Verification and validation of neural networks: a sampling of research in progress

Brian Taylor[*a], Marjorie Darrah[†a], Christina Moats[‡b]
[a]Institute for Scientific Research, Inc., Fairmont, WV 26554
[b]NASA IV&V Facility, Fairmont, WV 26554

ABSTRACT

Neural networks represent a class of systems that do not fit into the current paradigms of software development and certification. Instead of being programmed, a learning algorithm "teaches" a neural network using a set of data. Often, because of the non-deterministic result of the adaptation, the neural network is considered a "black box" and its response may not be predictable. Testing the neural network with similar data as that used in the training set is one of the few methods used to verify that the network has adequately learned the input domain.

In most instances, such traditional testing techniques prove adequate for the acceptance of a neural network system. However, in more complex, safety- and mission-critical systems, the standard neural network training-testing approach is not able to provide a reliable method for their certification. Verifying correct operation of neural networks within NASA projects, such as autonomous mission control agents and adaptive flight controllers, and within nuclear engineering applications, such as safety assessors and reactor controllers, requires as rigorous an approach as those applied to common programming techniques.

This verification and validation (V&V) challenge is further compounded by adaptive neural network systems; ones that modify themselves, or "learn," during operation. These systems continue to evolve during operation, for better or for worse. Traditional software assurance methods fail to account for systems that change after deployment.

Several experimental neural network V&V approaches are beginning to emerge, but no single approach has established itself as a dominant technique. This paper describes several of these current trends and assesses their compatibility with traditional V&V techniques.

Keywords: verification and validation, neural networks, software certification

1. INTRODUCTION

Neural networks are members of a class of software that has the potential to simulate characteristics of biological thinking and learning. Owing their origins to the study of the human brain, neural networks possess the ability to acquire and store knowledge. They are well suited for domains of non-linearity and high complexity that are ill defined, unknown, or just too complex for standard programming practices. Rather than explicit programming, learning algorithms are used to adjust neural networks to fulfill the needs of a desired function.

A real-time adaptable flight control system is a good example of a system that benefits from neural networks. The Intelligent Flight Control System (IFCS) is one such example being developed as a collaborative effort among the NASA Dryden Flight Research Center (DFRC), the NASA Ames Research Center (ARC), Boeing Phantom Works, the Institute for Scientific Research, Inc. (ISR), and West Virginia University (WVU). The IFCS goal is to develop and demonstrate a flight control concept that can efficiently identify aircraft stability and control derivatives using neural networks and utilize this information to optimize aircraft performance. Included in the IFCS are two neural networks; a fixed neural network that has been trained offline and a dynamic neural network that learns while the aircraft is in operation.

[*] btaylor@isr.us; phone 1.304.368.9300; fax 1.304.534.4106; www.isr.us
[†] mdarrah@isr.us; phone 1.304.368.9300; fax 1.304.363.4597
[‡] Christina.D.Moats@nasa.gov; phone 1.304.367.8340; fax 1.304.367.8203; www.ivv.nasa.gov

As technology enables aircraft and spacecraft to perform increasingly complex missions, maintaining control of the crafts becomes comparably more difficult. Consequently, the next generation of flight control systems will utilize adaptive and non-deterministic techniques to provide for more stable and maneuverable aircraft. Neural networks will play a progressively more important role in such systems since they can adapt in real-time to untested flight conditions including aircraft failures.

Developers of neural networks have been cautious about expanding their use into safety-critical domains due to the complexities and uncertainties associated with these non-deterministic software systems. This raises a concern for everyone from the project managers and systems engineers, who are responsible for developing these systems, to the customers and users, who must place their trust in these systems: How can we be sure that any system that includes neural network technology is going to behave in a known, consistent and correct manner?

Of particular concern is the trustworthiness and acceptability of dynamic neural networks that continue to adapt or evolve after the system is deployed. While some online learning neural networks (OLNN) may be given *a priori* knowledge of their input domain, the adaptation that they undergo offers no guarantee that the system is stable or continues to meet the original objectives.

An increased interest on the part of NASA has encouraged research in the area of V&V of adaptive systems over the past few years, but in general, the study of V&V of neural networks has been limited. Universities, government agencies, and a small number of companies are working on different aspects of this problem, but no single unifying standard or process has been established to help those who are developing neural networks.

As the facility responsible for improving software safety, reliability, and quality of programs and missions, the NASA Independent Verification & Validation (IV&V) Facility will be increasingly challenged to certify and evaluate software systems that contain neural network technologies. To prepare for these challenges, the NASA IV&V Facility and the ISR are developing a methodology for the IV&V of neural networks. The contents of this paper are the results of a literature survey conducted during the first year of this three-and-a-half year effort.

Because of the emerging nature of V&V techniques for neural networks, the amount of published material relating solely to the methods and tools used for V&V of neural networks is small. Methods that have potential usefulness for V&V of neural network are presented, including improved testing, run-time monitoring, formal methods, cross validation, and visualization.

2. SAFETY-CRITICAL CONCERNS

In assessing a safety-critical neural network system, a V&V expert must know what to look for and how to analyze the information. Many questions face the analyst regarding a neural network:

- Has the network learned the correct data, or has it learned something else that correlates closely to the data?
- Has the network converged to the global minimum or a local minima?
- How will the network handle data outside of the training set?
- Is there a quantifiable technique to describe the network's "memory" or data retention?
- Is the network making use of the right set of input parameters for the problem domain?

One oft-cited story[1] recounts a neural network pattern recognition system that was being developed for the army to identify the presence of enemy tanks. Once trained, the system appeared to work perfectly, able to identify tanks in the testing samples and in a completely separate data set. When taken to the field, however, the system failed. After analysis, it was discovered that the system was actually identifying qualities of the pictures it was being presented with. Every photo in the test set that had a tank hidden within it was taken on a cloudy day; coincidentally, every photo without a tank was taken on a clear day. The system had learned to identify cloudy skies and not tanks.

Stories like these push the software industry to establish standard V&V approaches for neural network processes. Both developers performing V&V on their projects and IV&V practitioners need well defined, field tested, and documented methods to apply.

3. METHODS FOR V&V OF NEURAL NETWORKS

One method that is used by neural network developers is to separate a set of available data into three sets: training, validation, and testing sets[2]. The training set is used as the primary set of data that is applied to the neural network for learning and adaptation. The use of the word validation here should not be confused with the second "V" of V&V. The validation set is used to further refine the neural network construction. The testing set is then used to determine the performance of the neural network by computation of an error metric.

This training-validating-testing approach is the first, and often the only, option system developers consider for the assessment of a neural network. The assessment is accomplished by the repeated application of neural network training data, followed by an application of neural network testing data to determine whether the neural network is acceptable.

However, V&V practitioners working on a neural network system for a safety- or mission-critical project require additional activities beyond this developer approach. Some of these activities will have to investigate the neural network in a manner similar to code-inspection and hazard and risk analysis for traditional programming, but taking into account the special circumstances of knowledge acquisition and possible non-deterministic operation.

The following techniques highlight several of the promising areas of research that may lead to the development of standard practices for neural network assurance. These techniques discussed in this paper are improvements in testing, run-time monitoring, formal methods, cross validation, and visualization.

3.1 Automated testing and test data generation methods

As was previously mentioned, the traditional training-validation-testing approach fails to give assurance that a neural network will meet the rigorous standards required for high reliability environments and safety-critical systems. Since most neural networks are not used in high-reliability environments, a testing set for error calculation may be sufficient for the developer to assess the system. For high assurance systems, in applications such as aviation, telecommunication, banking, and robotic exploration, applying a simple testing set may not be adequate. These systems, which require a high degree of system reliability, will need more than the usual limited set of testing data. Automated testing, in combination with test generation algorithms, may help to alleviate this problem and aid in reliability assessment.

To increase the amount of available testing data, system developers can rely upon test data generators that generate anywhere from partially new to wholly new sets of data. Most test data generators have been designed to work on discrete or single valued data. An important class of discrete data generators includes random test generators that can provide an unlimited number of new data points.

Such systems might work well for neural networks involved in classification problems where the inputs and outputs are single valued data. However, discrete data generators cannot aid in reliability estimation of neural network systems where inputs consist of sequences of data. These sequences of data are referred to as *data trajectories*; they occur with varying lengths where each element of the neural network data set is a function of time and a continuous extension over previous data points. Examples of data trajectories include the measurement of angular accelerations from aircraft, equations describing robotic control, speech analysis, and data that describes a chemical or nuclear reaction.

An automated trajectory generator technique[3] was developed for neural network systems that process data trajectories yet have a limited amount of testing data. This trajectory generation scheme relies on expanding existing small sets of test data to build regressive prediction models that are statistically similar to the original test trajectories. The models establish relationships between independent and dependent variables that allow for perturbation of the independent variables to generate new trajectories from the dependent variables.

When combined with automated techniques, the trajectory generator can aid a V&V practitioner within the testing stage of a neural network system. For fixed neural networks, the automated testing could uncover missing knowledge content and reveal a poor fit to the problem domain. For dynamic adaptive neural networks, reliability assessment through automated testing could increase the confidence that a system will behave controllably during operation.

3.2 Run-time monitoring

Run-time or operational monitoring methods appear to be the next evolution of V&V for neural networks. A run-time monitor would need to be developed as part of the system, perhaps beginning as early as the requirements stage of the project. The monitor, working like an oracle, would provide system specific solutions. These methods can be beneficial as a V&V technique because they can provide fault detection and can identify divergent behaviors that could lead to failures. Run-time monitors may be applied to fixed neural networks but they are more valuable as a means to assure real-time dynamic neural networks that adapt during system operation.

The benefits of run-time monitoring include that it requires little incremental effort over traditional testing and it can locate difficult-to-find errors. The drawbacks are the overhead that it adds to program execution, the difficulty in developing a system specific solution, and the possibility of falsely identifying non-existent problems. Furthermore, since run-time monitoring generally observes one execution, certain execution paths may not be covered in a specific run and, therefore, some errors may be missed.

Data sniffing is used to assess data as it enters and exits a neural network to determine whether a dynamic neural network has adjusted acceptably. While traditional run-time monitoring can evaluate the validity of the input and output, data sniffing helps assess and control the adaptive reasoning process of the program. This method utilizes a pre-alert agent and post-block agent to assess the target program[4]. The pre-alert agent captures the incoming data before it enters the system and determines whether or not it may cause unexpected (undesired) adaptations in the system. If so, it offers a warning and allows the data into the system with caution. The post-block agent examines the post-classification value, determines its "distance" to training class norms and, should the new value fall well outside the training domain, the agent prevents it from being used.

Data sniffing would enable an extra layer of protection in the extreme cases where outlier data could degrade system performance to unsafe conditions. In trials, the data sniffing method worked fairly well for data that is not highly relational[4].

Safety monitors are usually system specific implementations. Within the IFCS project, there are two safety monitors, one for the fixed neural network, a pre-trained neural network (PTNN), and one for the dynamic neural network, an OLNN.

The PTNN safety monitor is essentially a reduced table version of the knowledge of the PTNN. This safety monitor resides on a different computer system that has been rigorously tested and has a high level of safety assurance. The monitor checks the outputs from the PTNN to ensure it stays within pre-defined bounds. Should the PTNN output stray from accepted system norms, the monitor can signal disengagement to the flight control scheme.

For the OLNN, a second monitor was developed which verifies aircraft gain calculations computed from using the OLNN outputs. Since the expectations from the OLNN are unknown, this monitor attempts to ensure computations used with OLNN data stay within acceptable robustness bounds that have been selected based upon several different aircraft criteria including physical stress-loads and aircraft handling qualities.

Lyapunov stability analysis can play a critical role in the verification and validation of neural networks. Lyapunov's direct (second) method is widely used for stability analysis of linear and nonlinear systems, both time-invariant and time-variant. It can provide insight into a system's behavior without solving the system's mathematical model. Viewed as a generalized energy method, it is used to determine if a system is stable, unstable, or marginally stable.

The use of Lyapunov during runtime may provide valuable stability information while the system is in operation. It appears to have some promising results and could work well as a type of safety monitor that continually analyzes a network to determine if it is tending towards stability. Lyapunov stability is the current research focus for both NASA DFRC and NASA ARC with regards to run-time monitoring within the IFCS project[5].

3.3 Formal methods

The term *formal methods* refers to the use of techniques from formal logic and discrete mathematics in the specification, design and construction of computer systems and software. The purpose of applying formal methods is to make V&V of the software more objective by supplementing the traditional testing methods. The more rigorous the formal method the more effort and skill required to apply it and the more assurance the method will provide[6]. Formal methods for neural networks include specification languages and methods for knowledge representations including rule extraction and conversion to decision trees[7,8,9]. A formal model allows a system to be proven complete, correct, or consistent[10].

There are many specification languages designed to work for neural networks including CONNECT[11] and nn/xnn[12]. CONNECT is a neural network specification language which allows for the abstract specifications of neural networks and a corresponding compiler which translates the specification into C++ network classes. nn is a high-level neural network specification language and xnn is a graphical front end to allow for the visualization of the network data during training or testing. nn/xnn comes with some neural network architectures already in the language including backpropagation, GRNN, Hopfield, radial-basis functions, and self-organizing maps.

Wing[10] states, "Formal methods are based on mathematics but are not entirely mathematical." Within formal methods, there is a mapping of the mathematical world, represented by a model within a specification language, to the real world, represented by the customer requirements. Certainly, some top level requirements can be stated for a neural network, but since the network adapts opaquely into a function, its exact internal workings are not specifiable. One means of providing insight, which can be used with a formal specification of a neural network, is rule extraction.

Rule extraction can provide a tester with insight into what a fixed neural network has learned and assist in determining the acceptability of the network. After a neural network has been trained and tested to satisfactory levels, a system developer could then apply rule extraction to perform requirement refinement. The refined requirements would aid the system tester in the development of adequate testing procedures and test cases.

A V&V practitioner might use rule extraction for a different purpose. The V&V practitioner could apply rule extraction to obtain a set of rules that mimics the functionality of the network and then use these rules for comparison against the original set of requirements. This would provide information for review of the correctness of the function the network is approximating. Additionally, these rules could assist in the development a formal model of the neural network system. At a minimum, extraction of these rules would provide some sense of confidence that the network will behave as it was intended.

Extraction does not work as well for adaptive systems because the rules would need to be extracted after each iteration of learning and then judged for correctness. Similar techniques, *rule initialization* and *rule insertion*, are more applicable to dynamic neural networks. Through rule initialization, the network is given a starting point from which to adapt, which may offer some improved confidence in its behavior. Setting up a starting point through rule initialization could lead to a constrained learning regime. Rule insertion can be performed periodically, while in operation or offline, to steer a dynamic network toward a desired operational regime.

Another technique used to provide additional insight into the inner workings of a network is converting a neural network to a decision tree[9,13]. Examining a decision tree representing the knowledge of the neural network is more understandable than examining the neural networks actual structure. This method is another way to probe a network to determine if proper knowledge has been gained in training. The decision tree information can be utilized in the same way as the extracted rules, to check against requirements and to provide confidence in the neural network. This technique, like rule extraction, will be most useful for fixed networks.

The above methods can help with documenting the design features of a neural network that relate to its performance. Documentation can be a difficult task, especially when designers are unsure of how to describe the intended results of the neural network. This difficulty may lead to over simplified requirements levied upon the neural network and would reduce the goal of traceability analysis and test case generation. These extraction techniques can generate testable statements from neural networks that would be comparable to the system requirements and give insights into appropriate test selections for verification.

There are many algorithms and tools that make the process of rule extraction or conversion to decision trees accessible to the V&V practitioner[8]. One limitation that exists is that the algorithms and tools work only for a specific type of neural network or a limited number of types.

3.4 Cross validation

Cross validation is an approach similar to N-version programming. Its basis is “reliability through redundancy,” a concept from software engineering. The cross validation concept centers on combining diverse neural networks into an ensemble. The output of the component networks may then be checked against one another to affirm validity and appropriateness. In this case, validation is designed into the system itself instead of being performed after the fact. The system uses a group of neural networks to validate another neural network. This model is particularly suitable for safety-critical environments.

Note that cross validation has a different meaning here than the term “cross-validation” commonly used within the neural network community. In neural network development, cross-validation refers to a method for estimating generalization error[14]. However, in terms of V&V, cross validation is used by developing differing solutions for a single input domain and then comparing/combining these solutions against each other to obtain a single result.

While results indicate that using ensembles of neural networks increases the performance over a single neural network, redundancy alone does not ensure improved performance. It must first be determined what kind of diversity may lead to improved performance, and what is the best way of creating sets of neural networks that show this kind of diversity[15].

Sharkey and Sharkey[16] list numerous methods using such ensemble combinations. One such method relies on training neural networks from different starting points, or different initial conditions. Another method varies the network architecture or the algorithms. Other methods rely on varying the data, such as sampling, using different data sources, different preprocessing methods, distortion, and adaptive resampling.

If the ensemble is made up of differing types of networks, then the output (prediction) of each of the neural networks is collected and combined. The combination can be done in several ways, including voting, average, or weighted average. This method is used to improve the output of a neural network by having several “opinions” on which to base the final decision[17].

Cross validation could be employed in a manner similar to diversification for fault-tolerant software. Instead of using a single neural network, different neural network architectures or copies of the same neural network architecture with different training data sets can work in combination, providing checks and balances to system control. This technique filters out individual failures in a neural network because it can rely upon a set of neural networks. System reliability can be expected to improve by introducing support for error detection and recovery.

3.5 Visualization

Designers, end-users, and V&V practitioners need to understand the performance of the neural network system: how it operates and arrives at its decisions. One technique that can be useful in meeting the goal of fully understanding the performance of the neural network is visualization. Some forms of visualization involve transforming data into visual forms that can be more easily understood. Humans can comprehend vast quantities of data that have been visualized. Visualization techniques capitalize on a person’s highly developed visual pattern-recognition abilities.

Visualization tools and techniques sometimes used in the V&V of traditional software (code coverage or structure examination) may prove even more important for neural network software. Tools and techniques that assist understanding through visual representations may greatly assist a V&V practitioner in understanding the software they are called on to certify.

Visualization techniques may be effectively applied to a neural network during the training and testing stages. Graphical representations of the way weights and internal connections of a neural network change can aide human interpretation and analysis. A plot of a changing error function gives almost immediate clues as to how well a neural network is learning. These techniques can provide both insight into the decision-making and the learning processes of a neural network during training.

There are a number of visualization techniques for understanding the learning and decision-making processes of neural networks, as well as visual methods for testing the completed system containing a neural network as a component[18-23]. Methods used for the visualization of a neural network during the learning phase of its development include the hyperplane animator, trajectory diagrams, and visual techniques incorporated in the MATLAB Neural Network Toolbox. Additionally, visualization software can provide an interactive mechanism that enables the user to adjust parameters and quickly see the effects of the changes.

Two-dimensional diagrams, three-dimensional plots, or even multi-screen displays of parameters could be used to visually compare structures and adaptation of the neural networks. These activities could be used as a concept stage V&V activity to validate the constraints or limitations of the proposed neural network architecture.

Visualization can aid in other aspects of the V&V process for neural networks, including the design definition, training set examination, examination of weights and biases during training, examination of structure after training, and analysis of operation during testing.

Another process activity where visualization proves useful is testing. Neural networks are often tested as a black box; however, there are many visual techniques that would allow white box testing of neural network software by the developers or by V&V practitioners. The capabilities of MATLAB's Neural Network Toolbox demonstrate some of these techniques that give visual examination to the internal workings of the neural networks learning process[24]. For the IFCS project, several MATLAB plotting scripts were developed to visually observe the various results from the dynamic neural network (both in simulation and from a C version) to determine if the neural network was working correctly.

During the testing of a system, a simulation environment that incorporates visualizations of various aspects of the system can be an effective tool to provide information about real-time adaptation of a neural network. An example of a sophisticated simulation environment is the F-15 Flight Simulation developed by WVU for the IFCS project[23, 25].

This simulation, constructed in MATLAB and Simulink allows the researchers the ability to scope data as it passes throughout the system. Visual cues result from integration into a 3D modeling package called Aviator Visual Design Simulator (AVDS) where a pilot can operate the aircraft and watch how it behaves on the screen in front of them. The simulation user also has the option of selecting from among several different variables to plot in real-time, with the added ability of applying limits onto the plots for immediate identification of undesired behavior.

3.6 Other techniques

Some studies examine ways to assess selection of the best neural network architecture for a specific problem. Since a large number of different neural network solutions exist that are capable of approximating classification, linear, or non-linear functions to varying degrees of accuracy, providing for automatic selection may assist in the development stage of the life cycle. One report[26] suggests the creation of several different neural network solutions for a given training-testing set of data followed by a comparison and selection based upon statistical procedures. Perhaps a set of recommended neural network architectures could be generated based upon the results of the statistical procedures through user-given criteria or expert systems, which contain some knowledge concerning the expected input domain.

Another method for improvement of quality during the development life cycle may be the automatic optimizing of the neural network configuration. Some studies have been done to look at methods for providing recommendations for learning algorithm improvement in adaptive systems, internal parameter selection, and internal network connections. Ya[27]o presents a good overview of the work done in neural network optimizations. Several of these techniques rely upon the use of other forms of evolutionary computation such as genetic algorithms.

An analysis technique is discussed by Hull[28] that can be applied to static neural networks to provide a mapping between several independent variables to a dependent variable. Such mapping solutions are common to aerospace systems. The technique consists of separating the input domain into a series of test rectangles where the output for the neural network is known at the vertices of these rectangles. A guaranteed output bound for the neural network is found for the untested regions within the rectangles through the use of the maximum growth rate in a first-order Taylor series expansion. This analysis technique has been shown to provide bounded ranges on neural network output for some simple neural network systems. Hull, et.al. suggest that this technique may allow for the certification of neural networks in aerospace systems by the Federal Aviation Administration.

4. CONCLUSIONS

Different methods apply to the different stages of the neural network software life cycle. Cross validation, a technique that would be employed during the design phase of a project, is realized in the use of ensembles of networks of the same or differing types to accomplish the same task thus improving the dependability of the system. Rule extraction applies best to fixed neural networks that have undergone some level of training. If a set of rules that describes how the network will behave has been obtained, the rules can be used in requirements traceability to verify the network against a system specification. Run-time monitoring may best be applied to adaptive neural networks and is used after they have been deployed into operation. Lyapunov stability analysis can be used either during a network's development or while it is in operation. Lyapunov stability used during operation would be in conjunction with a system monitor that could decide the neural network's performance and make safety judgments. Improved testing techniques, utilizing the automated trajectory generator, would be useful during the testing phase after the neural network has undergone initialization and training. Visualization proves useful during network training to provide feedback on the learning and decision making processes and as an analysis tool to understand the results of testing.

Overall, the V&V field for neural networks is starting to receive attention; new tools and techniques have emerged in the past ten years. Still, with all of this effort, no complete methodology exists to provide a V&V practitioner with the ability to gain an assurance for neural networks in safety- and mission-critical systems.

ACKNOWLEDGEMENTS

This research was sponsored by NASA, under Research Grant NAG5-12069.

REFERENCES

1. Skapura, David M. and Peter S. Gordon, *Building Neural Networks*, Addison-Wesley, 1996.

2. Ripley, B.D., *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

3. Cukic, Bojan, Brian J. Taylor, and Harhsinder Singh, "Automated generation of test trajectories for embedded flight control systems," International Journal of Software Engineering and Knowledge Engineering, **12**(2), pp. 175-200, 2002.

4. Liu, Yan, Tim Menzies, and Bojan Cukic, "Data sniffing – Monitoring of machine learning for online adaptive systems," In *14th IEEE International Conference on Tools with Artificial Intelligence*, 2002.

5. Schumann, J., and Stacy Nelson, "Toward V&V of Neural Network Based Controllers," Workshop on Self-Healing Systems, 2002.

6. Nelson, Stacy and Charles Pecheur, "V&V of advanced systems at NASA," Produced for the Space Launch Initiative 2nd Generation RLV TA-5 IVHM Project, 2002.

7. Andrews, Robert; and S. Geva, "On the Effects of Initializing a Neural Network With Prior Knowledge," *Proceedings of the International Conference on Neural Information Processing*, pp. 251-256, Perth, Western Australia, 1999.

8. Andrews, Robert; J. Diederich; and A. B. Tickle, "A Survey and Critique Of Techniques For Extracting Rules From Trained Artificial Neural Networks." Knowledge Based Systems **8**, pp.373-389, 1995.

9. Boz, Olcay, 2002, "Extracting Decision Trees From Trained Neural Networks," KDD-2002 The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.

10. Wing, J. M., "A Specifier's Introduction to Formal Methods," IEEE Computer **23** (9), pp.8-24, 1990.

11. CONNECT webpage, http://www.first.fraunhofer.de/connect/

12. nn/xnn webpage, http://www.bgif.no/neureka/

13. Craven, M.W., and J.W. Shavik, "Extracting tree-structured representations of trained neural networks," Advances in Neural Information Processing Systems **8**, MIT Press, Denver, CO., 1996.

14. Bishop, C. M., *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 2000.

15. Sharkey, Amanda and Noel Sharkey, "How to improve the reliability of Artificial Neural Networks". Department of Computer Science, University of Sheffield, UK. Research Report CS-95-11, 1995.

16. Sharkey, A.J.C., N.E. Sharkey, and O.G. Chandroth, "Neural nets and diversity, *Proceedings of the 14th International Conference on Computer Safety, Reliability and Security*, pp. 375-389,Springer-Verlag, 1995.

17. Krogh, Anders and Jesper Veldelsby, "Neural network ensembles, cross validation, and active learning." Advances in Neural Information Processing Systems **7**, pp. 231-238, 1995.

18. Craven, Mark W., and Jude W. Shavlik, "Visualizing learning and computation in artificial neural networks," International Journal on Artificial Intelligence Tools **1** (3), pp. 399-425, 1992.

19. Wejchert, J., and G. Tesauro, "Neural network visualization," Advances in Neural Information Processing System **2**, pp. 465-472, 1990.

20. Munro, P., “Visualization of 2-D hidden unit space,” Technical Report LIS035/IS91003, School of Library and Information Science, University of Pittsburgh, Pittsburgh, PA, 1991.

21. Pratt, L. Y. and S. Nicodemus, “Case studies in the use of a hyperplane animator for neural network research,” *Proceedings of the IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence* 1, pp. 78-83, 1993.

22. Simmons, Reid and Gregory Whelan, “Visualization tools for validating software of autonomous spacecraft,” *Proceedings of the 4th International Symposium on Artificial Intelligence, Robotics, and Automation for Space*, Tokyo, Japan, 1997.

23. Perhinschi M. G., G. Campa, M. R. Napolitano, M. Lando, L. Massotti, M. L. Fravolini, “A Simulation Tool for On-Line Real Time Parameter Identification,” *Proceedings of the AIAA Modeling and Simulation Conference,* AIAA2002-4685, Monterey, CA, 2002.

24. Mathworks, Inc., *Neural Network User’s Guide (version 3)*, 1998.

25. Perhinschi, M. G., G. Campa, M. R. Napolitano, M. Lando, L. Massotti, and M.L. Fravolini, “Modeling and simulation of a fault tolerant flight control system.” Submitted to International Journal of Modelling and Simulation in April 2002.

26. Instant Recall, Inc., Final Report for the Verification and Validation of Neural Nets, 1997. http://www.irecall.com/neural/nnindex.htm

27. Yao, Xin, “Evolving Artificial Neural Networks,” *Proceedings of the IEEE* 87(9), pp.1423-1447, 1999.

28. Hull, J., David Ward, Radoslaw R. Zakrzewski, “Verification and Validation of Neural Networks for Safety-Critical Applications,” *Proceedings of American Control Conference,* Anchorage, AK, 2002.