2019

Rule Extraction and Insertion to Improve the Performance of a Dynamic Cell Structure Neural Network

Osama Amhamed Elsarrar
West Virginia University, oelsarra@mix.wvu.edu

Rule Extraction and Insertion to Improve the Performance of a Dynamic Cell Structure Neural Network

Osama Amhamed Elsarrar

Dissertation submitted
to the Eberly College of Arts and Sciences
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in

Mathematics

Marjorie Darrah, Ph.D, Chair
Harvey Diamond, Ph.D
Hany Ammar, Ph.D
Adam Halasz, Ph.D
Jessica Deshler, Ph.D

Department of Mathematics

Morgantown, WV

2019

Keywords: Neural Network, Rule Extraction, Rule Insertion

ABSTRACT

Rule Extraction and Insertion to Improve the Performance of a Dynamic Cell Structure Neural Network

Osama Amhamed Elsarrar

Artificial Neural Networks are extremely useful machine learning tools. They are used for many purposes, such as prediction, classification, pattern recognition, etc. Although neural networks have been used for decades, they are still often not completely understood or trusted, especially in safety and mission critical situations. Typically, neural networks are trained on data sets that are representative of what needs to be learned. Sometimes training sets are constructed in order to train the neural network in a certain way, in order to embed appropriate knowledge. The purpose of this research is to determine if there is another method that can be used to embed specific knowledge in a neural network before training and if this improves the performance of a neural network.

This research develops and tests a new method of embedding pre-knowledge into the Dynamic Cell Structure (DCS) neural network. The DCS is a type of self-organizing map neural network that has been used for many purposes, including classification. In the research presented here, the method used for embedding pre-knowledge into the neural network is to start by converting the knowledge to a set of IF/THEN rules, that can be easily understood and/or validated by a human expert. Once the rules are constructed and validated, then they are converted to a beginning neural network structure. This allows pre-knowledge to be embedded before training the neural network. This conversion and embedding process is called ***Rule Insertion***.

In order to determine whether this process improves performance, the neural network was trained with and without pre-knowledge embedded. After the training, the neural network structure was again converted to rules, ***Rule Extraction***, and then the neural network accuracy and the rule accuracy were computed. Also, the agreement between the neural network and the extracted rules was computed.

The findings of this research show that using Rule Insertion to embed pre-knowledge into a DCS neural network can increase the accuracy of the neural network. An expert can create the rules to be embedded and can also examine and validate the rules extracted to give more confidence in what the neural network has learned during training. The extracted rules are also a refinement of the inserted rules, meaning the neural network was able to improve upon the expert knowledge based on the data presented.

DEDICATION

To my friends, family, and all those interested in exploring the endless wonders and knowledge a study of mathematics affords.

ACKNOWLEDGEMENTS

First and above all, praise and gratitude to Allah, the Almighty, for providing this opportunity and granting me the ability to conclude my research successfully. Also, I wish to express my full gratitude to the messenger Mohammed and his progeny. Without Allah's blessings, this accomplishment would not have been realized.

I am inordinately grateful to my supervisor, Professor Marjorie Darrah. Over the last 4 years, she has provided unwavering support, advice, and expertise within the realm of my research. She went beyond the necessary responsibilities of a mentor, always willing and available to sit with me and exhaustively answer my questions to ensure both my confidence and success. In addition to that, she has also been a patient and friendly person, offering encouragement when I needed it. She would also take time to inquire about my personal life and family and ensure those were in balance with my thesis responsibilities.

Along with my supervisor, I would like to acknowledge the help, support, and encouragement of all staff and faculty of the Department of Mathematics. It has been my great pleasure to work with others in academia who share my love and deep abiding interest in the study and use of mathematics in understanding our complex world. I am indebted to the faculty and fellow students who provided a stimulating learning environment.

To Mom and Dad, though they are now with Allah, I would like to thank them for everything they did to get me to this point in my life. They created me, raised me lovingly, encouraged and supported my education - they even gave me the genes which made me intrigued by mathematics from a young age. *Inshallah*, I will tell them the same one day.

To my wife Hend, I wish to convey my deepest love and affection. She has always been the love of my life, a beautiful, smart, and kind woman who has supported me through many challenges. In the last 4 years in particular, she has been there to take care of our 6 children when I have been at times absorbed in my studies. She is always ready with a kind word, a smile after a long day, a recognition of my emotional needs. She keeps our family rooted and strong, priorities beyond my higher education. I could not ask for a more lovely and intelligent woman to be my partner in this life. I love you beyond all measure, Hend.

To my children Sajida, Nusaiba, Suheil, Serene, Sulafa, and Hussama, I would like them to know how much I love them, and also how sorry I am to them. Over the last 4 years of my thesis studies, I have not been present as much as I would have liked. I missed out on many of their achievements and activities. Though in the long run I know my education will ensure a better life for them, I also recognize the sacrifices they had to endure with their father often being away. I love you all so much, and I look forward to spending much more time with you in the future and seeing you grow up, learn, love, and succeed.

To all my many brothers and sisters, I would like to convey my undying love and recognition of their support. Though they are all physically many thousands of miles away in Libya, I feel like they are much closer. In our conversations, they have always expressed constant encouragement and pride that their brother was working hard to accomplish a very long-term goal. I love you all, and I hope to see you again in the near future.

Last but certainly not least, I would be negligent if I did not also express my sincere thanks to all my professors, colleagues, and friends who gave me emotional and academic support throughout this long period of my studies. They made my experience at West Virginia University a most pleasurable and memorable one. You became my second home for the last 7 years. It was a welcome and invigorating home. I thank you all, and I will never forget my WVU "family".

Thank You All

LIST OF FIGURES

LIST OF TABLES

CONTENTS

I. INTRODUCTION

Machine learning is programming computers to optimize a performance criterion using example data or past experience. Over the past two decades Machine Learning has become one of the mainstays of information technology and a central part of our lives. With the ever-increasing amounts of data becoming available smart data analysis is becoming a pervasive and necessary part of technological progress. Much of the art of machine learning is to reduce a range of fairly disparate problems to a set of fairly narrow prototypes. Much of the science of machine learning is then to solve those problems and provide good guarantees for the solutions

Artificial neural networks are an attempt at modeling the information processing capabilities of nervous systems. Thus, first of all, we need to consider the essential properties of biological neural networks from the viewpoint of information processing. This will allow us to design abstract models of artificial neural networks, which can then be simulated and analyzed. Natural neurons in the brain receive signals through synapses located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain threshold), the neuron is activated and emits a signal though the axon. This signal might be sent to another synapse, and might activate other neurons

Artificial neural networks are composed of nodes that are called “artificial neurons”. An artificial neuron is a computational model inspired by the natural neurons. The biological neural networks of the brain inspire machine learning. The “neurons” in artificial neural networks adapt through training on sets of data. This allows the artificial neural network to learn the patterns in the data. The artificial neural networks attempt to design themselves after the biological neural network, processing information through repetitive experience. The artificial neural network is only one kind of machine learning, used as a statistical tool within a multitude of fields.

1.1 Types of Neural Networks

From this point on we will refer to artificial neural networks as simply *neural networks*. This section presents three basic neural network structures.

Feedforward Neural Networks:

Feedforward neural networks consist of three or more layers (input layer, hidden processing layer(s), and output layer), each layer can only provide direct connection to the layer succeeding it. For example, the hidden processing layer can only send direct connections to the output layer. Feedforward neural networks are considered to be the simplest form, as the process can only travel in one direction.

Feedforward neural networks are often coupled with a backpropagation training algorithm. In order to revise and correct the neural signal, the error must be propagated back through the network, and appropriate changes are made to the weights before trying again. The message is propagated as many times as necessary to reduce the error down to an acceptable level.

Recurrence Neural Networks:

As opposed to the feedforward neural connection, which has clearly defined layers, the recurrence networks do NOT always have defined input or output neurons. Three types of recurrence can occur --- DIRECT or INDIRECT, or LATERAL recurrence. Direct recurrence (also referred to as self-recurrence) use neurons to strengthen themselves in order to reach their activation limits, whereas indirect recurrences occur when neurons hold connections with those preceding layer neurons. Lateral recurrence permits connections within ONE layer only. An example of this would be the fully recurrent neural network, which is the simplest form.

Completely Linked Neural Network:

Completely linked neural networks permit connections between ALL neurons. Every neuron is permitted to be connected with every other neuron, which results in every neuron having the potential to become an input neuron. An example of a completely linked network would be a self-organizing map.

Later in this paper, we will be focusing on one type of self-organizing map called the dynamic cell structure (DCS) neural network [1, 2, 3]. These neural networks are designed as topology representing networks whose roles are to learn the topology of an input space. The DCS neural network partitions the input space into Voronoi regions. The neurons within the neural network represent the reference vector (centroid) for each of the Voronoi regions. The

connections between the neurons, c_{ij}, are then part of the Delaunay triangulation connecting neighboring Voronoi regions through their reference vectors.

Given an input, v, the best matching unit (BMU) is the neuron whose weight, w, is closest to v. Along with the BMU, the neighbors of the BMU are found through the Delaunay triangulation. During adaptation, adjustments are made to the BMU and neurons within the BMU neighborhood based on the input.

The DCS algorithm consists of two learning rules, Hebbian and Kohonen. These two learning rules allow the DCS neural network to change its structure to adapt to inputs. The ability to adjust neuron positions and add new neurons into the network gives the DCS neural network the potential to evolve into many different configurations.

1.2 Uses of Neural Networks

In today's world, neural networks are becoming increasingly common in a multitude of sectors, including business, sports, science, technology, manufacturing, and so forth. In the business world, using neural networks, businesses and organizations can easily calculate risk, identify patterns in sales, and make predictions about future sales [4].

There are many uses instances of neural networks used in safety critical roles. In technological advances, neural networks aid in our ability to move toward advancements, such as the self-driving cars, by managing the steering processes. Another example in the technological field would be pattern recognition in relation to fingerprint scanning, voice recognition, and applications like image processing [5]. In medicine, this tool has been used to fine tune advancements like the cochlear implant, allowing the device to train itself to filter out particular audio noises. The Dynamic Cell Structure, mentioned in the last section, is a component of an intelligent flight control system developed by NASA, Boeing Phantom Works, the Institute for Scientific Research, Inc., and West Virginia University [6].

Neural networks have an extremely wide range of applications that ultimately affect us all. The ability to understand how they work and what they have learned from data is extremely important to their trusted use. Often neural networks are viewed as a "black box"; meaning that after training, it is hard to know exactly what they have learned and predict how they will react

with data outside the training set. Techniques must be developed to make sure that these very important tools can continue to be used safely and effectively.

1.3 Rule Extraction

A negative seen when using artificial neural networking is the fact that the "knowledge" is coded as a weights or activation values. This results in very few tools capable of validating the neural network process. Rule Extraction is a technique that can be used to make neural networks more understandable by assisting in revealing the internal knowledge of a trained neural network. The more accurate your rule extraction, the better it matches your neural network [7]. The predictions of a network can be explained through the rules extracted from it, making a neural network less of a "black box" of unexplained answers and more of an understandable process [8]. By using rule extraction, the degree of matching between network responses and rule classification allows the developer and user to understand the neural network inner workings and be confident in what it has learned [9].

Types of Rule Extraction

Rule Extraction is a technique that can be used with several different types of classification techniques, such as decision trees, support vector machines, and neural networks. For now, the focus will be on the algorithmic methods that have been developed using the three types of rule extraction: pedagogical, decompositional, and eclectic. Each type of rule extraction focuses on different aspects of the neural network.

The pedagogical or "black box" method extracts the rules by paying close attention to the input-output relationships, attempting to mirror the way the neural networks understand the relationship between the input-output signals as closely as possible. The pedagogical approach to algorithms is typically the fastest approach, because it does not take the time to scrutinize or analyze the internal weights of the network. However, because of this, this approach is also less likely to accurately obtain all of the rules that help describe the network's behavior [10]. The main advantage of using the pedagogical approach lies in the fact that it applies to most neural networks, whereas the decompositional approach can be more limited [11].

The decompositional, or “white box”, approach can be more difficult than the pedagogical, however, the extra effort it takes helps improve the accuracy of the rules extracted. The decompositional approach takes a look at the internal weights and make-up of the network in order to more accurately extract rules [10]. The advantage of this approach is that the analyzing of the internal weights and makeup help create an accurate set of rules for the entire neural network [11].

The eclectic approach, or “mixed box”, is an approach that uses parts of the pedagogical AND decompositional methods. Generally, this can take longer than the pedagogical approach because of the decompositional aspects it uses, but like the decompositional approach, the results are likely to be much more accurate than the pedagogical [10].

Types of Rules that can be extracted from a neural network include the following:
1. Propositional: IF .... THEN ...., ELSE ....
2. MofN rules: IF M of the given N conditions are satisfied, THEN .....
3. Fuzzy rules: IF X is large, THEN ....., ELSE IF X is medium, THEN .....

1.4 Rule Insertion

For safety critical uses of neural networks, accuracy and confidence are very important. The rigidity of the black box approach prevents the widespread application of neural networks in some safety-critical systems. There is a three step process that can assist in creating the most accurate outputs for neural networks and also provide confidence by allowing developers and users to better understand the internal workings of the neural network. First, a rule is inserted into the neural network using a specific program. This does not need to be a complete rule, as it is likely to be refined at the next step. Inserting the rule will change the dynamic from a symbolic representation to a neural representation. The second step is to train the specified program by using a standard neural learning algorithm, backpropagation, or other weight optimization methods. Performing this second step will correct the rules previously inserted so that they are consistent and accurate. The final step, rule extraction, occurs once the rules have been refined and symbolic information is then extracted.

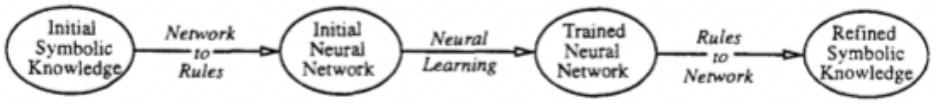

Figure 1. Rule Insertion, Train, Rule Extraction

The idea of rule insertion is that gathered knowledge is represented in a set of rules, which could possibly be incomplete or incorrect due to insufficient knowledge. A hybrid system refers to utilizing a combination of systems, which could use theoretical and empirical data. One such system is the sub-symbolic such as neural networks; another is the symbolic-based reasoning, such as expert system.

The way a hybrid system works is first to form rules that represent the gathered knowledge. The initial knowledge is inserted and processed by rules-to-network algorithms. The inserted knowledge is now the initial neural network, which in turn is the initial symbolic knowledge. The initial symbolic knowledge then goes through a stage of training and refinement. Upon completion of training, rules are extracted (output).

Kurd [12], discussed that the issue with the artificial neural networks (ANNs) is that its lifecycle relies on determining the specifications at the initial phase of development. This does not foster learning if the initial data are limited. The lifecycle of the hybrid systems can be described by the "W" model with the following levels (see Figure 2):

1. Symbolic Level: it is associated with symbolic information and deals with analysis in terms of symbolic knowledge.
2. Translation Level: This is where symbolic knowledge and neural architectures are joint or separated.
3. Neural Learning Level: This level uses neural learning to adapt and refine symbolic knowledge.

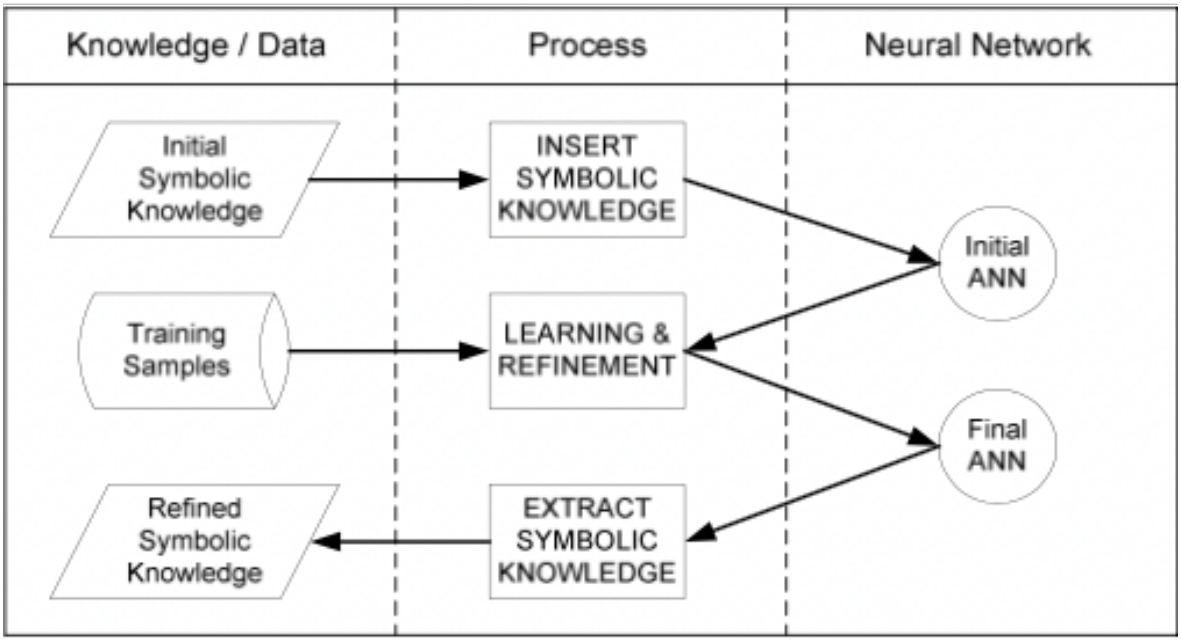

Figure 2. Rule Insertion Extraction W Model Kurd [12]

Towell and Shavlik [13], introduced the new algorithm named Knowledge-Based Neural Network (KBANN). They conjectured that this algorithm would improve the learning speed because it is not ignoring any information. They described this algorithm as a way to address the problems of training "deep" networks. KBANN is a hybrid learning system and is more effective at classifying examples compared to other machine learning algorithms (see Figure 3).

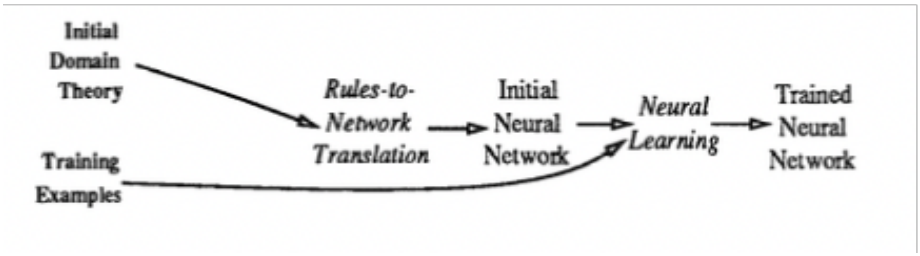

Figure 3. Rule Insertion Process Towell and Shavlik [13]

Giles and Omlin [14] also discuss methods for extracting, inserting and refining symbolic grammatical rules for recurrent networks. The issues also discussed in this paper include how rules are inserted into the recurrent neural network, how training and generalization is affected, and how the rules can be checked in order for correction. The method Giles and Omlin devised requires the network size to exceed the number of Deterministic Finite State Automata (DFA)

states (see Figure 4). It was expected that the training time would decline with rising rule strength, but the network does not easily recognize partial correct rule insertion if the rule strength is too great. An additional aspect of symbolic knowledge extraction and insertion is rule checking, allowing for the establishment of the validity of the knowledge. Rule checking compares rules extracted from trained networks with prior knowledge. However, rule checking becomes increasingly difficult with rising rule strength when incorrect rules are inserted into a network. Further, the authors suggest that network architecture can be altered during training with symbolic guidance, and symbolic information gained from undertrained networks could prove useful in determining the current network architecture. However, there is no limitation that these methods previously described only should be used with symbolic data. Future studies should investigate this further.

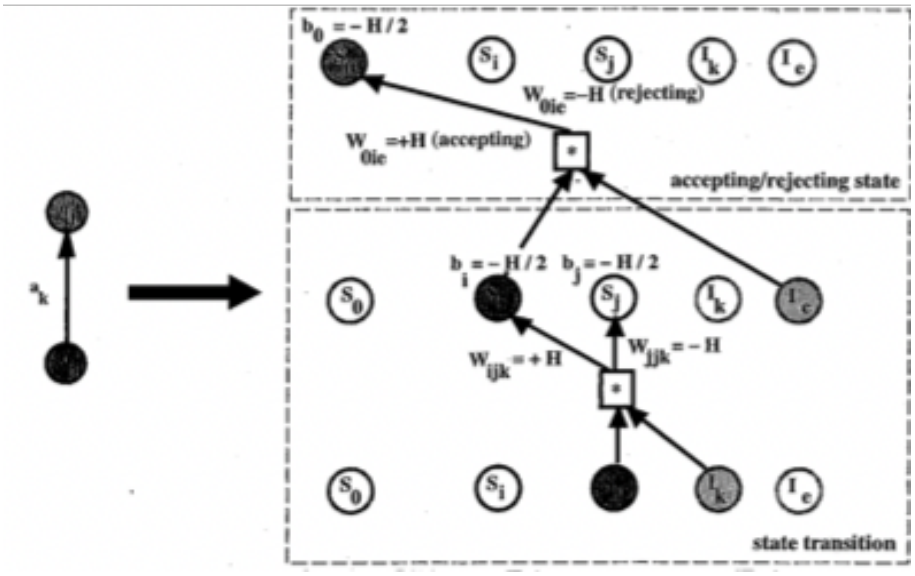

Figure 4. Rule Insertion Process Giles and Omlin[14]

This work produces a new Rule Insertion Method for the Dynamic Cell Structure (DCS) Neural Network. The new method creates rules from knowledge of an "expert" and then inserts the rules as pre-knowledge into the network before training. This thesis is structured as follows: Firstly, in Chapter 2 there is a discussion and application of the method of Rule Extraction for the DCS. This method is applied to the set of data to predict forest fires. This discussion illustrates how rules extracted from a neural network are useful in explaining the knowledge in a neural network, which is the first paper. Chapter 3, the second paper, discusses the new Rule

Insertion Method. The chapter outlines the new method of Rule Insertion and uses a benchmark set of data to test the method. The application of the new method to a large set of Social Science data is Chapter 4, the third paper. With the help of an "expert" in the field of Social Science, rules are developed based on the expert's knowledge. These rules are inserted into the DCS neural network and then the DCS is trained on the data. The method is tested by comparing the neural network analysis results with and without the pre-knowledge inserted. The final chapter, Chapter 5, discusses future directions for this work.

II. PAPER 1: ANALYSIS OF FOREST FIRE DATA USING NEURAL NETWORK RULE EXTRACTION WITH HUMAN UNDERSTANDABLE RULES

2.1 Abstract

Forest fires spread fast, uncontrollably, and may leads to massive destruction. This makes the prevention of them a safety critical issue. Neural networks are a sub-area of machine learning that can be used to analyze the complex behavior of natural systems and help to predict forest fires. To make the knowledge learned by a neural network more accessible, rules can be extracted from the neural network to demystify the system behavior and directly relate inputs to outputs. In this paper, we present a Dynamic Cell Structure (DCS) neural network used for forest fire data prediction, determining which environmental factors lead to fires. We apply an intuitive rule extraction algorithm to extract understandable rules for this prediction. The results are verified through direct comparison with the raw data.

2.2 Introduction

With the ever-increasing amount of data becoming available, smart data analysis is becoming pervasive in every aspect of life to solve a disparate set of problems. Machine learning seeks to reduce this range of disparate problems to a set of fairly narrow examples. The science machine learning is then used to solve these examples and guarantee their solutions [15]. One example of machine learning that can be used to provide analysis for a wide range of problems is the neural network. However, some refer a neural network as a black-box method that can be difficult to understand and trust. It is also sometimes challenging to know exactly how the inputs are related to the outputs of a neural network, and whether the selected inputs have any significant relationship to the outputs [16]. There are methods, such as rule extraction, that paired with neural networks make the knowledge the neural network has learned by being trained on the data a little easier to understand and can assist with the connection between input and output.

One significant threat to the environment and human life, where analysis would be beneficial is in the area of forest fire prediction. In the past, a large effort was made to collect data and build automatic detection tools that could assist Fire Management Systems (FMS). With respect

to forest fires, there are several potential methods that can be used. By utilizing meteorological approaches, satellites, and infrared/smoke scanners, the data can better predict when and where a fire could occur. Temperature, wind, relative humidity, etc. are factors that come into play when analyzing the meteorological aspect of it. Using such analysis methods helps strengthen fire management techniques [17].

Several researchers have applied various methods of analysis to the area of forest fire prediction. Clar, Drossel and Schwable [18] applied the idea of self-organization to the analyses os forest fire data. They introduced the "forest fire of self-organized criticality" model, which refers to the tendency of certain large dissipative systems to drive themselves into a critical state independent of the initial conditions and without fine tuning of the parameters. Grishin and Filkov [19] developed a deterministic-probabilistic expert system for prediction of forest fires. Their model included the drying of forest combustibles and determined the probability of the emergence of a forest fire within the j^{th} time range of the forest-fire period (dynamic model) and fire caused by meteorological conditions.

Eskandari [20] used fuzzy sets integrated with analytic hierarchy process (AHP) in a decision-making algorithm to model the fire risk in the study area. He used four major criteria (topographic, biologic, climatic, and human factors) and 17 subcriteria in his model. The fuzzy AHP method was used to express the relative importance and priority of the major criteria and sub-criteria in forest fire risk in the study area.

Principal Component Analysis (PCA) and Self-Organizing Map (SOM) techniques have been applied to visualize and classify fire risk distribution in forest regions based on a hot spot dataset [21]. Both methods are a suitable method for extraction the high dimensional data onto a low dimensional representation. The SOM map gave an excellent classification and visualization of fire risk in forest regions via the node clusters and useful method for analysis of large size datasets. The PCA explained most the cumulative variance of data, but had difficulty with revealing a representative data pattern when the technique was applied to available large-scale data sets.

Cortez and Morais [22] used several data mining techniques for predicting size of forest fires. testing a variety of techniques, including Support Vector Machines (SVM) and random forests, and four distinct feature selection setups they achieved a predictive accuracy of 46% given

a tolerance of 1 hectare and 61% given a tolerance of 2 hectares. It is worth noting that this accuracy is achieved using four independent variables.

Youssef and Bouroumi [23] used a backpropagation learning algorithm for predicting forest fires data. The neural network that they used is a multilayer perceptron whose number and size of hidden layers can be heuristically determined for each application using its available data examples. They improve the error rate (ER) from 25% to 9%. They fixed the Input layer to 12 neurons and the output to one neuron. Also, they used C++ to code the algorithm. I use the same data but different method.

In this work, we use a dynamic cell structure (DCS) neural network with an associated rule extraction method to analyze various meteorological and environmental input parameters. The goal is to determine from a set of given parameters, what conditions will likely result in a forest fire. The DCS does the analyses, but the rule extraction techniques are used to produce rules that can be easily understood and verified by experts. The combination of these methods produces more useful and implementable results.

The rest of the paper is organized as follows. In Section 2, we present background material on the DCS neural network and rule extraction method. Section 3 discusses the application of our technique to the forest fire data. Section 4 provides a comparison of previous forest fire analyses with our method. Section 5 give conclusions.

2.3 DCS Neural Network and Rule Extraction

This section discusses the Dynamic Cell Structure (DCS) neural network, the idea of rule extraction in general, and the specific rule extraction techniques developed for the DCS neural network.

2.3.1 Dynamic Cell Structure NN

One type of neural network is self-organizing map. The specific self- organizing map that we are working with is called the Dynamic Cell Structure (DCS) neural network [1, 2, 3]. This type of neural network is designed as a topology representing network whose role is to learn the topology of an input space. The DCS neural network partitions the input space into Voronoi regions (Fig. 5). The neurons within the neural network represent the reference vector (centroid) for each of the

Voronoi regions. The connections between the neurons, c_{ij} , are then part of the Delaunay triangulation connecting neighboring Voronoi regions through their reference vectors.

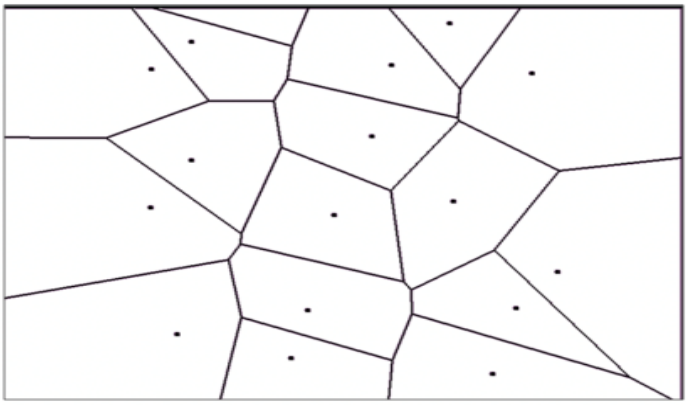

Figure. 5 Voronoi Diagram

Given an input to the DCS, v, the best matching unit (BMU) is the neuron whose weight, w, is closest to v. Along with the BMU, the neighbors of the BMU are found through the Delaunay triangulation. During adaptation, adjustments are made to the BMU and neurons within the BMU neighborhood based on the input.

The DCS algorithm consists of two learning rules, Hebbian and Kohonen (See below). These two learning rules allow the DCS neural network to change its structure to adapt to inputs. The ability to adjust neuron positions and add new neurons into the network gives the DCS neural network the potential to evolve into many different configurations.

$$\boldsymbol{c_{ab}} = \begin{cases} \mathbf{1} & \boldsymbol{a} \in [\boldsymbol{BMU}, \boldsymbol{SEC}] \ \wedge \boldsymbol{b} \in [\boldsymbol{BMU}, \boldsymbol{SEC}] \\ \boldsymbol{\alpha} \cdot \boldsymbol{c_{ab}} & \boldsymbol{\alpha} \cdot \boldsymbol{c_{ab}} > \mathbf{0} \\ \mathbf{0} & \boldsymbol{\alpha} \cdot \boldsymbol{c_{ab}} < \mathbf{0} \\ \mathbf{0} & \boldsymbol{a} = \boldsymbol{b} \end{cases} \quad (1)$$

$$\Delta w_{BMU_i} = \varepsilon_{BMU}(v_i - w_{BMU_i}) \quad (2)$$

$$\Delta w_i = \varepsilon_{NBR}(v_i - w_i) \quad (3)$$

2.3.2 Rule Extraction

A negative seen when using artificial neural networking is the fact that the knowledge is coded as weights or activation values. This results in very few tools capable of validating the neural network process. Rule Extraction is a technique that can be used to make neural networks more understandable by assisting in revealing the internal knowledge of a trained neural network in an attempt to explain the behavior of a given neural network (or the system that it represents) by converting the network into a set of rules. Subsequently, the rules may be used instead of the neural network, since they are closer to human understanding.

The more accurate your rule extraction, the better it matches your neural network. The predictions of a network can be explained through the rules extracted from it, making a neural network less of a black box of unexplained answers and more of an understandable process [8]. By using rule extraction, the degree of matching between network responses and rule classification allows the developer and user to understand the neural network inner workings and be confident in what it has learned [9].

2.3.3 Types of Rule Extraction

Rule Extraction is a technique that can be used with several different types of classification techniques, such as decision trees, support vector machines, and neural networks. For now, the focus will be on the algorithmic methods that have been developed using the three types of rule extraction: pedagogical, decompositional, and eclectic. Each type of rule extraction focuses on different aspects of the neural network.

The pedagogical, or black box, approach creates the rules by paying close attention to the input-output relationships, attempting to mirror the way the neural networks understand the relationship between the input-output signals as close as possible. The pedagogical approach to algorithms is typically the fastest approach because it does not take the time to scrutinize or analyze the internal weights of the network. However, because of this, this approach is also less likely to accurately obtain all of the rules that help describe the network's behavior [24]. The main advantage of using the pedagogical approach lies in the fact that it is applied to most neural networks, whereas the decomposition approach can be more limited [11].

The decompositional, or white box, approach can be more difficult than the pedagogical; however, the extra effort it takes helps improve the accuracy of the rules extracted. The de-

compositional approach takes a look at the internal weights and connections that make up the network in order to more accurately extract rules [24]. The advantage of this approach is that the analyzing of the internal weights and makeup help create an accurate set of rules for the entire neural network [11].

The eclectic, or "mixed box," approach combines the ideas of the pedagogical and decompositional methods. Generally, this can take longer than the pedagogical approach because of the decompositional aspects it uses, but like the decomposition approach, the results are likely to be much more accurate than the pedagogical [24].

There are several types of rules that can be formulated from the rule extraction process. The rules can take on the form of an IF..THEN...ELSE statement, or an M-of-N statement, or If "a variable is in range" THEN "statement"[6, 25, 26].

2.3.4 DCS Rule Extraction Algorithms

The original DCS Rule Extraction algorithm was developed to generate human-readable rules that could be examined and understood by a person [11]. The second rule extraction algorithm was developed to completely capture the internal structure of the network and agree with the network 100 percent of the time [11]. This algorithm generates deterministic rules from a trained DCS that can be used in a two-step process to help refine the rules generated by the original algorithm. Although these rules are not easily understood by a human, they can be implemented and function like a fixed neural network. Both algorithms were previously applied to real-world data [18]. In this paper we will focus on the human-understandable rule extraction algorithm.

The human-understandable algorithm developed for extracting rules from the DCS was a modification of the LREX algorithm that was used to extract rules from a radial basis function neural network. Before performing the rule extraction, the DCS was put into operation for some time (learning on inputs or training), the weights on the connection were then used as input to the rule extraction algorithm. During operation, the BMU (centroid of a region) corresponding to each data point presented is recorded and then these are used as inputs to the algorithm. The data that has been presented to the neural network during operation (or training) is divided into regions based on the BMUs that have been recorded. Then for each BMU, x_{lower} is the smallest value of the independent variable and x_{upper} is the largest value of that independent variable that has that same BMU. These two numbers form bounds for the intervals in the antecedent of the rule (i.e. variable

$\geq x_{lower}$ AND $\leq x_{upper}$). An interval is determined for each of the independent variables and the statements are connected by ANDs to form the full antecedent. The algorithm for extracting human-readable rules from the DCS is presented below.

Human Understandable Rule Extraction Algorithm for DCS:

Input:

Weights from a trained DCS (centers of Voronoi region)

Best matching unit for each input

Output:

One rule for each cell of the DCS

Procedure:

Apply input stimulus to DCS from training data

Record BMU for each input

Collect all inputs with common BMU to form cell

For each weight (w_i)

For each independent variable

x_{lower}= min{x | x has BMU= w_i}

x_{upper}= max{x | x has BMU= w_i}

Build rule by:

Independent variable in [x_{lower}, x_{upper}]

Join antecedent statements with

AND

Dependent variable = category

OR

Dependent variable in [y_{lower}, y_{upper}]

Join conclusion statements with

AND

Write Rule

Figure 6 shows a two-dimensional depiction of how the rules fit with the Voronoi structure of the DCS. The human-understandable rules do not fully capture the shape of the region, but they approximate the region and encompass all data that is in the region. The downside with this

approximation is that rules can sometimes overlap each other or sometimes overlap into another region. When the data is in a higher dimension, the overlap is less likely.

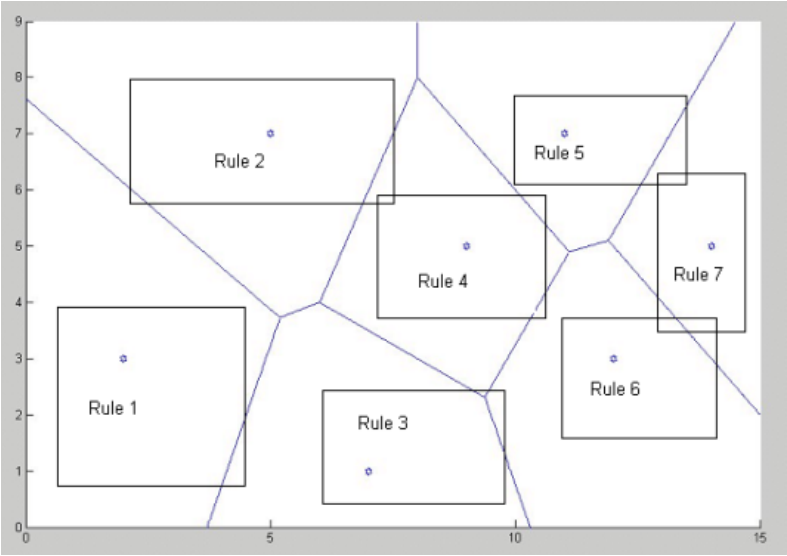

Figure 6. Voronoi Diagram

2.4 Test Results

2.4.1 Benchmark Testing with Iris Data

The DCS neural network was first trained on the Iris Data. The rule extraction algorithm, written in MATLAB, was employed. The Iris data set is available from the UCI machine learning database and has four independent variables (petal width, petal length, sepal width, sepal length) and one dependent variable (type of Iris). This data set is widely used to test different algorithms. The set is interesting because it is not linearly separable. After training the DCS on the Iris data, rules were extracted by applying the algorithms to the weights and connection matrix. Below is an example of the types of rules extracted from the DCS neural network for the Iris data set.

IF (SL $\geq$ 5.6 AND $\leq$ 7.9) AND (SW $\geq$ 2.2 AND $\leq$ 3.8) AND (PL $\geq$ 4.8 AND $\leq$ 6.9) AND (PW $\geq$ 1.4 AND $\leq$ 2.5) THEN Virginica

2.4.2 Forest Fire Data Set

The Forest Fire data set is available from the UCI machine learning [17]. It is composed of 517 instances and 13 attributes of data from the Montesinho Park in the Northeast region of

Portugal. The aim is to use the data to predict the likelihood of a forest fire given the conditions outlined in by the parameters.

The 13 attributes included in the Forest Fire data set:

- X - x-axis spatial coordinate within the Montesinho Park map:1 to 9
- Y - y-axis spatial coordinate within the Montesinho Park map: 2 to 9
- month - month of the year: 'Jan' to 'Dec'
- day - day of the week: 'mon' to 'sun'
- FFMC - FFMC index from the FWI system:18.7 to 96.20
- DMC - DMC index from the FWI system: 1.1 to 291.3
- DC - DC index from the FWI system: 7.9 to 860.6
- ISI - ISI index from the FWI system: 0.0 to 56.10
- temp - temperature in Celsius degrees: 2.2 to 33.30
- RH - relative humidity in %: 15.0 to 100
- wind - wind speed in km/h: 0.40 to 9.40
- rain - outside rain in mm/m2: 0.0 to 6.4
- area - the burned area of the forest (in hectare): 0.00 to 1090.84 (for our purposes coded as 0 no fire or 1 fire occurred)

2.4.3 Analyzing Forest Fire Data

The DCS software allows for the configuration of the neural network. One of the parameters that can be chosen is the number of cells (or Voronoi regions) that will be developed during training. There is the ability to allow the neural network to grow without bound, but the result in this situation would be an overfit the neural network to the training data and provide poor generalization to future data. The best configuration is the least number of cells with the best accuracy. This allows for more general rules that can then be used more successfully with data that is not the training data.

The table below shows how the accuracy of the neural network's predictive abilities for the Forest Fire data changed with the number of cells allowed to grow in the DCS. The number of cells is treated as an independent variable and modified to create a DCS neural network with the best ability to predict forest fire occurrence. The neural network is trained on a random 75% of the data set and human-understandable rules were extracted. Then the remaining 25% of the data set was

used as test data to check. The training and testing is run multiple times with different partitions of the data set each time; then an average is computed. The accuracy is judged in two ways. First, the accuracy of the neural network itself at predicting that forest fire is checked (NN accuracy). Second, the test data was processed by the rules to determine how accurate the rules were in predicting forest fire would occur (Rule accuracy).

Table 1. Determining the Number of Cells for Best Accuracy

Number cells	**NN accuracy**	**Rule accuracy**
16	0.67647	0.61765
14	0.64706	0.66667
12	0.73529	0.75758
10	0.67647	0.70588
8	0.66176	0.65306
6	0.67647	0.71698
4	0.72059	0.72414
3	0.66176	0.65574

As we see here in Table 1, when the neural network was restricted to growing only 12 cells, the neural network and the rules were the most accurate. Appendix A shows the complete set of human-understandable rules that were extracted from the neural network producing the best results when the neural network was restricted to 12 cells.

After the optimal number of nodes is established, then different subsets of the variables are used to determine if a smaller number of input variables can be used to accurately determine the output. All subsets from size two to number of independent variables (12) were run using allowing the network to grow 12 nodes. This is a large number of sets, in this case 122, so this is process is automated. Below in Table 2 the best subsets are listed.

Table 2. Determining the Best Subset of Variables for Best Accuracy

Parameters	NN Accuracy	Rule Accuracy
day, rain	0.81538	0.65672
x, y, month, day, ffmc, dmc, dc, isi, temp, rain	0.61538	0.59259
x, day	0.59231	0.54808
day, wind	0.58462	0.55963
day, wind, rain	0.58462	0.53488
x, y, month, day, ffmc, dc, isi, rh, wind, rain	0.57692	0.55446
x, y, month, day, ffmc, dc, isi, temp, rh, wind	0.57692	0.56122
x, day, rain	0.56154	0.53097
x, y, month, day, ffmc, dc, temp, rh, wind, rain	0.56154	0.55238
x, y, month, day, ffmc, dc, isi, temp, rh, wind, rain	0.54615	0.57009
x, y, isi, rh	0.53077	0.55172
x, y, isi, rh, rain	0.53077	0.54839
y, month, day, ffmc, dc, isi, temp, rh, wind, rain	0.50000	0.52427

From this table, it can be seen that using the two variables day and rain result in the most accurate classification of the output variable (fire occurred).

2.5 Conclusions

When obtaining data for events like forest fires, there are several potential methods that can be used. By utilizing meteorological approaches, satellites, and infrared/smoke scanners, the data can better predict when and where a fire could occur. Temperature, wind, relative humidity these factors come into play when analyzing the meteorological aspect of it. All of these methods help strengthen fire management techniques [15].

The Dynamic Cell Structure (DCS) neural network helps make the neural net- working process more understandable and helps understand the rules for the classification process in forest fire data. We show how this technique can be used to extract understandable forest fire classification rules that could be used to help predict the occurrence of forest fires.

III. PAPER 2: RULE INSERTION TECHNIQUE FOR A DYNAMIC CELL STRUCTURE NEURAL NETWORK TO IMPROVE PERFORMANCE

3.1 Abstract

This paper discusses the idea of capturing an expert's knowledge in the form of human understandable rules and then inserting these rules into a dynamic cell structure (DCS) neural network. The DCS is a form of self-organizing map that can be used for many purposes, including classification and prediction. This particular neural network is considered to be a topology preserving network that starts with no pre-structure, but assumes a structure once trained. The DCS has been used in several mission and safety-critical applications, including adaptive flight control and health-monitoring in aerial vehicles. The approach is to insert expert knowledge into the DCS before training. Rules are translated into a pre-structure and then training data is presented. This idea has been demonstrated using the well-known Iris data set and it has been shown that inserting the pre-structure results in better accuracy with the same training.

3.2 Introduction

Artificial Intelligence plays a key role in developing devices that can analyze situations like a human. Developing systems with a set of guiding knowledge that is then able to learn from new experiences to refine that knowledge is key to simulating human decision making. Neural-Symbolic learning systems play a key role by combining the benefits of both the neural and symbolic paradigms of artificial intelligence [27].

Accuracy and confidence are very important for safety-critical uses of neural networks. The rationale for using rule insertion is that expert knowledge represented in a set of rules, which could possibly be incomplete or incorrect due to insufficient knowledge, can be inserted to initialize a neural network before training is applied. The initial knowledge is inserted using a rules-to-network algorithm. The initial symbolic knowledge that is inserted becomes the initial neural network structure. This process creates a "neural-symbolic" system utilizing a combination of theoretical and empirical data. The initial symbolic knowledge then goes through a stage of training and refinement. Upon completion of training, rules are then extracted again

for comparison. This three step process can assist in ensuring the most accurate output, reduce training time, and also provide confidence by allowing developers and users to better understand the internal workings of the neural network through the inspection of the rules.

Neural networks are not recognized for their capacity to use symbolic knowledge, but rather from their capability "to be trained from data". They have become an acknowledged tool in machine learning toolboxes. Usually, neural networks "readily" store knowledge in distributed internal weights, not in symbolic form. Although neural networks are commonly used for generalizations, other applications may require the knowledge be used in symbolic form [28]. Therefore, investigation into the interchange of information between connections and symbolic representations is necessary for effective learning.

Kurd, Kelley, and Austin [12] discussed that the dilemma with the use of artificial neural networks in a safety-critical situation is that the software lifecycle relies on determining the specifications at the initial phase of development. This is not supported if the neural network starts with no initial internal structure, which is the case with the DCS self-organizing map that is the focus of our research. The lifecycle of the hybrid systems like the one we are suggesting can be described by the "W" model (Figure 7) [12].

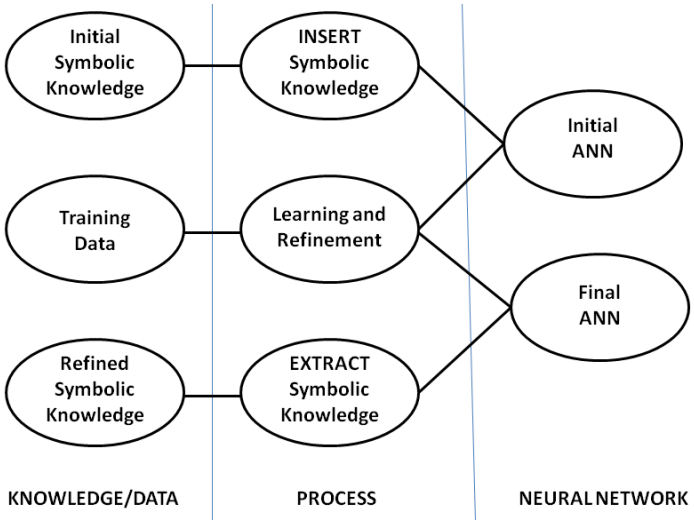

Figure 7. Rule Insertion/Extraction "W" Model [12].

In Figure7 the following levels are depicted:

Symbolic Level: This level is associated with symbolic information and deals with analysis in terms of symbolic knowledge.

Translation Level: This level is where symbolic knowledge and neural architectures are joined or separated.

Neural Learning Level: This level uses neural learning to adapt and refine symbolic knowledge.

In the past others have explored the idea of combining rule based knowledge and neural learning. Towell and Shavlik [13, 29] introduced the new algorithm named Knowledge-Based Neural Network (KBANN). He felt that this algorithm would improve the learning speed because it is not ignoring any information. He described this algorithm as a way to address the problems of training "deep" networks. Figure 8 shows the process that Towell and Shavlik used for Rule Insertion. KBANN is a hybrid learning system and is more effective at classifying examples compared to other machine learning algorithms. Unfortunately, the networks created by KBANN, known as KBANN-nets, have "deep" network properties that are not well suited to work with backpropagation. To address this issue, the Desired Antecedent Identification (DAID). algorithm was introduced

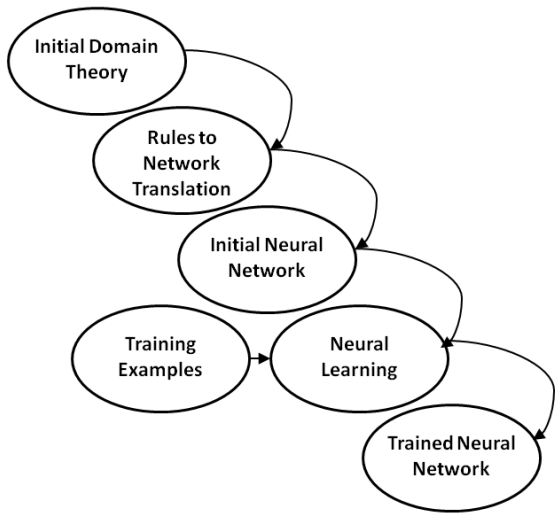

Figure 8. Towell and Shavlik method for Rule Insertion

The DAID was motivated by two observations. First, the “deep” neural networks cause trouble to the neural leaning techniques because error signals become diffused. Second, it had been shown that KBANN is most effective when antecedents are ignored by the network. The DAID aids in this issue by lessening error. Ultimately the DAID is most useful in deep structures due to its learning bias towards learning at the bottom, whereas backpropagation is most useful in shallow structures due to its bias towards learning at the top of chains.

Another idea from Giles and Omlin [14] discusses methods for extracting, inserting and refining symbolic grammatical rules for recurrent networks. The issues also discussed in this paper include how rules are inserted into the recurrent neural network, how training and generalization is affected, and how the rules can be checked in order for correction. The method Giles and Omlin [14] devised requires the network size to exceed the number of Deterministic Finite State Automata (DFA) states. It was expected the training time would decline with rising rule strength, but the network does not easily recognize partial correct rule insertion if the rule strength is too great.

An additional aspect of symbolic knowledge extraction and insertion is rule checking, allowing for the establishment of the validity of the knowledge. Rule checking compares rules extracted from trained networks with prior knowledge. However, rule checking becomes increasingly difficult with rising rule strength when incorrect rules are inserted into a network. Further, Giles and Omlin [14] suggest that network architecture can be altered during training with symbolic guidance, and symbolic information gained from under-trained networks could prove useful in determining the current network architecture.

This paper presents an approach to inserting rules to a specific neural network structure, the DCS neural network that has been used in several safety-critical applications including adaptive aircraft control [30] and on-board health state awareness for Unmanned Aerial Vehicles(UAVs) [31]. Section 2 discusses the process by outlining the DCS structure, the rule extraction process and the rule insertion process. Section 3 discusses the application of the process to a common benchmark data set. Section 4 provides the result of the experiment and Section 5 provides some conclusions that can be drawn.

3.3 The Process

3.3.1 Structure of the Dynamic Cell Structure Neural Network

As previously mentioned, one type of self-organizing map is called the Dynamic Cell Structure (DCS) neural network. [1, 2, 3] This type of neural network is designed as a topology representing network whose role is to learn the topology of an input space. The DCS neural network partitions the input space into Voronoi regions (Figure 9). The neurons within the neural network represent the reference vector (centroid) for each of the Voronoi regions (cells). The connections between the neurons, c_{ij}, are then part of the Delaunay triangulation connecting neighboring Voronoi regions through their reference vectors.

Given an input to the DCS, v, the best matching unit (BMU) is the neuron whose weight, w, is closest to v, and the second best matching unit (SEC) is the neuron whose weight is the second closest to v. Along with the BMU, the neighbors of the BMU are found through the Delaunay triangulation, which connect the centers of the Voronoi regions if they share a boundary. During adaptation, adjustments are made to the BMU and SEC neurons within the BMU neighborhood (NBR) based on the input.

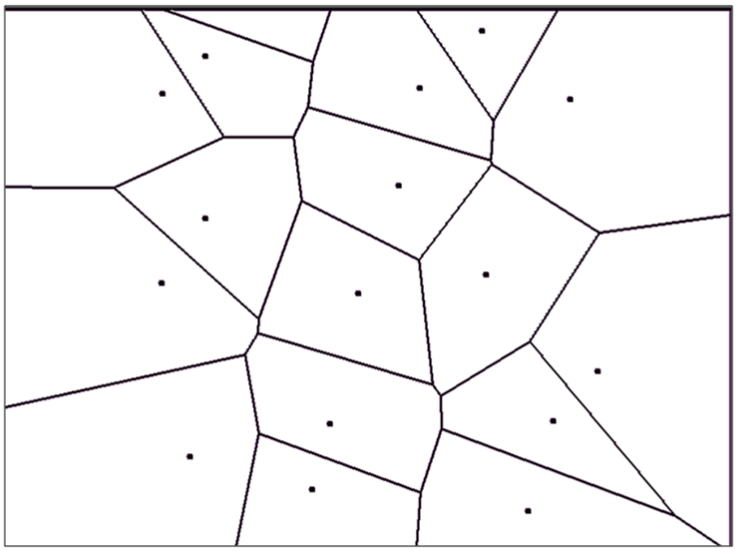

Figure 9. Voronoi Diagram.

The DCS algorithm consists of two types of learning rules, Hebbian and Kohonen, Equation 1, Equation 2, and Equation 3. These learning rules allow the DCS neural network to change its structure to adapt to inputs. The ability to adjust neuron positions and add new neurons into the network gives the DCS neural network the potential to evolve into many different configurations.

$$c_{ab} = \begin{cases} 1 & a \in [BMU, SEC] \ \wedge b \in [BMU, SEC] \\ \alpha \cdot c_{ab} & \alpha \cdot c_{ab} > 0 \\ 0 & \alpha \cdot c_{ab} < 0 \\ 0 & a = b \end{cases} \quad (1)$$

$$\Delta w_{BMU_i} = \varepsilon_{BMU}(v_i - w_{BMU_i}) \quad (2)$$

$$\Delta w_i = \varepsilon_{NBR}(v_i - w_i) \quad (3)$$

Below, the rule extraction process, DCS structure to Human Understandable Rules, is discussed first, since this process was established for the DCS neural network in previous work [6]. Then we discuss how the rule insertion process, Human Understandable Rules to DCS structure, would work.

3.4 Rule Extraction

A negative seen when using neural networks is the fact that the knowledge acquired during training is coded as weights or activation values. This results in very few tools capable of validating neural network techniques. By using rule extraction, a developer can, at least in part, determine the internal knowledge of the trained neural network and validate that what has been learned matches expert understanding and intended need [32].

Rule Extraction techniques have been developed for many neural network types [33, 34, 35]. This is a process that can help make neural network output more understandable by representing the internal knowledge of the neural network as a set of rules. The predictions or classifications of the network can be explained through the rules extracted from it, making neural networking less of a black box of unexplained answers and more of an understandable process [8]. Accuracy of the rules is generally judged by their agreement with the neural network [7].

The process of extracting a list of human readable rules from the cell list output of the DCS neural network is straight-forward. Each data point is assigned a BMU, the BMU is a centroid of a Voronoi region (cell). Then for each cell there is a list of points that are assigned to that region. From this list of points the minimum and maximum values are determined in each dimension and these values are used to create a bounding box in the parameter space. This bounding box is the smallest such n-dimensional box that contains each point in the cell. Each rule is simply a list of the boundaries of these bounding boxes. In pseudocode the algorithm is as follows:

```
For each(cell in cells):
        For each(datapoint in cell):
                For each(param in datapoint):
                        maxes[cell,param]=max(maxes[cell,param],
                        datapoint[param])
                        mins[cell,param]=min(mins[cell,param],
                        datapoint[param])
```

The following is an example of a list of extracted rules. The data set used to train the DCS in this case was the IRIS benchmark data that will be described later in the paper, with four input variables and three output types.

```
RULES FOR CELL1
IF (sepal_length>=6.7 AND <=7.4) AND
IF (sepal_width>=2.8 AND <=3.6) AND
IF (petal_length>=5.7 AND <=6.1) AND
IF (petal_width>=1.6 AND <=2.5)
        THEN...2

RULES FOR CELL2
IF (sepal_length>=4.3 AND <=5) AND
IF (sepal_width>=2.3 AND <=3.6) AND
IF (petal_length>=1 AND <=1.6) AND
IF (petal_width>=0.1 AND <=0.3)
        THEN...0

RULES FOR CELL3
IF (sepal_length>=6.3 AND <=6.9) AND
IF (sepal_width>=2.5 AND <=3.4) AND
IF (petal_length>=5.1 AND <=6) AND
IF (petal_width>=1.8 AND <=2.5)
```

THEN...2

RULES FOR CELL4
IF (sepal_length>=5 AND <=6) AND
IF (sepal_width>=2 AND <=2.9) AND
IF (petal_length>=3 AND <=4) AND
IF (petal_width>=1 AND <=1.4)
THEN...1

RULES FOR CELL5
IF (sepal_length>=5.5 AND <=6.1) AND
IF (sepal_width>=2.6 AND <=3) AND
IF (petal_length>=4 AND <=4.5) AND
IF (petal_width>=1 AND <=1.5)
THEN...1

RULES FOR CELL6
IF (sepal_length>=7.3 AND <=7.7) AND
IF (sepal_width>=2.6 AND <=3.8) AND
IF (petal_length>=6.3 AND <=6.9) AND
IF (petal_width>=1.8 AND <=2.3)
THEN...2

As mentioned previously, the rules make up the bounding boxes that loosely approximate the n-dimensional Voronoi regions. To illustrate the idea, Figure 10 is an example of a two-dimensional Voronoi diagram that uses two of the variables, sepal length and sepal width, from the rules in the preceding list. The coordinates of the centroids for these two variables were used to create the Voronoi diagram.

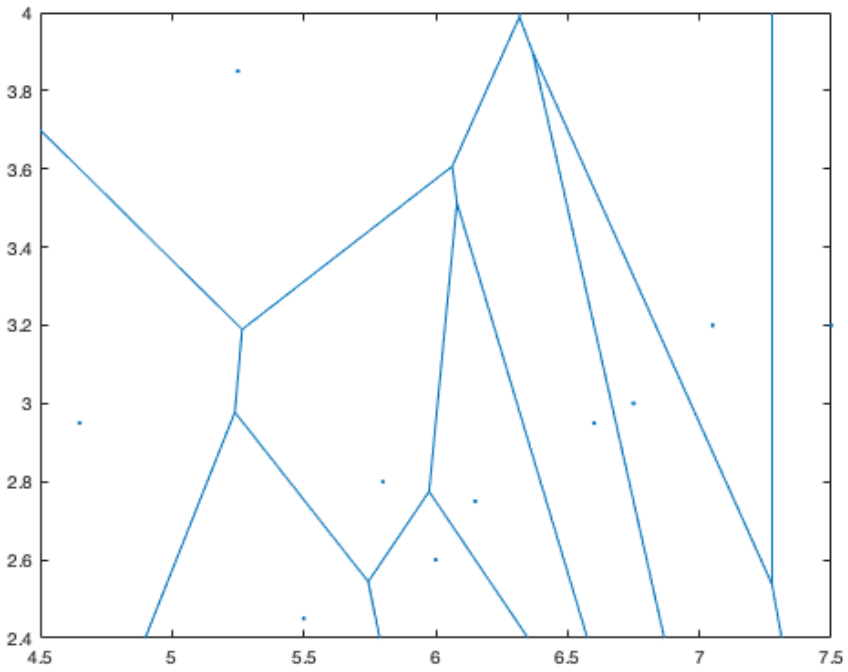

Figure 10. Voronoi diagram using a 2-dimensional projection of the centroids of the DCS.

Figure 11 shows some of the bounding boxes for the extracted rules in the previous list overlaid on the Voronoi regions. Observe how the bounding boxes approximate the cells of the Voronoi diagram, even though it is limited to just two dimensions. It can also be noted at this time that the approximation is not exact, there was a more exact rule extraction method developed [36], however the rules that were the output of that method are not considered human understandable," but were more mathematical.

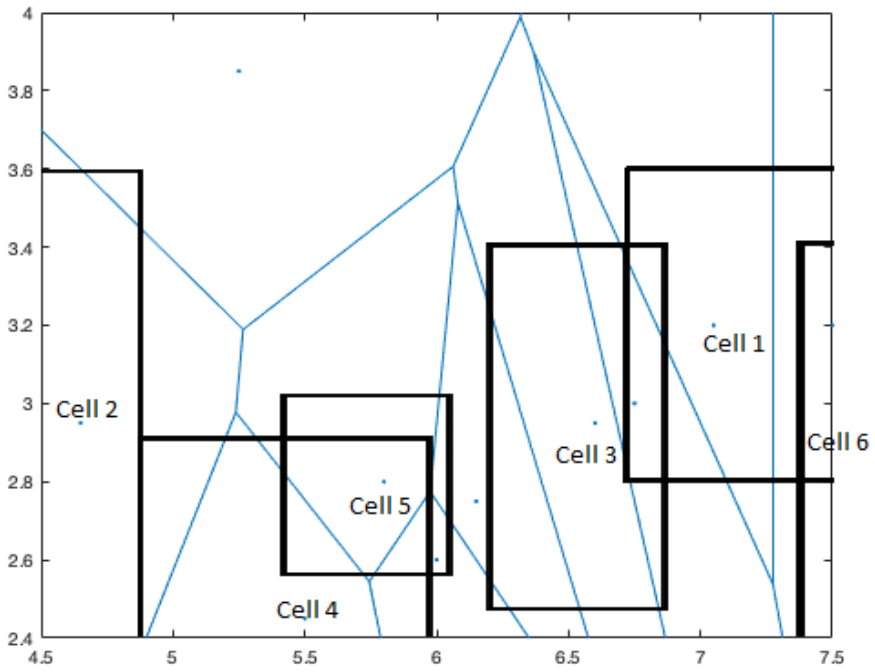

Figure 11. Voronoi diagram of a 2-dimensional projection of the centroids of the DCS with some of the extracted rules bounding boxes overlaid.

Such drastic overlapping does not occur when the rules are represented in all four dimensions.

3.5 Rule Insertion

Rule insertion is the process of supplying internal knowledge to influence the formation of the neural network before training occurs. The knowledge influences the formation towards a potential classification structure, which is used in initializing the neural network, and then trained upon, allowing the rules to be refined.

The hypothesis is that the neural network with rules inserted should be able to be trained faster and be more accurate than the original neural network. The human readable rules can be represented simply as a collection of labeled convex subspaces inside a parameter space, where the label is the category assigned to each subspace. These subspaces are described by a series of if-then statements for each input variable. For example, "if a is less than x and x is less than b AND c is less than y and y is less than d, then the dependent variable belongs to category 1". Using the boundaries of these convex subspaces (in this case a rectangle or bounding box), the rules are converted into a collection of centers for the bounding box or centroids for a Voronoi

region. These Voronoi centroids become the neurons of the DCS and provide the initial starting point for neural network training. The DCS usually starts with two or more randomly placed neurons and then either modifies their positions or "grows" by adding additional neurons based on the data.

Now suppose the rule list give previously is not the result of training the DCS, but for example was given to us by an expert botanist. Next, suppose we want to insert these rules to give the DCS some prior knowledge on which to train. In this case, we would take each rule and determine the middle values for each parameter. This n-dimensional point then becomes the centroid for a Voronoi region or a neuron of the DCS. The list of centroids is taken and directly used as the initial set of neurons for the DCS. The corresponding centroid list for the previous list of rules would look like:

{7.05, 3.2, 5.9, 2.05}, {4.65, 2.95, 1.3, 0.2}, {6, 2.6, 5.2, 1.7}, {6.75, 3, 4.7, 1.5}, {6.6, 2.95, 5.55, 2.15}, {5.5, 2.45, 3.5, 1.2}

The output 0, 1, or 2 for the rules would also be stored associated with the centroid (neuron).

In order to visual this data, we use two dimensions and take sepal length and sepal width as the horizontal and vertical axes (respectively). In Figure 12, we can see the boxes that depict the rules and the centers of the boxes that become the centroids of the Voronoi regions.

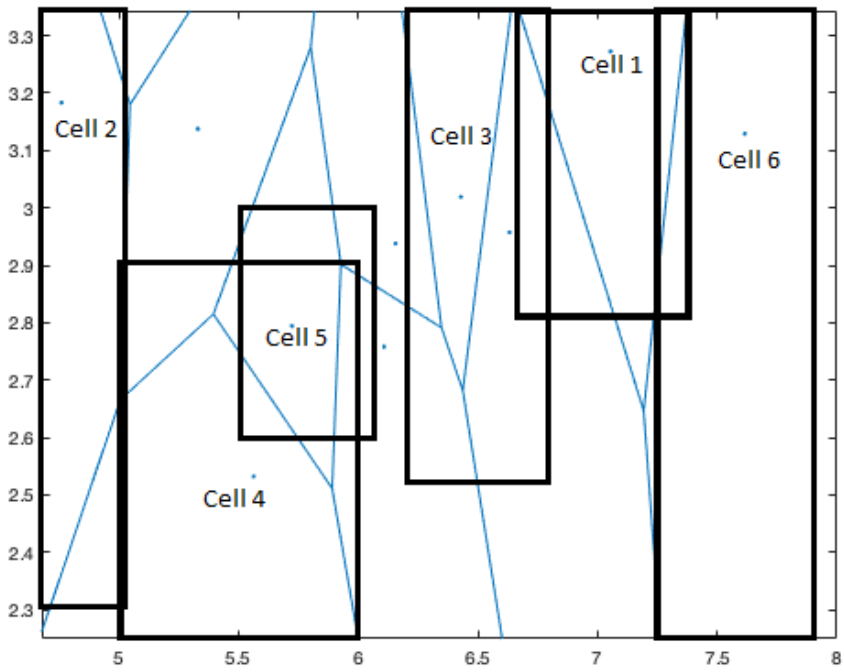

Figure 12. Rule Set Depicted as Boxes in 2-dimension Projection Overlaid on Voronoi Diagram Constructed from the Box Centers

The centroids given above produce the Voronoi regions in Figure 13. We note that Figure 13 Voronoi diagram is not exactly like Figure 10 Voronoi diagram, but they have some similarity in structure. Recall, the structure in Figure 10 resulted from training and the structure in Figure 13 resulted from using a set of rules to develop the structure.

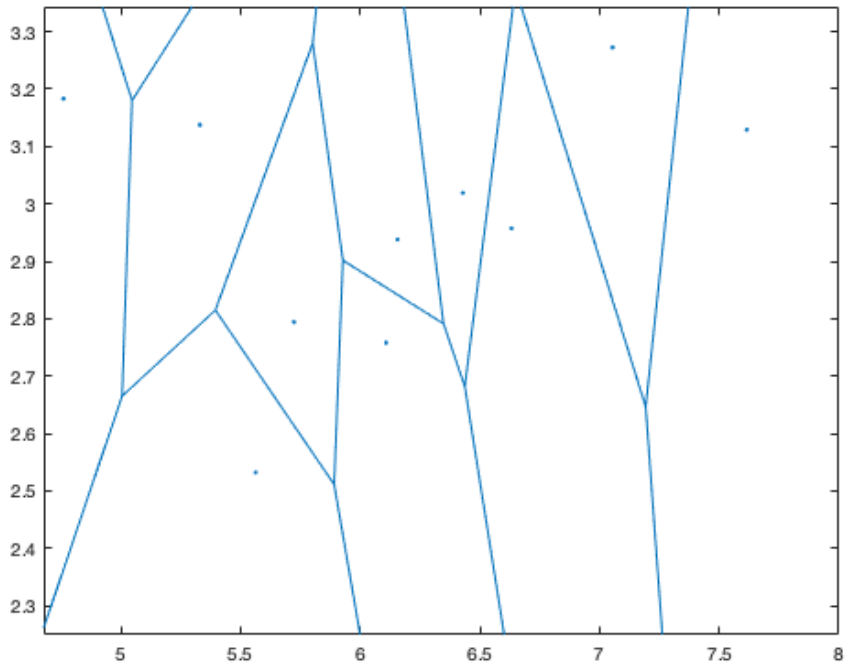

Figure 13: Voronoi Diagram of a 2-dimension Projection of the Centroids for an Inserted Rule Set

3.6 Application of Process to Benchmark Data Set

3.6.1 Iris Data Set

One of the most popular machine learning benchmark data sets is the Iris Data set. The problem to be solved is to learn which category an Iris flower belongs to based on four measurements: sepal length, sepal width, petal length and petal width. The three Iris categories are: Setosa, Versicolour, and Virginica. The dataset is available from the UCI Machine Learning Repository. It is composed of 150 instances divided evenly between the three categories (i.e. 50 instances per category.)

3.6.2 Comparing the Results

In this section we will test the efficacy of the rule insertion by first training the DCS neural network with no pre-knowledge (starting with the configuration of two random neurons) and training the DCS with inserted pre-knowledge (rule set inserted into neural network structure

as a set of starting neurons). Rule Insertion relies on the processes of engaging an expert to help formulate an initial set of rules. In the case of this proof of concept study using a benchmark Iris data set, no expert was available to construct a set of rules, so a "typical" set of rules was used. The rule set used as the pre-knowledge in the test was similar to rules sets that were extracted; the bounding values for the parameters were approximated in order to provide a starting set of neurons.

For each training epoch, the DCS neural network was trained using a random 75% of the data points from the IRIS data. The remaining 25% of the data points were used to test the accuracy of the resultant neural network. The "neural network accuracy" was calculated as the percentage of data points that were correctly classified by the neural network. In addition, each time the DSC was trained, Human Understandable rules were extracted using the process described earlier. The extracted rules were then tested and the "rule accuracy" was calculated as the percentage of data points that were correctly classified by the set of extracted rules. To avoid overfitting, the DCS was limited to only grow to the size of four neurons, which leads to only four cells.

For testing whether the network would be more accurate being initialized in the default way or initialized with the inserted rules, two experiments were conducted. The DCS NN was developed in both ways, trained using the Iris data ten times, rules were extracted at the end of each training. To compare the two methods of initialization, the accuracy of the trained DCS NN to predict Iris type and the accuracy of the extracted rules to predict Iris type were compiled.

First, the DCS NN was created with the default initialization of two random neurons. The DCS was trained on the Iris data ten different times. When the DCS was trained with the default initialization, the accuracy for the neural network prediction was on average \$92.4±1.86% and the average prediction from the set of the extracted rules themselves averaged \$90.2±1.66%.

Second, the DCS NN was created with the rules inserted. This initialization started with several nodes that were based on the rule set used (same rule set used each time). The DCS was again trained on the Iris data ten different times. When the DCS was trained with the rule-based

initialization, the accuracy for the neural network prediction was on average $94.7±1.57% and the prediction from the set of the extracted rules themselves averaged $94.2±1.28%.

This was an improvement of 2.5% for the network prediction and 2.2% for the extracted rule prediction.

3.7 Conclusions

Several methods of rule extraction from the DCS neural network had already been developed [36], but there was no previous rule insertion process investigated. Our research focused on developing a method for inserting rules into a DCS neural network structure. In this paper we determine a method for rule insertion for this type of neural network and tested its usefulness to produce results on a benchmark data set. These findings show that there is great potential for this technique to improve the accuracy of the neural network and also improve the accuracy of any rules extracted.

This opens up numerous possibilities for creating more efficient and more accurate neural networks. The initialization of the DCS with "expert" rules allows the neural network to come to a better solution in the same time, than can be developed by just training alone.

This DCS neural network has been used in several mission and safety critical applications, namely adaptive aircraft control [30] and on-board health state awareness for Unmanned Aerial Vehicles (UAVs) [31]. The ability to allow a developer to work with an expert to develop a better Neural-Symbolic system is important to further the usefulness of this neural network type.

IV. PAPER 3: IMPROVING PERFORMANCE BY EMBEDDING EXPERT KNOWLEDGE IN A SELF-ORGANIZING MAP NEURAL NETWORK

4.1 Abstract

This paper describes the process of capturing an expert's knowledge in the form of human understandable rules and then inserting these rules into a self-organizing map neural network. The Dynamic Cell Structure (DCS) is a form of self-organizing map that can be used for many purposes, including classification and prediction. This particular neural network has been used for various purposes including accommodating faults in a flight control system, health monitoring for an unmanned aerial vehicle, and classification of data. The DCS is considered to be a topology preserving network that starts with no pre-structure, but assumes a structure once trained. This paper explores applying the DCS to classifying social science data. The approach is to talk to an expert in the field who is familiar with the data. The expert provides knowledge and that knowledge is formed into if then rules, then these rules are embedded in DCS before training. This idea will be demonstrated on a set of social science data that is used to determine factors used to predict high mortality in an area. The authors have found, that with this data set, starting with the pre-knowledge embedded can provide increased accuracy, compared to simply training the neural network on the raw data.

4.2 Introduction

Artificial Neural Networks are acknowledged as valuable tools for machine learning. These networks come in many varieties, feed-forward, recurrent, self-organizing maps, etc., and are typically known for their capacity "to be trained on data", not for their understandable structures. Neural networks usually store knowledge in distributed internal weights, not in symbolic form. Although neural networks are commonly used for generalizations, other applications may require the knowledge be used in symbolic form [37]. Therefore, investigation into the interchange of information between connections and symbolic representations is necessary for effective learning.

Accuracy and confidence is very important for many uses of neural networks. The rationale for embedding expert knowledge by using the technique of rule insertion is to be able to

begin with some pre-knowledge that is deemed to be accurate, but possibly incomplete, and then train the neural network on a large data set to determine if there are things that the expert may have missed. In our example, the initial knowledge of the expert is formulated into if-then type rules, which are converted to a network structure. The initial symbolic knowledge becomes the initial neural network structure, and then the data is presented to the neural network as training or refining of the initial knowledge. Once the training is completed, rules can be extracted from the neural network for the expert to inspect and validate. This process creates a "neural-symbolic" system which utilizes a combination of theoretical (expert knowledge) and empirical data (training data set). This three step process, insert–train–extract, can ensure the most accurate output, reduce training time, and also provide confidence by allowing developers and users to better understand the internal workings of the neural network through the inspection of the rules.

Many others have explored the translation between neural structures and understandable symbolic logic. Kurd, Kelley, and Austin [12] focused on the use of artificial neural networks in a safety critical situation. In these situations, the software lifecycle relies on determining the specifications at the initial phase of development. These authors described the lifecycle of the hybrid systems like the one we are suggesting as having three levels: the symbolic level, the translation level and the neural level. [12]

In the past others have explored the idea of combining rule based knowledge and neural learning. Towell and Shavlik [13, 29] introduced the new algorithm named Knowledge-Based Neural Network (KBANN). They thought that this algorithm would improve the learning speed because it is not ignoring any information. They described a process whereby initial domain knowledge were expressed as rules and then translated to neural networks structure and then training was applied. Another idea from Giles and Omlin [14] discusses methods for extracting, inserting and refining symbolic grammatical rules for recurrent networks. We also discuss how rules are inserted into the recurrent neural network, how training and generalization is affected, and how the rules can be checked in order for correction. The method Giles and Omlin [14] devised requires the network size to exceed the number of Deterministic Finite State Automata (DFA) states.

An additional aspect of symbolic knowledge extraction and insertion is rule checking, allowing for the establishment of the validity of the knowledge. Rule checking compares rules extracted from trained networks with prior knowledge. However, rule checking becomes more difficult with rising rule strength when incorrect rules are inserted into a network. Further, Giles and Omlin [14] suggest that network architecture can be altered during training with symbolic guidance, and symbolic information gained from under-trained networks could prove useful in determining the current network architecture.

McGarry, Wermter, and MacIntyre [38] examined may techniques for integrating neural networks and symbolic components into powerful hybrid systems. They argued that neural networks have unique processing characteristics that enable tasks to be performed that would be difficult or intractable for a symbolic rule-based system. However, McGarry et al go on to explain that a stand-alone neural network requires an interpretation either by a human or a rule-based system and that this motivates the integration of neural/symbolic techniques within a hybrid system. They surveyed a variety of research and point out that there are number of integration possibilities and provided an overview and evaluation of several hybrid neural systems for rule-based processing.

More recently, Chong et al. [39] apply ideas that have been used for a rule-based neural network approach to model driver naturalistic behavior in traffic. Neural network acts as a driver simulator in this study. The neural network structure proposed here has four layers. The first layer is the input layer. Each node represents a continuous state variable. The second layer is the fuzzy membership layer. States are fuzzified into linguistic terms such as: ‘‘Speed is High’’ and ‘‘Speed is Low.’’ Each node is a discrete fuzzy set and has a membership function. The third layer is the fuzzy rule layer. Each node represents a fuzzy rule and is connected to a number of discrete fuzzy sets of the second layer. For each true, a firing strength function is defined to indicate its strength. The fourth layer consists of a number of action nodes. Each fuzzy rule chooses one action. The output action is the weighted average of the selected actions (where fuzzy rule strengths are the associated weights).

This paper presents an approach to inserting rules to a specific self-organizing map neural network, the Dynamic Cell Structure (DCS) neural network, which has been used in several safety critical applications including adaptive aircraft control [30] and on-board health state awareness for Unmanned Aerial Vehicles(UAVs) [31]. Section 2 discusses the Process of Embedding Expert Knowledge by outlining the DCS structure, the rule translation of rules to neural network structure, and the rule insertion process. Section 3 discusses the application of the process to a large data set of Social Science Data. This section outlines the data set, discusses how an expert was utilized to supply the initial knowledge, and how the process was applied. Section 4 provides the result of the experiment and a comparison where the DCS was trained with and without pre-knowledge. Section 5 provides some conclusions that can be drawn.

4.3 The Process

This section first describes the data set used in this study. Then the process of converting the expert knowledge into rules is explained. Next, the structure of the DCS is detailed as a precursor to the discussion of how the expert rules are then embedded in the neural network structure using Rule Insertion. Rule Insertion is the process of supplying internal knowledge to influence the formation of the neural network before training occurs. The knowledge influences the formation towards a potential classification structure, which is used in initializing the neural network, and then trained upon, allowing the rules to be refined.

4.3.1 Data Set

We tested the process of inserting expert rules into a neural network that would be used to analyze a large set of social science data. The data set contains a set of county-level variables that could be analyzed with respect to the part they play in the high mortality rate in the area. The data set has been studied by James and Cossman [40, 41] (and others) and we were given access to the data and also to the expert, Cossman. This allowed us to discuss previous findings and explore pre-determined ideas based on previous research.

Originally, to construct the data set two data sources were used. First, the mortality trend data from the Compressed Mortality File of the National Center of Health Statistics (CMF/NCHS) [42, 43, 44], a controlled access database documenting deaths by country, year,

state, county of residence, race, sex, age of death, and cause of death by International Classification of Diseases (ICD) Codes. [42, 43, 44] The analytical sample is the total number of US deaths from 1968 to 2012 (N=98,304,544). In 2012, there were 3,105 counties or county equivalents included in the data after Virginia's independent cities and other independent units were collapsed into the respective county comparisons.

Age-adjusted, all-cause death rates are calculated using the 2000 Standard Million's eleven age categories by years (less than 1, 1-4, 5-14, 15-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75-84, greater than 85). The proportion of the total population for each age group is used as a weight in the age-adjusted mortality rate calculations. Using this method, the urban and rural mortality rates are based on the same standard population distribution, permitting direct county comparisons. [45]

The second data source was the Area Health Resource File's (AHRF) [46] county-level estimates of population, socioeconomic status, and health care infrastructure (2000-2007). These predictors of mortality precede the mortality rate (i.e, 2012) in the multivariate analysis to account for the lag between exposure to social conditions and death.

The researchers also modified the data of several of the variables in order to put the variable into the same range. Most of the variables were already in percent (0-100), so some of the others that were not in this format, for example meansofexchange, were converted to percentage. There were 3058 data vectors in the final set of data, with 20 independent variables and one dependent variable.

4.3.2 Converting Expert Knowledge

To develop rule for inserting into the neural network, the expert determined how the data should be grouped. The expert described how the factors would affect mortality of a region and looked at a way to group variable into three scales, social, socioeconomic status, and access.

From the data sources mentioned above, the following variables were used and grouped into the three scales, with the age-adjusted mortality used for the dependent variable for the analyses.

Table 3. Mortality Correlation Scales

Social Scale	
religious	All denominations/groups--rates of adherence per 1,000 population
security	Rates of violent crimes per 1,000 total county population
shelter	Percent occupied housing units
healing	Percent of adults uninsured
Socioeconomic Status (SES) Scale	
edu904pl	Percent adults with 4 years of college or higher
food	Percentage of total students eligible for Free Lunch Program
work	not employed in labor force, female age 16+
meansofexchange	Per capita income in the past 12 months (changed to percentile)
Access Scale	
transport	Percent of households with 1 vehicle
info	Rate of high speed internet
play	Rate of access to recreational facilities
social	Proportion voter turnout
Dependent Variable	
ageadjrate_all	All-cause age-adjusted mortality

Once the variables were determined, rules were made for each set of variable. The rules took the form of

$$IF\ (variable_1\ \geq x_1\ and\ \leq y_1)$$
$$THEN\ Output$$

The Output in this case is 2-High Mortality, 1-Average Mortality, 0-Low Mortality. The High, Average, and Low were determined by using Quartiles. The variables were not all normally distributed, so looking at standard deviations above or below the mean was not the right approach. The expert used background knowledge to determine how the rules should be stated. For example, the rules for the Social Scale are below.

$$IF\ (religious \geq 0\ and\ \leq 38.71)\ AND$$
$$IF\ (security \geq 3.29\ and\ \leq 21.52)\ AND$$
$$IF\ (shelter \geq 0\ and\ \leq 79.6)\ AND$$
$$IF\ (healing \geq 17.7\ and\ \leq 38)$$
$$THEN\ \mathbf{2}$$

$$IF\ (religious \geq 38.71\ and\ \leq 62.49)\ AND$$
$$IF\ (security \geq 0.96\ and\ \leq 3.29)\ AND$$
$$IF\ (shelter \geq 79.6\ and\ \leq 89.9)\ AND$$
$$IF\ (healing \geq 10.9\ and\ \leq 17.7)$$

$THEN$ **1**

$IF\ (religious \geq 62.49\ and\ \leq 192.46)\ AND$
$IF\ (security \geq 0\ and\ \leq 0.96)\ AND$
$IF\ (shelter \geq 89.9\ and\ \leq 98.3)\ AND$
$IF\ (healing \geq 0\ and\ \leq 10.9)$
$THEN$ **0**

As can be seen, sometime low values of the variable correlate with high mortality. For example, low "religious" correlates with high mortality, while high "security" (rate of violent crime) correlates with high mortality. So the expert was able to give the information which was then put into rules.

4.4 Embedding Knowledge

This section explains how the knowledge of the expert, which was converted to rules, was then used to give the neural network "beginning knowledge". We will first outline the structure of the Dynamic Cell Structure (DCS) neural network. Then we will explain how rules are inserted for this particular set of data. A thorough treatment of this complete process can be found in our previous paper. [47]

4.4.1 Structure of the Dynamic Cell Structure Neural Network

One type of self-organizing map is called the Dynamic Cell Structure (DCS) neural network. [1, 2, 3] The DCS neural network partitions the parameter space into Voronoi regions (Figure 14). The neurons within the neural network represent the reference vector (centroid) for each of the Voronoi regions (cells). The connections between the neurons, c_{ij}, are then part of the Delaunay triangulation connecting neighboring Voronoi regions through their reference vectors.

Given an input to the DCS, v, the best matching unit (BMU) is the neuron whose weight, w, is closest to v, and the second best matching unit (SEC) is the neuron whose weight is the second closest to v. Along with the BMU, the neighbors of the BMU are found through the Delaunay triangulation, which connect the centers of the Voronoi regions if they share a

boundary. During adaptation, adjustments are made to the BMU and SEC neurons within the BMU neighborhood (NBR) based on the input.

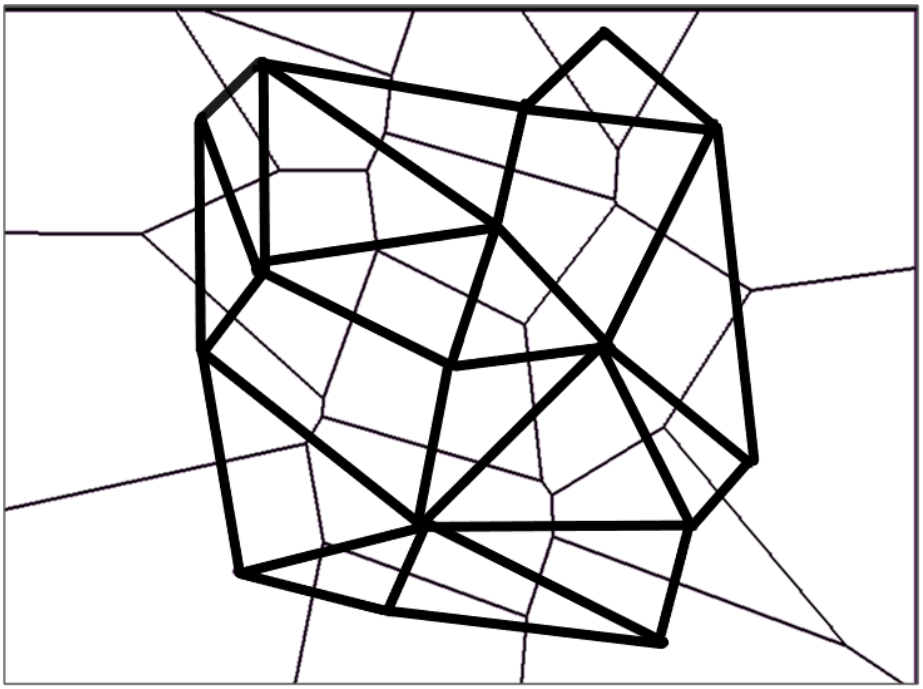

Figure 14. Voronoi Diagram with Delaunay Triangulation

The DCS algorithm consists of two types of learning rules, Hebbian and Kohonen, Equation 1, Equation 2, and Equation 3. These learning rules allow the DCS neural network to change its structure to adapt to inputs, which gives the DCS neural network the potential to evolve into many different configurations.

$$c_{ab} = \begin{cases} 1 & a \in [BMU, SEC] \ \wedge b \in [BMU, SEC] \\ \alpha \cdot c_{ab} & \alpha \cdot c_{ab} > 0 \\ 0 & \alpha \cdot c_{ab} < 0 \\ 0 & a = b \end{cases} \quad (1)$$

$$\Delta w_{BMU_i} = \varepsilon_{BMU}(v_i - w_{BMU_i}) \quad (2)$$

$$\Delta w_i = \varepsilon_{NBR}(v_i - w_i) \quad (3)$$

4.4.2 Insertion of Expert Knowledge in to the DCS

Typically, the DCS is initialized with two random points in the parameter space. Then the training data is presented to the neural network and it begins to conform the structure to the data. The hypothesis is that the neural network with rules inserted should be able to be trained faster and be more accurately than the original neural network.

The expert rules that were given in IF/THEN format can be represented simply as a collection of labeled convex subspaces inside a parameter space, where the label is the category assigned to each subspace. For example, consider the rules defined previously by the expert.

$IF\ (religious \geq 0\ and\ \leq 38.71)\ AND$
$IF\ (security \geq 3.29\ and\ \leq 21.52)\ AND$
$IF\ (shelter \geq 0\ and\ \leq 79.6)\ AND$
$IF\ (healing \geq 17.7\ and\ \leq 38)$
$THEN\ \mathbf{2}$

$IF\ (religious \geq 38.71\ and\ \leq 62.49)\ AND$
$IF\ (security \geq 0.96\ and\ \leq 3.29)\ AND$
$IF\ (shelter \geq 79.6\ and\ \leq 89.9)\ AND$
$IF\ (healing \geq 10.9\ and\ \leq 17.7)$
$THEN\ \mathbf{1}$

$IF\ (religious \geq 62.49\ and\ \leq 192.46)\ AND$
$IF\ (security \geq 0\ and\ \leq 0.96)\ AND$
$IF\ (shelter \geq 89.9\ and\ \leq 98.3)\ AND$
$IF\ (healing \geq 0\ and\ \leq 10.9)$
$THEN\ \mathbf{0}$

Using these rules we formed the boundaries of these convex subspaces, the rules are converted into a collection of "centers" for the bounding convex hulls and then these centers become the centroids for a Voronoi regions. In 2-dimensions the bounding shape is a rectangle, in 3-dimension it would be a rectangular prism, etc. These Voronoi centroids then become the neurons of the DCS and provide the initial starting point for neural network training.

In this case, we would take each rule and determine the middle values for each parameter. This 4-dimensional point then becomes the centroid for a Voronoi region or a neuron of the DCS. The list of centroids is taken and directly used as the initial set of neurons for the DCS. The corresponding centroid list for the previous list of rules would look like:

{19.36, 12.41, 39.8, 27.85} related to output 2
{50.6, 1.69, 84.75, 14.3} related to output 1
{127.48, 0.48, 94.1, 5.45} related to output 0

The output 0, 1, or 2 for the rules would also be stored associated with the centroid (neuron). The neural network is then initialized with these neurons and the network will be trained on the training data.

4.5 Testing Process

In this section we will test the efficacy of the rule insertion by first training the DCS neural network with no pre-knowledge (starting with the configuration of two random neurons) and then training the DCS with inserted pre-knowledge of the expert (rule set inserted into neural network structure as a set of starting neurons). For testing whether the network would be more accurate being initialized in the default way or initialized with the inserted rules, two experiments were conducted.

The DCS NN was developed in the two ways mentioned above, trained using the data set ten times, rules were extracted at the end of each training. Rules similar to the ones inserted, that captured the neural network knowledge, were extracted after each training epoch using the Rule Extraction process describe in [6, 47]. This was an important part of the process so that the extracted rules could be presented to the expert for inspection and validation. The extracted rules were then tested and the "rule accuracy" was calculated as the percentage of data points that were correctly classified by the set of extracted rules. To compare the two methods of initialization, the accuracy of the trained DCS NN and the accuracy of the extracted rules were both compiled.

4.5.1 Preliminary Testing of the Neural Network on the Data Set

After getting the data set from Cossman and before meeting to discuss the rules she would develop, we ran the DCS with 20 independent variables testing different number of nodes “cells” to see which size of the neural network would produce more accurate results. We tried five, seven, nine, 11, 13, and 15 nodes. Five and 15 nodes gave the good results.

The rest of the preliminary study was run with the DCS limited to growing five nodes. In the tables below, the average of 10 training epochs is given for the neural network accuracy to predict the correct output, the rule accuracy to predict the correct output, the agreement between the neural network and the rules on the data points that they both classified, and finally what percentage of the data could be classified by both methods.

First, with the set of 20 variables, the results can be seen below.

Table 4. Preliminary Results for All 20 Variables

% of data that NN accurately classified	% of data that rules accurately classified	% agreement between NN and rules	% of data that both classified
0.596	0.603	1	0.971

For this particular epoch, approximately 3000 of the data points were able to be classified, leaving only about 60 left unclassified. This means that the neural network and the rules could correctly classify the data approximately 60% of the time and almost all the data was classified. Below is a sample set of extracted rules from this run.

RULES FOR CELL1
IF (edu90somecol>=12.8 AND <=40.1) AND
IF (housekeeping>=1.4 AND <=25.5) AND
IF (religious>=30.6489 AND <=1924.6089) AND
IF (transport>=9.9 AND <=52.7) AND
IF (air>=0 AND <=105) AND
IF (hygiene>=0 AND <=1748.9) AND
IF (persrel>=2.2 AND <=20.3) AND
IF (security>=0 AND <=16.39) AND
IF (water>=0 AND <=7.2) AND
IF (edu904pl>=6 AND <=53.4) AND
IF (food>=0 AND <=69.0111) AND
IF (info1>=1 AND <=5) AND
IF (play>=0 AND <=57.5) AND
IF (shelter>=22.9 AND <=98.3) AND
IF (work>=0 AND <=0.13797) AND
IF (edu90hs>=13.5 AND <=46.9) AND
IF (healing>=3.8 AND <=22.6) AND
IF (meansofexchange>=24378 AND <=64381) AND
IF (refedu>=4.5 AND <=43.2) AND
IF (social>=0.35 AND <=0.85)
THEN...0

We looked for smaller data subsets that would predict mortality (dependent variable). Cossman, the field expert chose 12 variables expected to have the most effect on the dependent variable, mortality; she also grouped the explanatory variables into three sets of four, labeled SES, Social, and Access (as mentioned before).

A preliminary study was done using these 12 variables. First, all 12 were used, then two groups of four were used (eight variables), and then each group of four was run independently. See the tables below.

Table 5. Preliminary Results for 12 Variables

% of data that NN accurately classified	% of data that rules accurately classified	% agreement between NN and rules	% of data that both classified
0.727	0.732	1	0.961

For this run, 3026 of the data points were able to be classified, leaving 32 left unclassified.

Sample rules for all 12 variables

RULES FOR CELL1
IF (religious>=30.6489 AND <=1397.99) AND
IF (transport>=15.2 AND <=48.8) AND
IF (security>=0.14 AND <=13.34) AND
IF (edu904pl>=11.1 AND <=53.4) AND
IF (food>=0 AND <=64.8064) AND
IF (info1>=3 AND <=5) AND
IF (play>=0 AND <=57.5) AND
IF (shelter>=22.9 AND <=98.3) AND
IF (work>=0 AND <=0.10425) AND
IF (healing>=3.8 AND <=21) AND
IF (meansofexchange>=31377 AND <=59149) AND
IF (social>=0.48 AND <=0.82)
THEN...1

Next we tested each group 2 groups (3 groups of 8)

Table 6. Preliminary Results for 8 Variable Groups

Groups	% of data that NN accurately classified	% of data that rules accurately classified	% agreement between NN and rules	% of data that both classified
SES & Social	0.731	0.731	1	0.963
Social & Access	0.878	0.873	0.946	0.0719
SES & Access	0.707	0.708	1	0.979

Sample rules for SES & Social

RULES FOR CELL1
IF (edu904pl>=8.7 AND <=52.8) AND
IF (food>=6.6261 AND <=65.6337) AND
IF (work>=0 AND <=0.11882) AND
IF (meansofexchange>=26901 AND <=64381) AND
IF (religious>=52.38 AND <=1924.6089) AND
IF (security>=0 AND <=16.39) AND
IF (shelter>=27.8 AND <=95.9) AND
IF (healing>=4.7 AND <=22.6)
THEN...1

Sample rules Social & Access

RULES FOR CELL1
IF (religious>=117.67 AND <=511.35) AND
IF (security>=0 AND <=12.62) AND
IF (shelter>=22.7 AND <=95.9) AND
IF (healing>=7.5 AND <=34.3) AND
IF (transport>=18.2 AND <=48.7) AND
IF (info1>=1 AND <=5) AND
IF (play>=0 AND <=62.4) AND
IF (social>=0.32 AND <=0.77)
THEN...2

Sample rules SES & Access

RULES FOR CELL1
IF (edu904pl>=15.6 AND <=53.4) AND
IF (food>=0 AND <=64.8064) AND
IF (work>=0.012984 AND <=0.091734) AND
IF (meansofexchange>=34630 AND <=64381) AND
IF (transport>=9.9 AND <=52.7) AND
IF (info1>=3 AND <=5) AND

IF (play>=4.6 AND <=57.5) AND
IF (social>=0.48 AND <=0.82)
THEN...1

Then we tested each group individually.

Table 7. Preliminary Results for 4 Variable Groups

Groups	% of data that NN accurately classified	% of data that rules accurately classified	% agreement between NN and rules	% of data that both classified
SES	0.757	0.761	1	0.980
Social	0.858	0.875	0.964	0.073
Access	0.854	1	0.8	0.007

Sample rules for SES

RULES FOR CELL1
IF (edu904pl>=4.6 AND <=42.9) AND
IF (food>=0.91935 AND <=90.1203) AND
IF (work>=0 AND <=0.18708) AND
IF (meansofexchange>=19187 AND <=22890)
THEN...2

Sample rules for Access

RULES FOR CELL1
IF (transport>=9.9 AND <=52.7) AND
IF (info1>=1 AND <=5) AND
IF (play>=0 AND <=49.9) AND
IF (social>=0.27 AND <=0.86)
THEN...1

Sample rules for Social

RULES FOR CELL1
IF (religious>=649.3789 AND <=1413.7) AND
IF (security>=0 AND <=8.61) AND
IF (shelter>=38.3 AND <=98.3) AND
IF (healing>=5.1 AND <=31.6)
THEN...1

Some results were good, others were not. To improve results, we modified the data and the variables were all put in a similar scale. The raw data had variable in different formats, some were in percentages, some in dollars, some in occurrences per 1000, etc. The varying scales

caused trouble when creating convex hulls “rules” in the parameter space. To rectify this situation, we recalculated each variable into roughly the same scale, (0-100).

4.5.2 Training the Neural Network without Rules

The DCS was initialized as usual with two random centroids to begin the learning and growing process. Each of the scales, Social, SES, and Access, were run separately, the groups of two were run, and then finally the group of all 12 variables were used. For each training epoch, the DCS neural network was trained using a random 75% of the data points from the data set described previously. The remaining 25% of the data points were used to test the accuracy of the resultant neural network. For the set of 12 variables, the DCS was allowed to grow to the size of 15 neurons, which leads to only 15 cells and 15 rules. For the sets of eight variables, the DCS was allowed to grow to the size of 15 neurons, which leads to only 15 cells and 15 rules. For the sets of size four variables, the better results were when the DCS was allowed to grow four neurons, four cells, and then four rules.

4.5.3 Training the Neural Network after Rules are Inserted

The DCS was initialized with the centroid sets described above that were constructed from the expert rules. Each of the scales, Social, SES, and Access, were run separately, the groups of two were run, and then finally the group of all 12 variables were used. For each training epoch, the DCS neural network was trained using a random 75% of the data points from the data set described previously. The remaining 25% of the data points were used to test the accuracy of the resultant neural network.

4.5.4 Comparing the Results

For the test described above, five training epochs were completed, and an average taken. The results are summarized in Table 8 below. After the rules were inserted the performance of the neural network (in most cases) improved. Some of the improvement is small, but in some cases, the improvement is between 3% - 7% (highlighted in the table), which represents 100 - 200 additional data points that were correctly categorized. The Access group seemed to be the

most troublesome group. The variables in that group need to be examined and some of them dropped out of the analysis for better prediction results.

Table 8. Comparing Results Without and With Rules Inserted for All Size Groups

Groups	% of data that NN accurately classified		% of data that rules accurately classified		% agreement between NN and rules		% of data that both classified	
	without	with	without	with	without	with	without	with
All	0.652	0.712	0.674	0.716	0.994	1	0.898	0.896
SES & Social	0.672	0.71	0.678	0.718	0.996	1	0.924	0.926
SES & Access	0.636	0.708	0.64	0.71	0.996	1	0.93	0.938
Social & Access	0.786	0.798	0.492	0.564	0.79	0.738	0.078	0.11
SES	0.674	0.694	0.674	0.696	1	1	0.990	0.992
Social	0.672	0.678	0.672	0.678	1	1	0.990	0.990
Access	0.730	0.720	0.564	0.618	0.980	0.984	0.306	0.648

4.6 Conclusions

Several methods of Rule Extraction from the DCS neural network had already been developed [21] and the Rule Insertion process was investigated and simulated on a small benchmark data set in Chapter 3 [47]. Our research focused on applying the method developed in Chapter 3 [47] for inserting rules into a DCS neural network structure. This research represents the first attempt to use a data set supplied by an expert and work with that expert to develop the pre-knowledge to be embedded in the neural network. The Rule Insertion process for the DCS neural network is a promising technique to provide more accurate data analysis. Starting with the pre-knowledge of an expert in the form of inserted rules proved to increase the accuracy of the neural network by up to 7%. The technique also has the benefit of providing the corresponding rules that can be inspected and validated by the expert to see what the neural network has actually learned.

V. FUTURE DIRECTIONS

The artificial neural network is one method of machine learning that is used for many applications. The intention for future research is to continue to focus on uses for ***Rule Insertion*** in various fields. Since, the purpose of Rule Insertion is to start with internal knowledge and train the neural network to see how rules are refined, then experts can better understand and use the knowledge learned by the neural network. To continue to test whether the process is effective, and the rules are useful, my future plan to concentrate on the DCS self-organizing map, but also learn more about Rule Extraction and Insertion techniques for other types of neural network. This could allow me to compare our work with others.

We will continue to work with Cossman on social science data, continuing compare our technique to previous techniques that have been used to analyze the data. Also, we will work with experts in other fields. We have had discussions with Dr. Hatim Al-Jaroushi, a Pulmonologist in the medical school, about our method for analyses of data sets. He has access to medical data sets, and he can serve as the expert in this field to give us information to form the rules for insertion. The hope is that this method will give insight to experts about the data that they cannot get from other techniques.

REFERENCES

[1] J. Bruske and G. Sommer, "Dynamic cell structures," *Advances in neural information processing systems*, pp. 497-504, 1995.

[2] B. Fritzke, "Growing cell structures a self-organizing network for unsupervised and supervised learning," *Neural networks*, vol. 7, no. 9, pp. 1441-1460, 1994.

[3] T. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," In *ICANN'93*, pp. 427-434, 1993.

[4] E. Y. Li, "Artificial Neural Networks and Their Business Applications," *Information and Management*, vol. 27. No. 5, pp. 303-314, 1994.

[5] D. De Ridder, R. P. W. Duin, M. Egmont-Peterson, L. J. Van Vliet, and P. W. Verbeek, "Nonlinear Image Processing using Artificial Neural Networks," *Advances in Imaging and Electron Physics*, Vol. 126, pp. 351-450, 2003.

[6] M. Darrah, B. J. Taylor, and Spiro T. Skias. "Rule Extraction From Dynamic Cell Structure Neural Network Used in a Safety Critical Application," in *Proceedings of FLAIRS Conference*, 2004.

[7] S. M. Kamruzzaman and A. R. Hasan, "Rule Extraction using Artificial Neural Networks," *arXiv: 1009.4984*, 2010.

[8] R. Setiono and H. Liu, "Understanding neural networks via rule extraction," in *IJCAI*, vol. 1, pp. 480-485, 1995.

[9] G. Bologna, "A study on rule extraction from neural networks applied to medical databases," in *The 4th European Conference on Principles and Practice of Knowledge Discovery (PKDD2000)*, Lyon, France, 2000.

[10] M. G. Augasta and T. Kathirvalavakumar, "Rule extraction from neural networks? a comparative study," in *Pattern Recognition, Informatics and Medical Engineering (PRIME)*, 2012 International Conference on, pp. 404–408, IEEE, 2012.

[11] M. A. Darrah and B. J. Taylor, "Rule extraction to understand changes in an adaptive system," *Adaptive Control Approach for Software Quality Improvement*, vol. 20, p. 115, 2011.

[12] Z. Kurd, T. Kelly, and J. Austin, "Safety criteria and safety lifecycle for artificial neural networks," In *Proceedings of Eunite*, vol. 2003, 2003.

[13] G. G Towell and J. W. Shavlik, “Extracting refined rules from knowledge-based neural networks,” in *Machine learning*, vol. 13, no. 1, pp. 71-101, 1993.

[14] C. L. Giles and C. W. Omlin, “Extraction, insertion and refinement of symbolic rules in dynamically driven recurrent neural networks,” *Connection Science*, vol. 5, no. 3-4, pp. 307-337, 1993.

[15] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, “Graph kernels,” *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1201–1242, 2010.

[16] K. Sudheer, “Knowledge extraction from trained neural network river flow models,” *Journal of Hydrologic Engineering*, vol. 10, no. 4, pp. 264–269, 2005.

[17] P. Cortez and A. Morais, A Data Mining Approach to Predict Forest Fires using Meteorological Data, in J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, in *Proceedings of the 13th EPIA 2007*, Portuguese Conference on Artificial Intelligence, (2007), December, Guimaraes, Portugal, 512 - 523, ISBN-13 978-989-95618-0-9.2007

[18] S. Clar, B. Drossel, and F. Schwabl, “Forest fires and other examples of self-organized criticality,” *Journal of Physics: Condensed Matter*, vol. 8, no. 37, p. 6803, 1996.

[19] A. Grishin and A. Fil’Kov, “A model of prediction of forest-fire hazard,” *Journal of engineering physics and thermo- physics*, vol. 76, no. 5, pp. 1139–1144, 2003.

[20] S. Eskandari. "A new approach for forest fire risk modeling using fuzzy AHP and GIS in Hyrcanian forests of Iran." *Arabian Journal of Geosciences,* vol. 10, no. 8 pp. 190, 2017.

[21] S. Annas, K. Suwardi, T. Kanai, and S. Koyama. “Principal Component Analysis and Self-Organizing Map for Visualizing and Classifying Fire Risk in Forest Regions,” *Agricultural and Information Research*, vol. 16, no. 2, pp. 44-51, 2007.

[22] P. Cortez, and AJR Morais, "A data mining approach to predict forest fires using meteorological data," in J. M. Neves, M. F. Santos, J. M. Machado, eds. - *New trends in artificial intelligence : proceedings of the 13th Portuguese Conference on Artificial Intelligence* (EPIA 2007), Guimarães, Portugal, 2007, Lisboa: APPIA, p. 512-523, 2007. ISBN 978-989-95618-0-9.

[23] S. Youssef, and A. Bouroumi. "Prediction of forest fires using artificial neural networks." *Applied Mathematical Sciences* vol. 7, no. 6, pp. 271-286, 2013.

[24] M. G. Augasta and T. Kathirvalavakumar, “Rule extraction from neural networks? a comparative study,” in *Pattern Recognition, Informatics and Medical Engineering (PRIME)*, 2012 International Conference on, pp. 404–408, IEEE, 2012.

[25] MW. Craven, and JW. Shavlik. "Using sampling and queries to extract rules from trained neural networks." In *Machine Learning Proceedings* 1994. Morgan Kaufmann, 1994. 37-45.

[26] AB. Tickle M. Orlowski, and J. Diederich. "DEDEC: decision detection by rule extraction from neural networks," in *QUT NRC*, 1994.

[27] A. S. Garcez, D. M. Gabbay, K. B. Broda, *Neural-Symbolic Learning Systems: Foundations and Applications*, Springer-Verleg, 2002.

[28] M. Hoehfeld and S. E. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 602-611, 1992.

[29] G. G Towell and J. W. Shavlik, "Using symbolic learning to improve knowledge-based neural networks," In *AAAI*, pp. 177-182, 1992.

[30] M. Charles and C. Jorgensen, *Direct adaptive aircraft control using dynamic cell structure neural networks.* NASA Technical Memorandum, Ames Research Center, 1997.

[31] M. Darrah, A. Rubenstein, E. Sorton, and B. DeRoos, "On-board health-state awareness to detect degradation in multirotor systems," In *Proceedings of International Conference on Unmanned Aircraft Systems (ICUAS),* pp. 1134-1141, 2018.

[32] L. L. Pullum, B. J. Taylor, and M. Darrah, *Guidance for the Verification and Validation of Neural Networks*, vol. 11. John Wiley & Sons, 2007.

[33] B. J. Taylor and M. A. Darrah, "Rule extraction as a formal method for the verification and validation of neural networks." In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, vol. 5, pp. 2915-2920, 2005.

[34] Guido Bologna and Yoichi Hayashi, "A Comparison Study on Rule Extraction from Neural Network Ensembles, Boosted Shallow Trees, and SVMs," Applied Computational Intelligence and Soft Computing, vol. 2018, Article ID 4084850, 20 pages, 2018. https://doi.org/10.1155/2018/4084850.

[35] Yan-Xin Liu, Faiyaz Doctor, Shou-Zen Fan, and Jiann-Shing Shieh, "Performance Analysis of Extracted Rule-Base Multivariable Type-2 Self-Organizing Fuzzy Logic Controller Applied to Anesthesia," BioMed Research International, vol. 2014, Article ID 379090, 19 pages, 2014. https://doi.org/10.1155/2014/379090.

[36] M. Darrah, B. J. Taylor, M. Webb, and R. Livingston. "A geometric rule extraction approach used for verification and validation of a safety critical application," in *Proceedings of FLAIRS Conference*, 2005.

[37] M. Hoehfeld and S. E. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," IEEE Transactions on Neural Networks, vol. 3, no. 4, pp. 602-611, 1992.

[38] K. McGarry, S. Wermter, and J. MacIntyre, "Hybrid neural systems: from simple coupling to fully integrated neural networks," Neural Computing Surveys, vol. 2, no. 1, pp. 62-93, 1999. ISSN 1093-7609

[39] L. Chong, M. M. Abbas, A. M. Flintsch, and B. Higgs, "A rule-based neural network approach to model driver naturalistic behavior in traffic," *Transportation Research: Part C Emerging Technologies*, vol. 32, pp. 207.223, 2013.

[40] W. James and J. S. Cossman, "Long-Term Trends in Black and White Mortality in the Rural United States: Evidence of Race-Specific Rural Mortality Penalty," *The Journal of Rural Health*, vol. 33, pp. 21-31, 2017.

[41] J. S. Cossman, W. L. James and R. E. Cossman, "Underlying Causes of the Emergin Neonmetropolitan Mortality Penalty," *American Journal of Public Health*, vol. 100, no. 8, pp. 1417-1419, 2010.

[42] National Center for Health Statistics. Compressed Mortality File: 1968-1988, as compiled from data provided by the 57 vital statistics jurisdictions through the Vital Statistics Cooperative Program. Atlanta, GA: National Center for Health Statistics, Centers for Disease Control and Prevention.

[43] National Center for Health Statistics. Compressed Mortality File: 1989-1998, as compiled from data provided by the 57 vital statistics jurisdictions through the Vital Statistics Cooperative Program. Atlanta, GA: National Center for Health Statistics, Centers for Disease Control and Prevention.

[44] National Center for Health Statistics. Compressed Mortality File: 1999-2012, as compiled from data provided by the 57 vital statistics jurisdictions through the Vital Statistics Cooperative Program. Atlanta, GA: National Center for Health Statistics, Centers for Disease Control and Prevention.

[45] Swanson D, Siegel J. The Methods and Materials of Demography, 2nd ed. Amsterdam: Elsevier Science and Technology Books; 2004.

[46] US Dept. of Health and Human Services. Area Health Resources Files (AHRF). Rockville, MD: US Department of Health and Human Services, Health Resources and Services Administration, Bureau of Health Workforce; 2014-2015.

[47] O. Elsarrar, M. Darrah, and R. Devine, "Rule Insertion Technique for a Dynamic Cell Structure Neural Network to Improve Performance," to be submitted.