Evaluation of Neural Networks for Data Classification, Recognition, and Navigation in the Marine Environment

by
Brian Patrick Howell

Bachelor of Arts
Physics
Bridgewater College
1978

Master of Engineering
Electrical Engineering
University of Virginia
1981

A dissertation submitted to the Department of Marine and Environmental Systems at Florida Institute of Technology in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Ocean Engineering

Melbourne, Florida
May, 2004
The author grants permission to make single copies____________________________
Abstract

Evaluation of Neural Networks for Data Classification, Recognition, and Navigation in the Marine Environment

Author:
Brian Patrick Howell

Advisor:
Stephen Wood, Ph.D., P.E.

The purpose of this research is to explore a family of strategies using neural networks for automating certain tasks of underwater exploration. Specifically, this research explores the use of neural networks to create basic blocks for detection and classification of a variety of sensor inputs as well as for vehicle control in marine settings. This body of work can then be used to implement fully autonomous detection and navigation systems for use in the marine environment. Several candidate neural network paradigms were evaluated for use in the research and are discussed. This research has been broken into two main portions based on the functional task desired and the nature of the data to be analyzed.

The first project deals with passive acoustic data from hydrophones. In this effort, different preprocessing and network strategies are evaluated for utility in discerning different acoustic sources. Both unsupervised (Kohonen Map) and supervised hybrid paradigms were tested (Kohonen/Multi-Level Perceptron). Sources include surface and underwater vehicles, geophysical sounds, underwater mammals of several types, and several fish species. Recognition rates of 100% are achieved for man made sources and most cetacean sources. Fish are problematic for a combined network, but improved results are achieved using a “fish only” network with pulse energy gathering. With wavelet preprocessing, recognition rates of 31% to 72% are reported for fish only data with 7 species of fish.

The second project examines the case of multiple sensor inputs, including temperature, turbidity, salinity, and pressure. The concept of primitive and emergent behaviors is developed, and the structures are tested on both theoretical and “real world” data sets. A standardized set of common “primitive features” is defined for all sensor types and complex environmental features are recognized as combinations of the primitive features using a multi-level perceptron network. For primitive features in 5% noise, feature extraction of 75.5% is demonstrated. For emergent features, recognitions of 83% are achieved, with some features such as tidal inlets and hydrothermal vents being recognized with 100% correct recognition. Recognition rates as functions of input data format, noise levels, and output category structure are also presented.
# Table of Contents

List of Keywords ........................................................................................................... iiix
List of Exhibits .................................................................................................................. x
List of Abbreviations ....................................................................................................... xiii
List of Symbols ................................................................................................................ xv
Acknowledgements ......................................................................................................... xvi
Dedication ........................................................................................................................... xvii

1. Introduction .................................................................................................................. 1
   1.1. Needs in Ocean Exploration .................................................................................. 1
   1.2. Autonomous Measurements ............................................................................... 3
       1.2.1. Types of Instrumentation .......................................................................... 3
       1.2.2. Environmentally Based Navigation ......................................................... 4
   1.3. Processing Methodologies ................................................................................... 8

2. Structure of Paper ....................................................................................................... 11
   2.1. Overview ............................................................................................................... 11
   2.2. Passive Acoustics (Chapter 5) ......................................................................... 15
   2.3. Multi-Sensor Feature Detection (Chapter 6) .................................................. 16

3. Neural Network Overview ........................................................................................ 18
   3.1. Introduction ......................................................................................................... 18
   3.2. Basic Concepts .................................................................................................... 20
       3.2.1. Neurons .................................................................................................. 20
4.6.2. Types of Wavelets ............................................................. 64
4.6.3. Use/Comparison with FFT ............................................. 65
4.7. Data Preprocessing Conclusions ....................................... 66

5. Project 1: Passive Acoustics ............................................... 67

5.1. Introduction .................................................................. 67
5.2. Sound Data Sets ............................................................. 69
5.3. Sound Characteristics ...................................................... 71
5.4. Paradigm Selection .......................................................... 78
5.5. Preprocessing strategies .................................................... 80
  5.5.1. Spectrogram ................................................................. 80
  5.5.2. Scaling .................................................................. 82
5.6. Dataset 1 ....................................................................... 82
5.7. Dataset 2 ....................................................................... 84
5.8. Self Organizing Map Results .......................................... 85
  5.8.1. Data set 1 Results: Sample Frame Experiment ............... 85
  5.8.2. Data set 1 Results: FFT Variation Experiment ............... 90
5.9. Multi-level Perceptron Map ............................................. 94
5.10. Hybridnet Design .......................................................... 96
5.11. Hybridnet Results .......................................................... 97
5.12. Fish Sounds ................................................................. 100
  5.12.1. Pulse Gathering ......................................................... 102
  5.12.2. FFT Results ............................................................... 104
  5.12.3. Wavelet Results ....................................................... 106
5.13. Two category Network Results .............................................................. 108
5.14. Discussion .......................................................................................... 109

6. Project 2: Geophysical Data Fusion .......................................................... 111

6.1. Environmental Features ......................................................................... 111
   6.1.1. Temperature .................................................................................... 115
   6.1.2. Salinity ........................................................................................... 116
   6.1.3. Pressure .......................................................................................... 116
   6.1.4. Water Clarity .................................................................................. 116
   6.1.5. Fluorometry .................................................................................... 117
   6.1.6. Optical Absorption Spectroscopy .................................................... 117
   6.1.7. Mass Spectroscopy ......................................................................... 118
   6.1.8. Inertial Measurements ................................................................... 118
   6.1.9. Active Acoustic .............................................................................. 119
   6.1.10. Passive Acoustic .......................................................................... 119

6.2. Primitive Behaviors .............................................................................. 120

6.3. Emergent Behaviors ............................................................................ 122

6.4. Paradigm Selection ................................................................................ 126

6.5. Data Set One: Primitive Behaviors ....................................................... 127
   6.5.1. Description ..................................................................................... 127
   6.5.2. Functional Test ............................................................................... 131
   6.5.3. Mean and Span Test ..................................................................... 136
   6.5.4. Fitting /Difference Preprocessing Test .......................................... 139
   6.5.5. Data Set 1 Conclusion .................................................................. 140
6.6. Vailulu'u ................................................................................................. 141
6.7. Data Set 2 .............................................................................................. 143
   6.7.1. Global vs Local Scaling Experiment .............................................. 145
   6.7.2. History/Temporal Feedback ............................................................. 148
6.8. Feature Synthesis Mapping ................................................................. 150
   6.8.1. Multiple Feature Maps ................................................................. 152
   6.8.2. Multiple Feature Maps: Results .................................................... 157
   6.8.3. Two Category Testing ................................................................. 163
6.9. Discussion ............................................................................................ 164
7. Conclusions ............................................................................................ 167
8. Recommended Future Work .................................................................... 169
   8.1. Data Gathering ................................................................................... 169
   8.2. Distributed vs Ensemble Networks ................................................... 170
   8.3. Pre-processing .................................................................................... 170
   8.4. Alternate Paradigms .......................................................................... 171
   8.5. Hardware Implementations ............................................................... 172
   8.6. Final Remarks .................................................................................... 173
References .................................................................................................. 174
Appendix A: Glossary ............................................................................... 189
Appendix B: Software Listings ............................................................... 194
List of Keywords

Acoustic navigation
Artificial Intelligence
Backpropagation
Cetacean
Department of Marine and Environmental Systems (DMES)
Environmental
Fast Fourier Transform (FFT)
Florida Institute of Technology (FIT)
Florida Tech
Hybrid Neural Network
Kohonen
Matlab®
Navigation
Neural Network
Perceptron
Seamount
Self Organizing Map (SOM)
Underwater Measurements
Underwater Technology Laboratory (UTL)
Vailulu’u
Wavelet
List of Exhibits

Exhibit 1.1. Table of High Level Mission Commands 6
Exhibit 2.1. Conceptual AUV Model 13
Exhibit 3.1. Individual Neuron Structure 21
Exhibit 3.2. Typical Training Error Patterns 26
Exhibit 3.3. Network Overfit Mean and Variance Training Errors 28
Exhibit 3.4. Sample Confusion Matrix 30
Exhibit 3.5. Kohonen Self Organizing Map 34
Exhibit 3.6. Neighborhood Definition 35
Exhibit 3.7. SOM Category Ambiguity 37
Exhibit 3.8. Multi-Level Perceptron 41
Exhibit 3.9. Hybrid Network Block Diagram 48
Exhibit 4.1. Envelope Filtering of Pulses 57
Exhibit 4.2. Use of Polynomial Regression to Smooth Data (cubic) 58
Exhibit 4.3. Coiflet order 3 Scale Expansion 61
Exhibit 4.4. Filtering effects of wavelets 62
Exhibit 4.5. Coiflet 3 and Discrete Meyer Wavelets 64
Exhibit 5.1. External Sound Sources Used 72
Exhibit 5.2. Ship Frequency Spectrum 73
Exhibit 5.3. Shrimp Frequency Spectrum 74
Exhibit 5.4. Fish Time Domain Signals 75
Exhibit 5.5. Humpback Whale Time and Spectra

Exhibit 5.6. Input Definitions for Module SPGRAM

Exhibit 5.7. Ensemble Sound Data Set 1

Exhibit 5.8. Fish Source Files for Data Set 2

Exhibit 5.9. Time Frame Experiment Results

Exhibit 5.10. FFT Resolution Experiment Results-SOM

Exhibit 5.11. MLP Re-mapping of SOM Output

Exhibit 5.12. FFT Resolution Experiment Results-Hybridnet

Exhibit 5.13. Fish Mapping, 512 pt FFT Raw Data, Hybridnet 4

Exhibit 5.14. Comparative Time, Frequency and Wavelet for Cow Nose Ray

Exhibit 5.15. Data Set 2 Pulse Responses, FFT Processing

Exhibit 5.16. Coiflet3, Level 3, Decomposition Data Set 2 Samples, Pulse 1

Exhibit 5.17. Data Set 2 Pulse Responses, Wavelet Coiflet3, Level 3 Processing

Exhibit 6.1. Ensemble Neural Classifier Block Diagram

Exhibit 6.2. Distributed Neural Classifier

Exhibit 6.3. Primitive Recognition Categories

Exhibit 6.4. Primitive Response to Street Corner Event

Exhibit 6.5. Description of Data Set 1 Features

Exhibit 6.6. Type 4 Region Profiles

Exhibit 6.7. Type 5 Boundary, Leaving Region of Interest

Exhibit 6.8. Evaluated Multi-Level Perceptron Network Parameters

Exhibit 6.9. Functional Noise Performance 200000 iteration
Exhibit 6.10. Confusion Matrix for Dataset 1, .1 added noise 133
Exhibit 6.11. Noisy Signal vs Ideal, Type 1 134
Exhibit 6.12. Ideal Signal vs Noisy Signal, Type 4 135
Exhibit 6.13. Difference in Input Signal Span and Offsets 137
Exhibit 6.14. Summary Results of Sigmoid/Mean/Span Tests 138
Exhibit 6.15. Alternative Input Format Test Results 140
Exhibit 6.16. Vailulu’u Bathymetry (Staudigel) 142
Exhibit 6.17. Typical Tow-Yo data set 144
Exhibit 6.18. Variable Ranges, Dataset 2 145
Exhibit 6.19. Confusion Matrix for Input Scaling Experiment 147
Exhibit 6.20. Ambiguous Data Samples 148
Exhibit 6.21. Feedback of Feature Category to Neural Network 149
Exhibit 6.22. Confusion Matrix for Data Set 2 History Experiment 150
Exhibit 6.23. Feature and Event Map 154-156
Exhibit 6.24. Emergent Processor Parameters 157
Exhibit 6.25. Event Map Results: 39 Categories 157-159
Exhibit 6.26. Event Map Results: No Day/Night 161-162
Exhibit 6.27. Thermocline Only Recognition Results 164
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ART</td>
<td>Adaptive Resonance Theory</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ASA</td>
<td>Acoustical Society of America</td>
</tr>
<tr>
<td>AU</td>
<td>Audio File (SUN/NEXT/DEC combined format)</td>
</tr>
<tr>
<td>AUV</td>
<td>Autonomous Underwater Vehicle</td>
</tr>
<tr>
<td>BPN</td>
<td>Back Propagation Network</td>
</tr>
<tr>
<td>BPS</td>
<td>Bits per second</td>
</tr>
<tr>
<td>DAT</td>
<td>Digital Audio Tape</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HUGO</td>
<td>Hawaii Undersea Geo-Observatory</td>
</tr>
<tr>
<td>INS</td>
<td>Inertial Navigation System</td>
</tr>
<tr>
<td>IRL</td>
<td>Indian River Lagoon</td>
</tr>
<tr>
<td>KHZ</td>
<td>Kilohertz</td>
</tr>
<tr>
<td>LEO</td>
<td>Long-Term Ecosystem Observatory</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Squares</td>
</tr>
<tr>
<td>LVQ</td>
<td>Learning Vector Quantization</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Level Perceptron</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>MP3</td>
<td>MPEG Audio Stream, Layer 3</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>NOAA</td>
<td>National Oceanic and Atmospheric Administration</td>
</tr>
<tr>
<td>PMEL</td>
<td>Pacific Marine Environmental Laboratory</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>SOEST</td>
<td>School of Ocean</td>
</tr>
<tr>
<td>SOM</td>
<td>Self Organizing Map</td>
</tr>
<tr>
<td>URI</td>
<td>University of Rhode Island</td>
</tr>
<tr>
<td>UUV</td>
<td>Unmanned Underwater Vehicle</td>
</tr>
<tr>
<td>WAV</td>
<td>Microsoft Pulse code modulated .WAV sound file</td>
</tr>
</tbody>
</table>
List of Symbols

k   Time index
k+1 Future time index
K   Learning rate multiplier
µ   Learning rate
µ₀  Initial learning rate
x(k) Input vector
Y(k) Output vector
w⁽ⁱ⁾ Weight matrix for “ith” layer
v⁽ⁱ⁾ Linear output vector from “ith” layer
xout⁽ⁱ⁾ Non-linear neuron output for “ith” layer
d(k) Desired output vector
E(k) Error vector
f(--) Non-linear function of neuron
g(--) Derivative of non-linear function of neuron
α   Sharpness coefficient for sigmoid function
I would like to thank the following people whose efforts have made this project possible...

My advisor Dr. Stephen Wood who has continuously encouraged me to explore areas which might be helpful as well as to allow me access to the Underwater Technology Lab’s AUV’s so the work can be tailored to the vehicles and possible future missions.

Dr. Andrew Zborowski who shared my enthusiasm for marine vehicles and who has guided me in selecting appropriate resources to tackle the vehicle control work.

Dr. Eric Thosteson who gave me guidance and training in both the modeling of the marine environment and the selection of tools to use in this research. In addition, he also provided an infectious enthusiasm for tackling the more theoretical aspects of the project.

Dr. Semen Koksal whose suggestions and guidance helped define the neural network paradigms to use and the methods for evaluation.

Dr. George Maul, whose oceanography class and data resources helped me to take raw electrical engineering knowledge and tailor it to the specifics of the ocean.

Mr. Bill Battin, whose “can do” attitude gave me encouragement to get the job done.

The Link Foundation for providing me with a fellowship which made this possible and more importantly, who believed in my vision for ocean research.

My wife Cheri, whose tireless encouragement helped bolster me through hard times.

Marine Resources Development Foundation and Chris Olstad, for letting me set up equipment in the underwater habitat and to give me initial training in underwater operations.

Dolphins Plus and Art Cooper, who let me make several recordings of their cetacean residents.
Dedication

To my wife, Cheri, who believed in my dreams even when I could not.

To my sons who gave me the desire to make myself something special for them and to give them access to this very special place called the ocean.

To my mom and dad, for believing in me and seeding my dreams and getting me in and on the water.

To my brother Mike, who in his death ordered me to chase my dreams and forever tied me to that mistress, the sea.

To my Lord, who gave me the tools, skills, and circumstances to undertake this task.
Introduction

Needs in Ocean Exploration

At the present time, the ocean is a resource with vast regions remaining to be explored and utilized. The cost of manned exploration remains high, so most exploration missions and instruments are focused on specific military, scientific, or industrial applications such as mine hunting (McCarthy, 2000), telecommunications cable laying (Asakawa, 1996), or submarine detection (Gorman, 1988). There is a need to reduce costs by developing autonomous systems to perform long term data gathering and processing with minimal human intervention (Porto, 2000).

Interest in the ocean environment has grown into new areas in recent years as a result of observed damage to the ocean environment and increased need for resources. Sea level rise and potential global warming, coral reef changes, fisheries declines, and changes to cetacean populations have given rise to research for improved methods for detection and characterization of various parameters in the ocean and estuarine environments. Recently, defense requirements have also expanded from bluewater applications to include coastal and estuarine environments for coastal seaport security (Harbor Fence, 2003), and point source pollution detection (Orlov, 1993). In each of these cases, the mission requires not just passive data taking, but some form of on board intelligence to determine if an “event” has occurred that requires human intervention or awareness, such as counting, observation, or active intervention.

The purpose of this research is to explore the use of neural networks in the tasks of pattern recognition and classification of information in various sensor data typical of the marine environment. In this work, it is desired to develop a common approach to many of these primitive detection problems. By developing a structure for application of the neural networks to these problems it is hoped that this family of methods can be seen as a set of tools that can be used in conjunction with or in replacement for all the other classical methods mentioned. In some cases, a classical method, while more limited, provides a simpler approach, which may be more attractive in computation, physical
complexity, or cost. Nonetheless, the neural networks prove themselves to be a powerful tool in the suite of tools available for underwater exploration.

Autonomous Measurements

Types of Instrumentation

The increased demand for exploration and data and the difficulty of operating in the marine environment have produced the impetus for autonomous methods of exploration. Examples of both fixed point and vehicle-based instrumentation platforms have been demonstrated in the last 10 years (Duennebier, Purcell, 2000). In fixed point monitoring, underwater platforms such as LEO (Alt, 1992) have met with success as well as newer buoy measurements systems (Seabuoy, 2003). Such instruments allow cost effective, remote sensing of oceanic parameters for extended periods. In these systems, temperature, salinity, turbidity, water chemistry, as well as acoustic data have all been successfully collected.

Vehicles are required where the mission is to search out a feature of interest, whether that is of scientific, military, environmental, or commercial importance or to scan larger areas (Pereira, 1996). Several variations of vehicle have evolved such as the Aqua Explorer for cable inspection (Asakawa, 1996), the Hugin series for deepwater work (Hagen, 2003), and Remus for coastal use (Purcell, 2000). Other vehicles are in use such as the Ocean Voyager (An, 1996), Urashima (Tamura, 2000). The future direction in military AUV or in Navy lingo unmanned underwater vehicles (UUV) indicates that more sensor payloads will be applied to autonomous vehicles, expanding the data available to include chemistry and new types of SONAR payloads (Fletcher, 2000).

In all these platforms, the sensory payloads are used primarily in a data recording mode (Linnett, 2003). Data is taken, recorded, and either stored or transmitted to the human observer. There is very little if any processing of the data onboard the platform. This research, in the second project, examines the utilization of many of the types of data taken by both fixed point and vehicle platforms for feature detection and navigation. It will then be shown that neural networks can then generate feature categories that can be used for navigation, or data reduction purposes.
Environmentally Based Navigation

For operating in the marine environment, navigation is currently based on either dead reckoning, or external acoustic referencing systems. The most recent experimental systems have been simple combinations of acoustic, inertial, and GPS navigation methods (Butler, 2001, Bennanoun, 1996). In this context, missions are planned based on specific course headings and duration criteria. If one examines many of the missions envisioned for an autonomous vehicles or moorings, the goals involve detection of or navigation to or from some environmental feature of interest (Feder, 1998). Examples in the geophysical realm include metal objects, temperature and salinity structures in the water column, environmental pollutants or metabolites in the water (Orlov, 1993), patterns of change in turbidity and bioflora (McCarthy, 2000), and odors (Grasso, 1998). Based on this observation, a different paradigm for navigation and detection emerges. Features from the environmental data already being logged by the vehicle or mooring can be extracted to provide cues for navigation or other function (Zalzala, 1996). In addition, by combining information from multiple data sources in the vehicle or mooring, the likelihood of correctly identifying a feature can be increased over that of only a single sensor (Rajapakse, 1990, Eggers, 1990).

The concept of environmentally based navigation can find its roots as far back as the early Polynesian concept of Wayfinding (Kawaharada, 2003). Such organizations as the Polynesian Voyaging Society have resurrected this methodology for long distance navigation without the use of compass or GPS. In this approach, a sailor makes observations based on recognized patterns in wave action, wind action, star position, and semi-permanent structures in the ocean including patterns of fish feeding and upwelling. Based on these cues, voyages have been executed over distances in excess of 1000 nautical miles. By taking that kernel concept and applying it to the sensors available to an automated system, a new way of approaching ocean exploration is proposed.

The goal of this research is to develop the basic building blocks required to create a feature based detection and navigation system. Specifically, through the use of neural
networks to execute feature extraction and classification from input sensor data, this research develops the methods to recognize high level features in the marine environment. In this scenario, a mission planner could execute commands which are directly related to the high level mission objectives (See Exhibit 1.1.).

<table>
<thead>
<tr>
<th>High Level Mission Command</th>
<th>Low Level Mission Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect illegal fishing</td>
<td>Scan until boat + fish pattern recognized</td>
</tr>
<tr>
<td>Detect pollutants</td>
<td>Scan until chemistry pattern indicates region</td>
</tr>
<tr>
<td>Navigate to estuary</td>
<td>Scan until temp+salinity+acoustic+turbidity indicates region</td>
</tr>
<tr>
<td>Detect Submarines</td>
<td>Scan until acoustic+magnetometer+chemistry+efield</td>
</tr>
<tr>
<td>Navigate thermal vents</td>
<td>Scan and follow thermal+turbidity+chemistry features</td>
</tr>
<tr>
<td>Detect Harbor Security Breach</td>
<td>Detect, count and categorize incoming vessels/ subs/ divers</td>
</tr>
<tr>
<td>Follow Bottom of Thermocline</td>
<td>Scan for patter in temperature vs depth</td>
</tr>
<tr>
<td>Perform Mission on Rainstorm/Hurricane Event</td>
<td>Scan for temp+salinity+wave+acoustic for desired patterns</td>
</tr>
<tr>
<td>Detect Algal Bloom Conditions</td>
<td>Scan for Temp+salinity+chemistry+chlorophyll+turbidity</td>
</tr>
</tbody>
</table>

Exhibit 1.1. Table of High Level Mission Commands

Neural networks allow the system to perform recognition and fusion of sensor data in ways not achievable before (Miller, 1992). As an artificial intelligence (AI) method, neural networks have some unique features such as the ability to generalize (Yang, 1991), the ability to work with noisy or incomplete data sets (Menon, 1988), fault tolerance (Masters, 1993), and the ability to change its own structure (Carpenter, 1991).

The ability to change structure extends the basic advantages of neural networks beyond simple navigation. In the case of a marine sensor input, if an unknown environmental feature appears, a neural network is able to flag this as an unknown feature and provide
triggers for recording the feature or whatever behavior the mission planner requires. If needed, the unknown feature can be added to the network as a new feature category automatically. While detecting this unknown feature, it can also generalize to continue its mission function. For example, if the network has been trained to have feature categories for five pollutants only, and then detects an unknown pollutant, it can add a new feature category to the network representing the new pollutant, and datalog the information to allow researchers to identify the unknown at a later date.
Processing Methodologies

In the systems for data sensing and processing on both fixed point and vehicle based platforms, data is acquired and must then be stored, processed, and communicated to the human observer for whom the data is intended. Within the task of data processing, several methodologies have been used to structure the information received from the environment.

For data processing, two primary methods have historically been used to process the incoming data. Classical regression/curve fitting is used to interpret the data from the environment and to derive cause effect relationships in a few dimensions. This is the typical fitting to a function such as a power series that is taught in every undergraduate science and engineering curriculum. Similarly, statistical methods can be used to observe meaningful underlying patterns in the incoming data and classification schemes have been developed around Bayesian statistics for target classification (Deming, 1998). Unfortunately, these methods require having sufficient data to completely define the data space analytically or statistically before or during the measurement. In the case of the ocean environment, much of the data taking is done in areas with no a priori knowledge of the characteristics of the physical environment.

More recently neural methods have been demonstrated for specific detection and processing applications in the marine environment (Baran, 1991, Porto, 2000). Particularly in the area of SONAR, early work has been performed on test data signals (Casselman, 1991, Solinsky, 1991, Cottle, 1991). Similarly, in the case of vehicle control, these same tools have been shown to be useful for some specific applications. For example, the depth control problem (Yuh, 1990), the ship heading control problem
(Burns, 1995), and ship heave compensation (Laniotis, 1993) have been evaluated using neural networks. The attractive characteristic of the neural approach is three-fold. First, the statistics of the data space need not be constant for all time but merely stationary for the interval of evaluation (Ham, 2001). If changes happen in the underlying data, the neural network methods explicitly can adapt to the new structure. Secondly, the data space need not be completely understood before the commencement of operations (Haykin, 1999). The neural networks by nature are required to “learn” the data space, and by careful selection of the network structure, the network should be able to discover even subtle or hidden non-linear relations in the observed data (Masters, 1993). Third, but of less importance to this work, these methods have also been demonstrated to be able to dynamically adjust for damage to a vehicle changing the control model of the vehicle (Wilson, 1995, Troudet, 1992) and to noise in data images (Lu, 1990). These applications demonstrate the robustness of the neural approach.

Given this background, it is appropriate to comprehensively evaluate neural methods as applied to autonomous classification, recognition, and control in the marine environment. The unique feature for this work is the development of a common system structure that is applicable to all data types. This research focuses exclusively on artificial neural networks as opposed to fuzzy logic, traditional artificial intelligence (AI) or other analytical methods. In addition, most of this research will be limited to work on three neural paradigms. Unsupervised Kohonen maps, supervised multi-level perceptrons, and a hybrid of the two are utilized in both projects. Much of the research is spent in evaluating the sensitivity of recognition rates to changes in input presentation methods, denoising methods, and output classification structure. Finally, the difference between
using one large neural network for all categories versus smaller, two category networks is evaluated.

Structure of Paper

Overview

The structure of this research is designed to both explore the suitability of neural networks on specific data fields and develop a common approach for a variety of problems in the marine environment. To this end the project is divided into two main parts. First, neural networks are evaluated for a single sensor (albeit complicated signal) problem, that of classification of passive acoustic signals. This project, dubbed project one (chapter 5), is used to determine the types of networks suitable for the processing, the range of numerical values used for the network structure, and the types of preprocessing which might be used to improve recognition accuracy. Two standard acoustic data sets were created where first, sounds from a broad cross section of sound types are used. Based on the results observed from processing this data set, a second set of sounds involving fish exclusively is used to attempt to understand problems of recognition of fish.

Experiments performed in this project include:

1) Supervised versus unsupervised learning
2) Fast Fourier transform versus Wavelet methods
3) Resolution of FFT versus recognition accuracy

Project two (chapter 6) involves the merging of several sensor signal sources such as temperature, salinity, and spectroscopy into higher levels of recognition. In this project, primitive features common to all sensor types are categorized. Then a neural network is developed to recognize these primitive features. The fusion of these primitive recognitions by
another neural network is used to perform recognition of features such as described in Exhibit 1.1. For assistance to the reader, a brief overview of neural networks used and preprocessing methods for the data are presented in chapters 3 and 4. Finally, conclusions and suggestions for future research are reported.

In order to understand this approach, the reader is referred to a conceptual diagram of a prototype AUV (See Exhibit 2.1.) As part of the thesis of this paper, the tools applied to the AUV problem are completely applicable to fixed-point underwater and surface platforms as described earlier.

Exhibit 2.1. Conceptual AUV Model

It is observed in the diagram that the networks are applicable to almost all portions of the vehicle. The thesis of this research is that a common form of neural network can be used to process data from any of the sensors in the AUV and then use the recognized features for navigation of the vehicle. In essence, the networks will allow vehicles to first identify an object of interest (acoustic signal, magnetometer reading, temperature anomaly, or chemical anomaly) and then navigate to it.

Matlab® is used for all the program development of this project. Matlab® is a software environment optimized for matrix mathematics written by Mathworks, Inc. It has been expanded to include hardware control, wavelet analysis, graphical interfacing, and data acquisition. The software is capable of running as a standalone sensor control system on several platforms, so the neural systems developed here can be rapidly deployed in later work for field testing and can be rapidly modified. Additionally, the software contains an embedded “C” language compiler which allows the user to write more optimized programs as needed.
For both projects, some common analyses will be performed. After a paradigm is chosen for evaluation, training and testing data sets will be created. Additionally, a repeated measure confidence interval method will be used to evaluate the performance of the network. Then one or more variations on the data, the structure, or training will be applied and results compared. The areas to be modulated for performance analysis will include as appropriate:

1) Network structure parameters: These include learning rate, training iterations, number of neurons, initialization, and training order.
2) Data preprocessing: Comparisons between preprocessing methods will be evaluated while keeping the network structure parameters fixed. For a given project, it may include transformations or simple scale and offset ranges.
3) Data set representation: Modifications to an individual data set may be made and compared to determine if an overfit condition exists for the available data. Specifically, some of the datasets were made too “open” containing combinations of inputs not likely in the actual environment. In some cases, reducing the degrees of freedom in the training and testing data itself to reflect real world conditions provides improvements from the artificial “all possible inputs” datasets of project 2.
4) Output presentation: Number of categories, way in which categories or outputs are represented as well as scaling and offset issues.

While all these are important, this document does not purport to span the entire range of values possible for all these categories. After initial experiments to determine a “good” network candidate as defined by the neural network overview of chapter three, the network structure is help constant in each case, or data set held constant in order to optimize the parameters in another part of the data space. For the purposes of this work, the primary figure of merit used was the optimization of recognition or matching rates or minimization of the figure of merit error for a each of the two projects researched. It will be left to future research to perform a comprehensive optimization for a given parameter, i.e. network structure for all possible combinations of input, network structure, and preprocessing parameters.

**Passive Acoustics (Chapter 5)**

The first project is to evaluate the use of the neural networks on single sensor data only. Passive acoustics will be used as the input to be processed. Passive acoustics is the body of ocean operations in which an operator determines the contents of the surroundings by ambient sounds in the water column. These sounds may be of man-made, geophysical, or biological origin. To this end, collections of underwater sounds were gathered and processed through several neural network paradigms. As part of the research, comparative results were
made based on changes to network structure, preprocessing methods, and methods of classification.

**Multi-Sensor Feature Detection (Chapter 6)**

In order to evaluate the neural network behavior with multiple sensor sources, a simpler data suite, that of data from various geophysical sensors is examined. In this section, the concept of primitive versus emergent behaviors is introduced and the feasibility of using neural networks to detect slowly varying changes in data fields is evaluated. Lastly, the ability of the networks to fuse this data to recognize features is examined. Variations due to noise, data presentation, and categorization are also examined. In some cases, the fusion results simply improve the confidence intervals of detection than that of a single sensor. In other cases, it allows recognition of high-level features not possible by single dimensional recognition. While acoustics are not tied to the geophysical data in the research to date, it is shown that the summary results from an acoustic network can be easily included with the geophysical data to create enhanced detection capabilities. Both a ground truth theoretical data set and real world data from the Vailulu'u expeditions of 2000 and 2001 are used to examine these questions. These are referred to as Project 2, dataset 1 and 2.
Neural Network Overview

Introduction

To begin, many engineers and scientists are unfamiliar with the details of a “neural network”. First the term neural network needs to be defined. In the context of this research, the definition of Simon Haykin is appropriate:

“A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1: Knowledge is acquired by the network from its environment through a learning process.

2: Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.” (Haykin, 1999)

Every network evaluated in this paper shares this common definition. It reflects both the structure of the network as well as its distinctiveness compared with classical methods such as equation modeling, regression, curve fitting, and statistical methods.

First, the method of iterative learning, while not unique to neural networks, certainly is dominant in neural networks. In all networks, input data is presented one sample at a time, and the outputs are compared to some ideal case. An error is computed, and in some manner the internal structure of the network is modified to reflect the error calculations. Then another input matrix or vector is applied and the process repeated.

The variations of how the error is reduced and which portion of the network are modified as well as the core topology or paradigm of the network is the focus of ongoing research in the neural networks field. The detailed study of neural network advancement is beyond the scope of this work, yet having a basic understanding of how these processes operate is critical for the ocean engineer in selecting and modifying a network for oceanic use. Therefore, a brief overview of the neural networks used or recommended in this body of research is presented, along with basic terms and definitions. For a more thorough
understanding of the subject, the reader is referred to any number of excellent textbooks on

**Basic Concepts**

**Neurons**

To begin to adequately communicate the results of this research, some basic
knowledge of the terminology and underlying structures of neural networks is required.

All neural networks stem from attempts at modeling biological systems. As such,
every system begins with a structure similar to a biological neuron (See Exhibit 3.1).

As can be seen in the exhibit, the neuron consists of a body that sums the
weighted signals from a variety of inputs. These inputs may be multiple samples in time,
space, or from multiple data types. In addition, the input data may be bipolar (-1, +1),
binary (0, 1), or continuous values (-1 < x < 1). The choice of input and output type is
one of the many variables that may be investigated when optimizing network
performance.
The weights of the inputs provide the storage location for information in the neural networks. Weights are initialized to small random values before training. As the network learns the mappings desired, weights are adjusted among the inputs to minimize error. After training, the weights are frozen and the network, if functioning properly, can perform the desired task.

**Activation Functions**

After the weighted inputs are summed, the output is fed to a non-linear element of some type. Much research in the neural network community focuses exclusively on the nature of the non-linear element. Four types of transfer functions are commonly utilized.

The simplest form of activation function is the linear function (Ham, 2001). This form is used within Self-Organizing Maps (SOM) and in neural implementations of linear algebra solutions. The transfer function is:

\[
    y = \begin{cases} 
        ax & \text{if } y < 0 \\
        1 & \text{if } y > 0 \\
        \tanh(sx) & \\
        1/(1+e^{-a(y)}) & \\
        \text{Other Choices}
    \end{cases}
\]
\[ Y(k) = x(k). \]

Here, \( Y(k) \) is the output of the non-linear element, and \( x(k) \) is the input, which itself is the summation of all the weighted inputs.

The opposite extreme to the linear or identity activation function is the “hard limiter”. This function is represented by:

\[
Y(k) = \begin{cases} 
0 & \text{if } x(k) < 0 \\
1 & \text{if } x(k) > 0. 
\end{cases}
\]

The third class of function is the saturating linear function. Outside of a certain range of input, the function is identically 0 or 1 for the binary case or \(-1\) and 1 for the bipolar case. Within a certain range of \( x(k) \) however, the output is linearly varying.

**Binary:**
\[
Y(k) = \begin{cases} 
0 & \text{if } x(k) < -0.5 \\
x(k) + 0.5 & \text{if } -0.5 < x(k) < 0.5 \\
1.0 & \text{if } x(k) > 0.5 
\end{cases}
\]

**Bipolar:**
\[
Y(k) = \begin{cases} 
-1.0 & \text{if } x(k) < -1.0 \\
x(k) & \text{if } -1.0 < x(k) < 1.0 \\
1.0 & \text{if } x(k) > 1.0 
\end{cases}
\]

The fourth class of activation function commonly used is the family of sigmoid functions. There are two types of sigmoid functions that will concern the user in ocean engineering. The bipolar and binary versions of the sigmoid are both used, depending upon what the nature of the input data is. If data is valued between 0 and 1, then the binary form is used. This is represented by:
\[ Y(k) = \frac{1}{1+\exp(-\alpha^*x(k))} \]

And its derivative by:

\[ \frac{dY(k)}{dx(k)} = \alpha^*Y(k)*(1-Y(k)). \]

Here \( \alpha \) is a coefficient that represents the sharpness factor of the sigmoid. The higher the value of \( \alpha \), the closer to a hard limiter function and the lower the value of \( \alpha \), the closer to the linear function it models.

For the bipolar version:

\[ Y(k) = \tanh(\alpha^*x(k)) \]

And its derivative is:

\[ \frac{dY(k)}{dx(k)} = \alpha^*(1+Y(k))/(1-Y(k)) \quad (\text{Ham, 2001}) \]

For all the work in this research, the linear and the sigmoid are used, depending on the network paradigm selected. \( \alpha \) is modified as well, depending on the problem. For example, in project one, \( \alpha \) was optimized to 1.0. For project 2, an \( \alpha \) value of 1.0 prevented convergence. For that work, \( \alpha = 0.5 \) was used with improved results.

The selection of binary or bipolar is somewhat arbitrary but may be tested in the process by performing two trials, one with each data format on the input and output. Work in this research found that for these datasets, the binary format provided higher confidence intervals. Consequently for this research, both inputs and outputs are scaled to a binary range. Also, since the neuron outputs saturate at 0.0 and 1.0, the data is scaled below this to a range of either 0.0 to 0.9 or 0.1 to 0.9 depending on the project. This allows the maximum use of the sensitivity of the neuron.
Learning Rate

The amount by which weights are changed for each learning cycle is determined by a value known as the learning rate, often represented by the greek letter $\mu$. Because the learning process is iterative, the learning rate must be gradually reduced in order for the network to converge. Several strategies have been employed but for this work, a simple form has been used. For all the work in this research, the learning rate $\mu(k) = \mu_0 * K$, where $K$ is a constant less than 1, usually on the order of $K=.9999$. Here $k$ is the time index, thus signifying that the learning rate is updated for each cycle of training.

Error Criteria

There really are two types of error to be dealt with in the training and operation of the neural network. As will be discussed in section 3.3, there is always a discrepancy between the desired response of the network and the actual response. This error is evaluated for each training cycle and is calculated by several methods. During training, this error is called the *training error*, and for a learning network, should decrease during training to a small value that is usually used as a stopping condition for training (See Exhibit 3.2).
Exhibit 3.2. Typical Training Error Patterns

In operation of the network however, this error is of limited usefulness. In some cases, extremely low error can sometimes reflect that the network, while trained well on its training set, cannot generalize to new cases as shown in a testing set. Certainly, as a rule of thumb, if the training error begins to increase as the network iterations increase, then the system is overfitted and the limits of the network have been reached. Similarly, if a “well trained” network has difficulty generalizing, then it is once again in an overfit condition. At this point, the only recourse the researcher has is to reduce the number of neurons or increase the size of the training data set (Masters, 1993).

The point at which the network is overfit can be used as a stopping condition as well. In a quantitative sense, the network overfit will result in increased average epoch training error as well as rapidly increasing variance of the epoch training error. This is
shown in Exhibit 3.3. Even if the error drops after this peak, the network will exhibit an inability to generalize.

The other type of error that is more useful in determining the functionality of the network is the repeated measure error. Basically, it is simply the score of the network to correctly classify the testing set as compared to the desired answers. For example, if the network correctly identifies 77 out of 100 patterns in testing, then the error would be 33%. A related measure is the confidence level that has a variety of interpretations depending upon the conditions. The reader is referred to an excellent overview of quantifying confidence measures (Masters, 1993). In addition, the concept of confidence interval and confidence level has been thoroughly developed for active SONAR systems (Urick, 1983).

![Exhibit 3.3. Network Overfit Mean and Variance Training Errors](image-url)
In interpreting the data from all the neural networks the following process was used:

1: Verify that training and testing sets have samples from all data categories of interest for that test.

2: After training, assume that the response neuron(s) with the largest output reflects the output of the network. As a practical matter, take the maximum of the output neurons. For this work, no encoding scheme is used, if there are 28 categories of interest, there are 28 output neurons.

3: Score the results for all categories. Report the percentage of correct answers as the confidence level for that experiment. The remaining percentage refers exclusively to mislabeling of an event.

4: If the maximum value in the output is less than 50% of the output span, label that network response as “indeterminate” and count it as a mistake.

This level of accuracy assumes multiple categories. If a network is broken into a series of two category problems, then the analysis methods of SONAR are more useful. In those cases, systems can be evaluated based on the traditional concepts of detection probability vs false alarm probability.

**Confusion Matrix**

One of the strengths of the neural approach is its ability to function in data spaces where the statistical distribution of the data is only partially known. In that environment, it is difficult to quantify from a distribution the probability of a given categorization. A more useful tool to the researcher is the reporting of both the total number of mistakes along with what the mistakes were. This is reported in a matrix table called a confusion matrix, where
each row represents the respective truth categories and each column represents a network selection. Items on the diagonal represent correct responses and off diagonals represent errors. As will be seen in project two, there are certain specific categories which cause the greatest problems, and it is these types of patterns that only show up on the confusion matrix. A simple example would be to make a hypothetical network with 6 categories. It is trained to 80 percent accuracy and is tested with 40 samples. The confidence measure would give 8 incorrect responses and that is all. But in the confusion matrix a pattern emerges (See Exhibit 3.4).

<table>
<thead>
<tr>
<th>Test</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>xxxx</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Exhibit 3.4. Sample Confusion Matrix

In the example, there are indeed 8 errors, mapped by the symbol “x”. But rather than a uniform distribution through the decision space, there is a strong cluster between categories 1 and 4. It appears to be somewhat symmetric, i.e. 1 mistaken for 4 occurs as well as 4 mistaken for 1. As a researcher, closer attention needs to be paid to preprocessing methods which might improve the discernment of the network between these two patterns.

Unsupervised vs Supervised Learning

There are a large variety of neural networks and many ways of classifying them. Parameters such as structure, learning method, output type and ability to learn are all used as parameters to classify neural networks. One of the primary classification schemes used is the way in which the network learns. Unsupervised networks are those that “discover”
patterns in the data by rewarding the neuron connection weights that give the strongest response for a given input. Supervised learning requires previously obtained output examples as exemplars that must be matched by the network for a given input. The details of how each network implements these two approaches are in the field of basic neural network research, but some of the variables salient to underwater recognition are discussion in this section.

For unsupervised networks, the primary structural decision is in the way to update weights based upon the “winning” node. The simplest form of unsupervised network, the Kohonen map, updates its weights by increasing the weights and thus the strength of the connection to the neuron with the highest output at the end of a training example (Fausett, 1994). The variations on this process include awarding the winner and all nearest neighbors in various shapes by the same amount or by reduced amounts. The reader is directed to a variety of works which discuss the tradeoffs in detail (Fausett, 1994, Haykin, 1999, Lin, 1998). The primary decision about if the network has learned or not is based on the relative change in weights for each run. If \( W(k+1) \) is close to \( W(k) \), for a given iteration, usually determined by the Euclidian norm of the change of \( W \), then the network is described as trained to within this norm.

Learning method

For the supervised networks, there is not only the network to choose from, but the way in which weights are updated and also the possibility of adapting the activation function as well. It is these changes that are referred to as learning method. Another feature is the way in which error is evaluated. There is now error because the training
data provides the desired outputs for a given input. One of the most frequently used learning methods is called the *Least Mean Squares* (LMS) algorithm and it is the one used for all the supervised networks in this research. In this method, based on the computation of the mean square error (MSE) of the responses of the network to a given input, the LMS algorithm uses the gradient of this MSE calculation to point to the next values to use in the network weights for learning. If one visualizes the values of MSE as forming a surface for all ranges of inputs and weights, then there will be one or more places where this error surface is a minimum. By following the gradient descent to this minimum, the system can converge at some point to an optimal answer. However, parameters such as learning rate, $\mu$, affect the speed at which the system descends to this optimal position as well as how close the system can approach the best value. Too small a learning rate and the system proceeds to convergence very slowly. Too large a learning rate and the system passes over the minimum, then oscillates back and forth over the minimum, unable to reach the best value.

In summary, one selects the family of neural networks, unsupervised or supervised, then the structure, the way in which the system learns, and finally, the error criteria used to optimize the network. At this point, one may begin to evaluate the suitability of the given network to solve the problem of interest.

**Self Organizing Maps (SOM)**  
Project one examines the problem of enhancing discrimination of passive SONAR contacts. The probability of recording every possible sound category in the ocean is difficult if not impossible to achieve. This has two impacts on the choice of network paradigm. First, the possibility that the testing data set will be comprehensive is small. In
addition, the chance of even knowing what all the categories of interest might be is also small. Given these two issues, unsupervised learning and specifically, the Kohonen self-organizing map (SOM) approach was selected for use in this project.

The simplest unsupervised network possibility is that of the Kohonen network. The network, developed by Tuevo Kohonen, was originally used for recognition of phonemes in the Finnish language (Fausett, 1994). The network is a simple two layer network, without even a nonlinear element at the output of the neurons (See Exhibit 3.5).

![Kohonen Self Organizing Map](Exhibit 3.5. Kohonen Self Organizing Map)

The network relies on a brute force approach for classification of patterns. It relies on the principle of competitive learning. As mentioned above, this learning style rewards the output neuron with the largest output to a given input by strengthening the weights to that output. In variations of structure of the network, only the winning neuron may be
rewarded, or the neurons surrounding the neuron may also be rewarded. The distance to the farthest rewarded neuron is called the *neighborhood* of the neuron (See Exhibit 3.6).

![Exhibit 3.6. Neighborhood Definition](image)

Much like the “search then converge” changes to learning rate for certain networks (Ham, 2001), the Kohonen map usually varies the neighborhood distance during learning. In early training iterations, the neighborhood function is large (2 or 3). After this *ordering* phase, the neighborhood is reduced eventually to only the winning neuron itself (neighborhood=0). The second portion of training is referred to as the *convergence* phase and is where the network remains until training is terminated (Ham, 2001). The network error criterion for termination is when the ensemble weight changes for consecutive patterns become smaller than a predetermined threshold.:

\[
    w_{ij}(k+1) - w_{ij}(k) < \text{Threshold}
\]

The standard weight update equation is used for the winning neuron:

\[
    w_{ij}(k+1) = w_{ij}(k) + \mu (x_i(k) - w_{ij}(k))
\]
and for the non winning nodes:

\[ w_{ij}(k+1) = w_{ij}(k). \]

**Mapping Variability**

One issue of the unsupervised networks in general and the SOM in particular is the variability of specific mapping categories. In a typical supervised network, each mapping is assigned a category descriptor, usually a number, i.e. “1” for a whale, “2” for a ship and so forth. Since the SOM is unsupervised, its training is meant to discover the structure of the data space, not to map to specific values a priori. As a brief example, imagine three, well defined data sources each with unique characteristics. Furthermore, imagine that several samples have been extracted for use in training and testing sets for an SOM. Let the SOM have 5 output neurons (See Exhibit 3.7). The network is trained on the same training set, three separate times. Because the weights are initialized to small random values, the category which “Wins” the mapping for the first exemplar will in general be different for each training run. As the training progresses, this initial winner becomes reinforced. At the same point in each training evaluation, the network has converged to the same degree. Yet the mappings are different for each training trial.
Now once a given training session is complete, the mapping will remain constant for those categories. That is, if for that run, Object 1 maps to neuron A, then for every example of Object 1, it too will map to Neuron A. If the network is re-initialized and trained again, it may be mapped to C for that example.

Another observation is important here. Notice that even with different details of the mapping for each training session, the network always discovers 3 categories. This is a valuable tool in using an SOM to evaluate data preprocessing methods. The network will tend to find a definite number of classes in the input data. If the network architect wishes, these may be mapped into relatively few classes and the SOM helps to serve as a data reduction tool. If the designer uses a large number of output categories, then the network will only map to the number of categories for which it recognizes patterns as existing. For SOM, adding extreme complexity to the network does not improve performance.

---

1 The author wishes to coin a phrase “natural categories” as those classes of data which a map can find on its own, as opposed to “artificial categories” as those whose structure is imposed on a neural network via supervised learning.
This tool can be used to determine if a preprocessing step has value and also if additional processing is required. For example, in Project 2, the data patterns seen in geophysical parameters are discussed. Prior to performing the experiments outlined in Chapter 6, the data was presented to the SOM1D function. Given that there were 6-7 desired categories in the data stream, the network could at best, only recognize three categories as existing. If there is insufficient difference between two artificial categories, then even a supervised network will have difficulty learning. It is up to the network architect to assess this and spend time on preprocessing strategies to enhance the differences between categories in order to simplify the neural network design and operation. Indeed in the Project 2 case, the artificial categories of entering a region and leaving a region were the most difficult to resolve. As a test of the network, if the SOM map provides consistently fewer categories than desired, other neural paradigms must be evaluated.

In summary, the SOM network is an unsupervised network paradigm which learns the structure of the data space to certain limits based on the size of the network and the resolution of the input data. Since the problem of phoneme recognition (Kamm, 1990) is very similar to the recognition problem of acoustic signals in the water, the SOM is used for initial evaluation of the passive acoustics problem.

**Multi-level Perceptrons (MLP)**

**Basic Structure**

The workhorse of the neural network field for the last ten years has been the multi-level perceptron (MLP) structure. Its roots go back to the Rosenblatt Perceptron in 1958.
(Haykin, 1999). After work in the early eighties (Parker, 1987) on training strategies, the backpropagation training method was established. Often, the literature will mistakenly refer to the MLP as the “backprop” network because of the prevalence of this training method.

There are several variations to the MLP structure (See Exhibit 3.8). The most common variation involves selecting the number of hidden layers. It has been demonstrated that with 3 total layers, the MLP paradigm can perform as a universal approximator of non-linear mappings, given a sufficient number of neurons and training data (Masters, 1993). This then is the structure that is used in this work whenever “MLP” is evoked. Within the structure of the MLP network, there are several parameters which can be varied to improve the performance of the network.

![Exhibit 3.8. Multi-Level Perceptron](image)

First, the number of neurons in each layer can be varied. For the output layer, the number of neurons must match the desired number of outputs. On the input side, the same restriction does not always apply. The number of input neurons can be made larger than the number of inputs available. For all of this work, the number of input neurons is held to the same value as the number of inputs.
Perhaps most importantly, the number of hidden layer neurons determines the functionality of the network (Fausett, 1994). Too few neurons, the network cannot learn adequately. Too many neurons, and the network becomes slow to converge and can result in an “overfit” condition, in which the network has learned the training data set so well, it cannot generalize and thus gives poor testing results. In addition, it may even detect patterns that are not there as a result of the overfit. The closest analogy would be that of an individual staring at a blank wall for hours on end. With so little data to stimulate the brain, eventually, that individual will begin to see patterns in the wall! From a theoretical perspective, where the optimal number of hidden nodes N<sub>m</sub> can be expressed as:

\[ N_m = \log_2 M_{\text{max}} \]

for N hidden layer neurons and M<sub>max</sub> separate classes (Lin, 1996), for 7 categories, as in project 2, the number of hidden nodes would be 2.8074. For the purposes of this research, early “tuning” tests were performed, varying the number of hidden layer neurons n<sub>2</sub>, from half the number of input neurons, n<sub>1</sub>, to double the number of input neurons. Optimal results occurred with the number of hidden neurons equaling the number of input neurons. So as to focus on the data and preprocessing strategies, for all tests using an MLP network in this research, the values of n<sub>1</sub> and n<sub>2</sub> are the same as the number of inputs, n<sub>0</sub>. The number of output neurons equals the number of output categories, n<sub>3</sub>. All the code written for this project first measures the dimensionality of the input data, and resizes the neural network to match the input size of the data. The number of output categories is manually set according to the specifics of the experiment.
Training and Convergence

As mentioned, the typical training method for the MLP is via backpropagation of the errors to the inputs and subsequent updating of the weights. One of the difficulties of the MLP network is that it sometimes is difficult to converge. Though there are several methods which approach the problem on the weight update side, these still do not address the relative instability due to variations on the input. For the methods such as momentum learning and conjugate gradient descent, the reader is referred for further study (Ham, 2001, Haykin, 1999). For the purposes of this research, only three enhancements to the basic LMS algorithm were required.

The first enhancement is that the initial weights of the network must be within a range that the network can converge rapidly to the minimum error condition. All weights must be at first randomized, but this raises the question of what value to randomize to. Nguyen and Widrow have provided an initialization method that bases the initial weight values on the size of the network and the randomized values initially used (Nguyen, 1990). The complete initialization of the weights is performed according to the following process.

Define $n_1$ to be the number of input neurons, and $n_2$ to be the number of neurons in the hidden layer. Let $\gamma$ be a scaling factor. The Nguyen-Widrow Initialization then performs the following steps:

1) Compute scaling factor: $\gamma=0.7*n_2^{1/n_1}$
2) Initialize the weights of a layer, as random numbers $-0.5 < w_{ij} < 0.5$
3) Reinitialize the weights as:
   \[ w_{ij}=\gamma^2 w_{ij}/S(\sum_{i=1}^{n_1} (w_{ij}^2)) \]
4) For the ith neuron in the hidden layer, set the bias to be a random number between \(-w_{ij}\) and \(w_{ij}\). (Ham, 2001)

For this body of work, this single change along with proper input scaling resulted in extremely robust MLP responses project 2 emergent categories in chapter 6.

The second enhancement is the presentation of inputs. For one group of problems as in chapter 6, the use of strictly binary inputs scaled to the response of the neuron proves effective. For cases where the input is analog, as in project 1, normalization of the data and biasing the data to the optimal response of the neuron is important.

The third enhancement for the network is learning rate. The learning rate determines how large the changes are in the weights from one training pattern to another. If the weight is too large, the network will “overshoot” the minimum and oscillate back and forth around it. If the weights are too small, the network cannot converge, even after several hundred thousand training cycles.

After initially implementing the first two enhancements described above, it was determined that a simple fractional reduction in learning rate would be applied to each cycle. Therefore, for all the networks in this work use a simple rule. If \(\mu_0\) is the initial learning rate, then at the end of each training pattern, \(\mu_0\) is multiplied by a fractional coefficient, \(\varepsilon\). For this work, typically \(\mu_0\) is between 0.5 and 0.99 and \(\varepsilon\) is between 0.999 and 0.99999.

The actual algorithm for the backpropagation of errors and weight updates is standard for the MLP network. The following notation is that used by (Ham, 2001).

Define the following variables:

- Present time index: \(k\)

- Future time index: \(k+1\)
Neuron layer index: j
Neuron index: h
Number of neurons in layer j: n(j) or nj
Layer input index: i
Input vector: x(k)
Output vector: Y(k)
Weight matrix for layer j: w(j)
Linear summation of inputs*weights for layer j: v(j)
Output of nonlinear element for layer j: x(j)out
Desired output vector: d(k)
Error vector: E(k)
Non-linear element function: f(\(\cdot\))
Derivative of non-linear element: g(\(\cdot\))

Given the structure of a multi-level perceptron (exhibit 3.8), an input pattern, x(k) is applied. The network proceeds via feedforward to sum the weighted connections, process the non-linear elements and send the signals forward layer by layer until an output pattern is generated Y(k). The ensemble error E, is calculated as (in vector form):

\[ E = \frac{1}{2} (d - x^{(3)}_{\text{out}})^T (d - x^{(3)}_{\text{out}}) \]

The variation of the error with respect to the linear summing output \(v_h^{(3)}\) of the hth neuron is computed (in scalar form) for the output layer (3):
\[ \frac{cE_q}{cV_h^{(3)}} = -(d_{qh} - x_{out,h}^{(3)})g(V_h^{(3)}) = -\delta_h^{(3)} \]

This factor, \( \delta_h^{(3)} \), is then used to compute the updated weights for layer (3) as:

\[ w_{hi}^{(3)}(k+1) = w_{hi}^{(3)}(k) + \mu^{(3)} \delta_h^{(3)} x_{out,i}^{(2)} \]

where \( h \) and \( i \) are indices from the \( i \)th input to the layer to the \( h \)th neuron.

Going to layer two, the process is repeated, but an additional summation is required since errors from a single output backpropagate to multiple previous neurons.

For all remaining layers, \( s = 1, 2 \), the form is:

\[ w_{hi}^{(s)}(k+1) = w_{hi}^{(s)}(k) + \mu^{(s)} \delta_h^{(s)} x_{out,i}^{(1)} \]

with:

\[ \delta_j^{(s)} = \sum_{h=1}^{n(s+1)} \delta_h^{(s+1)} w_{hj}^{(s+1)} g(V_h^{(s)}) \quad (\text{Ham, 2001}) \]

At the point where one reaches the index \( s = 1 \), the weights have been all updated, and the next input pattern \( x(k+1) \), is applied.

**Hybrid Networks**

One of the more novel approaches to use with neural networks involves the hybridization of two kinds of network. Several types of network combinations have been evaluated on a limited basis (Ghosh, 1992), but none for ensemble performance. In this paper, the method has proven useful in Project 1. The idea behind the hybrid network is to cascade stages of similar or dissimilar network paradigms to function as building blocks in analyzing a complex data structure. For example, the Hybridnet program developed here utilized the SOM program as a front end to reduce the dimensionality of
the analysis problem for the MLP from a problem of separating dozens of spectrogram
types to re-mapping the categories of the SOM to the desired categories provided by the
architect (See Exhibit 3.9).

![Exhibit 3.9. Hybrid Network Block Diagram](image)

Notice that the function of each network is different. The SOM layer is grouping
sounds by distinctive sound characteristics. The MLP now has the task or organizing
those sound classes into the entities desired to be recognized.

In terms of network size and speed, the results are impressive. For example, if one
computes a network size by the number of weights required, then the SOM, MLP, and
Hybridnet can be compared. For a 256 x 10 input array and 25 output categories the
following sizes can be calculated. For the sake of simplicity, let the number of internal
SOM categories equal the number of MLP categories, 25. The results are:
These results show that the hybrid approach can have some beneficial results. In addition, each stage of the network can be modified with a different paradigm to yield the desired results.

One of the other advantages to using a hybrid network is improved structuring of the input to the MLP. As mentioned in the literature (Ham, 2001), the MLP network is sometimes slow to converge and difficult in some cases to converge on the global minimum error throughout the data space. That is why much of the research in MLP networks has gone into improving convergence speed and reliability. However, by limiting the input set to strictly binary values, the network operates adequately for general use.

One cautionary note; In some cases, there is subtlety in the data which is lost in the hybrid network (Masters, 1993). By performing the massive dimensional reduction in the first stage, only the predominant features remain. This can be of benefit in improving the speed or size of the system, but does reduce the ability of the network to gain information from subtle detail in the input data.

**Comparison/Tradeoff Analysis**

There are two major ways to organize the neural network paradigms in order to select the most promising candidates. First, one must decide whether supervised or unsupervised learning is most appropriate to the problem. Secondly, the exact nature of the problem must be defined. There are two distinct types of problems for which neural
networks excel. The first is data classification, which involves organizing and in some cases reducing large amounts of raw data into a manageable structure. The second is mapping, which involves reorganizing data from one form to another, hopefully more useable form for the end user. One must define explicitly what steps are involved in the specific problem to decide if a network is the best paradigm. For example, MLP was not considered for the front end processing of acoustic signals specifically because of its large size and difficulties in convergence. But in mapping the primitive categories of sound or structure to more complicated features or structures, the MLP was the ideal choice. In neural networks, perhaps more so than with any other analysis tool, the problem statement must be carefully defined in order to select the right tool.
Data Preprocessing

Introduction
As referred to several times in Chap 3, half of the work in neural networks includes processing the data in an appropriate fashion. Some of the tasks include scaling, filtering, and transformation of the data. The goal is to enhance any features within the data set to maximize the performance of the network in recognizing those features. In addition, by combining techniques, enhanced performance may be achieved via data fusion (Brooks, 1998). These methods are classical methods taught in most science and engineering curricula. This chapter outlines some of the issues and methods most pertinent to this research. They are only here to identify the approach taken by the research to date. These concerns have either been addressed in this body of work or are part of the next phase of enhancement to the larger body of work.

Neuron Requirements
In order to take advantage of the transfer curve of the sigmoid, the neurons are scaled to operate with either a binary \((0 < x < 1)\) or bipolar \((-1 < x < 1)\) input and output range. In order to take advantage of the neurons’ response curves, it is important to scale both input and output into a range compatible with the above values. For the binary mode selected for this paper, the range of \(.1 \text{ to } .9\) is usually chosen (Masters, 1993). This scaling is applicable for all paradigms and all types of preprocessing inputs.
Temporal/Spatial

A careful study of the data must be made since there is ambiguity in most ocean research. Because the scale of many measurements commonly made in the ocean sciences is so large relative to the instrument, it is unlikely that measurements will be made concurrently for a given feature of interest. This is particularly difficult in passive SONAR applications for boats and submarines (Baran, 1991). In addition, for Project 2, there is also an ambiguity between spatial sensor events (thermocline) and temporal events (sunset). If it is anticipated that such an ambiguity will exist, then additional care must be take in the design of the network. For example, in the passive SONAR application, the output of the network must be combined in subsequent processing with other information. The sound signature of a submarine and a boat overlap at different speeds and distances. Therefore, the neural network will have an ambiguous answer for strictly the passive SONAR signal. By adding an active channel which can provide range information, or by examining the relative frequency attenuation using analytical methods, the ambiguity can be resolved.

In the case of the temperature ambiguity, it may require that time explicitly be included as an input to the network. Events happening in a short time window would then map to one category, while long window events would map to another.

Denoising

Noise is of particular importance in oceanic signals. A wealth of resources is available on general denoising of signals (Ham, 2001, Medwin, 1998, Misiti, 2000). Use of various filters as well as “whitening” of input signals produces useful results in denoising a data sample (Ghosh, 1992, Ham, 2001). For this body of work, the raw
signals of the data samples under investigation were applied to the network processing structures to get “worst case” evaluation of the system performance.

The noise issue may be addressed at several levels. In the case of passive SONAR data, the first level is at the hardware electronics level. The next level is by software preprocessing of the input using traditional filter methods. The last level is by training the network to signals with a variety of added noise. The MLP network of project 2 for example, demonstrated adequate (>70%) recognition rates with up to 10% noise added.

Envelope Detection

The information stored in oceanic data is at multiple levels. In early work, the use of frequency domain techniques has proven reliable when the stationarity of the signal is longer than the period of measurement (Ham, 2001). However in the case of acoustic signals, additional information is gained not just in the individual sound events, but also in the larger scale of when those events occur. Therefore, the envelope of the signal is useful as additional information for the neural network. In detection of fish, this forms the basis of wavelet spectra as well as location of individual pulses in the signal.

One requirement for the fish analysis was the gathering of pulses for training and testing. This would also apply to any highly episodic signal. For signals which consist of a train of pulses, the energy in the signal is concentrated in these pulses. If the signal is sampled with uniform sampling intervals, several samples will consist of one pulse with long spaces of noise, such that the signal energy of the sample is small. To the neural network, this would create a training and testing sample series which contain mostly background noise as opposed to the signal of interest. In some manner, the training and
testing data must consist of acoustic signal from the pulses themselves and not the background noise. To that end, an energy averaging threshold system is used, much as a primitive “squelch” circuit for the incoming fish signals. For this research, this function was coded in the Matlab function “ENVELOPE_THRESH”. When the energy in the wave exceeds a certain threshold, a “one” is placed in a sample file which is of the same size as the number of time samples. Ones are then added to this “bit masking” file until the energy returns below the threshold level. After the average signal drops below the energy threshold the masking file samples are filled with zeros. After the completion of the run, the bit mask is “anded” with the original data file and the pulses are extracted (See Exhibit 4.1). Every time sample that has an equivalent “1” in the masking file is retained and every time sample that has an equivalent “0” in the masking file is replaced with a “0”.

In addition the bit mask file now has a record of the spacings of the pulses. By locating the transitions between “0” and “1” in the masking file, the time index for the start and stop locations of the pulses may be determined. The signal is then compressed by removing the dead space between pulses, leaving only a rapid sequence of pulses with little if any ambient noise alone.

Though performed in batch mode for this project, this method was the most amenable to conversion to real time use. Other methods such as the whitening methods (Ham, 2001, Ghosh, 1992) require batch operation of the processor. In order to implement this in real time, a long (several second) buffer for the incoming time signal would be required in order to determine the statistics of the signal to proceed with whitening.
Exhibit 4.1. Envelope Filtering of Pulses

Regression/Curve Fitting

Another method evaluated for reducing noise effects were the concepts of regression or curve fitting. In the case of project 2, the noise on the input data resulted in incorrect mapping of features at all levels. One approach to dealing with noise corrupted data is to perform a regression analysis on the data. This method was applied using the “polyfit” function in Matlab. For example, if a cubic polynomial is attempted, the result is a much smoother curve for the neural network (See Exhibit 4.2).
Exhibit 4.2. Use of Polynomial Regression to Smooth Data (cubic)

For general use as a preprocessing tool, two issues exist for the user. First, what type of regression analysis is to be used? Secondly, how is this data to be presented to the network? Among the options are:

1) Present only the regression coefficients
2) Present the regression function over the span of the inputs
3) Present the original corrupted data and the regression coefficients
4) Present the original corrupted data and the regression version.

In this research, all four approaches outlined above were attempted, and the more successful candidate results in terms of recognition rate are reported in chapter 6.

**Frequency/Spectrum Analysis**

For the purposes of project 1, the primary preprocessing method was Fourier transform analysis. Fourier analysis consists of the breaking up of a complex signal into a continuous sum of sinusoids of varying frequencies. It is written in classical form as:

\[ F(\omega) = x(t) * e^{-j\omega t} \text{ integrated from } -\hbar \text{ to } +\hbar \quad (Selby) \]
For many oceanic signals, transfer to the frequency domain results in obvious patterns emerging. For neural networks, each frequency or frequency bin can become an input for a network. In this work, two projects rely on frequency domain analysis. Project “1” deals exclusively with passive acoustic problems. In that realm speech processing and active SONAR have demonstrated for years that maximum discrimination of differences between signals occurs by looking at the frequency composition of the signals (Medwin, 1998, Urick, 1983).

Wavelet Analysis

Wavelets

Wavelets, a much more recent invention, are seeing extensive use in signal processing. Whereas the Fourier transform is made up of sinusoids that exist for all time, the wavelets are made up of small sinusoid “packets” that exist for a finite period of time and have a particular “frequency band” associated with it. Thus they are localized in time and frequency. A transformation from the time domain to the “wavelet” domain analogous to the Fourier transform is defined as:

\[
C(\text{scale,position}) = \int x(t) \times X(\text{scale,position},t) \, dt \quad \text{(Misiti, 2000)}
\]

where \( X(\text{scale,position},t) \) is the wavelet used to transform the time domain signal. The “C” function is a sequence of real amplitudes, here called coefficients, analogous to the Fourier transform being a function of amplitudes versus frequency.

The wavelets themselves are modified during the transformation. In the equation above, the wavelet is a function of “position” and “scale”. By scale, the function refers to
the degree of time expansion of the wavelet (See exhibit 4.3). In this exhibit, a Coiflet waveform is expanded to 6 scale levels resulting in stretched versions of the waveform. Typically, multiple scales of wavelets are used in turn for a given transform.

Exhibit 4.3. Coiflet order 3 Scale Expansion

To perform the wavelet transform, the signal of interest is compared with the wavelet chosen for the transform. The “degree of similarity” is measured, that is the correlation of the wavelet to the waveform over the sample interval with the wavelet at the specific time location, and the magnitude of this is recorded as a coefficient value “C”. A string of coefficients is created with one scale of wavelet, then the wavelet scale is changed and the process is repeated. In this manner, the “C” vector reflects the similarity of the signal to the wavelet at a variety of scaled wavelet factors. The process of applying the wavelet at different scales occurs iteratively and can be thought of as a filtering operation (see exhibit 4.4).
Exhibit 4.4. Filtering effects of wavelets (Misiti, 2000)

In this example, a chirp with noise is applied to a single level wavelet transform, the Haar wavelet. Notice that there are two blocks of coefficients in the lower right box. The bottom looks like an intensity picture of the original picture in the upper right. This is called the *approximate* “Ca” signal in wavelet jargon. The top row reflects the coefficients of the residuals (in wavelet terms the *details* “Cd”), and the remaining time signal is shown at the lower left. These reflect the transformed signal and the residuals. For higher levels of scaling, the signal is passed on and processed the same way with a scaled version of the wavelet. This is repeated as required to extract the features of interest in the specific waveform. It is the coefficients from these layers that are plotted in the lower right of the figure. This becomes the equivalent of the Fourier transform as the input to the neural network system.
In general then, the result of this decomposition by repeated application of the transform above is a series of residual functions such that:

- \( \text{Signal} = \text{Approx}1 + \text{Detail}1 \) 1 scale
- \( \text{Signal} = \text{Approx}2 + \text{Detail}2 + \text{Detail}1 \) 2 scales
- \( \text{Signal} = \text{Approx}3 + \text{Detail}3 + \text{Detail}2 + \text{Detail}1 \) 3 scales

**Types of Wavelets**

There are an infinite variety of wavelets available. Two of the more applicable for this research are the Discrete Meyer (DMEY) wavelet, and the Coiflet wavelet (COIF) (See Exhibit 4.5). The exhibit shows one scale of the given wavelets. Within that family there are several phases and periodicities of the wavelet, which when combined, form an orthogonal set which can serve as the basis for covering an entire data space.

Exhibit 4.5. Coiflet 3 and Discrete Meyer Wavelets (Misiti, 2000)

Reports have shown that the Coiflet wavelet is useful in fish classification (Wood, 2002). In that article, the particular wavelet used was not specified nor was the coefficient ordering method.
Use/Comparison with FFT

In the context of neural network application, the wavelets can be used much like the FFT algorithm. A wavelet transform can be applied to a waveform of interest and a set of coefficients are generated. Depending upon the level of the transform, there may be multiple sets of coefficients. These arrays are placed end to end to form a one dimensional array of coefficients that defines the magnitude of the wavelet transform. Alternatively, the continuous wavelet transform creates a 2 dimensional array with coefficients on 1 axis and scale factor on the other. These presentations are unique for many waveforms. This wavelet array is then applied to the inputs of the neural network in a similar fashion to time, spatial, or FFT inputs. Care must be taken to follow all the normalization guidelines as was used for all other input types. Other options include pulling only the nonzero coefficients and using the indices of these coefficients in descending order to make a feature vector for the waveform (Wood, 2002). A first attempt at using Coiflet wavelets was performed here on fish passive acoustic signals, but a comprehensive analysis of wavelets for feature vector extraction is beyond the scope of this work. Another usefulness of wavelets but also reserved for future work is in the denoising area of low frequency signals (Misiti, 2000).

Data Preprocessing Conclusions

This section briefly introduced methods of preprocessing data appropriate to the neural network problems of this dissertation. After identification of the network paradigm to use, the preprocessing methods are the next most important step in the system design.

For optimizing network convergence and response, scaling and mean centering the data is necessary. For removing noise, either classic filter approaches, or regression
methods should be applied. Lastly, in order to enhance the presentation of the data, transformation of the data from the time or spatial domain to the frequency (Fourier) or wavelet domain, can enhance recognition.
Project 1: Passive Acoustics

Introduction

Sound in the marine environment is like light in a terrestrial application. SONAR, both active and passive, is the dominant detection and exploration tool in all marine applications. Not surprisingly, neural networks have been evaluated for issues such as buried mine classification and SONAR return classification (Malkin, 1993, Ramani, 1992, Robinson, 2003). Application to biological active acoustic systems such as dolphin echos was also examined in the past for modeling purposes (Moore, 1991). However early research papers indicated the possibility of using neural networks in passive SONAR applications as well (Ghosh, 1992, Baran, 1991, Beck, 1991, Pao, 1991). In these papers, data sets were limited to either two category data, or to a simulated test data set of tones meant to simulate actual oceanic signals. Ghosh (1992) outlined the difficulties of dealing with short duration oceanic signals because they:

1) “are highly non stationary and impulsive;”
2) “show significant variations in spectral characteristics and SNR due to differing sources or propagation paths, and due to multi-path propagation;”
3) “may overlap with one another”
4) “show rapid variation of spectral characteristics with both frequency and time; and”
5) “often require event association for proper identification” (Ghosh, 1992)
More recently, the use of neural networks specifically for fish identification has been examined (Lin, 1998). In this paper, with very limited test sets, the results were encouraging. In Lin’s work recognition rates of 50-90% were achieve for a limited sample of Black Drum, Perch, and Sea Trout sounds (Lin, 1998). In Ghosh’s work on standardized tone and pulse training signals used to test networks for the military, recognition rates from 96.1% to 100% were achieved (Ghosh, 1992).

As mentioned in section 1, one of the purposes of this research is to evaluate the behavior of the networks in the full spectrum of oceanic inputs. In the context of the passive SONAR realm, this means looking at all classes of sounds. The goal is then to develop a common approach to processing these sounds for applications as disparate as fisheries research and nuclear test verification (Wood, Ham, 2001).

The research for this project is presented in a sequential fashion, reflecting the order in which it was performed. It is instructive at each point to see the strengths and weaknesses of each method as it highlights some characteristics of the sounds as discussed below. First, the Kohonen Map will be evaluated. Then based on those results, the multi-level perceptron will be used. This will then generate the basis for Hybridnet, and the results from this structure will be discussed.

On the preprocessing issues, once the network topology is frozen, different methods (FFT, wavelet) will be evaluated for recognition rate. Finally, the difference between ensemble recognition and distributed recognition will be briefly examined pointing the way to future research.
Sound Data Sets

To determine if the neural network approach would be viable, it was necessary to determine the characteristics of the source signals encountered. The mission planning and hardware also determine the bandwidth to be observed and characterized.

Previously recorded sound sources were acquired through various Internet and physical sources as mentioned. The organizations offering recordings included NOAA, Cornell, Cetacean Research, and even a submarine enthusiast’s site in Japan (Steel in the Deep, 2003). These were received and stored in a variety of formats, specifically as digitized sound in MP3, AU, and WAV format. MP3 and AU files are a compressed format while WAV is not. After reviewing these data sources, it was further discovered that many of these recordings were oversampled when compared with the original bandwidth. For example, many of the University of Rhode Island files were analog recordings which had been filtered to <750 Hz, yet were stored as 44 khz sampled streams. Exhibit 5.1 summarizes the sources and the original data format.

Having completed this review, it was also necessary to examine the means by which future Florida Tech field data would be recorded. Since the desire was to eventually make this low cost as well as autonomous vehicle deployable, conventional PC sound cards in a PC/104 form factor were selected for the initial study. These cards can sample in stereo to 44 khz sampling rate, thus providing a 22khz bandwidth. For non autonomous work, the fact that DAT recorders are on the decline, forced a decision to use new, hard drive style recorders. After review, it was discovered that only one vendor, Creative Labs, produced a data recorder, the NOMAD 3 system, which could record and store in an uncompressed (WAV) format.

Since the emphasis of this work was on the neural network processing as opposed to high frequency research, it was decided to limit the bandwidth for field recordings to 24khz (48 ksps), the maximum allowed by the NOMAD. Any additional bandwidth reduction would be performed just prior to application of the neural network. In addition, the sound source files would be re-sampled to the lowest data rate of the set. Thus, the data sets were all made uniform to 8ksps data rates, 16 bit PCM encoding, monaural. This is not seen as a major limitation to this research as much of the information involved in the recognition falls from D.C. to 4khz with the majority of the signal information below 1 khz.

Sound Characteristics

Having outlined the technical aspects of recording and storing the file information, attention is now turned to the sound itself. Sound in the ocean comes from three major categories.

The first, man made, involves mostly machinery sources from boats and subs as well as the equipment related noises of divers. Typically, these sounds consist of regular vibrations with only a few harmonics (See Exhibit 5.2). These sounds tend to have a
much longer stability then do biological sounds. Sound sources in this category include
propeller noise, hull transmitted engine noise, fin swishing of a diver, and various pump
and motor sounds. For military and harbor security applications, these are the sounds of
the utmost interest.

<table>
<thead>
<tr>
<th>Agency</th>
<th>Program</th>
<th>Data Type</th>
<th>Sample Rate</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOAA</td>
<td>VENTS</td>
<td>Ship</td>
<td>8khz/16bit speedup</td>
<td>Conv. 8kbps 16kbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ship at Hphone</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quake</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artec</td>
<td>DolphinEar</td>
<td>Boat</td>
<td>88-176kbps WAV</td>
<td>Conv. 8kbps 16kbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Humpback</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Whale</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dolphin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cetacean Research</td>
<td>1997,1998</td>
<td>Orcas</td>
<td>MP3 32kbps</td>
<td>Conv. 8kbps 16kbps</td>
</tr>
<tr>
<td></td>
<td>survey</td>
<td>Bivalves</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Boat</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Humpback</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Humpback</td>
<td>.AU 64kbps</td>
<td>Conv. 8kbps 16kbps</td>
</tr>
<tr>
<td>Acoustical Society of</td>
<td>Demo.</td>
<td>Humpback</td>
<td></td>
<td>Conv. 8kbps 16kbps</td>
</tr>
<tr>
<td>America</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submarine Project</td>
<td>Steel in the</td>
<td>Sub Sounds</td>
<td>Mp3 64kbps</td>
<td>Conv. 8kbps 16kbps</td>
</tr>
<tr>
<td></td>
<td>Deep</td>
<td>C-Launch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOEST Hawaii</td>
<td>HUGO</td>
<td>Volcanic</td>
<td>.AIFF 8kps</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Submersible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Univ of Rhode Isl.</td>
<td>Archive</td>
<td>Fish</td>
<td>CD audio various</td>
<td></td>
</tr>
</tbody>
</table>

*Exhibit 5.1. External Sound Sources Used*

Yet the more complicated sounds lie in the other two categories. The geophysical
sounds are grouped in that they are generated by natural geophysical processes. The wind
related Knudsen noise, rainfall, tremor and earthquake sounds, as well as sand scrub and
surf noise near inlets and littoral zones all contribute to this category. They are denoted by long period episodic envelopes (Medwin, 1998).

Exhibit 5.2. Ship Frequency Spectrum

The most difficult sounds of all are the biologically related sounds. These involve three main sub groups. First, the invertebrate sounds, such as barnacle and snapping shrimp noise, is a familiar sound to anyone who has spent the night on a boat in warm climates.

These sounds are closer to the man made and geophysical sounds in that they occur for long durations and are not modulated heavily during the length of the signal occurring in a broad band manner below 4 khz.

Next are the fish sounds (See Exhibit 5.4). These sounds come from more than a hundred species, and are generated in several manners, from bladders, fins, and during certain regular behaviors such as feeding.
These sounds are highly episodic, short duration, and contain many harmonics. These are very difficult to work with using conventional methods.

The last biological group is the sound from Cetacean sources. Every whale or dolphin in the world exhibits complex vocalization patterns each very distinctive by species, behavior, and individual. These sounds are characterized by being highly variable, episodic, and having much recognition information packed into the longer string of vocalizations (see Exhibit 5.5).
In addition, there really are two sets of sounds to be dealt with in this group. Cetaceans use two broad bands for different purposes. For vocalization and routine behaviors, much of the sound is below 10 kHz. For echolocation, the sounds begin at 20 kHz and can exceed 100 kHz in some cases. For this research, the sounds used for recognition are those below 4 kHz.
Exhibit 5.5. Humpback Whale Time and Spectra
Paradigm Selection

In choosing the neural network paradigm for this work, several factors needed to be considered. It is unreasonable to expect to have every sound the network will encounter in the ocean available for training and testing. As a matter of fact, one of the missions envisioned for this tool is the exploration and discovery of new sounds and thus new features in the ocean, be it of geophysical or biological origin. Therefore, the network needs to operate in a fully autonomous manner and also be able to identify an unknown category as unique, for the purpose of either undergoing additional training in situ to categorize the sound, or trigger an event whereby the sound event may be documented using all the instrument’s sensors.

Another issue based on this variability of the input space is the ability of the network to converge during autonomous training. The workhorse paradigm, MLP/BP network, has severe problems with convergence if the training data is not managed properly. In some cases, the learning strategy must include scaling of the sharpness of the sigmoid as well as the network weights again adding to sample time and complication.

Two additional constraints are the size of the network and the speed of training for the network. In previous work, researchers have limited the size of the inputs to 8 frequency bins or up to 24 coefficient feature extraction (Lin, 1998, Ghosh, 1992). It is desired to have a network fast enough to train in 1.5 hrs maximum with the full data set up to 200,000 training iterations and which can fit in a small memory footprint.

Given this, the first candidate is the Kohonen self organizing map. Not only can it be trained in an unsupervised manner, it also is very stable and converges well, even with poorly structured data on the inputs. The SOM also can serve as a classifier, such as in its original intent of recognizing phonemes (Fausett, 1994). One unique feature of this
approach will be the selection of input and output dimensionality. It is chosen to represent each class of sound as a single output category. Thus, for 28 sound sources, the number of outputs would be at least 28 or more classes for a single, ensemble neural map.

On the input side, data will be presented from the pre-processor as one or two dimensional data. Therefore, two versions of the SOM exist. SOM1D will be used in conjunction with wavelet processing, and consists of a variable size one dimensional input and output. SOM2D allows input data to be presented as rectangular array for each sample, while the output remains a linear array. This rectangular array consists of a 2-d spectrogram with one dimension being the spectrum, and the second, the time sequence. Both SOM1D and SOM2D have versions which can be given a Matlab cell array of file names and the program prepares the input file as well as a functional form which assumes another external function has created the input training and testing files.

**Preprocessing strategies**

Extensive work has been done in the processing of acoustic signals. This includes such areas as room acoustics, SONAR, speech processing, and audio recording. In acoustic recognition, time domain signals (Lin, 1996), 2 dimensional Fourier Transforms (Lin, 1998), and Wavelets (Wood, 2002) have all been evaluated.

**Spectrogram**

The bulk of the work here was performed using Fourier Transform processing of the incoming sound signal. The actual processing algorithm used in Matlab® is a variation of the Cooley-Tukey decomposition method. Matlab® allows the input signal to be of length other than $2^n$ (n is an integer) if required. This allows variable time length samples...
to serve as inputs. No additional tapering or phase adjustments are applied other than what is embedded in the Matlab® algorithm. A single spectrum sequence capture program was written for this work, called “SPGRAM” in appendix B. This program allows the user to enter the parameters for the 2d tempo-spectrogram and the program returns a 2 dimensional array of FFT’s for different points in time. In this module, the individual sample width, the overlap between samples, the starting offset for the series, the number of spectra to be made, and the number of points in the FFT are parameters (see Exhibit 5.6). Internally, error checking makes sure the file doesn’t overrun the end of file (EOF) marker without letting the user know. Also, if the sample window has too few points for the desired FFT, the system aborts. If the file is too short, the output file is filled with blanks to prevent subsequent divide by zero errors.

Exhibit 5.6. Input Definitions for Module SPGRAM
Scaling

As mentioned previously, the input data must be scaled to match the response span of the neurons. Since the SOM, BPN, and Hybridnet programs are designed to be multi-purpose, the scaling of the data is provided internally to the program. Several values of scaling parameters were used for this work, the inputs and outputs are all scaled to the 0.1-0.9 binary range. More detailed examination of the scaling and offset effects are presented with the project 2 data set, in section 6.5.3.

The number of points for the FFT analysis is varied as well in this work. For the FFT work, several values of FFT are examined in both SOM and Hybridnet systems. To examine the relative behavior of wavelet processing versus FFT, an initial experiment of wavelet processed inputs for the Hybridnet and SOM is also performed. In this manner, the relative sensitivity of network confidence intervals to network structure versus input processing is evaluated.

Dataset 1

The initial thesis of this project was to test if a single neural network could perform adequately on an ensemble dataset of various types. To this end, a data set was created using the available sound resources. The sounds are listed in Exhibit 5.7.
<table>
<thead>
<tr>
<th>Sound</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tarpon</td>
<td>University of Rhode Island (URI)</td>
</tr>
<tr>
<td>Ship</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Ship Close to Phone</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Japanese Diesel Sub</td>
<td>Sub Website</td>
</tr>
<tr>
<td>Earthquake</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Tremor</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Atlantic Herring</td>
<td>URI</td>
</tr>
<tr>
<td>Humpback Trumpet</td>
<td>Acoustical Society of America (ASA)</td>
</tr>
<tr>
<td>Humpback Whistle</td>
<td>ASA</td>
</tr>
<tr>
<td>Blue Whale 26south</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Sperm whale</td>
<td>Cetacean Research</td>
</tr>
<tr>
<td>Cownose Ray</td>
<td>URI</td>
</tr>
<tr>
<td>Orca</td>
<td>Cetacean Research</td>
</tr>
<tr>
<td>Blue Whale 24 south</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Humpback Cry</td>
<td>ASA</td>
</tr>
<tr>
<td>Boat Indian River Lagoon</td>
<td>Florida Tech</td>
</tr>
<tr>
<td>Tremor</td>
<td>NOAA/PMEL</td>
</tr>
<tr>
<td>Dolphin</td>
<td>Arrtec</td>
</tr>
</tbody>
</table>

**Exhibit 5.7. Ensemble Sound Data Set 1**

**Dataset 2**

Since the focus of this research is on the application of the neural nets, the problem categories were identified from the first set. A new data set was created just to evaluate the limits of the neural networks and perform some examinations of strategies to improve performance on these items. In the case of this research, fish seem to be the most problematic. Since the only available sound files for fish were from the URI collection, this source was used exclusively for the research (See Exhibit 5.8).
<table>
<thead>
<tr>
<th>Fish</th>
<th>URI File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cownose Ray (Rhinoptera bonasus)</td>
<td>Disk 1, Track 02</td>
</tr>
<tr>
<td>Tarpon (Megalops atlantica Valanciennes)</td>
<td>Disk 1, Track 03</td>
</tr>
<tr>
<td>Atlantic Herring (Clupea harengus harengus)</td>
<td>Disk 1, Track 05</td>
</tr>
<tr>
<td>Black Drum (Pogonias cromis)</td>
<td>Disk 2, Track 12</td>
</tr>
<tr>
<td>Bluesetriped Grunt (Haemulon sciurus)</td>
<td>Disk 1, Track 69</td>
</tr>
<tr>
<td>Sea Catfish (Galeichthys felis)</td>
<td>Disk 1, Track 09</td>
</tr>
<tr>
<td>Silver Perch (Bairdiella chrysura)</td>
<td>Disk 2, Track 06</td>
</tr>
</tbody>
</table>

**Exhibit 5.8.** Fish Source Files for Data Set 2

This data set required some manual processing to make it suitable for this work. Since Matlab can read and write .wav and .au files directly, this is the desired format with which to interface to the program. The sound files from URI are presented in CD audio form. First, these files must be converted from CD audio track to .wav or .au file. Then, each file has embedded within it the narrator speech identifying the sound track the time and place of recording as well as technical information. This must be stripped out before the file can be used. In addition, the files were originally recorded at various low frequency bandwidths using analog tape in the 1950’s and 1960’s. These were then resampled to 44kbps using digital recording. Before these can be used for the neural networks, the file must be resampled again to the desired sample rate, in this case 8000 bps. All of these utility processes can be performed in Matlab, or using another program such as Awave Studio by FMJ Software.
Self Organizing Map Results

Data set 1 Results: Sample Frame Experiment

The recognition of the neural networks is dependent upon the amount of time allotted to the sample relative to the structure of sound in the sample (Kamm, 1990). Therefore, the first experiment to be performed is the evaluation of recognition rate as a function of sample frame size. All of the sound files in data set 1 were loaded into a Matlab workspace. For this experiment, the SOM2d program was used which internally formats the data.

The network parameters (learning rate, neighborhood function, weight initialization) were held constant. The minimum frame width is limited in this case by the FFT resolution desired.

For (at 8000 sample/sec rate):

- 256 point FFT: .032 sec min
- 512 point: .064 sec
- 1024 point: .128 sec
- 2048 point: .256 sec
- 4096 point: .512 sec

The experimental values used were time frame 0.07 < t < 0.5 sec, FFT=512, 0.25 frame offset, 0.1 sec train offset, 3 sec test offset, 10 spectra, 10000 iteration, neighborhood 2, 100 iteration neighborhood cutoff, learning rate, 0.99.

The results are shown in Exhibit 5.8. Note that because of the short duration of most of the sound files, only three trials for training and three independent trials for testing are given.
It should be noted that in using the SOM paradigm, the specific number for a mapping changes with each run. For example, the ship was correctly isolated in all frames. But in each run, it was mapped to a different specific number 1, 7, or 4.

The second observation is that the sample window makes a dramatic difference on the categorization of the files. Man made objects were simple for the network to recognize. Except for the submarine and ship at 0.5 sec frame, the ship, boat and sub map to separate categories.

Next as frame rate is widened, the ability to even partially categorize fish goes away. By 0.5 sec, all three species of fish map to a single category. It is interesting to note that at the shorter intervals, the cow nose ray is separated from the other two species of fish.

For the earthquake and the tremor, there is good separation from the other categories, but the two map to distinctly different categories. Therefore the system would recognize a tremor as a distinct event from the full blown earth quake.

<table>
<thead>
<tr>
<th>Frame Rate (Sec)</th>
<th>.07</th>
<th>.1</th>
<th>.3</th>
<th>.5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sound</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tarpon</td>
<td>14,14,14</td>
<td>3,3,3</td>
<td>1,1,1</td>
<td>1,1,1</td>
</tr>
<tr>
<td>Ship</td>
<td>1,1,1</td>
<td>7,7,7</td>
<td>4,4,4</td>
<td>4,4,4</td>
</tr>
<tr>
<td>Loud Ship</td>
<td>1,1,1</td>
<td>7,7,7</td>
<td>4,4,4</td>
<td>4,4,4</td>
</tr>
<tr>
<td>Sub</td>
<td>1,1,5</td>
<td>10,10,10</td>
<td>10,10,10</td>
<td>4,4,4</td>
</tr>
<tr>
<td>Quake</td>
<td>7,7,7</td>
<td>9,9,9</td>
<td>3,3,3</td>
<td>11,11,11</td>
</tr>
<tr>
<td>Humpback</td>
<td>6,6,13</td>
<td>8,8,8</td>
<td>2,14,2</td>
<td>2,2,4</td>
</tr>
<tr>
<td></td>
<td>Cry</td>
<td>Atlantic Herring</td>
<td>Ship</td>
<td>Humpback Trumpet</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>------------------</td>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>14,14,14</td>
<td>3,3,3</td>
<td>1,1,1</td>
<td>1,1,1</td>
</tr>
<tr>
<td></td>
<td>1,1,1</td>
<td>7,7,7</td>
<td>4,4,4</td>
<td>4,4,4</td>
</tr>
<tr>
<td></td>
<td>2,2,2</td>
<td>11,11,11</td>
<td>7,7,7</td>
<td>15,15,7</td>
</tr>
<tr>
<td></td>
<td>2,2,2</td>
<td>11,11,11</td>
<td>7,2,6</td>
<td>15,12,15</td>
</tr>
<tr>
<td></td>
<td>2,6,2</td>
<td>11,11,11</td>
<td>7,7,7</td>
<td>12,12,15</td>
</tr>
<tr>
<td></td>
<td>6,12,6</td>
<td>13,1,1</td>
<td>2,2,2</td>
<td>2,2,7</td>
</tr>
<tr>
<td></td>
<td>13,13,4</td>
<td>12,15,6</td>
<td>12,1,12</td>
<td>1,1,2</td>
</tr>
<tr>
<td></td>
<td>14,8,10</td>
<td>3,6,12</td>
<td>12,1,12</td>
<td>1,8,1</td>
</tr>
<tr>
<td></td>
<td>6,6,6</td>
<td>2,2,2</td>
<td>7,7,7</td>
<td>12,12,12</td>
</tr>
<tr>
<td></td>
<td>4,3,3</td>
<td>1,11,7</td>
<td>5,7,7</td>
<td>15,15,9</td>
</tr>
<tr>
<td></td>
<td>8,3,4</td>
<td>4,5,8</td>
<td>12,12,15</td>
<td>8,12,7</td>
</tr>
<tr>
<td></td>
<td>14,14,2</td>
<td>3,3,11</td>
<td>7,7,1</td>
<td>1,15,15</td>
</tr>
<tr>
<td></td>
<td>5,7,5</td>
<td>10,10,9</td>
<td>10,13,13</td>
<td>4,7,12</td>
</tr>
</tbody>
</table>

**Exhibit 5.9.** Time Frame Experiment Results

The cetaceans demonstrate even more interesting variability. For example, the dolphin at 0.07 sec maps to other categories of interest, for example, the diesel submarine. At 0.1 and 0.3, the partial mapping to the submarine continues, but by 0.3, the dolphin mapping has evolved into a unique category, 13. By 0.5 sec, the averaging of the large time interval merged the dolphin with the boat and the other whale categories.

For the whales, there is tremendous category splitting, based on the complexity of the verbalization of the animal. For example, the humpback whale “cry” vocalization maps to various degrees to unique categories from the other vocalization. The blue whale samples are from two separate individuals, yet combine very well. However there is the tendency for the blue whale to merge into other categories, particularly at shorter
intervals. In other test runs, it has also demonstrated the tendency to group with earthquake and tremor events.

The orca whale sounds have a unique set of problems due to the variety of vocalizations. For the test data used here, the category splits three ways which is common to other orca vocalizations as well. The sounds map distinctly to boat and other fish categories, depending on the behavior of the orca at the time of recording.

The humpback has one of the richest vocabularies observed. From the standpoint of this classification it has no less than 4 separate categories for vocalization, unique to the sampling and type of call. In addition, some portions of the vocalizations map to fish and boats of different types.

Lastly, the network response to the sperm whale vocalizations suffers from a similar variability. In addition to unique categories, it often maps to humpback sound types.

Data set 1 Results: FFT Variation Experiment

Having established an adequate range of sample frame size for operation, the next step is to examine the network performance as a function of FFT spectral resolution. Before performing the FFT experiment, one ambiguity needed to be resolved. Was the structure seen in the frame experiments uniquely due to the time only? The question posed some additional experimentation which was performed as a preliminary experiment to this work. In the structure of the input data, there were 3 variables that could be modified. Variable 1 involves the time width of the frames already discussed. Variable 2 is the number of spectra included in each 2 dimensional spectrogram. It was observed that dramatic improvements in recognition came from 1 to 10 spectra, but after
that there was no significant improvement. So throughout this work, the number of spectra was held constant at 10.

Variable 3 is the FFT density. This essentially gives the frequency resolution of the sample. For this experiment, all other parameters will be held fixed, and the response of the network will be examined versus FFT resolution.

Three values of FFT were chosen, 128, 512, and 2048. The same dataset was applied to the network and this time, the frame for measurement was fixed at .3 sec. The results are in Exhibit 5.10. Frame =0.3 sec, 0.25 frame offset, 0.1 sec train offset, 3 sec test offset, 10 spectra, 10000 iteration, neighborhood 2, 100 iteration neighborhood cutoff.

A similar pattern emerges between major groupings of sounds (See Exhibit 5.10). The man made sounds generally classify well, although with a 128 point FFT the submarine and the ship are indistinguishable, corresponding to a frequency separation of less than 31.25 Hz to recognize the difference.

The quake and the tremor remain distinguishable, but with a 128 point FFT, the tremor and blue whale and fish overlap to category 13. The Herring and the Tarpon remain indistinguishable through all the data points, and the Ray also at 128 point, but begins to be separable at 512 points. There is a diminishing return for Cetaceans with increasing resolution. With 2048 point FFTs, the whales all merge into a single category with the exception of the orca. The orca continues to demonstrate a mixture of fish, vehicle, and its own unique category of sounds.

<table>
<thead>
<tr>
<th>Sound</th>
<th>FFT 128pt</th>
<th>512</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13,13,13</td>
<td>3,3,3</td>
<td>12,12,12</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>Tarpon</td>
<td>9,9,9</td>
<td>2,2,2</td>
<td>5,5,5</td>
</tr>
<tr>
<td>Ship</td>
<td>9,9,9</td>
<td>2,2,2</td>
<td>5,5,5</td>
</tr>
<tr>
<td>Loud Ship</td>
<td>9,9,9</td>
<td>6,6,6</td>
<td>7,7,7</td>
</tr>
<tr>
<td>Sub</td>
<td>8,8,8</td>
<td>8,8,8</td>
<td>4,4,4</td>
</tr>
<tr>
<td>Humpback</td>
<td>2,2,2</td>
<td>11,10,11</td>
<td>8,8,14</td>
</tr>
<tr>
<td>Atlantic</td>
<td>13,13,13</td>
<td>3,3,3</td>
<td>12,12,12</td>
</tr>
<tr>
<td>Herring</td>
<td>9,9,9</td>
<td>2,2,2</td>
<td>5,5,5</td>
</tr>
<tr>
<td>Humpback Trumpet</td>
<td>13,13,2</td>
<td>7,7,7</td>
<td>14,14,14</td>
</tr>
<tr>
<td>Humpback Whistle</td>
<td>3,2,2</td>
<td>7,11,15</td>
<td>14,14,14</td>
</tr>
<tr>
<td>Blue Whale</td>
<td>13,13,5</td>
<td>7,7,7</td>
<td>14,14,14</td>
</tr>
<tr>
<td>26S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sperm Whale</td>
<td>2,2,2</td>
<td>11,11,11</td>
<td>14,14,8</td>
</tr>
<tr>
<td>Ray</td>
<td>13,13,13</td>
<td>1,3,9</td>
<td>14,4,4</td>
</tr>
<tr>
<td>Orca</td>
<td>10,13,5</td>
<td>1,3,15</td>
<td>1,12,13</td>
</tr>
<tr>
<td>Blue Whale</td>
<td>5,13,13</td>
<td>7,7,7</td>
<td>14,14,14</td>
</tr>
<tr>
<td>24S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Humpback HauntCry</td>
<td>4,13,13</td>
<td>14,7,7</td>
<td>14,14,14</td>
</tr>
<tr>
<td>Boat_IRL</td>
<td>7,15,4</td>
<td>1,12,12</td>
<td>15,15,15</td>
</tr>
<tr>
<td>Tremor</td>
<td>13,13,13</td>
<td>7,7,3</td>
<td>12,14,12</td>
</tr>
<tr>
<td>Dolphin</td>
<td>8,9,9</td>
<td>6,9,12</td>
<td>7,13,7</td>
</tr>
</tbody>
</table>

**Exhibit 5.10. FFT Resolution Experiment Results-SOM**

The results of the 2048 point FFT are confusing. Earlier work by the author on a more limited data set indicated that the higher the resolution of the FFT, the more improvement (Howell, 2003). However, that data was for 0.5 frame width samples. As was seen in the previous section, the reduction of sample width to 0.3 in and of itself seems to provide improvement in recognition rates.

There are three significant conclusions that arise from these experiments. First, for man-made and geologic sounds, recognition is relatively straightforward for several
sampling intervals and spectral resolutions. However, there seems to be a lower bound somewhere above a 128 point FFT.

Secondly, cetaceans consistently map to some unique categories, but also contain structures which alternatively may map to boats, fish, or earthquakes. The relative amount and variety depends upon species.

Lastly, fish, because of their episodic nature, are the most difficult to classify. After a review of the fish data, it was found that during most sampling intervals, the signal energy due to the fish in each pulse is less than 1% of the total energy in the signal interval. Therefore, it is hypothesized that by looking exclusively at the pulse itself, recognition rates will improve. This requires a separate approach from the ensemble network evaluated here.

For the ensemble map, within the experimental matrix performed here, the optimum recognition preprocessing is when sample frames are between 0.1 and 0.3 sec if sampled at 8khz, and FFT values of 512 point (7.8 hz bins) seem to provide the best tradeoff between recognition and complexity. For the cetaceans, some means of combining these various categories into one need to be devised. And for the fish, an alternate means of processing is required. This becomes the focus of the next set of experiments, discussed in section 5.9, using the MLP paradigm.

**Multi-level Perceptron Map**

Results of the early research indicated that for cetacean sound sources, several classes might be generated from one source using the SOM. In order to re-map those categories to a single output type (i.e. humpback whale, earthquake, catfish) a static supervised network was envisioned. The MLP network was chosen for this limited
function where the inputs and outputs can be known and scaled thus improving convergence. Rather than feeding the raw data into the network, it was proposed to first test if the MLP could take static data as was generated from the previous experiments and map it to correct outputs. The reduced problem uses the MLP design shown in Exhibit 3.6. To test this concept, several runs of the SOM neural network were performed and the raw output was captured (See Exhibit 5.11).

<table>
<thead>
<tr>
<th>Name</th>
<th>Som Output</th>
<th>SOM</th>
<th>Desired</th>
<th>Mapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tarpon</td>
<td>0.07 0.27 0.90 0.00 -02 0.19 0.01 0.26 0.40 0.62</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ship</td>
<td>0.13 0.35 0.02 0.90 -01 0.14 0.29 0.47 02 0.25</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Close Ship</td>
<td>0.10 0.46 0.02 0.90 0.00 0.08 0.33 0.59 0.01 0.22</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Sub</td>
<td>0.03 0.32 0.00 0.29 0.00 0.03 0.90 0.47 02 0.05</td>
<td>7</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Earthquake</td>
<td>0.00 0.07 0.06 0.13 -01 0.11 0.03 0.44 04 0.15</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Humpback Cry</td>
<td>0.05 0.90 0.10 0.26 -01 0.04 0.22 0.24 0.27 0.13</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Tarpon</td>
<td>0.12 0.26 0.90 0.04 0.00 0.13 0.01 0.35 0.25 0.55</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ship</td>
<td>0.13 0.35 0.02 0.90 -01 0.14 0.29 0.47 02 0.25</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Humpback Trumpet</td>
<td>0.14 0.07 0.09 0.17 -01 0.90 0.03 0.36 0.10 0.25</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Humpback Whistle</td>
<td>0.04 0.46 0.18 0.02 -01 0.10 0.02 0.23 0.90 0.17</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Bluewhale 26s</td>
<td>0.04 0.04 0.05 0.05 0.00 0.07 0.01 0.57 0.02 0.09</td>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Sperm Whale</td>
<td>0.22 0.16 0.09 0.22 0.00 0.14 0.22 0.90 0.09 0.18</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Cownose Ray</td>
<td>0.17 0.22 0.36 0.29 0.00 0.23 0.06 0.43 0.16 0.90</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

**Exhibit 5.11.** MLP Re-mapping of SOM Output

Notice that for each row, there is one maximum output. This designates the category for that sample. For this map, the humpback whale vocalizations mapped to three different categories. This was used as the input to an MLP network. A desired set of mappings was applied as the training targets. For this run, only two fish were included, each with different mappings so as to focus exclusively on the whale vocalizations. At 20000 iterations, the system created the mapped categories as shown. In comparing the desired targets maps to the actual mappings, 100% accuracy is observed. In principal, the MLP can be used to address one of the shortfalls of the SOM map seen earlier. For sources with rich multi-category vocalizations or sources which sometimes blend with
other sources, the MLP network can combine the different vocalizations into one category such as “humpback whale” or “earthquake”.

Hybridnet Design

Cetacean sounds created unique challenges for the simple, unsupervised self-organizing map (SOM) network as demonstrated in the previous experiments. After working with the MLP structure, it was determined that re-mapping of the SOM outputs would effectively recombine those categories split by the SOM without reducing the resolution of the network to other categories.

Hybrid neural networks have been utilized for a variety of applications (Masters, 1993, Ghosh, 1992). They are most often used when the problem is too big for a single network to handle, resulting in long training times, large resource consumption, or poor performance. In this case though, the goal is one of filtering and synthesis of categories already resolved into component parts.

The structure of Hybridnet 4 was discussed earlier, consisting simply of a cascade of an unsupervised network (SOM) which serves as a pattern classifier and a supervised stage (MLP) that serves as an output mapping synthesis stage. The version used here has certain specifics. First, the training method is sequential and batch in nature. For the best performance for this network, the SOM stage is trained to completion before training the MLP to the patterns. Secondly, the output of the SOM stage is not supervised in any way, (i.e., it is allowed to discover the natural structure present in the signals applied). This is distinctly different than the Learning Vector Quantization (LVQ) network that was applied to fish samples where training categories were applied at each cycle (Lin, 1998).
The number of training loops for both the SOM stage and the MLP stage are set independently. In order to predict the time for training, the number of training iterations were set to a fixed number rather than using an error threshold criteria for training. In addition, separate learning rates are applied to each stage.

After the SOM has been trained, the training set is applied to the input of the SOM with no weight changes on the SOM side. The outputs are fed onto the MLP with only neuron scaling to insure operation in the range of the neurons. MLP weights are initialized by the Nguyen and Widrow method, and the MLP network is trained as discussed earlier. After termination has occurred, the network weights are frozen in the MLP stage and the system is tested.

One desirable feature of this partitioning is that in future work, several candidate paradigms can be inserted into the unsupervised and supervised blocks of the system.

**Hybridnet Results**

As a test of the network performance, the data set “1” from the SOM work earlier was applied to Hybridnet. Since Hybridnet uses supervised learning for the output stage, correct training categories for data set “1” were added in order to execute the training.

<table>
<thead>
<tr>
<th>Sound</th>
<th>FFT 128pt</th>
<th>512</th>
<th>2048</th>
<th>Desired Cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tarpon</td>
<td>1,1,1</td>
<td>1,10,1</td>
<td>1,1,1</td>
<td>1</td>
</tr>
<tr>
<td>Ship</td>
<td>2,2,2</td>
<td>2,2,2</td>
<td>2,2,2</td>
<td>2</td>
</tr>
<tr>
<td>Loud Ship</td>
<td>2,2,2</td>
<td>2,2,2</td>
<td>2,2,2</td>
<td>2</td>
</tr>
<tr>
<td>Sub</td>
<td>3,3,3</td>
<td>3,3,3</td>
<td>3,3,3</td>
<td>3</td>
</tr>
<tr>
<td>Quake</td>
<td>4,4,4</td>
<td>4,4,4</td>
<td>4,4,4</td>
<td>4</td>
</tr>
<tr>
<td>Humpback Cry</td>
<td>5,5,5</td>
<td>5,5,5</td>
<td>8,5,5</td>
<td>5</td>
</tr>
<tr>
<td>Atlantic Herring</td>
<td>1,1,1</td>
<td>1,1,1</td>
<td>6,6,1</td>
<td>6</td>
</tr>
<tr>
<td>Ship</td>
<td>2,2,2</td>
<td>2,2,2</td>
<td>2,2,2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>----</td>
</tr>
<tr>
<td>Humpback Trumpet</td>
<td>4,4,4</td>
<td>5,5,5</td>
<td>5,5,5</td>
<td>5</td>
</tr>
<tr>
<td>Humpback Whistle</td>
<td>4,5,5</td>
<td>5,5,5</td>
<td>5,4,5</td>
<td>5</td>
</tr>
<tr>
<td>Blue Whale 26S</td>
<td>4,4,4</td>
<td>7,7,7</td>
<td>7,7,7</td>
<td>7</td>
</tr>
<tr>
<td>Sperm Whale</td>
<td>8,8,3</td>
<td>8,8,8</td>
<td>8,8,8</td>
<td>8</td>
</tr>
<tr>
<td>Ray</td>
<td>10,10,10</td>
<td>10,9,9</td>
<td>12,9,9</td>
<td>9</td>
</tr>
<tr>
<td>Orca</td>
<td>10,3,3</td>
<td>10,10,4</td>
<td>10,10,10</td>
<td>10</td>
</tr>
<tr>
<td>Blue Whale 24S</td>
<td>4,4,4</td>
<td>7,7,7</td>
<td>7,7,7</td>
<td>7</td>
</tr>
<tr>
<td>Humpback HauntCry</td>
<td>5,5,5</td>
<td>5,5,5</td>
<td>5,5,5</td>
<td>5</td>
</tr>
<tr>
<td>Boat_irl</td>
<td>11,11,11</td>
<td>11,11,11</td>
<td>11,11,11</td>
<td>11</td>
</tr>
<tr>
<td>Tremor</td>
<td>4,4,4</td>
<td>4,5,4</td>
<td>4,5,4</td>
<td>4</td>
</tr>
<tr>
<td>Dolphin</td>
<td>12,12,12</td>
<td>10,8,12</td>
<td>12,8,12</td>
<td>12</td>
</tr>
</tbody>
</table>

**Exhibit 5.12. FFT Resolution Experiment Results-Hybridnet**

For example, the vocalization of a humpback whale may alternatively sound like the grunt of a fish, the whine of a boat, or the rumble of an earthquake but in all cases it is desired for this project to map the vocalizations to “humpback whale”. The results along with the mapping category data is in exhibit 5.12.

Several aspects of the results are noteworthy. First, the problems with cetacean mapping have indeed been resolved. As can be seen here, by 512 points, the humpback, blue, and sperm whale all have 100% mapping over the testing interval. At 2048 points, there is one test for the humpback whistle and humpback cry that maps to another known category, but in general, Hybridnet corrected the problem it was designed for. Note, the diesel sub now consistently maps to its own category. The dolphin and orca sounds still map to alternate categories periodically but by 2048 point FFT, the orca has achieved 100% correct mapping over the testing set. And finally, by 2048 points the fish species have begun to be distinguishable. There are still errors (1 out of 3) but for the first time,
all three categories do have some correct mappings. This tends to corroborate that the spectral signal in the fish are of such small amplitude, and/or are so finely spaced, that special examination must be applied to the fish sounds, separately from the other species. This then directs the experimentation of section 5.12.

**Fish Sounds**

Of the three fish sounds in data set one, two consistently mapped to the same category, the herring and the tarpon. Since the energy of the fish sound pulses is a small part of the energy in the sample window, the averaging caused by the FFT will cause the distinctives of these categories to disappear until the high number FFT has been applied. One approach for the fish is to perform the FFT on pulses only, evaluating each pulse as a single sound source with the FFT applied over the entire pulse. In earlier work (Lin, 1998, Wood, 2002), the individual pulses have been selected and combined to form a series of high energy samples. Repeated pulses are used then for training and testing.

Both FFT and wavelets analysis were applied to individual pulses.

As a benchmark for subsequent work, the second data set was applied to Hybridnet with a 512pt FFT processing of DS2 (see exhibit 5.13).

As can be seen, the data is inconclusive. In order to verify that these results are a problem of the data and not the map, the maximum neuron output for each category is included. Notice that only the cow nose ray sound has confidence numbers high enough to be believed. Even the herring, though giving the correct mapped category, has outputs below .5, and therefore the network is essentially indeterminate.
Exhibit 5.13. Fish Mapping, 512 pt FFT Raw Data, Hybridnet 4

<table>
<thead>
<tr>
<th>Category</th>
<th>Mapping</th>
<th>Maximum Network Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cownose Ray</td>
<td>1,1,1</td>
<td>.6271</td>
</tr>
<tr>
<td>Tarpon</td>
<td>3,3,1</td>
<td>.4940</td>
</tr>
<tr>
<td>Herring</td>
<td>3,3,3</td>
<td>.4365</td>
</tr>
<tr>
<td>Black Drum</td>
<td>3,1,3</td>
<td>.4389</td>
</tr>
<tr>
<td>Bluestriped Grunt</td>
<td>7,1,5</td>
<td>.4879</td>
</tr>
<tr>
<td>Catfish</td>
<td>6,3,3</td>
<td>.4467</td>
</tr>
<tr>
<td>Silver Perch</td>
<td>4,1,3</td>
<td>.5059</td>
</tr>
</tbody>
</table>

The reasoning behind this is that there are insufficient structural differences in the inputs to be discerned by the neural network. There are two possible solutions. If patterns can be seen by a human observer, then work must be done to modify the network paradigm in order to make the network better able to resolve the subtleties in the structure. Alternatively, the input presentation should be refined to enhance the structural differences between input samples. It was decided to look at refining the preprocessing of the inputs, such as pulse gathering.

Pulse Gathering

The first step to exclusively examine fish was to gather the individual beats from the sound streams. This increases the energy content of the fish vocalizations in the signals. At first, hand editing was performed, but this allowed subjectivity into the process of when a pulse begins and ends. To eliminate the subjectivity of pulse selection, the process was automated by writing the Matlab program “FISHWHACK_FFT.M” and “ENVELOPE_THRESH.M”.

Several possible methods of preprocessing data was discussed in Section 4. Two tasks must be performed to automate the pulse compression task. First, the point at which
a pulse starts and stops must be determined. Secondly, the pulses must be gathered and placed into a compressed form.

“ENVELOPE_THRESH.M” has been discussed in the preprocessing section. It performs the first task by monitoring the total energy in the acoustic signal and creating a “mask file” for “pulse” and “no pulse.” “FISHWHACK_FFT.M” uses the mask from “ENVELOPE_THRESH.M” which indicates the pulse locations and defines the start and stop indices for the pulses. The longest pulse determines the size of the output array since Matlab requires the array to be rectangular. For pulses that are shorter than the maximum, the remaining training time interval is padded with zeros. The start times are all kept at the same point as the wavelet analyses are sensitive to time position and gives a fixed reference point. At that point, the pulses are removed and stacked together into a continuous array, one dimension being the pulse signal, the other being the number of the pulse. Finally the program performs a standard FFT and computes the power spectral density function for the signal, and lastly computes a wavelet transform. These values may then be used as inputs to any neural network. For this version of the software, the wavelet used was Coiflet level 3 with the discrete wavelet transform, level 3. The typical appearance of the time pulse, the FFT, and the wavelet appear in Exhibit 5.14.

The number and sizes of the pulses depend upon two factors, threshold and averaging value, settable by the user. The threshold determines the relative energy “value” at which the system assumes the pulse starts or stops. The averaging value is the number of points to include in a boxcar average to apply as a low pass filter smoothing behavior to the energy curve, which is passed down to “ENVELOPE_THRESH.M” to
determine the exact pulse locations. Stronger pulses have a longer duration as they have a longer time above the energy threshold.

Exhibit 5.14. Comparative Time, Frequency and Wavelet for Cow Nose Ray

FFT Results

A Total of 202 pulses were captured from the species in data set 2. These pulses were divided to include all species in both training and testing. To explore the data space, the data was applied to both the “SOM1D” program and the “BPN_3.5” multi-level perceptron program used for mappings (see exhibit 5.15). This was to determine if the data reduction due to Hybridnet was responsible for the poor recognition. As an added attempt at increase, the “BPN_3.5” program used a 512 point FFT input for increased resolution.
As can be seen, the results are not acceptable. The percentages listed for SOM categorization rate are merely the percentage times, that category of data gave the same answer. It should be noted that there are only three distinct categories mapped by the SOM. The “natural” categories appear to be Ray-Grunt-Perch-Drum, Tarpon-Herring, and Catfish.

When an attempt is made to force categorization through a supervised method, even with additional resolution, the results are inadequate. Only four of the seven categories mapped at all to the target categories and all well below 50% correct recognition. And in this case, the network mapped to an unknown case “10”. Given that the training error did map to low values, it appears that with the FFT, there is insufficient structure to resolve multiple categories of fish at once.

**Wavelet Results**

Before examining the wavelet results, the inputs to the system are compared (see exhibit 5.16).
The cow nose ray wavelet coefficients are observed in exhibit 5.14. Because of the time dependency of the wavelets and the fact that wavelets are also dependent on the sample interval, it was necessary also to use the program “FISH_WHACKFFT” to gather pulses and put them in a fixed reference frame. Also, the sample length was truncated or extended so that all wavelet transforms came out to 560 samples. In the Exhibit 5.16, the last 60 points are not plotted since they were all zero in magnitude. First, notice that the wavelet transform is extremely sparse. To the left are the Cd1 coefficients from the first level mapping, followed by Cd2 and Cd3, lastly followed by Ca3. By the time of the third level of decomposition, there is little energy left in the function. For example, the catfish and tarpon have unique signatures at the first level, but the other features at the higher levels are sparse, and could contribute to recognition. Rather than select only certain level coefficients from the wavelet decomposition, it was chosen for this experiment to map the entire transform to the inputs of the network.

In the same manner as in the previous FFT experiments described in section 5.8, the wavelet data was applied to the SOM and BPN networks. The results are in exhibit 5.17.

The results are again disappointing. The network did in fact converge, but mapped each input to multiple categories in the repeated trials. The percentage listed is the percentage of samples that were mapped to the maximum category. The network appears to be implying that there are 4 structure types in the sounds, and that every fish contains some of each depending upon the specific sample.
<table>
<thead>
<tr>
<th>Source</th>
<th>SOM Categories</th>
<th>SOM % Coif3 L3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray</td>
<td>7,4(8)</td>
<td>40%</td>
</tr>
<tr>
<td>Tarpon</td>
<td>4(7,8,3)</td>
<td>31.8%</td>
</tr>
<tr>
<td>Herring</td>
<td>4(7,8)</td>
<td>66%</td>
</tr>
<tr>
<td>Black Drum</td>
<td>8(4,3)</td>
<td>70%</td>
</tr>
<tr>
<td>Grunt</td>
<td>4(8,7,3)</td>
<td>40%</td>
</tr>
<tr>
<td>Catfish</td>
<td>4(7,8)</td>
<td>45%</td>
</tr>
<tr>
<td>Silver Perch</td>
<td>7(4,3,8)</td>
<td>72%</td>
</tr>
</tbody>
</table>

Exhibit 5.17. Data Set 2 Pulse Responses, Wavelet Coiflet3, Level 3 Processing

Two Category Network Results

One possible way to improve on the results of Section 5.12 is to reduce the problem from an ensemble problem to a two category problem. For this the same data sets were reapplied to the SOM network, but this time the network was constrained to map to one of two categories Both the FFT and the wavelet versions were applied. For the FFT input, the SOM network recognized no structural differences between pulses at 512 point FFT images. For the wavelet, some structure was discovered, but did not follow any particular fish category.

Discussion

In general, the results are very encouraging. Examining the four major classes of sounds, one observes significant differences in results however. For the class of man made sounds, both neural networks paradigms with FFT inputs performed extremely well with adequate frequency resolution, greater than 512 points at 8kbps sampling rate. The geological sounds were easy to map as well, but tended to represent different categories. Also, the complex verbalizations of the whales were often mapped to multiple categories. In both cases though, the mappings were unique. Thus the application of a second stage of recognition through the use of Hybridnet organized the classes into a fashion more easily used by an autonomous detection system. The fish remain very problematic. Non neural approaches of other researchers (Deuser, 1979) can only yield relative densities of “fish” with no selectivity of species. In earlier work by the author, mapping all fish to one category in a neural paradigm allowed the fish, as a group, to be separated from the other sound files (Howell, 2003). This would allow a separate network and/or preprocessing strategy to be applied. In this work, gathering pulses, as well as more detailed FFT and wavelet transformations were applied but with poor results. Based on the work of others with single and two species groups, wavelets appear to be the most promising candidate for future research in this area (Wood, 2002). That combined with multi-sensory approaches to detection hold the greatest promise for fish recognition.
Project 2: Geophysical Data Fusion

Environmental Features

As mentioned in the introduction, it would be of benefit for mission planners to be able to define missions based on time events or spatial/sensorial features of interest rather than simple navigation coordinates in a plow like fashion, leaving the scientist to reduce the data. This is the core of “environmentally based” navigation systems. In such a system, the neural processor would merge data from a variety of sensor sources, perform recognition of those patterns of interest, and then perform actions based on those higher level recognized features.

Project 2 is designed to answer these questions:

1) Can changes in data from simple sensors be classified into events or regions that can then be used to define more complicated events or regions?
2) Can a neural network be used to operate on these primitive behaviors even within a noisy environment? If so, what strategies must be applied in order to improve recognition accuracy?
3) Can data from multiple sensors and events from these sensors be fused by a neural processor to recognize complicated environmental features in space, time, or both?

If the answer to these questions is demonstrated to be yes, then an environmental navigation system and mission planner can be created for many of the applications outlined earlier.

There has been some research performed in these areas. In areas of control systems, the idea of merging mutual information to create “virtual” sensors has been explored (Deignan, 2000). A more thorough overview of multi-sensor fusion in the area of satellite imagery has also been written (Paola, 1993). In Paola’s work, as in available texts (Brooks, 1998), the model for data fusion is that of one master processing unit merging together data from several sources (See Exhibit 6.1.) Preprocessing may be performed on the data prior to input to the processor, but all decision making is performed within that single unit.
Exhibit 6.1. Ensemble Neural Classifier Block Diagram

As could be seen in the passive SONAR case previously, the use of one ensemble network requires very large networks, huge training/testing sets, and extremely fast hardware in order to implement a useable system.

Another approach that is developed in the marine context in this dissertation is the idea of primitive and emergent behaviors. This is based in part on strategies in ultrasonic navigation of factory environments (Zalzala, 1996) as well as the goal based hierarchy controller for robotic piloting (Zalzala, 1996). In both of these projects, a complicated task, such as avoiding an obstacle or navigating a corridor, is broken into simpler, easier to define detection or motion tasks, (e.g., keeping a certain distance from a wall). While the emphasis in available literature is on the primitive behaviors in control, the same principles are applied here to the problem of complex detection. The end result is an environmentally based recognition system (See Exhibit 6.2.)
Exhibit 6.2. Distributed Neural Classifier

Notice that in this system, each sensor system is responsible for performing low-level recognition of patterns in its respective sensory field. The summary categories are then reported to the feature processor to provide recognition of a complex environmental feature.

What types of sensor data is this approach applicable to? After reviewing the available payloads in both fixed point and vehicle platforms, a list of possible sensor sources to be examined was created; temperature, salinity, pressure, water clarity, fluorometry, optical spectroscopy, mass spectroscopy, inertial measurement, active acoustic, and passive acoustic instruments. Each source will now be introduced in detail.

**Temperature:**

One of the simplest and most useful sensory inputs is that of temperature. Temperature changes are present in everything from thermoclines to estuary water boundaries, to detection of thermal vents, and day/night cycles. Yet as can also be observed, temperature in and of itself is somewhat ambiguous as a sole measure of changes. For this paper, temperature is recorded over spatial or time dimensions and as will be seen, the ambiguity between spatial measurements over time and strict time measurements will degrade the performance of a classification system.
Salinity:

Salinity is recorded in the same manner as temperature. Examples of features where salinity is variable include rainstorms, estuary and inlet features, oceanic features, springs, and SOFAR channel.

Pressure:

Pressure is presented as a sequence of measurements and provides an orientation capability to other measurements. It is shown useful in distinguishing between vertical and horizontal approaches to certain features and to separate thermoclines from inlets.

Water Clarity:

From the Vailulu’u data (Hart, 2002), extensive nephelometry data is provided. However, for general use, any turbidity or nephelometry, or alternative measure of water clarity is useful. For example, in the Vailulu’u crater, the cloud layer in the crater provides an obvious cue that the feature is present. Clarity also provides additional corroboration for features as varied as inlets, to algal bloom, to rain.

Fluorometry:

Also in the Vailulu’u data is a channel for fluorometry for measuring trace dye in water movement. In general, Ultraviolet fluorescence spectroscopy is used in identification of low (ppb) quantities of impurities in the water. In the case of pollution, underwater vents, and potentially coral reef studies, this data channel would be a useful marker if tuned to a desired feature such as in the Vailulu’u data. For this experiment, the fluorometry data is used not just as a dye tracer, but also as a marker for individual vent activity. As presented to the environmental network, all that is presented as input is the change of the total response of the
channel. Depending on the mission, it might be the envelope of the fluorometry channel, or the changes due to one optical frequency of interest.

**Optical Absorption Spectroscopy:**

The workhorse for analytical water analysis, the UV-VIS absorption channel would be useful in detecting the algae concentration in the water and can also be used for water chemistry analyses. Here, the total energy change of the spectrum or the change in response to an individual absorption band may be used as input to the classifier. If multiple signals are required, then multiple inputs to the classifier may be implemented if more features are needed in the recognition, or higher rates may be achieved.

**Mass Spectroscopy:**

The mass spec has only recently been introduced to the water environment for in situ measurements (Short, 2001). As such it is being used for a variety of pollution studies. For this research, it is assumed that a mass number corresponding to a feature of interest is selected and that changes to this channel represent possible features to the classifier.

**Inertial Measurements:**

It is assumed that for any vehicle application, some form of inertial or acceleration measurement is available. There are several ways to present this data. The most advanced way would be to have individual channels for each degree of freedom as inputs to the classifier. This however may complicate the network. For this research, it was assumed that only the average magnitude of all the accelerometer inputs was used to get a measure of the degree of turbulence the vehicle was experiencing. It is also assumed that this type of turbulence would be observed in currents, vents, and estuary environments. Careful field work in the future may also determine if other events carry a signature in this sensory field and whether individual inputs are mandatory.
Active Acoustic:

The signal from the active acoustic channel which is used for input can be of two types. If a target is moving or the platform is descending to the bottom, the strength of the acoustic return should change. But in addition, the time of return could change as well. Experiments on recognition rate would need to be performed with both these methods of interaction with the environmental classifier to determine which provides the better recognition rates.

Passive Acoustic:

For purposes of this network, only the summary results from the network of project one are provided. And to this end, only changes in envelope amplitude of either a separated feature of interest, or the envelope of all hydrophone data is used to represent changes and thresholds in the passive acoustic environment. For example, in approaching an inlet at tide change, or during a rainstorm, it is assumed that the incoming acoustic signal will be increased by the amount of additional noise created. For all of project “2” (Chapter 6), we assume that the sound has not been recognized, but merely that the total energy in the SONAR signal is changing according to one of the seven primitive categories. For future work, the addition of a recognized sound signal can be used as well. However for the available data, concurrent acoustic data was not available with the geophysical data, so that enhancement was not attempted in this body of work.

Primitive Behaviors

The ways in which transitions in data occur define the classes of primitive behaviors to be used. In image work the boundary between regions is considered abrupt, and the bulk of the analysis involves getting uniform edge detection for different approach angles (Geiger, 2003). In the case of geophysical data, the boundaries are not abrupt, and for feature recognition, the approach angle is not important.

First, the pattern of each data field must be broken into its most basic parts. After reviewing sample data from the Hawaii Underwater Research Lab’s 2000 and 2001
Expeditions to Vailulu’u seamount (Hart, 2002), it became clear that changes in each data channel could be broken into seven basic categories (See Exhibit 6.3.).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1: Constant</td>
<td>No significant changes in mean value over sampling interval, slope=0.</td>
</tr>
<tr>
<td>Type 2: Increasing</td>
<td>Signal is increasing in amplitude over interval in continuous manner. May be increasing in logarithmic, quadratic, or linear manner.</td>
</tr>
<tr>
<td>Type 3: Decreasing</td>
<td>Signal is decreasing in amplitude over interval in continuous manner. May be increasing in logarithmic, quadratic, or linear manner.</td>
</tr>
<tr>
<td>Type 4: Entering</td>
<td>Distinct change in mid sample interval from 0 slope (type 1) to some positive (type 2) Or negative (type 3) slope, indicating the transition to a region of change.</td>
</tr>
<tr>
<td>Type 5: Leaving</td>
<td>Distinct change in mid sample from positive (type 2) or negative (type 3) slope to flat (type 1), 0 slope.</td>
</tr>
<tr>
<td>Type 6: Maximum</td>
<td>Distinct change from positive (type 2) to negative (type 3) slope with a maximum value near mid interval.</td>
</tr>
<tr>
<td>Type 7: Minimum</td>
<td>Distinct change from negative (type 3) to positive (type 2) slope with a maximum value near mid interval.</td>
</tr>
</tbody>
</table>

Emergent Behaviors

Emergent behaviors are those behaviors or recognitions which humans would recognize as discrete objects or events. Walking down the street, the event “Street corner” in
the pedestrian’s mind evokes a variety of patterns from all the several senses. The curb would provide a step change that would be felt as accelerations and pressure changes on the foot. The sight of moving cars, the traffic light, and the pedestrian “walk/don’t walk” signals would be the visual stimulus. The sound of cars and, in some places, a buzzer at the street corner provide somewhat unique audio stimuli. The smell of exhaust fumes or the taste of the cup of coffee the walker has each day at that curb are also valid inputs.

Notice that each individual sensor input is NOT unique to “street corner”. For each stimulus described, there are multiple events that have some of the same characteristics (See Exhibit 6.4). In this case, the visual cues are the predominate features identifying this as “street corner”, but depending upon what the walker is looking for, another cue may be as significant.

In other words, this event, “street corner” is made up of three domains. First is the individual primitive sense domain. In this realm, all that is observed are patterns and changes to patterns. For example, the change from soft sound to loud sound maybe recognized as a car, maybe just as an unpleasant noise.

<table>
<thead>
<tr>
<th>Sense</th>
<th>Street Corner Behavior</th>
<th>Other Possibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sight</td>
<td>Cars moving, Signal, walk sign, people, buildings</td>
<td></td>
</tr>
<tr>
<td>Touch</td>
<td>Sudden drop at curb, Pressure changes on foot</td>
<td>Uneven terrain, door sill</td>
</tr>
<tr>
<td>Hearing</td>
<td>Car sounds, People talking</td>
<td>Mid street, Parking lot, Auto garage</td>
</tr>
<tr>
<td>Taste</td>
<td>Coffee, Doughnut</td>
<td>Home, Office, Krispy Kreme</td>
</tr>
<tr>
<td>Smell</td>
<td>Car fumes</td>
<td>Garage, Anywhere w/ cars</td>
</tr>
</tbody>
</table>

Exhibit 6.4. Primitive Response to Street Corner Event

The second realm is the emergent realm. In this realm, the multiple senses have organized these patterns and are providing some sort of stimuli based on the primitive
changes. For example, the emergent processor may be so programmed to respond negatively to loud sounds, that even without recognition that a sound is a car, the person jumps back. More typically, the person must cross or not cross the street as a choice. At any rate, the fusion of these primitive recognitions together causes recognition of a time and space event labeled “street corner”.

The third domain is the activity domain. What the processor does with the recognition of “street corner” depends upon the mission programming. As above, if the person is fearful of loud noises, “loud noise” may simply swamp out “street corner” as the event. Or, at the other end of the spectrum, if the person came to the street corner to witness a specific car going by, no response may occur until “1957 Chevy” event occurs. This wide variety of responses depends upon what the walker (or its mission planner) wants it to do.

In the case of underwater instrumentation, the primary goals are exploration, protection, navigation, and counting. If one reviews all the possible missions, it is observed that the vast majority come under these categories. Going one level deeper, the specifics of the mission usually involve the activities of recognition, movement (phobic or phillic), and interaction. In terms of movement, there is an embedded assumption of detection and recognition. For example, in Biewald’s work on neural navigation, every movement behavior is tied to some feature of importance. Using only active acoustics and a regularly organized environment, all the movement commands involved active acoustic recognition of such features as “corridor”, “T shaped cross roads”, or “corner” and some reaction to that feature. So recognition of multi-sensor features is at the core of all the useful autonomous behaviors of utility to marine research.

Given this brief background it becomes clear that each event in time or space in the marine environment as well as each goal for a fixed point or vehicle based system,
can be broken into several component pieces. For this research, the breakdown of features has been carried to the lowest possible level, to the developmental level observed in infants just responding to light and dark, touch, sound, taste, and smell. The first data set (project 2, data set “1”) is the simple response at this lowest level, as outlined in Exhibit 6.3. But what emergent events might be of interest? One can begin with features that are spoken of in existing underwater missions. As examples of each of the four general cases mentioned above, a mission might be, “find and map a geothermal event”, “alarm if a diver gets under this boat”, “dive to 100 m if it rains”, “count the number of fishing boats coming in the channel each day”, or “track the blue whale heard 30 degrees to starboard.” The limit of this research is the recognition of the complex data objects such as “geothermal vent”, “scuba diver”, “rain”, “diesel fishing boat”, or “blue whale.” A system needs to be able to detect these emergent patterns in order to perform the actions described above.

**Paradigm Selection**

Given the problem described in Section 6.3, one can generate a short list of characteristics with which to select the network paradigm(s) to try. First, the primitive system takes real valued time and spatial samples and map them to one of seven output categories as outlined above. For any given training and testing data, the output categories are known. In addition, the system dynamic range can be well controlled. Therefore, for the primitive processor, the multi-level perceptron was chosen as the prototype network. For this project, all work will be done via backpropagation training of a three-layer MLP network. For data set one, various modifications to the input data presentation are attempted and percentage tallies are presented. As a practical matter, the later data set from Vailulu’u, data set 2, has very few exemplars of minima, so the number of categories used for this work will be the six
listed above, but with no category 7 minimum. For the environmental processor, since the structures are made up artificially, class seven is included once again, and in features which key on minimum values, minimum exemplars would need to be included in training and testing sets

**Data Set One: Primitive Behaviors**

**Description**

The first task is the evaluation of a neural network’s ability to recognize the primitive behaviors outlined above. The data set one to be applied to the network structure is an artificially constructed set of slopes, edges, curves, and boundaries, of a wide variety of ranges. This more than spans the input range seen in actual datasets from oceanographic reports. The first collection of samples is the ideal cases. A separate testing set is then created from the training set by the addition of more noise (See Exhibit 6.5).

In order to better understand categories four and five, additional graphics are provided. A structure called “Entering a Region” is defined as type 4. In every emergent feature of interest, there is a transition at the boundary of the region. This transition is from an area of no change in data parameter, which is defined as a type 1 area, to a region of change which may be increasing or decreasing in value (see Exhibit 6.6).

<table>
<thead>
<tr>
<th>Feature Type/ Category Number</th>
<th>Ranges/Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant  Type 1</td>
<td>Various Levels .1-.9</td>
</tr>
<tr>
<td>Linear Increase Type 2</td>
<td>Slopes from .1-.5, offsets from .2-.6</td>
</tr>
<tr>
<td>Linear Decrease Type 3</td>
<td>Slopes from .1-.5, offsets from .2-.6</td>
</tr>
<tr>
<td>Cubic Increase Type 2</td>
<td>Amplitude from .1-.5</td>
</tr>
<tr>
<td>Cubic Decrease Type 3</td>
<td>Amplitude from .1-.5</td>
</tr>
<tr>
<td>Tanh Increase Type 2</td>
<td>$1 &lt; a &lt; 3 \tanh(ax)$</td>
</tr>
<tr>
<td>Tanh Decrease Type 3</td>
<td>$1 &lt; a &lt; 3 \tanh(ax)$</td>
</tr>
<tr>
<td>Linear Entrance Boundary Type 4</td>
<td>Flat to Positive Slope</td>
</tr>
<tr>
<td>Description</td>
<td>Type 4 Region Profiles</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Cubic Entrance Boundary Type 4</td>
<td>Flat to Positive Slope</td>
</tr>
<tr>
<td>Tanh Entrance Boundary Type 4</td>
<td>Flat to Positive Slope</td>
</tr>
<tr>
<td>Linear Exit Boundary Type 5</td>
<td>Negative Slope to Flat</td>
</tr>
<tr>
<td>Cubic Exit Boundary Type 5</td>
<td>Negative Slope to Flat</td>
</tr>
<tr>
<td>Tanh Exit Boundary Type 5</td>
<td>Negative Slope to Flat</td>
</tr>
<tr>
<td>Linear Entrance Boundary Type 4</td>
<td>Flat to Negative Slope</td>
</tr>
<tr>
<td>Cubic Entrance Boundary Type 4</td>
<td>Flat to Negative Slope</td>
</tr>
<tr>
<td>Tanh Entrance Boundary Type 4</td>
<td>Flat to Negative Slope</td>
</tr>
<tr>
<td>Linear Exit Boundary Type 5</td>
<td>Positive Slope to Flat</td>
</tr>
<tr>
<td>Cubic Exit Boundary Type 5</td>
<td>Positive Slope to Flat</td>
</tr>
<tr>
<td>Tanh Exit Boundary Type 5</td>
<td>Positive Slope to Flat</td>
</tr>
<tr>
<td>Maximum</td>
<td>Varying slopes</td>
</tr>
</tbody>
</table>

**Exhibit 6.5.** Description of Data Set 1 Features

**Exhibit 6.6.** Type 4 Region Profiles
For example, when approaching an acoustic source, a point will be reached where the signal is above the threshold of the detector. After that, the envelope of the signal will be steadily increasing. The thermocline is another example. When approached from the top, there is a region of little if any change in the temperature profile of the water. But at the boundary, the temperature of the water begins to decline. In both these scenarios, it is the boundary that is of importance to recognize the feature. Whether the slope is positive or negative, or is varying in a concave or convex manner is irrelevant for the recognition process.

For completeness, the profiles possible for leaving a region are also shown (See Exhibit 6.7).

![Exhibit 6.7. Type 5 Boundary, Leaving Region of Interest](image)

Since the training data is very deterministic in nature, and spans both offsets and amplitudes for these basic shapes, it was decided that rather than make another very similar block for testing, noise of various amplitudes would be applied to the training data to create a
testing data set. This is closer to what the system would see in actual operation, where the curvature of temperature is very similar, because of the laws of thermodynamics, but where several noise terms may be added. The testing set for data set one is the training set with additional noise terms, specifically, 0.01, 0.05, and 0.1 random normal distribution, absolute amplitude, variance=1.0. In scaled terms, this corresponds to 1.25%, 6.25%, and 12.5% noise for the scaled input.

**Functional Test**

The first experiment performed with data set 1 was a simple functional test. The input frame was chosen to be ten samples wide. In practice, these could be ten time values or an array of ten sensors on the instrument. The internal structure of the network consists of three layers, 10 input neurons, 10 hidden layer one neurons, and 6 output neurons. After several early experiments, sweeping alpha, learning rate, # iterations, and weight initializations, the following network parameters were established (See Exhibit 6.8).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range Tested: Final Value</th>
</tr>
</thead>
</table>
| Initial Learning Rate $\mu$ | 0.1, 0.5, 0.8, 0.9: $0.8$  
(Changed to .9 for span and regression experiments) |   |
| Alpha                  | 0.5, 1, 2: $0.5$                                                                         |
| Initial weights        | 0.1, 0.4, 0.5 amplitude random normal distribution, unit variance: $0.5$            |
| #Iterations            | 20,000 100,000 200,000 500,000: $200,000$                                              |

**Exhibit 6.8. Evaluated Multi-Level Perceptron Network Parameters**

The program used was Matlab® combined with the "BPN_3.5" program. Previous experiments with enhancements such as momentum learning resulted in no improvement.
in convergence, so this version of the program utilizes the basic algorithm plus a weight initialization scheme of Nguyen and Widrow (Nguyen, 1990).

The results of the data are shown below (See Exhibit 6.9). In this, the network demonstrated an ability to separate the categories nicely for low noise environments. However the noise does degrade performance rapidly. It is unclear whether the network performance is actually degraded or has the noise reached such a magnitude as to mask the pattern of interest.

<table>
<thead>
<tr>
<th>Run/Data</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01 noise, var=1.0</td>
<td>97%</td>
</tr>
<tr>
<td>0.05 noise var=1.0</td>
<td>88%</td>
</tr>
<tr>
<td>0.1 noise var=1.0</td>
<td>57.5%</td>
</tr>
</tbody>
</table>

**Exhibit 6.9.** Functional Noise Performance 200000 iteration

Examining the confusion matrix, (See Exhibit 6.10), a pattern does emerge.

<table>
<thead>
<tr>
<th>Test</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>xxxx</td>
<td>xxxx</td>
<td>xxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td>xxx</td>
<td>xxx</td>
</tr>
<tr>
<td>3</td>
<td>xxx</td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>xxxxx</td>
<td>xxx</td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>xx</td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Exhibit 6.10.** Confusion Matrix for Dataset 1, .1 added noise
Of the errors, there are two main categories. First the confusion between a type 1 region and all others accounts for 37.8% of all the errors above. Secondly, errors where a boundary (type 4 or 5) is confused with a ramp (type 2 or 3) accounts for 45.1% of the errors. A careful examination frame by frame may be of assistance. Indeed it can be seen that for both types of error, the noise is distorting the input making it difficult to recognize even for the human observer (See Exhibits 6.11, 6.12).

Exhibit 6.11. Noisy Signal vs Ideal, Type 1

Notice that in what is an absolute zero slope ideal signal, the noise added gives a perceived slope. Indeed, if a regression is performed on this line, the slope becomes non-zero.
Exhibit 6.12. Ideal Signal vs Noisy Signal, Type 4

In addition, the noise term in 6.12, blurs the boundary until only a slope can be discerned.

From this, it is apparent that various methods need to be examined to improve the noise performance of the network by preprocessing the inputs. In addition, it can be surmised that it is important to have low noise measurements or at least some form of hardware low pass filtering for input data, prior to presentation to the network.

Mean and Span Test

One of the variables that is important to neural network performance is the scaling of inputs and outputs. For example, since the output of the binary sigmoid is $0 < y(k) < 1$, it would be foolhardy to scale outputs to 1.0 or larger. So, the network data must be matched to the network range. In the preliminary example, the data was scaled from 0 to 0.9 on the output and 0.1 to 0.9 on the input. In addition, the minimum of the signals was subtracted from the span, thus referencing the signals to 0.1. This worked in a low noise environment, but it is useful to repeat the experiment with a variety of input ranges. A sequence was
devised for this, holding the output span constant, and varying the input behavior of the signals (See Exhibit 6.13). In this series of tests, the data was processed to reference the signal either to the minimum value of the data (range 0.1 to 0.9) or to mean center the data at the center of the sigmoid curve (0.5) and again scale. The idea is that for baseline signals, place the base of the signal near the center of the sigmoid where the neuron is most responsive, rather than amplify the low level changes amplifying the noise as well before scaling. For all these experiments, the .05 and 0.1 random normal testing set was used. Learning rate for this experiment was raised to 0.9 in the hope of speeding the learning process. Also, the comparison between using a bipolar sigmoid form or binary sigmoid form was included. The summary results are as shown in Exhibit 6.14. As before, the percentages represent total number correct out of a 200 point testing set. All networks were trained to 200,000 iterations.

Exhibit 6.13. Difference in Input Signal Span and Offsets

<table>
<thead>
<tr>
<th>Noise Set</th>
<th>Span/Baseline</th>
<th>Binary Sigmoid</th>
<th>Bipolar Sigmoid</th>
</tr>
</thead>
</table>


The results of this experiment provide more guidance as to how to format the signals for improved recognition performance. Remembering that the initial functional experiments achieved 57.5% and 88% recognition for the .05 and 0.1 noise test sets respectively, it is observed that the 0.1 performance has improved significantly while the .05 performance has not been as high. The changes in learning rate may be the cause as that is the only parameter changed in the 0.1- 0.9 span comparison. Either value will give functional results, however. In this experiment, there are similar conflicts. For example, for .05 noise, the binary sigmoid run outperforms the bipolar sigmoid for 0.5 mean centering, but not for 0.1 offset referencing.

For the 0.1 noise, the case is completely reversed. In general, the binary sigmoid gives slightly better performance on average, and once this is selected, there is no significant difference as to which spanning and offset schemes are used. As long as the input span is kept to within the range 0.1-0.9, and a few learning rates are compared, then reasonable results may be achieved.

Fitting /Difference Preprocessing Test

As mentioned earlier, at 0.1 random normal additive noise, the signal itself begins to be lost. Two of the simpler methods to provide more information to the neural network is to look at the discrete differences within each sample and also to look at a polynomial fit of the data. For this experiment, the same function, bpn_3.5, is used for the neural paradigm. In addition, the learning rate, alpha, number of iterations, span, and offset are all held constant. What is changed is the presentation of the input data. This experiment presents fitted versions of the data or combinations of fitted coefficients and data to the network for recognition.
For the difference test, the discrete differences between points within the sample are evaluated using the Matlab “diff” function. With this, for 10 inputs in the data field, there will be nine differences between points. These nine differences are appended onto the 10 original points of the data field, thus giving 19 points to the input of the neural network.

For the polynomial fitting test, three different formats were applied. First, the Matlab “polyfit” function was used to compute the coefficients for a cubic fit to the input patterns. Second, the input data was presented with the fit coefficients only. Then, the input data with the polynomial fit of the data appended was presented. Lastly, the cubic fit of the data alone was presented to the network. For all cases, learning rate was set to 0.9, alpha 0.5, number iterations to 200000, span 0.1 to 0.9, binary sigmoid. Testing data is the 0.1 random normal additive noise set. The results are in Exhibit 6.15.

<table>
<thead>
<tr>
<th>Input Form</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training + Diff Training</td>
<td>42%</td>
</tr>
<tr>
<td>Polyfit Coefficients Only</td>
<td>53.5%</td>
</tr>
<tr>
<td>Training + Polyfit Curve</td>
<td>61%</td>
</tr>
<tr>
<td>Polyfit Curve Only</td>
<td>61%</td>
</tr>
</tbody>
</table>

**Exhibit 6.15. Alternative Input Format Test Results**

**Data Set 1 Conclusion**

Data set one demonstrated that the various regions in a sensor data set can be recognized as seven basic categories by a neural network. In addition, a multi-level perceptron neural network can be trained to recognize these regions uniquely. As noise increases, the data becomes ambiguous, resulting in rapidly increasing misclassification of patterns. After experimenting with network structure and input ranges as well as neuron transfer functions, these errors seem to be more a function of the noise rather than any parameter in the neural network producing noise sensitivity. A first attempt at reducing the noise did not produce significant improvement. The methods of including derivative information and fitting a cubic curve to the noisy input data produced no improvement in recognition rates. The best achievable rate with .1 random normal additive noise, was 63.5% and for 0.05 random normal additive noise, 75.5%.
Vailulu’u

With a functional demonstration available as well as some network structural parameters established, it is necessary to evaluate the network performance on actual oceanographic data. Datasets are available from several sources (Hart, Cetacean Research, NOAA). One area of particular interest was the work of the NOAA sponsored Hawaii Underwater Research Laboratory. In 2000 and 2001, this organization sent two missions to explore an underwater volcanic source by the name of Vailulu’u in American Samoa (See Exhibit 6.16) (Hart, 2002).

Exhibit 6.16. Vailulu’u Bathymetry (Staudigel, 2002)

The feature consists of a seamount rising from the abyssal plain, with a crater whose rim is at approximately 500m depth. The bottom of the crater is at 1000m. There are fissures in the wall which impact the hydraulic flow in and around the crater. Within
the crater are several hydrothermal vents. The location of the vents is as yet unknown, but studies of Mn and He$^3$ concentrations as well as turbidity studies indicate a strong series of hydrothermal vents, much more productive than Atlantic mid ocean smokers (Staudigel, 2002).

**Data Set 2**

Data Set 2 consists of a sequence of primitive features selected from temperature, salinity, nephelometry, or fluorometry data streams from the 2001 expedition. The files were received in Excel format and included passes SAM14-SAM29 over and around the crater feature. In practice, the data was taken from a towed fish which was allowed to oscillate between depths during a single tow. This “Tow-yo” method involved speeding up and slowing down the boat to allow the fish to drop to near the bottom of the crater and then to rise back above the edge. Some passes involve a single descent and return. Others had several (4-8) descent and return oscillations, typically between 500-700m (See Exhibit 6.17).

No effort was made to “cherry pick” the best samples, however, each sample was taken from a representative feature type, that is a “maximum” sample came from a nephelometry “peak” in the data as seen on a broader scale. Data from 18 separate casts were used for the training and testing sets. All four of the sensor types were merged sequentially into single training and testing files. One of the underlying assumptions is that the network will train to the features discussed in data set one without regard for the physical significance of the measurement. It would then be possible to create a single processor to handle all of the primitive feature extractions.
Sample features totaling 208 features were extracted, 52 from temperature, salinity, nephelometry, and fluorometry channels. Within these, members of the six categories were evenly distributed among the samples. Then, each set was split into training and testing portions. The 26 training values from each set were then merged into a 10x104 element training set. The testing set was created in a similar manner. A training and testing desired output map was created listing the correct feature map (1-6) as was used for dataset 1. At this point, no attempt was made to scale or preprocess the data. It was stored in engineering units as provided from the 2001 expedition data.

**Global vs Local Scaling Experiment**

The first observation in the data is that the units do vary considerably whether from temperature or from nephelometry for example. In order to extract the desired features, it is therefore important to scale and reference the data in a way which matches the input to the sensitivity of the neural network. A question emerges though as to what is the best way to scale this data. Should each data frame be scaled to its own range, or should the ensemble...
dataset be scaled to the range of the entire set. Much of this depends on the noise within the 
data set and the dynamic range of the incoming data. Also, from an operational view, can an 
ensemble data set be created in real time?

Given these questions, it is helpful to evaluate the performance of the network in both 
circumstances. From the Vailulu’u dataset created, a survey of the numerical range data is 
performed (See Exhibit 6.18).

<table>
<thead>
<tr>
<th>Measured Parameter</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>5.3693-29.5807 deg C°</td>
</tr>
<tr>
<td>Salinity</td>
<td>34.4775-36.3660 S/m</td>
</tr>
<tr>
<td>Nephelometry</td>
<td>.0281-1.1958 raw volts</td>
</tr>
<tr>
<td>Fluorometry</td>
<td>0-1.8070 raw volts</td>
</tr>
</tbody>
</table>

Exhibit 6.18. Variable Ranges, Dataset 2

As can be seen, there are multiple ranges to be dealt with. Remembering that the data 
is presented one sample at a time, it is possible to scale the data in several ways. The first 
method, which is denoted local scaling, involves taking each individual sample and scaling it 
to the maximum and minimum of that sample. For example, if sample 1 is a temperature 
sample whose max is 28.3C and min is 23.4C, then the values of that sample will be 
referenced to the minimum, then scaled by the difference (28.3 - 23.4) of the two values. This 
will create a value in the range of 0 to 1. Then, in order to avoid the extremes of the sigmoid 
function, the data is offset and scaled to a range of 0.1- 0.9.

The other approach is to scan the entire training and testing data set and scale all the 
values to the global maximum and minimum of the collected training or testing dataset. This 
is more problematic from a real-time implementation perspective but has merit in terms of 
final recognition rates.

For this experiment, the primary variable to be changed is whether the data is scaled 
based upon global or local maxima and minima. As in the dataset one experiments, network 
parameters are frozen to compare changes in the data presentation. For this experimental 
matrix, learning rate is 0.9, alpha is 0.5, iterations is 200000, binary sigmoid, 10 input 
neurons, 10 level 2 neurons and 6 output neurons.
The results of the basic global and local experiments are 59.6% recognition for local scaling and 64.4% for global scaling. The confusion matrix is shown in Exhibit 6.19. There are some common features to both local and global errors. In the “no prescale” categories, confusion between category 6 (maximum) and 1 (flat) accounts for 45% and 48.6% of the errors respectively. The confusion between straight ramps (cat 2 and 3) and boundaries (cat 4 and 5) accounts for 33% and 35% of the errors as well.

Again, a frame by frame review of the samples shows that in many cases, the actual categorization of the sample is difficult even by a human observer (See Exhibit 6.20).

<table>
<thead>
<tr>
<th>Test</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>L</td>
<td>L</td>
<td>G</td>
<td>G</td>
<td>LLLLLLL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>G</td>
<td>LL</td>
<td>GG</td>
<td>L</td>
<td>GG</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GG</td>
<td>G</td>
<td>LLLL</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GGG</td>
<td>GG</td>
<td>G</td>
<td>LLLL</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GGG</td>
<td>G</td>
<td>GGGGGG</td>
<td>LLLL</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>GGGGG</td>
<td>G</td>
<td>GGGGG</td>
<td>L</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

Exhibit 6.19. Confusion Matrix for Input Scaling (G=Global Norm, L=Local)
Another approach to improving recognition of the Vailulu’u data is the possibility of reducing the degrees of freedom of the problem. For example, it is not reasonable to have a maximum event immediately following a continuously decreasing event. Based on this, a review of the data set was performed. The history of each event consisting of the previous classification category is added to the input plane (See Exhibit 6.21).
Exhibit 6.21. Feedback of Feature Category to Neural Network

So to all the events in the training and testing sets, the classification of the previous frame in the original data is now added. The results are encouraging. For the conditions listed above, and with global scaling, the recognition rate is 76.9%. This is a greater improvement over the results with no history previously seen. In addition, when observing the types of errors present in the confusion matrix (See Exhibit 6.23), there are two modalities. 45% of the errors involve confusion with type 6 (maximum), 29% with type 1 (flat), and 37.5% with type 4/5 boundary with type 2/3 slopes. The improvement to the recognition rate is significant as compared with variation in the scaling and span/offset shown earlier.

Exhibit 6.22. Confusion Matrix for Data Set 2 History Experiment
Notice that the history experiment doesn’t simply store more of the sample, i.e. 20 input points, vs 10. Instead, it makes a decision based upon the kth sample frame and uses that decision as additional information for the k+1 sample frame.

**Feature Synthesis Mapping**

To this point in the research, only primitive features have been evaluated. Yet as shown in Exhibit 1.1, the desired goal is to recognize multi-sensor features in the environment. This section of research now looks at several environmental features, defines the primitive parameters assumed to be associated with those features and then evaluates the suitability of the neural network for mapping the primitive features to the emergent environmental structures.

For this work, several features which might be of mission importance are identified. These include:

1) Thermocline
2) Seamount Crater (i.e. Vailulu’u rim )
3) Hydrothermal Vent
4) Ocean current
5) Pollution Source
6) Tidal Inlet
7) Estuary
8) Passive SONAR Signal
9) Rain
10) Day/.Night

Each one of these must now be defined by the primitive sensory structures that define the event. Without extensive data, assumptions are made as to possible sensor responses which may define the event.

**Multiple Feature Maps**

Having established the menu of test features, two more tasks must be performed prior to developing the neural processor for multiple feature maps. The first task is to recognize that each feature in reality has several ways in which the vehicle or sensor pod may interact
with it. For example, a thermocline is a variation of temperature with depth. However, as a large feature, much larger than the vehicle, it must be broken down further into the various parts. A list of possible interactions with the thermocline must be developed.

For a vehicle, the thermocline may be entered from the top or the bottom. It may also leave the thermocline from the top or the bottom. In addition, it was chosen to distinguish between rising in the thermocline or descending within the region. This is the vertical equivalent of the increasing or decreasing categories of the primitive behaviors. So in essence, there are 6 interactions associated with the feature thermocline. By breaking this into the smaller components, there are now navigational events which can be used. If a mission planner wishes to force the vehicle to operate at the bottom of the thermocline, the vehicle would recognize an “entering from the top” event, followed by “Within Descending”, and then stop at the “Inside Leaving Bottom” event.

Similarly for all the features listed above, there are sub features that identify where the event is on the larger feature. If one looks carefully at the events, one can see that by selecting a specific chemical, acoustic signal, or acoustic return, the vehicle can be very selective at operating around a feature, while retaining the basic structure above.

The second task is to define each feature in terms of its distinguishing sensor characteristics. Using the sensorial fields outlined earlier, a reader can easily define in terms of sensor behavior each event.

Example 1: A thermocline is a layer of temperature change versus depth in a water column. For the feature map, the temperature and pressure channels would have particular behaviors while the rest of the sensors would be irrelevant. This is equivalent to the statement, “perfume smells good.” No one cares how it tastes, feels, or looks, as long as it smells good.

Example 2: During a rain storm, there is a lot of rain noise coming from the surface. The temperature at the surface drops, and the salinity at the surface drops.

Example 3: As one approaches an inlet from the ocean as the tide goes out, there is some acoustic noise and the salinity changes. The temperature also changes, the pressure remains constant, the vehicle gets deflected and jostled, and the algae and the clarity of the water may change.

In most systems, the actual data from the sensors are used. This would present a massive challenge in neural network design. Here, the processor only receives the summary results, 1 of 7 primitive features, from the sensors. In the thermocline example, the approaching the top boundary from above would yield a type 4 (entering a region) temperature event, concurrent with a type 2 (increasing) pressure event. Motion downward
within the thermocline would yield a type 3 (decreasing) temperature event, with a continued type 2 (increasing) pressure event. In Exhibit 6.23, the list of sensors which are most involved are in fact reporting these types of events to the environmental processor.

Other sensors that do not have involvement are given a “don’t care” status, borrowing lingo from digital computing. This provides the means to create a training and testing data set of valid measure. Fields that are don’t cares are made to have all possible values in the training and testing sets within the limits of the number of samples per training/testing epoch.

The basic features are broken into 39 respective sub features. From a neural network perspective, no encoding is used, simply 39 outputs from the network, one of which reaches a maximum if that event occurs (see Exhibit 6.23).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Event</th>
<th>Primary Factor</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermocline</td>
<td></td>
<td>Temp, pressure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outside Approach Top</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Inside Approach Top</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Within Descending</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Crater</td>
<td></td>
<td>Passive acoustic, temp, pressure, turbidity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Horiz Approach</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Vert Approach</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Horiz Departure</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Vert Departure</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>Thermal Vent</td>
<td></td>
<td>Passive acoustic, temp, pressure, spectroscopy, turbidity, inertial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Horiz Approach</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Vert Approach</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Horiz Departure</td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

Exhibit 6.23. Feature and Event Map
<table>
<thead>
<tr>
<th>Feature</th>
<th>Event</th>
<th>Primary Factor</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vert Departure</td>
<td>Current</td>
<td>Temp, salinity, inertial</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Approach</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Departing</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>Pollution Source</td>
<td>Approach</td>
<td>Turbidity, fluoroscopy, spectroscopy</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Departing</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>Inlet/Tidal/(Ocean)</td>
<td>Approach</td>
<td>Acoustic, temp, salinity, turbidity, optical, inertial</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Departing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hydrophone Event</td>
<td>Departing</td>
<td>Acoustic only</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Rain</td>
<td>Departing</td>
<td>Acoustic, temp, salinity</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Within</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Leaving</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Centered</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Beginning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Ending</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Noon</td>
<td>Optical, temp, acoustic</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Sunset</td>
<td></td>
<td>37</td>
</tr>
</tbody>
</table>
Exhibit 6.23. Continued

234 separate training and testing samples were created. Since the inputs are well scaled outputs from the primitive behaviors, no noise is added to the testing set. The independence comes with the random selection of the “don’t care” inputs. Specifically, if a sunset event occurs, then the turbidity can be maximum, minimum, entering, leaving, or anything else. So for the training and testing, it is forced to all these conditions. Given 7 input possibilities per channel, and 9 sensor inputs for this simulation, the number of inputs to the network is 63 for this experiment.

Multiple Feature Maps: Results

The neural network structure for this entity is summarized as follows (See Exhibit 6.24). The test results are summarized in Exhibit 6.25.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Learning Rate</td>
<td>0.9</td>
</tr>
<tr>
<td>Sigmoid/alpha</td>
<td>Binary, 0.5</td>
</tr>
<tr>
<td>Inputs</td>
<td>63</td>
</tr>
<tr>
<td>Outputs</td>
<td>39</td>
</tr>
<tr>
<td>Hidden layer</td>
<td>63</td>
</tr>
<tr>
<td>Convergence</td>
<td>200000 iteration</td>
</tr>
</tbody>
</table>

Exhibit 6.24. Emergent Processor Parameters

<table>
<thead>
<tr>
<th>% Correct</th>
<th>Event</th>
<th>Miscategories</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermocline (tcline)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33%</td>
<td>Outside Approach Top</td>
<td>Passive acoustic</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Day/Night</td>
<td></td>
</tr>
<tr>
<td>16%</td>
<td>Inside Approach Top</td>
<td>Passive acoustic</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Day/Night</td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>Within Descending</td>
<td>Day/Night</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Estuary</td>
<td></td>
</tr>
<tr>
<td>33%</td>
<td>Within Ascending</td>
<td>Day/Night</td>
<td>4</td>
</tr>
</tbody>
</table>
## Exhibit 6.25. Event Map Results: 39 Categories

<table>
<thead>
<tr>
<th>%Correct</th>
<th>Event</th>
<th>Miscategories</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td>33%</td>
<td>Inside App Bottom</td>
<td>Passive acoustic</td>
<td>5</td>
</tr>
<tr>
<td>16%</td>
<td>Outside App Bottom</td>
<td>Day/Night Tcline (2)</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Day/Night Tcline(3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passive acoustic</td>
<td></td>
</tr>
<tr>
<td>Crater</td>
<td>Horiz Approach</td>
<td>Day/Night Within Crater</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Current</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rain</td>
<td></td>
</tr>
<tr>
<td>83%</td>
<td>Within</td>
<td>Passive acoustic</td>
<td>8</td>
</tr>
<tr>
<td>66%</td>
<td>Vert Approach</td>
<td>Current</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rain</td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Horiz Departure</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>100%</td>
<td>Vert Departure</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>Thermal Vent</td>
<td>Horiz Approach</td>
<td>Crater Horiz App</td>
<td>12</td>
</tr>
<tr>
<td>66%</td>
<td>Within</td>
<td>Estuary</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passive acoustic</td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Vert Approach</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>100%</td>
<td>Horiz Departure</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>100%</td>
<td>Vert Departure</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Current</td>
<td>Approach</td>
<td>Passive acoustic Inlet</td>
<td>17</td>
</tr>
<tr>
<td>50%</td>
<td>Within</td>
<td>Thermocline</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passive acoustic</td>
<td></td>
</tr>
<tr>
<td>16%</td>
<td>Departing</td>
<td>Day/Night Inlet</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passive acoustic</td>
<td></td>
</tr>
<tr>
<td>Pollution Source</td>
<td>Approach</td>
<td>Rain</td>
<td>20</td>
</tr>
<tr>
<td>33%</td>
<td></td>
<td>Passive acoustic Estuary</td>
<td></td>
</tr>
</tbody>
</table>

Exhibit 6.25. Continued

<table>
<thead>
<tr>
<th>%Correct</th>
<th>Event</th>
<th>Miscategories</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>Within</td>
<td>Inlet</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Current</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66%</td>
<td>Maximum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>Departing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83%</td>
<td>Approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>Departing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83%</td>
<td>Within</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estuary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Departing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inlet/Tidal/(Ocean)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Departing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Within</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83%</td>
<td>Approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>Departing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83%</td>
<td>Within</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hydrophone Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33%</td>
<td>Approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>Leaving</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66%</td>
<td>Centered</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83%</td>
<td>Beginning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Ending</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66%</td>
<td>Maximum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day/Night</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Noon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Sunset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Sunrise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Night</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exhibit 6.25. Continued</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Several interesting patterns emerge. First, it is observed that many features are confused with one of the day/night categories (36-39). This is due to a fundamental ambiguity between spatial and time events that underlies this system. To this point, there is no recognition that the vehicle is moving over a feature and this requires time. Therefore, if sensor inputs are changing, this system does not have anyway of knowing if the change is due to time strictly or actual spatial variations. Secondly, several events are
confused with passive SONAR categories. If one examines the training and testing set, it is observed that several patterns for a desired result given the “don’t care” approach contains a data event as an artifact in the passive SONAR category. Many of the other errors actually are still within the same feature, i.e. leaving the approaching the crater maps to within the crater, or the individual vent maps to the crater as a whole. All the other errors involve the key features being diluted by the system still looking at the entire sensor suite during the decision making process.

For the day/night ambiguity problem, the test was repeated, this time with only 35 categories and the day/night structure removed from the training and testing See Exhibit 6.26).

<table>
<thead>
<tr>
<th>% New Correct / % Old Correct</th>
<th>Event</th>
<th>Miscategories</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermocline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%/33%</td>
<td>Outside Approach Top</td>
<td>Passive acoustic</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Day/Night</td>
<td></td>
</tr>
<tr>
<td>50%/16%</td>
<td>Inside Approach Top</td>
<td>Passive acoustic</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Day/Night</td>
<td></td>
</tr>
<tr>
<td>66%/50%</td>
<td>Within Descending</td>
<td>Crater Passive acoustic</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pollution source</td>
<td></td>
</tr>
<tr>
<td>50%/33%</td>
<td>Within Ascending</td>
<td>Crater Passive acoustic</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pollution source</td>
<td></td>
</tr>
<tr>
<td>66%/33%</td>
<td>Inside App Bottom</td>
<td>Pollution source</td>
<td>5</td>
</tr>
<tr>
<td>50%/16%</td>
<td>Outside App Bottom</td>
<td>Tcline (83%) Pollution source</td>
<td>6</td>
</tr>
<tr>
<td>Crater</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33%/50%</td>
<td>Horiz Approach</td>
<td>Crater (10)</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Passive acoustic</td>
<td></td>
</tr>
<tr>
<td>83%/83%</td>
<td>Within</td>
<td>Inlet</td>
<td>8</td>
</tr>
<tr>
<td>100%/66%</td>
<td>Vert Approach</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>100%</td>
<td>Horiz Departure</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>100%</td>
<td>Vert Departure</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>Thermal Vent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%/0%</td>
<td>Horiz Approach</td>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>
### Exhibit 6.26. Event Map Results: No Day/Night

<table>
<thead>
<tr>
<th>% New Correct / % Old Correct</th>
<th>Event</th>
<th>Miscategories</th>
<th>Cat #</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pollution Source</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33%/33%</td>
<td>Approach</td>
<td>Vent Thermocline Current Passive acoustic</td>
<td>20</td>
</tr>
<tr>
<td>0%/0%</td>
<td>Within</td>
<td>Passive acoustic Inlet</td>
<td>21</td>
</tr>
<tr>
<td>66%/66%</td>
<td>Maximum</td>
<td>Thermocline Inlet</td>
<td>22</td>
</tr>
<tr>
<td>50%/50%</td>
<td>Departing</td>
<td>Thermal vent Current Passive acoustic</td>
<td>23</td>
</tr>
<tr>
<td><strong>Inlet/Tidal/(Ocean)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>Approach</td>
<td>Vent</td>
<td>24</td>
</tr>
<tr>
<td>100%</td>
<td>Departing</td>
<td>Thermocline</td>
<td>25</td>
</tr>
<tr>
<td>100%</td>
<td>Within</td>
<td>Passive acoustic</td>
<td>26</td>
</tr>
<tr>
<td>Estuary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83%/83%</td>
<td>Approach</td>
<td>Vent</td>
<td>27</td>
</tr>
<tr>
<td>83%/50%</td>
<td>Departing</td>
<td>Thermocline</td>
<td>28</td>
</tr>
<tr>
<td>100%/83%</td>
<td>Within</td>
<td>Passive acoustic</td>
<td>29</td>
</tr>
<tr>
<td><strong>Hydrophone Event</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16%/33%</td>
<td>Approach</td>
<td>Thermocline Crater Inlet</td>
<td>30</td>
</tr>
<tr>
<td>50%/0%</td>
<td>Leaving</td>
<td>Thermocline Pollution source</td>
<td>31</td>
</tr>
</tbody>
</table>
Categories that were being confused with day/night have received improved recognition rates. Categories which key on only one or two parameters are confused by “Don’t care” inputs in other categories. In other words, a thermocline pattern that was presented with signal in the passive SONAR channel automatically classified as a passive SONAR signal. As seen in the passive SONAR project, the decision making process needs to be distributed to a more localized basis. For example, a net exclusively defined for thermoclines would no longer take input from the spectroscopy source. Likewise, chemistry payloads, when looking for a specific pollutant, would look exclusively at that data channel, ignoring the rest.

Two Category Testing

To test the theory of limiting the degrees of freedom for improvement of recognition rates, the categories in the environmental maps were mapped to “feature of interest” or “not feature of interest. The categories within the thermocline feature retained the maps from before consisting of 6 possible interactions. However, now all other categories were mapped to a catch all “not a thermocline” category, 7. The network was trained to the new map. The results can be observed in exhibit 6.27. What is seen is an improvement in recognition rates. The limits in this case are the number of samples of the thermocline features in the training and testing data sets. Another feature of this is the false positives when in the other features. The total false positives are 11% for all categories not a thermocline event.

<table>
<thead>
<tr>
<th>Feature/Map #</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside Top Approach</td>
<td>83%</td>
</tr>
<tr>
<td>Inside Top</td>
<td>83%</td>
</tr>
<tr>
<td>Approach</td>
<td>Recognition Rate</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Within Descending</td>
<td>83%</td>
</tr>
<tr>
<td>Within Ascending</td>
<td>83%</td>
</tr>
<tr>
<td>Inside Bottom Approach</td>
<td>66%</td>
</tr>
<tr>
<td>Outside Approach Bottom</td>
<td>83%</td>
</tr>
</tbody>
</table>

**Exhibit 6.27. Thermocline Only**

**Recognition Results**

**Discussion**

The concept of environmental feature recognition consists of two fundamental components. First is the ability to reduce the complex variety of data into primitive data structures which can be recognized by a basic neural network. Then, combinations of these basic patterns must be recognized as more complex structures which this author refers to as emergent features. Both halves of this problem have been demonstrated to better than 75% recognition accuracy using a basic MLP network. One of the features not mentioned before is the consequence of a mistake. Fortunately, the data field here represents approximately 10 seconds of samples with a 1 sec update rate as was used on the Vailulu’u expedition. This limits the maximum position error for recognizing the boundary. The most troublesome confusions in the primitive categories involved boundaries and the maximum versus the within the region categories. But since all of these are single point events, the software can look at the next two or three frames of data and deduce that a boundary has occurred. For example, when entering the thermocline area, the network may misclassify the boundary as an increasing temperature event. But averaging the classes on both side of the boundary, the observer will see that first there were only type “1” events, now there are only type “2” events. From this transition it can be concluded that a boundary was indeed crossed within a few sample cycles of that boundary.

All of the categorization errors in the Vailulu’u data set had to do with high levels of noise in the patterns and frame timing errors. For all this work, a 10 position sequence of sensor data is used as an input sample. As an example, envision a boundary occurring at the 9th position in the 10 position frame. This will almost certainly be categorized as one side of the boundary or the other. But as the event moves through the frame, when the boundary reaches the middle, say position 5 or 6 out of the 10 sample frame, then the network will correctly recognize the event. Patterns were picked out of a data stream with no optimization for boundary centering selected. In other words, the maximum (primitive category 6) events, might have had the maximum point at position 7 of the 10 position frame. For instance, certain patterns, such as the maximum misclassifications, if the pattern had been framed one sample before or after that taken, the sample would have been correctly classified.

In terms of the emergent behavior, there is the space-time ambiguity to deal with. There are two ways of working around this. First, as was demonstrated, by separating time events from spatial events, the ambiguity is broken. If one emergent event system is needed, the addition of a clock input would be in order.
Yet for up to 9 major categories, the system had very little ambiguity. Those features which were uni-sensor type, such as a “passive SONAR” event, are easily confused with a multi-sensor event. It would therefore be prudent to process the low dimensional events in a priority manner, such that, if a recognition occurs in one of those events, it would, “lock out” the other sensors while recognizing the event. For example, when a “blue whale” is recognized, the other sensors should not be examined for input to the classifier. However, they may be used for explicit scientific data.
Conclusions

Functionality of neural networks to recognize features in passive acoustic signals and multi-sensor environmental features has been demonstrated. For man made acoustic sources as well as most cetacean sources, recognition rates of 100% were achieved. For certain troublesome categories, the ensemble approach to recognition is non-optimal. Similar to regional processing in the brain, after initial coarse classification has been made, particularly for problematic inputs (i.e. fish), it may be necessary to develop local recognition tools based on neural, wavelet, statistical, or analytical methods. The best achievable recognition for fish using wavelets, pulse gathering, and hybrid networks was 72% in a seven fish set.

It was demonstrated that several features in the ocean can be broken into simple components by sensor feature, then recognized by reduced versions of those sensor classes. Recognition of the primitive features reached 76.9% with the inclusion of the previous primitive feature recognized as an input.

Emergent features reached 100% recognition for such objects as “Thermal Vent” and “Tidal Inlet”. Areas of overlap in recognition have to do with features that have limited sensor definition overlapping the sensor definition of a multi-sensor feature. For example, generic “current” overlapping tidal flow of an “inlet” because “current” is one component of “inlet”.

While not ready for a fully autonomous tool, the neural approach is ready to be used in conjunction with other preprocessing methods to enhance performance. In addition, the data gathering activity associated with training and testing of neural networks for these applications produces a benefit in that the data itself is of scientific value and encourages detailed data taking in areas not previously studied.
Recommended Future Work

Data Gathering

Based on this body of work, it is obvious that one of the primary tasks to perform as an ongoing effort is the gathering of more field data. This process in the acoustics has already begun with the use of hydrophones on the Florida Tech summer Marine Field Projects Cruises. Given the location of Florida Tech, it would be appropriate and useful to establish listening and geophysical sensor posts at Sebastian Inlet, near Port Canaveral, and at the edge of the gulf stream. Also, a research effort in in situ chemistry monitoring needs to begin. The chemistry signals, either from mass spectroscopy, or optical methods can provide an entire area of research for students both in conjunction with environmentally based navigation and for pure scientific research.

In addition, The University needs to develop a library of sound and geophysical data records and as a separate task for undergraduate DMES and perhaps marine biology students to maintain those records in an internet accessible form. This would be a valuable set of exercises independent of the use for neural networks, and are absolutely essential in order to validate any new designs.

Distributed vs Ensemble Networks

In project two, the recognition rates improved by changing the environmental mapping problem from one multi-category network, to a series of two category networks. The next phase of this research should include fully investigating this in terms of recognition, speed, and memory requirements of the implementation. It also applies to the passive SONAR project in that, despite the poor results shown for fish, it may be that making the problem into a series of two class problems on a series of simple features in the fish sound files, i.e. specific frequencies, one might achieve much better rates.

Pre-processing

Wavelet analysis was only looked at from the aspect of use in the same manner as FFT’s. Within the subject of wavelets, the different families of wavelets, the different ways to look at the coefficients, and the different specific wavelet transforms, all need to be examined in the context of improved recognition rates.

Alternate Paradigms

Another issue is the particular neural paradigms used in this research. On purpose, all the networks were made with uniform interfaces to the data sets and target patterns, so that other unsupervised networks and supervised networks could be exchanged for the SOM and MLP paradigms used individually and together in Hybridnet.

For the unsupervised portion, the author wishes to continue examining the work of Barry Grossberg, specifically the Adaptive Resonant Theory Networks (Carpenter, 1991). In addition to having similar characteristics to the Kohonen Map, they feature a
network adaptation that allows additional nodes to be added if new categories are discovered.

Another neural paradigm to be looked at in the future known as Radial Basis Functions (RBF), relies on a set of basis vectors to define an input space (Zalzala, 1996). To date, the simple gaussian basis vectors are used for the most part. So far, this paradigm has been used successfully in every application the MLP paradigm has been used. For future work in this area, it would also seem appropriate to use wavelets as basis functions for this system. In addition, it intrinsically distributes the decision making into local regions with two element decisions, either “member of set” or “not a member of set”. This is moving to the same approach that the author has seen in improving recognition of large ensemble data sets. Also, the RBF functions address a criticism of MLP networks in that the MLP, though rooted in the biological concept of the neuron, processes and stores the information globally in the network. This portion does not follow from observations of actual biological systems.

**Hardware Implementations**

The advantage of developing a standardized set of primitive behaviors, independent of the specific data stream is that now, a hardware PIC, ASIC, or analog neural network can be created just for that process. And one of the best features of neural networks is the supposed fault tolerance of the system. This is completely lost in the software implementation that the author has used and in fact that most researchers use for working with neural networks. However, if such a hardware implementation is created, then in both vehicle and networked sensor applications, the mission becomes resilient against single point failures.

**Final Remarks**

It has been a privilege to investigate this body of research. The field of neural networks is as vast as the human mind. But perhaps more than any other analytical method, the use of neural networks requires the investigator to carefully define the goal, the steps of the problem, and the heuristics of the problem in order to define a functional solution. As a tool for investigating the ocean, though, the artificial neural network (ANN) will see continued and growing use as more users begin to understand what an ANN is, how to use it, and what its limitations are.
References


Alt, C., and J. Grassle. “LEO-15 An Unmanned Long Term Environmental Observatory”,

An, P.E., A.J. Healey, S.M. Smith, S.E. Dunn.” New Experimental Results on GPS/INS

  Underwater Vehicle AQUA EXPLORER 1000 for Inspection of Underwater Cables.”

Baran, R. H., and J. P. Coughlin. “A Neural Network for Target Classification Using Passive
  198.

Beck, S., L Deuser, R. Still, and J. Whiteley. “A Hybrid Neural Network Classifier of Short
  Duration Acoustic Signals”. Proceedings, IEEE Joint Conference on Neural

  GPS/INS/Sonar Navigation System for Autonomous Underwater Vehicle


active sonar – A form of underwater detection of objects where an acoustic signal is sent out from a transducer and the strength and time of the reflected signal indicates the range and size of the detected object.

Adaptive Resonance Theory – A neural paradigm discovered by Barry Grossberg at Boston University for unsupervised pattern recognition and classification.

artificial neural network (ANN) – An artificial structure, implemented either in hardware or software, which is based on and mimics a biological neural system.

ASIC- acronym “Application Specific Integrated Circuit”; a custom integrated circuit.

backpropagation (BPN) – A method of training a neural network by taking the error at the output of the network and “backpropagating” the error back to the inputs and then adjusting the network structure according to one of a variety of methods.

bit mask – A string of 1’s and 0’s which, when logically “ANDED” with a data vector, selectively leaves data only where the logic 1’s were located.

cetacean – Any of a number of species of whale or dolphin.

cherry pick – to carefully select a sample for only the most desired features.

classifier – A form of neural network designed to sort input data into one of several classes or categories.

Coiflet- A form of wavelet invented by Debauchies at the request of Coifman.

convergence – The process of reducing the training error of a neural network to some minimum value set by the user.

denoising – The process of separating or removing noise from a data sample.

emergent behavior - See emergent feature.

emergent feature – A complex event or pattern made up of a several simple, single dimensional primitive features.
envelope detection – The detection of the general shape or “envelope” of a signal, usually defined in terms of the low frequency energy variation of the signal.

error surface – A three dimensional representation of the average error of a neural network for various combinations of network weights.

Excel® - A spreadsheet program product of Microsoft Corporation used in many branches of science for tabulating data

Fast Fourier transform (FFT) – An algorithm which efficiently implements the equivalent of the Fourier transform for discrete data.

fluorometry – an optical spectroscopy method where light is injected into a sample that either fluoresces directly from the source light or that contains a dye which exhibits large fluorescence coefficients. The wavelength of the fluorescence is monitored and the output is directly proportional to the concentration of the chemical constituent. Usually more sensitive than absorption spectroscopy.

ground truth – A fundamental standardized training and testing set designed to test and compare the performance of one neural network or several. These data sets have both the signal and the noise characterized a-priori in order to better analyze the network.

Haar wavelet – A fundamental wavelet which effectively implements a low pass filter operation.

hidden layer – A layer of neurons not directly connected to inputs or outputs of the network but only to another layer of the network itself.

hybrid network – Any neural network which consists of two or more basic neural network paradigms.

learning rate – A numerical parameter used to scale the amount by which weights are changed during neural network training.

Learning Vector Quantization (LVQ) – A supervised neural paradigm, similar in structure to SOM but instead training by adjusting weights of the network to more closely approach the desired output of the network.

Least mean squares – An error evaluation and neural training method which minimizes the mean square error of a network using a gradient descent approach.

Matlab® – A software program which uses a matrix as its primitive data type to optimize it for matrix and engineering computations.
mean square error – The ensemble error of the network for a given input over all the outputs. Usually defined in terms of the Euclidean norm.

multi-level perceptron (MLP) – a network made up of several layers of fully connected perceptions.

Meyer wavelet – a wavelet useful for modeling time domain signals in the marine environment

neighborhood – The neurons around the neuron with the largest output in a self organizing map network which are rewarded by strengthening the weights during training.

nephelometry – a relative measurement of cloudiness in the water based upon scattering at various angles of an incident light sample. The photodetector is placed at some angle (7-180 degree) relative to the source and the amount of scattered light is measured.

neural network – A network of simpler structures known as neurons, which are highly connected and store information based on the strength of connection between the neurons.

neuron – An adaptable structure that consists of a summing point from several weighted inputs followed by a thresholding element whose output is based on some function of the sum of the inputs.

pc/104® – A hardware standard which implements a compact version of the IBM PC® architecture using stackable connectors. Suitable for portable and low powered systems.

passive sonar – A form of underwater detection that relies on recognizing differences in the ambient sounds in the ocean from different sources.

perceptron – a form of neural network consisting of an array of primitive neurons.

phillic – Having an attraction to or a tendency to combine with another object.

phobic – Having an aversion to or a tendency to be repulsed by another object.

PIC® – A family of microcontrollers from Microchip corporation that contain all the components necessary for embedded sensing elements

primitive feature – A feature which consists of simple changes to single sensor data. Includes maximum, minimum, edge leaving, edge approaching, constant, rising, and falling.
radial basis function – Any orthogonal set of functions which can completely define a data space.

radial basis function network (RBF) – A neural paradigm in which the input data is first processed by a sum of the responses of several radial basis functions to the input pattern, then linearly summed to give an output response.

Self Organizing Map (SOM) – A 2 layer, unsupervised network paradigm designed by Tuevo Kohonen for use in classifying phonemes in the Finnish language.

standalone network – A neural network structure that consists of a single neural paradigm and that functions independently of other neural structures.

supervised – A class of network or learning in which training is performed by presenting an input to a network, then comparing the actual output to some “desired” output and then calculating the error of the computation.

termination condition – The condition set by the network designer at which training is halted. Usually is determined by a certain number of training cycles or by a maximum allowable error criterion.

tow-yo – a towing pattern, used in oceanography, where a towed instrumentation pallete is pulled up and down through various depths while maintaining forward motion. The pattern then looks similar to a sinusoid. Useful for getting multiple depth information during a single pass in the ocean.

turbidity – a measure of the relative cloudiness of the water column.

universal approximator – A neural network that will, after training, have the capacity to represent any form of mathematical function given a set of input points.

unsupervised – A class of network or training method in which training proceeds by strengthening the weights of a neuron with the largest output for a given input.

Vailulu’u – an underwater seamount east of American Samoa that was the subject of expeditions in 2000 and 2001.

wavelet – A waveform family which is localized in space and time and which is useful for modeling complicated waveforms as a sum of wavelets of different time and spatial scales.

whitening – The process of normalizing the variance of a signal.
Appendix B
Software Listings

All software written in Matlab “m” file language, Release 13 version 6.5.1

sigmoid.m
% Function sigmoid.m
% activation function
% Use this functional block to install nonlinear response element
% version 1.0, February 5, 2003
% Brian P. Howell
% Model binary sigmoid function

function output=sigmoid(input)
global alpha %slope parameter

% output=tanh(alpha*input); %bipolar form
output=1./(1+exp(-alpha.*input)); %binary form
% end of sigmoid.m

der_sigmoid.m

% Function der_sigmoid.m
% derivative of activation function
% Use this functional block to install nonlinear response element
% version 1.0, February 5, 2003
% Brian P. Howell
% Model binary sigmoid function

function soutput=der_sigmoid(input)
global alpha % slope parameter

store=sigmoid(input);
%soutput=alpha*(1+store)*(1-store); %bipolar form
soutput=alpha*store.*(1-store); %binary form
% end of der_sigmoid
function s=spgram(inputfile,numfft,numspec,framestart,framesize,offset)

[readfile,freq,bitres]=wavread(inputfile,100);

freq;  % just have these here so we can look at them for troubleshooting
bitres;  % Ditto

sampstart=round(framestart*freq);  % Conversion to samples

sampsize=round(framesize*freq);  % Number of Samples per frame

sampoffset=round(offset*sampsize);  % number of sample offset

filesize=wavread(inputfile,'size');

filesize=filesize(1);

timelength=filesize/freq;

% 'Sample Size is:', timelength;  trap for file overrun
if sampsize<numfft
    'error, to small sample size for fft!'
s=1;
end

% if (sampstart+numspec*sampoffset+sampsize)<filesize  "original"
if (sampstart+(sampsize+((numspec-1)*(sampoffset))))<filesize

% Set up for data collection
s=[];

for j=1:numspec
    sbuff=[];
    readfile=[];
    % First load sample
    startsample=((j-1)*sampoffset+sampstart+1);
    lastsample=((j-1)*sampoffset+sampstart)+sampsize;

    [readfile,freq,bitres]=wavread(inputfile,[startsample,lastsample]);

end
sbuff=(fft(readfile,numfft));    % get fft
sbuff=sbuff.*conj(sbuff)/numfft; % do psd
denom=max(sbuff);                 % get max so we can normalize
if denom==0
    denom=1;
end
sbuff=.95*sbuff/denom;           % go ahead and scale it
s=[s sbuff];                     % add this spectrum onto the stack
end
s;

% Yell real loud if we are overrunning the sample to give bad results!!!
else
    'Too many samples, reduce offset, or numspecs, or sample size!!'
s=1;
end
% end spgram
envelope_thresh.m

% function envelope_thresh
% Brian P. Howell
% September 15, 2003
% Rev 1.5
% works, try average of 100 points and .01 threshold

% infile is the name of the input file either a matlab variable or the name
% of a generic disk file if in quotes

% numpoints is the number of points for averaging to create the envelope

% thresh is the scaled threshold value used for setting the start and
% stop points

function output=envelope_thresh(infile,numpoints,thresh)

index=size(infile);  % get how big the file is
index=index(1)-numpoints;  % now chop off the average amount
e=infile.*infile;         % square the amplitudes so we get something that looks like power
e=e/max(e);               % now ratio it
for k=1:index
    mvalue=mean(e(k:(k+numpoints)));  % now create a sliding average and check
                                       % against the thresh value
    if mvalue>thresh
        output(k)=1;                      % if its bigger, store a 1
    else
        output(k)=0;                      % if its smaller, store a zero
    end
end
output;                              % Report back an array that has ones wherever there is a pulse

% End of envelope_thresh

fish_whackfft.m

% Trial at autofish processor
% Brian Howell
% Rev 2.0
% October 10, 2003

% this function takes the signal, gathers the pulses, and performs fft and
% cwt (continuous wavelet decomposition) on the collected pulses
function [numpulse, outputpulses, spec, wave] = fish_whackfft(signal, avg, threshold, fftspec)
index = size(signal);
index = index(1); % signal comes in as a column vector

t = envelope_thresh(signal, avg, threshold); % make a bit mask
if t(1) > 0
    t = diff(t); % differentiate the bit mask to make start and stop flags
    t(1) = -1;
    t(index) = -1;
else
    t = diff(t);
end

start_stop = find(t) % find the nonzero derivative locations, that's your
% pulse edges for getting 'em

numpulse = size(start_stop);
numpulse = numpulse(2);

% If you want to trap for full sample window, do it here
% Two possibilities: 1, no signal, in which case return nada or return
% whole signal

numpulse = numpulse/2;
numchk = fix(numpulse);
if numpulse - numchk ~= 0
    if t(1) < 0
        start_stop = [1 start_stop];
    else
        start_stop = [start_stop index];
    end
end

% this go round, we used the continuous wavelet transform, with a coifl3 6
% layer decomposition
end

numpulse=numchk+1;
end

width=zeros(numpulse,1);

% now we have to make the sample files big enough for the biggest pulse

for j=1:numpulse
    % First get max pulse width
    width(j)=start_stop(2*j)-start_stop((2*j)-1);
end

buffsize=max(width);
outputpulses=zeros(buffsize,numpulse);

% gather up the pulses

for j=1:numpulse
    outputpulses(1:(1+(start_stop(2*j)-start_stop((2*j)-1))),j)=
        signal((start_stop((2*j)-1)):start_stop(2*j));
end

% generate FFT coefficient file fftspec resolution and returning a
% power spectral density representation of the signal

for k=1:numpulse
    spec(:,k)=fft(outputpulses(:,k),fftspec);
    spec(:,k)=spec(:,k).*conj(spec(:,k))/fftspec;
end

% generate wavelet coefficient file using coiflet level3 6 level
% decomposition

for k=1:numpulse
    [wave(:,:,k)]=cwt((outputpulses(:,k))',1:6,'coif3');
end

% end fish_whack_fft.m
SOM1D.m  version 4.0
(note: this version is the version that requires the signal to be processed into a 1d spectrum externally, the SOM2D file version next shows the version with internal processing. Either file can use either form)

% Starting over Self organizing Map 1d
% Brian Howell Rev4
% January 23, 2004
% this is not written as a function but as an executable m file
% This version assumes that files are processed externally
% in this case the following holds true:
% xspec_train is the input containing a series of 1-d spectrograms for
% training
% xspec_test is the input containing a series of 1-d spectrograms for
% testing
clc
x=[];

clear som_train; %these are our variables for internal use
clear som_test;
dimen_input=size(xspec_train); % use this to autoscale the internal parts
dftspec=dimen_input(1);
outepoch_train=dimen_input(2); % Number of samples per epoch
som_train=zeros(size(xspec_train));

% now get the data into our .1-.8 span with the zero level at .1

maxi=max(xspec_train);
mini=min(xspec_train);
delta=(maxi - mini);
for j=1:outepoch_train
som_train(:,j)=.8*((xspec_train(:,j) - mini(j))/delta(j))+.1;
end

mu=.999; % Learning Rate
R=2; % Neighborhood
m_SOM=2; % Number of output categories for map to sort into

iteration=20000; % Number of loops to termination
neighcount=100; % number of loops until neighborhood = 0

% now make the index files
mindex=[]; % output category index
spindex=[]; % spectrum index

for j=1:m_SOM
    mindex=[mindex j];
end

for k=1:dftspec
    spindex=[spindex k];
end
' Starting Run....'

dimen=size(xspec_train);

w=.01*randn(dimen(1),m_SOM); % Initialize Weight vectors

d=zeros(m_SOM); % Output Vector
errortrend=[];
for j=1:iteration
    pointcase=mod(j,outepoch_train)+1; % Sets the point for indexing into the input files
    accum=[];
    xtest=som_train(:,pointcase); % Pull a spectrogram out
    if iteration > neighcount % Check for neighborhood function
        R=0;
    end
    for c=1:m_SOM % Calculate all the weight errors
        error=(xtest-w(:,c))*(xtest-w(:,c));
        accum=[accum norm(error)]; % And Accumulate Them
    end

    accum;

    [smallerror,J]=min(accum);

    % Now find a neighborhood around J
    % Reinforce all weights in neighborhood J-R-J+R,
    Jmin=J-R; % Keep the indices straight if minimum or maximum
    if Jmin<1
        Jmin=J;
    end
    Jmax=J+R;
    if Jmax>m_SOM
        Jmax=m_SOM;

    end

end

% update the weights
for L=Jmin:Jmax
    w(:,L)=w(:,L)+mu*(xtest-w(:,L));
end
errortrend=[errortrend smallerror];
mu=mu*.9999;
end

% Added to look at actual SOM output!
% this way we get the biggest output (not minimum error)
for z=1:outepoch_train
    y=zeros([1,m_SOM]);
    for i=1:m_SOM
        for j=1:dftspec
            y(i)=y(i)+som_train(j,z)*w(j,i);
        end
    end
    y;
end

'Completed Training'
' Error trend at end of training'
(errortrend(iteration)+errortrend(iteration-1)+errortrend(iteration-2)+errortrend(iteration-3))/4

'Begin Testing'
' Now looking at the behavior of the network'
figure(1)
plot(errortrend)

'Now test with same data'
% This module gives the selected categories of the network
ymatrix=[];
for j=1:outepoch_train
    accum=[];
    xtest=som_train(:,j);
for k=1:m_SOM
    error=(xtest-w(:,k))'*(xtest-w(:,k));
    accum=[accum norm(error)]; % And Accumulate Them
end
accum;
[smallerror1,cat]=min(accum);

ymatrix(1,j)=j;
ymatrix(2,j)=cat;
end

ymatrix % Report the initial mapped categories

'Now try from different parts of the data sample'
dimen_input=size(xspec_test);
dftspec=dimen_input(1);
outepoch_test=dimen_input(2); % Number of samples per epoch

% scale the testing data to the same .1-.8 span zero at .1 offset
maxi=max(xspec_test);
mini=min(xspec_test);
delta=(maxi-mini);
for j=1:outepoch_test;
som_test(:,j)=.8*((xspec_test(:,j)-mini(j))/delta(j))+.1;
end

calcmatrix=zeros(1,outepoch_test);
% this is now where we report the selected category for each test data
% sample
for m=1:outepoch_test
    accum=[];
    xtest=som_test(:,m);
    for k=1:m_SOM
        error=(xtest-w(:,k))'*(xtest-w(:,k));
        accum=[accum norm(error)]; % And Accumulate Them
    end
    [smallerror,cat]=min(accum);
calcmatrix(m)=cat;
cat;
smallerror;
end
end
calcmatrix  % report mapped categories
ymatrix     % report original mappings of training set

SOM2D.m  version 4.1
(note: this version is the the version with internal processing, the SOM1D file
version previously shows the version that requires the signal to be processed into a
1d spectrum externally. Either file can use either form)

% Starting over Self organizing Map 2d
% Brian Howell Rev2
% this is not written as a function but as an executable m file
% Real Soundfiles, uses data set 2
clc
[a,b]=size(ds2);
x=[];
timeslice=.1;  % Time in seconds for frame
m_SOM=15;  % Number of output categories for map to sort into
outepoch=a;  % Number of samples per epoch
numspecs=10;
mindex=[];

num_traintrials=3;
um_testtrials=3;

trainoffset=.1;  % Training Data Starting Point (sec)
testoffset=3;  % Index for sliding spectral frame (sec)
frameoffset=.25  % frame overlap, 0 = 100% overlap, 1= sequential (no overlap)
dftspecc=512;
mu=.999;  % Learning Rate
R=2;  % Neighborhood

iteration=10000;  % Number of loops
neighcount=100;
spindex=[];
for j=1:m_SOM
    mindex=[mindex j];
end

for k=1:dftspec
    spindex=[spindex k];
end

% xspec_train is the input containing a series of 2-d spectrograms for training
% xspec_test is the input containing a series of 2-d spectrograms for testing

% create input spectrogram
% dimension 1 is the number of fft points
% dimension 2 is the number of spectra
% dimension 3 is the number of samples in an epoch

xspec_train=zeros(dftspec,numspects,ouepoch*num_traintrials);
for j=1:ouepoch
    for k=1:num_traintrials
        scan_offset=trainoffset+(k-1)*(timeslice+timeslice*frameoffset*(numspects-1))
        xspec_train(:,:,((j*num_traintrials) - (num_traintrials - k)))=spgram(ds2{j},dftspec,numspects,scan_offset,timeslice,frameoffset);
    end
end

'Completed Sample Prep'

' Starting Run....'

dimen=size(xspec_train)
w=.05*randn(dimen(1),dimen(2),m_SOM); % Initialize Weight vectors
d=zeros(m_SOM); % Output Vector
errortrend=[];
for j=1:iteration
    pointcase=mod(j,ouepoch)+1; % Sets the point for indexing into the input files
    accum=[];
    xtest=xspec_train(:,:,pointcase); % Pull a spectrogram out
    if iteration > neighcount % Check for neighborhood function

R=0;
end

for c=1:m_SOM  
    % Calculate all the weight errors
    error=(xtest-w(:,:,c))'*(xtest-w(:,:,c));
    accum= [accum norm(error)];  
    % And Accumulate Them
end
accum;

[smallerror,J]=min(accum);

% Now find a neighborhood around J
% Reinforce all weights in neighborhood J-R-J+R,
Jmin=J-R;  
% Keep the indices straigt if minimum or maximum
if Jmin<1
    Jmin=J;
end
Jmax=J+R;
if Jmax>m_SOM
    Jmax=m_SOM;
end
for L=Jmin:Jmax
    w(:,:,L)=w(:,:,L)+mu*(xtest - w(:,:,L));
end
errortrend=[errortrend smallerror];
mu=mu*.999;
end

% Added to look at actual SOM output!
for z=1:outepoch*num_traintrials
    y=zeros([1,m_SOM]);
    for i=1:m_SOM
        for j=1:dftspec
            for k=1:numspecs
                y(i)=y(i)+xspec_train(j,k,z)*w(j,k,i);
            end
        end
    end
end
y;
'Completed Training'
' Error trend at end of training'
(errortrend(iteration)+errortrend(iteration-1)+errortrend(iteration-2)+errortrend(iteration-
3))/4

'Begin Testing'
' Now looking at the behavior of the network'
figure(1)
plot(errortrend)

'Now test with same data'
% This module gives the selected categories of the network
ymatrix=[];
for j=1:outepoch*num_testtrials
    accum=[];
    xtest=xspec_train(:,:,j);
    for k=1:m_SOM
        error=(xtest-w(:,:,k))*(xtest-w(:,:,k));
        accum=[accum norm(error)]; % And Accumulate Them
    end
    accum;
    [smallerror1,cat]=min(accum);
    ymatrix(1,j)=j;
    ymatrix(2,j)=cat;
end
ymatrix

'Now try from different parts of the data sample'

xspec_test=[];
calcmatrix=zeros(outepoch,num_testtrials);
for j=1:a
    for m=1:num_testtrials
        scan_offset=testoffset+(m-1)*(timeslice+timeslice*frameoffset*(numspects-1));
        xspec_test(:,:,((j*num_traintrials)-(num_traintrials-
m)))=spgram(ds2{j},dftspec,numspects,scan_offset,timeslice,frameoffset);
    end
for m=1:num_testtrials
    accum=[];
    xtest=xspec_test(:,:,((j* num_traintrials)-(num_traintrials-m)));

    for k=1:m_SOM % Calculate all the weight errors
        error=(xtest-w(:,:,k))'*(xtest-w(:,:,k));
        accum=[accum norm(error)] ; % And Accumulate Them
    end % end SOM
    [smallerror,cat]=min(accum);
    calcmatrix(j,m)=cat;
    j;
    cat;
    smallerror;
end
end

calcmatrix

ymatrix

% end SOM2D

BPN3_5.m listing with history experiment

% all scalar neural network
% Rev 3.1 October 24, 2003
% Generic Back prop using external files
% Needs workspace (xspec_train( rows are fft, columns are samples))
% xspec_test(ditto but for testing)
% target for desired outputs each column gives desired category for that
% sample
% This has no embellishments to the algorithm save Nguyen Widrow
% initialization
% Now thats all we need
% Here we input the network parameters
% xspec_train is the externally processed training data
% train_history is the generated primitive feature selected for the last
% time index
% xspec_test is the externally processed test data
% test_history is the generated primitive feature for the last time index
% normalize input data

number_outputs=6; % set this and in functional form, pass as argument
[dummy, outepoch]=size(train_targets) % Get structure of output maps
bpn_train=zeros(size(xspec_train));
offsets=zeros(size(xspec_train));
% alternate try
means=mean(xspec_train);
for j=1:10
offsets(j,:)=xspec_train(j,:)-means; if you want to center to small signals at .5 leave this in
end

% set span and offset
maxi=max(offsets);
mini=min(offsets);
delta=(maxi - mini);
delta=max(delta)*ones(1, outepoch); % This causes inputs to be scaled globally (one max)
for j=1:outepoch
bpn_train(:,j)=(.8*((offsets(:,j) - mini(j))/delta(j)))+.1;
end

% add on the history information
bpn_train=[bpn_train; train_history]
% End Normalization

output=zeros(number_outputs, outepoch)+.1;

for j=1:outepoch
    output(train_targets(j), j)=1;
end
output=.9*output;
output;

[n0, numsamp]=size(bpn_train) % This way, we automatically adjust

n1=n0; % L1 neurons
n2=n0; % L2 neurons
[n3, epoch]=size(output) % this way, we automatically adjust for the file input later
% now initialize weights
%well globalize all these so we don't need to pass them around

global w1
global w2
global w3
global mu0
global alpha

mu0=.90     % initial learning rate
alpha=.5  % sharpness of sigmoid
countlimit=200000;  % sample cycles, divide by outepoch to get number of epochs

w1=.5*randn(n1,n0);   % input neurons
w2=.5*randn(n2,n1);   % hidden layer neurons
w3=.5*randn(n3,n2);   % output neurons

% Nguyen and Widrow Scaling Factor

gamma1=.7*n1^(1/n0);
gamma2=.7*n2^(1/n1);
gamma3=.7*n3^(1/n2);

% For W1 initialization
for j=1:n0
    dist=0;
    for i=1:n1
        dist=dist+w1(i,j)*w1(i,j);
        end
    end
    for i=1:n1
        w1(i,j)=gamma1*w1(i,j)/dist;
    end
end

% For W2 initialization
for j=1:n1
    dist=0;
    for i=1:n2
        dist=dist+w2(i,j)*w2(i,j);
        end
    end
    for i=1:n2
        w2(i,j)=gamma2*w2(i,j)/dist;
    end
end
% For W3 initialization
for j=1:n2
    dist=0;
    for i=1:n3
        dist=dist+w3(i,j)*w3(i,j);
    end
    for i=1:n3
        w3(i,j)=gamma3*w3(i,j)/dist;
    end
end

count=0;
errorplot=[];
%do feedforward

while count<countlimit
    mu=mu0*.9999;
    %s=mod(count,numsamp)+1; %sequential index
    s=1+fix(rand(1)*(numsamp-1)); % Random index
    xtemp=0;
    % xout1
    for j=1:n1
        xtemp=0;
        for i=1:n0
            xtemp=xtemp+w1(j,i)*bpn_train(i,s);
        end
        v1(j)=xtemp; % get our summation
    end
    xout1=sigmoid(v1); % do the nonlinear thing

    % xout2
    for j=1:n2
        xtemp=0;
        for i=1:n1
            xtemp=xtemp+w2(j,i)*xout1(i);
        end
        v2(j)=xtemp; % get our summation
    end
    xout2=sigmoid(v2); % do the nonlinear thing
end
% xout3
for j=1:n3
    xtemp=0;
    for i=1:n2
        xtemp=xtemp+w3(j,i)*xout2(i);
    end
    v3(j)=xtemp; % get our summation
end
xout3=sigmoid(v3); % do the nonlinear thing

% end of feedforward

% now training

% layer3
for j=1:n3
    delta3(j)=(output(j,s)-xout3(j))*der_sigmoid(v3(j));
end

% layer 2
for j=1:n2
    deltatemp=0;
    for h=1:n3
        deltatemp=deltatemp+delta3(h)*w3(h,j);
    end
    delta2(j)=deltatemp*der_sigmoid(v2(j));
end

% layer 1
for j=1:n1
    deltatemp=0;
    for h=1:n2
        deltatemp=deltatemp+delta2(h)*w2(h,j);
    end
    delta1(j)=deltatemp*der_sigmoid(v1(j));
end

% Now update weights
for j=1:n3
    for i=1:n2
        w3(j,i)=w3(j,i)+mu*delta3(j)*xout2(i);
    end
for j=1:n2
    for i=1:n1
        w2(j,i)=w2(j,i)+mu*delta2(j)*xout1(i);
    end
end

for j=1:n1
    for i=1:n0
        w1(j,i)=w1(j,i)+mu*delta1(j)*bpn_train(i,s);
    end
end

% Figure out our errors
error=0;
for i=1:n3;
    error=error+(output(i,s)-xout3(i))^2;
error=.5*error;
end

errorplot=[errorplot error];
count=count+1;
end
subplot(1,1,1)
plot(errorplot)

% Now test the data
% First with the original data

for s=1:numsamp
    for j=1:n1
        xtemp=0;
        for i=1:n0
            xtemp=xtemp+w1(j,i)*bpn_train(i,s);
        end
        delta1(j)=der_sigmoid(xtemp);
        xout1(j)=sigmoid(xtemp);
    end
    %xout2
    for j=1:n2
xtemp=0;
for i=1:n1
    xtemp=xtemp+w2(j,i)*xout1(i);
end
delta2(j)=der_sigmoid(xtemp);
xout2(j)=sigmoid(xtemp);
end

%xout3

for j=1:n3
    xtemp=0;
    for i=1:n2
        xtemp=xtemp+w3(j,i)*xout2(i);
    end
    delta3(j)=der_sigmoid(xtemp);
xout3(j,s)=sigmoid(xtemp);
end
end

' Final Results with original Data'
train_targets(1:outepoch)
xout3(:,1:outepoch)
% NOw load test data set
% Feed forward with fixed weights
% Normalization

[dummy,outepoch]=size(xspec_test) % Get structure of output maps
bpn_test=zeros(size(xspec_test));
offsets=zeros(size(xspec_test));
%alternate try
means=mean(xspec_test)
for j=1:10
    offsets(j,:)=xspec_test(j,:)% - means;
end
maxi=max(offsets);
mini=min(offsets);
delta=(maxi-mini);
delta=max(delta)*ones(1,outepoch);
for j=1:outepoch
    bpn_test(:,j)=.8*((offsets(:,j) - mini(j))/delta(j))+.1;
end
bpn_test=[bpn_test;test_history]
% End normalization

for s=1:numsamp
for j=1:n1
    xtemp=0;
    for i=1:n0
        xtemp=xtemp+w1(j,i)*bpn_test(i,s);
    end
    delta1(j)=der_sigmoid(xtemp);
    xout1(j)=sigmoid(xtemp);
end

%xout2
for j=1:n2
    xtemp=0;
    for i=1:n1
        xtemp=xtemp+w2(j,i)*xout1(i);
    end
    delta2(j)=der_sigmoid(xtemp);
    xout2(j)=sigmoid(xtemp);
end

%xout3
for j=1:n3
    xtemp=0;
    for i=1:n2
        xtemp=xtemp+w3(j,i)*xout2(i);
    end
    delta3(j)=der_sigmoid(xtemp);
    xout3(j,s)=sigmoid(xtemp);
end
end

test_targets(1:outepoch) % what were we supposed to get
xout3(:,1:outepoch) % what did we really get

%plot some stuff so we can see how screwed up it
%all is
figure(2)
subplot(1,2,1)
hold on
for j=1:outepoch
    plot(bpn_train(:,j))
end
hold off
subplot(1,2,2)
hold on
for j=1:outepoch
    plot(bpn_test(:,j))
end
hold off
HYBRIDNET_4_1.m

% Hybrid neural network, now adding enhancements to back prop
% Brian Howell Rev4
% January 9, 2004
% enhancement 1 Nguyen and Widrow Intitialization algorithm

% Hybridnet 4 decouples training of SOM and Backprop

% SOM Structure and training Structure

% This version of the program assumes all processing has been done external
% to the program it looks for the following variables in the workspace:

% xspec_train is the 2d input spectra stack for training
% xspec_test is the 2d input spectra stack for testing

% train_targets are the correct categories for training
% just put the category numbers into a column vector
% this program converts that vector into a matrix scaled for the nn.

% test_targets are the correct categories for testing
% just put the category numbers into a column vector
% this program converts that vector into a matrix scaled for the nn.

clc
x=[];

countlimit=5000; % Number of loops for som training
countlimit_bpn=20000; % number of loops for bpn training

dimen_input=size(xspec_train);
dftspec=dimen_input(1); % number of points for fft
numspecs=dimen_input(2); % number of spectra in lofagram
outepoch=dimen_input(3); % Number of samples per epoch

m_SOM=20; % Number of output categories for map to sort into
m_bpn=15; % number of categories at output of MLP

% This transfers the categories in the output to 1's and 0's form
numcategory_train=zeros(m_bpn,outepoch)+.1;
for j=1:outepoch
    numcategory_train(train_targets(j),j)=.9;
numcategory_train

% end of fixing outputs

numtrials=10;  %For SOM Testing (we'll delete for this run but save )
mu_SOM=.99;  % Learning Rate SOM
R=1;   % Neighborhood
neighcount=75;  % number of loops where we drop the neighborhood distance to zero

% First Create Parameters from SOM for BPN
n0=m_SOM;
n1=m_SOM;
n2=m_SOM;
n3=m_bpn;
global w1_bpn;
global w2_bpn;
global w3_bpn;
global mu0_bp;
global alpha;
w1_bpn=.5*rand(n1,n0);
w2_bpn=.5*rand(n2,n1);
w3_bpn=.5*rand(n3,n2);
mu0_bp=.5;  % initial mlp learning rate
alpha=.8;  % sharpness of mlp neuron sigmoid

%Nguyen and Widrow Scaling Factor

gamma1=.7*n1^(1/n0);
gamma2=.7*n2^(1/n1);
gamma3=.7*n3^(1/n2);

% For W1 initialization
for j=1:n0
    dist=0;
    for i=1:n1
        dist=dist+w1_bpn(i,j)*w1_bpn(i,j);
    end
    for i=1:n1
        w1_bpn(i,j)=gamma1*w1_bpn(i,j)/dist;
    end
end
end

w1_bpn;

% For W2 initialization
for j=1:n1
    dist=0;
    for i=1:n2
        dist=dist+w2_bpn(i,j)*w2_bpn(i,j);
    end
    for i=1:n2
        w2_bpn(i,j)=gamma2*w2_bpn(i,j)/dist;
    end
end

w2_bpn;

% For W3 initialization
for j=1:n2
    dist=0;
    for i=1:n3
        dist=dist+w3_bpn(i,j)*w3_bpn(i,j);
    end
    for i=1:n3
        w3_bpn(i,j)=gamma3*w3_bpn(i,j)/dist;
    end
end

w3_bpn;

errorplot_bp=[];
errortrend_SOM=[];

%%% This portion just makes the indices for plotting data
spec_index=[];
m_SOMindex=[];
for j=1:m_SOM
    m_SOMindex=[m_SOMindex j];
end

for k=1:dftspec
    spec_index=[spec_index k];
end
% End MLP Structure and Training Structure

numcategory=numcategory_train;
xspec=xspec_train;

'Completed Sample Prep'

' Starting Run....'

% This version of the program, we cascade the data through,
% then calculate the SOM w change and backprop sequentially, then run
%the next data

w_SOM=.05*randn(dftspec,numspects,m_SOM); % Initialize Weight vectors
d=zeros(m_SOM); % Output Vector

for train_index=1:countlimit
    mu=mu0_bp*.99999;
    pointcase=mod(train_index,outepoch)+1; % Sets the point for indexing into the input files
    accum=[];
    xtest=xspec(:,:,pointcase); % Pull a spectrogram out
    if countlimit > neighcount % Check for neighborhood function
        R=0;
    end
    for c=1:m_SOM
        % Calculate all the weight errors
        error=(xtest-w_SOM(:,:,c)).*(xtest-w_SOM(:,:,c));
        accum= [accum norm(error)]; % And Accumulate Them
    end
    [smallerror,J]=min(accum);

% Added to look at actual SOM output!
y_SOM=zeros([1,m_SOM]);

for i=1:m_SOM
    for j=1:dftspec
        for k=1:numspecs
            y_SOM(i)=y_SOM(i)+xspec(j,k,pointcase)*w_SOM(j,k,i);
        end
    end
end

% Now find a neighborhood around J
% Reinforce all weights in neighborhood J-R-J+R,
Jmin=J-R;
if Jmin<1
    Jmin=J;
end
Jmax=J+R;
if Jmax>m_SOM
    Jmax=m_SOM;
end
for L=Jmin:Jmax
    w_SOM(:,:,L)=w_SOM(:,:,L)+mu_SOM*(xtest - w_SOM(:,:,L));
end
errortrend_SOM=[errortrend_SOM smallerror];
mu_SOM=mu_SOM*.999;
end

%%% Ending SOM Training %%%%%%%%

' End of SOM Training'
'SOM Epoch Error trend at end of training'
errorcounter=0;
for i=1:outepoch
    errorcounter=errorcounter+errortrend_SOM(countlimit+1-i);
end
errorcounter=errorcounter/outepoch

%%% Beginning BPN Training %%%%%%%

% Begin BPN Training
% Feedforward for Backprop
countlimit=countlimit_bpn;
for train_index=1:countlimit

mu=mu0_bp*.99999;

pointcase=mod(train_index,outepoch)+1; % Sets the point for indexing into the input files
% First, for each case, preprocess using SOM

y_SOM=zeros([1,m_SOM]);
for i=1:m_SOM
  for j=1:dftspec
    for k=1:nums specs
      y_SOM(i)=y_SOM(i)+xspec(j,k,pointcase)*w_SOM(j,k,i);
    end
  end
end

% Now Scale Output
if max(y_SOM)~=0
  % Now To Train Backprop, we feedforward SOM Results on trained Network
  %y_SOM=.9*(y_SOM/(max(y_SOM)))-.5; %BipOLAR VERSION
  y_SOM=.9*(y_SOM/max(y_SOM)); %binary version %Very important to scale output
else
  y_SOM=.9*y_SOM;
end

% Backprop feedforward section
xtemp=0;
% xout1
for j=1:n1
  xtemp=0;
  for i=1:n0
    xtemp=xtemp+w1_bpn(j,i)*y_SOM(i);
  end
  v1(j)=xtemp;
end
xout1=sigmoid_hyb(v1);
%xout2
for j=1:n2
    xtemp=0;
    for i=1:n1
        xtemp=xtemp+w2_bpn(j,i)*xout1(i);
    end
    v2(j)=xtemp;
end
xout2=sigmoid_hyb(v2);

%xout3
for j=1:n3
    xtemp=0;
    for i=1:n2
        xtemp=xtemp+w3_bpn(j,i)*xout2(i);
    end
    v3(j)=xtemp;
end
xout3=sigmoid_hyb(v3);

%end of feed forward

%Now Calculate weight updates for Backprop
%layer3
for j=1:n3
    delta3(j)=(numcategory(j,pointcase)-xout3(j))*der_sigmoid_hyb(v3(j));
end

% layer 2
for j=1:n2
    deltatem=0;
    for h=1:n3
        deltatem=deltatem+delta3(h)*w3_bpn(h,j);
    end
    delta2(j)=deltatem*der_sigmoid_hyb(v2(j));
end

%layer 1
for j=1:n1
    deltatem=0;
    for h=1:n2
        deltatem=deltatem+delta2(h)*w2_bpn(h,j);
    end
delta1(j)=deltatemp*der_sigmoid_hyb(v1(j));
end

% Now update weights

for j=1:n3
    for i=1:n2
        w3_bpn(j,i)=w3_bpn(j,i)+mu*delta3(j)*xout2(i);
    end
end

for j=1:n2
    for i=1:n1
        w2_bpn(j,i)=w2_bpn(j,i)+mu*delta2(j)*xout1(i);
    end
end

for j=1:n1
    for i=1:n0
        w1_bpn(j,i)=w1_bpn(j,i)+mu*delta1(j)*y_SOM(i);
    end
end

error_bp=0;

for i=1:n3;
    error_bp=error_bp+(numcategory(i,pointcase)-xout3(i))^2;
    error_bp=.5*error_bp;
end

errorplot_bp=[errorplot_bp error_bp];

end

'Completed Training'

' Now looking at the behavior of the network'
figure(2)
subplot(1,2,1);
plot(errortrend_SOM)
title('SOM error')
subplot(1,2,2);
plot(errorplot_bp)
title('bpn error')

% End of Training Portion Now Leave Weights Constant
% Read New test data set

dimen_input=size(xspec_test);
dfts(()=>dimen_input(1);
numspecs=dimen_input(2);
outepoch=dimen_input(3); % Number of samples per epoch

mapping=zeros(outepoch,2); %storage spot for results
%BPN Structure and Training Structure
%add bipolar inputs
%numcategory=numcategory-1;

xscep=xspec_test;

for z=1:outepoch
    accum=[];
    xtest=xspec(:,:,z);
    for k=1:m_SOM
        error=(xtest-w_SOM(:,:,k))'*(xtest-w_SOM(:,:,k));
        accum=[accum norm(error)]; % And Accumulate Them
    end
    [smallerror1,cat]=min(accum);
    ymatrix(1,z)=z;
    ymatrix(2,z)=cat;

    % This determines Final SOM mapping
end

% And Translates it to a positive output

'Testing Results'
% go get the MLP results for everything in the testing file
for z=1:outepoch
    y_SOM=zeros([1,m_SOM]);
    for i=1:m_SOM
        for j=1:dfts(()=>
            for k=1:numslocals
                y_SOM(i)=y_SOM(i)+xspec(j,k,z)*w_SOM(j,k,i);
            end
        end
    end
end
end
end
end
'cat', z

% y_SOM=.9*(y_SOM/(max(y_SOM)))-.5;  % Bipolar VERSION
y_SOM=.8*(y_SOM/(max(y_SOM)));  % Binary VERSION

% Feedforward for MLP
xtemp=0;
% xout1
for bpindex=1:n1
    xtemp=0;
    for i=1:n0
        xtemp=xtemp+w1_bpn(bpindex,i)*y_SOM(i);
    end
    v1(bpindex)=xtemp;
end
xout1=sigmoid_hyb(v1);

% xout2
for bpindex=1:n2
    xtemp=0;
    for i=1:n1
        xtemp=xtemp+w2_bpn(bpindex,i)*xout1(i);
    end
    v2(bpindex)=xtemp;
end
xout2=sigmoid_hyb(v2);

% xout3
for bpindex=1:n3
    xtemp=0;
    for i=1:n2
        xtemp=xtemp+w3_bpn(bpindex,i)*xout2(i);
    end
    v3(bpindex)=xtemp;
end
xout3=sigmoid_hyb(v3);

% end of feedforward
[smallerror1,cat]=min(accum);
'category', z
y_SOM
xout3  % what did the Hybridnet decide for this sample
% now that we have xout3, report whether the max value is legit or is it
% really still confused
[maxvalue,catnum]=max(xout3);
if maxvalue<.5
    'Indeterminate'
end
'Category selected:',catnum
'Max value',maxvalue

mapping(z,1)=catnum;  % summary data for short answers
mapping(z,2)=maxvalue;% summary data for short answers
end

ymatrix  % just to compare lets see what the SOM looked like