Manufacturing Feature Identification for Intelligent Design

Alice E. Smith, Ph.D., P.E.
Department of Industrial Engineering
University of Pittsburgh
Pittsburgh, PA 15261

Cihan H. Dagli, Ph.D.
Department of Engineering Management
University of Missouri - Rolla
Rolla, MO 65401

A Chapter in *Intelligent Systems in Design and Manufacturing* (C. Dagli and A. Kusiak, Editors), forthcoming from ASME Press.

1. Overview of This Chapter

Design is the first and most influential of manufacturing activities. It is also the most creative and least structured. Design relies greatly on the designer's expertise, judgment and use of heuristics. Recent developments in Computer Aided Design (CAD) have improved the speed and quality of design, however intelligent techniques can further assist the process. Knowledge based systems are built to contain human reasoning ability so that investigations and decisions are made in an efficient, comprehensive and standardized way. Knowledge based systems can be used in manufacturing design to replicate human abilities to assemble, manipulate and specify features in an acceptable, and even creative, design. Artificial neural networks are excellent pattern classification techniques when dealing with noisy and complex data. They adapt well to new and changed data. Neural networks offer assistance with feature identification tasks by classifying low level features, or their components, for use by higher level methods (e.g. knowledge based systems or human designers) to manipulate and combine. The objective in using intelligent computing for design is to consistently and efficiently develop product designs that are robust during manufacture. The design should also result in functionally adequate, quality products which are economical to manufacture.

This chapter will start with an overview of design in a manufacturing context and an explanation of why features are the fundamental building blocks of design. A brief introduction to the components of knowledge based systems leads to an overview of some operational and prototypic feature based design systems. There are serious limitations to expert systems for feature identification, and these introduce the need for neural network pattern recognition. Various artificial neural network paradigms are described, with emphasis on the multi-layered perceptrons trained with backpropagation. Previous work describing and building neural network based feature design systems will be discussed. Finally, the current problems with neural networks are presented along with the outlook for composite intelligent systems.

This chapter uses the terms expert system and knowledge based system interchangeably although, strictly speaking, the latter is more descriptive. Design systems are based on knowledge from different sources, and do not generally replicate one expert. The terminology used for neural networks is common although other names do exist (e.g. neurons can be called processing units).

2. Introduction to the Design Process

The design process begins the manufacturing life cycle by specifying shapes, dimensions, materials, tolerances, performance and assemblies of products, parts and mechanisms. After design, process planning, production, quality control, distribution and service take place. Design is important for these downstream activities. It affects process planning, that is the choice of manufacturing methods, tools, routing and sequencing. Design impacts the inventory and supplier process by specifying parts and materials. Design affects the manufacturability of the product, which has a direct bearing on the unit cost and quality of the end product. Design determines in part the product functionality and durability, which can have ultimate effects on marketing, sales and profitability. The final impact of design ranges from 65% to 75% of total manufactured cost (Khoshnevis and Park 1988, Suri and Shimizu 1989, Swift 1987). Design seeks to balance cost efficiency and manufacturability with quality and functionality, and as such must satisfy many, sometimes conflicting, constraints.

When viewing the design function, three classes emerge (Brown and Chandrasekaran 1988). Class 1 design is innovative, breaking barriers to develop a new product or process. Class 2 involves new techniques or requirements for known products or problems. Class 3 selects previously known alternatives. Class 3 is still complex since there are many components and numerous combinations. Present intelligent systems are best capable of addressing Class 3 design efforts since they are not sophisticated enough to create new and innovative design solutions.

Design tasks are normally done in a linear sequence beginning with fundamental shapes and components, and refining these until the desired part is reached. Generic steps are (1) identify needs, (2) set strategies, (3) establish design concept, (4) select feasible alternatives, (5) select and specify parameters, and (6) evaluate and implement (Suri and Shimizu 1989). Design must specify these aspects of each part:

- Shapes
- Dimensions
- Materials
- Tolerances
- Static and Kinetic Interactions
- Assemblies.

CAD systems support engineering functions such as mass properties calculations, interference checking and geometry definitions for finite elements, drafting, and numerical control, but considerable human interaction is still needed for successful design (Shah and Rogers 1988). This is because CAD systems fall short in two important areas of design: (1) feature identification and synthesis, and (2) creativity and heuristic use. Intelligent systems can replicate some of the human abilities of recognition and reasoning needed in these two areas, especially for Class 3 designs which do not require creativity or innovation.

3. Importance of Features

Mechanized systems addressing design, such as CAD solid modelers, are based upon features. Features are primitive or low level designs with their attributes, qualifiers and restrictions which affect functionality and/or manufacturability. Features can describe form (size and shape), precision (tolerances and finishing), or materials (type, grade, properties and treatment) (Shah and Rogers 1988), and vary with product and manufacturing process. Table 1 partially lists typical features for the major manufacturing processes.

Table 1. Typical Features for Manufacturing Processes (adapted from Cunningham and Dixon 1988).

Forging	Extrusion	Stamping
Planes	Webs	Slots
Ribs	Ribs	Fingers
Bosses	Tongues	Corners
Fillets	Walls	Holes
Webs	2D Intersections	Cut-Outs
3D Corners	Hollowness	Notches
Bends/Twists	Corner Breaks	Windows
Casting	**Tool & Die**	**Structural**
Planes	Circles	Plates
Cylindricality	Webs	Shells
Projections	Walls	Beams
Depressions	Corner Breaks	Frames
Hulls	Slots	Trusses
Solids	Hollowness	Columns
Holes	Tongues	Arches

To accomplish design, features must be created, deleted, modified, copied, moved, detailed, interrogated, and have attached properties, attributes, and restrictions (Pratt 1988). Existing CAD systems have been successful at improving design through manipulating geometric data rapidly and precisely, in both a two dimensional (surface) and three dimensional (solid or volume) format. Volume representation is natural and useful in defining machining operations, while surface representation is needed to access the faces of the volumes for positioning, attaching attributes and tolerancing (Kusiak 1990). Drawing storage, modification and costing have also been greatly eased through mechanization. Since design implies subsequent redesign, the ability to quickly generate alternate versions of a design is vital, and CAD fulfills this need.

Central to a feature based CAD system is the library of features. This library contains the features needed for product design with their associated properties. The library eliminates the need to define and create features during the design process. The designer selects from the library, then modifies the feature to suit the product. Dimensions, tolerances, orientation, placement, materials and so on are individualized, then the product is constructed by combining the individualized features. Although the

library is efficient for features stored within, new and fundamentally changed features or properties cannot be accommodated without expanding or modifying the library.

4. Overview of Knowledge Based Systems

CAD systems cannot replicate heuristics and reasoning when combining and specifying features to form a product. To standardize and to improve the quality and quantity of design, expert systems can be used to supplement human abilities, which vary significantly with the person, the design task, and the environmental conditions and constraints. Knowledge based systems, also known as expert systems, are formulated to replicate and improve upon human reasoning, handling symbolic processing and logical structures well (Harmon and King 1985). Knowledge is commonly stored either in production rules or in frames, creating the knowledge base. The former are structures normally of If/Then type (*modus ponens* reasoning) with antecedent conditions, attribute matching, and consequent actions or conclusions. Frames store information according to components with attributes, a type of object oriented programming. Components are arranged hierarchically so that they may inherit, pass on and relate to attributes of other components.

The reasoning of a knowledge based system is accomplished by the inference engine, which directs the query of the knowledge base. Inference engines are forward chaining (data driven or antecedent reasoning) and/or backward chaining (goal driven or consequent matching) (Swift 1987). Forward chaining systems begin with the known facts and search solution avenues until each is exhausted, or until a conclusion is reached. This type of reasoning is inefficient when large numbers of facts and possible solutions exist. Backward chaining begins with desired goals and works through antecedent conditions until all are matched with known facts, or avenues are exhausted. Some hybrid manufacturing systems have effectively combine both types of reasoning (Kusiak 1990).

5. Applying Knowledge Based Systems to Features

The design area is broader than many others which apply expert systems - there are enormous ranges of artifacts, tasks and domains (Kusiak 1990). Many prototypic and operational systems have been developed for different processes of manufacturing design. Some are rule based and some are frame oriented, some are integrated with CAD modelers and others are not. For feature extracting expert systems, a hierarchical set of features is normally developed to reduce combinatorial explosion, then forward chaining logic is applied since the starting point (the input geometric model) is completely known (Hirschtick and Gossard 1986). This hierarchy of features is essentially analogous to the feature library discussed earlier.

CASPER is a frame based Lisp system operating with a color CAD system to assist aluminum casting design (Luby et al. 1986). First-Cut is also object oriented and is integrated with a solid modeler (Cutkosky et al. 1988). It assists machining design, and also formulates high level process plans. Impard similarly uses a solid modeler to advise on injection molding part manufacturability with a limited number of features (Vaghul et al. 1985). Rules are also used in Extrusion Advisor (Hirschtick and Gossard 1986). It takes a different approach by identifying features which will cause difficulties in manufacturing using characteristic patterns for walls, hollows, edges, etc. Standard roller chain-drive design and process selection was addressed by CDDES, a system combining frames and rules which converses in English and Chinese (Wang and Yu 1988). Swift developed a rule based system integrated with a two dimensional modeler consisting of 246 rules aimed at assembly (Swift 1987). It advises the designer of difficulties the proposed design presents for automatic handling, and suggests remedies. It also estimates the equipment cost for the designed components. A recent investigation partitions a knowledge base for design based on dynamically acquired user preferences and on operational information (Sykes and White 1991). This approach improves search

for good design solutions, and is more readily accepted by users because of the preference capability.

There are two large classes of problems when using knowledge based systems for feature recognition and design. First, the recognition itself is still best done by humans with their faculty for categorizing visual images into groups; although knowledge based systems can effectively work with the features once they are identified. To design a set of rules or frames so that a given feature can be properly identified is difficult and lengthy. For example, an expert system may be fed the geometry of a heat sink with two fins and identify it as three walls or eight milled surfaces (Hirschtick and Gossard 1986).

A second inherent problem with expert systems is their somewhat brittle approach to problems. Systems cannot interpolate or extrapolate knowledge and must work with a well defined, clearly bounded domain. Decisions are dependent on the situational factors fitting the knowledge base. Closely related to the brittleness problem, is that knowledge based systems rely on how the problem domain was originally defined and how the knowledge was extracted and recorded. Problem selection and knowledge engineering are crucial to any intelligent system's performance, and the latter is the primary development bottleneck.

6. Applicability of Neural Networks

Feature recognition is best accomplished by humans whose eye/brain system is capable of immediate and robust performance. Artificial neural networks were originally designed to replicate some of this human ability and, while the technology is still far from reaching human standards, it does offer significant advantages over other computing options for feature recognition. Based upon the biological brain's process of thought transmission, stimuli recognition and physical response, neural networks are massively parallel computing mechanisms. They are valued because they deal successfully with noisy, uncertain and partial inputs, and they are fault tolerant and degrade gracefully. They are useful for feature recognition because they can generalize

their knowledge to corrupted, partial, modified or new inputs. Instead of a static library of features, a neural network is capable of interpolating between features and of learning new features.

There are many neural network paradigms and variations, and the reader is directed to comprehensive works such as (Lippmann 1987, Simpson 1990, Wasserman 1989). All networks do share common aspects and mechanics, which are described briefly here. A network works by directing vectorized input through parallel and serial elements where it is evaluated and combined, and finally directed to a meaningful vectorized output state, as shown in Figure 1. Most neural networks learn input/output relationships by storing a distributed model as fixed weights along each connection between neurons.

Figure 1. Generic Components of a Neural Network.

In most neural network paradigms, weights are initially random and allowed to change during training as the network seeks the best combination of weights to represent the input/output model. Training can be done by guiding the network towards the optimum model using desired output, or teacher vectors. This guiding is called supervised training and has the advantage of explicitly declaring the outputs the final neural network should model. A second form of training is self organization, or

competitive learning. where the network changes its connection weights so that similar input vectors are grouped together. This, too, forms an input/output model but the exact form of the model cannot be specified precisely prior to training.

Number of neurons may vary from layer to layer, as may the evaluation and combination methods. Combination is typically done by summing the input times the weight of each connection across all neurons, sometimes with an added bias term. The summation is compared to a threshold value and then subjected to an activation or transfer function (often the sigmoid, step or linear threshold) which calculates an output. Feedforward networks have connections only to the next layer, while feedback or recurrent networks have connections to the next layer, to a previous layer, or within a layer. Recurrent networks are usually used to model relationships which have a temporal aspect, while feedforward networks generally handle static relationships.

Generic pattern classification or recognition is accomplished by preprocessing an input into binary and/or continuous vector components. A supervised network will have been trained to signal an output class based on the input vector. Again, the output may be binary or continuous, but most pattern classification networks use binary outputs where a 1 signals a match with that class of patterns. Pattern classification can be hetero-associative, that is association of an input vector with a class grouping. Another aspect of pattern recognition is not the classification of patterns, but the output of an exemplar pattern when the input pattern is partial or noisy. This is termed content addressable or auto-associative memory (input of degraded pattern A outputs complete pattern A).

7. Multilayered Perceptrons Trained by Backpropagation

The most well known and commonly used neural network paradigm is the multilayered perceptron trained by backpropagation (Rumelhart et al. 1986). This is popularly called a backpropagation network and is shown in Figure 2. This network is discussed in length here because it is versatile, simple and handles pattern classification tasks, of which feature identification is one, well. Backpropagation can accommodate

both binary and continuous input or output. From a statistical viewpoint, backpropagation is the optimal supervised training method, as it converges to a non-linear estimator with the maximum likelihood of being true (Movellan 1990, Werbos 1988). Backpropagation is readily available in software form, and increasingly, in hardware form. For these reasons backpropagation, or a variation of backpropagation, is usually the network of choice for pattern classification when teacher data is available.

Figure 2. Typical Backpropagation Network During Training With Sigmoidal Transfer Function.

A backpropagation network is a feedforward network, typically consisting of an input layer, one or more hidden layers, and an output layer all containing varied numbers of neurons. It has been arithmetically proven that this network with at least three layers (input, output and one hidden layer) and a non-linear transfer function can implement any function (Hornik et al. 1990, Hornik 1991). However, for continuous input vectors, the superiority of using two hidden layers has been theoretically and empirically noted (Lapedes and Farber 1988, Smith and Dagli 1991).

Backpropagation works by using the theory of least squares - calculating the derivatives of error with respect to the connection weights, and adjusting the weights based on steepest error surface descent. Weights are modified during training until they

reach a stable state (convergence); this translates to the network achieving a state of error minimization. Since weights are usually adjusted after each vector input instead of after the whole training set, the resultant descent is not necessarily the steepest, but closely approximates it (Khanna 1990). Although classical backpropagation cannot escape from local minima during its descent along the error surface, in practice a local minimum significantly far from the global minimum is rarely encountered (Smith et al. 1991).

There are several critical factors when formulating and training a backpropagation network:

1. Number of Input Neurons.
2. Number of Output Neurons.
3. Number of Hidden Layers.
4. Number of Neurons in Each Hidden Layer.
5. Size and Composition of Training Set.
6. Size and Composition of Test Set for Trained Network Validation.
7. Training Rate and Modifications.

The number of input and output neurons are usually dictated by the problem (e.g. a network categorizing features, each described by an input vector of length 40, into 4 categories might have 40 input neurons and 4 output neurons).

Practically speaking, choice of number of hidden neurons is tied to choice of number of hidden layers. Hidden layers can provide translation and scaling invariance (Roth 1990), i.e. the positioning and size of the feature would not critical to its identification. In applications, the choice of number of hidden layers is generally confined to one or two. Once the number of hidden layers is selected the size of each layer can be estimated. The objective is to find the least number of hidden neurons which can achieve adequate performance. Decreasing number of hidden neurons increases generalization and is more computationally efficient, but impairs the network's ability to learn. Overfitting with too many hidden neurons tends to decrease generalization because of

training set "memorization", and is computationally inefficient. Estimating optimum hidden neurons is usually done by trial and error. Some heuristics involving the similarity and regularity of the training set have been proposed (Ahmad and Tesauro 1989, Baum and Haussler 1989, Gutierrez et al. 1989, Kung and Hwang 1988, Perugini and Engeler 1989).

The size and composition of the training set are critical. It should be a sample with fixed probability distribution chosen randomly from the expected values, which also have a fixed probability distribution (Hecht-Nielsen 1989). The training set is presented to the network many times during training. It should be in random order and contain all the training features. Upper and lower bounds for number of training pairs have been directly related to numbers of neurons and weights, and rate of classification error (Baum and Haussler 1989). It has also been shown that as training set size increases, so does probability of correct network pattern classification (Ahmad and Tesauro 1989).

A disadvantage of backpropagation is the long training time it often takes a network to work its way down to the bottom of the error surface. This is in part determined by the step size or training rate. A large training rate implies large step size which will decrease convergence time, however moving too quickly down the error surface increases the chance of non-convergence (Kung and Hwang 1988) and paralysis (Wasserman 1989). A small learning rate avoids these problems but increases training time. The learning rate is often constant through training, but can be decreased during training to avoid network instability. It may also be somewhat randomized so that the network can arbitrarily "jump" out of local minima. Training rate can be sized based on the number of hidden neurons (Kung and Hwang 1988).

The training disadvantage of backpropagation is not significant when building feature identification systems. Training can be assumed to be an off line, infrequent task and therefore speed is not essential. The requirement of teacher vectors is also not a problem for feature identification as they are readily available. A problem that is significant with

backpropagation for feature identification is that once a network is trained, new patterns cannot easily be added to its internal model. It is usually best to build and train a new network to incorporate the new feature(s).

8. Other Neural Network Paradigms Useful for Feature Identification

While backpropagation is the network of choice for most pattern classification tasks, there are other neural network paradigms which offer some advantages in specific areas. The other networks discussed below are counterpropagation networks, adaptive resonance theory, probabilistic networks and fuzzy networks. These networks have certain capabilities for feature identification.

8.1 Counterpropagation Networks

Counterpropagation networks were introduced by Hecht-Nielsen as a combination of two established neural network paradigms, backpropagation and the Kohonen self organizing network (Hecht-Nielsen 1987). Because of the long training times for backpropagation networks, counterpropagation aims at the same mapping functionality but with less generality, and significantly less required training cycles. A counterpropagation network can work equally well for binary and continuous vectors. It is essentially a look up table for stored patterns.

The counterpropagation network is defined to be five layers fanning in and then fanning out. Layers 1 and 5 are normalized input layers, layer 3 is a Kohonen layer (Kohonen 1984), and layers 4 and 5 are Grossberg outstar layers (Grossberg 1982). Normalized vector input is directed to the Kohonen layer, which self-organizes, then is passed to the Grossberg layers, which are trained through supervision, for output.

The Kohonen layer works by classifying similar vectors together without the benefit of a teacher vector. It is trained through competitive learning, where only one Kohonen neuron outputs a 1 (turns on or fires) for a given input. All other Kohonen neurons are inactive (output = 0). The winning Kohonen neuron is that which has the largest summation of the dot products between each input vector from layers 1 and 5, and their

weights to layer 3. During training, weights are adjusted only for the winning Kohonen neuron so that eventually training causes each of the weights from the input layers to each Kohonen neuron to become like the average of those vectors being classified (turned on) by that neuron. By itself the Kohonen layer approximates the probability density function of the input vectors (Jakubowicz and Ramanujam 1990, Lippmann 1987). In addition the final weights are organized so that topologically close neurons are sensitive to physically similar inputs (Lippmann 1987).

This network is advantageous in that it is a good classifier of input vectors without extensive training times and can be used for rapid prototyping (Wasserman 1989). The number of classifications must be known *a priori*, and a large amount of training data must be available. In one hard pattern classification task, a counterpropagation network was found to converge more quickly than backpropagation but provide poorer recognition (Shea and Lin 1989).

8.2 Adaptive Resonance Theory (ART)

Adaptive resonance theory, better known as ART, is a good candidate for feature identification because of its unusual ability to acquire new knowledge to an existing network (Carpenter and Grossberg 1987 a, Grossberg 1976). ART is a combination of self organizing and supervised learning, where like patterns are grouped together, but the fineness of the groups is controlled *a priori* by the user through use of a vigilance factor. When new inputs are entered to the network, it checks its stored patterns to see if the new pattern is close to any stored pattern. If it is acceptably close, the closest stored pattern is signaled as the correct classification. If the new pattern is not close to any stored pattern, the new pattern is stored as an additional pattern in the network memory. In this way, new features or significantly altered features can be added to the network memory dynamically without retraining or rebuilding.

ART 1, the original ART network, only handles binary inputs and outputs, but ART 2 handles continuous inputs as well (Carpenter and Grossberg 1987 b). ART 3 is a

parallel search version of ART designed for network hierarchies (Carpenter and Grossberg 1990). ARTMAP is a recent version that rapidly self organizes to stable class mappings of optimal size (Carpenter et al. 1991). ARTMAP handles only binary inputs and outputs. A new version of ARTMAP is fuzzy ART, which will be discussed in Section 8.4.

ART does have disadvantages, particularly in tuning the fineness of the classes using the vigilance factor. The network must be tuned so that it only adds new stored patterns when the pattern is novel enough, as defined by the user. This is often not easy to determine, and successful use of ART may require some trial and error. Besides ART 2 and Fuzzy ART the paradigm handles only binary inputs and outputs, and ART 2 handles only binary outputs. These may be limiting depending on how feature identification is handled.

8.3 Probabilistic Networks

Probabilistic networks are supervised, feedforward classifying networks which use an exponential transfer function and approximate Bayes classifiers (Specht 1990a, Specht 1990b). The principle advantage of a probabilistic network over a backpropagation network is the very much shortened training time since only one pass through the training set is required. Two other advantages over backpropagation are that the network trains successfully with sparse data, and new patterns can overwrite old when retraining.

The probabilistic network stores each and every training pattern, which makes it infeasible for very large domains. It is not clear from the research whether performance of probabilistic networks is equal to backpropagation, but they are an alternative when training time is critical or only a small training set is available. They can also be used for quick prototyping and exploration.

8.4 Fuzzy Networks

There are several versions of fuzzy networks. All share a common base in the seminal work of Zadeh in fuzzy set theory (Zadeh 1965). While the reader is directed

elsewhere for an adequate discussion on fuzzy set theory (e.g. Klir and Folger 1988), briefly fuzzy set theory attempts to capture imprecision. An object belongs to a set with a membership degree, usually between 0 and 1, where 0 indicates the object is definitely not a member of the set and 1 indicates the object is definitely a member of the set. Values in between indicate degrees of membership of the object in the set. Fuzzy set theory provides an ordered method to combine and infer fuzzy relationships.

This folds in nicely with feature identification because of the potential imprecision of some features classes, especially when dealing with identification of new, corrupted or altered features. Several neural network paradigms incorporating fuzzy concepts have been described recently. Probably the most well known is the fuzzy associative memory (FAM) which combines evidence of membership degree with conflict resolution (Bahrami et al. 1991, Kong and Kosko 1992, Kosko 1991). FAMs can be trained by supervision, such as backpropagation, or by self organization, called adaptive FAM. Each FAM rule, or instance, provides a sample point in the model. Unfamiliar input to FAM is estimated by combining known FAM rules through fuzzy set theory. FAMs are especially noted for their robustness.

Other network paradigms incorporating fuzzy concepts include a fuzzy version of backpropagation (Hunt et al. 1991). This can improve standard backpropagation classification accuracy by including membership degrees in the error calculation during training. Another major paradigm adapted for fuzzy use is ART. Fuzzy ART is a continuous extension of ART 1 by using a minimum operator instead of an intersection operator (Carpenter et al. 1991).

9. Feature Recognition With Neural Networks

A trained network can be assigned identification tasks based on features, and then automatically integrate diverse features into a set. For networks to process features, physical attributes must be translated into meaningful input. Preprocessing of data takes

the form of devising a compact scheme which uniquely represents the important differences among features. Figure 3 shows two variations of input vector coding.

Figure 3. Two Schemes for Preprocessing Input Vectors.

For more complex feature detection, more complex preprocessing is needed. First, the number of features must be limited and the ones selected must sufficiently discriminate among classes (Roth 1990). Aspects such as orientation, size, placement and edge detection must be considered. One schema involves breaking at inflection and orientation discontinuities, and keying on the ratio of chord to arc length, the relative lengths scaled to the longest contour, the number and degree of corners, and the degree of inflections and discontinuities (Krishnan and Walters 1988). By concentrating on the piece parts of features, rather than whole objects, some robustness in regards to differences in size and position is gained.

Besides the features themselves networks can be trained for typical attributes, or for a range of attributes. One approach trained a neural network on randomly chosen attribute values from a typical range so that the network could recognize infeasible attribute values (Knapp and Wang forthcoming). For example the feature "hole" was input with typical values of its attributes, depth, diameter, size tolerance, positional tolerance, circularity, straightness and surface finish.

A problem is that the combination of features and attributes is explosive, and the proposed neural network could quickly grow unwieldy in size and training set. One approach to alleviate this is to combine hierarchies of networks, which may even be of different paradigms. The first level of network might separate features into broad categories based on outstanding differences, perhaps through self organization. The second level could discriminate among the broad class by finer aspects of the feature. A third level could group the identified feature by attribute value. Since neural networks work well in hierarchy the potential for this approach is good. This approach is also modular which reduces the need for retraining when new features or attributes must be added.

10. Limitations of Neural Networks

As an immature technology, neural computing has many drawbacks for operational applications. Foremost, there are few commercially available neural network chips, either electronic or optical. Therefore, implementation of this technology usually now depends on simulation by traditional serial computers with special software, and sometimes hardware. There is little doubt, however, that widespread use of neural networks in the near future will rely on hardware implementations.

A second problem is that the construction of neural network architectures is non-determinative. First, a paradigm or cascade of paradigms must be selected. This task alone is significant as new paradigms are being prodigiously developed, and each paradigm has its own advantages and drawbacks. Most developers select several paradigms and evaluate the results of each before going forward with one (Orlando et al. 1990, Shea and Lin 1989). Next, the number of neurons, connections and layers must be chosen. As discussed earlier in this chapter, this is a trial and error procedure with some known properties and some heuristics. The training method and training set also affect the network, as does determining the point at which training is complete and operation may begin.

Structuring feature recognition problems so that they can be conveniently input and output as vectors, either binary or continuous, to neural networks or hierarchies of neural networks is a very difficult problem by itself. Features must be selected and coded in such a way as to distinguish between classes, but not overly encumber data pre and post processing. The selection of training set will ultimately determine the neural network model, therefore the training set must properly reflect the population of features and attributes to be classified. Bias or incompleteness of the training set can cause an incorrectly operating network although networks are fairly robust to less than optimal training sets (Twomey and Smith 1992).

11. Composite Intelligent System Architectures

Given a knowledge based system to assist design, integrating it with both a CAD system to model geometrically and kinematically, and neural networks to recognize and classify features will have synergistic results. The knowledge base serves as user interface, integrator and reasoner. John Hopfield, a pioneer in neural networks, believes that composite systems will be the venue for using neural networks (Myers 1990). Expert systems are good at logic while neural networks excel at fuzzy analysis and pattern recognition. He believes the two will not compete, but rather neural networks will act as front ends to logical back ends. These so called expert networks are becoming popular (Caudill 1991). A prototype expert network for feature identification uses neural networks to extract features from a machine vision system for use by an expert system which previously relied on manual selection of features (McAndless et al. 1991). Figure 4 shows a generic intelligent system for feature identification.

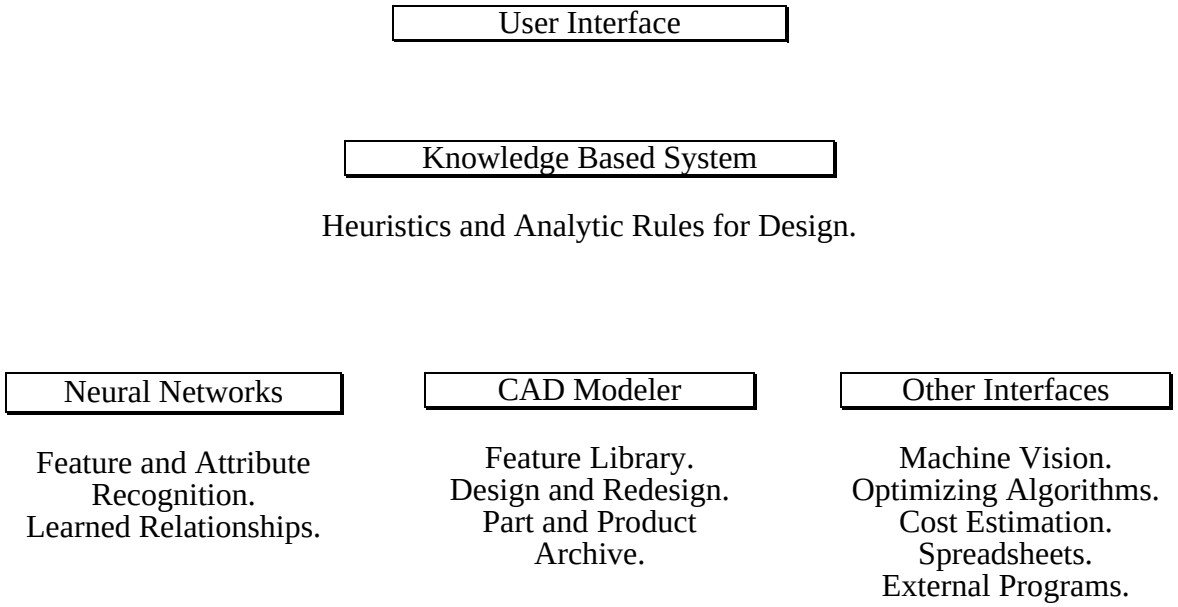

Figure 4. Generic Composite System Architecture.

The knowledge base contains heuristics and analytic rules on the combination and manipulation of features for product design. Features are recognized via pattern recognition neural networks by transforming low level features and attributes into vector representations. A neural network can dynamically feed a CAD feature library. CAD acts as the workhorse, receiving direction from the neural nets on feature identification and from the knowledge base on how to assemble them. CAD saves and stores designed parts for later redesign. It acts as a geometric modeler, feature library, and drawing and specification archive.

12. Concluding Remarks

Since manufacturing design is becoming largely automated through use of CAD modelers, adding intelligence in the form of knowledge based systems and neural networks is synergistic. The designer need not search a catalog of applicable features and attributes, and apply them consistently. The system can do those tasks. Designers can also be more productive in both quantity and quality by being assisted in applying the proper design rules to ensure functionality and manufacturability.

Expert system technology is becoming mature, and has many operational systems. Modeling by integrating an expert system with CAD has been studied by researchers and appliers with good results. A shortcoming of this approach - inability to robustly recognize and classify features - can be addressed by neural networks. Though this technology is far from mature, it has been applied on a limited scale to feature extraction and classification.

Several substantial drawbacks to the intelligent system approach remain. Intelligent systems are all customized, thereby demanding substantial development and testing. Knowledge acquisition varies depending on the problem definition and the available expert sources. Temporal reasoning and kinetic knowledge expansion are areas which need improvement. Preprocessing features and their attributes to vectors to be recognized and classified by neural networks or hierarchies of neural networks requires considerable effort and much trial and error. Selecting neural network paradigms and training the networks are very much based on heuristics and expertise. Training can be lengthy and very dependent on network architecture, and training set and technique. Most neural networks are currently software simulated on serial computers at quite a computational cost.

References

Ahmad, Subutai and Tesauro, Gerald, 1989, Scaling and Generalization in Neural Networks: A Case Study. In David S. Touretzky, Ed., *Advances in Neural Information Processing Systems I* (San Mateo, CA: Morgan Kaufmann Publishers), 160-168.

Bahrami, Ali, Dagli, Cihan and Modarress, Batoul, 1991, Fuzzy Associative Memory in Conceptual Design. *Proceedings of the International Joint Conference on Neural Networks*, I-183-188.

Baum, Eric B. and Haussler, David, 1989, What Size Net Gives Valid Generalization? *Neural Computation*, 1, 151-160.

Brown, D. C. and Chandrasekaran, B., 1988, Expert Systems for a Class of Mechanical Design Activity. In D. T. Pham, Ed., *Expert Systems in Engineering* (Exeter, U.K.: Springer-Verlag).

Carpenter, Gail A. and Grossberg, Stephen, 1987a, A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing*, 37, 54-115.

Carpenter, Gail A. and Grossberg, Stephen, 1987b, ART 2: Stable Self-Organization of Pattern Recognition Codes for Analog Input Patterns. *Applied Optics*, 26, 4919-4930.

Carpenter, Gail A. and Grossberg, Stephen, 1990, ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures. *Neural Networks*, 3, 129-152.

Carpenter, Gail A., Grossberg, Stephen and Rosen, David B., 1991, Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, 4, 759-771.

Caudill, Maureen, 1991, Expert Networks. *Byte*, 16, 108-116.

Cunningham, J. J. and Dixon, J. R., 1988, Designing With Features: The Origins of Features. *Proceedings of the ASME Conference on Computers in Engineering*, 237.

Cutkosky, M. R., Tenenbaum, J. M. and Muller, D., 1988, Features in Process-Based Design. *Proceedings of the ASME Conference on Computers in Engineering*, 557.

Grossberg, Stephen, 1976, Adaptive Pattern Classification and Universal Recoding, II: Feedback, Expectation, Olfaction, and Illusions. *Biological Cybernetics*, 23, 187-202.

Grossberg, Stephen, 1982, *Studies of Mind and Brain* (Dordrecht, Holland: D. Reidel Publishing Co.).

Gutierrez, Mario, Wang, Jennifer and Grondin, Robert, 1989, Estimating Hidden Unit Number for Two-Layer Perceptrons. *Proceedings of the International Joint Conference on Neural Networks*, I-677-681.

Harmon, Paul and King, David, 1985, *Expert Systems* (New York: John Wiley & Sons, Inc.).

Hecht-Nielsen, Robert, 1987, Counterpropagation Networks. *Proceedings of the International Joint Conference on Neural Networks*, II-19-32.

Hecht-Nielsen, Robert, 1989, Theory of the Backpropagation Neural Network. *Proceedings of the International Joint Conference on Neural Networks*, I-593.

Hirschtick, J. K. and Gossard, D. C., 1986, Geometric Reasoning for Design Advisory System. *Proceedings of the ASME Conference on Computers in Engineering Volume 1*, 263.

Hornik, Kurt, 1991, Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4, 251-257.

Hornik, Kurt, Stinchcombe, Maxwell and White, Halbert, 1990, Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks. *Neural Networks*, 3, 551-560.

Hunt, B. R., Qi, Y. Y. and DeKruger, D., 1991, Fuzzy Classification using Set Membership Functions in the Back Propagation Algorithm. In C. H. Dagli, S. R. T. Kumara and Y. C. Chin, Ed.s, *Intelligent Engineering Systems Through Artificial Neural Networks* (New York: ASME Press), 267-276.

Jakubowicz, Oleg and Ramanujam, Sridhar, January 1990, A Neural Network Model for Fault-diagnosis of Digital Circuits. *Proceedings of the International Joint Conference on Neural Networks*, II-611-614.

Khanna, Tarun, 1990, *Foundations of Neural Networks* (Reading, MA: Addison-Wesley).

Klir, George J. and Folger, Tina A., 1988, *Fuzzy Sets, Uncertainty, and Information* (Englewood Cliffs, NJ: Prentice-Hall).

Knapp, Gerald M. and Wang, Hsu-Pin, forthcoming, Acquiring, Storing, and Utilizing Process Planning Knowledge Using Neural Networks. *Journal of Intelligent Manufacturing*.

Khoshnevis, Behrokh and Park, Joo, 1988, Real Time Manufacturing Process Planning. *University of Southern California Working Paper*.

Kohonen, T., 1984, *Self-Organization and Associative Memory* (Berlin: Springer-Verlag).

Kong, Seong-Gon and Kosko, Bart, 1992, Adaptive Fuzzy Systems for Backing up a Truck-and-Trailer. *IEEE Transactions on Neural Networks*, 3, 211-223.

Kosko, Bart, 1992, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence* (Englewood Cliffs, NJ: Prentice-Hall).

Krishnan, Ganapathy and Walters, Deborah, 1988, Psychologically Plausible Features for Shape Recognition in a Neural-Network. *Proceedings of the International Joint Conference on Neural Networks*, II-127.

Kung, S. Y. and Hwang, J. N., 1988, An Algebraic Projection Analysis for Optimal Hidden Units Size and Learning Rates in Back-Propagation Learning. *Proceedings of the International Joint Conference on Neural Networks*, I-363-370.

Kusiak, Andrew, 1990, *Intelligent Manufacturing Systems* (Englewood Cliffs, NJ: Prentice Hall, Inc.).

Lapedes, Alan and Farber, Robert, 1988, How Neural Nets Work. In D. Z. Anderson, Ed., *Neural Information Processing Systems* (New York: American Institute of Physics), 442-456.

Lippmann, Richard P., 1987, An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, 4, 4-22.

Luby, S. C., Dixon, J. R. and Simmons, M. K., 1986, Creating and Using a Features Data Base. *Computers in Mechanical Engineering*, 5, 25.

McAndless, Elizabeth, Stacey, Deborah, Rueb, Kurt and Wong, Andrew K. C., 1991, A Hybrid Artificial Neural Network/Rule Based Approach to a Real Time Machine Vision System. In C. H. Dagli, S. R. T. Kumara and Y. C. Chin, Ed.s *Intelligent Engineering Systems Through Artificial Neural Networks* (New York: ASME Press), 909-914.

Movellan, Javier R., 1990, Error Functions to Improve Noise Resistance and Generalization in Backpropagation Networks. *Proceedings of the International Joint Conference on Neural Networks*, I-557-560.

Myers, Ware, 1990, Artificial Neural Networks are Coming. *IEEE Expert*, 5, 3-6.

Orlando, Jim, Mann, Richard and Haykin, Simon, 1990, Radar Classification of Sea-Ice Using Traditional and Neural Classifiers. *Proceedings of the International Joint Conference on Neural Networks*, II-263.

Perugini, N. K. and Engeler, W. E., 1989, Neural Network Learning Time: Effects of Network and Training Set Size. *Proceedings of the International Joint Conference on Neural Networks*, II-395-401.

Pratt, M. J., 1988, Synthesis of an Optimal Approach to Form Feature Modeling. *Proceedings of the ASME Conference on Computers in Engineering*, 263.

Roth, Michael W., 1990, Survey of Neural Network Technology for Automatic Target Recognition. *IEEE Transactions on Neural Networks*, 1, 28.

Rumelhart, David E., McClelland, James L. and the PDP Research Group, 1986, *Parallel Distributed Processing, Volume 1* (Cambridge, MA: The MIT Press).

Shah, J. J. and Rogers, M. T., 1988, Feature Based Modeling Shell: Design and Implementation. *Proceedings of the ASME Conference on Computers in Engineering*, 255.

Shea, Patrick M. and Lin, Vincent, 1989, Detection of Explosives in Checked Airline Baggage Using an Artificial Neural System. *Proceedings of the International Joint Conference on Neural Networks*, II-31-34.

Simpson, Patrick K., 1990, *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations* (New York: Pergamon Press).

Smith, Alice E. and Dagli, Cihan H., 1991, Relating Binary and Continuous Problem Entropy to Backpropagation Network Architecture. In *Applications of Artificial Neural Networks II*, Volume 2 (Bellingham, WA: SPIE), 551-562.

Smith, Alice E., Dagli, Cihan H. and Raterman, Elaine R., 1991, An Empirical Analysis of Backpropagation Error Surface Initiation for Injection Molding Process Control. *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1529-1534.

Specht, Donald F., 1990 a, Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification. *IEEE Transactions on Neural Networks*, 1, 111-121.

Specht, Donald F., 1990 b, Probabilistic Neural Networks. *Neural Networks*, 3, 109-118.

Suri, Rajan and Shimizu, Masami, 1989, Design for Analysis: A New Strategy to Improve the Design Process. *University of Wisconsin - Madison Technical Report*, No. 89-3.

Swift, K. G., 1987, *Knowledge Based Design for Manufacture* (Englewood Cliffs, NJ: Prentice-Hall).

Sykes, Edward A. and White, Chelsea C. III, 1991, Multiobjective Intelligent Computer-Aided Design. *IEEE Transactions on Systems, Man, and Cybernetics*, 21, 1498-1511.

Twomey, Janet M. and Smith, Alice E., 1992, An Examination of Performance Measures for Pattern Classification Backpropagation Neural Networks, to appear in *Proceedings of ANNIE 92*, (New York: ASME Press).

Vaghul, M., Dixon, J. R. and Sinsmeister, G. E., 1985, Expert Systems in a CAD Environment: Injection Molding Part Design as an Example. *Proceedings of the ASME Conference on Computers in Engineering*, 77.

Wang, Q., Zhou, J. and Yu, J., 1988, A Chain-Drive Design Expert System and CAD System. In D. T. Pham, Ed. *Expert Systems in Engineering* (Exeter, U.K.: Springer-Verlag).

Wasserman, Philip D., 1989, *Neural Computing: Theory and Practice* (New York: Van Nostrand Reinhold).

Werbos, Paul J., 1988, Backpropagation: Past and Future. *Proceedings of the International Joint Conference on Neural Networks*, I-343.

Zadeh, Lotfi, 1965, Fuzzy Sets. *Information and Control*, 8, 338-353.