Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks

Kenneth McGarry, Stefan Wermter and John MacIntyre
School of Computing, Engineering and Technology
University of Sunderland
St Peter's Campus, St Peter's Way
Sunderland, SR6 ODD, UK

Abstract

This paper describes techniques for integrating neural networks and symbolic components into powerful hybrid systems. 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, a stand-alone neural network requires an interpretation either by a human or a rule-based system. This motivates the integration of neural/symbolic techniques within a hybrid system. A number of integration possibilities exist: some systems consist of neural network components performing symbolic tasks while other systems are composed of several neural networks and symbolic components, each component acting as a self-contained module communicating with the others. Other hybrid systems are able to transform subsymbolic representations into symbolic ones and vice-versa. This paper provides an overview and evaluation of the state of the art of several hybrid neural systems for rule-based processing.

1 Introduction

In recent years there has been an explosive growth in the successful use of hybrid intelligent systems in many diverse areas such as robotics [34], medical diagnosis [45], speech/natural language understanding [97], fault diagnosis of industrial equipment [2], monitoring/control of manufacturing processes [9] and financial applications [50]. The main contributing factor for the development of hybrid systems has been the increased use of neural networks for pattern recognition, classification and optimization tasks [41, 42, 58]. The ability of neural networks to perform tasks that would otherwise prove difficult or intractable to symbolic computing systems is now recognized and they are often used as modules within intelligent hybrid systems. The activity of module integration has initiated the field of hybrid system development [51] which is one of the most promising research areas for building intelligent systems.

Generally the reasons given for coupling neural and symbolic components involve such issues as reducing the brittleness [43] of the rule-based component by incorporating the robustness of a neural network. This is the well known "mesa effect" [13] which illustrates how most expert systems experience a sharp drop off in operational capability when confronted with novel situations for which no specific rules exist. Other reasons include the lack of rule-based adaptability to changing external conditions which can only be addressed through manual modification of the knowledge base. However, by fusing the two techniques together in tighter configurations a number of interesting opportunities arise for increasing the power of neural networks.

[0]This work is supported by European funding under the auspices of Brite-Euram III, project number BE-1313. Comments should be sent to Ken McGarry at cs0kmc@cis.sunderland.ac.uk.

These new opportunities come at the cost of increased operational complexity and computational overheads but empower neural systems with symbolic processing abilities and vice versa.

Previous work reviewing the field of hybrid systems has been somewhat unconstrained since most of the review effort has attempted to encompass very different hybrid intelligent techniques [30, 40] e.g. neural networks, rule-based systems, genetic algorithms and neuro-fuzzy logic. In order to give detailed technical descriptions of hybrid systems we focus upon those hybrid systems using the two most commonly used elements, namely rule-based components and neural networks. The next section will review the main characteristics of neurosymbolic systems, their general advantages/disadvantages and we will specify the most powerful features which should be retained in a hybrid system. Section two will present the criteria used to classify hybrid systems. Such issues as functionality, coupling and complexity will be discussed. Sections three to five describe the main features of the unified, transformational and modular hybrid systems respectively and several applications are given in each category. It is not the intention to give an exhaustive description of all possible systems of hybrid integration but to highlight key developments in hybrid neurosymbolic integration.

1.1 General considerations

The theoretical basis of hybrid system operation is based upon certain common operational similarities between neural networks and rule-based systems. Table 1 shows these features.

Table 1: Correspondence between neural networks and knowledge-based systems

Activity	**Neural Network**	**Symbolic Rule-Based System**
Knowledge format	Connections, network architecture	Rules
Computation elements	Nodes	Premises, conclusions
	Weights	Rule strength
	Thresholds	Predicates
Processing	Continuous activations	Discrete symbols

Table 1 shows that neural networks and rule-based systems have different methods to perform information processing. There appears to be some similarity of function between the goals of the computation process and this may suggest that the two techniques are not so different. However, the distributed nature of neural networks means that the internal nodes and associated weights have no individual conceptual meaning and therefore cannot directly correspond to a rule or its antecedents.

However, both neural network and symbolic processing share the general working hypothesis that cognition may be modeled by computation [82]. While neural networks are better suited to some tasks than symbolic systems and vice versa it should be made clear that either computing technique can perform any task the other is capable of. However, in this case the architecture and representation may not be very efficient. The issues that need to be considered when developing hybrid systems are related to the computational efficiency, accuracy, problem formulation and knowledge representation suitability of the particular computing technique for the task at hand. We have characterized neural networks as a subsymbolic computing technique and rule-based systems as a symbolic computing technique. However, what does it mean by saying that a system can reason subsymbolically or symbolically?

1.2 Subsymbolic representation characteristics

In this section we discuss the characteristics of distributed neural network processing. We have stated that neural networks operate at a subsymbolic level. The basic units of information are continuous values computed by the individual neurons. The neurons output these values in response to the information presented at their inputs. The inputs are numerical values and originate from either other neurons or from the outside world, e.g. sensor readings or today's share prices. Several neural network configurations are possible, such as feedforward, self-organizing and recurrent networks but they have common features. The most important is the use of weighted connections for information storage. The information learned by a neural network is held in the weights and biases as a result of exposure to the training set [31, 27]. The information stored inside the weights consists of real-valued numbers and therefore neural network computing can be classified as a subsymbolic type of processing. The comprehensibility of this knowledge is further compounded since the training set data is distributed over the internal weights. Neural networks have the following features and advantages:

- Compact knowledge representation in the form of weight and threshold value matrices.
- Very fast and simple operation due to the above feature.
- They can operate with noisy or missing data and can generalize quite well to similar unseen data.
- They learn inductively from training data and their ability to process non-linear functionality is an important feature for managing real-world data.

The strength of the neural network paradigm results from the distributed nature of the knowledge, i.e. the learned patterns are stored across all weights and thresholds. This property is known as superposition and enables the efficient storage and recall of individual patterns. By the same token any individual weight or thresholds with a suboptimal value is unlikely to contribute to an incorrect classification. Also noisy input patterns that do not deviate too much from the training patterns will result in good classifications. However, neural networks also have their limitations:

- They may need lengthy training times and may not necessarily converge to an acceptable solution.
- The use of random weight initializations may lead to different solutions.
- Neural network topology design is by empirical means, several attempts to develop an acceptable model may be necessary.
- They have limited explanation facility which may prevent their use in certain applications.
- It is difficult to incrementally add new knowledge and may be difficult to share knowledge with other networks.

In general, the architecture of a neural network, i.e. the number of layers and the number of units in each layer is not a decomposable object. The network is of a distributed nature so that no particular neuron actually encodes a specific concept or feature. This is the main reason for the restricted ability to transfer learning tasks between networks. The inability to isolate the encoded input features means that the network must be considered as a monolithic block. For example, if we had a neural network trained to recognize every aircraft type built by European manufacturers we could not directly extract a subnetwork to recognize the subtask of identifying only French or Italian aircraft. Furthermore, it is only possible to add features if additional training is provided. Adding extra training examples may cause problems unless re-training is carried out using the original training set along with the new examples, otherwise "catastrophic interference" may occur [78].

1.3 Symbolic representation characteristics

A symbol is a discrete token that represents some concept, feature or entity that within the context of the computer program can be manipulated and transformed along with other symbols. The transformations performed upon the symbols enables new relationships to be expressed that ultimately may lead to the solution of a problem. This is the basis of the physical symbol systems hypothesis [65] which states the requirements for general intelligent action. The meaning attached to these symbols occurs as a result of the operations and the human interpretation process. The designer of a system is able to attach labels to the symbols and thus most of the interpretation is carried out by a human being. The external environment of signals and other computing systems may also provide a degree of interpretation in which case symbol grounding will play a role [38, 72, 73].

In addition to processing carried out at the level of symbols there exists the possibility for suitably complex systems to reason at a higher level called the knowledge level [66]. The knowledge level is a rather abstract notion that occurs as a result of the interaction between symbol structures and their processes; it is independent of the symbol level which is used to implement it. The problem solving ability of many symbolic systems is based on state space search [65] in which a state represents the progress towards finding a solution at a single instant in time. The state space often has a high dimensionality; i.e. the input variables cause a combinatorial explosion. Heuristic search can be used to address this problem, where a heuristic refers to an informal procedure that provides a short cut to exhaustively testing all possible solutions [70]. Often a heuristic search procedure cannot guarantee to discover the optimum solution but it will provide reasonably good several candidate solutions.

There are a number of symbolic representation methods such as frames, semantic nets, object-value-attribute triples and rules. Rules are probably the most common form of knowledge representation and they are present in most AI applications such as expert systems and decision support systems. The rules in most symbolic systems are produced by human experts as a result of a lengthy knowledge engineering process [57] or based on machine learning programs [74]. Rules are useful for the following reasons: Rules can be interpreted easily and can be subject to mathematical rigor, i.e. formal logic. The use of rules enables modular systems to be built and a lot of human reasoning can be expressed as rules.

Rules also have some disadvantages: Rules are brittle when presented with noisy data that contains unexpected values or incomplete data with missing values. Also data with nonlinear relationships may be difficult to express by rules. Rules do not scale up very well and inconsistencies and errors may occur. Furthermore, manual encoding of rules is time consuming and expensive [57].

The main cause of brittleness within rule-based systems is the requirement that every possible combination of antecedents and their values must be explicitly provided. Otherwise the system will fail when presented with novel combinations of input data. For example consider the following case:

$$\text{IF } (\textit{temperature} > 60)\ AND\ (\textit{temperature} = \textit{rising})\ OR\ (\textit{temperature} > 70)\ THEN\ \textit{alarm} = \textit{on}$$

For cases where the temperature is less than 60 but rising or when the temperature is 65 but is dropping further rules are required to account for this. Thus in order to cope with the complexities of real-world data a practical symbolic system must have a large number of rules to account for the most common cases that should occur. Except for smaller, restricted applications it is most unlikely that any given rule-based system will be robust or complete enough (i.e. have sufficient rules) to provide reasonably correct answers when presented with deviant input.

The key features that would be desirable are: the ability to reason with noisy and incomplete data and to learn incrementally from new experience. The ability to generalize and to explain the chain of reasoning motivates the exploration of hybrid techniques. Having described the characteristics of subsymbolic/symbolic computing and how these features arise we are now in a position to consider how they may be coupled together to form hybrid systems.

2 CLASSIFICATION OF HYBRID SYSTEMS

There are several possible ways of describing the features of hybrid systems based upon functionality and the degree of inter-connectivity. An entire continuum of integration techniques is possible and not all will neatly fit into the predefined categories. This is part of the richness of hybrid system development. Several schemes for classifying hybrid systems currently exist and most schemes concentrate upon a narrow definition and cannot fully describe all characteristics a system may possess. First, the motivation for providing a new scheme based upon the latest developments in hybrid technology shall be given.

2.1 Prior approaches towards classification of hybrid systems

There have been some attempts for describing hybrid architectures in the past. Probably the earliest scheme proposed by Medsker [61] is concerned with the degree of coupling between neural network and expert system components. Medsker has defined the classes of loosely, tightly and fully integrated systems to describe the degree of coupling between the modules. In loosely coupled systems the communication is performed by shared files while tightly and fully integrated models use shared memory structures. In a strict sense the two remaining classes, i.e. standalone and transformational methods cannot really be classified as hybrid systems. In the standalone model the neural network and symbolic elements are separate and there is no interaction between the components. The transformational model merely checks for the most efficient implementation method through duplicating the development work by building both a rule-based system and a neural network. Medsker's classification scheme only covers the degree of coupling and makes no attempt to describe the module hierarchy or configuration of a hybrid system.

The classification scheme proposed by Hilario [40] has two hybrid classifications for integration, namely the unified and hybrid approaches. The hybrid approach integrates separate symbolic and neural elements using four distinct classes based upon the flow of data between the modules and has two degrees of coupling (loosely and tightly coupled). The term "hybrid" as used by Hilario confers little descriptive advantage since all the systems she describes are hybrids. Those systems classified under the unified approach use neural networks to implement all the processing activities including those normally reserved for symbolic ones. The theoretical basis for a system implemented entirely by neural network elements is provided by the evidence from neurobiology and cognitive science. The main issue is that the symbolic processing capabilities of the human mind originate with the low-level, fine grained processing carried out by biological neurons. A number of such systems have been developed [75, 76, 84]. The term *"unified"* is fairly descriptive and has been retained within our classification scheme.

Goonatilake and Khebbal [30] have defined their scheme on the basis of functionality, architecture and communication requirements. This scheme is more general than Medsker's scheme and can therefore be applied to systems with components other than neural and symbolic elements. The authors have based their scheme upon the motivations most users have for building hybrid systems and as a result have decided upon a three class scheme. The authors have used the terms function-replacing, intercommunicating and polymorphic to describe the major groupings of hybrid systems. The polymorphic class is similar to the unified class of Hilario [40]. However, the function-replacing class rather belongs as a sub-class within the intercommunicating class. The term function-replacing is quite descriptive of the purpose and actions carried out by the hybrid system, e.g. using a neural network to replace the inferencing engine of an expert system but intercommunicating is a redundant term since all hybrid systems no matter what their purpose must communicate. A review of hybrid methodologies was presented in [87].

2.2 Motivation for a new classification scheme

Several hybrid classification schemes exist, with different terminology and objectives but with some degree of overlap of description with the other schemes. Therefore, what can be gained from having a different view? First, we believe that our scheme ties together several threads from the other classification schemes into a consistent approach and that recent developments in hybrid technology require a new perspective not covered by existing schemes. Second, by having a common approach to classifying hybrid systems the developers of such systems will be in a better position to describe the operation and aims of their systems while others will be able to evaluate such systems with a clearer understanding of the computing and information processing issues. Hybrid system development is in a continual state of advancement and our scheme may represent another stage in the development of hybrid systems.

Analysis of the current state of the art in hybrid systems has led us to believe that a comprehensive classification scheme can be composed of three major groups. This grouping occurs naturally because of the configuration of the internal modules and the conceptual understanding of the processing required that occurs within any given hybrid system, e.g. those systems classified as unified require a change in the conceptual understanding of symbolic processing since these activities are performed by a neural architecture. The terminology we have used is based on the most common descriptions but we have extended terms wherever necessary. While there have been related hybrid classification schemes, e.g. for hybrid language systems [98], here we focus on hybrid rule-based systems.

The first group, *"unified hybrid systems"*, consists of those systems that have all processing activities implemented by neural network elements (see figure 1). Such systems may be classified under the unified approach. So far these systems have had only limited impact upon real world applications, due to the complexity of implementation, issues of model scalability and rather limited knowledge representation capability. However, given the relative novelty of the field we can expect further technical improvements and innovations. The second group of systems can transfer a symbolic representation into a neural one and vice versa, see Figure 1. It is with the second category, *"transformational hybrid systems"*, that hybrid systems begin to demonstrate some unique properties. The most interesting feature is the ability to insert, extract and refine symbolic knowledge within the framework of a neural network system. The third category of *"modular hybrid systems"* covers those hybrid systems that are modular in nature, i.e. they are comprised of several neural network and rule-based modules which can have different degrees of coupling and integration. An important aspect is that they do not involve any changes regarding the conceptual operation of either the neural network or rule-based elements. The vast majority of hybrid systems fall into this category. The main reason is that they are powerful processors of information and are relatively easy to implement. For the purposes of consistency and to assist with the description of the various hybrid systems we have produced the diagrams with a uniform appearance, the symbolic rule-based components are block shapes while the neural network elements are oval in shape.

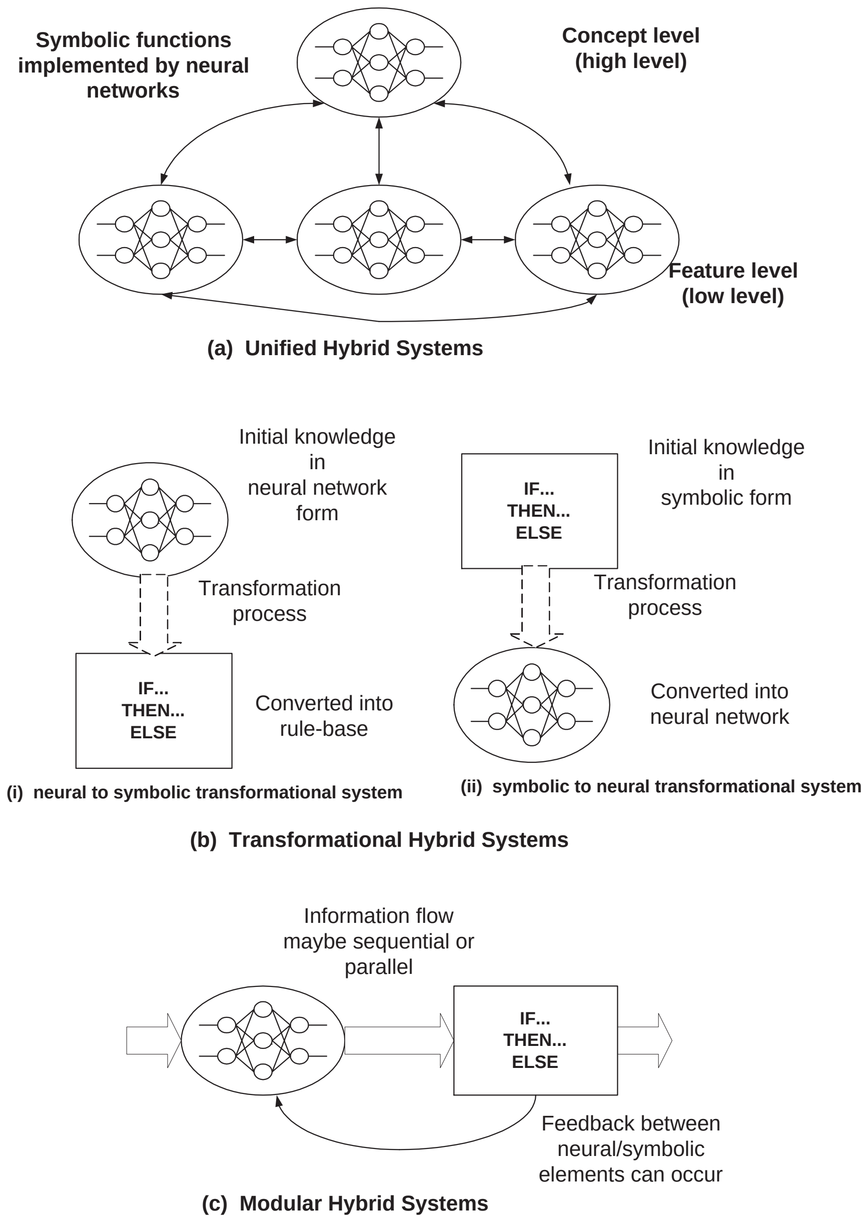
Symbolic functions implemented by neural networks
Concept level (high level)
Feature level (low level)
(a) Unified Hybrid Systems
Initial knowledge in neural network form
Transformation process
IF... THEN... ELSE
Converted into rule-base
(i) neural to symbolic transformational system
Initial knowledge in symbolic form
IF... THEN... ELSE
Transformation process
Converted into neural network
(ii) symbolic to neural transformational system
(b) Transformational Hybrid Systems
Information flow maybe sequential or parallel
IF... THEN... ELSE
Feedback between neural/symbolic elements can occur
(c) Modular Hybrid Systems

Figure 1: Hybrid system classification

3 UNIFIED HYBRID SYSTEMS

The work of researchers such as Feldman [19, 20], Ajjanagadde [4, 5], Smolensky [82] and Shastri [79] has given important insights into why neural networks are suitable for implementing higher cognitive functions usually associated with symbolic processes. Other work in the field has shown the necessity for symbolic structures to carry out intelligent information processing [21, 15, 37]. Dorffner [16] discusses the implications of "radical connectionism" which is a bottom-up approach to modeling intelligence and identifies several properties that neural network structures must possess. The advantages to be gained by combining symbolic structures with the inductive power of neural networks has been more atomized by the construction of hybrid systems that use specialized neural network elements for the implementation of symbolic functions.

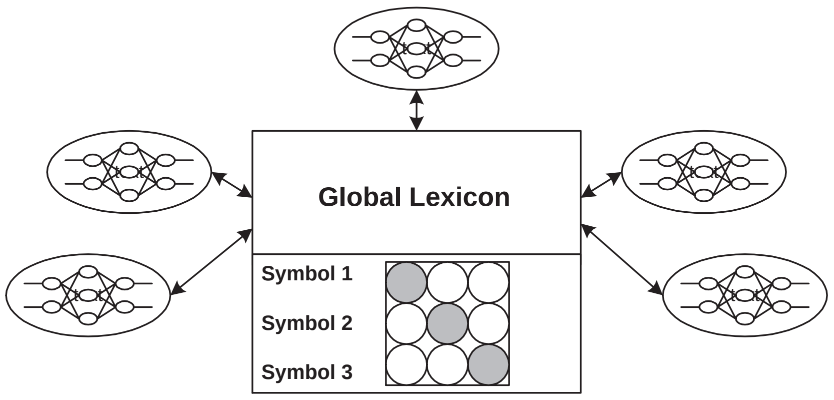

Figure 2: Symbolic recirculation within a unified hybrid system

A key feature of neural networks in general is the distinction made between localist and distributed representations. This difference is used within unified systems to create levels of hierarchy. A local representation employs individual neurons to represent a concept or feature while the distributed representation stores knowledge over a number of units [17]. In addition, a network based upon local representations is effected less by the "catastrophic interference" phenomena [78] and therefore can learn incrementally. However, a distributed representation is more robust in the presence of noise because several neurons contribute to the overall classification accuracy, while in a localist scheme any error may lead to an overall classification failure.

The learning mechanism employed by distributed representations generally requires a longer training time consisting of repeated iterations of the training set. Localist networks require a shorter training time where some networks may only need a single exposure to the training set. Unified hybrid systems often have their symbols encoded within a global lexicon which enables them to be created dynamically during training. A process called symbolic recirculation may occur within a unified network through the activity of learning [63]. See Figure 2 for a hybrid system of five networks that uses distributed symbols for its input and output. Symbol recirculation enables symbols encoded as distributed representations to be updated with new values based upon its relationships with the other symbols in the global lexicon.

3.1 Unified CONSYDERR for evidential robust reasoning

CONSYDERR stands for CONnectionist System with Dual-representation for Evidential Robust Reasoning [85, 86] and is a unified hybrid system with a two-level neural network system that reasons subsymbolically using a distributed representation and symbolically using a localist representation. The architecture was designed to have two different levels in order to overcome the brittleness and uncertainty involved with knowledge based systems. The inspiration is derived from the assumption that human cognitive processes are formed from different levels of processing. Although CONSYDERR is implemented entirely by neural networks, it is able to reason symbolically by a carefully designed hierarchy of levels. The bottom level consists of microfeatures, which are fine grain elements implemented by nodes that have linkages to the higher concept nodes in the top level. Figure 3 shows the hierarchy and the sequence of phase activations.

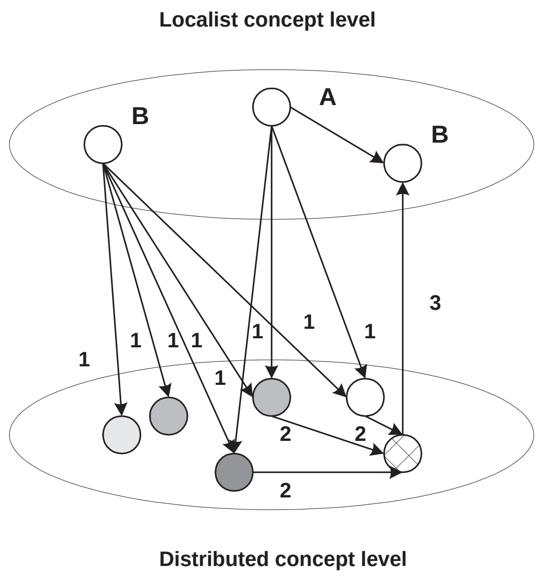

Figure 3: CONSYDERR hybrid system

The system operates in a three phase cycle that begins with a top-down phase that activates the distributed representations in accordance with those distributed nodes connected to nodes at the localist level. Phase two consists of a settling operation that has the activations from the localist level propagating down to the distributed level. The lateral links within the distributed level which represent similarity between the distributed representations now cooperate with the local concepts at the sub-conceptual level. Phase three initiates a bottom-up propagation of activations that eventually results in the inferencing of new concepts. Every node in the system corresponds to a concept or feature and may have several connections to the other nodes according to the relationships between concepts. A high level symbolic interpretation may be assigned to each node depending upon the current pattern of activations. Further work implementing metaphor interpretation has enabled CONSYDERR to learn [85]. While previously CONSYDERR had to have its concepts and microfeatures hand-coded.

3.2 Unified RUBICON for rule-based processing

The RUBICON hybrid system by Samad [76] incorporates both distributed and localist forms of neural network architecture. The integration of both types of architecture enables the representation of complex knowledge structures that are able to manipulate structured information. RUBICON was designed with the goals of reducing the knowledge acquisition bottleneck problem and of reducing the brittleness problem normally associated with expert systems. Compared with the other unified systems described in this section RUBICON operates conceptually more along the lines of a traditional expert system with rules, clauses, attributes and values. Internally, the neural network architecture is more or less invisible to the user.

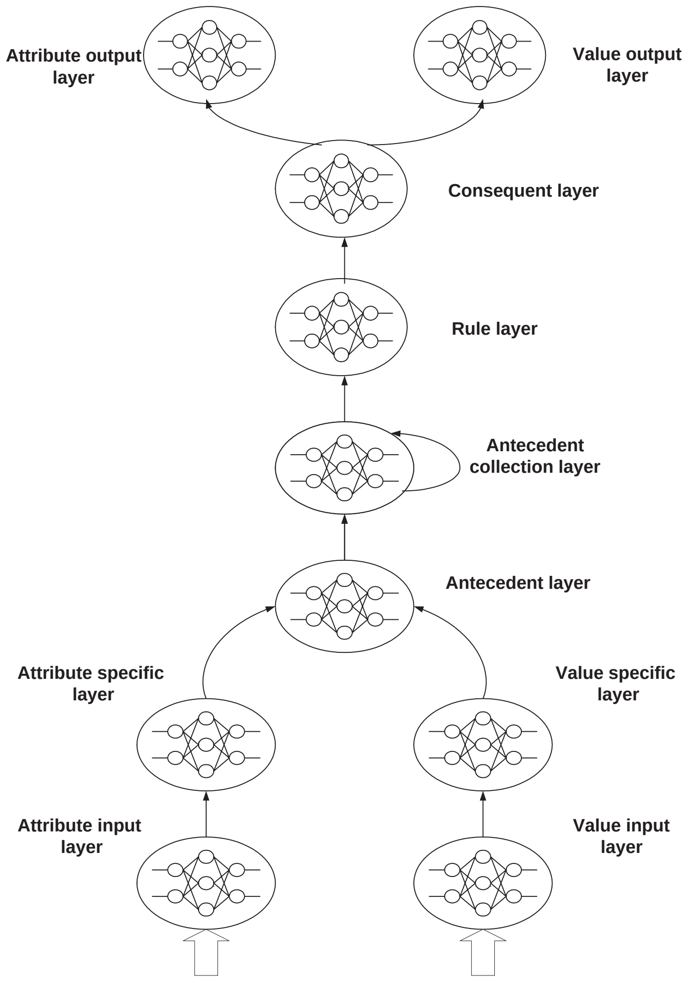

Figure 4: RUBICON architecture

The system consists of a number of layered networks. This is necessary to provide the low level "micro-features" that correspond to the components of the rules, see Figure 4. The input and output layers are composed of distributed architecture units while the other layers consist of localist networks. The localist networks enable the dynamic creation, modification and deletion of the internal knowledge structures. The conversion mechanism from a localist to a distributed representation is achieved quite simply. The distributed output layer is composed of activations from winner-take-all-layers. RUBICON is partitioned into two communicating components: an attribute section and a value section. This separation enables the creation of the specific format of the rules implemented by RUBICON; namely rules consisting of an antecedent clause and a consequent clause. Each clause may have a variable number of attribute-value pairs and the architecture enables the assertion and retraction of clauses. In addition, negated expressions are supported.

Currently, RUBICON does not manage real-valued inputs. This will be a major shortcoming for any industrial, scientific or technical task that deals with continuous real-world data. Unfortunately, variable binding which enables a form of symbolic reasoning similar to expert systems to occur has not been implemented. The author does provide details of how it could be implemented within the RUBICON framework. Further work based upon the internal architecture of RUBICON was implemented as the search facility of a large knowledge-based system integrated within a knowledge acquisition tool for building expert systems. The browser based upon a localist/distributed architecture overcomes certain limitations of this expert system tool.

3.3 Unified SC-NET for dynamic rule-based processing

The SC-NET of Hall and Romaniuk [75, 33] is a hybrid network that operates upon attribute value pairs and has certain similarities to RUBICON. SC-NET is applied to the twin spirals problem and an industrial diagnosis problem. The main form of internal data structure within SC-NET is the cell. A cell is similar to a typical neuron in the sense that it is comprised of an activation function, a threshold function, and associated set of weight values for each connection. The output activation levels are constrained to take three values; 0 for false, 0.5 for unknown and 1 for true. A key feature of SC-NET is a dynamic cell growth algorithm that creates concept nodes when needed. The RCA (recruitment cell growth algorithm) is active during training. Depending upon the severity of error calculated for a given exemplar a new cell will be recruited.

Whenever possible, similar concept cells will represent similar exemplars, otherwise generalization performance will suffer. However, even relatively simple learning problems such as XOR can result in the growth of a large number of nodes. Figure 5.a and Figure 5.b show the network created by exposure to the first exemplar in the XOR problem, demonstrating how cell creation proliferates very quickly. Two types of variables are implemented within SC-NET: a fuzzy variable designed to process continuous information and a scalar variable to process symbolic information. The cell growth is necessary in order to implement the fine grain "microstructure" which is necessary to represent variables. However, it is possible to prune the size of the final network by the global attribute covering algorithm (GAC). This routine analyzes the relationships between the cells and their associated links using various measures of quality to assess their importance prior to optimization. The process of reducing the size of the network also improves the generalization capacity because the simplified network will apply to more cases.

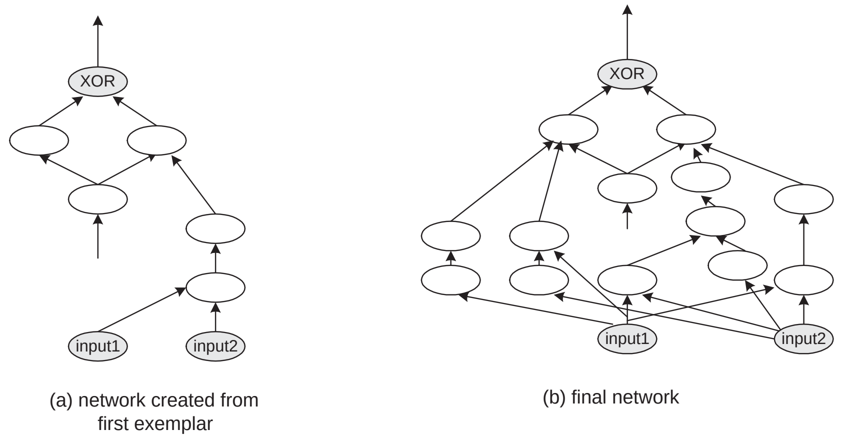

Figure 5: SC-NET implementation of XOR function

3.4 Summary

Unified hybrid systems implement the symbolic processing capabilities of a rule-based system by neural network elements. The motivation for this line of research is that certain forms of rule-based processing can be carried out by a neural network. The use of localist and distributed representations allows the generalization capabilities of neural networks to be supported by the ability to assign conceptual meanings to individual neurons or groups of neurons. Currently, most unified systems do not appear to scale-up effectively to support real-world problems, although research in this direction is continuing.

4 TRANSFORMATIONAL HYBRID SYSTEMS

The transformational models are able to convert an initial symbolic domain into a modified neural network architecture or vice-versa. The transfer process can be a complete compilation of all information from one form into another or it may create intermediate stages. The transformational process creates special opportunities for building hybrid systems that can operate between the two levels of neural/symbolic knowledge representation. Such bi-directional flows of information enable an iterative process of knowledge insertion, extraction, and refinement to be carried out upon symbolic data that resides within a neural network architecture. The symbolic knowledge is often represented as rules but automata [28, 29, 67, 68, 99] and grammars may also be used.

An important capability of transformational hybrid systems is the possibility to build architectures that confer the benefits of both symbolic and neural processing in a single system. Such hybrid systems are essentially neural networks but are generally sparsely connected and the neurons correspond to high level concepts. The networks are able to learn in a process that is similar to supervised networks by means of training examples and often use gradient descent algorithms. In addition, these systems are also able to manipulate the initial architecture and domain knowledge of the neural network, hence they are often described as knowledge-based neural networks [89]. These transformational architectures possess a number of interesting high level features that enable neural networks to perform the following functions:

- The possibility of incremental learning, which means that the neural network need not be retrained with all previous training data in addition to newly, acquired data. New classes may also be included in addition to new training samples for existing classes.

- The inclusion of prior knowledge will have the effect of speeding up the learning process and will be useful in those situations where training examples are scarce. This is called the knowledge insertion, extraction and refinement stage in many systems [89, 62].

- A more deterministic architecture is possible rather than the empirical process that must occur with multi-layer perceptron networks in order to discover a very good architecture, i.e. the number of layers, hidden units etc.

- The reasoning and classification operations are rendered more transparent, although some knowledge-based neural network architectures (KBNN) require a further process of symbolic rule extraction [90].

The experimental work carried out by a number researchers on different knowledge-based neural network architectures has produced some impressive results [54, 23]. They show good performance in terms of classification accuracy, speed of training, reasoning with noisy and missing data and good generalization capability with small training sets. Figure 6 illustrates the knowledge insertion, extraction and refinement phase that is incorporated in many transformational hybrid systems. The use of prior domain knowledge in the form of rules can be used to define the architecture of an initial neural network. The network can be refined by inductive learning when supplied with examples. This may entail changes to the original topology, weights and biases. The learning algorithm may be a gradient descent type such as backpropagation with modifications to account for the sparse number of connections found in KBNNs. The improved performance of the neural network can be used to refine the initial rule base by a process of knowledge extraction from the neural network. The nodes and connections of the KBNN correspond to the symbolic meaning of the initial domain knowledge and are easily converted back into a symbolic format. The entire process can be repeated several times until the system shows an overall improved performance.

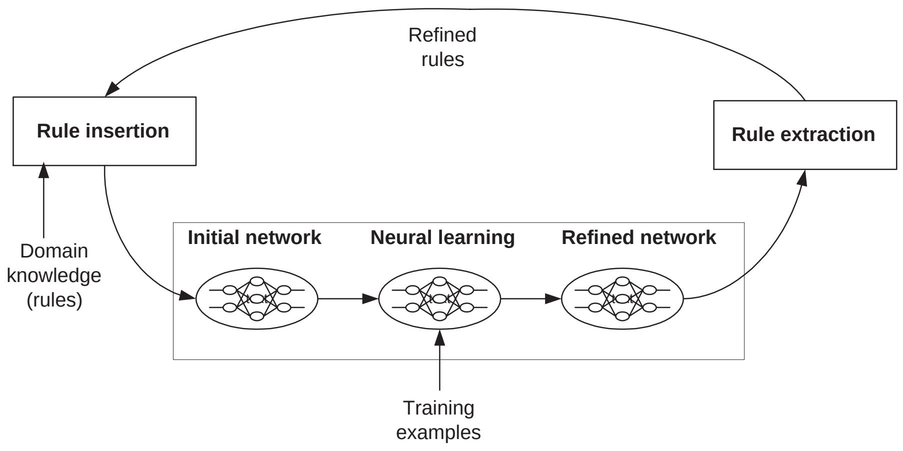

Figure 6: Cyclic rule extraction, insertion and refinement

The combination of using domain knowledge and inductive learning has proved to be a major factor in the success of transformational systems. The next two subsections describe the techniques used to implement the processes that occur within transformational hybrid systems.

4.1 Neural to symbolic transformation

A direct way of converting neural to symbolic knowledge is through rule extraction. This process provides a limited form of an explanation facility of how a neural network may classify any given input pattern. Rule extraction is a process that discovers the hyperplane positions of the input-to-hidden units and the hidden-to-output units of a neural network. These positions are then formulated as IF..THEN rules with the most important input unit labels acting as the rule antecedents. The discovery of the hyperplane positions can be found by a number of techniques that analyze the weights and biases of the neural network. Rule extraction can be carried out with a variety of neural network types such as multi-layer perceptrons [92, 10, 11, 14], Kohonen networks [93], radial basis functions [7, 48] and recurrent networks [67].

In recent years there has been a great deal of interest in exploring techniques for extracting symbolic rules from neural networks [6]. The benefits of extracting rules from neural networks are:

- Provision of an explanation facility by examining extracted rules for various input configurations.
- Deficiencies in the original training set may be identified, thus the generalization of the network may be improved by the addition/enhancement of new classes.
- Analysis of previously unknown relationships in the data. This feature has a huge potential for data discovery/mining and possibilities may exist for scientific induction.
- Once having extracted rules from a neural network we have a rule base that has the potential to be inserted back into a new network with a similar problem domain. This is similar to the heuristics given to expert systems. Also like the heuristics the extracted/inserted rules may be refined, as more

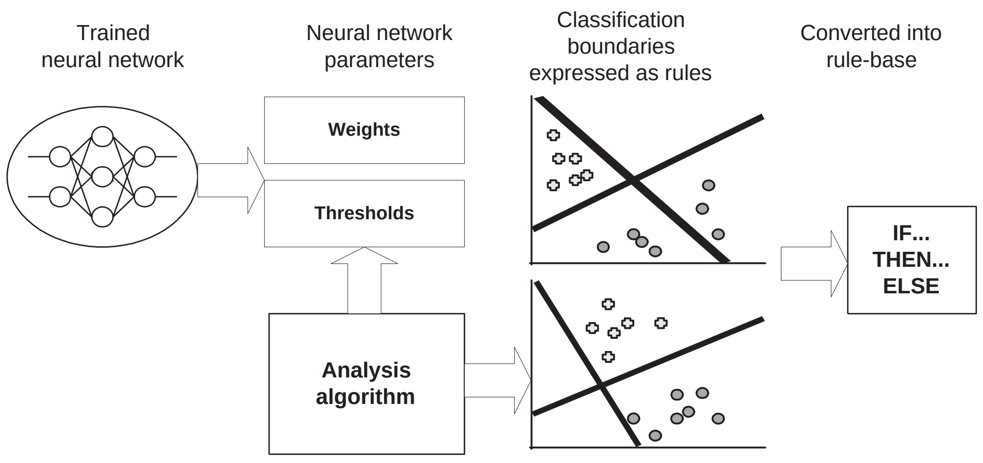

Figure 7: Rule extraction process

information becomes available about the problem. This process may be a step in the right direction towards alleviating the so-called knowledge acquisition bottleneck.

An important part of rule extraction is concerned with the preprocessing of the derived rules and is a process that must be carried out to ensure that a compact set of good quality adequately describes the initial neural network.

- Rule subsumption: when rule extraction algorithms operate they may generate more rules than are actually required to describe the neural network. Subsumption is a way to reduce the number of such superfluous rules by checking for more general rules. A rule can be said to subsume another if it is more general than one or more specific rules. In this case the specific rules can be deleted.

- Rule resolution: if two or more rules have the same number of antecedents and the same conclusion but differ by one antecedent being a negation then they can be resolved into a single rule.

Extracting rules from a trained neural network is not the same as using decision trees or some similar symbolic rule-based induction program directly upon some data set. The reason for this difference is that a well-trained neural network has the capacity to generalize, i.e. to correctly identify similar but previously unseen data. That means, the information extracted from the neural network is an abstraction of the initial training set. In addition, further distortions are introduced by the nonlinear nature of the neural network learning algorithm.

Neural networks that have learned a complex input-output mapping may not be appropriate for transformation into symbolic rules [92]. The extracted rules may not be able to express the input-output mappings in a concise way. This means that probably all rule extraction methods have limits on what they can describe and there will always be a degree of abstraction. While extracted symbolic rules may be less precise they also provide a higher abstraction from the network knowledge. Sometimes extracted rules might also perform better than a network, although this is not necessarily the case.

4.2 Symbolic to subsymbolic transformation

One technique that can convert a symbolic representation into a subsymbolic representation is to create a neural network topology without the requirement to discover the optimum parameters experimentally. It is possible to generate a decision tree [12] from the neural network training data that effectively enables the number of units in a two hidden layer neural network to be pre-determined. The non-leaf nodes correspond to the number of hidden units required in the first layer while the leaf nodes correspond to the number of units in the second hidden layer. The non-leaf nodes perform hyperplane tests while the leaf nodes act as class labels. Figure 8 shows a decision tree generated from data which is then used to specify the architecture of a neural network.

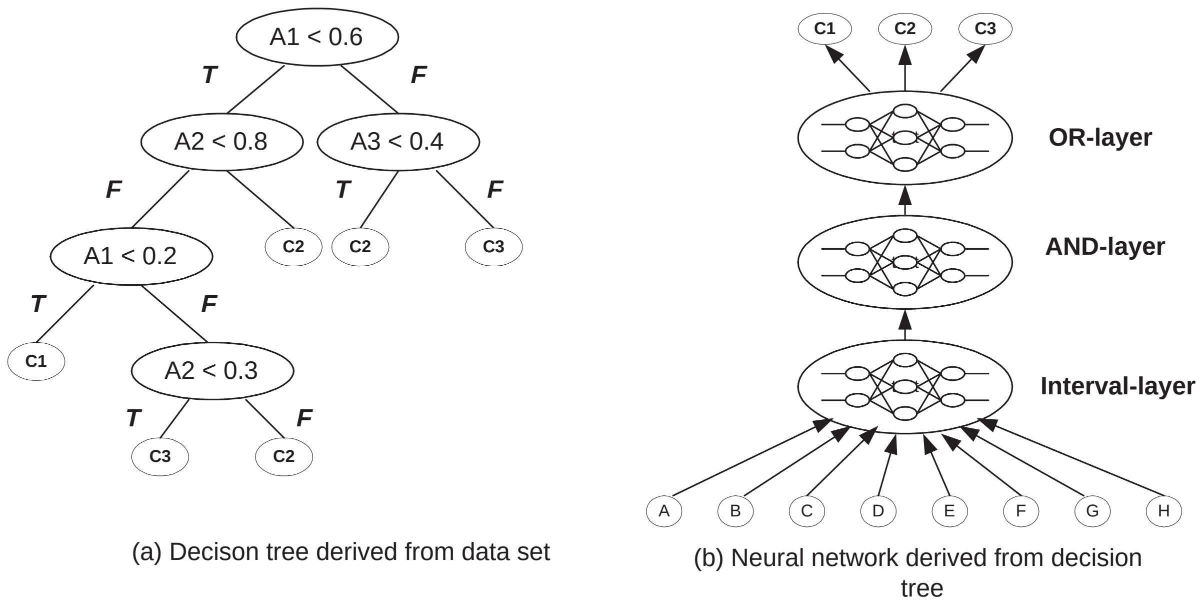

Figure 8: Symbolic to subsymbolic conversion

Having determined the topology, the neural network may be trained by backpropagation or a similar algorithm. The main advantage of this method is the time and effort saved in avoiding the exhaustive testing of different neural network topologies to obtain an acceptable solution. It is also possible to embed neural elements within a decision tree architecture [94]. These Perceptron trees integrate linear threshold units (LTU) within each leaf node of the decision tree. A similar technique exists that uses sigmoid functions in place of the LTUs and is called an entropy net [77]. The tree-based neural net (TBNN) system of Ivanova and Kubat [46] generalizes quite well when compared with other inductive learning systems when classifying unseen data. TBNN appears to operate conceptually along similar lines to KBANN as both systems create partially connected, approximately correct networks. TBNN uses a numeric version of ID3 to generate the tree structure. All real-valued attributes are transformed into intervals and a fuzzification scheme is used improve the accuracy of the interval classes.

4.3 Transformational KBCNN for rule insertion and extraction

The Knowledge Based Conceptual Neural Network (KBCNN) of Fu [24, 25] can revise and learn knowledge by translating the initial domain rules into a network. Figure 9 shows how the initial rules from the domain knowledge were transformed into a neural network. The functions of the knowledge base and the inference engine are combined into an entity called a "conceptualization" where an individual neuron represents a concept and the inter-neuron links represent relationships between the concepts. In addition, KBCNN provides a bi-directional flow of information between the neural network and rule-based elements. A rule can be mapped into a network and vice-versa. Some changes have been made to the learning algorithm so

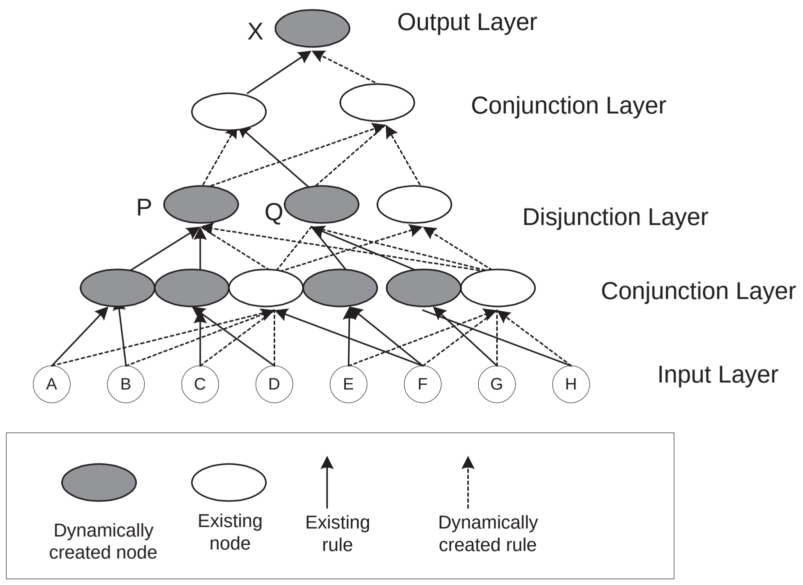

Figure 9: A KBCNN network

that the neural architecture may be modified adaptively. The hidden units are used to represent concepts extracted from the rule base antecedents. The concepts are then clustered and those hidden units covering the same space are removed. This compacting of the network simplifies the search routines by removing redundant units. KBCNN has been tested on a real-world genetic database for recognizing gene promoters in DNA strings. KBCNN was able to revise the domain theory by discovering the gaps in the initial knowledge after examining the training set. KBCNN is an improvement of earlier KBNN systems developed by Fu through the incorporation of revision of incorrect rules and a more efficient knowledge representation system.

4.4 Transformational KBANN for rule insertion and extraction

The KBANN (Knowledge Based Artificial Neural Networks) system developed by Towell and Shavlik [89] seeks to translate a set of approximately correct rules into a neural network architecture. The system is able to make these mappings by taking advantage of a number of conceptual correspondences between knowledge bases and neural networks. KBANN consists of two distinct algorithms: the first converts the rule base into a feedforward network and the second algorithm refines the neural network. The initial rules are in the form of logic-like notation which may be rewritten so that the disjuncts have only one antecedent. This process is required to simplify the rules to network translation and also maintains the rule hierarchy. Having translated the rules into a network structure, a modified version of the backpropagation algorithm is used to refine the networks parameters (weights and topology) and as a result nodes may be added or deleted, see Figure 10. The authors present results of operation that are quite impressive compared with purely symbolic or neural techniques. The authors performed a series of tests that demonstrate KBANN's superiority and effectiveness in the presence of noise against several other learning techniques such as standard backpropagation, Cobweb, EITHER and Labyrinth-K. The KBANN system is able to recover from incorrect initial domain knowledge while at the same time dealing with spurious training examples despite fewer training samples.

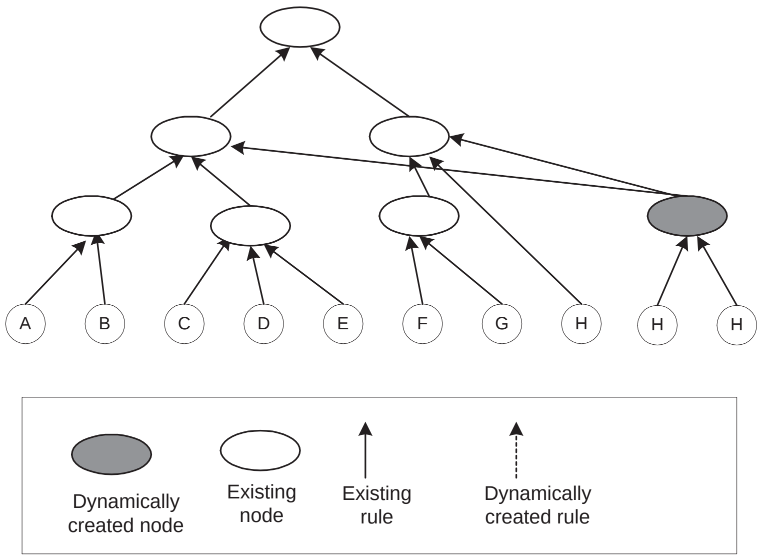

Figure 10: KBANN hybrid system

4.5 Transformational MACIE for dynamic rule creation

One of the earliest knowledge based neural network systems is MACIE developed by Gallant [26] to diagnose medical conditions. Referring to Figure 11 the architecture is essentially a feedforward one with connections existing between layers and within layers. MACIE was developed with the aim of alleviating the knowledge acquisition bottleneck by using the inductive powers of a neural network to generate a model from examples and also to reduce the brittleness of the inferencing engine. MACIE's inferencing engine is implemented by a three-layer neural network and has the ability to reason with partial information. The neurons or cells as Gallant describes them are organized into input, hidden and output layers that encode for symptoms, diseases and treatments respectively.

Additional cells may be added to improve generalization by the learning algorithm during training. The output values of the nodes are constrained to take values of +1, 0 and -1 to encode for concepts of true, false and unknown. In addition to both forward and backward inferencing MACIE also uses the neural network for finding unknown input variables and giving justifications for any inference made. MACIE has a number of limitations, for example it is a requirement of the training algorithm (pocket algorithm) to specify the correct activation of every unit in the network. Although this characteristic is not an disadvantage of the medical domain task where the hidden units encode for specific diseases it will cause problems for potential applications that requires an input-to-output mapping. Despite the serious drawbacks MACIE documents an important first step towards knowledge-based neural networks.

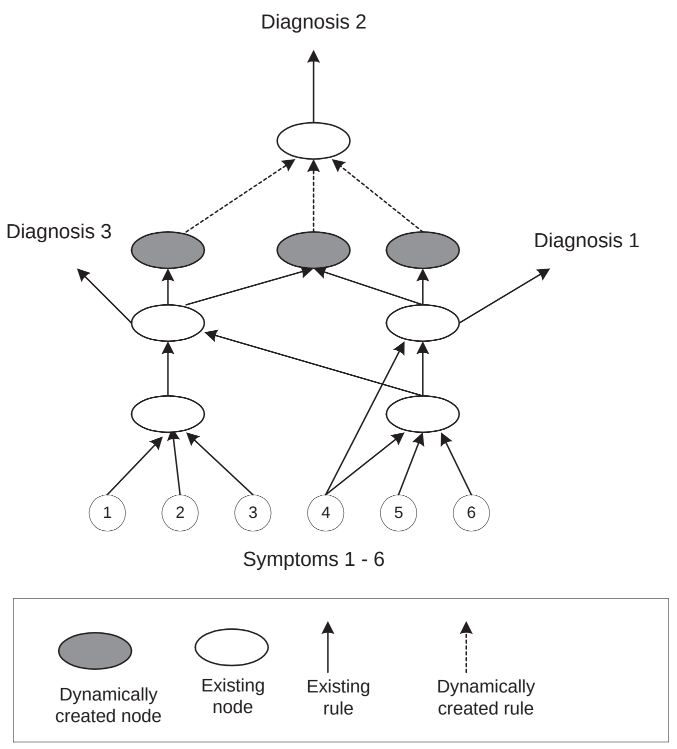

Figure 11: MACIE hybrid system

4.6 Transformational Expert networks for rule refinement

The Expert Network system proposed by Hruska et al [44] is an attempt to refine an initial set of sparse rules gained from the domain expert. This is another system designed to overcome the knowledge acquisition bottleneck. The initial rule base is mapped into nodes representing antecedents, and the interconnections represent the certainty factors. By using a combination of training algorithms the accuracy and robustness of the initial rules is greatly improved. The authors test their system upon the wine advisor database but try to demonstrate its effectiveness by pruning nodes and connections and retraining the network for its ability to reconstruct its knowledge base.

Figure 12 shows the architecture for the 4-2-4 encoder/decoder problem. The nodes are not the typical feedforward multi-layer perceptron (MLP) neurons since this system is event-driven and the nodes possess such features as internal states, processing states and a simple multi-tasking communication facility that is able to detect the states of the other nodes. Referring to the diagram we can see that some nodes are dedicated to the functions AND and NOT. The system always attempts to expand the number of rules to reduce the complexity of any given rule. This simplifies the functionality of a node but obviously the number of rules can grow quickly in a real-world task. The implementation of the expert network creates a much larger matrix (for weights, biases and connection details) than those created in typical MLP type networks, the network in Figure 12 has 16 which creates a matrix of 16 x 16. Further work upon the training algorithms (backpropagation) [54] demonstrates its capabilities with reasoning with partial knowledge and noisy or missing data. A detailed analysis has been carried out showing how expert networks are able to generalize better than neural networks and why they require fewer training examples [23].

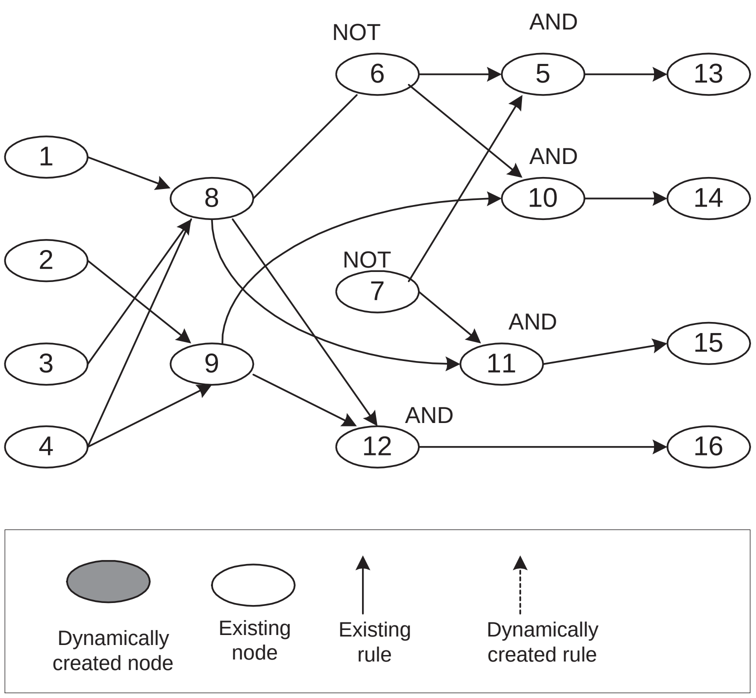

Figure 12: Expert network

4.7 Summary

The transformational models are a successful technique for integrating the characteristics of neural networks and rule-based systems. The main function of a transformational system is to convert one representational technique into another. Rule extraction from neural networks may give some combinatorial problems but this effect can be reduced with rule subsumption and rule resolution. Most of the transformational systems discussed consist of a neural network architecture that is able to assign conceptual meaning to the links and units. This enables prior domain knowledge (if it exists) to be inserted into the network. Prior knowledge of a task may assist in the learning of new tasks by speeding up training times or in those situations where data is scarce. This prior knowledge may then be refined through the use of learning algorithms operating upon additional data. Several transformational systems have been developed some of which are able to cope with the demands of real-world tasks.

5 MODULAR HYBRID SYSTEMS

Several features characterize modular hybrid systems. The most obvious characteristics are the hierarchy of module configuration. The configuration determines the complexity of information flow between the modules. Sequential flow implies that one process must be completed before data may be passed on to the next module. Parallel flow may involve simultaneous operation upon data or even feedback between the modules that can influence the course of future processing. Depending on the particular task the complexity of modular systems can vary greatly. Some systems consist of only a few modules with simple coupling and limited information flow between them. Some modular hybrid systems are more complex and are composed of many modules. As hybrid systems technology has matured, more complicated and sophisticated systems are beginning to be developed. Complexity can be measured in terms of information flow between the modules, which can be uni-directional or bi-directional. Another characteristic is the degree of coupling between the modules. This is again determined by how the modules are configured. Coupling can be identified as loosely coupled, tightly coupled or interleaved.

5.1 Module configurations

Depending on the processing requirements a hybrid system may take a number of architectures, see Figure 13. The neural network and rule-based components are used to implement the various functions required. Some of the newer and larger hybrid systems have many neural and symbolic modules connected in different configurations.

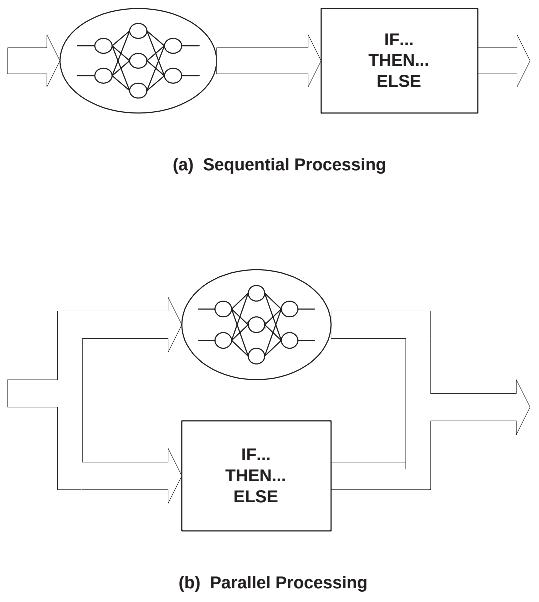

Figure 13: Hybrid system processing configurations

- Sequential configuration: The main feature of this configuration is the serial processing of data as it is passed from one module to the next. One module acts as a preprocessor of data extracting the required features into a form suitable for the next module. A neural network could act as a preprocessor for a rule-based system by converting signal level information into a form more suitable for symbolic level decision making. It is also the case that a rule-based module can preprocess data for a neural network by identifying the relevant parameters for the appropriate input units.

- Parallel configuration: In this configuration a neural network and rule-based system operate in parallel on some common data. The reason for this approach is to compare the classifications obtained for greater confidence and reliability. Another possibility for parallel operation is where the neural network and rule-based elements operate on different data but combine their results for an overall classification. Parallel configurations have the capability to use feedback of information from the output of one module into the input of another enabling a more sophisticated degree of control to be implemented. Time/sequence dependent information may be used to alter the operation of the system.

5.2 Module Coupling

The criterion selected to describe the coupling in our classification scheme is based upon the degree of communication activity between the modules and information flow. The simplest being passive coupling and the most complex and interactive being interleaved integration. Figure 14 shows three types of communication activity possible for a hybrid system.[1]

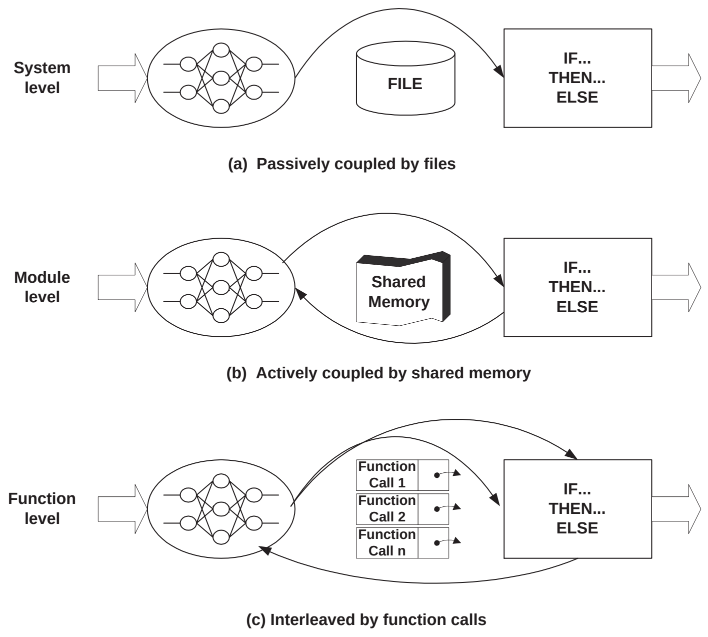

Figure 14: Hybrid system coupling

- Passively coupled: The simplest method of integration is called passively coupled because of the almost autonomous existence of the neural and rule-based components. The method of communication is usually by a data file shared by the components. After the first module finishes its computation it deposits the results in a file to be read by the second module. Passive coupling does not require any sophisticated handshaking control to synchronize the modules since the flow of communications is normally uni-directional and all the data is usually deposited in a single action. A typical configuration would involve a neural network processing an input data stream and saving the output unit activations as vectors/arrays stored in a file.

- Actively coupled: Actively coupled systems are more complex than passively coupled models since communication is by means of shared memory/data structures and so greater effort must be taken to ensure module synchronization. Inter-module communication by shared data structures enables faster runtime performance and allows more sophisticated messages to be passed. Communications can be bi-directional enabling feedback to occur between the modules, which allows a module to alter its operation based on its effect on another module. The use of feedback enables the behavior of a system to be dynamically alterable which in certain hybrid system applications such as speech understanding and industrial process control is necessarily based on changing external conditions.

[1]The passive, active and interleaved categories of coupling roughly relate to Medsker's hybrid classification scheme [61] based upon the loose, tight and fully coupled categories.

- Interleaved: The communication occurring within interleaved systems operates at the fine-grained level of function calls, the modules are highly active with a large degree of module interaction. Again, the use of feedback plays an important part of interleaved strategies. Interleaved systems have a greater capacity for interaction. An interleaved system integrates the strengths of neural networks within rule-based systems by replacing various symbolic modules with neural components. Externally, the individual neural network and symbolic modules cannot be differentiated within such fully integrated architectures. To enable such an integration to occur, interleaved hybrid systems must implement a specialized communication protocol. The protocol must be sophisticated enough to manage a diverse range of inter-module commands and data exchange formats required for sequential and parallel operation.

Several modular hybrid system applications will now be described to illustrate the configuration and coupling possibilities. The systems will be presented with increasing orders of interaction.

5.3 Sequential passive coupling in a vibration spectrum analyzer

The architecture of the vibration spectrum analyzer system represents the simplest of the modular hybrid architectures. It consists of a sequential, feedforward configuration that is coupled by function calls. The data processed by each module must be completed before it can be passed on to the next module. Data feedback or partial computations are not possible with this particular sequential architecture. The block diagram of the system is shown in Figure 15.

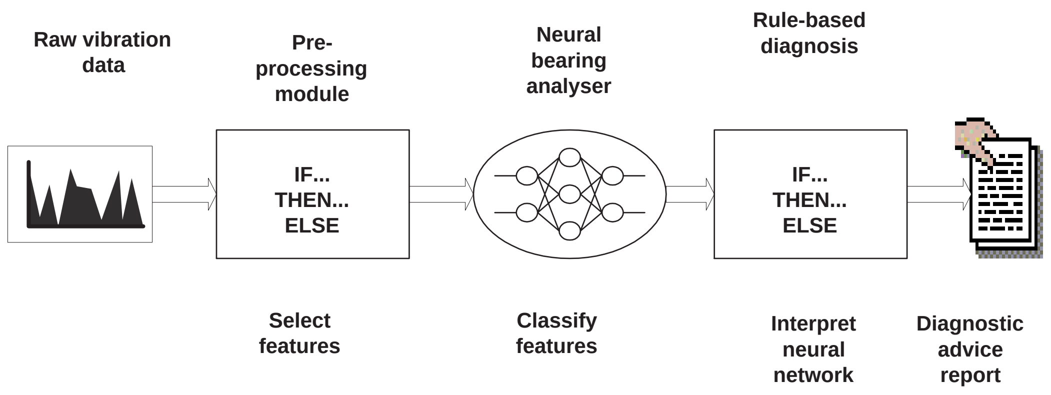

Figure 15: Vibration diagnostic hybrid system

The diagnostic hybrid system [60] is designed to detect and report bearing faults in heavy rotating machinery such as large motors and fans. The diagnostic information is based upon vibration data gathered from sensors connected to the various items of plant under observation. A great deal of preprocessing must be carried out since the raw vibration data has a high dimensionality and only a small fraction can be used to train the neural networks.

The preprocessing module reduces the dimensionality of the raw input spectra by selecting the most important parameters, which are easily calculated using heuristics. The transformed data is then passed onto the neural network module, which is designed to detect a number of bearing faults. A neural network is required for this task since several faults exhibit the same symptoms. The output of the neural network is interpreted by a rule-based diagnosis module which provides details of the faults and is also able to provide trend analysis.

A number of different neural network models were trained and tested on data generated by a specially built test rig and later using real world data. The system performed well using multi-layer perceptrons and later radial basis functions were tried with equal success leading to several large-scale projects [2]. A similar hybrid project using Kohonen networks is the vibration diagnostic system of Kirkham [52]. This is a more complex system and has the potential to use symbolic knowledge from clustering carried out by the Kohonen network. Other similar systems include the medical diagnosis hybrid system of Hudson [45] and the ultrasound inspection system of Kang [49].

5.4 Parallel active coupling in a neural network inferencing engine

The hybrid system developed by Tirri [88] was developed to assist the inferencing engine of an expert system by a set of trained neural networks. The hybrid system requires parallel interaction to occur between its modules and therefore employs active coupling. The system consists of three knowledge bases: a rule base, a fact base which contains the working memory and a neural network of several trained radial basis function networks (RBF) [64, 69, 71]. Each neural network corresponds to a specific symbolic predicate in the system. In fact, neural networks perform the tasks normally carried out by rules in a conventional knowledge-based system.

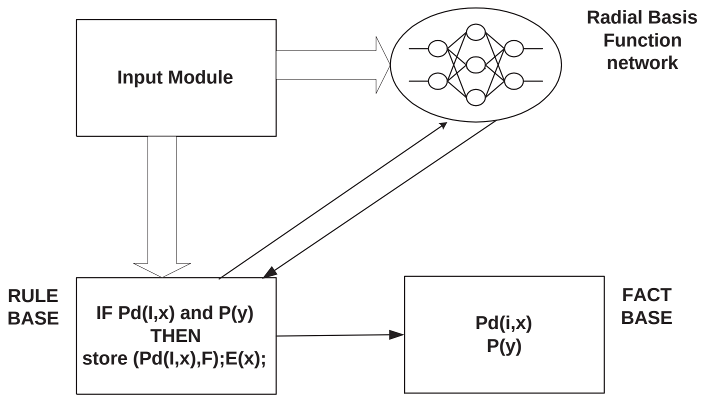
Radial Basis Function network
Input Module
RULE BASE
IF Pd(I,x) and P(y) THEN store (Pd(I,x),F);E(x);
Pd(i,x) P(y)
FACT BASE

Figure 16: Neural network based inferencing engine

During operation the neural network associated with a particular predicate carries out a classification test that is essentially a logical test on its input data. The end result is a conversion process that translates the input data into symbolic facts. Such newly 'discovered' relationships are added to the fact base. The neural network conversion process only operates when the standard symbolic inference engine cannot correctly classify the input data. Figure 16 shows the relationship between the neural and symbolic elements as the execution of a rule with a Boolean predicate occurs. Should the fact base contain information that is proved to be incorrect by the system failing to find a satisfactory conclusion then it is possible for the RBF network to undergo dynamic retraining. The adaptive process is carried out upon the RBF network with the data and the correct classification label, the characteristics of the RBF are well suited for online, fast and incremental learning. Similar systems include the CONKAT medical diagnosis system of Ultsch [93] and the robotic skill acquisition system of Handelman [34].

5.5 Parallel interleaved integration in a CBR/NN data analysis system

The hybrid system developed by Lees et al [56] uses a case-based reasoning (CBR) [1, 55] module interleaved with several neural network modules. The system requires interleaved integration since the neural network modules undergo fine-grained, on-line manipulation of their parameters. The system is intended as a tool for data mining and knowledge discovery within large databases. It has been used successfully in two large real-world domains: Analyzing oceanographic data for the prediction of ocean parameters and a civil engineering design project. Figure 17 shows the interaction between the neural network module and the CBR module. The neural network module provides the CBR adaption phase with the ability to learn. The

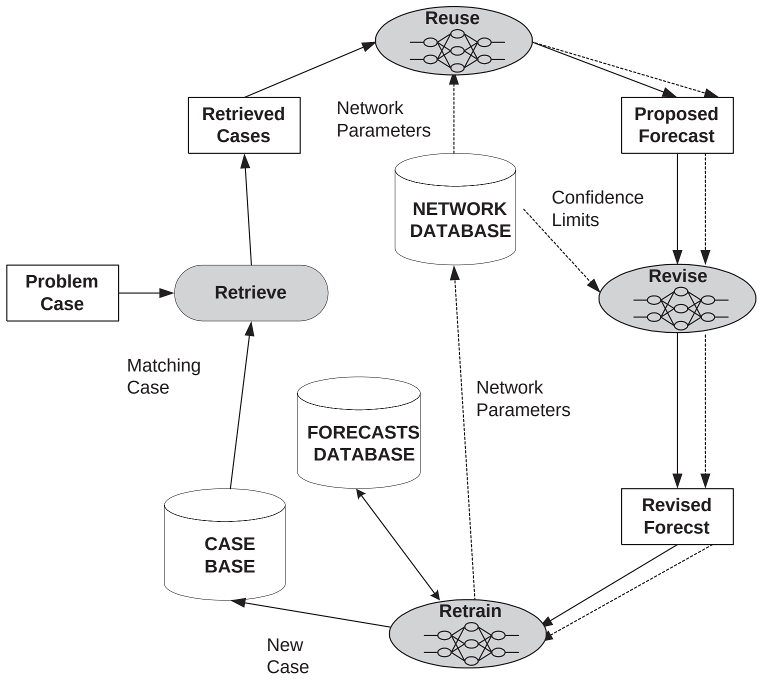

Figure 17: CBR and neural network module interaction

characteristics of the radial basis function (RBF) network are useful for a real-time application. Rapid learning and insensitivity to the order of the training patterns are the main requirements. The dashed lines in Figure 17 illustrate the flow of data between the neural network and the CBR module. The flow of information within this module can run in parallel with the symbolic CBR element.

The CBR cycle consists of four basic operations: The first phase requires that the satellite data stored in the case base is used to generate cases. The problem cases are generated from the current sensor readings. The retrieve phase searches the case base and matches the closest case with the current problem case. The reuse phase sees the RBF networks parameters that were calculated from the most recent forecast taken from the network database and used with several other cases. The RBF network is then able to learn from a number of cases rather than a single best case. During the revise phase, the new forecast is modified according to the accuracy of the previous forecasts and new confidence limits are calculated. The forecast is then described in the form of upper and lower intervals. The forecast is placed in temporary store until actual measurements taken from the ocean are made and compared with the predicted values. Depending upon the accuracy of the forecast it may be discarded or saved as a new case for future use.

An important feature of this hybrid system is the interaction between the RBF network and the rest of the system. The RBF network is able to transfer its learned knowledge for further use to the CBR module.

This initial RBF network will have its knowledge updated by retraining upon new data. The use of prior domain knowledge and inductive learning is a powerful combination for use as an adaptive module. Similar hybrid systems using a CBR module is the hybrid CBR of Yao [101] and the knowledge extraction system of Egri [18].

5.6 Parallel interleaved integration for speech and language analysis

The SCREEN (Symbolic Connectionist for Robust EnterprisE for Natural language) hybrid system was developed to learn the acoustics, syntax, semantics and pragmatics of spoken language [96]. Due to the complexity of this task SCREEN requires an interleaved architecture for learning spoken language analysis [98]. The analysis of speech is more complicated than the understanding of written text due to its inherent irregularity such as pauses, repetitions, the use of poor grammar and unforeseeable semantic constructions.

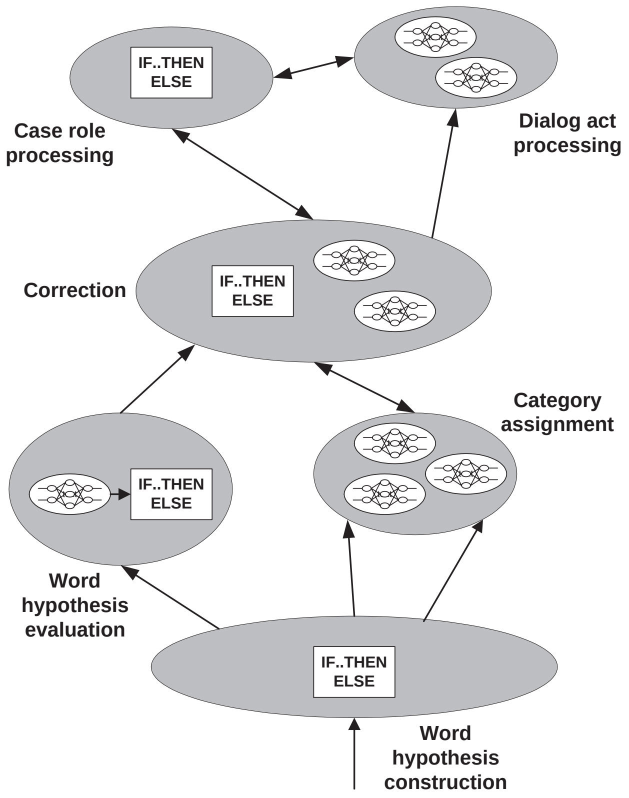

Figure 18: SCREEN spoken language analysis system

The architecture of SCREEN has six functional parts and each one consists of a mixture of neural network and symbolic modules. Some modules operate sequentially while others operate in parallel and require feedback from other cooperating modules as they process information. Flow of information starts initially with the speech construction part. In this module sentences are constructed from word hypotheses generated from a speech-recognizer. The speech evaluation part computes the syntactic and semantic plausibilities required for selecting the best sentence hypotheses. The category part performs a syntactic and semantic analysis at word and phrase level. The correction part deals with errors and repairs them. The dialog act processing uses knowledge at the dialog level to assign an utterance to a particular act, e.g. accept, reject or suggest. The case role processing part provides the ability to retrieve the analyzed utterances for later use.

Figure 18 shows a simplified architecture of the SCREEN system. Control within the SCREEN system is symbolic but most modules are based on neural networks. The interface between the neural network and the symbolic modules is by a common message type structure that can support parallel and sequential processing. Externally, each module communicates using this structure and no difference is made between neural network and symbolic module communication. Through this interleaved communication SCREEN can symbolically represent the information processed by feedforward and recurrent networks. The communication mechanism is similar to that provided by the hybrid chemical analysis system CONNCERT of Wilson and Hendler [100], which uses symbolic supervisors to oversee the processing of neural networks.

SCREEN works incrementally, which allows the system to integrate knowledge sources very early. The use of recurrent neural networks within the category assignment module provided a means of automating the syntactic and semantic knowledge acquisition process. The recurrent networks also provided a method of identifying those relationships and regularities in the speech data not easily captured by symbolic rules. Another innovation is the reduction of noise that is introduced by the human speaker and also through the speech recognizer itself. A speech correction part consisting of several interleaved neural network/symbolic modules was developed to overcome this. The system is able to remove pauses and repetitions of words by symbolic means. The analysis and repair of words/phrases is a more complex task and is implemented by neural networks.

Experimental work showed that the recurrent networks were able to give good generalization accuracy and were quick to train. Also, SCREEN required few training examples and compared quite well with a symbolic based system and with other hybrid systems using neural networks [98]. The main reason for SCREEN's robustness is the use of neural networks for those speech analysis tasks that are noisy and require good generalization capability.

Similar systems dealing with the complexities involved with speech/natural language understanding include the SCAN system of Wermter [95] which was designed to flat scanning for written texts. The TRAINS planning system of Allen [3] can reason about time, actions and events within a natural language framework. The PARSEC system of Jain [47] uses neural networks for the control of sentence parsing and symbolic structures for performing the necessary transformations.

5.7 Summary

Modular hybrid systems are the most common and powerful type of hybrid systems. However, a continuum of modular system complexity exists that is dependent on the degree of coupling and inter-module communication required by the application. Many of the earlier systems only required a limited uni-directional channel of communication between the modules. Some more complex newer systems have been developed to manage tasks with a high degree of dimensionality. Such systems consist of several cooperating modules with several channels of communication often including some form of feedback. Although modular hybrid systems communications may become complex, the individual modules still remain conceptually as separate neural network or rule-based elements.

6 CONCLUSIONS

This paper has reviewed some approaches of hybrid systems in terms of how systems composed of neural and rule-based elements may interact. Three categories of hybrid systems have been identified, namely the unified, transformational and modular types. Each hybrid type is carrying out its own particular subsymbolic to symbolic data manipulation and information flow. In order to give a greater appreciation of the need for hybrid systems the necessary background knowledge of the strengths and weaknesses of neural networks and symbolic systems has been reviewed. The importance of the underlying theoretical considerations of subsymbolic/symbolic computing is often neglected. We have tried to place the various hybrid architectures in context with their underlying connectionist/symbolic representations.

The choice of which hybrid architecture to use is highly dependent upon the task. For example, if only a simple interpretation of neural network output units is required then a modular hybrid system with sequential coupling will be appropriate. Modular hybrid systems scale up quite well and a system can be composed of many connectionist and symbolic modules performing a variety of tasks in different configurations. Transformational architectures become useful when prior domain knowledge in symbolic form is available. This knowledge can then be enhanced with training examples through the inductive learning capabilities of a connectionist network. One of the most promising techniques in the transformational hybrid category is symbolic rule extraction from neural networks. Rule extraction can act as a bridge between the connectionist and symbolic levels conferring several benefits the most important of which is giving connectionist networks an explanation facility.

7 ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers for their comments on an earlier version of this paper.

References

[1] A. Aamodt and E. Plaza. Case based reasoning: foundational issues, methodological variations and system approaches, *AI Communications*, 7(1):39-59, 1994.

[2] A. Adgar, C. Emmanouilidis, J. MacIntyre, P. Mattison, K. McGarry, G. Oatley and O. Taylor. The application of adaptive systems in condition monitoring. In R. Rao, editor, *International Journal of Condition Monitoring and Diagnostic Engineering Management*, 1(1):13-17, 1998.

[3] J. F. Allen, L. K. Schubert, G. Ferguson, P. Heeman, C. H. Hwang, T. Kato, M. Light, N. G. Martin, B. W. Miller, M Poesio and D. R. Traum. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI* 7:7-48, 1995.

[4] V. Ajjanagadde. Reasoning with function symbols in a connectionist system. In *Proceedings of the Cognitive Science Conference*, pages 285-292, 1990.

[5] V. Ajjanagadde and L. Shastri. Reasoning with rules and variables in neural networks. In S. Goonatilake and S. Khebbal, editors *Intelligent Hybrid Systems*, pages 209-220. John Wiley and Sons, Chichester, 1995.

[6] R. Andrews, J. Diederich and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems*, 8(6):373-389, 1995.

[7] R. Andrews and S. Geva. Rules and local function networks. *Proceedings of the Workshop on Rule Extraction From Trained Artificial Neural Networks*, AISB96, Brighton UK, April 1996.

[8] J. A. Barnden and K. J. Holyoak, editors. *Advances in Connectionist and Neural Computation Theory*, Volume 3. Ablex Publishing Corporation, 1994.

[9] W. Becraft, P. Lee and R. Newell. Integration of neural networks and expert systems for process fault diagnosis. *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 1-2:832-837, 1991.

[10] J. Benitez, J. Castro and J, I. Requena. Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks*, 8(5):1156-1164, 1997.

[11] G. Bologna and C. Pellegrini. Accurate decomposition of standard MLP classification responses into symbolic rules. *Proceedings of the International Work Conference on Artificial and Natural Neural Networks*, 616-627, Lanazrote, Canaries, 1997.

[12] R. Brent. Fast training algorithms for multilayer neural nets. *IEEE Transactions on Neural Networks*, 2(3):346-354, 1991.

[13] M. Caudill. Using neural networks: hybrid expert networks, *AI Expert*, pages 49-54, 1990.

[14] T. Corbett-Clarke and L. Tarassenko. A principled framework and technique for rule extraction from multi-layer perceptrons. *Proceedings of the 5th International Conference on Artificial Neural Networks*, pages 233-238, Cambridge, England, 7-9th July 1997.

[15] J. Diederich. Instruction and high-level learning in connectionist networks, *Connection Science*, 1(2):161-180, 1989.

[16] G. Dorffner. Radical connectionism: a neural bottom-up approach to AI. In G. Dorffner, editor, *Neural Networks and a New AI*, pages 93-132, Chapman and Hall, London, UK, 1996.

[17] M. G. Dyer. Distributed symbol formation and processing in connectionist networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:215-239, 1990.

[18] P. A. Egri and P. F. Underwood. HILDA: Knowledge extraction from neural networks in legal rule based and case based reasoning, *Proceedings of the IEEE International conference on neural networks*, 1(1-6):1800-1805,1995.

[19] J. A. Feldman and D. H. Ballard. Connectionist models and their properties, *Cognitive Science*, 6:205-254, 1982.

[20] J. A. Feldman and G. Lakoff and D. R. Bailey and S. Narayanan and T. Regier and A. Stolcke. The first five years of an automated language acquisition project, *AI Review*, 8, 1996.

[21] J. Fodor and Z. Pylyshyn. Connectionism and cognitive architectures: a critical analysis, *Cognition*, 28:3-72, 1988.

[22] J. Fodor and Z. Pylyshyn. Structured connectionist models and language learning, *Artificial Intelligence*, 7(5):301-312, 1993.

[23] L. Fu. Learning capacity and sample complexity on expert networks, *IEEE Transactions on Neural Networks*, 7(6):1517-1520, 1996.

[24] L. Fu. Integration of neural heuristics into knowledge based inference, *Connection Science*, 1(3):325-340, 1989.

[25] L. Fu. Mapping rule-based systems into neural architectures, *Knowledge-Based Systems*, 3(1):48-56, 1990.

[26] S. Gallant. Connectionist expert systems, *Communications of the ACM*, 31(2):152-169, 1988.

[27] G. Garson. Interpreting neural-network connection weights, *AI Expert*, pages 47-51, 1991.

[28] C. L. Giles, C. B. Miller, H. H.Chen, G. Z. Sun and Y. C. Lee. Learning and extracted finite state automata with second-order recurrent neural networks, *Neural Computation*, 4(3):393-405, 1992.

[29] C. Lee Giles and C. W. Omlin. Extraction, insertion and refinement of symbolic rules in dynamically driven recurrent neural networks. *Connection Science*, 5:307-337, 1993.

[30] S. Goonatilake and S.Khebbal. *Intelligent Hybrid Systems*, John Wiley and Sons, Chichester, 1995.

[31] R. Gorman and T. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets, *Neural Networks*, 1:75-89, 1988.

[32] M. Gutknecht and R. Pfeifer. An approach to integrating expert systems with connectionist networks, *AI Communications*, 3(3):116-127, 1990.

[33] L. Hall and S. Romaniuk. Performance issues of a hybrid symbolic, connectionist learning algorithm. In A. Kandel, editor, *Hybrid Architectures for Intelligent Systems*, pages 106-133, 1992.

[34] D. Handelman, S. Lane and J. Gelfand. Robotic skill acquisition based on biological principles. In A. Kandel, editor *Hybrid Architectures for Intelligent Systems*, pages 301-327, CRC Press, 1992.

[35] V. Honavar and L. Uhr. Generative learning structures and processes for generalised connectionist networks, *Information Sciences*, 70:75-108, 1993.

[36] V. Honavar. Symbolic artificial intelligence and numeric artificial neural networks: Towards a Resolution of Dichotomy. In R. Sun and L. Bookman, editors, *Computational Architectures Integrating Symbolic and Neural Processes*, pages 351-388, Kluwer, New York, 1994.

[37] V. Honavar and L. Uhr. Integrating symbolic processing systems and connectionist networks, In S. Goonatilake and S.Khebbal, editors *Intelligent Hybrid Systems*, pages 177-208. John Wiley and Sons, Chichester, 1995.

[38] S. Harnad. The symbol grounding problem, *Physica D*, 42:335-346, 1990.

[39] J. A. Hendler Marker passing over microfeatures: towards a hybrid symbolic connectionist model, *Cognitive Science*, 13:79-106, 1989.

[40] M. Hilario. An overview of strategies for neurosymbolic integration. In R. Sun and L. Bookman, editors, *Computational Architectures Integrating Symbolic and Neural Processes*, Kluwer Academic Publishers, 1994.

[41] G.E. Hinton. Connectionist learning procedures, *Artificial Intelligence*, 40:185-234, 1989.

[42] G.E. Hinton. How neural networks learn from experience. *Scientific American*, pages 105-109, September 1992.

[43] J. Holland. Escaping brittleness. In R. Michalski, J. Carbonell and T. Mitchell, editors *Machine Learning*, Morgan Kaufmann, Vol. 2, pages 593-623, 1986.

[44] S. Hruska, D. Kuncicky and R. Lacher. Hybrid learning in expert networks. *Proceedings of the International Joint Conference on Neural Networks*, Vol. 2, pages 117-120, 1992.

[45] D. Hudson, P.W Banda, M.E Cohen and M.S Blois. Medical diagnosis and treatment plans derived from a hybrid expert system. In A. Kandel, editor, *Hybrid Architectures for Intelligent Systems*, pages 330-244, CRC Press, 1992.

[46] I. Ivanova and M. Kubat. Initialization of neural networks by means of decision trees, *Knowledge-Based Systems*, 8(6):333-343, 1995.

[47] A. N. Jain. Parsing complex sentences with structured connectionist networks. *Neural Computation*, 3(1):110-120, 1991.

[48] J. S. R. Jang and C. T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions Neural Networks*, 4:156-158, 1993.

[49] S. Kang and R. Kwon. Hybrid knowledge based architecture for building an intelligent non-destructive signal inspection system, *Knowledge Based Systems*, 8(1):21-31, 1995.

[50] R. Kerber, B. Livezey and E. Simoudis. A hybrid system for data mining, In S. Goonatilake and S. Khebbal, editors, *Intelligent Hybrid Systems*, pages 121-421. John Wiley and Sons, Chichester, 1995.

[51] R. Khosla and T. Dillon. Fusion of knowledge-based systems and neural networks and applications. *1st International Conference on Knowledge-Based Intelligent Electronic Systems*, pages 27-44, Adelaide, Australia, 21st-23rd May 1997.

[52] C. Kirkham and T. Harris. Development of a hybrid neural network/expert system for machine health monitoring. In R. Rao, editor, *Proceedings of the 8th International Congress on Condition Monitoring and Engineering Management*, pages 55-60, 1995.

[53] C. Kitzmiller and J. Kovalik. Coupling symbolic and numeric computing in knowledge based systems, *AI Magazine*, 8(2):85-90, 1987.

[54] R. Lacher, S. Hruska, D. Kuncicky. Backpropagation learning in expert networks, *IEEE Transactions on Neural Networks*, 3(1):62-72, 1992.

[55] D. B. Leake. *CBR in context: the present and future*, AAAI Press/MIT Press, California, 1996.

[56] B. Lees, B. Kumar, A. Mathew, J. Corchado, B. Sinha and R. Pedreschi. A hybrid case-based neural network approach to scientific and engineering data analysis, In *Proceedings of the Eighteenth Annual International Conference of the British Computer Society Specialist Group on Expert Systems*, pages 245-260, Springer, Cambridge, 1998.

[57] D. Lenat. CYC: a large scale investment in knowledge infrastructure, *Communications of the ACM*, 38(11):33-38, 1995.

[58] R. P. Lippmann. An introduction to computing with neural nets, *IEEE ASSP Magazine*, pages 4-22, April 1987.

[59] D. R. Mani and L. Shastri. Reflexive reasoning with multiple instantiation in a connectionist reasoning system with a type hierarchy, *Connection Science*, 5(4):205-241,1993.

[60] J. MacIntyre, P. Smith and T. Harris. Industrial experience: the use of hybrid systems in the power industry. In L. Medsker, editor, *Intelligent Hybrid Systems*, pages 57-74, Kluwer Academic Press, 1995.

[61] L. Medsker *Hybrid Neural Network and Expert Systems*, Kluwer Academic Publishers, Boston, 1994.

[62] C. McMillan, M. Mozer and P. Smolensky. The connectionist scientist game: rule extraction and refinement in a neural network. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pages 424-430, Hillsdale, NJ: Lawrence Erlbaum, 1991.

[63] R. Miikkulainen. *Subsymbolic natural language processing.* MIT Press, Cambridge, MA, 1993.

[64] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, pages 281-294, 1989.

[65] A. Newell and H. Simon. Computer science as empirical enquiry: symbols and search, *Communications of the ACM*, 19(3):113-126, 1976.

[66] A. Newell. The knowledge level, *Artificial Intelligence*, 18:82-115, 1982.

[67] C. W. Omlin and C. L. Giles. Extraction and insertion of symbolic information in recurrent neural networks. In V. Honavar and L. Uhr, editors, *Artificial Intelligence and Neural Networks: Steps Towards principled Integration*, pages 271-299, Academic Press, San Diego, 1994.

[68] C. W. Omlin and C. L. Giles. Extraction of rules from discrete-time recurrent neural networks, *Neural Networks*, 9(1):41-52, 1996.

[69] J. Park and I. W. Sandberg. Universal approximation using radial basis function networks, *Neural Computation*, 3:246-257, 1991.

[70] J. Pearl. Heuristics, New York: Addison-Wesley, 1984.

[71] T. Peterson and R. Sun. An RBF network alternative for a hybrid architecture. *International Joint Conference on Neural Networks*, Ancorage, AK, May, 1998.

[72] E. Prem. Symbol grounding revisited, *Technical Report TR-94-19*, Austrian Research Institute for Artificial Intelligence, Vienna, 1994.

[73] E. Prem. Dynamic symbol grounding, state construction and the problem of teleology, *Lecture Notes in Computer Science*, 930:619-626, 1995.

[74] J. R. Quinlan. Induction of decision trees, *Machine Learning*, 1:81-106, 1986.

[75] S. Romaniuk and L. Hall. Injecting symbol processing into a connectionist model. In B. Soucek, editor, *Neural and Intelligent Systems Integration*, pages 383-405, John Wiley and Sons, 1991.

[76] T. Samad. Hybrid distributed/local connectionist architectures. In A. Kandel, editor, *Hybrid Architectures for Intelligent Systems*, pages 200-218, 1992.

[77] I. Sethi. Entropy nets: from decision trees to neural networks, *Proceedings of the IEEE*, 78(10):1605-1613, 1990.

[78] N. E. Sharkey and A. J. C. Sharkey. An analysis of catastrophic interference, *Connection Science*, 7(3):301-329, 1995.

[79] L. Shastri. The relevance of connectionism to AI: a representation and reasoning perspective. In J. Barnden, editor, *Advances in Connectionist and Neural Computation Theory*, Vol. 1, pages 259-283, 1991.

[80] L. Shastri. and V. Ajjanagadde. From simple associations to systematic reasoning - a connectionist representation of rules variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:3:417-451, 1993.

[81] J. Shavlik, R. Mooney, G. Towell. Symbolic and neural learning algorithms: an experimental comparison, *Machine Learning*, 6:111-143, 1991.

[82] P. Smolensky. On the proper treatment of connectionism, *Behavioral and Brain Sciences*, 11:1-74, 1988.

[83] S. Suddarth and A. Holden. Symbolic-neural systems and the use of hints for developing complex systems, *International Journal of Man-Machine Studies*, 35:291-311, 1991.

[84] R. Sun. CONSYDERR: a two level hybrid architecture for structuring knowledge for commonsense reasoning. *Proceedings of the IEEE International Conference on Neural Networks*, pages 1475-1480, 1994.

[85] R. Sun. A microfeature based approach towards metaphor interpretation *Proceedings of the International Joint Conference on Artificial Intelligence*, 1:424-429, Montreal, Canada, 1995.

[86] R. Sun. Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):214-295, 1995.

[87] R. Sun and F. Alexandre, Proceedings of the Workshop on Connectionist-Symbolic Integration: From Unified to Hybrid Approaches, Montreal, 1995.

[88] H. Tirri. Replacing the pattern matcher of an expert system with a neural network, In S. Goonatilake and S.Khebbal, editors *Intelligent Hybrid Systems*, pages 47-62. John Wiley and Sons, Chichester, 1995.

[89] G. Towell and J. Shavlik. Knowledge-based neural networks, *Artificial Intelligence*, 70:119-165 1994.

[90] G. Towell and J. Shavlik. Extracting rules from knowledge-based neural networks, *Machine Learning*, 13:71-101, 1993.

[91] S. Thrun. Extracting provably correct rules from artificial neural networks. *Technical Report IAI-TR-93-5*, University of Bonn, Institut fur Informatik III, D-53117, Bonn, 1993.

[92] S. Thrun. Extracting rules from artificial neural networks with distributed representations. In G.Tesauro, D. Touretzky and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, San Mateo, CA, MIT Press, 1995.

[93] A. Ultsch, R. Mantyk and G. Halmans. Connectionist knowledge acquisition tool: CONKAT. In J. Hand, editor, *Artificial Intelligence Frontiers in Statistics: AI and statistics III*, pages 256-263, Chapman and Hall, 1993.

[94] P. E. Utgoff. Perceptron trees: a case study in hybrid concept representations, *Connection Science*, 1(4):377-391, 1989.

[95] S. Wermter. *Hybrid Connectionist Natural Language Processing* Chapman and Hall, Thompson International, London, UK. 1995.

[96] V. Weber and S. Wermter. Using hybrid connectionist learning for speech/language analysis. In S. Wermter, E. Riloff, E.Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 87-101, Springer-Verlag, 1996.

[97] S. Wermter. Hybrid approaches to neural network-based language processing, *Technical Report TR-97-030*, International Computer Science Institute, Berkeley, California, 1997.

[98] S. Wermter and V. Weber. SCREEN: learning a flat syntactic and semantic spoken language analysis using artificial neural networks *Journal of Artificial Intelligence Research*, 6:35-85, 1997.

[99] S. Wermter. Knowledge extraction from transducer neural networks. *Journal of Applied Intelligence*, 1999.

[100] A. Wilson and J. Hendler. Linking symbolic and subsymbolic computing. *Connection Science*, 5:395-414, 1993.

[101] B. Yao and Y. He. A hybrid system for case-based reasoning. *World Congress on Neural Networks, International Neural Network Society Annual meeting*, 4:442-446, 1994.

[102] H. Xu and R. Baird. Synergism of neural networks and expert systems for system identification, *Expert Systems with Applications*, 5:25-33, 1992.