accordingly. If the winning neural network was from the same vehicle then that data sample would be recorded as a true positive, otherwise a false positive was recorded. Figure 13 depicts this process of pushing through two sections of data, both from the data set AR_10_12, through one combination of neural networks: (NetAR_10_12/NetBR_10_17/NetCR_2_24).

In both examples NetAR_10_12 produced in the smallest MSE, resulting in a correct classification for both sections of data (9.9-10 seconds and 10-10.1 seconds).

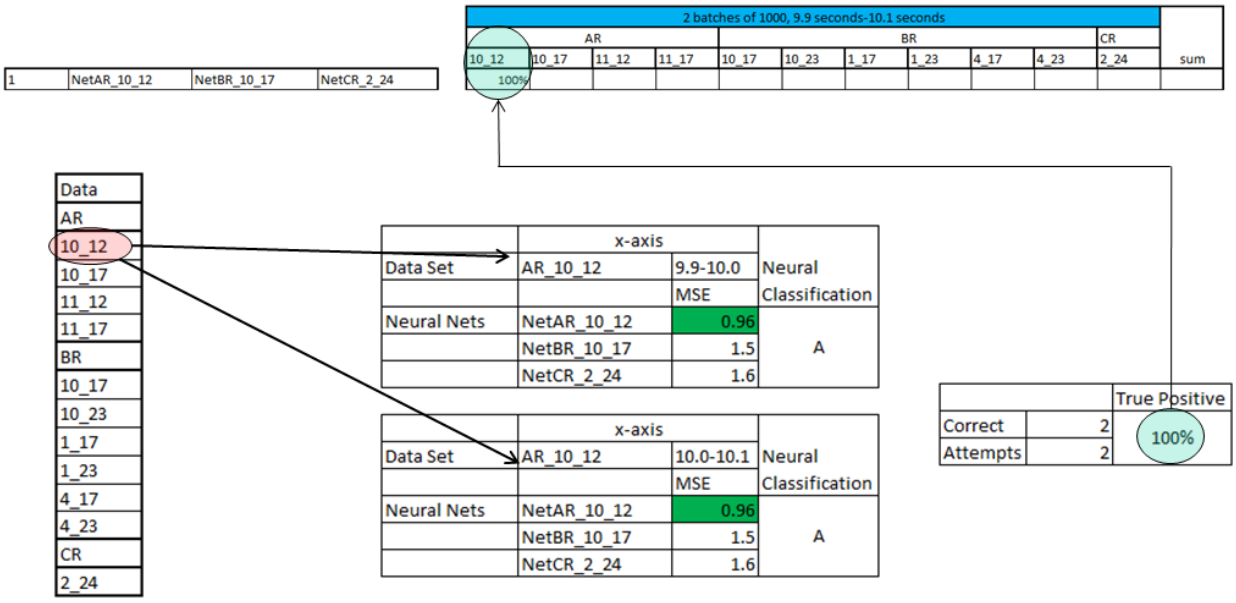


Figure 13 Neural Network Classification

Figure 13 depicts this process for only one combination of neural networks from the rear of vehicles A, B, and C in the x-axis. When each of the data sets were separately passed through this combination, the vehicle associated with the neural network which produced the smallest reconstruction error would result in a classification, if the classified vehicle was in fact the vehicle of origin then a correct classification would be achieved. In the example shown in Figure 14, the combination that resulted in the highest true positive classification rate was the neural networks NetAR_10_12, NetBR_10_17 and NetCR_2_24. These neural networks produced a 100% true positive rate in this example. Figure 14 below shows all the neural network combinations and the resulting true positive rate when passed through their

corresponding training data. The best combination in this example was the first combination resulting in a 100% correct classification rate. .

	Network Combinations			True Positive Rate 2 batches of 1000, 9.9 seconds-10.1 seconds											
				AR				BR						CR	
				10_12	10_17	11_12	11_17	10_17	10_23	1_17	1_23	4_17	4_23	2_24	sum
1	NetAR_10_12	NetBR_10_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
2	NetAR_10_12	NetBR_10_23	NetCR_2_24	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	9
3	NetAR_10_12	NetBR_1_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
4	NetAR_10_12	NetBR_1_23	NetCR_2_24	100%	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	9
5	NetAR_10_12	NetBR_4_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
6	NetAR_10_12	NetBR_4_23	NetCR_2_24	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	9
7	NetAR_10_17	NetBR_10_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
8	NetAR_10_17	NetBR_10_23	NetCR_2_24	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	7
9	NetAR_10_17	NetBR_1_17	NetCR_2_24	100%	100%	100%	100%	100%	0%	100%	50%	100%	100%	100%	9.5
10	NetAR_10_17	NetBR_1_23	NetCR_2_24	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	9
11	NetAR_10_17	NetBR_4_17	NetCR_2_24	100%	100%	100%	100%	100%	50%	100%	50%	100%	100%	100%	10
12	NetAR_10_17	NetBR_4_23	NetCR_2_24	100%	0%	100%	50%	100%	100%	100%	100%	100%	100%	100%	9.5
13	NetAR_11_12	NetBR_10_17	NetCR_2_24	100%	100%	100%	50%	100%	100%	100%	100%	100%	100%	100%	10.5
14	NetAR_11_12	NetBR_10_23	NetCR_2_24	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	9
15	NetAR_11_12	NetBR_1_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
16	NetAR_11_12	NetBR_1_23	NetCR_2_24	100%	0%	50%	0%	100%	100%	100%	100%	100%	100%	100%	8.5
17	NetAR_11_12	NetBR_4_17	NetCR_2_24	100%	100%	100%	100%	100%	50%	100%	100%	100%	100%	100%	10.5
18	NetAR_11_12	NetBR_4_23	NetCR_2_24	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	9
19	NetAR_11_17	NetBR_10_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
20	NetAR_11_17	NetBR_10_23	NetCR_2_24	100%	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	9
21	NetAR_11_17	NetBR_1_17	NetCR_2_24	100%	100%	100%	50%	100%	100%	100%	100%	100%	100%	100%	10.5
22	NetAR_11_17	NetBR_1_23	NetCR_2_24	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	9
23	NetAR_11_17	NetBR_4_17	NetCR_2_24	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	11
24	NetAR_11_17	NetBR_4_23	NetCR_2_24	100%	0%	100%	0%	100%	100%	100%	100%	100%	100%	100%	9

Figure 14 Neural Network Best Combination Rear X-axis Networks

This procedure was completed for all the combinations of neural networks through each data set related to the same location on the vehicle and across the three axes. As described this process resulted in the best combination of three neural networks (one from each vehicle) from the front and rear portion of the vehicle in the x-axis, y-axis, and z-axis.

Table 3 lists the total number of combinations of neural networks tested in order to derive the best combination from each vehicle side and axis. When testing the x-axis of the vehicles' front, the BR networks were used to replace the missing data sets from BF.

This process of determining the best combination of neural networks enabled the model to establish the relative reconstructive strength of each neural network when compared to the neural networks from the other vehicles in the same axis. Ideally, a network trained to the front of vehicle A would produce a small MSE when given an AF data set and a large MSE when

given a BF or CF data set, and similarly for neural networks trained to the BF and CF respectively, which would enable the model to identify the correct vehicle by the smallest MSE. No preference was given to sensor or run number, meaning that the sensor location on the vehicle which may have been closer to the engine compartment was not given any higher rating than a sensor located far from the engine. In addition, the characteristic of the run, which determined at what level the engine was operating, was similarly not given a preference. Figure 15 explains the neural networks and their trained data sets which resulted from the network combination optimization stage.

Table 3 Neural Network Combinations per Axis

	Front			Rear		
	x	y	z	x	y	z
Combinations	540	576	1584	24	18	240

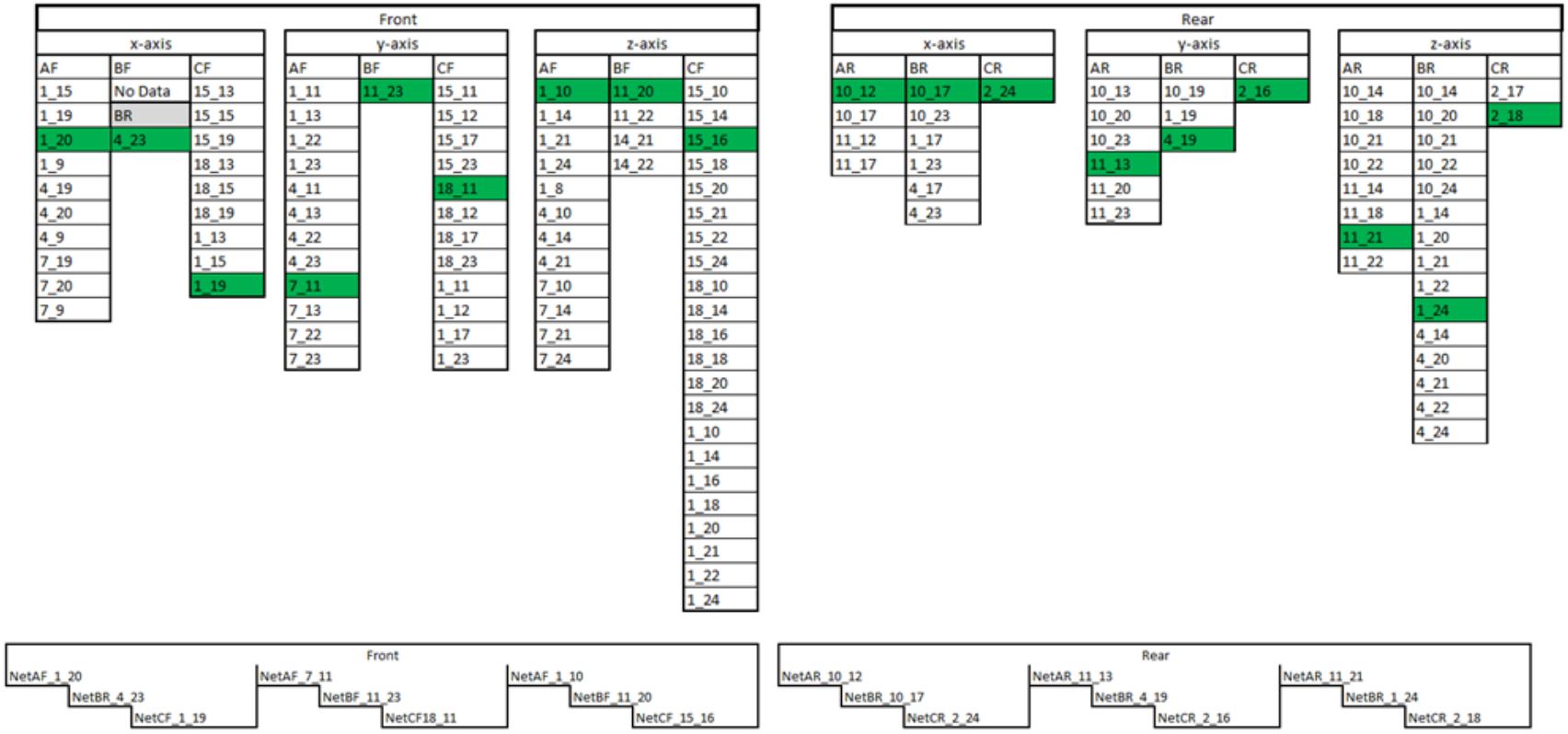


Figure 15 Neural Network Best Combinations

Data Set Classification

In all, there were 18 neural networks, 3 networks from 2 locations (front/rear) in the 3 axes (x, y, and z), identified and pushed forward to step 6 in the classification process, Algorithm Verification, Figure 16 lists the data sets which the 18 neural networks were trained.

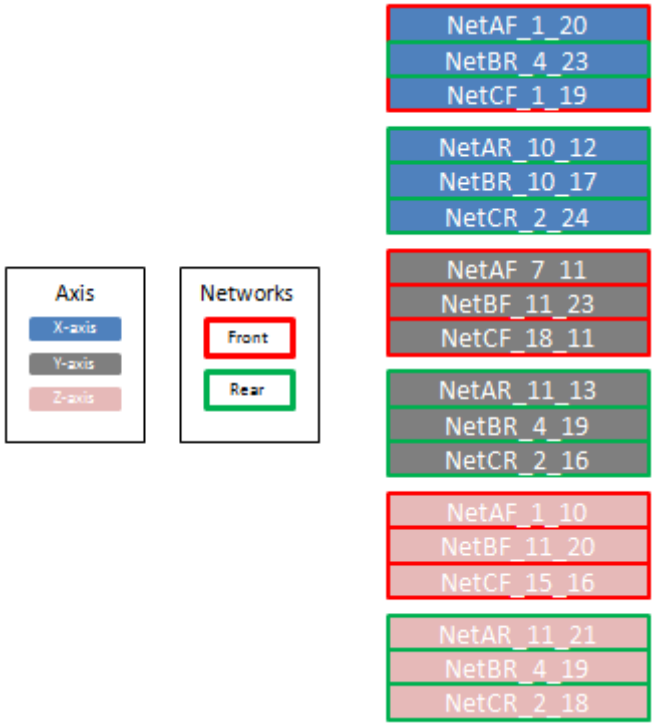


Figure 16 Neural Networks Associated Data Sets

The data set AF_13_9 will be used to demonstrate vehicle classification from the best neural network combination. Sensor 9 from run #13 for the AF monitored the x-axis; therefore, it will be pushed through the best combination from the front networks in the x-axis that resulted during the network combination optimization routine in order to derive a classification. These networks are located in the top box of Figure 16: NetAF_1_20, NetBR_4_23, and NetCF_1_19. Table 4 illustrates the resulting classification when a batch size of 100 of the AF_13_9, from the validation data segment, is pushed through the neural networks associated with the same vehicle side (front) and axis (x-axis). NetAF_1_20 calculated the lowest MSE and therefore vehicle A

was declared the vehicle. The original data set was from vehicle A resulting in a correct classification.

Table 4 A Front Run #13 Sensor 9 Classification Example

	x-axis		
Data Set	**AF_13_9**	20.00-20.01	Neural
		MSE	Classification
Neural Nets	NetAF_1_20	0.96	A
	NetBR_4_23	1.5	
	NetCF_1_19	1.6	

This process of classification was repeated for 1,000 batches, each of size 100. Table 5 shows the resulting number and percentage of correctly classified sections and incorrectly classified sections for sensor 9 during run #13 of the AF. Over 77% of the sections were correctly classified. Each data batch of 100 was only one hundredth of one second.

Table 5 AF_13_9 Classification

Time: 20.00 seconds - 29.99 seconds, .01 sec increments					
Verification	Front Data	Front x-axis networks			Batch Size: 100
	"A"	**"B"**	**"C"**		**Total Samples**
AF_13_9	773	67	160		1000
	77.30%	6.70%	16.00%		100%

Data Exemplars

To create a robust classification algorithm the goal was to use information from all three axes available. The data was collected from accelerometers spread across the vehicle during a run number, each sensor monitoring a different axis of interest. For proof of concept purposes, each sensor combination (one from each axis) for a given run number was treated as an independent data exemplar. Table 6 lists run #13 for the AF and the sensors which collected data

during this run for each axis. The assumption was made that the data was collected at the same frequency and started/stopped at the same time for each sensor. As described in the table below, there were three sensors for the x-axis, four for the y-axis, and four for the z-axis. In all, this resulted in 48 unique combinations (3 x 4 x 4) of sensors for run #13 of the AF.

Table 6 Data Sets: A Front Run #13 All Sensors

AF		
X-axis	Y-axis	Z-axis
13_9	13_11	13_10
13_19	13_13	13_14
13_20	13_22	13_21
	13_23	13_24

Classification Rule

With three axes all contributing toward classification, decision fusion was examined through a majority voting scheme to determine the final vehicle classification. If two or more networks from each axis claimed the same vehicle then that vehicle would be classified. If none of the axis agreed (all axes had different vehicle neural networks win) then a non-declaration would be issued.

Classification Verification

Classification verification step used the training data sets but the validation segment. Most neural networks, because 128 were trained, did not make the final cut of 18 neural networks. Therefore, although a neural network was trained to these testing data sets, over 80% of the data sets' vibration information was not accounted for in any final model neural network, but was appropriately separated from validation data sets. The validation data sets had no neural networks trained to them during step 3, as seen in Figure 7.

To examine the classification verification the AF run #13 will be examined. Each of the combinations of run #13 was passed through the nine neural networks trained to the fronts of the vehicles in each axis. The first data exemplar, sensor 13_9, 13_11, and 13_10 were each passed through the group of neural networks trained to their respective axis. Data set 13_9, from the x-axis, was passed to neural networks trained from AF_1_20, BR_4_23, and CF_1_19. Data set 13_11, from the y-axis was passed to neural networks trained from AF_7_11, BF_11_23, and CF_18_11. Data set 13_10, from the z-axis was passed to neural networks trained from AF_1_10, BF_11_20, and CF_15_16. Table 7 lists each data set within the specified date exemplar and associated neural networks which created the system classification. This example shows an incorrect classification of the vehicle C even though the data provided came from the front of vehicle A.

Table 7 AF Run #13 Classification Example

	x-axis		Neural	System
Data Set	AF_13_9			
		MSE	Classification	Classifcation
Neural Nets	NetAF_1_20	0.96		
	NetBR_4_23	1.5	A	
	NetCF_1_19	1.6		
	y-axis			
Data Set	AF_13_11			
		MSE		
Neural Nets	NetAF_7_11	1.26		
	NetBF_11_23	1.35	C	C
	NetCF_18_11	0.91		
	z-axis			
Data Set	AF_13_10			
Neural Nets	NetAF_1_10	1.11		
	NetBF_11_20	1.25	C	
	NetCF_15_16	1.01		

In the Table 7 example each neural network was given a batch size of 100 corresponding from the same time period from the respective data set. The x-axis data was provided to the x-axis trained neural networks. Of the three x-axis trained neural networks, the vehicle from which the neural network that produced the lowest MSE would be classified from that data set. The process was repeated for the y-axis and z-axis. The respective data sets were sent to the matching axis neural networks. The winning networks would result in an appropriate axis classification. A majority voting rule was used to dictate which vehicle was classified. In order for the vehicle A to be named from data sets AF_13_9, AF_13_11, and AF_13_10, then at least two A neural networks across the three axes would have to result in the best performance in its axis. The neural network trained to the CF in the y-axis resulted in the best performance (lowest MSE) when compared to the neural networks trained to AF and BF when given a section of batch size 100 from the data set AF_13_11 and the neural network trained to CF in the z-axis resulted in the best performance when compared to the neural networks trained to BF and AF when given a batch size of 100 from the data set AF_13_10. Vehicle A would not be named and instead the model would incorrectly classify C as the vehicle of interest no matter the result of the x-axis neural networks. The green colored box represents the resulting neural network classification with the far right box listing the system classification of the C vehicle.

Table 8 lists the resulting true-positive performance when sections of data from AF run 13 of length 100 were given to the neural networks to classify the vehicle. Run #13 of the AF had 48 unique data exemplars and each exemplar had 1,000 non-overlapping data sections classified, this resulted in 48,000 classifications.

Table 8 AF_13 Majority Voting Confusion Matrix – Front Networks

Time: 20.00 seconds - 29.99 seconds, .01 sec increments					
Verification	Front Data	Front Networks		Batch Size: 100	
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AF_13	39904	2951	1383	3762	48000
	83.13%	6.15%	2.88%	7.84%	100%

As seen from AF run 13 the true-positive percentage was 83.13%. Over 83% of the time, two of three or all three neural networks trained to the AF resulted in the smallest MSE when compared to the neural networks trained to the BF and the CF for each axis. This percentage resulted when front data exemplars were passed through the neural networks trained to the front data sets. During real world operations, one would not know if the incoming data was from the front or the rear of a particular vehicle. Table 9 lists the resulting classifications when the AF data exemplars were passed through the networks trained to the rear data sets.

Table 9 AF_13 Majority Voting Confusion Matrix – Rear Networks

Time: 20.00 seconds - 29.99 seconds, .01 sec increments					
Verification	Front Data	Rear Networks		Batch Size: 100	
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AF_13	35005	6788	827	5380	48000
	72.93%	14.14%	1.72%	11.21%	100%

There was a 10% reduction in correct classification when the front data exemplars from the AF run #13 were passed through the rear networks versus the front networks. In another example, data exemplars from the BR went from over 90% correct classification from the rear networks to a 0% correct classification from the front networks.

Methodology Verification

Verification and validation are two separate but necessary steps in the development of any model to ensure it meets the requirements and specifications. Methodology verification encompasses the question, "did I build the model correctly".

The model is verified in a step-by-step fashion. First, the training phase was examined to determine if the neural networks were build properly. MatLab® generates code after utilizing their graphical user interface NTSTool (Beale, Hagan, & Demuth, 2013). This tool allows users to specify the characteristics of the various types of time series neural networks available. After selecting the NAR network, the underlying code was automatically generated. This code was manipulated which allowed for the training model to select the best neural structure for each data set. This module was repeated for each data set of interest resulting in multiple trained NAR networks for each vehicle across all three axes. The training phase and its associated NAR networks could be successfully verified in this regard.

Attempting to find the set of networks for each axis from the two sides of the vehicles which would produce the best classification an optimization step was developed. Figure 17 defines the goal of this step given the 128 trained neural networks down to the best 18 networks.

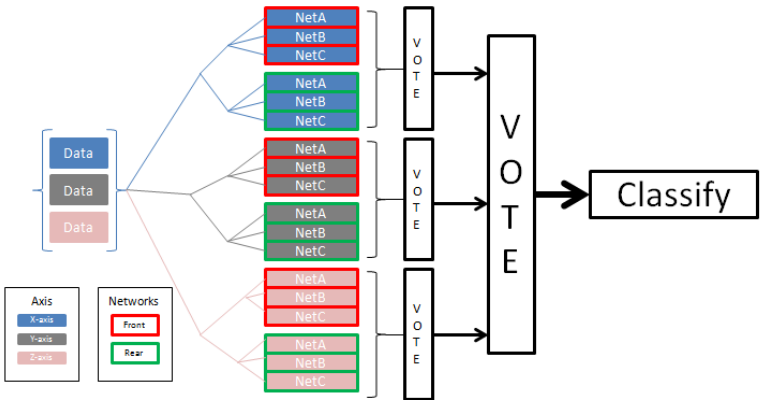


Figure 17 Step 5 Neural Network Combination Optimization

Step 5, the network combination optimization phase, could be verified easily as well. During this module, a section of code from the available data samples in a particular axis was sent through a combination of three networks, one from each vehicle from one axis, to generate the networks' performance generate via MatLab®. This was done separately for both areas of the vehicles, front and rear, and across all three axis. Using 'For' loops and 'If' statements, MatLab® enabled this optimization phase to run without interference. All that was required for user input was to define the data sets to send through the defined combinations of networks. Underlying this optimization was the routine that the smallest mean-square error resulted in a classification. If the network which was derived from the vehicle provided resulted in the smallest performance metric, then a true-positive was recorded. Each data set was sent through a specific combination of networks then the overall true-positive rate was recorded. The combination of networks that maintained the best overall true-positive rate (which was calculated by summing across all the data sets) would be declared the best combination. If two combinations resulted in a tie (the same true-positive rate) then the first combination tested would remain the winner. Satisfied with the verification of the optimization phase, the testing phase proceeded.

As described above in the methodology section, the classification model selected was the compilation of the six optimized network combinations. A network combination consisted of three NAR networks trained to one axis each from a different vehicle. The three network combinations from the vehicles' front were combined to the three network combinations from the vehicles' rear. Together these six network combinations included a total of eighteen trained neural networks. Similar to the testing phase, it was necessary to define all the combinations of sensors which encompassed a data sample. Each vehicle consisted of multiple run numbers,

each with different characteristics and multiple sensors collecting data across different points of the vehicle. To test the performance of the model, any combination of three sensors, one from each axis, from one part of the vehicle (front or rear) was defined as a data set. For example, run number 13 of the AF had 48 unique combinations of sensors. One data set included sensor #9 from the x-axis, sensor #11 from the y-axis, and sensor #10 from the z-axis. Together this data set was sent to the performance module to determine if the networks would identify vehicle A as the vehicle (a true positive), claim the vehicle of origin was a different vehicle, or determine it was unable to make a classification (each network in an axis claimed a different vehicle). To step through the verification of the testing phase, it was possible to segment each performance module to verify the correct network was being identified as the winner. Once this was accomplished, because the majority rule was in effect, the code was tasked to add together the number of identifications for each vehicle then pass the winning vehicle back to the testing phase via a confusion matrix. Through iterations of various data sets, I was able to verify the model was built correctly.

There were two main functions created in MatLab® to process the data and produce an answer. The first function, known as testnets, would load the data and the neural networks, then cycle through the various combinations of sensors and send these unique sets to the performance function. Testnets would identify the vehicle of origin along with the neural networks associated with that vehicle. This function would also pass to the performance function the incorrect neural networks, enabling the performance function to know if a true positive was obtained or a false positive was the result. Every time the performance function was called, it would cycle through a pre-determined number of non-overlapping data batches. Three performance functions were created to cycle through data batches of size 100, 500 and 1,000.

Figure 18 lists how the data and neural networks were loaded into the testnets function before they were sent to the performance function.

```
load A_x.mat
A_x=A;
load B_x.mat
B_x=B;
load C_x.mat
C_x=C;
load A_y.mat
A_y=A;
load B_y.mat
B_y=B;
load C_y.mat
C_y=C_y;
load AF_z.mat
A_z=A;
load BF_z.mat
B_z=B;
load CF_z.mat
C_z=C;
load ntsnets_x.mat
load ntsnetsC_x.mat
netA_xR=net17;
netB_xR=Bnet1;
netC_xR=Cnet13;
netA_xF=net9;
netB_xF=Bnet10;
netC_xF=Cnet9;
load ntsnets_y.mat
load ntsnetsC_y.mat
netA_yR=net24;
netB_yR=Bnet10;
netC_yR=Cnet17;
netA_yF=net17;
netB_yF=Bnet1;
netC_yF=Cnet5;
load ntsnets_zrear.mat
load ntsnetsC_zrear.mat
netA_zR=net7;
netB_zR=Bnet17;
netC_zR=Cnet2;
load ntsnets_zfront.mat
load ntsnetsC_zfront.mat
netA_zF=net9;
netB_zF=Bnet1;
netC_zF=Cnet3;
```

Figure 18 Testnets Load Data and Neural Networks

Figure 19 shows the process of testing a combination of data sets to determine the neural networks' performance in correctly identifying run 10 of the AF. Lines 59 through 61 set the unique combination of sensors from run #10 of the AR. All unique combination of sensors be tested with the for loops. Lines 62 through 79 set the known correct and the known negative neural networks. All of the neural networks trained to vehicle A front or rear were clearly defined and set correctly. In this example, line 82 calls the performance function TN500con. The 500 represents the performance function which tests non-overlapping batch sizes of 500. The result of the performance function is returned in the variable "confusion". This matrix embodied in it the number of true positive results, the number of false positive results (distinguished between each false vehicle) and the number of times no vehicle met the majority rule (all three axis had a neural network from a different vehicle with the best performance).

```
52    %AF 13
53 -  j=0;
54 -  for t=1:3
55 -      for u=1:4
56 -  j=j+1;
57 -  k=0;
58 -          for v=1:4
59 -              data=A_x(t);
60 -              datay=A_y(u);
61 -              dataz=A_z(v);
62 -              netL=netA_xF;
63 -              netLy=netA_yF;
64 -              netLz=netA_zF;
65 -              netN=netB_xF;
66 -              netNy=netB_yF;
67 -              netNz=netB_zF;
68 -              netN2=netC_xF;
69 -              netN2y=netC_yF;
70 -              netN2z=netC_zF;
71 -              netLo=netA_xR;
72 -              netLyo=netA_yR;
73 -              netLzo=netA_zR;
74 -              netNo=netB_xR;
75 -              netNyo=netB_yR;
76 -              netNzo=netB_zR;
77 -              netN2o=netC_xR;
78 -              netN2yo=netC_yR;
79 -              netN2zo=netC_zR;
80
81
82 -  confusion=TN100con(data,netL,netN,netN2,datay,netLy,netNy,netN2y,dataz,netLz,netNz,netN2z,...
83                        netLo,netNo,netN2o,netLyo,netNyo,netN2yo,netLzo,netNzo,netN2zo)
84 -  k=k+1;
85 -  n=n+1;
86 -  Vtracker(1,1)=Vtracker(1,1)+confusion(1);
87 -  Vtracker(1,2)=Vtracker(1,2)+confusion(2);
88 -  Vtracker(1,3)=Vtracker(1,3)+confusion(3);
89 -  Vtracker(1,4)=Vtracker(1,4)+confusion(4)
90
91 -          end
92 -      end
93 -  end
94 -     savefile2='Front100Perform.mat';
```

Figure 19 Testnets Function Set Data and Neural Networks

The variable “tracker” would maintain a running total of the number of false positives and true positives associated with each vehicle enabling the program to output a confusion matrix from every sensor combination within each run from the three vehicles.

The performance function of interest in this example (“TN500con”) had the data and all eighteen networks passed to it. Through the transfer from one function to the other in MatLab® one is able to maintain the placement of each variable being transferred. All of the networks associated with the data source were known as “netL” while all the incorrect networks were known as “netN”. With eighteen networks, it was necessary to maintain healthy bookkeeping to ensure the function was build correctly. The data was passed from each function in the format of a cell. This enabled large amounts to move within MatLab® with ease. Once the data was called in the TN500con it was necessary to reformat from a cell into a double. The function “cell2mat” enabled this to occur. Figure 20 depicts the function TN500 and the process of calculating the reconstruction error each neural network obtained from the data source. Lines 27, 32, and 37 were these reconstruction errors from the networks netL, netN, and netN2 respectively. In this example, knowing the source data was from AR, netL was the best neural network trained to the x-axis of the AF. This reconstruction error calculation was repeated for all of the eighteen networks pass to the function TN500con.

```
function [Confusion] = TN500con(data,netL,netN,netN2,...
                    datay,netLy,netNy,netN2y,dataz,...
                    netLz,netNz,netN2z,netLo,netNo,...
                    netN2o,netLyo,netNyo,netN2yo,netLzo,netNzo,netN2zo)
truepos=0;
falsepos=0;
falsepos2=0;
falsepos3=0;
falsepos4=0;
Correct=0;
wrong1=0;
wrong2=0;
wrong3=0;

test1=cell2mat(data);
test1y=cell2mat(datay);
test1z=cell2mat(dataz);
for x = 400:598
test2=test1(1+x*500:500 +x*500,1);
test2y=test1y(1+x*500:500 +x*500,1);
test2z=test1z(1+x*500:500 +x*500,1);
targetSeries = tonndata(test2,false,false);

[inputsL,inputStatesL,layerStatesL,targetsL] = preparets(netL,{},{},targetSeries);
outputsL = netL(inputsL,inputStatesL,layerStatesL);
errorsL = gsubtract(targetsL,outputsL);
performanceL = perform(netL,targetsL,outputsL);

[inputsN,inputStatesN,layerStatesN,targetsN] = preparets(netN,{},{},targetSeries);
outputsN = netN(inputsN,inputStatesN,layerStatesN);
errorsN = gsubtract(targetsN,outputsN);
performanceN = perform(netN,targetsN,outputsN);

[inputsN2,inputStatesN2,layerStatesN2,targetsN2] = preparets(netN2,{},{},targetSeries);
outputsN2 = netN2(inputsN2,inputStatesN2,layerStatesN2);
errorsN2 = gsubtract(targetsN2,outputsN2);
performanceN2 = perform(netN2,targetsN2,outputsN2);
```

Figure 20 Performance Function

With all of the eighteen reconstruction errors calculated, then the best performing network from each axis was identified. Figure 21 shows the number of "If" statements required to determine a true positive within one axis (netL or netLo winning). This process was repeated for all vehicles across the x, y and z axis.

```
trueposx=0;
falsepos1x=0;
falsepos2x=0;
if performanceL<performanceLo
    if performanceL<performanceN
        if performanceL<performanceN2
            if performanceL<performanceNo
                if performanceL<performanceN2o
                    trueposx=1;
                end
            end
        end
    end
else
    if performanceLo<performanceN
        if performanceLo<performanceN2
            if performanceLo<performanceNo
                if performanceLo<performanceN2o
                    trueposx=1;
                end
            end
        end
    end
end
```

Figure 21 True Positive Calculation

The trueposx, falsepos1x, falsepos2x variables allowed the function to keep track of the winning neural network from the x-axis. Once this process was repeated for all the networks for each axis then the winning vehicle from each axis was determined. To satisfy the majority rule one final calculation was required. Figure 22 shows the final calculation which determined which vehicle was classified by the performance function for the particular batch of data.

```
teststat(1:4)=0;

teststat(1)=trueposx+trueposy+trueposz;
teststat(2)=falsepos1x+falsepos1y+falsepos1z;
teststat(3)=falsepos2x+falsepos2y+falsepos2z;
if teststat(1)>1
    Correct=Correct+1;
elseif teststat(2)>1
    wrong1=wrong1+1;
    elseif teststat(3)>1
    wrong2=wrong2+1;
else
    wrong3=wrong3+1 ;
end
end
Confusion(1:4)=0;
Confusion(1)=Correct;
Confusion(2)=wrong1;
Confusion(3)=wrong2;
Confusion(4)=wrong3;
```

Figure 22 Majority Rule Calculation

Line 384 shows that the variable teststat was set to zero after each iteration. Teststat(1) was a true positive result while teststat(2) and teststat(3) were false positives. The variable Confusion was an array which kept track of the total number of false positives and true positives. Line 396 would trigger the variable wrong3 to increase by 1. This variable kept track on non-declarations, or each axis resulted in a different vehicle being claimed.

As seen in line 18 of Figure 20, this process was repeated for a pre-determined number of times to allow for non-overlapping batches of the same data set to be tested. This example shows the performance function, TN500con, cycled through 199 non-overlapping batches of the data set provided from the function of origin and sent back to the original function an array consisting of the number of true-positives, false-positives and non-declarations.

After reviewing the code line by line in a logical process, confidence in the classification process is achieved.

IV. Analysis

Vehicle Classification Algorithm

Step 7, algorithm validation, encompassed the main objective of the problem, to correctly identify the vehicle of origin from vibrometry data in a time series format. The successful identification of vehicles was evident (converging to 100% as batch size increases) when the vehicles' front data was passed through the neural networks trained to the vehicle front. The same was true, convergence to 100%, for the vehicles' rear data when passed through the neural networks trained to the vehicle rear. Unfortunately this classification rate does not hold true, with the majority voting rule, if vibrometry data from a vehicle's front was passed through the neural networks trained to the rear data. The reverse, data from a vehicles rear passed through the neural networks trained to the front, also resulted in a deceased correct classification rate.

In the field, an operator would theoretically not know if the data was from the vehicles front or rear and anything below a 50% classification rate would not help distinguish between vehicles. Neither set of nine neural networks, trained to the front and the rear, appeared to perform better or at an acceptable level when given data from the opposite end of the vehicle from which it was trained.

An idea to alleviate this problem was to classify the data as front or rear data before sending it through the neural networks for classification. This concept did not have acceptable results and other avenues were explored.

The next proposal was to use all eighteen neural networks, nine from the front and nine from the rear, in one classification algorithm. In order to make this work, a data set was sent through the six neural networks trained to the same axis. The x-axis data was processed by all the neural networks trained to the x-axis. This included the two networks trained to vehicle A,

one from the front and one from the rear, as well as the two networks trained to vehicle B and the two networks trained to vehicle C. Of these six networks the one with the best performance, no matter if it was a front or a rear trained network, would supply a vote to the classification of the vehicle. The majority voting rule remained in place for this eighteen network method. The best performing network from the x-axis coupled with the best performing networks from the y and z-axis would create the classification. The results from this eighteen network method proved superior to the previous results. Figure 23 shows the addition of neural networks from the original algorithm to the proposed 18 network algorithm. With this new algorithm, data set AF_13_9, from the x-axis, would be pushed through six networks, 3 trained to the front of the vehicles and 3 trained to the rear of the vehicles, for an MSE calculation. The network with the smallest MSE would provide the vote from the x-axis towards the system classification. In this example, the system classification changed from "C" when using nine networks to "A" when using all eighteen networks.

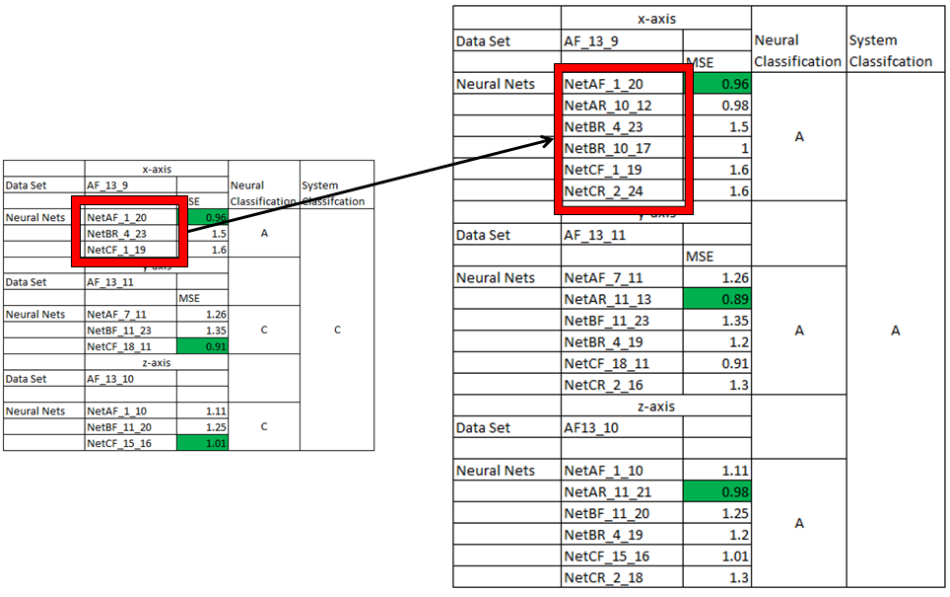


Figure 23 Eighteen Network Classification Algorithm

When all data exemplars from A run #13 were classified by the eighteen network algorithm, and batch sizes of 100, overall system classification increased to 89.49% compared to 83.13% (front networks) and 72.93% (rear networks). Table 10 lists the confusion matrix when AF run #13, all data exemplars, were classified the algorithm using both networks.

Table 10 AF Run #13 Classification Matrix – Both Networks

Time: 20.00 seconds - 29.99 seconds, .01 sec increments					
Verification	Front Data	All Networks		Batch Size: 100	
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AF_13	42956	2514	485	2045	48000
	89.49%	5.24%	1.01%	4.26%	100%

Model Analysis

Once satisfied with the classification algorithm (using all eighteen networks), the validation data was examined for classification to determine the performance strength of the proposed classification technique using Nonlinear Autoregressive Neural Networks. Figure 24 lists the data runs from each vehicle used for classification validation.

AF		
x-axis	y-axis	z-axis
13_19	13_11	13_10
13_20	13_13	13_14
13_9	13_22	13_21
	13_23	13_24
14_19	14_11	14_10
14_20	14_13	14_14
14_9	14_22	14_21
	14_23	14_24

AR		
x-axis	y-axis	z-axis
1_12	1_13	1_14
1_17	1_16	1_18
	1_20	1_21
	1_23	1_22
		1_24
4_11	4_10	4_14
4_12	4_13	4_18
	4_19	4_22
	4_20	4_9
	4_23	

BR		
x-axis	y-axis	z-axis
13_17	13_13	13_14
13_23	13_19	13_21
14_17	14_19	14_14
14_23		14_20
		14_21
		14_22
		14_24

CF		
x-axis	y-axis	z-axis
4_13	4_11	4_10
4_15	4_12	4_14
4_19	4_17	4_16
	4_23	4_18
		4_20
		4_21
		4_22
		4_24

Figure 24 Validation Data Sets

Table 11 shows the confusion matrix when the neural networks classify data batch sizes of 100 from the vehicles' front validation data. The BF did not have any complete data sets for a

run across all three axis; therefore, any classification as the BF would be a false positive. Still the results are consistent across both the A and the C vehicles.

Table 11 Vehicle Front Validation Data Classification Matrix

Time: 20.00 seconds - 29.99 seconds, .01 sec increments					
Validation	Batch Size: 100				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AF	89.49%	5.24%	1.01%	4.26%	48000
BF	N/A	N/A	N/A	N/A	0
CF	0.64%	1.84%	95.41%	2.10%	96000

Table 12 shows the confusion matrix when the neural networks classify data batch sizes of 100 from the vehicle front validation data. The CF did not have any validation data sets for a therefore any classification as the CF would be a false positive. The BR vehicle was classified at 61.41% which was much lower than any other vehicle classification using validation data.

Table 12 Vehicle Rear Validation Data Classification Matrix

Time: 20.00 seconds - 29.00 seconds, .01 sec increments					
Validation	Batch Size: 100				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AR	84.84%	7.50%	1.30%	6.36%	72080
BR	16.25%	61.41%	5.45%	16.89%	16218
CR	N/A	N/A	N/A	N/A	0

At 10 KHz a batch size of 100 data points is one hundredth of one second. This small amount of time still provides a high true positive rate for the front and rear of the vehicles. In fact, as one increases the batch size provided to the neural networks for classification, the correct classification rate increases across the board. Table 13- Table 15 illustrate this increasing performance when larger batch sizes are provided. Table 13 lists batch sizes of 500 while Table 14 lists batch sizes of 1,000 for the rear validation exemplars. Table 15 lists batch sizes of 500 while Table 16 lists batch sizes of 1,000 for the front validation exemplars.

Table 13 Rear Data Batch Size 500 Matrix

Time: 20.00 seconds - 29.90 seconds, .05 sec increments					
Validation	Batch Size: 500				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AR	98.45%	0.35%	0.00%	1.21%	15920
BR	3.85%	95.70%	0.00%	0.45%	3582
CR	N/A	N/A	N/A	N/A	

Table 14 Rear Data Batch Size 1,000 Matrix

Time: 20.00 seconds - 29.90 seconds, .1 sec increments					
Validation	Batch Size: 1,000				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AR	98.90%	0.00%	0.00%	1.10%	8000
BR	1.22%	98.78%	0.00%	0.00%	1800
CR	N/A	N/A	N/A	N/A	0

Table 15 Front Data Batch Size 500 Matrix

Time: 20.00 seconds - 29.90 seconds, .05 sec increments					
Validation	Batch Size: 500				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AF	99.72%	0.22%	0.00%	0.06%	9600
BF	N/A	N/A	N/A	N/A	0
CF	0.00%	1.71%	98.10%	0.19%	19200

Table 16 Front Data Batch Size 1,000 Matrix

Time: 20.00 seconds - 29.90 seconds, .1 sec increments					
Validation	Batch Size: 1,000				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
AF	100.00%	0.00%	0.00%	0.00%	4800
BF	N/A	N/A	N/A	N/A	0
CF	0.00%	0.11%	99.85%	0.03%	9600

When the batch size is increased to 0.05 seconds (500 points) and 0.1 seconds (1000 points) the performance of the classification algorithm converges to 100% from the testing data set.

Results

Table 17– Table 19 show the increasing performance when larger batch sizes are provided to the classification algorithm for all the validation exemplars.

Table 17 Validation Data Classification Matrix Batch Size 100

Time: 20.00 seconds - 29.00 seconds, .01 sec increments					
Validation	Batch Size: 100				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
A	86.70%	6.60%	1.18%	5.52%	120080
B	16.25%	61.41%	5.45%	16.89%	16218
C	0.64%	1.84%	95.41%	2.10%	96000

Table 18 Validation Data Classification Matrix Batch Size 500

Time: 20.00 seconds - 29.90 seconds, .05 sec increments					
Validation	Batch Size: 500				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
A	98.93%	0.30%	0.00%	0.78%	25520
B	3.85%	95.70%	0.00%	0.45%	3582
C	0.00%	1.71%	98.10%	0.19%	19200

Table 19 Validation Data Classification Matrix Batch Size 1,000

Time: 20.00 seconds - 29.90 seconds, .1 sec increments					
Validation	Batch Size: 1,000				
	"A"	**"B"**	**"C"**	**Non-Dec**	**Total Samples**
A	99.31%	0.00%	0.00%	0.69%	12800
B	1.22%	98.78%	0.00%	0.00%	1800
C	0.00%	0.11%	99.85%	0.03%	9600

Summary

The 18 trained neural networks should all be used to compete for the overall classification. When these networks compete, using a majority voting rule, then the system classification averages over 98%, when batch sizes of 1,000 are used. Smaller batch sizes, 100

and 500, which correspond to one hundredth and five hundredths of one second respectively can be used as a quick look classification and still provide over 90% correct classification in most cases.

V. Conclusion

Results

When compared to similar testing on the same data sets, the time series neural networks developed in conjunction with this thesis slightly outperform similar classification attempts. This indicates that utilizing the information contained in the time series of vibrations provides information that can be used to classify between vehicles. Results appear to converge to a 100% correct classification for all vehicles types as larger data batches are provided to the classification algorithm. When lengths of 1,000 (0.10 seconds) are used the overall correct classification rate achieved is 99.39%.

Research Conclusion

This research provided to the body of knowledge already developed using vibrometry data for ATR. This new technique could also be used as a building block for future research. One limiting factor during this research was the limitation of data. Only vehicles at idle were used to train and validate the algorithm. Of interest would be how various engine rotations per minute affect training and testing classification rates. Could the algorithm developed here be used to correctly classify vehicles not operating at idle, would be the most obvious question to answer going forward.

Future Research

Three areas appear to be available for future research. As referenced above, the ability of this algorithm to correctly classify the three objects not operating at idle only would be one area to examine. Another potential for future research could be the addition of more vehicles to the classification algorithm. Although a re-look at the optimization of neural network combinations

might have to be re-examined and time consuming, the ability to add more vehicles to the algorithm could show the robustness of the technique amongst its peers. Finally, this technique and developed algorithm could be used in conjunction with Principle Component Analysis and frequency domain transformations as a fusion of classifiers to achieve a more robust algorithm when combined.

WORKS CITED

Atiya, A. F., El-Shoura, S., Shaheen, S. I., & El-Sherif, M. (1999). A Comparison Between Neural-Network Forecasting Techniques-Case Study: River Flow Forecasting. *IEEE Transactions on Neural Networks*, 402-409.

Azcarraga, A. P., Hsieh, M.-H., & Setiono, R. (2008). Market research applications of artificial neural networks. *IEEE Congress on Evolutionary Computation (CEC)*, 357-363.

Bauer, K. W., Alsing, S. G., & Greene, K. A. (2000). Feature Screening Using Signal-to-Noise Ratios. *Neurocomputing*, 29-44.

Beale, M., Hagan, M., & Demuth, H. (2013). *Neural Network Toolbox User Guide.* Natick: Mathworks.

Beck, N., King, G., & Zeng, L. (2000, March). Improving Quantitative Studies of International Conflict: A Conjecture. *American Political Science Review, 94*(1), 21-35.

Bihl, T. J., & Bauer, K. W. (to be submitted: 2014). Statistical analysis of state of the union addresses. *Presidential Studies Quarterly*.

Bowerman, B. L., O'Connell, R. T., & Koehler, A. B. (2005). *Forecasting, Time Series, and Regression: An Applied Approach* (4th ed.). Belmont, CA: Thomson Brooks/Cole.

Burke, H. B., Goodman, P. H., Rosen, D. B., Henson, D. E., Weinstein, J. N., Harrell, F. E., . . . Bostwick, D. G. (1997). Artificial neural networks improve the accuracy of cancer survival prediction. *Cancer, 79*, 857-862.

Busse, S., Helmholz, P., & Weinmann, M. (2012). Forecasting Day Ahead Spot Price Movements of Natural Gas. *Multikonferenz Wirtschaftsinformatik*, 10.

Castellini, P., & Revei, G. (2000). An Experimental Technique for Structural Diagnostic Based on Laser Vibrometry and Neural Networks. *Shock and Vibration*, 381-397.

Chakraborty, K., Mehrotra, K., Mohan, C. K., & Ranka, S. (1992). Forecasting the Behavior of Multivariate Time Series Using Neural Networks. *Neural Networks*, 961-670.

Chow, T., & Leung, C. (1996). Nonlinear Autogregressive Integrated Neural Network Model for Short-Term Load Forecasting. *IEEE Proceedings on Generation, Transmission, and Distribution*, 500-506.

Crider, L., & Kangas, S. (2012). *Exploiting Vibrometry Based Signatures for Automatic Target Recognition.* Wright Patterson AFB: United States Air Force Research Labratory.

Dierking, M., Heitkamp, F., Roth, B., & Armstrong, E. (2012). *Evaluation of Feature and Classifier Methods 2003 - 2007.* WPAFB: Air Force Research Labratory.

Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics, 35*, 352-359.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification.* New York: Wiley.

Friesen, K. D., Bihl, T. J., Bauer, K. W., & Friend, M. A. (2013). Contextual Anomaly Detection Cueing Methods for Hyperspectral Target Recognition. *American Journal of Science and Engineering*, *2*(1), 9-16.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan College Publishing.

Ivry, T., & Michal, S. (2013, February 13). *License Plate Number Recognition Using Artificial Neural Network* . Retrieved from Introduction to Computational and Biological Vision: http://www.cs.bgu.ac.il/~icbv061/StudentProjects/ICBV061/ICBV-2006-1-TorIvry-ShaharMichal/index.php

Jain, A. K., Duin, R. P., & Mao, J. (2000). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4-37.

Jameson, D. F. (2007). *Effects of Spatial Modes on LADAR Vibration Signatures Estimation.* Dayton: University of Dayton.

Kaastra, I., & Boyd, M. (1996). Designing a Neural Network For Forecasting Financial and Economic Time Series. *Neurocomputing 10*, 215-236.

Kucheva, L., Whitaker, C., & Shipp, C. (2003). Limits on the Majority Vote Accuracy in Classifier Fusion. *Pattern Analysis and Applications*, 22-31.

Kuo, R. J., Chen, C. H., & Hwang, Y. C. (2001). An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy Sets and Systems, 118*, 21-45.

Kurt, I., Ture, M., & Kurum, A. T. (2008). Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease. *Expert Systems with Applications*, *34*(1), 366-374.

Laine, T. I., Bauer, K. W., Lanning, J. W., Russell, C. A., & Wilson, G. F. (2002). Selection of input features across subjects for classifying crewmember workload using artificial neural networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 32*(6), 691-704.

Leap, N. J., Clemans, P. P., Bauer, K. W., & Oxley, M. E. (2008). An Investigation of the Effects of Correlation and Autocorrelation on Classifier Fusion and Optimal Classifier Ensembles. *International Journal of General Systems*, 37-41.

Lee, Y., & Chang, T. (2009). Application of NARX Neural Network in Thermal Dynamics Identifcation of a Pulsating Heat Pipe. *Energy Consversion and Management*, 1069-1078.

Lisboa, P. J. (2002). A review of evidence of health benefit from artificial neural networks in medical intervention. *Neural Networks, 15*(1), 11-39.

Liu, P., Chen, S.-H., Yang, H.-H., Hung, C.-T., & Tsai, M.-R. (2008). Application of Artificial Neural Network and SARIMA in Portland Cement Supply Chain to Forecast Demand. *Fourth International Conference on Natural Computation*, 97-101.

Loeffelholz, B., Bednar, E., & Bauer, K. W. (2009). Predicting NBA Games Using Neural Networks. *Journal of Quantitative Analysis in Sports, 5*(1), 1-15.

Manel, S., Dias, J.-M., & Ormerod, S. J. (1999). Comparing discriminant analysis, neural networks and logistic regression for predicting species distributions: a case study with a Himalayan river bird. *Ecological Modelling, 120*, 337-347.

McClelland, J., & Rumelhart, D. (1988). *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercise.* Cambridge: MIT Press.

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.

Microsoft. (2007). *Microsoft Excel [Computer Software]*. Redmond, Washington: Microsoft.

O'Brien, P. (2012, February 13). *ESS 265*. Retrieved from UCLA.edu : http://www-ssc.igpp.ucla.edu/personnel/russell/ESS265/Ch9/linear_predict/

Odom, M., & Sharda, R. (1991). A Neural Network Model for Bankruptcy Prediction. *IEEE International Joint Conference on Neural Networks*, 163-168.

Pacelli, V., Bevilacqua, V., & Azzollini, M. (2011). An Artificial Neural Netowkr Model to Forecast Exchange Rates. *Journal of Intelligent Learning Systems and Applications*, 57-69.

Paliwal, M., & Kumar, U. A. (2009). Neural networks and statistical techniques: A review of applications. *Expert Systems with Applications, 36*, 2-17.

Phillips, E.-J., Phillips, P., & Hurrell, M. (2013). Optimising insurance fraud detection and classification of vehicle accident damage by using neural networks to identify patterns in behavior, linked cases and vehicle recognition. *Third International Conference on Digital Information Processing and Communications (ICDIPC2013)*, 611-620.

Roth, B. D. (2013, December 13). Data Collection Methodologies. (M. R. Ward, Interviewer)

Skidmore, T. A. (1991). *The Electroencephalographic Human-Computer Interface.* Athens, OH: PhD Dissertation, Ohio University.

Smetek, T. (2007). *Hyperspectral Imagery Target Detection Using Improved Anomaly Detection And Signature Matching Methods.* Air Force Institute of Technology: MS Thesis.

Soman, P. C. (2008). *An Adaptive NARX Neural Network Approach for.* New Brunswick: Rutgers Univeristy.

Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., & Lendasse, A. (2007). Methodology for long-term prediction of time series. *Neurocomputing*, 2861-2869.

Speights, D. B., Brodsky, J. B., & Chudova, D. L. (1999). Using neural networks to predict claim duration in the presence of right censoring and covariates . *Casualty Actuarial Society Forum*, 255-278.

Stack, J. R. (2003). *Patent No. 7146846B2.* United States of America.

Stack, J. R., Harley, R. G., Springer, P., & Mahaffey, J. A. (2003). Estimation of Wooden Cross-Arm Integrity using Artificial Neural Networks and Laser Vibrometry. *IEEE Transactions On Power Delivery*, 1539-1544.

Surkan, A., & Singleton, J. (1991). Neural Networks for Bond Rating Improved by Multiple Hidden Layers. *IEEE International Joint Conference on Neural Networks*, 157-162.

Temurtas, H., Yumusak, N., & Temurtas, F. (2009). A comparative study on diabetes disease diagnosis using neural networks. *Expert Systems with Applications, 36*(4), 8610-8615.

Trippi, R., & DeSieno, D. (1992). Trading Equity Index Futures With a Neural Network. *J. Portfolio Management*, 27-33.

Turkoglu, I., Arslan, A., & Ilkay, E. (2003). An Intelligent System for Diagnosis of the Heart Valve Diseases with Wavelet Packet Neural Networks. *Computers in Biology and Medicine*, 319-331.

Turnquist, B. R. (2011). *Fusion Schemes for Ensembles of Hyperspectral Anomaly Detection Algorithms.* Air Force Institute of Technology: MS Thesis.

Ubeyli, E. D. (2008). Measuring saliency of features extracted by model-based methods from internal carotid arterial Doppler signals using signal-to-noise ratios. *Digital Signal Processing, 18*(1), 2-14.

Ubeyli, E. D. (2009). Measuring saliency of features representing EEG signals using signal-to-noise ratios. *Expert Systems with Applications, 36*(1), 501-509.

Verikas, A., & Bacauskiene, M. (2002). Feature selection with neural networks. *Pattern Recognition Letters, 23*, 1323-1335.

Young, W. A., & Weckman, G. R. (2008). Evaluating the effects of aging for profesional football players in combine events using performance-aging curves. *International Journal of Sports Science and Engineering*, *2*(3), 131-143.

Young, W. A., Bihl, T. J., & Weckman, G. R. (July 2013). Artificial Neural Networks for Business: A Starting Point. *Encyclopedia of Buisniess Analytics and Optimization*, 1st Edition, accepted.

Young, W. A., Bihl, T. J., & Weckman, G. R. (July 2013). Artificial Neural Networks for Business: A Starting Point. *Encyclopedia of Buisniess Analytics and Optimization*, 1st Edition, accepted.

Young, W. A., Millie, D. F., Weckman, G. R., Anderson, J. S., Klarer, D. M., & Fahnenstiel, G. L. (2011). Modeling net ecosystem metabolism with an artificial neural network and Bayesian belief network. *Environmental Modelling & Software, 26*, 1199-1210.

Zhang, G. P. (2000). Neural Network for Classification: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 451-462.

Zhang, G. P. (2003). Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model. *Neurocomputing*, 159-175.

Zhang, G. P. (2007). Avoiding Pitfalls In Neural Network Research. *IEEE Transactions on Systems, Man, and Cybernetics*, Part C, 37(1), 3-16.

Appendices

Appendix A

Function	Input	Output	Description	Subfunction
ntscreate.m	Vehicle A and B x-axis Data Sets	Trained Neural Networks	This function will go through every data set provided and call the subfunction "ntsattempt2.m" in order to run through the neural structure optimization routine.	ntsattempt2.m
ntscreate_y.m	Vehicle A and B y-axis Data Sets	Trained Neural Networks	Same as "ntscreate.m"	ntsattempt2.m
ntscreate_zfront.m	Vehicle A, B, and C front z-axis Data Sets	Trained Neural Networks	Same as "ntscreate.m"	ntsattempt2.m
ntscreate_zrear.m	Vehicle A, B, and C rear z-axis Data Sets	Trained Neural Networks	Same as "ntscreate.m"	ntsattempt2.m
ntscreatC.m	Vehicle C x-axis Data Sets	Trained Neural Networks	Same as "ntscreate.m"	ntsattempt2.m
ntsattempt2.m	Vehicle Data Set	Neural Network	This function optimizes the neural structure by training multiple neural structures to the same data set and pushing forward only the best neural network	
comparenets_x	Neural Networks, Data Sets	Best combination	Comparenets cycles through all the combinations of neural networks for each axis and side and determines the best combination	testnet1000XYZ
comparenets_y	Neural Networks, Data Sets	Best combination	same as "comparenets_x.m"	testnet1000XYZ
comparenets_yrear	Neural Networks, Data Sets	Best combination	same as "comparenets_x.m"	testnet1000XYZ
comparenets_zfont	Neural Networks, Data Sets	Best combination	same as "comparenets_x.m"	testnet1000XYZ
comparenets_z rear	Neural Networks, Data Sets	Best combination	same as "comparenets_x.m"	testnet1000XYZ
testnet1000XYZ	1 Data Set, 3 Neural Networks	Accuracy	Based upon the data set neural networks passed to testnet1000XYZ, this function will calculate the performance of each neural network based upon the data and classify the vehicle based upon the best performing neural network	Performance

Function	Input	Output	Description	Subfunction
Testnetsfrontbothnets 1k.m	All best combinations, data sets	Confusion Matrix	Testnets will cycle through all the front data exemplars to calculate the confusion matrix based upon the data sets and the best combinations within each axis	TN1kcon.m
Testnetsrearbothnets1 k.m	All best combinations, data sets	Confusion Matrix	Testnets will cycle through all the rear data exemplars to calculate the confusion matrix based upon the data sets and the best combinations within each axis	TN1kcon.m
Testnetsfrontbothnets 500.m	All best combinations, data sets	Confusion Matrix	Testnets will cycle through all the front data exemplars to calculate the confusion matrix based upon the data sets and the best combinations within each axis	TN500con.m
Testnetsrearbothnets5 00.m	All best combinations, data sets	Confusion Matrix	Testnets will cycle through all the rear data exemplars to calculate the confusion matrix based upon the data sets and the best combinations within each axis	TN500con.m
Testnetsfrontbothnets 100.m	All best combinations, data sets	Confusion Matrix	Testnets will cycle through all the front data exemplars to calculate the confusion matrix based upon the data sets and the best combinations within each axis	TN100con.m
Testnetsrearbothnets1 00.m	All best combinations, data sets	Confusion Matrix	Testnets will cycle through all the rear data exemplars to calculate the confusion matrix based upon the data sets and the best combinations within each axis	TN100con.m

Appendix B

Contact Information

If anyone is interested in the code that was used in MatLab to perform all of the operations and analysis mentioned earlier, please refer to the contact information below.

Dr. Kenneth Bauer
Kenneth.bauer@afit.edu

Marc R. Ward, Capt. (USAF)
Marc.ward@us.af.mil

Vita

Captain Marc R. Ward graduated from Venture High School in San Ramon, California. He entered the United States Air Force Academy (USAFA) in Colorado Springs, Colorado where he graduated with a Bachelor of Science degree in Operations Research and commissioned in 2008.

His first assignment was at Peterson AFB as a Test Analyst for the Air Force Operational Test and Evaluation Center, Detachment 4. In May 2011, he served as the Enhanced Polar System Test Director for AFOTEC Det 4. In September 2012, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned as a faculty member of the Department of Mathematical Sciences at USAFA.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
27-03-2014	Master's Thesis	Oct 2012 – Mar 2014

4. TITLE AND SUBTITLE

Automatic Target Recognition Using Nonlinear Autoregressive Neural Networks

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S

Marc R. Ward Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-14-M-33

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Lt Col David M. Ryer, PhD
2241 Avionics Circle
Wright-Patterson AFB, OH 45433
david.ryer@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RYA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A Approved For Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

9. ABSTRACT

Accurate combat identification is critical to military interactions. Laser radar for vehicle identification is a rapidly developing field that could possibly assist in combat identification by providing information about operating characteristics of a particular vehicle based on measured vibrations. This research focuses on simulated laser radar data collected from mounted vibrometers on idling vehicles. An approach to identify vehicles using nonlinear autoregressive neural networks for classification is developed and employed. The resulting algorithm combines the trained neural networks across three dimensions of vibration readings. This method offers improved performance over literature in successfully identifying a vehicle through vibration measurements alone.

15. SUBJECT TERMS

Automatic Target Recognition; Artificial Neural Networks; Nonlinear Autoregressive Neural Network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**Dr. Kenneth Bauer**
U	**U**	**U**	**UU**	84	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18